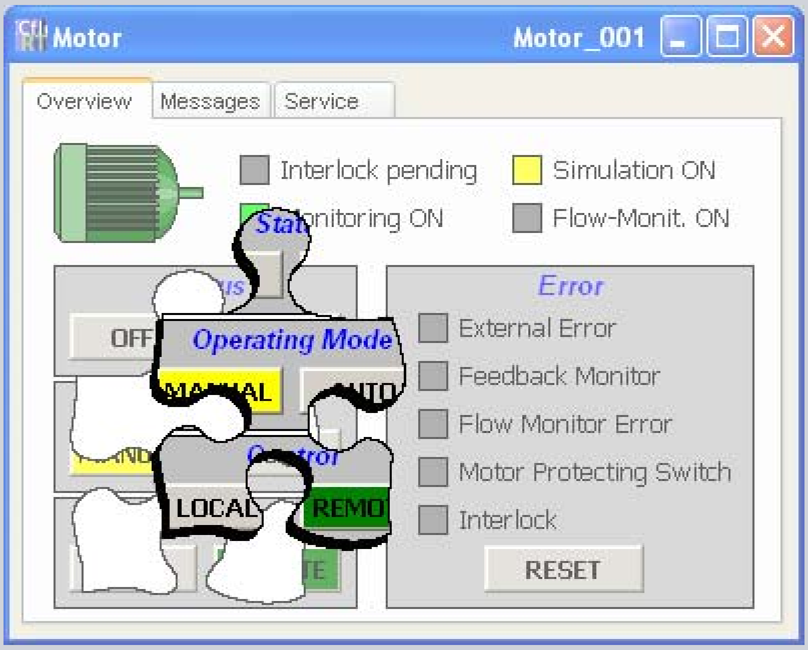
Sample Blocks for STEP 7 and WinCC flexible - Supplements

WinCC flexible

Application description • September 2010



Applications & Tools

Answers for industry.

SIEMENS

Industry Automation and Drives Technologies Service & Support Portal

This article is taken from the Service Portal of Siemens AG, Industry Automation and Drives Technologies. The following link takes you directly to the download page of this document.

http://support.automation.siemens.com/WW/view/en/36435784

SIEMENS

SIMATIC WCF_BLOCKS_Supplements

Application

Warranty and Liability

Note The application examples are not binding and do not claim to be complete regarding configuration, equipment and any contingencies. The application examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These application examples do not relieve you of the responsibility to use sound practices in application, installation, operation and maintenance. When using these application examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these application examples at any time without prior notice. If there are any deviations between the recommendations provided in this application example and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

It is not permissible to transfer or copy these application examples or excerpts of them without having prior authorization from Siemens Industry Sector in writing.

For questions about this document please use the following e-mail address:

online-support.automation@siemens.com

Table of Contents

Automation Task 1

Introduction

The sample blocks for STEP 7 and WinCC flexible are to offer the user various automation functions or the use of these blocks as templates for the configuration of individual blocks.

Description of the automation task

The task of this supplement documentation is to show the project engineers ways in which to use existing sample blocks in a real life situation.

This is to be achieved by:

- dynamic positioning of the faceplates at runtime.
- multiple use of a faceplate and therefore:
  - an economical way of using controller tags (“power tags”) and the resulting cost savings.
  - and a careful use of resources regarding the number of the objects used in the picture whilst considering the system limits.

2 Automation Solution

Introduction

An individual faceplate icon will be placed for each motor in the project. Via address multiplexing of the data block used, the faceplate icons address only one single faceplate window (See chapter 3.2 – Multiple use through address multiplexing).

Alternatively, the faceplate can assume an alternative position via mouse click on its header (See chapter 3.1 – Dynamic positioning at runtime).

Description of the automation solution

This document shows an example of the necessary configuration by means of the “MOTOR” faceplate to be able to expand a faceplate of this application by the following characteristics.

- Dynamic positioning at runtime

  Via mouse click in the header of the faceplate, it assumes an alternative position. Another mouse click shifts the faceplate to its original position.

  This makes it possible to operate and monitor the remaining picture, covered by an open faceplate, without having to close it and having to do without the provided information.

- Multiple use through address multiplexing

  By using address multiplexing, one single faceplate addresses several assigned objects without having to configure individual faceplates and tags for these objects.

ATTENTION	**Before using the block in your own projects, check the proper functioning of the block and adjust it to your individual requirements where necessary. The block described in this application is only intended as a template for creating your own blocks.**

2.1 Hardware and software components used

The application was generated with the following components:

Hardware components

Table 2-1

Component	Number	Note
Development system	1	PC to configure the controller and WinCC flexible. The hardware requirements for STEP 7 and WinCC flexible apply.
S7-300 CPU or S7-400 CPU	1	Alternatively, the controller can also be simulated with PLCSIM.

Software components

Table 2-2

Component	Number	MLFB / Order number	Note
STEP 7 V5.4 SP5	1	6ES7810-4CC08-0YA7	
WinCC flexible 2008 SP2	1	6AV6613-0AA51-3CA5	Incl. Update 1
S7-PLCSIM V5.4	1	6ES7841-0CC05-0YA5	[As an option]

Sample files and projects

The following list contains all files and projects that are used in this example.

Table 2-3

Component	Note
36435784_S7_WCF_Blocks_SUPPLEMENTS.zip	The zip file contains the STEP 7 project with the integrated WinCC flexible project.
36435784_S7_WCF_Blocks_DOCUMENTS_d.zip	All documents for this application.

3 Functional Mechanisms

Introduction

The functional mechanisms described in this chapter are intended as additional information to the “Sample Blocks for STEP 7 and WinCC “ flexible application.

This is why the two functional mechanisms are described separately from each other and can be used independently.

When using the dynamic positioning, for example, it is not necessary to also use address multiplexing.

3.1 Dynamic positioning at runtime

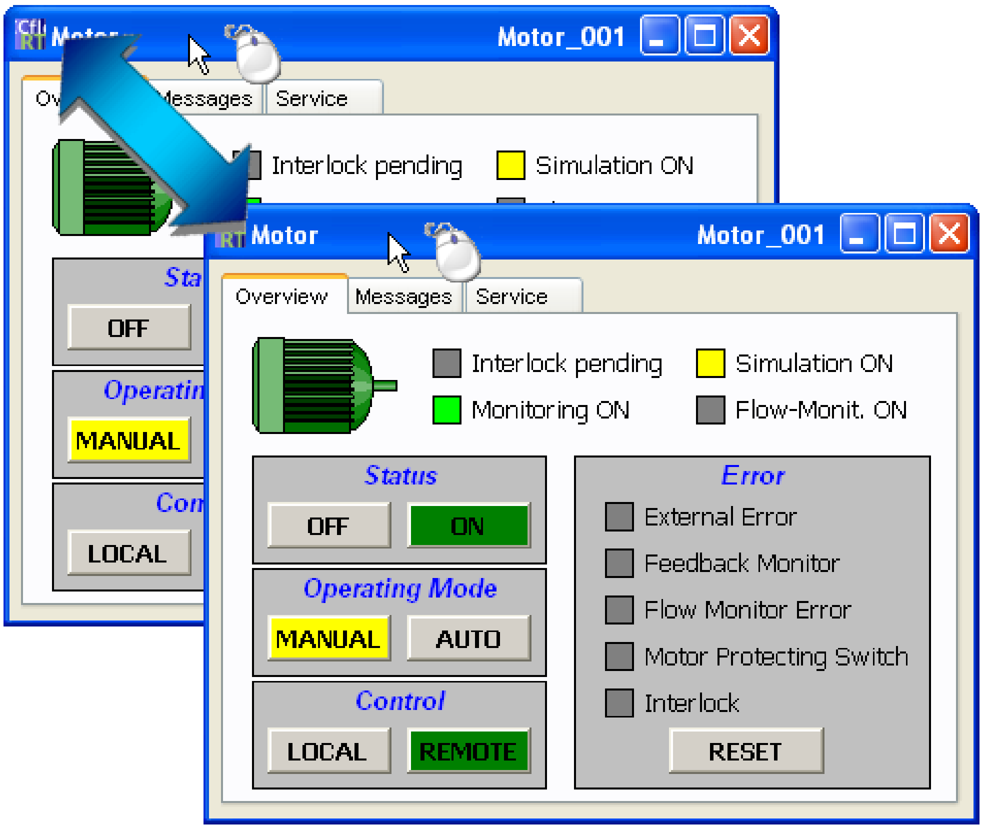
Introduction

It is frequently necessary to move the faceplate on the user interface to get to displays which are positioned behind it. The faceplate window which is still opened provides additional information or is necessary for the operation.

It is **not** possible to move the faceplate window within WinCC flexible as usual for the graphical user interface of an operating system.

Via mouse click, the faceplate assumes an alternative position on its header. Another mouse click shifts the faceplate to its original position.

Figure 3-1



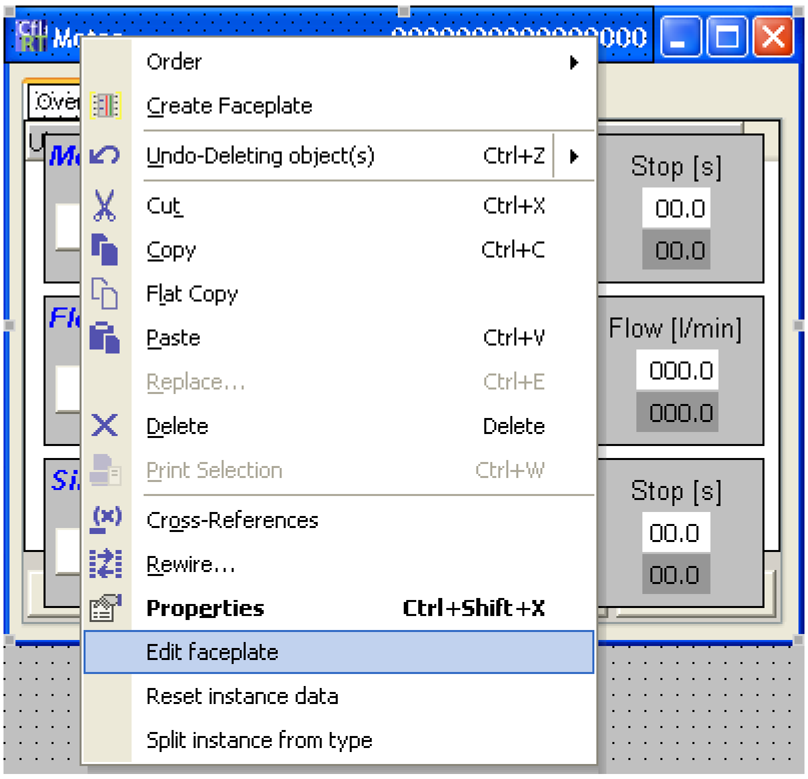
The following steps are necessary for implementation:

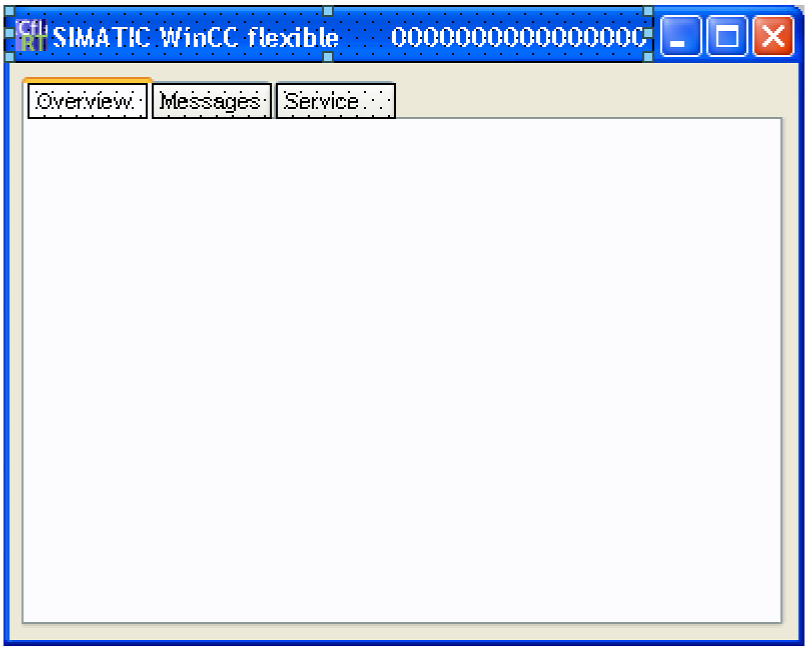
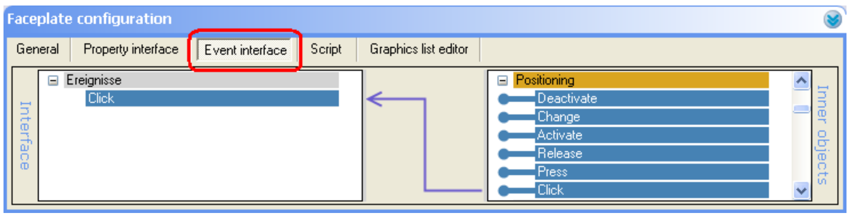
- modification of faceplate window
- creating and connecting of positioning tags
- creating and connecting of scripts to toggle the position

3.1.1 Modification of faceplate window

To reposition the faceplate window on its header via mouse click, it is necessary to place a button in this area.

Table 3-4

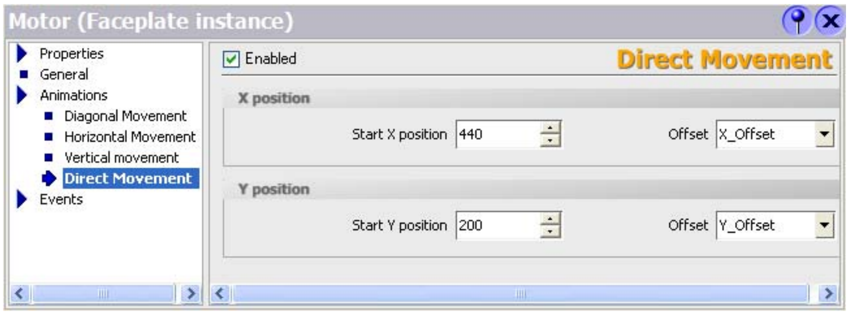
Step	Description
1.	**Edit faceplate** • Right-click on an existing instance of a faceplate or the corresponding template in the library. • Click “Edit faceplate”.  Order Create Faceplate Undo-Deleting object(s) Ctrl+Z Cut Ctrl+X Copy Ctrl+C Flat Copy Paste Ctrl+V Replace... Ctrl+E Delete Delete Print Selection Ctrl+W Cross-References Rewire... Properties Ctrl+Shift+X Edit faceplate Reset instance data Split instance from type Stop [s] 00.0 Flow [l/min] 000.0

Step	Description
2.	**Creating button** Place a button in “Level 0” of the faceplate editor in the header area of the faceplate window with the following characteristics: • General – Button mode: invisible • Properties – Appearance – Width focus:0 • Properties – Layout – Position X, position Y: 10, 10 • Properties – Layout – Size X, Size Y: 325, 29 • Properties – Misc – Name: Positioning • Properties – Misc – Layer: 0 • Animations – Visibility: Enabled, tags: Visibility, hidden  SIMATIC WinCC flexible 000000000000000C Overview Messages Service
3.	**Add click event in the interface** • Select the “Event interface” of the faceplate in “Faceplate configuration“. • Select the “Positioning” object in “Inner objects”. • Drag the “Click” event via Drag&Drop to the interface of the faceplate. • Close the faceplate editor.  Faceplate configuration General Property interface Event interface Script Graphics list editor Interface Ereignisse Click Positioning Deactivate Change Activate Release Press Click Inner objects

3.1.2 Creating and connecting positioning tags

To position the faceplate window, two positioning tags are connected via its property dialog.

Table 3-5

Step	Description
1.	**Creating positioning tags** Create two tags each for every instance of your faceplate window: Name: X_Offset, connection: Internal tag, data type: Int Name: Y_Offset, connection: Internal tag, data type: Int
2.	**Connecting positioning tags** • Right click on an existing instance of faceplate in a picture. • Select "Properties". • Connect the previously created tags to the faceplate: Animations – Direct Movement – enabled Animations – Direct Movement – X position - Offset: X_Offset Animations – Direct Movement – y position - Offset: Y_Offset  Motor (Faceplate instance) Properties / General / Animations / Diagonal Movement / Horizontal Movement / Vertical movement / Direct Movement / Events Direct Movement ☑ Enabled X position: Start X position 440, Offset X_Offset Y position: Start Y position 200, Offset Y_Offset

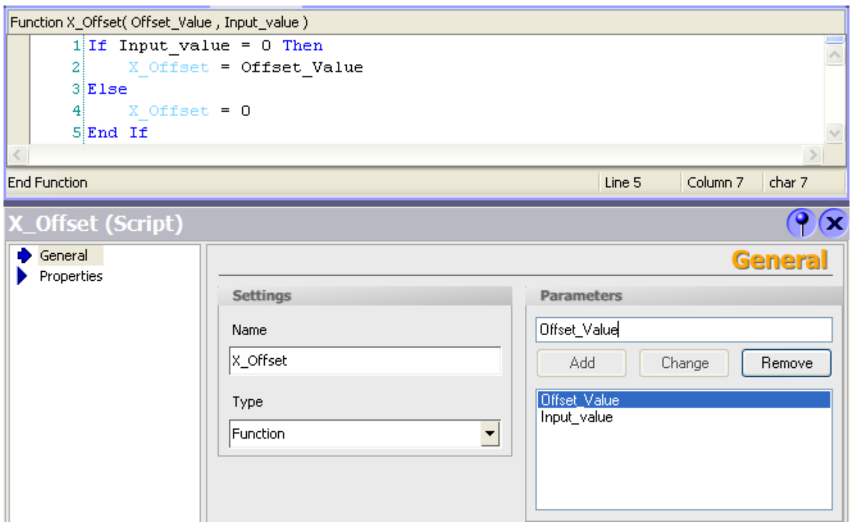
3.1.3 Creating and connecting scripts

Two scripts are created as interface for the click event of the faceplate and its positioning tags.

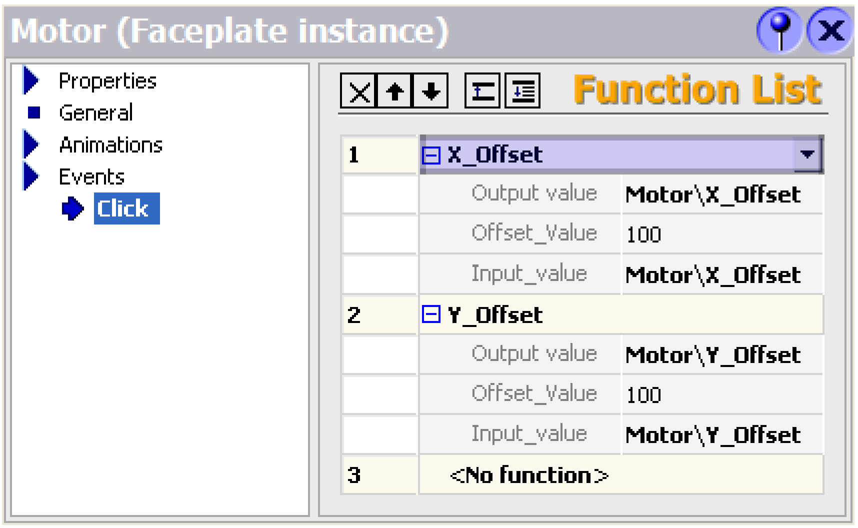
Through the parameter dialog, the scripts can be used as often as desired.

Table 3-6

Step	Description

1.	**Creating scripts**
	Create a script each with the following properties for both positioning tags (as an example explained for “X_Offset”):
	• General – Settings – Name: X_Offset
	• General – Settings – Type: Function
	• General – Parameter: Offset_Value, Input_value
	• Code:
	If Input_value = 0 Then
	    X_Offset = Offset_Value
	Else
	    X_Offset = 0
	End If
	
	Function X_Offset(Offset_Value , Input_value)
	`If Input_value = 0 Then`
	`    X_Offset = Offset_Value`
	`Else`
	`    X_Offset = 0`
	`End If`
	End Function Line 5 Column 7 char 7
	X_Offset (Script)
	General
	Settings: Name: X_Offset; Type: Function
	Parameters: Offset_Value; Input_value

2.	**Connecting scripts** • Right click on an existing instance of faceplate in a picture. • Select “Properties”. • Connect the previously created scripts to the faceplate: Events – Click – Function: X_Offset Events – Click – Function: Y_Offset • Connect the following tags to the parameter interface (as an example explained for “X_Offset”): Output value: X_Offset Offset_Value: <offset value> (may be a constant or a variable) Input_value: X_Offset

Step	Description
	Note: To be able to reuse the scripts, the same tag has to be connected for the “Output value” and “Input_value” parameters. 

Motor (Faceplate instance)

1	X_Offset	
	Output value	Motor\X_Offset
	Offset_Value	100
	Input_value	Motor\X_Offset
2	Y_Offset	
	Output value	Motor\Y_Offset
	Offset_Value	100
	Input_value	Motor\Y_Offset
3	<No function>	

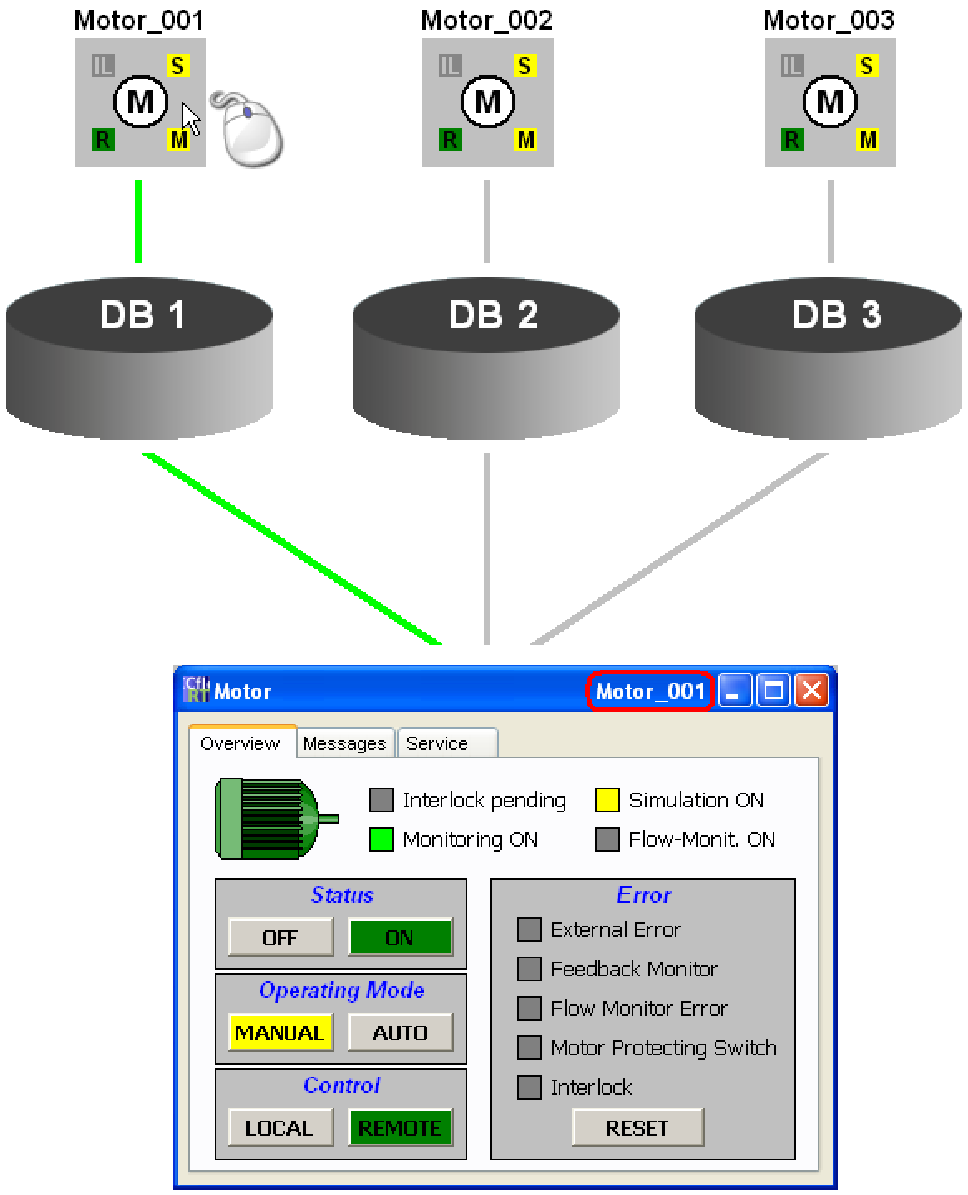
3.2 Multiple use through address multiplexing

Introduction

By using address multiplexing, one single faceplate addresses several assigned objects without having to configure individual faceplates and tags for each of these objects. In doing so, the value of a pointer is changed for the DB address of the faceplate tag.

The necessary pointer for the corresponding memory area is solely created in WinCC flexible and is set via mouse click on the faceplate icon – support by the controller is not necessary.

Figure 3-2

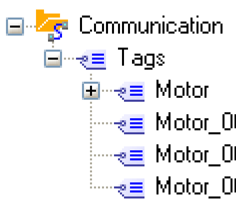
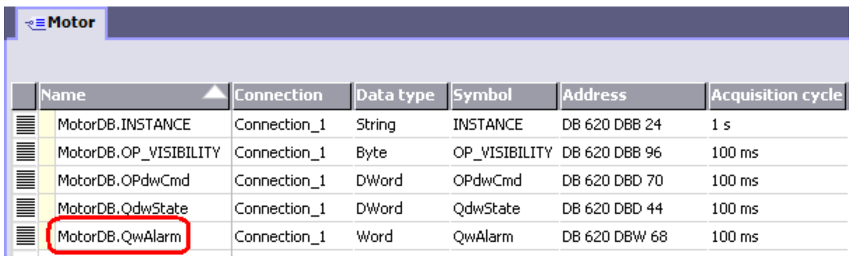


3.2.1 Editing tags

For reasons of clarity, this documentation describes the necessary configuration steps as an example by means of the “Motor” faceplate and three areas of use (“Motor_001”, “Motor_002” and “Motor_003”).

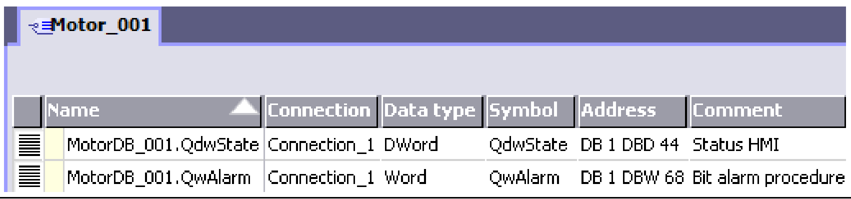
The concept can naturally also be transferred to other faceplates of this application and several locations.

Table 3-7

Step	Description
1.	**Creating a folder** Create an additional folder for each faceplate instance (location), e.g. “Motor_001”.  Communication Tags Motor Motor_001 Motor_002 Motor_003
2.	**Deleting of existing tags** Delete the “QwAlarm” tag in the “Motor” main folder. 

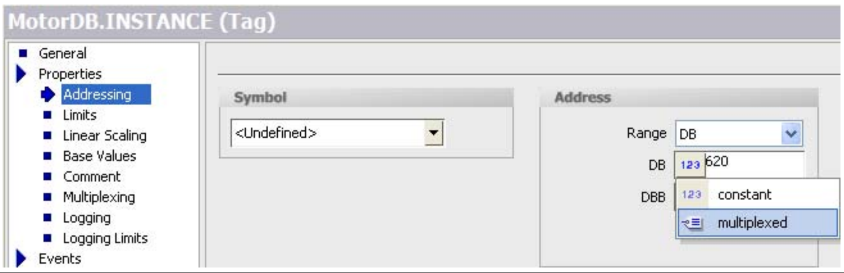
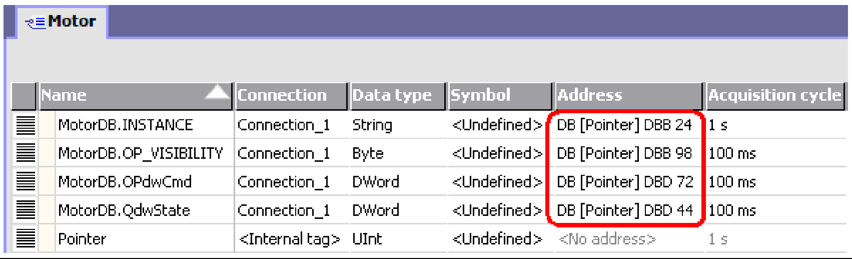
Motor

Name	Connection	Data type	Symbol	Address	Acquisition cycle
MotorDB.INSTANCE	Connection_1	String	INSTANCE	DB 620 DBB 24	1 s
MotorDB.OP_VISIBILITY	Connection_1	Byte	OP_VISIBILITY	DB 620 DBB 96	100 ms
MotorDB.OPdwCmd	Connection_1	DWord	OPdwCmd	DB 620 DBD 70	100 ms
MotorDB.QdwState	Connection_1	DWord	QdwState	DB 620 DBD 44	100 ms
MotorDB.QwAlarm	Connection_1	Word	QwAlarm	DB 620 DBW 68	100 ms

Step	Description
3.	**Creating new tags** • For each instance create the tags “QdwState” and “QwAlarm” in the respective instance folder, e.g. “Motor_001”. • Create an internal “Pointer” tag for each faceplate type of the “UInt” data type in the “Motor” main folder (see figure in step 5). **Note:** The “QdwState” tag is used to display the faceplate icons, the “QwAlarm” tag is used to display messages. Both tags always have to be updated, irrelevant of the respective pointer value. This is why they have to be connected by symbol and must **not** be addressed via address multiplexing. 

Motor_001

Name	Connection	Data type	Symbol	Address	Comment
MotorDB_001.QdwState	Connection_1	DWord	QdwState	DB 1 DBD 44	Status HMI
MotorDB_001.QwAlarm	Connection_1	Word	QwAlarm	DB 1 DBW 68	Bit alarm procedure

Step	Description
4.	**Multiplexing of existing tags** • Open the “Addressing” event dialog of the all the remaining tags in the “Motor” main folder. • Now right click the “123” button next to the DB number and change it from 620 to “Multiplex”. • Open the tag selection dialog on the DB field and select the “Pointer” tag. • Proceed like this with all tags of the “Service_FlowMonitoring”, “Service_Monitoring” and “Service_Simulation” sub folders.  MotorDB.INSTANCE (Tag) General Properties Addressing Limits Linear Scaling Base Values Comment Multiplexing Logging Logging Limits Events Symbol: <Undefined> Address Range: DB DB: 620 DBB constant multiplexed
5.	**Result** Make sure that all tags of the “Motor” main folder (with the exception of the “Pointer” tag) and its three sub folders were changed to the DB address “[Pointer]”. 

Motor

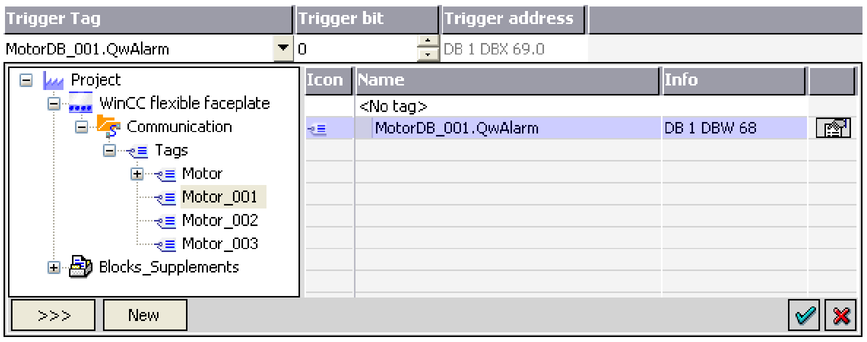
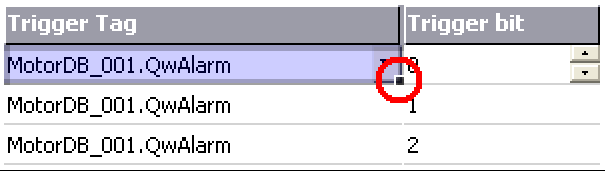
Name	Connection	Data type	Symbol	Address	Acquisition cycle
MotorDB.INSTANCE	Connection_1	String	<Undefined>	DB [Pointer] DBB 24	1 s
MotorDB.OP_VISIBILITY	Connection_1	Byte	<Undefined>	DB [Pointer] DBB 98	100 ms
MotorDB.OPdwCmd	Connection_1	DWord	<Undefined>	DB [Pointer] DBD 72	100 ms
MotorDB.QdwState	Connection_1	DWord	<Undefined>	DB [Pointer] DBD 44	100 ms
Pointer	<Internal tag>	UInt	<Undefined>	<No address>	1 s

3.2.2 Editing messages

Because the “QwAlarm” tag was deleted in the “Motor” main folder, the messages have lost their trigger tag. Since it is important to always receive the messages of all faceplates, it could not be changed to address multiplexing and was therefore superfluous.

Instead, the newly created message tags are now linked to the instance folders.

Table 3-8

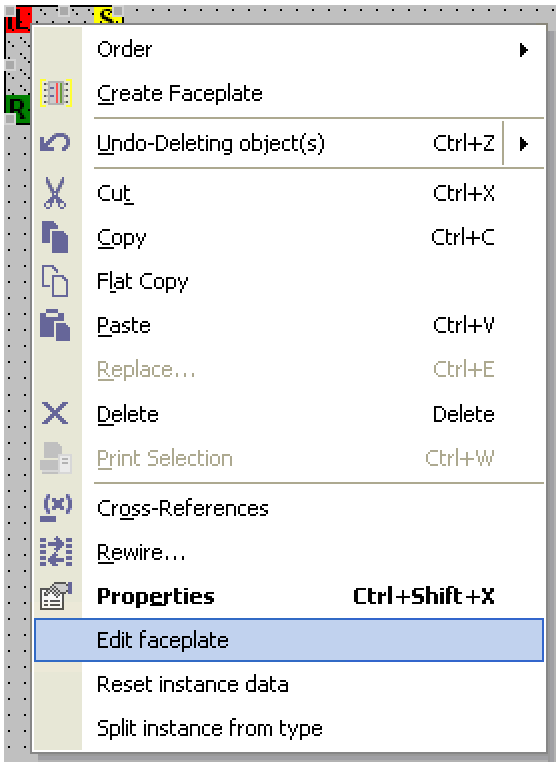
Step	Description
1.	Open the bit message editor under “Messages” > “Bit messages”.
2.	Open the tag selection dialog and select the “QwAlarm” tag corresponding to the messages from one of the instance folders.  Trigger Tag: MotorDB_001.QwAlarm · Trigger bit: 0 · Trigger address: DB 1 DBX 69.0 Project / WinCC flexible faceplate / Communication / Tags / Motor, Motor_001, Motor_002, Motor_003 / Blocks_Supplements Icon · Name · Info: <No tag>; MotorDB_001.QwAlarm · DB 1 DBW 68 >>> · New
3.	Copy the tag to all 16 messages of the same faceplate instance. • For this purpose, select the “Trigger Tag” field of the first message of a faceplate instance with bit number “0”, so that it is highlighted in blue. • Holding the mouse button down, on the right base point of the field, drag the trigger tag to all 16 messages of the same faceplate instance. The bit number is automatically incremented.  Trigger Tag · Trigger bit: MotorDB_001.QwAlarm · 0; MotorDB_001.QwAlarm · 1; MotorDB_001.QwAlarm · 2
4.	Repeat the first three steps for all faceplate instances.

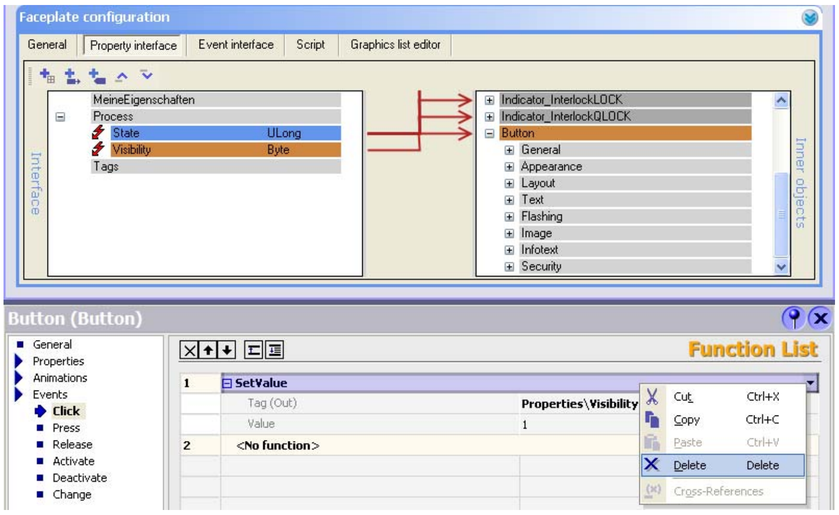
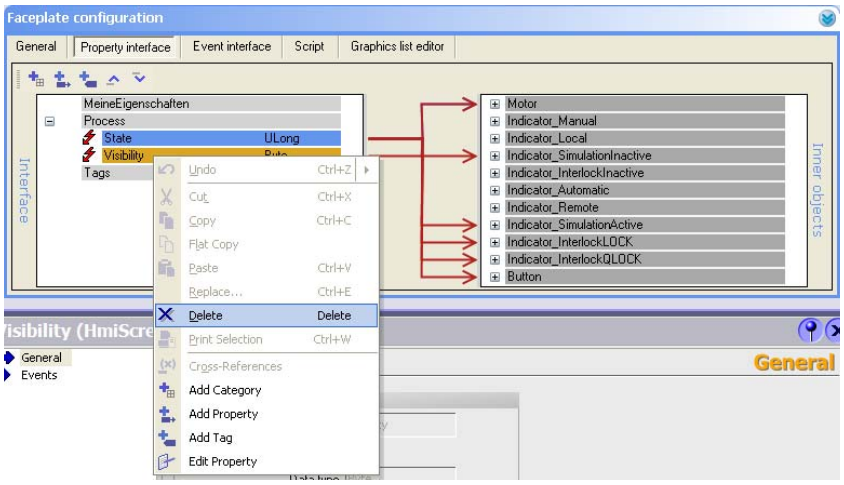
3.2.3 Modification of the faceplate icon

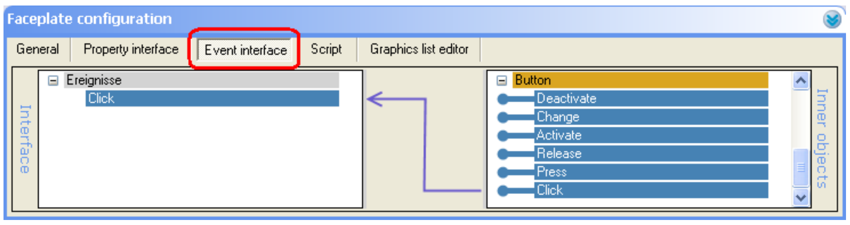
To be able to handle the pointer via faceplate icon and therefore the addressing of the faceplate icon, it is necessary to process several value assignments consecutively and in the correct sequence.

For this purpose, the “Click” event of the faceplate icon is moved outward to be able to connect the “SetValue” functions.

Table 3-9

Step	Description
1.	**Editing faceplate** • Right-click on an existing instance of a faceplate or the corresponding template in the library. • Click “Edit faceplate”.  Order Create Faceplate Undo-Deleting object(s) Ctrl+Z Cut Ctrl+X Copy Ctrl+C Flat Copy Paste Ctrl+V Replace... Ctrl+E Delete Delete Print Selection Ctrl+W Cross-References Rewire... **Properties Ctrl+Shift+X** Edit faceplate Reset instance data Split instance from type

Step	Description
2.	**Deleting the "SetValue" function** • In the event dialog in the properties of the faceplate configuration, open the "Button" button. • Select the "Click" event and delete the "SetValue" function. 
3.	**Deleting the "Visibility" property** • In the property interface of the faceplate configuration in the "Process" category, open the "Visibility" property of the property dialog with the right mouse button. • Delete the "Visibility" property. 

Step	Description
4.	**Add click event in the interface** • Select the "Event interface" of the faceplate in "Faceplate configuration". • Select the "Button" object in "Inner objects". • Drag the "Click" event via Drag&Drop to the interface of the faceplate. • Close the faceplate editor.  Faceplate configuration General Property interface Event interface Script Graphics list editor Interface Ereignisse Click Button Deactivate Change Activate Release Press Click Inner objects

3.2.4 Using the faceplate icon

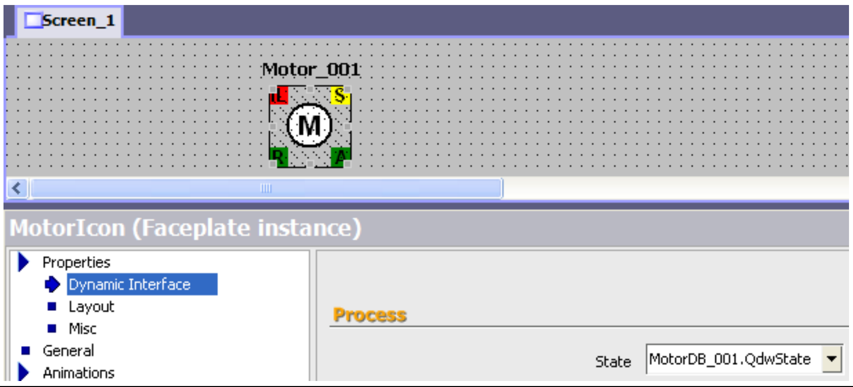
Since the faceplate icons always have to display the status of the corresponding motor, irrespective of the pointer (just like the messages), it is necessary to link the status tags of the instance folders.

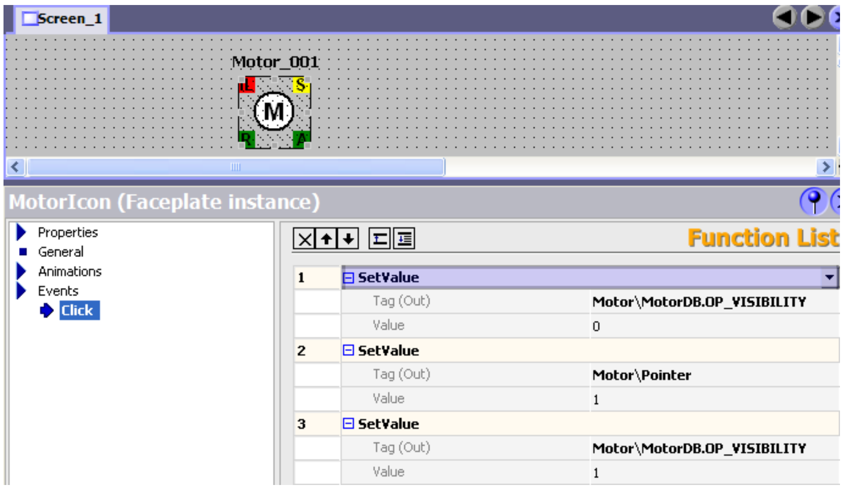
Furthermore, the faceplate icon has to handle the pointer value and therefore also has to readdress the faceplate instances. It is therefore necessary to configure an individual faceplate icon for each instance.

Note

It is always possible to change between the individual faceplate instances. When the faceplate window of “Motor_001” is open, the value of “Motor_002” can be instantly displayed when clicking on its faceplate icon.

Table 3-10

Step	Description
1.	**Selecting faceplate icon** Open the “Dynamic Interface” in the property dialog of the relevant faceplate icon.
2.	**Exchanging the status tags** Open the tag selection dialog in the field of the “State” property and select the “QdwState” tag relevant for the instance from one of the instance folders.  Screen_1 Motor_001 MotorIcon (Faceplate instance) Properties Dynamic Interface Layout Misc General Animations Process State MotorDB_001.QdwState

Step	Description
3.	**Creating the "SetValue" functions** In the "Click" event, assign the "SetValue" function for the following tags in the "Motor" main folder: • "OP_VISIBILITY" with the value "0" This marks the value of the previous instance of the faceplate window as "closed". • "Pointer" with the value "1" This addresses the new faceplate instance correctly. **The value depends on the respective instance or DB address!** • "OP_VISIBILITY" with the value "1" This opens the new faceplate instance with the first tab.  MotorIcon (Faceplate instance) Function List 1 SetValue — Tag (Out): Motor\MotorDB.OP_VISIBILITY — Value: 0 2 SetValue — Tag (Out): Motor\Pointer — Value: 1 3 SetValue — Tag (Out): Motor\MotorDB.OP_VISIBILITY — Value: 1

3.2.5 “VISIBILITY” - status in the controller (optional)

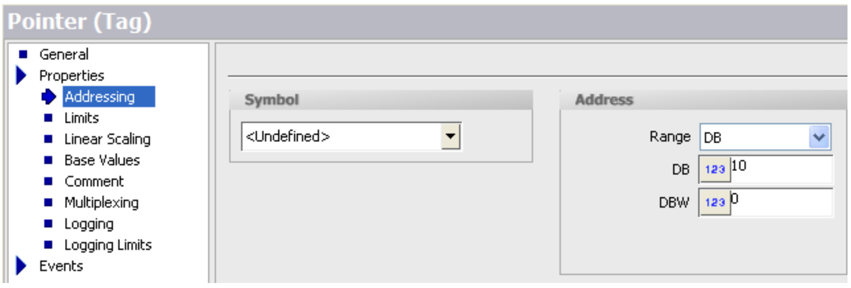
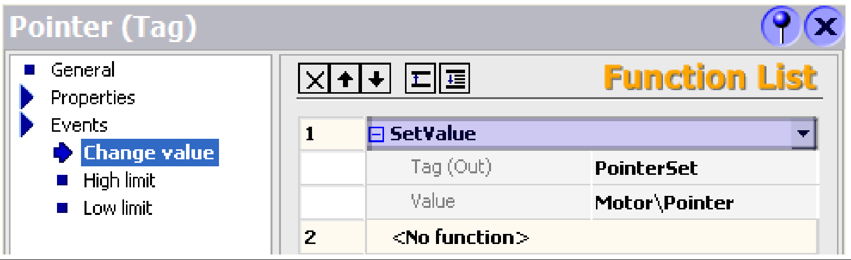
Since only one faceplate window is opened when address multiplexing, the “VISIBILITY” value always only refers to the respective instance.

This is why it is important to find out the pointer value before reading and/or writing the “VISIBILITY” value in the controller and to address the respective DB.

For reading, it is sufficient in WinCC flexible to declare the internal “Pointer” tag of the “UInt” data type in the “Motor” main folder as controller tag of the “Int” data type.

The value of the pointer indicates the data block number from which the “VISIBILITY” value can be read. However, if you would like to influence the faceplate window and its tab from the controller, additional steps are necessary.

Table 3-11

Step	Description
1.	**Declaring “Pointer” tag as controller tag** • Open the “Addressing” event dialog of the “Pointer” tag in the “Motor” main folder. • Assign a fixed address or select a symbol via the tag selection dialog.  Pointer (Tag) Symbol: <Undefined> Address: Range DB, DB 10, DBW 0
2.	**Declaring “PointerSet” tag as controller tag** • Create a new “PointerSet” tag in the “Motor” main folder. • Assign a fixed address or select a symbol via the tag selection dialog.
3.	**Configuring value transfer** To confirm the value transfer of the operator panel when changing the pointer value, the changed value has to be written back to the controller from the operator panel. • Open the “Addressing” event dialog of the “Pointer” tag in the “Motor” main folder. • Configure the “SetValue” function on the “Change value” event. Tag: “PointerSet” Value: “Pointer” (the tag)  Pointer (Tag) – Function List 1 SetValue Tag (Out): PointerSet Value: Motor\Pointer 2 <No function>

Step	Description
4.	**Value diagnostics in the controller** When the value of the “Pointer” tag and the value of the “PointerSet” tag are identical, then the pointer was accepted by the operator panel. Now the display of the faceplate window can be influenced (described) via the “VISIBILITY” block input of the respective data block.

4 Literature

This list is by no means complete and only presents a selection of suitable literature.

Table 4-12

	Subject	Title
\1\	Reference to the entry	http://support.automation.siemens.com/WW/view/en/36435784
\2\	Siemens I IA/DT Customer Support	http://support.automation.siemens.com
\3\	FAQ regarding address multiplexing	http://support.automation.siemens.com/WW/view/en/21808320

5 History

Table 5-13 History

Version	Date	Modification
V1.0	01.09.2010	First issue