

BECKHOFF New Automation Technology

Manual | EN

TF8040

TwinCAT 3 | Building Automation

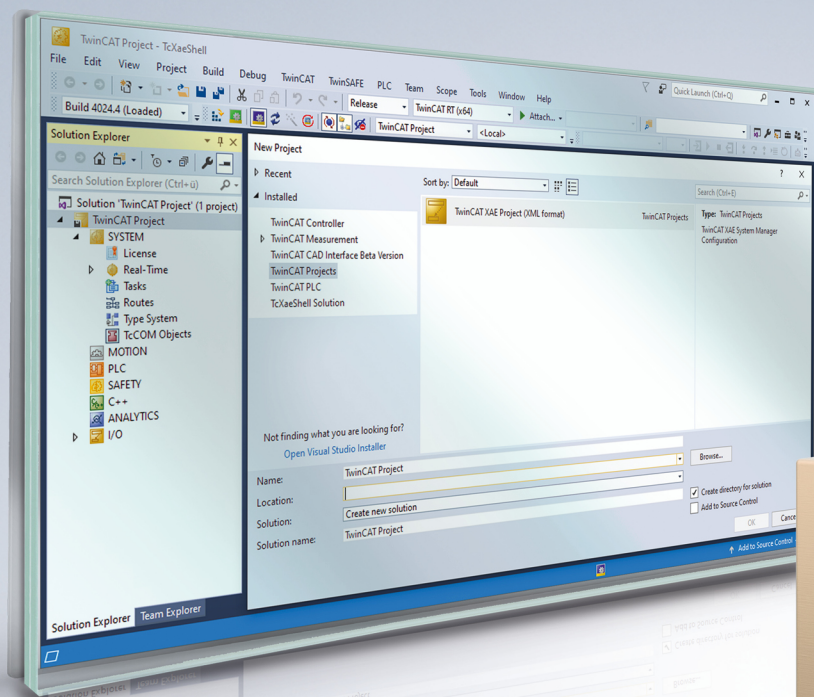


Table of contents

1 Foreword	5
1.1 Notes on the documentation	5
1.2 For your safety	6
1.3 Notes on information security	7
2 Introduction	8
2.1 Target groups	8
2.2 Requirement profile	8
2.3 Hardware requirements	8
2.4 System requirements	8
2.5 Installation	9
2.6 Licensing	12
3 Concepts	15
3.1 PLC	15
3.1.1 User roles	15
3.1.2 Control and regulation mechanisms	15
3.1.3 Objects	28
3.2 HMI	38
3.2.1 BaInterface	38
3.2.2 BaTemplate	39
3.2.3 Feedback	43
3.2.4 Generic	44
3.2.5 Trending	45
4 Tutorials	50
4.1 PLC	50
4.1.1 Starting a project	50
4.2 HMI	56
4.2.1 Starting a project	56
4.2.2 Generic HMI	63
5 Examples	74
5.1 Concept examples	75
5.1.1 PLC	75
5.1.2 HMI	82
5.2 Template samples	88
5.2.1 PLC	88
5.2.2 HMI	89
6 Programming	91
6.1 PLC	91
6.1.1 General	91
6.1.2 Libraries	92
6.1.3 PLC project templates	606
6.1.4 Templates	609
6.2 HMI	821
6.2.1 TcHmiBa	821

7	Tools	1024
7.1	Site Explorer	1024
7.2	Symbol Explorer	1042
7.2.1	Definitions	1042
7.2.2	Introduction	1042
7.2.3	User interface	1042
7.2.4	Find and replace	1052
7.2.5	Getting started	1057
7.2.6	Extensions	1063
7.2.7	Examples of regular expressions	1066
7.3	Template Repository	1066
8	Appendix	1073
8.1	Third-party components	1073
8.2	Support and Service	1073

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702

with corresponding applications or registrations in various other countries.



EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.

Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

DANGER

Hazard with high risk of death or serious injury.

WARNING

Hazard with medium risk of death or serious injury.

CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product



This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our <https://www.beckhoff.com/secguide>.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at <https://www.beckhoff.com/secinfo>.

2 Introduction

By using the function TwinCAT3 Building Automation, all PLC programs, including the central heating plant, the air conditioning plant and the room automation functions can be programmed with TwinCAT PLC Control and are then available as function blocks within the building automation libraries.

The PID controllers, the sequence controllers and the sequence linkers required for the TwinCAT 3 Building Automation library (Tc3_BA) can be found in the pre-installed library Tc3_BA_Common.

The functions and controllers required for the TwinCAT 3 Building Automation library (Tc3_BA2) can be found in the pre-installed library Tc3_BA2_Common.

2.1 Target groups

This software is intended for building automation system partners of Beckhoff Automation GmbH & Co. KG. The system partners operate in the field of building automation and are concerned with the installation, commissioning, expansion, maintenance and service of measurement, control and regulating systems for the technical equipment of buildings.

2.2 Requirement profile

The user requires basic knowledge of the following.

- TwinCAT 3
- Programming knowledge in IEC61131-3
- PC and network knowledge
- Structure and properties of the Beckhoff Embedded PC and its Bus Terminal system
- Knowledge of heating, ventilation, air conditioning and sanitary systems as well as room automation
- Relevant safety regulations for building technical equipment

2.3 Hardware requirements

The software can be used on all PC based hardware platforms, except on devices with the TwinCAT/RTOS operating system.

2.4 System requirements

PLC libraries

Technical data	Requirement
Operating system	Windows 7/10, Windows Embedded Standard 7, Windows CE7
Target platform	PC architecture (x86, x64 and ARM)
TwinCAT version	TwinCAT 3.1 Build 4024.35 or higher
Required TwinCAT setup level	TwinCAT 3 XAE, XAR
Required TwinCAT license	TF8040 TwinCAT 3 Building Automation, TF8020 TwinCAT 3 BACnet

Tools

The tools included in TF8040 run on **Windows 10** systems.

SiteExplorer still requires the .NET Desktop Runtime > v6.0.

TcHmiBa

The included TcHmiBa - packages must be divided into two areas.

- The packages that are used on the HMI client
 - Beckhoff.TwinCAT.HMI.BA.Icons
 - Beckhoff.TwinCAT.HMI.BA.Framework
 - Beckhoff.TwinCAT.HMI.BA.Controlsare used in a web browser. Currently, the **Chrome engine** is supported here.
When using web browsers with a different engine, proper functioning is not guaranteed.
- The package **Beckhoff.TwinCAT.HMI.BA.BaSite**, which is used in the HMI server as a server extension, runs under **Windows 10** and also requires the .NET Desktop Runtime > v6.0.

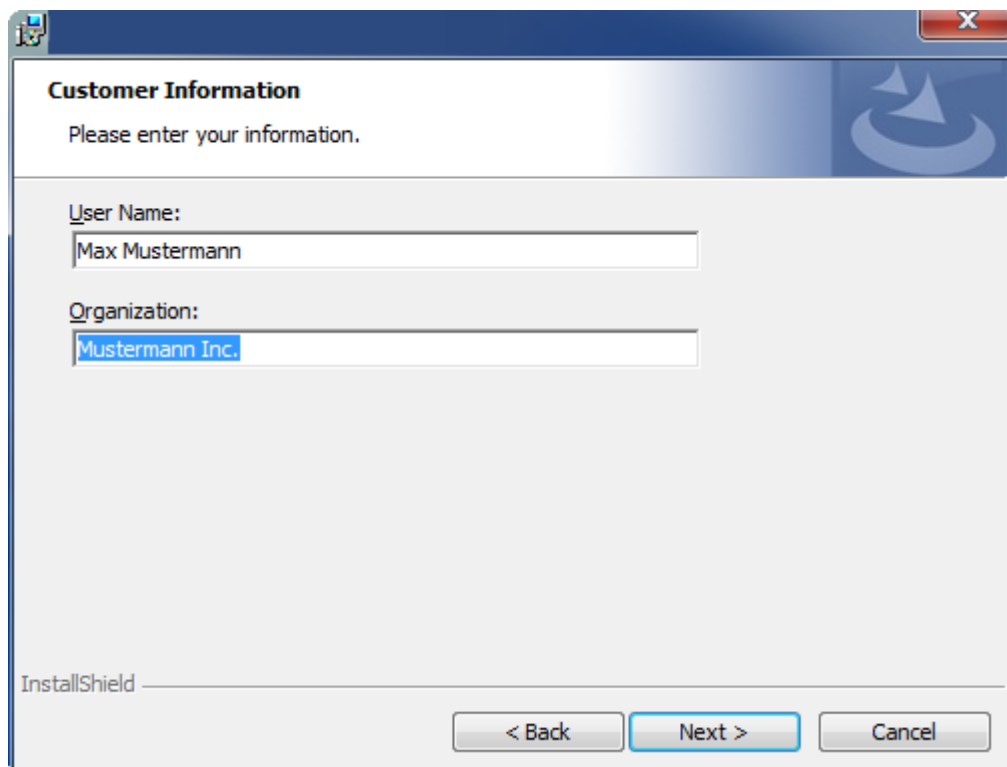
2.5 Installation

The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

- ✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.
1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.
 - ⇒ The installation dialog opens.
 2. Accept the end user licensing agreement and click **Next**.

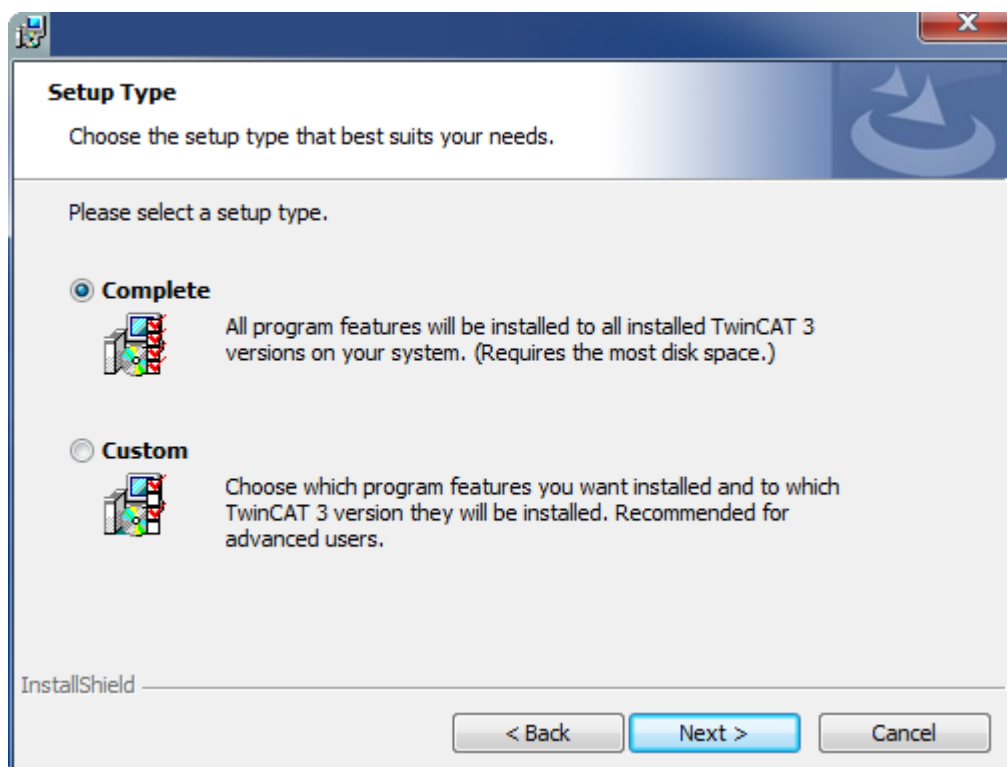


3. Enter your user data.



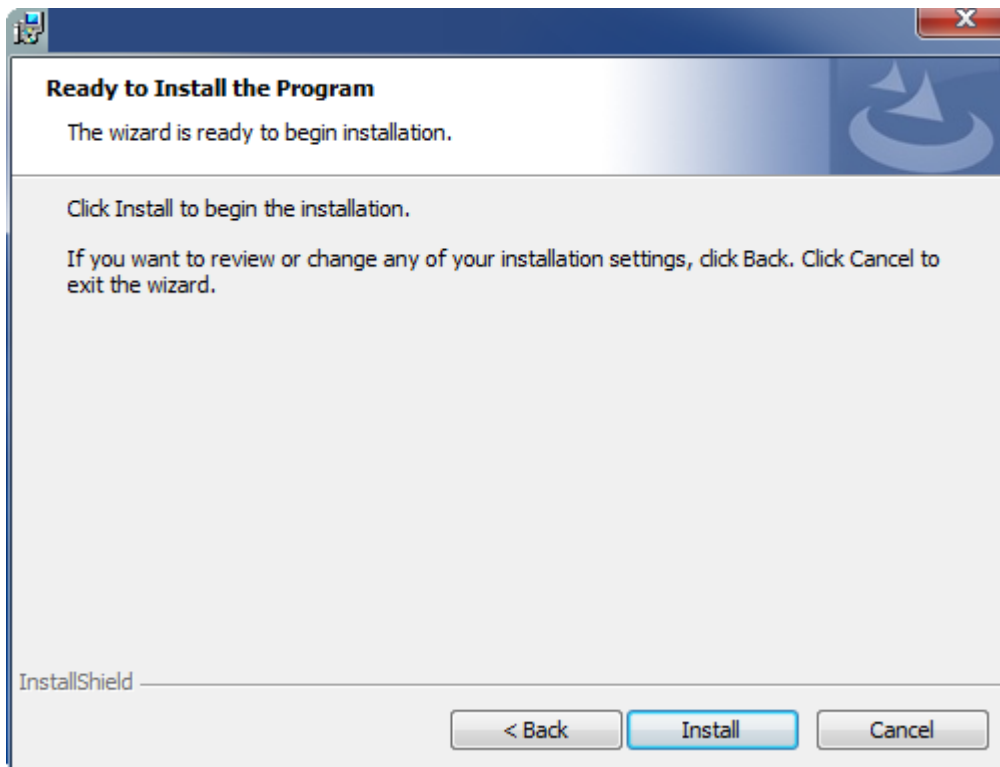
The 'Customer Information' dialog box is shown. It has a title bar with a standard Windows icon and a close button. The main area has a blue header with the title 'Customer Information' and a sub-header 'Please enter your information.' Below this, there are two text input fields. The first is labeled 'User Name:' and contains the text 'Max Mustermann'. The second is labeled 'Organization:' and contains the text 'Mustermann Inc.'. At the bottom, there is a progress bar labeled 'InstallShield' and three buttons: '< Back', 'Next >', and 'Cancel'.

4. If you want to install the full version of the TwinCAT 3 Function, select **Complete** as installation type. If you want to install the TwinCAT 3 Function components separately, select **Custom**.



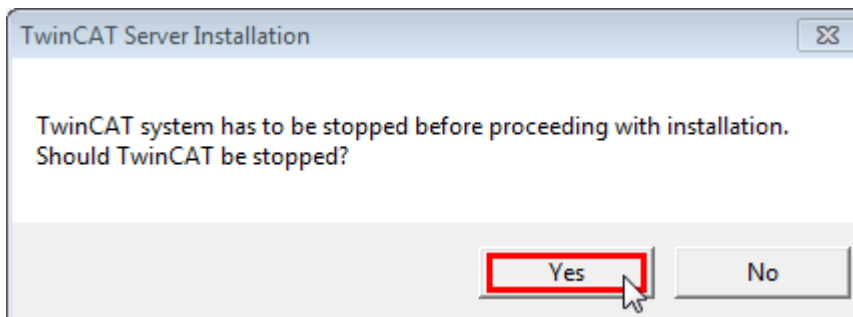
The 'Setup Type' dialog box is shown. It has a title bar with a standard Windows icon and a close button. The main area has a blue header with the title 'Setup Type' and a sub-header 'Choose the setup type that best suits your needs.' Below this, there is a text label 'Please select a setup type.' and two radio button options. The first option is 'Complete', which is selected, and has a description: 'All program features will be installed to all installed TwinCAT 3 versions on your system. (Requires the most disk space.)'. The second option is 'Custom', which is not selected, and has a description: 'Choose which program features you want installed and to which TwinCAT 3 version they will be installed. Recommended for advanced users.' At the bottom, there is a progress bar labeled 'InstallShield' and three buttons: '< Back', 'Next >', and 'Cancel'.

5. Select **Next**, then **Install** to start the installation.

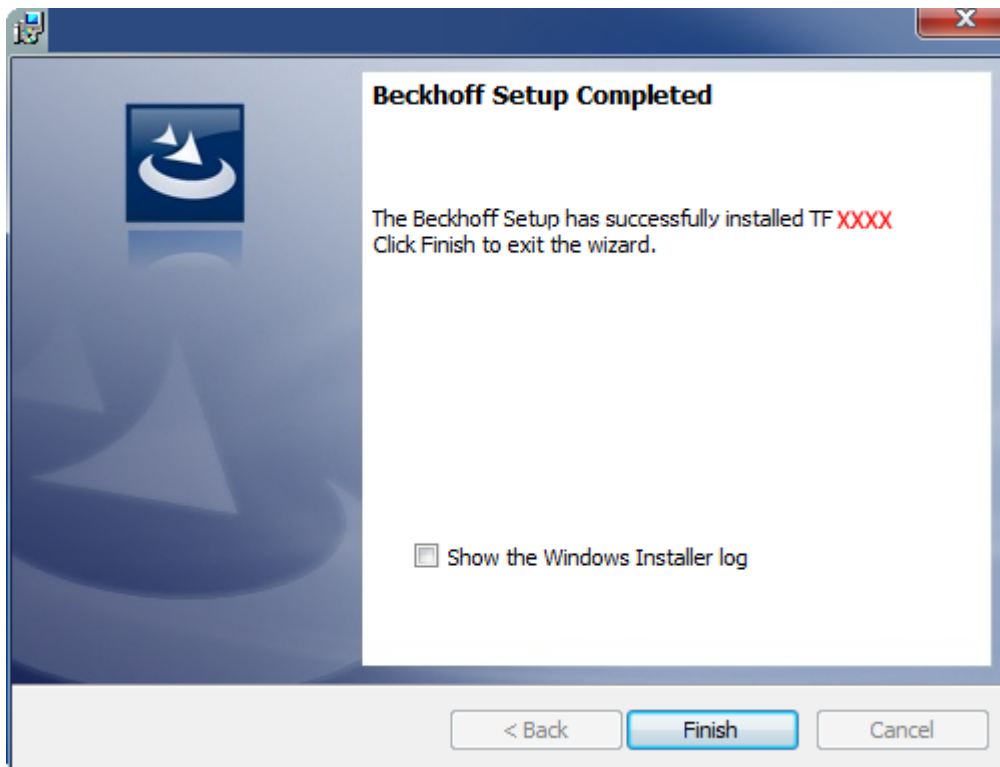


⇒ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

6. Confirm the dialog with **Yes**.



7. Select **Finish** to exit the setup.



⇒ The TwinCAT 3 Function has been successfully installed and can be licensed (see [Licensing](#) [► 12]).

2.6 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

Licensing the full version of a TwinCAT 3 Function

A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "[TwinCAT 3 Licensing](#)".

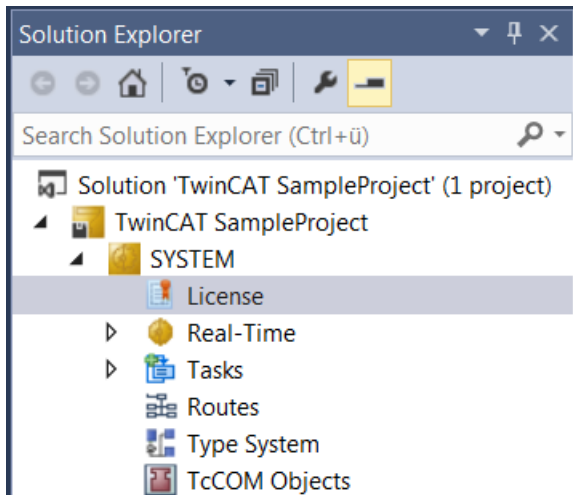
Licensing the 7-day test version of a TwinCAT 3 Function



A 7-day test version cannot be enabled for a [TwinCAT 3 license dongle](#).

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
 - ⇒ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



⇒ The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").

Order Information (Runtime) **Manage Licenses** Project Licenses Online Licenses

☐ Disable automatic detection of required licenses for project

Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	☐ cpu license
TF3650	TC3 Power Monitoring	☐ cpu license
TF3680	TC3 Filter	☐ cpu license
TF3800	TC3 Machine Learning Inference Engine	☐ cpu license
TF3810	TC3 Neural Network Inference Engine	☐ cpu license
TF3900	TC3 Solar-Position-Algorithm	☐ cpu license
TF4100	TC3 Controller Toolbox	☒ cpu license
TF4110	TC3 Temperature-Controller	☐ cpu license
TF4500	TC3 Speech	☐ cpu license

6. Open the **Order Information (Runtime)** tab.

⇒ In the tabular overview of licenses, the previously selected license is displayed with the status "missing".

7. Click **7-Day Trial License...** to activate the 7-day trial license.

Order Information (Runtime) | Manage Licenses | Project Licenses | Online Licenses

License Device: Target (Hardware Id) [Add...]

System Id: 2DB25408-B4CD-81DF-5488-6A3D9B49EF19 | Platform: other (91)

License Request

Provider: Beckhoff Automation [Generate File...]

License Id: | Customer Id: |

Comment: |

License Activation

7 Days Trial License... | License Response File...

- ⇒ A dialog box opens, prompting you to enter the security code displayed in the dialog.

Enter Security Code [X]

Please type the following 5 characters:

Kg8T4

[]

OK | Cancel

8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
- ⇒ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.
- ⇒ The 7-day trial version is enabled.

3 Concepts

3.1 PLC

3.1.1 User roles

The basic framework of the PLC does not know any users, but only roles. Therefore, the further description deals with the meaning of the roles.

An application (e.g. [TcHmiBa \[► 821\]](#)) defines users and then assigns them a corresponding role. According to the role of a logged-in user, functions are made available in the application or not.

Roles

Different access rights are provided for different users:

Role	Description
Guest	Lowest access permissions. Users cannot change parameters [► 32] and can only read current values that are in the Tc3 XBA [► 92] under VAR_INPUT CONSTANT PERSISTENT. Recommended for standard accesses without user login (e.g. generally accessible control panels).
Basic	Restricted access permissions. Users can view rudimentary parameters and hardly change any values. Recommended for operators with little knowledge of the system.
Advanced	Extended access rights. Users have insight into various parameters and e.g. authorization to change setpoints or timer programs. Recommended for operators with basic plant knowledge and instruction to supervise these plants.
Expert	Full access rights. Recommended for commissioning and for service personnel, as more in-depth interventions (e.g. adjustment of controller parameters) are also possible.
Internal	For Beckhoff support only.



The user's access area is evaluated at different points in the application to enable or hide certain functions.

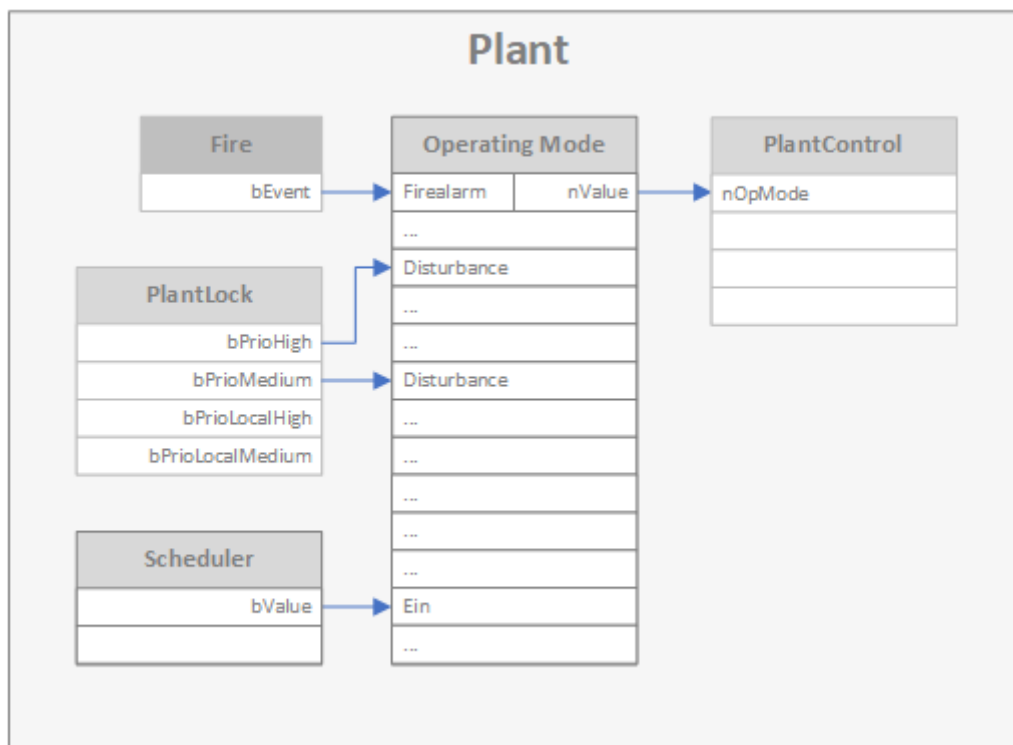
3.1.2 Control and regulation mechanisms

3.1.2.1 Plant control

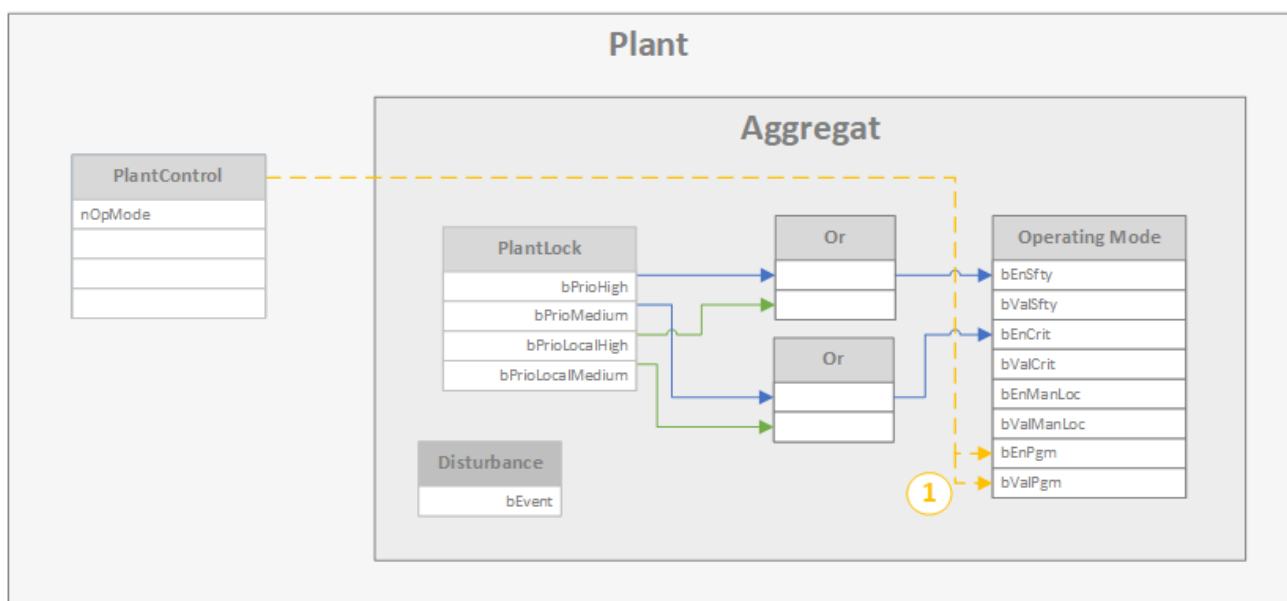
The purpose of a plant control is to command individual aggregates depending on certain plant operation modes.

Use case

The plant operation mode is determined in the plant template and passed on to the **PlantControl**:



The **PlantControl** switches all commanded aggregates into the respective defined states or operation modes:



Commanding an aggregate on priority **Program control**

Which aggregate is commanded with which operation mode is precisely defined in a control matrix.

Plant-operating-modes					
Operating-mode:	Disturbance	Fire alarm	Off	On (Slow)	On (Fast)
Priority:	Critical	Life safety	Program	Program	Program
Aggregate-operating-modes (assigned to each plant-operating-mode)					
Aggregate	Smoke extraction <i>no conditions no delays</i>	Off	On	Off	Off
	Gate <i>Waits for events no delays</i>	Closed	Open	Open	Open
	Motor <i>no conditions no delays</i>	Off	Off	Step 1	Step 2
	Conveyor 1 <i>no Conditions Delay switch forward</i>	Off	Off	On	On
	Conveyor 2 <i>no Conditions Delay switch backward</i>	Off	Off	Off	On

Example: Control matrix of 5 plant operation modes and 5 aggregates

Plant operation modes

- Are defined from left to right.

- **Initial state**

The first plant operation mode, which is commanded with the priority program control, is automatically recognized as initial state (system fault-free and switched off).



This state should be the *idle state* of an aggregate.

- **Switched-on state**

A plant operation mode is evaluated as *switched-on* if at least one assigned aggregate has an operation mode that is greater in the respective plant operation mode than in the initial state of the plant.

To create a **defined state**, the system is switched to the initial state once (when the PLC is started).

During this initialization phase:

- All aggregates are commanded accordingly.
- No conditions and delays are considered.

Aggregates

- Are defined from top to bottom.

- **Aggregate operation modes**

are each assigned to a plant operation mode.

- If an aggregate does not change its operation mode when switching the plant operation mode, no conditions and delays are taken into account.

Example:

When switching the plant operation mode from *On (Slow)* to *On (Fast)*:
The aggregate *Conveyor 1* is switched from *On* to *On*.

- If a plant operation mode is **skipped** when switching forward, delays from all skipped aggregate operation modes are also executed.

Example:

Change plant operation mode from *Off* to *On (Fast)*.

- **Conditions**

- **Wait for processes**

As long as an aggregate reports one or more active processes, it waits before switching to the next aggregate.

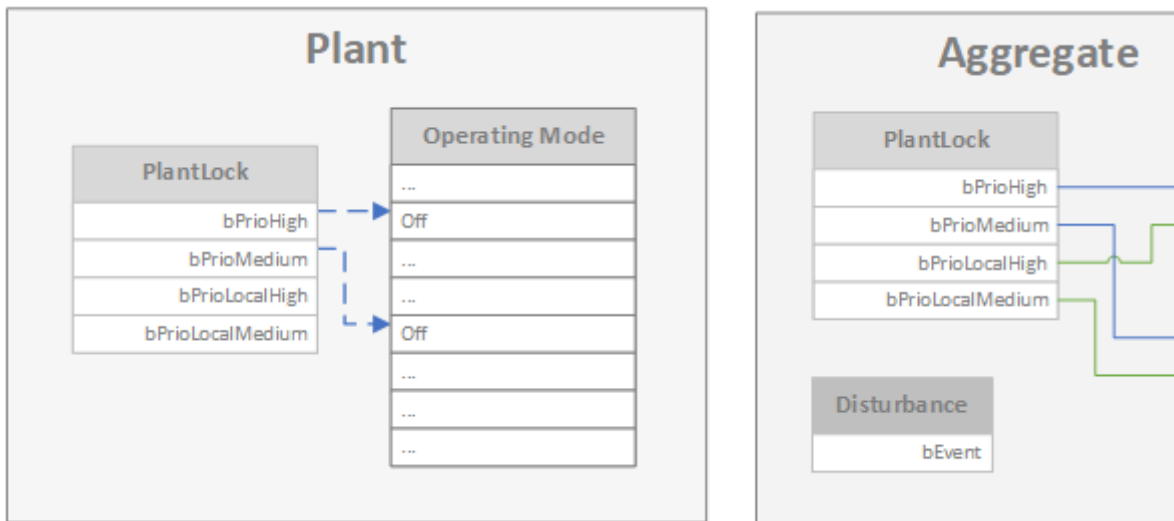
The switching is continued only after completion of all processes.

- **Wait for events**

As long as an aggregate reports active events, the system waits before switching to the next aggregate.



Since active events of the lock priorities Local Medium and Local High only switch off locally, i.e. only affect aggregates and not plants, the plant control would wait with the switching forward until the affected events expire.



Example: Effect of local lock priorities only on aggregates.



Conditions only affect the *plant control* when switching forward.

As soon as the control of an aggregate is completed, *conditions* have no influence (anymore) on the *plant control*.

- **Delays**

- **Delay step up**

Switching to the next aggregate takes place after an adjustable time has elapsed.

- **Delay step down**

Switching back to the previous aggregate takes place after an adjustable time has elapsed.

Control

The *control* or the *command* takes place in two directions.

During the control, conditions and delays (according to the configuration) of the respective aggregates are evaluated.



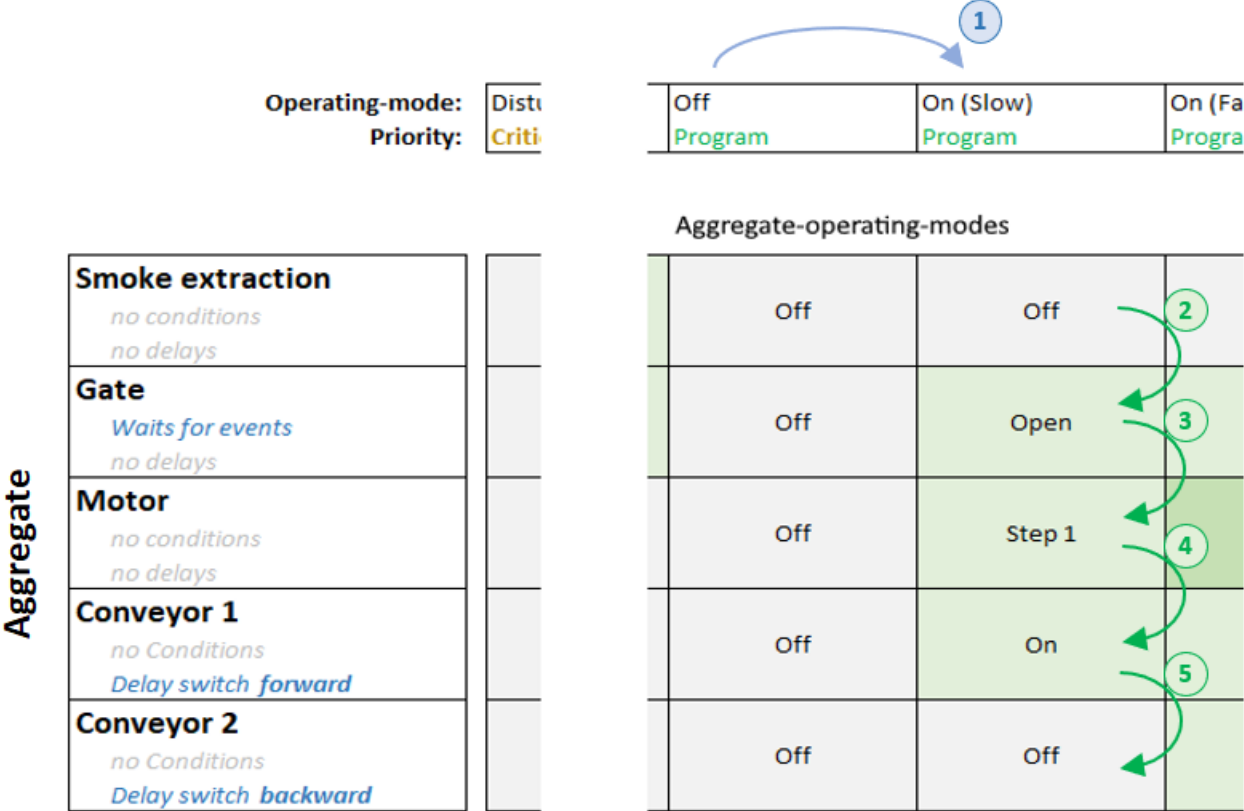
Delays are skipped if the commanded aggregate operation mode is not different from the last aggregate operation mode (the aggregate operation mode of the last commanded plant operation mode).

Stepping up

The plant operation mode is switched upwards (from left to right) in case of *stepping up* or *switching forward*.

All aggregates are set to the respective operation mode one after the other (from top to bottom):

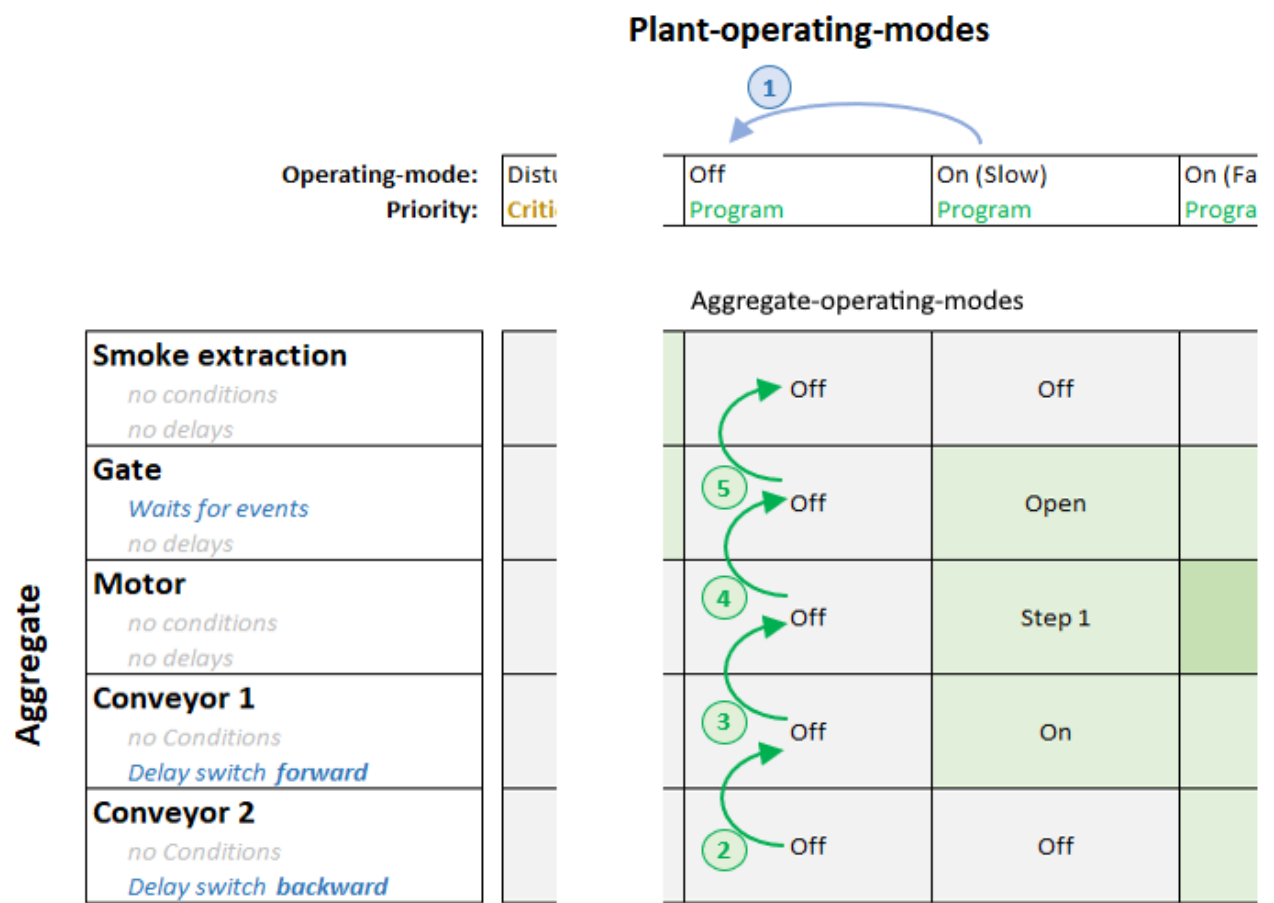
Plant-operating-modes



- 1. *Stepping up* the plant operation mode
- 2. to 5. *Commanding* individual aggregates by level.

Stepping down

The plant operation mode is switched upwards (from right to left) in case of *stepping down* or *switching back*.
All aggregates are set to the respective operation mode one after the other (from bottom to top):



- 1. Stepping down the plant operation mode
- 2. to 5. Commanding individual aggregates by level.

Special features

Interruptions

Changing the plant operation mode, **while** switching down or back is called an interruption.

Direction inversion

The direction is changed by a *interruption* (e.g. from *On (Slow)* to *Off*).

i

Conditions continue to be evaluated.

Continuation

The direction is maintained by a *interruption* (e.g. from *On (Slow)* to *On (Fast)*).

i

Conditions continue to be evaluated.

Repetition

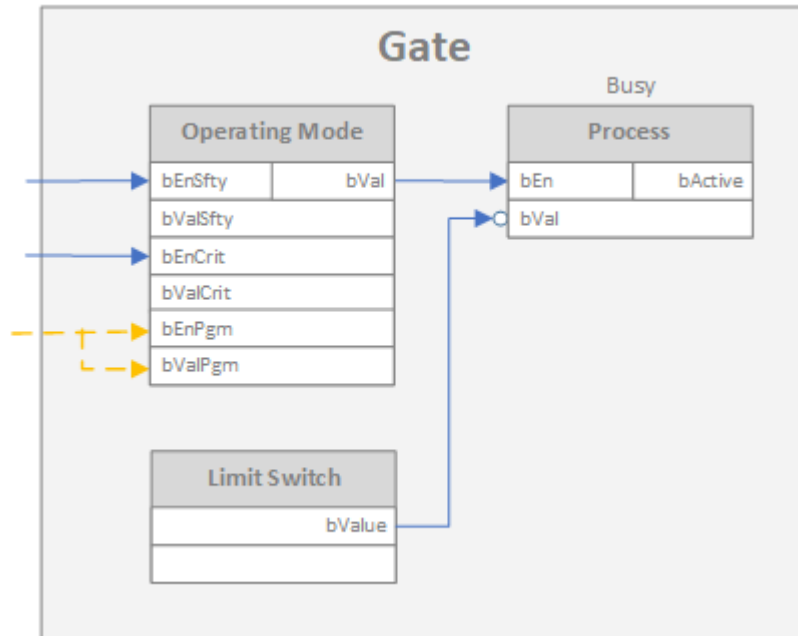
The *repetition* describes the commanding of a plant operation mode which has the same direction of action as the previous plant operation mode (e.g. from *On (Slow)* to *On (Fast)*), but without interrupting an active control.



Conditions are still **not** evaluated.

Processes

A *process* represents a specific workflow within a plant or aggregate.



Example: opening a door. The process *Busy* is active as long as the door is controlled but its limit switch has not yet responded.

Range

A *process* can be evaluated **plant-wide**.

This means that:

- the plant control can react accordingly to the *processes* (see conditions).
- an aggregate *A* can query the *process* of aggregate *B* as long as they are within the same plant.



This enables independent development.

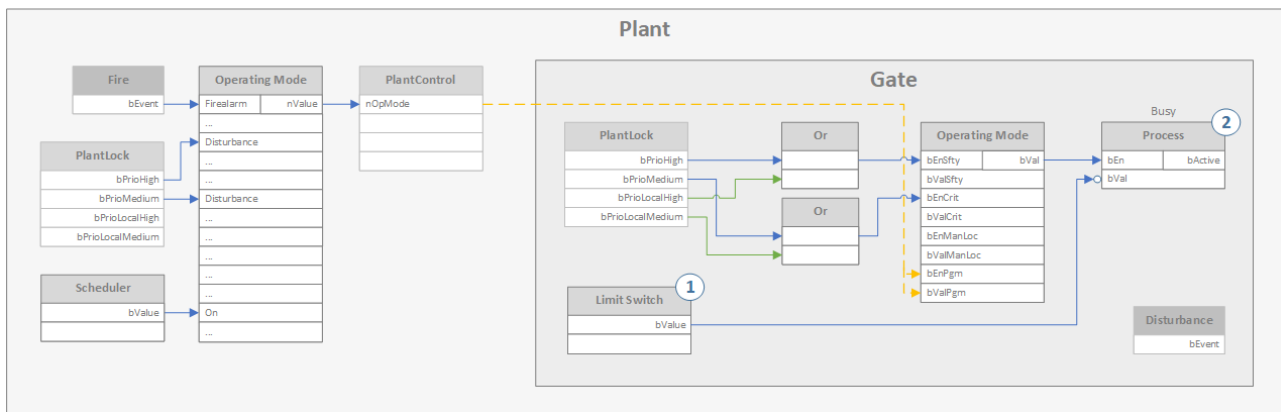
In this way, different aggregates can exchange information with each other via a process without knowing during the development phase whether aggregate *A* exists in the project planning phase (see example Aggregates and processes).

Examples

• Plant control and processes

The example illustrates the described use case and refers to the commanded aggregates from the control matrix.

The *plant control* commands the door aggregate:



When the *door* is switched on, it signals that it is busy by *Process Busy* (2):

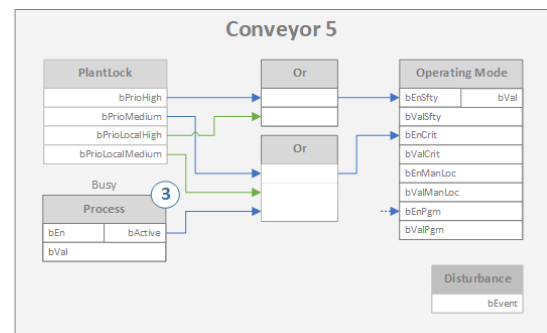
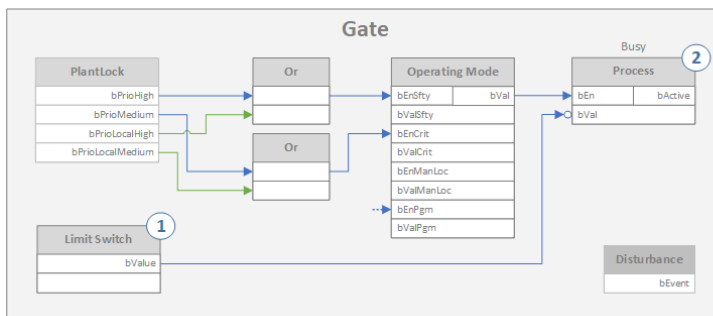
The plant control waits before switching forward.

As soon as the end position (1) is reached, the *process* is completed and switching forward is enabled.

- **Aggregates and processes**

The example illustrates the two aggregates door and conveyor belt 5 in a plant that is not controlled by a plant control.

The aggregates are switched on simultaneously:



The *door* signals that it is busy by *Process Busy* (2).

Conveyor 5 evaluates the same *Process Busy* (3) and goes into a system safe state as long as it remains active.

As soon as the end position (1) is reached, the *process* is completed and *conveyor belt 5* is set to program control priority.



This example illustrates the **independent development**:

In the **development phase** it is not known whether the *door* template will be implemented together with the *motor* template.

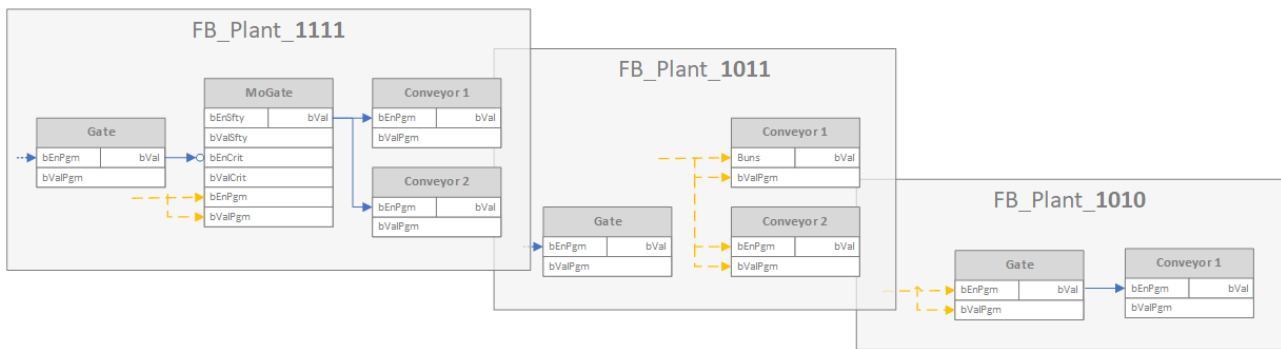
Nevertheless, both constellations will work:

- If there is no aggregate controlling a *Process Busy*, the *motor* (as desired) will never wait for this *process*.

- However, if one (or more) aggregates exist (such as the *door*) that control the *Process Busy*, the *motor* will (as desired) automatically wait for this *process*.

The requirement to cover as many use cases as possible (which can arise during the project planning phase) with a plant template typically results in a large number of template variants:

Different template variants provide different aggregate constellations.



Constellation1111: Contains all intended system components.

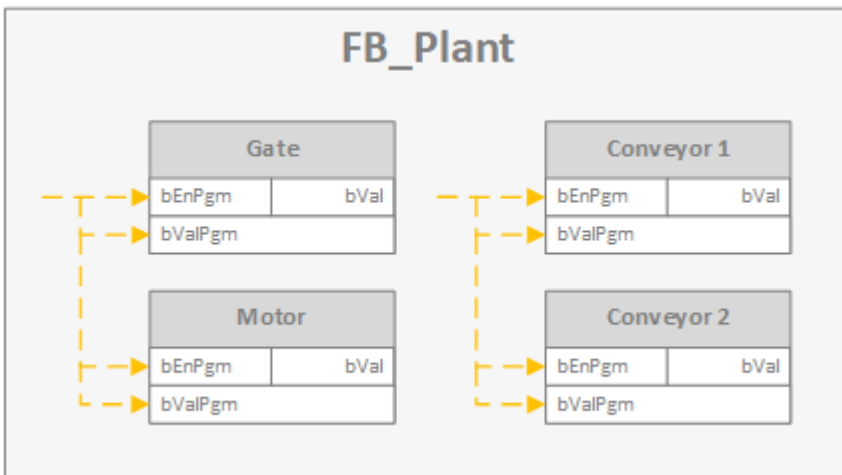
Constellation1011: Contains **no** motor, but door and conveyor belts 1 and 2.

Constellation1010: Contains **no** motor, but door and conveyor belt 1

This problem can be avoided through processes:

Since all aggregates are aware of the relevant processes, all dependencies (e.g. door blocks motor during opening) are clear. This also eliminates the need to program the dependencies.

The example contains all the intended system components. The integrator removes unneeded aggregates without having to consider any dependencies:



PlantLock

Provides information about active, *shutdown* events within its own environment.

The environment always refers to the referenced parent (see Notification).

Switch-off

The desired effect can be achieved by appropriate programming.

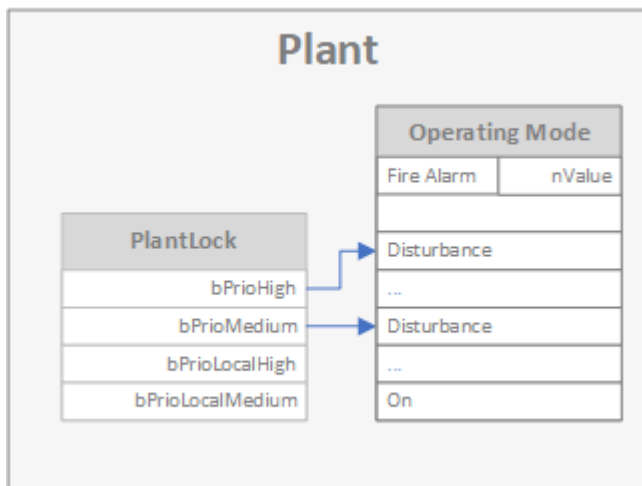


The term *switch-off* arises because the effect of an event is a shutdown.

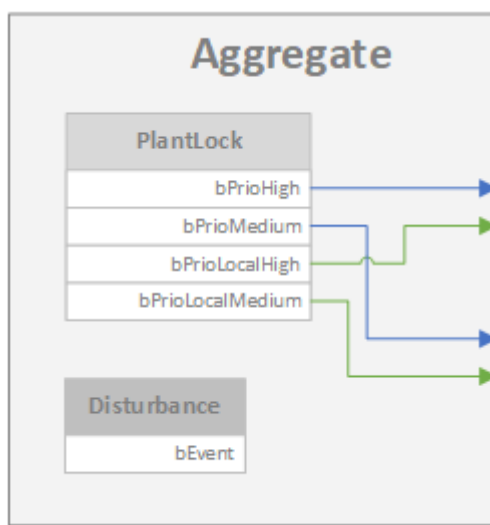
Accordingly, the designation should be *Forcing* or *Holding*. This is because an event enforces a certain state via the lock priority.

There are two types of *switch-off*.

- Higher-level switch-off
The *higher-level* Lock priorities Medium and High are usually used to switch off systems:

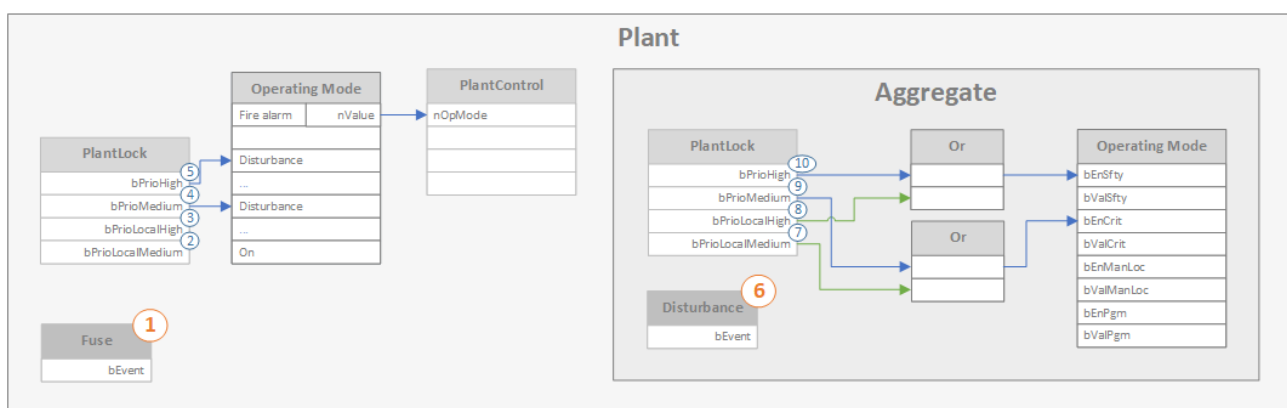


- Local switch-off
The *local* Lock priorities, Local Medium and Local High are usually used to switch off aggregates:



The **higher-level** Lock priorities (Medium and High) should result in the aggregate switch-off.

Examples



- Higher-level switch-off**
The *security* (1) is configured as lock priority medium.
In the event of a fault, the following states result:

Output	1	2	3	4	5	6	7	8	9	10
State	TRUE	-	-	TRUE ₁	-	-	-	-	-	-

1) Effect on the plant, because it is interconnected.

- **Local switch-off**

The *fault* (6) is configured as a lock priority **Local High**.

In the event of a fault, the following states result:

Output	1	2	3	4	5	6	7	8	9	10
State	-	-	TRUE ₁	-	-	TRUE	-	TRUE ₂	-	-

1) **No** effect on the plant, because it is not interconnected.

2) Effect on aggregate, because it is interconnected.

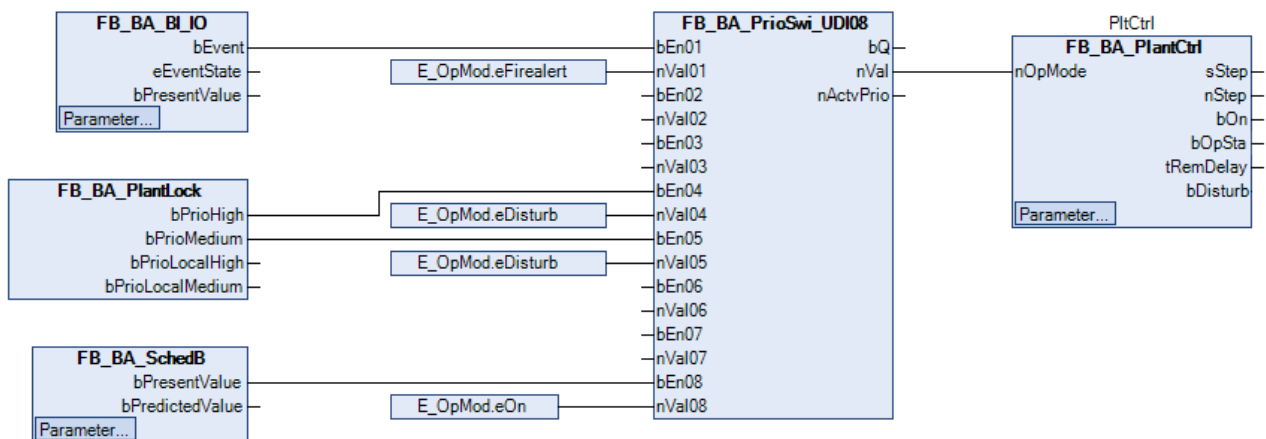
Programming

Plant control

- **Determining the plant operation mode**

The described use case is executed by corresponding *programming*.

Determining the plant operation mode and connecting to the PlantControl:



- **Configuration**

The described use case is executed by corresponding *Initialization*.

Commanding aggregates according to the plant operation mode:

```

PltCtrl : FB_BA_PlantCtrl := (
  iParent
  stOperation := (
    := THIS*,
    aOpMode := [ E_OpMod.eDisturb, E_OpMod.eFire, E_OpMod.eOff, E_OpMod.eSlow, E_OpMod.eFast ],
    aPriority := [ E_Prio.eCritical, E_Prio.eLifeSafety, E_Prio.eProgram, E_Prio.eProgram, E_Prio.eProgram ],
    aAggregates := (
      iReference := SmokeExtract,
      aAggMode := [ E_OpMod.eOff, E_OpMod.eOn, E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eOff ],
    ), (
      iReference := Door,
      aAggMode := [ E_OpMod.eClose, E_OpMod.eOpen, E_OpMod.eClose, E_OpMod.eOpen, E_OpMod.eOpen ],
      bWaitForProcesses := TRUE,
    ), (
      iReference := Motor,
      aAggMode := [ E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eStep1, E_OpMod.eStep2 ],
    ), (
      iReference := Conveyor1,
      aAggMode := [ E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eOn, E_OpMod.eOn ],
      tDelayStepDown := T#0S,
      tDelayStepUp := T#5S,
    ), (
      iReference := Conveyor2,
      aAggMode := [ E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eOff, E_OpMod.eOn ],
      tDelayStepDown := T#5S,
      tDelayStepUp := T#0S,
    ),
  )
);

```

Programmatic creation of a control matrix.



Only *Plant operation modes* > 0 are detected.

- **Control**
Only one aggregate is stepped up or down per cycle,
because each aggregate must be called in order to have the opportunity to evaluate its conditions.

Examples

Alternative initialization

As an alternative to initialization in the declaration part (tabular form), *initialization methods* can be used within *FB_init()*:

```

1      PlantCtrl1.Clear();
2
3      // Add plant operation modes:
4      PlantCtrl1.AddOperationMode(E_BA_AC_OpMod01.eOff, E_BA_Priority.eProgram);
5      PlantCtrl1.AddOperationMode(E_BA_AC_OpMod01.eOn, E_BA_Priority.eProgram);
6      PlantCtrl1.AddOperationMode(E_BA_AC_OpMod01.eFrost, E_BA_Priority.eCritical);
7      PlantCtrl1.AddOperationMode(E_BA_AC_OpMod01.eFire, E_BA_Priority.eLifeSafety);
8
9      // Add aggregates:
10     PlantCtrl1.AddAggregate(Aggregate1, TRUE, T#0S, T#5S);
11     PlantCtrl1.AddAggregate(Aggregate2, FALSE, T#0S, T#0S);
12     PlantCtrl1.AddAggregate(Aggregate3, FALSE, T#0S, T#0S);
13
14     // Apply aggregate operation modes dependant on corresponding plant operation modes:
15     PlantCtrl1.AssignAggregate3x(E_BA_AC_OpMod01.eOff,      Aggregate1, TO_UDINT(0),      Aggregate2, TO_UDINT(FALSE),  Aggregate3, E_BA_MdltCmd.eOff);
16     PlantCtrl1.AssignAggregate3x(E_BA_AC_OpMod01.eOn,      Aggregate1, TO_UDINT(80),   Aggregate2, TO_UDINT(TRUE),   Aggregate3, E_BA_MdltCmd.eOn);
17     PlantCtrl1.AssignAggregate3x(E_BA_AC_OpMod01.eFrost,   Aggregate1, TO_UDINT(0),   Aggregate2, TO_UDINT(FALSE),  Aggregate3, E_BA_MdltCmd.eOff);
18     PlantCtrl1.AssignAggregate3x(E_BA_AC_OpMod01.eFire,    Aggregate1, TO_UDINT(0),   Aggregate2, TO_UDINT(FALSE),  Aggregate3, E_BA_MdltCmd.eOff);

```

1. Define available plant operation modes (AddOperationMode()).
Note the order of the added operation modes.
2. Define aggregates to be commanded (AddAggregate()).
Note the order of the added aggregates.
3. Assign aggregate operation modes to the respective plant operation modes.
The order of the assigned aggregate operation modes has no effect.

Processes

Configuration

- Define referenced parent.
- Referenced label



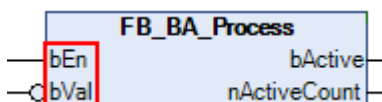
Same *processes* must apply the same *process label*.

Example

Activating a process

Instantiate *process* (e.g. in descriptive aggregate A).

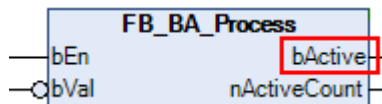
The *process* is active as long as *bEn* and *bVal* have the value TRUE.



Evaluating a process

Instantiate *process* (e.g. in processing aggregate B).

Query *process* by *bActive*.



3.1.2.2 Scheduler

Timing control of building and room automation systems is an important part of building automation. The targeted use of schedules can optimize the efficiency of the building and the comfort for the users. If, for example, a heating system is controlled by a schedule, the room temperature can be adapted to the times of use, thus saving heating energy when the building is not in use.

The implementation of schedules with TF8040 is described in this chapter.

Planning

In general, there are several ways to set a schedule.

Weekly planning

First, there is the weekly schedule. This is the schedule that applies every week, so it describes a general week.

It is set by the parameter `aWeek` [► 188] of the function blocks `FB_BA_SchedA` [► 157], `FB_BA_SchedB` [► 158] and `FB_BA_SchedM` [► 159].

Exceptions

The weekly schedule that applies during standard weeks, or normal operation, of the building must always be modified by exceptions. Examples of such exceptions would be:

- Vacation
- Public holidays
- Meeting room booking
- etc.

The exceptions that can be defined for a schedule are divided into two categories.

Local exceptions

The local exceptions of a schedule are exceptions that apply only to that explicit schedule. I.e. they have no effect on other schedules and are set directly on the schedule object. They are parameterized via the parameter `aException` [► 188] of the function blocks `FB_BA_SchedA` [► 157], `FB_BA_SchedB` [► 158] and `FB_BA_SchedM` [► 159].

The exceptions can be configured as follows:

- **Date:**
An exception can be defined on a specific date. On this date you can then set what should happen at what time. For example, the heating might not be turned down at 5pm that day, as in the weekly schedule, but at 8pm, due to a meeting.
- **Date range:**
Exceptions can be set over a complete date range. If, for example, the building is not planned to be used in a certain week due to company vacations, it can be set that the weekly schedule is overwritten in such a way that the building does not switch to comfort mode at the set operating times during the defined time, thus saving energy.
- **Week and day:**
In addition to a date range, recurring exceptions can also be defined on specific weeks on specific days. This allows exceptions such as "every second Wednesday of the month" to be implemented.

Global exceptions

Global exceptions are defined in the same way by date, date range and week and day. However, as the name suggests, these exceptions do not apply to a specific schedule, but globally. Global exceptions are defined via calendar objects, which are then passed to specific schedules via the parameter [aCalendar](#) [► 188].

Such a calendar could, for example, define the school vacations in the respective state. Thus, all schedules in a school could be overwritten during the vacations.

Programming

The operation of schedules and their exceptions should be understood at this point and programming will now be discussed.

Weekly schedule

The parameter [aWeek](#) [► 188] is an array and therefore it is quite complicated to parameterize it manually. To make it easier for the user, the parameter builder [F BA WeeklyScheduleBuilder](#) [► 208] is provided for this purpose. The use of this parameter builder is described in its documentation.

Local exceptions

The parameter [aException](#) [► 188] is an array and therefore it is quite complicated to parameterize it manually. To make it easier for the user, the parameter builder [F BA ExceptionScheduleBuilder](#) [► 200] is provided for this purpose. The use of this parameter builder is described in its documentation.

Global exceptions

The parameter [aCalendar](#) [► 188] is an array and therefore it is quite complicated to parameterize it manually. To make it easier for the user, the parameter builder [F BA ScheduleCalendarBuilder](#) [► 207] is provided for this purpose. The use of this parameter builder is described in its documentation.

3.1.3 Objects

3.1.3.1 Events

Events can be generated by event-enabled objects for the following purposes:

- Making the operator aware of relevant events or states.
- Triggering appropriate (e.g. safe) system states.

Event classes

An event class describes a very specific configuration (or variant) of an event (e.g. a *simple* alarm event), where properties are defined by means of corresponding parameters.

By providing a fixed set of event classes for each controller, it is possible to define exactly which different configurations are available at all.

The desired event behavior of an event-enabled object is specified by assigning it to an appropriate event class.

Behavior

Event behavior

The behavior of an event-enabled object is determined by the following parameters:

Event type

The respective type of an event is only used for a more detailed description or representation (e.g. HMI) for the viewer.

Type	Symbol
Alarm	
Fault	
Maintenance	
Notification	
Miscellaneous	

Alarm mode

The alarm mode determines whether an event (or an object) must be acknowledged and / or reset.

A distinction is made between the following modes:

Alarm mode	Description
Simple	An <i>event</i> is automatically acknowledged after the transition to the <i>normal state</i> .
Standard	An <i>event</i> continues to be indicated for information purposes after the transition to the <i>normal state</i> . An acknowledgement confirms the <i>event</i> and hides it.
Advanced	An <i>event</i> is not only still displayed after the transition to the <i>normal state</i> , the <i>fault state</i> is also maintained. A simple acknowledgement confirms the <i>event</i> . Only resetting (acknowledging again) puts the <i>event</i> (or the object) back into the <i>normal state</i> and hides it.



Depending on the acknowledgement mode, a simple acknowledgement may be sufficient to trigger the reset.



The default behavior of individual alarm modes can be adjusted by means of global parameters. This may be necessary if, for example, operators require compliance with certain specifications. All settings are carefully preset.

Changes have a **significant** effect on the event behavior of an object and should be implemented with caution!

Transition state

An event-enabled object always assumes one of the following transition states (Transition):

State	Description
Normal	Error-free normal state.
Abnormal	A <i>logical error</i> is present (for example, an active event).
Error	A <i>physical error</i> is present (for example, defect in the connected hardware).

Representation state

The representation of an *event* depends on the following circumstances in addition to the event type of an event-enabled object:

- Event type
- Alarm mode
- Acknowledge and reset state

This results in the following possibilities for displaying an *event* (illustration using the example of an *alarm event*):

State	Graphic	Description
Hidden		The <i>event</i> is not pending.
Indicated*		The <i>event</i> is not (no longer) pending, but is indicated for information purposes until it is acknowledged.
Past and acknowledged**		The <i>event</i> is not (no longer) pending. However, it has already been acknowledged but not yet reset.
Past**		The <i>event</i> is not (no longer) pending. However, it was neither acknowledged nor reset.
Pending and acknowledged		An <i>event</i> is pending and has been acknowledged.
Pending		The <i>event</i> is pending.

- * Only possible with *standard* alarm mode.
- ** Only possible with *advanced* alarm mode.

The following representations result for each event type:

State	Alarm	Fault	Maintenance	Notification	Miscellaneous
Hidden	-	-	-	-	-
Indicated					
Past, Acknowledged					
Past					
Pending, Acknowledged					
Pending					



An *event* is called *active* as soon as it is no longer in the *Normal state* (Hidden).

Acknowledging and resetting

The user can interact with *active events*. He has the following options (depending on the configured alarm mode):

- **Acknowledging**
Signals (e.g. to maintenance staff) a *perceived event*.
In terms of understanding, it should be possible to deduce that a corresponding action is now required.

The acknowledgement is therefore for information purposes.

- **Reset**
In the *extended* alarm mode, an *event* (or an object) must not only be *acknowledged*, but also *reset* in order to restore an already past *event* to the *normal state*.

Resetting therefore prevents the occurrence of undefined states (e.g. uncontrolled restarting of systems) and thus provides additional safety.

Lock priorities

Define the *priority* for shutdown events which, for example, have a desired effect on the plant control or the PlantLock.

Local medium

Enables a local shutdown of *medium** priority.

Local high

Enables a local shutdown of *higher*** priority.

Medium

Enables a higher-level shutdown of *medium** priority.

High

Enables a higher-level shutdown of *higher*** priority.

* Used for system-safe program sections.

** Used for personnel-safe program sections.

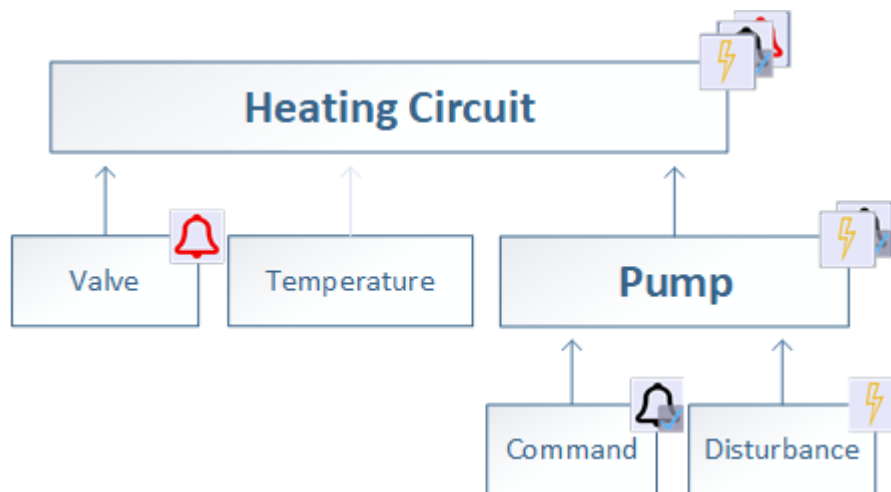
Background services**Notification**

Each event-enabled object reports changes to its parent regarding its *event*.

This results in an overview of the subordinate *events* of each parent or view object.

Furthermore, not only the immediate parent of an object is notified, but all of its higher-level views!

As a consequence, the top view object has knowledge of all *events*.

**Programming****Configuration**

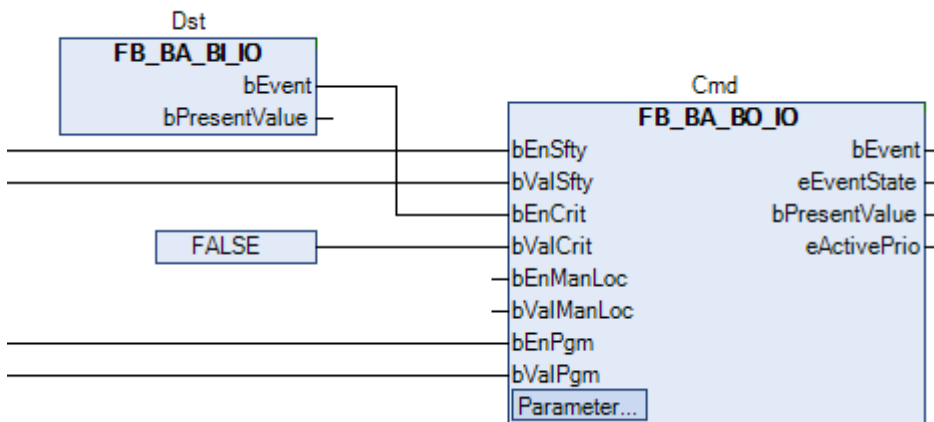
```
Simple      : FB_BA_EC := (
               nInstanceID      := 12,
               eEventType        := E_BA_EventType.eDisturb,
               eAlarmMode        := E_BA_AlarmMode.eSimple,
               aPriority          := [ 10, 11, 12 ]
             );
Disturb     : FB_BA_BI_IO := (
               nEventClassID     := 12
             );
```

Declaration of a simple fault by referencing an appropriately configured event class.

Evaluation

Direct

An active *event* is indicated by the output *bEvent* of an event-enabled object.



Example: Switching off by an event by means of conventional interconnection.

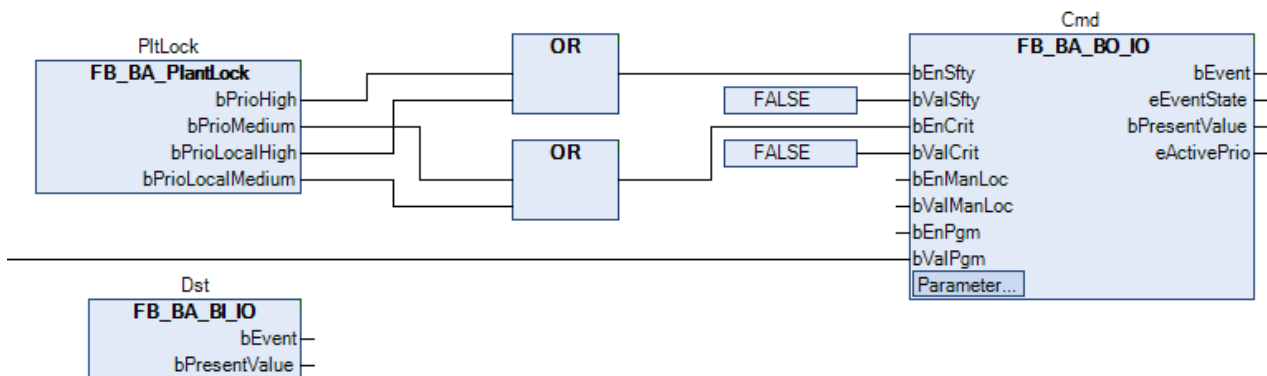
Indirect (PlantLock)

Active, shutdown *events* are output at PlantLock.

```

12  VAR
13      Dst      : FB_BA_BI_IO := (
14          eEnPlantLock      := E_BA_LockPriority.eMedium;
15      );
16  END_VAR

```



Example: Shutdown by one or more events using PlantLock.

3.1.3.2 Parameter

An object has various parameters that are used to parameterize or control an object.

There are two different categories of parameters.

Default parameters

Default parameters refer to the object properties of the BACnet standard. These are usable by the PLC, a BACnet client, the Site Explorer and the HMI.

Examples of default parameters are:

- Object name
- Description
- COV Increment
- Active Text
- ...

Additional parameters

Additional parameters are only in the PLC but not in the BACnet stack. However, they are usable in the Site Explorer and the HMI.

These parameters have been introduced so that an object can be extended by any parameters that are to be displayed in the HMI, for example, but are of no interest to a BACnet management level.

An example of this would be a lighting template that does not need to be controlled from the management level. Here the parameters to control the lamp could be additional parameters.

Access rights

The read and write rights [► 15] to a parameter are defined by roles [► 15]. Read and write accesses to various parameters are defined via the "SetDefWriteAccess" and "SetDefReadAccess" methods.

Default parameters

For default parameters the roles [► 15] are defined in the XBA Param [► 116].

However, the roles can be customized by using the FB_BA Param [► 124]:

Example of setting write and read rights of various parameters.

The following example shows how write access to the parameter *EventEnable* is executed using the method *SetDefWriteAccess* with the aid of the function block FB_BA Param [► 124] on all objects of the base framework. The role "Internal" defines an unrestricted access. The "Basic" role defines a read-only access

The call of this sample is shown via the *FB_Init* method in a function block.

```

FB_BA_Parameter.FB_init  ▶ ×
1  METHOD FB_init : BOOL
2  VAR_INPUT
3      bInitRetains : BOOL; // if TRUE, the retain variables are initialized (warm start / cold start)
4      bInCopyCode : BOOL; // if TRUE, the instance afterwards gets moved into the copy code (online change)
5  END_VAR
6  VAR
7      Parameters      : FB_BA_Param;
8  END_VAR
9
1
2  //Writeaccess on EventEnable Bits (Property 35.)
3  Parameters.SetDefWriteAccess(E_BA_Parameter.eEventEnable, E_BA_ROLE.eInternal);

```

Declaration:

```

METHOD FB_init: BOOL
VAR
    Parameters      : FB_BA_Param;
END_VAR

```

Implementation:

```

//Writeaccess on EventEnable Bits (Property 35.)
Parameters.SetDefWriteAccess(E_BA_Parameter.eEventEnable, E_BA_ROLE.eInternal);

//ReadOnly on Parameter Polarity
Parameters.SetDefWriteAccess(E_BA_Parameter.ePolarity, E_BA_ROLE.eBasic);

```

Additional parameters

The roles for additional parameters are defined via attributes (see below).

Initialization and editing

Default values for the initialization of parameters are described within the templates of TF8040 within the method *FB_Init*.

Reading the data from the method *FB_Init* into the memory for the persistent data of the controller only takes place in the first cycle after loading the function block for the first time. As soon as the initial values from the *FB_Init* exist in the persistent data of the controller, they are not read again even after the values in the *FB_init* have been changed.

A reload of all initial values can be forced by an overall reset of the control system. However, as soon as a part of the program has already been put into operation and extensive parameterizations have been made, an overall reset of the control system is no longer useful. Otherwise, many of the already edited parameters would possibly be deleted.

The editing should be done with the [Site Explorer \[► 1024\]](#) or the [Symbol Explorer \[► 1042\]](#) after the initialization by the method *FB_Init*.

In order to safely avoid a loss of once edited parameters, it is advisable to always perform data backups after and during commissioning. A backup is created with the tool [Symbol Explorer \[► 1042\]](#) by taking a snapshot.

Programming

Additional parameters

Any variable in a function block that inherits from [FB_BA_Object \[► 195\]](#) can be made an additional parameter by using certain attributes.

To declare a PLC variable as an additional parameter, only the *BaParam* attribute must be defined.

```
VAR
    {attribute 'BaParam'}
    SomeParameter : INT;
END_VAR
```

To further describe the additional parameter, the following attributes are available:

Title

Adjusts the name of a parameter.

Identifiers are defined for one or more languages.

Default

The default value corresponds to the symbol name of the variable.

Sample:

```
{attribute 'BaParam.Title.DE' := 'Irgendein Wert'}
{attribute 'BaParam.Title.EN' := 'Some value'}
```

Access

Adjusts [access rights \[► 15\]](#) of a parameter.

Default

The default settings are:

Access	Role
Read	Internal
Write	-

Sample

```
{attribute 'BaParam.Access.Read' := 'Advanced'}  
{attribute 'BaParam.Access.Write' := 'Expert'}
```

Enumerations

Adjusts the name of an enumeration value.

Identifiers are defined for one or more languages.

Default

The default value corresponds to the symbol name of the enumeration value.

Sample

```
{attribute 'BaParam.Enum.eAdd.DE' := 'Hinzufügen'}  
{attribute 'BaParam.Enum.eAdd.EN' := 'Add'}  
{attribute 'BaParam.Enum.eSubstract.DE' := 'Abziehen'}  
{attribute 'BaParam.Enum.eSubstract.EN' := 'Substract'}  
TYPE E_Mode :  
(  
    eAdd,  
    eSubstract  
);  
END_TYPE
```

Languages

Values can be defined in multiple languages for certain attributes:

Language	Abbreviation
English	EN
German	DE

3.1.3.3 Object description texts

The following object parameters are summarized as *texts*:

- ObjectName
- Description

The texts of a [object](#) [► 28] are generated automatically only if a placeholder is included. This is typically the case when the PLC starts for the first time.



Since from now on all texts have a defined value (which is also persistently stored) a new generation no longer takes place.
Texts are changed from this point only by the integrator or the operator!

Labels

A Label FB (e.g. in a GVL) can be used in the declaration of one or more objects to generate its texts automatically.

A distinction is made between the following label types:

- Simple labels (FB_BA_Label)
Provides the possibility to define texts for general use.
This means that simple labels can be reused in all objects, but also in further labels.
- Dual labels (FB_BA_Label2x)
Provides the option of defining the equipment identifier of an object.
This means that both aggregate and function information can then be described.

Any (single or double) label can be assigned to each object.

Each view should be assigned a simple label which is interpreted as aggregate information. Function information is not supported by views.

Programming

Placeholder

Placeholders represent a value that will be generated later. For example, the parameters *ObjectName* and *Description* are initialized with default placeholders as follows:

```
sObjectName : T_MaxString := '{}';
sDescription : T_MaxString := '{}';
```

During object initialization, the mechanism for generating a title parses for defined placeholders.

Only if a valid placeholder exists, the mechanism can replace it with an automatically generated text.

On the other hand, if a manually generated text had been inserted instead of the placeholder, then it would not be overwritten.

After the texts have been created, the variables no longer contain placeholders.

Even if the control system is restarted and the parameters (persistent variables) are saved, the current texts remain active (automatically generated, manually defined or modified by the user at runtime) and are not overwritten by the text generation mechanism.

Purpose of use

- Substitute
A placeholder (here an index) is substituted for a value:
sDescription := '{Idx}'
- Defining
A placeholder (here an index) is used to define a specific value:
sDescription := '{Idx=10}'

Indexing

- Automatic indexing
The equipment is indexed automatically. Indices (e.g. for aggregates) are automatically appended after the text.

While indices for the *ObjectName* parameter are normally always generated, *Description* indices appear depending on the current setting.

- Manual indexing
The indexing can be specified by placeholders and their corresponding index attributes.

Manually defined indices (regardless of the purpose) are always displayed in the generated text.

A specified index is automatically also used for parameters of subsequent objects if these do not also explicitly define an index.

Example:

Since an index of 10 is defined for Pump1, index 11 is automatically used for Pump2.

```
Pump1 : FB_BA_View := (iLabel:=SomeLabel, sObjectName := '{Idx=10}');
```

```
Pump2 : FB_BA_View := (iLabel:=SomeLabel);
```



Manually assigned indices must be assigned carefully, as they are not validated by the mechanism! The use of the same *ObjectName* in different objects is prohibited, as some Supplement objects perform a unique identification via the *ObjecName* parameter.

Concatenation

The texts of an object are concatenated with texts of parent objects under certain conditions:

- Parent settings
- Settings in the DPAD-FB used

- Manual overwriting using index attributes
- Global settings
- Placeholder with concatenation prohibition

Use cases

Use case 1

A placeholder refers to a specific parameter.

Sample: Initialization of the ObjectName

```
sObjectName := '{}'
```

Use case 2

A placeholder represents another parameter.

In this case, the placeholder must be addressed with its corresponding abbreviation.

Sample:

```
sDescription := '{} - ID {InstID}'.
```

Parameter	Shortcut
Event Transition Text	EvtTrans
Present Value	PrVal
Device Type	DevType
Instance ID	InstID
Object Name	ObjName
Description	Descr

Attributes

Attributes are typically used in the application (for example, in a template) because the application-specific details are known there, but not the operator-specific details.

Attributes	Shortcut	Type
Aggregate Index	Idx	Unsigned number
Function Index	Idx	Unsigned number

Index attributes

Whether an index refers to an aggregate or a function depends on the order.

- A stand-alone index refers to the function.
- Two indices refer to the aggregate (index 1) and the function (index 2)

Syntax

Syntax	Sample	Description
{ }		Placeholder
{Value}	sObjectName := '{MyValue}'	Placeholder that overwrites parameter values.
{Attribute} {Attribute1,Attribute(n), ...}	sDescription:= '{Idx}'	Placeholder as representing attribute.
{Attribute=Value} {Attribute1=Value,Attribute(n)=Value,...}	sDescription:= '{Idx=10}'	Placeholder as defining attribute.
{Value{Attribute=Value}}	sDescription:= '{Pump{Idx=10}}'	Placeholder as its defining attribute, which overrides a parameter value [► 32].
{Placeholder{Attribute1=Value,Attribute2=Value}}		Placeholder with multiple attributes.
{Parameter}	sDescription:= '{ } {InstID}'	Placeholders as representing parameters [► 32].
!{ }		Placeholder that does not allow concatenation.
{Value{!Attribute}}		Placeholder that does not allow certain attributes.

Samples

Samples of correct usage:

Placeholder	Description
sObjectName := '{ }	<i>ObjectName</i> with a default placeholder (concatenates autogenerated values).
sObjectName := '{AAA01_BB01}'	Concatenated <i>ObjectName</i> with overwritten value.
sDescription:= '{ } North'	Concatenated description with a placeholder and fixed additional information that is not replaced.
sDescription:= '{Idx}'	Concatenated description with a placeholder containing a representing index attribute.
sObjectName := '{Idx=10}'	Concatenated object name with placeholder representing a defining index attribute.
sObjectName := '{Idx,Idx=10}'	Concatenated description with a placeholder containing representing (aggregate) and defining (function) index attributes.
sDescription:= '{Pump{Idx=5}}'	Concatenated description with overwritten value and a defining index attribute.
sDescription:= '{ } - ID {InstID}'	Concatenated description with placeholder and a representing parameter.
sObjectName := '!{ }	<i>ObjectName</i> with default placeholder but rejected concatenation.
sDescription:= '{PU{!Idx}}'	Concatenated <i>ObjectName</i> with overwritten value but denied index attribute.

3.2 HMI

3.2.1 BalInterface

TcHmiBa controls that support the BalInterface can also be used without the BaSite and TF8040 PLC templates. Functionality and behavior may vary.

Use

The `BaInterface` attribute from the `BaData` category requires a `TcHmi` symbol.

Without specification of structure and content by a TF8040 PLC template, the assignment of the variables for the control must be done via the attribute `BaInterfaceSymbolNames` from the category `BaData`.

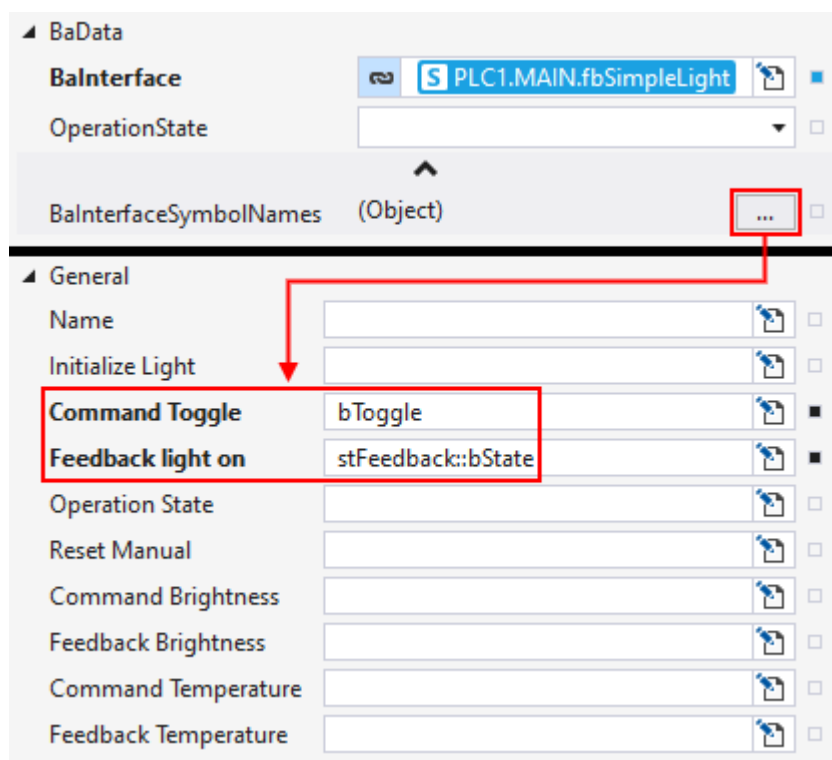
Sample

Explanation using the sample of a self-created PLC function block `FB_SimpleLight` for toggling a light. The state-variable `bState` is stored in a structure.

```
FUNCTION_BLOCK FB_SimpleLight
VAR_INPUT
    bToggle      : BOOL;
END_VAR
VAR_OUTPUT
    stFeedback   : LightFeedback;
END_VAR
```

This is to be used by the `TcHmiBa` control `Light` [► 945].

For this purpose, the symbol of the function block is transferred to the `BaInterface` attribute and then the variables are assigned via the `BaInterfaceSymbolNames` attribute.



Using nested symbols

Nested symbols can be accessed with the following syntax.

```
stFeedback::bState
```

3.2.2 BaTemplate

The `BaSiteExtension` provides the `TcHmi` with the structure of the PLC in a special form. `BaObjects` are created from this, which can then be linked to the `BaObject` property from the `BA` category of a `TcHmiBa` template [► 901].

The `TcHmiBa` templates are implemented as `TcHmi-FrameworkControls` and take all the values required for their function from the `BaObject`.

Creating such a control requires more effort, but the result is a more performant and comfortable TcHmiBa template for the user.

BaTemplateDescription

TcHmiBa templates are available for almost all TF8040 standard PLC templates. The scope of functions includes a validation of the linked BaObject using the *BaTemplateDescription*. It defines the symbol names including hierarchy in the delivery state, which a TcHmiBa template requires in *BaObject*.

In case of changed symbol names, due to project specific peculiarities, the new name can be communicated to the TcHmiBa template via the attribute *BaTemplateDescription* from the BA category.

Template handling

Explanation of the functionality using the example of the standard TF8040 PLC template *FB_BA_AC_CoIT_02* for a cooler.

The matching TcHmiBa template *FB_BA_AC_CoIT_02* expects the following subelements (TF8040 PLC templates) in the linked *BaObject* in the predefined hierarchy.

Subelements:

Symbol name	PLC template	Description
Fdb	FB_BA_AI_IO	Feedback
MdIt	FB_BA_AO_IO	Setpoint
MV	FB_BA_AI_IO	Feedback
TFI	FB_BA_SensorAnalog_IO	Flow temperature sensor
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv	Valve

Hierarchy:

- BAObject
 - TFI
 - MV
 - TRt
 - MV
 - Vlv
 - Fdb
 - MdIt

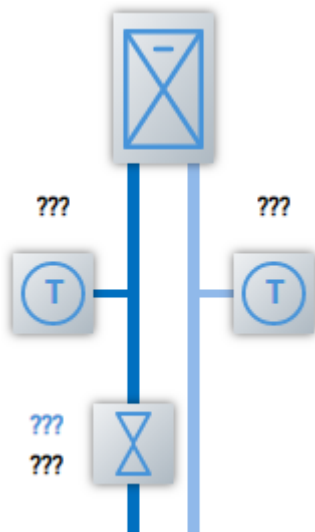
States

The states of a TcHmiBa template depend on the *BaObject*.

No BaObject

The cooler without linked *BaObject*. All subelements are present, but without value display.

Presentation:



Subelement missing

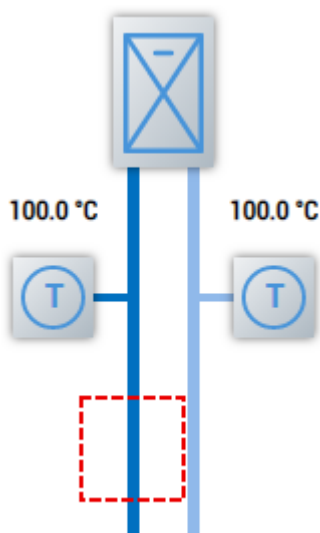
The cooler has been assigned a *BaObject* in which the valve (Vlv) is missing.

The validation registers that a top-level subelement does not exist or has a different symbol name, and therefore hides it.

Hierarchy:

- BAOBJECT
- TFI
- MV
- TRt
- MV

Presentation:



Symbol name renamed

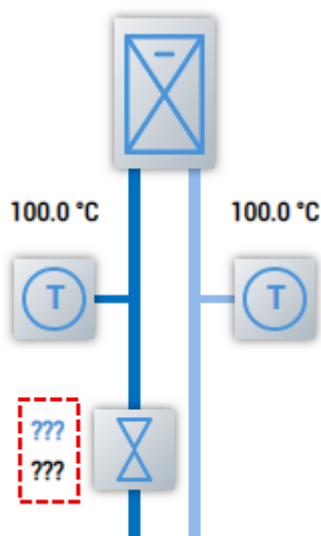
In this *BaObject* the top level subelements are correctly present, but the symbol name for Fdb from the Vlv has been renamed in the PLC template.

Since the deviation occurs in one of the lower levels, the affected subelement is shown with the value display in the error state.

Hierarchy:

- BAObject
 - TFI
 - MV
 - TRt
 - MV
- Vlv
 - Fdb_Test
 - MdlT

Presentation:

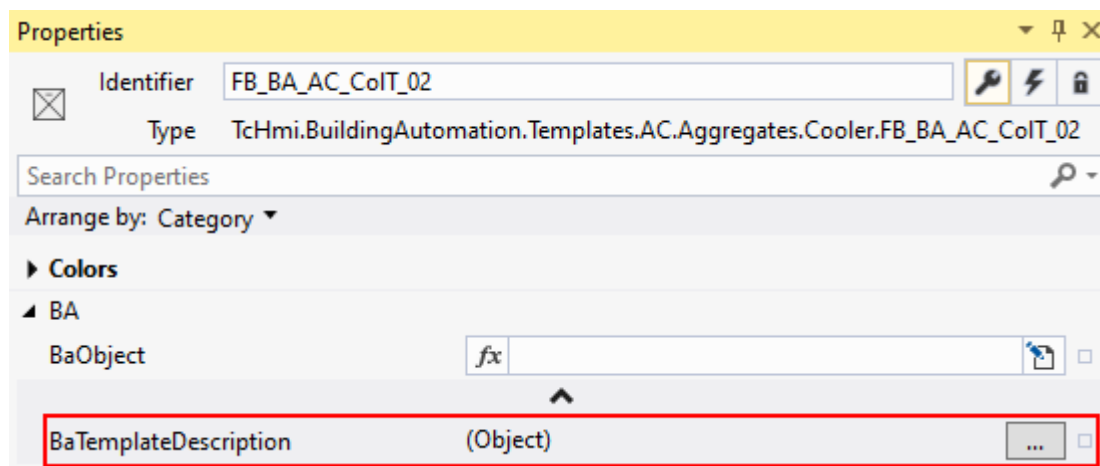


In addition, an error message indicates the subelement.



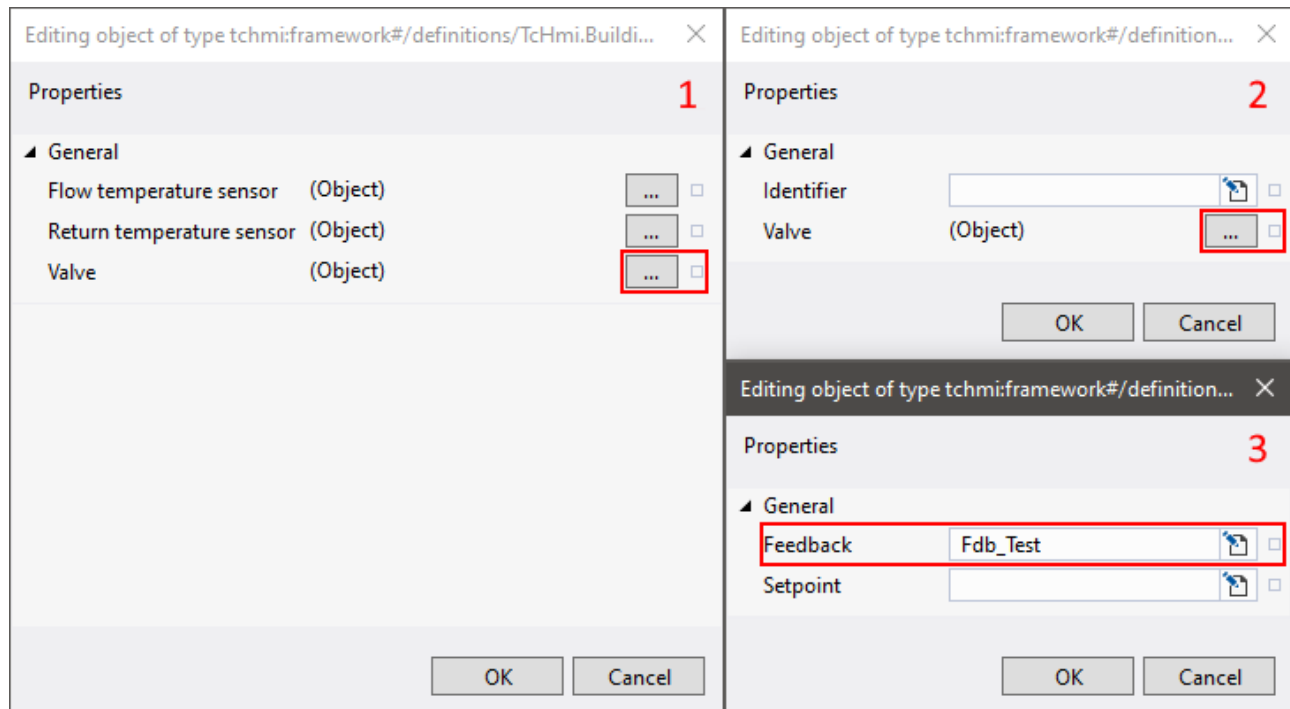
Customize symbol names

New symbol names can be communicated via the **BaTemplateDescription**.



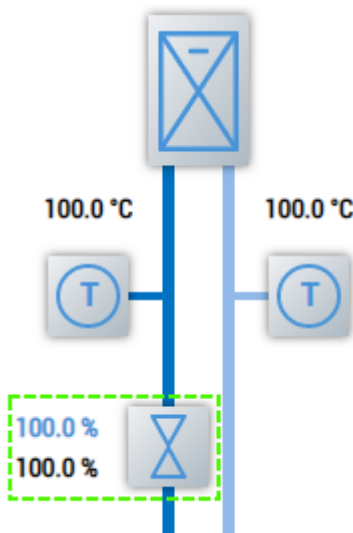
A dialog lists all subelements and allows navigation through the further levels [1]. The **identifier** refers to the symbol name of the current level [2]. If this specification is no longer possible, then the last level of the subelement is reached [3].

At **Feedback** the current symbol name for **Fdb** from the PLC template is to be entered.



After confirming the input, the TcHmiBa template performs the validation of the BaObject again.

Presentation:



The cooler is fully functional again.

Using nested BaObjects

If the BaObjects, e.g. for the command of a motor, are not on the level provided by default, but in a lower level, these nested objects can be accessed with the following syntax.

```
MyView::Motor::Cmd
```

3.2.3 Feedback

In some controls, a so-called *feedback* is used.

The feedback should help to be able to give the user a feedback, if a BV is written, but the confirmation (feedback) of the PLC is still pending, for example.

Accordingly, the controls always display only the feedback value (e.g. *StateFeedback* of the *checkbox*), which ensures that the current value is always displayed in the HMI.



Changing the value of *State* has no effect on the display in the HMI, as only the value of *StateFeedback* is displayed.

Procedure

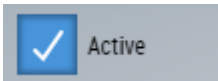
The procedure using the example of the [checkbox \[► 872\]](#) is as follows:



1. Write value by marking the checkbox
2. Value is written to the *State* symbol
3. Display of the loading animation



4. Response of the PLC:
Hiding the loading animation



There are only two possible outputs.

- a. Value of symbol linked to *StateFeedback* changes to the same value written to *State*.
- b. Value of *StateFeedback* does not change.

3.2.4 Generic

The creation of an HMI for a building automation system can be very complex and only parts of an HMI can be reused in different projects. Often the budget available for an HMI is also limited. For this reason, it is worthwhile using an HMI that already generates certain parts generically.

This generic is possible with the TF8040 Building Automation solution. This means the [Tc3_XBA \[► 92\]](#) library for the PLC and the for the HMI.

The goal of TF8040 is for the integrator to develop his system at only one central point - the PLC.

Due to the way in which systems are implemented with TF8040 and the resulting structures, it is possible to derive generic functions for the HMI.



For more information, see the documentation for [Generic HMI \[► 63\]](#).

Project structure

The generic functions within *TcHmiBa* are based on the project structure, which is established in the PLC by a child/parent relationship. With this structure it is possible to derive a [generic navigation \[► 63\]](#) through all objects on a controller.



Further information on the project structure can be found in the documentation.

Events

The events of all linked controllers can be collected and displayed centrally in an [event list](#) [► 886]. This function also results from the engineering in the PLC and does not require any further configuration in the HMI.

Trending

In addition, the generic design makes it possible to implement various trend functions without having to configure them separately for the HMI.



For more information, see the documentation on [Trending](#) [► 45].

3.2.5 Trending

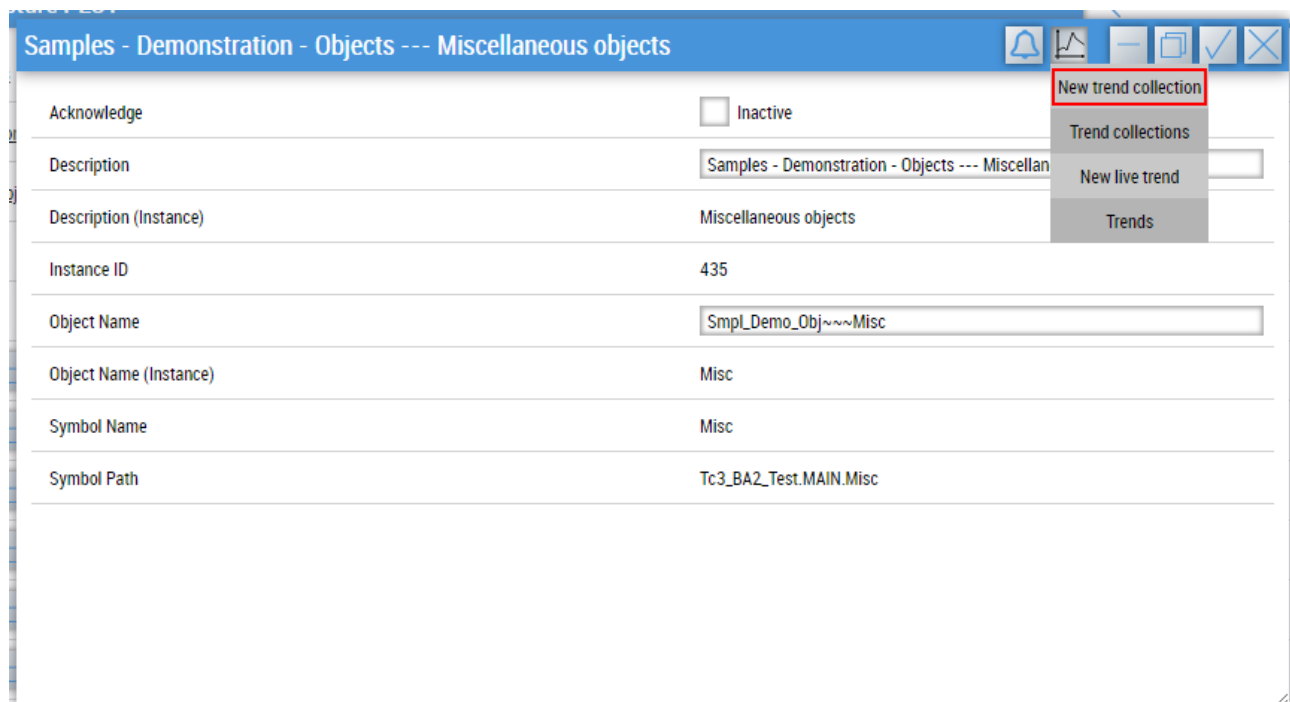
Trends can be used and managed in the HMI. The functions described here can only be used with the [generic functions](#) [► 63] of TcHmiBa.

Trend collections

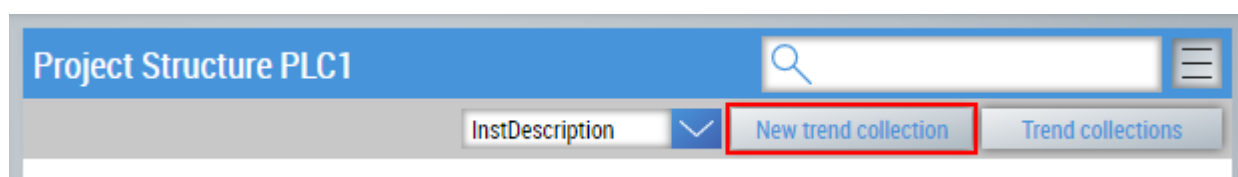
Displaying individual trend curves is often not sufficient, as they are difficult to compare in this way. For this reason there is an option to create trend collections in order to compare the trend curves in a chart.

Configurator

The configurator can be opened via the menu in the [Parameter window](#) [► 891] of a view and then only displays objects from this level.

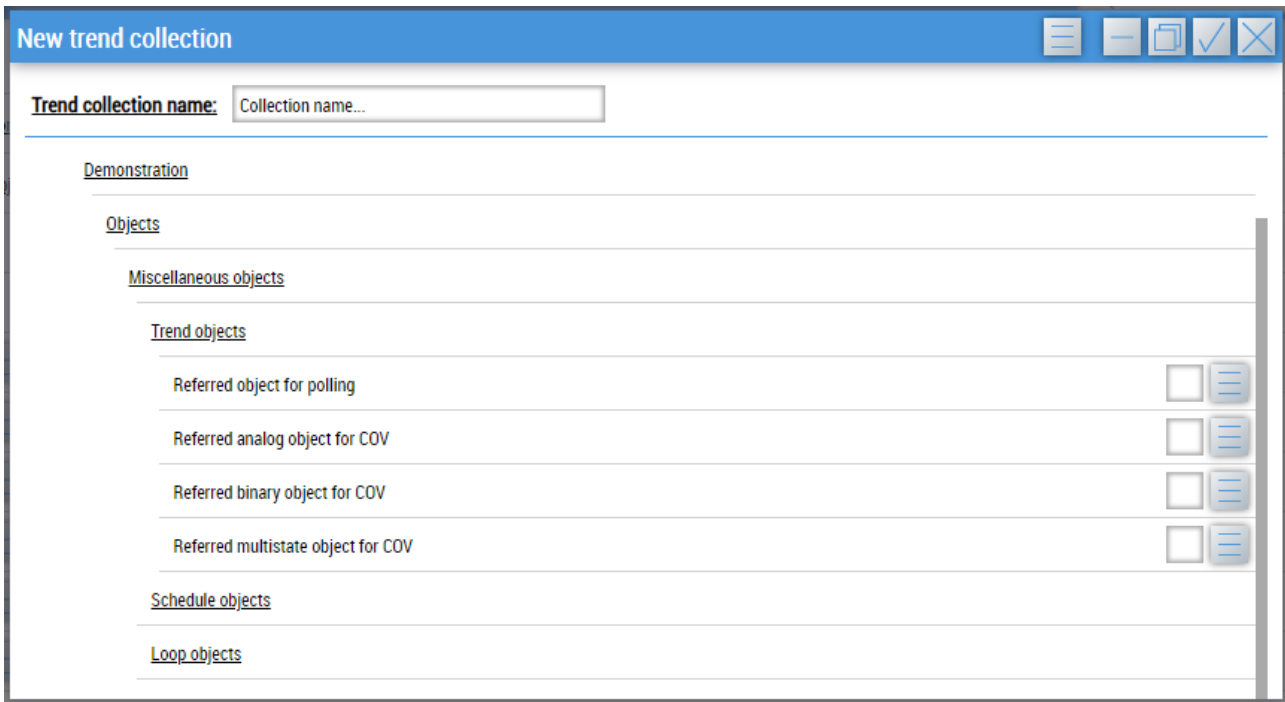


In addition, it can also be opened via the menu of [ProjectNavigationTextual](#) [► 889].

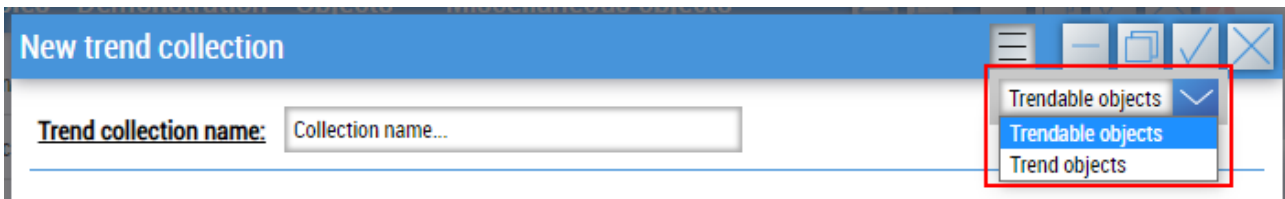


Creating trend collections

Trend collections are created with the configurator. After assigning a name, the trendable objects can be added to the trend collection via the checkboxes.

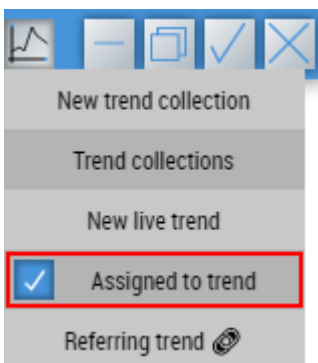


The menu offers the option to select the objects to be displayed.

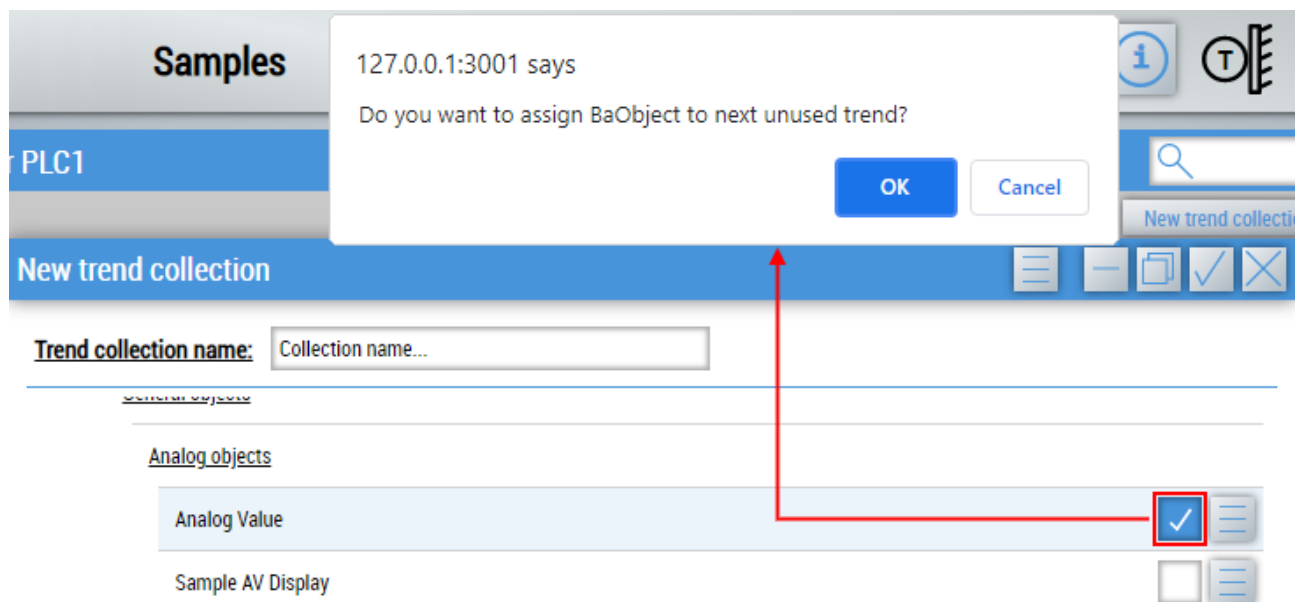


- **Trendable object:** Displays all trendable objects (e.g. *FB_BA_AV*).
- **Trend object:** Displays all trend objects (*FB_BA_Trend*).

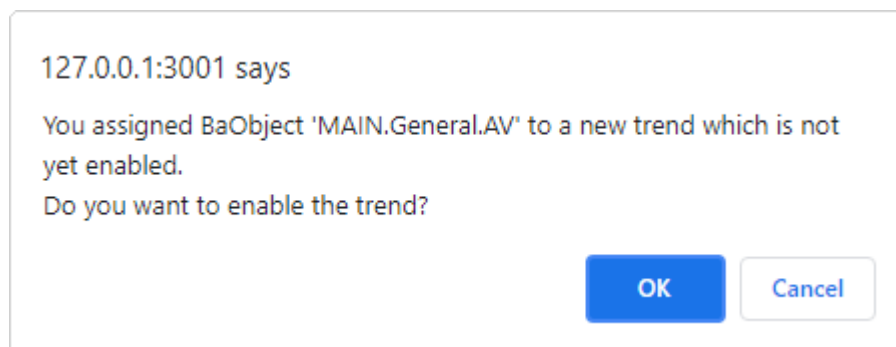
A **Trendable object** must be assigned to a trend object for use in the trend. The state for this can be viewed and also changed in the menu of the Parameter window [► 891].



If an assignment has not yet been made at the time of selection, this is done automatically by confirming the query with **OK**.



Subsequently, the trend can be activated directly with **OK**.



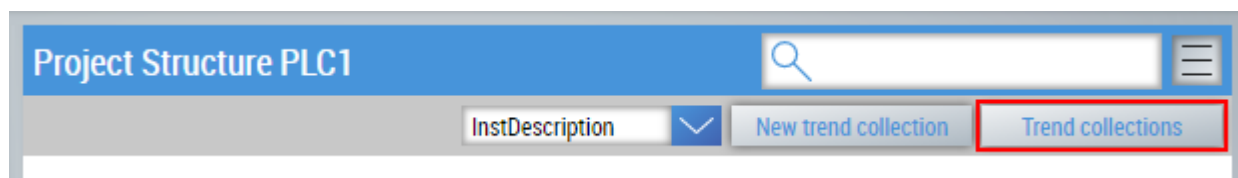
When the trend collection is complete, it will be available in the trend collection view after confirming the dialog.

Observing trend collections

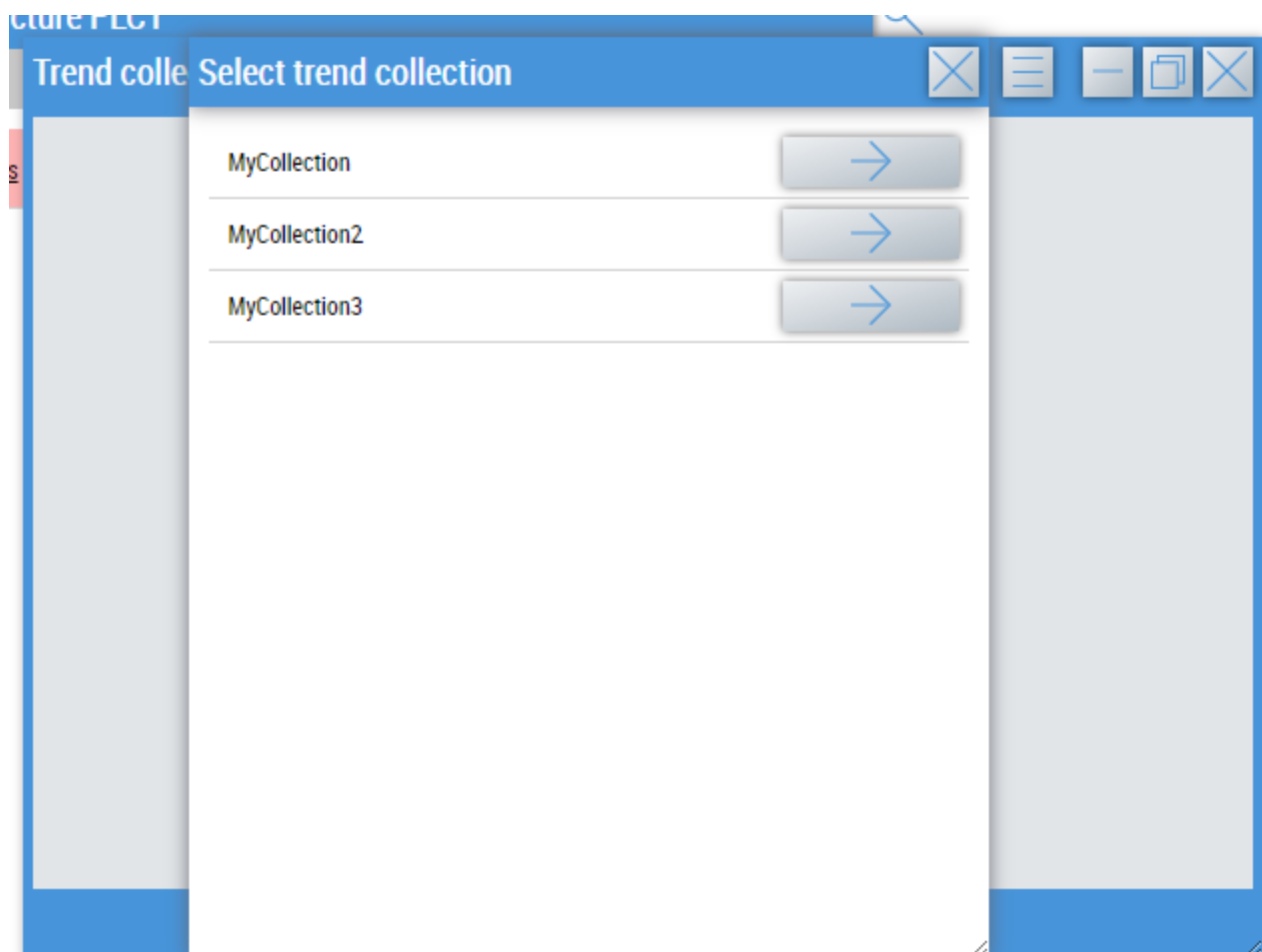
The created trend collections can be viewed and added in the trend collection view.

This can be called up in two ways.

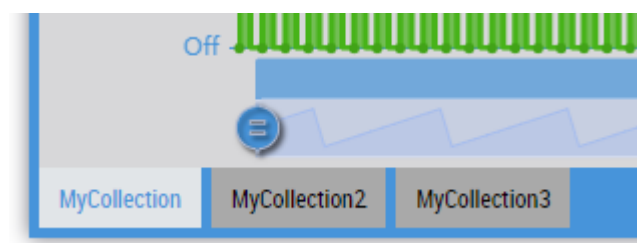
1. By calling the function `OpenTrendCollectionView` [► 1003].
2. From the `ProjectNavigationTextual` [► 889].



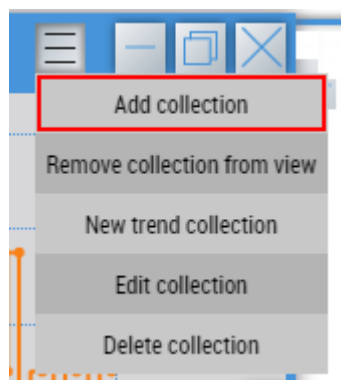
When the dialog is opened for the first time, an overview appears. A trend collection can be selected from it and then displayed in a `Trend Control` [► 897].



The organization is done in tabs and allows you to quickly switch between the different trend collections.



Additional trend collections can be added via the menu.



Further actions are available in the menu for managing the trend collection.

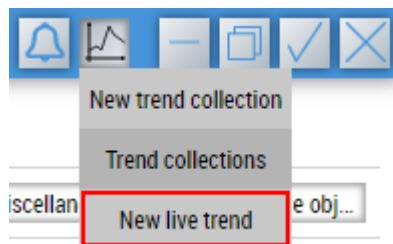
- **Remove collection from view:** Removes the trend collection in the active tab from the view (it will not be deleted).
- **Edit collection:** Opens the configurator to edit the trend collection in the active tab.

- **New trend collection:** Creates a new trend collection. At this point, all objects of the project structure are displayed. The first call may therefore take some time.
- **Delete collection:** Opens the trend collection view. The selected trend collection is permanently deleted.

Live Trends

If the behavior of a value is to be observed only briefly and not assigned to a trend object, it is possible to create a live trend via the menu of the object in the Parameter window [► 891].

This action is only available if the object itself is trendable (e.g. FB BA MV [► 169]) or is a view with trendable objects.

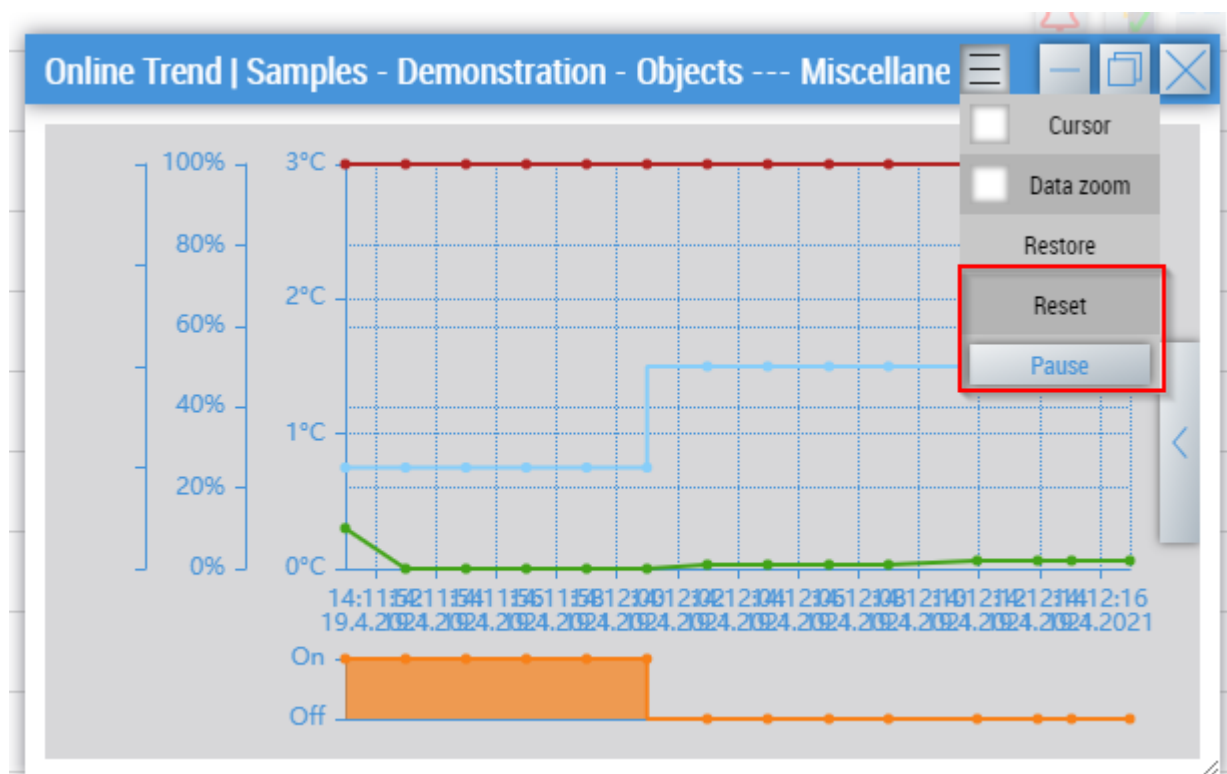


After pressing the button, the live trend opens, which records either only one trendable object or all trendable objects of the view in a FIFO memory. The functions in the menu are similar to those of the Trend Control [► 897].

NOTICE

Loss of data

When the window is closed, the recorded data is lost.



The menu contains two additional actions:

- *Reset:* Deletes all recorded values.
- *Pause:* Pauses the recording of new values.

4 Tutorials

Below are some tutorials to help you get started with TF8040.

Getting started

PLC

Creating a [TwinCAT PLC project](#) [► 50].

HMI

Creating a [TcHmiBa project](#) [► 56].

4.1 PLC

4.1.1 Starting a project

This chapter describes how to start a project.

Procedure

The procedure described here includes all settings that are relevant for a functioning project.



When using the [TF8040 PLC project templates](#) [► 606] individual steps are already prepared accordingly.

Updating the runtime

If the runtime on the target device is not up-to-date, it should be updated accordingly:

XAR

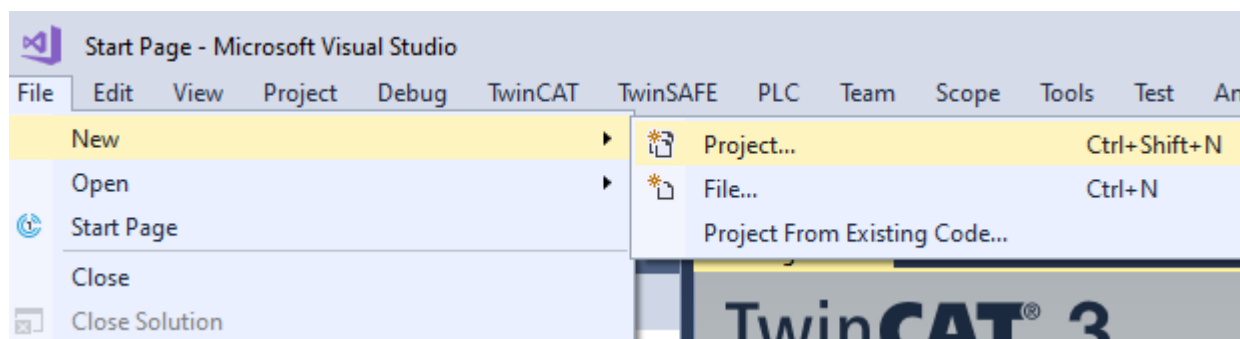
Install current XAR on full Windows systems.

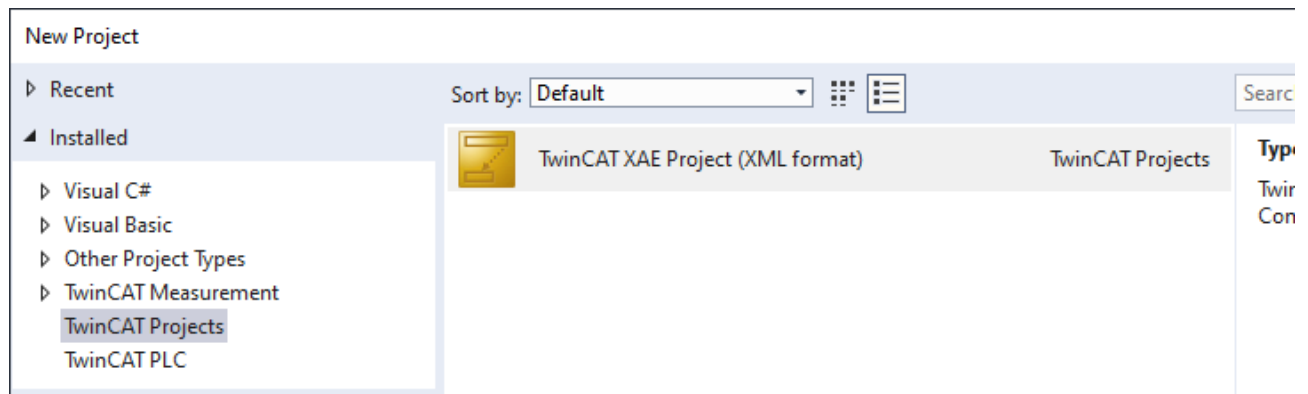
Image

Install current image on the other systems (e.g. Windows Compact 7).

Creating a TwinCAT project

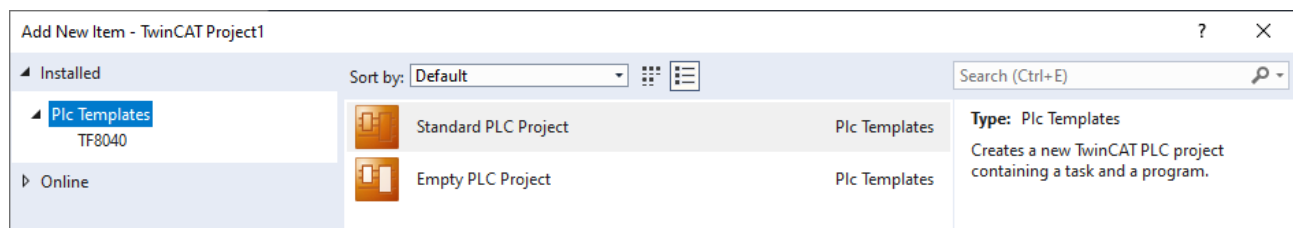
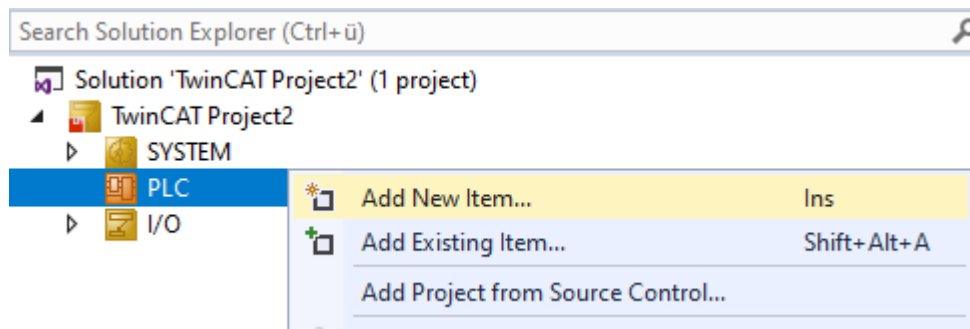
- Create new Solution.





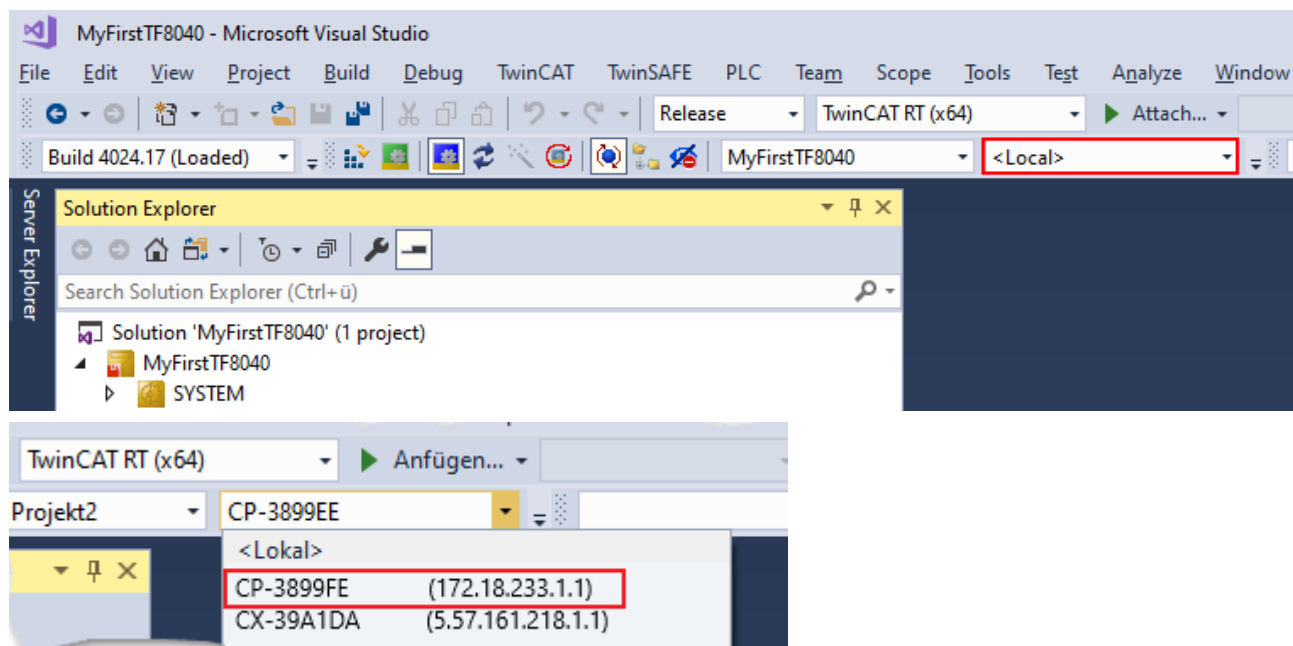
Adding a PLC project

- Add matching PLC project template [► 606]:



Choose Target System

- To proceed with the project settings, you must first select a controller as the target system.





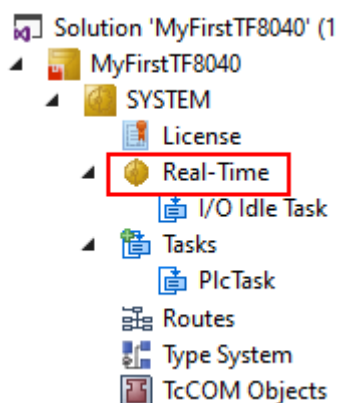
For more information refer to the chapter [Choose Target System](#).

Project settings

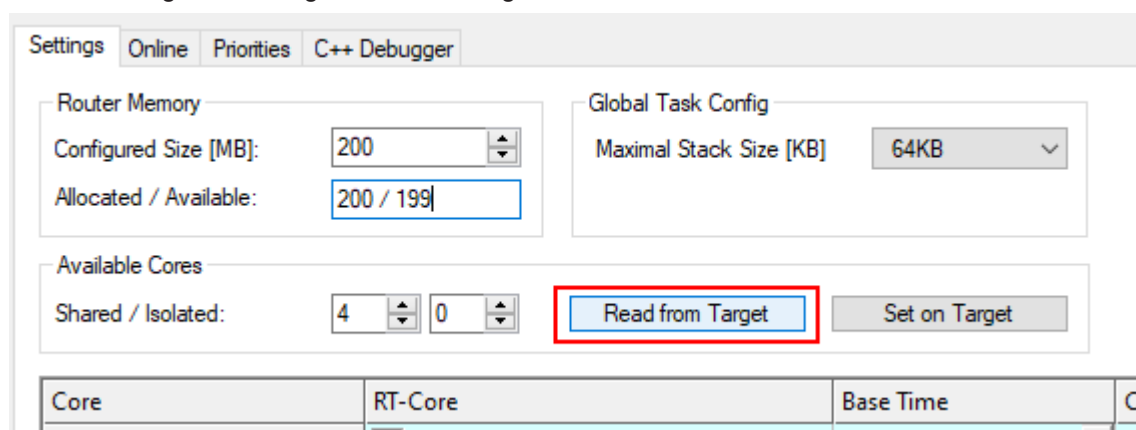
After the target system is selected, you can proceed with the project settings.

System

Real-time

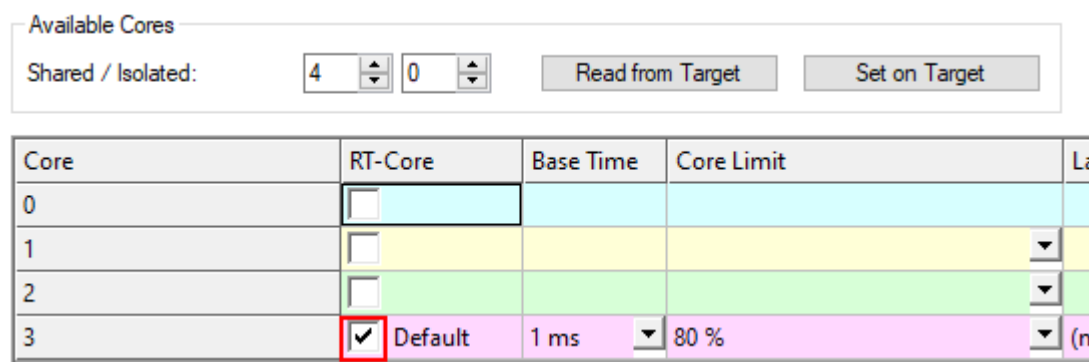


- Reading the existing hardware configuration:



- Selecting a core:

If there are several cores to choose from, the last core is recommended, since the load generated by the operating system tends to be small there.

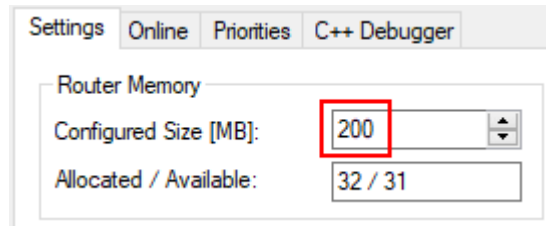


NOTICE

Do not use isolated cores.

- Configuring the router memory

The memory should generally be set to 200 MB:



Settings Online Priorities C++ Debugger

Router Memory

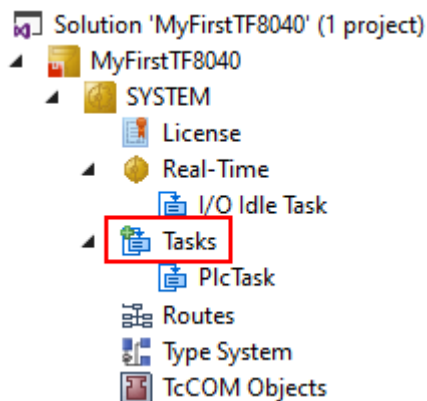
Configured Size [MB]: 200

Allocated / Available: 32 / 31



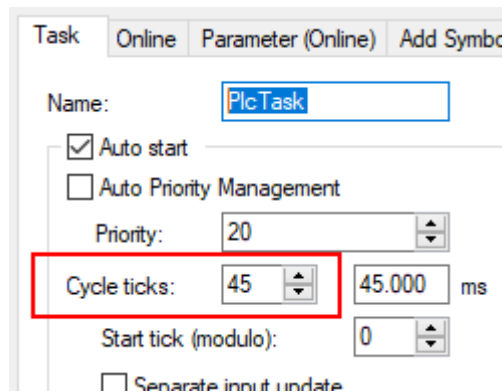
A restart of the operating system (on the target device) is required to apply the setting.

Tasks



Settings for the PLC task:

- Recommended cycle time: 45 ms



Task Online Parameter (Online) Add Symbol

Name: PlcTask

☒ Auto start

☐ Auto Priority Management

Priority: 20

Cycle ticks: 45 45.000 ms

Start tick (modulo): 0

☐ Separate input update

PLC

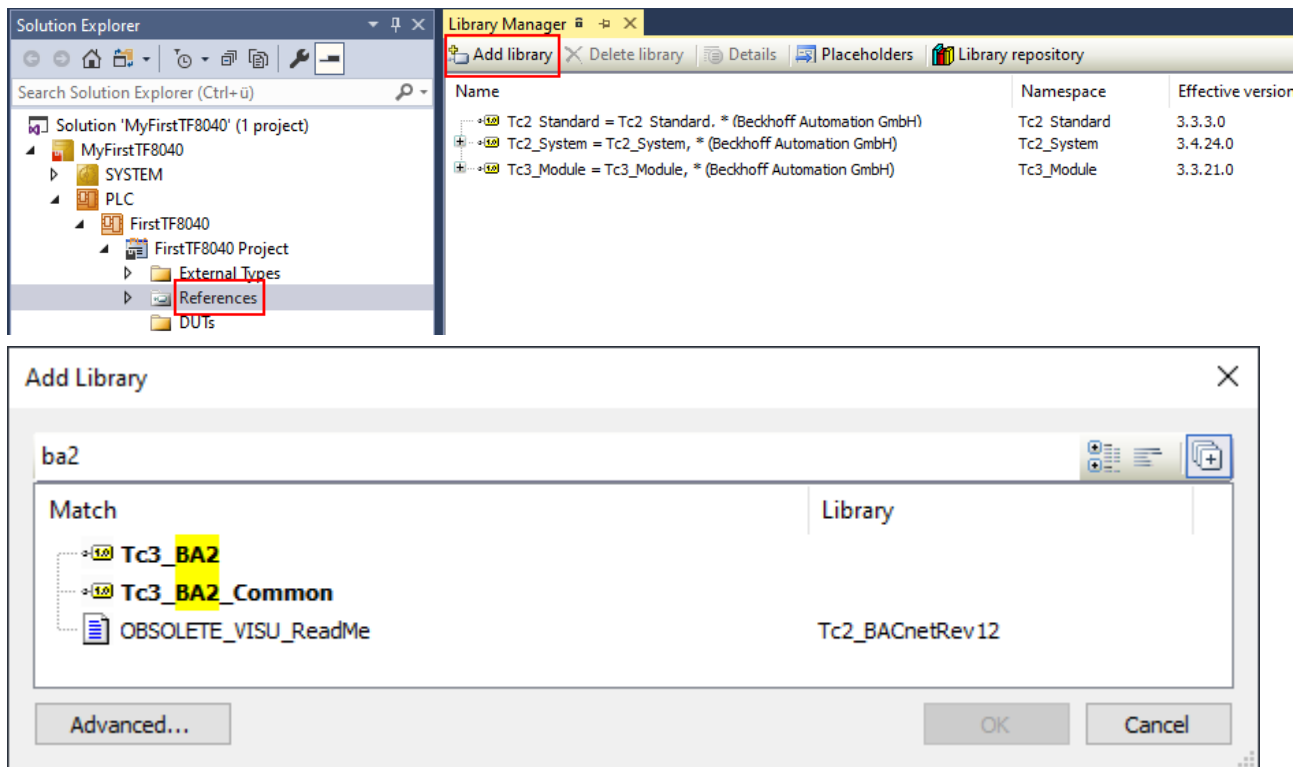


The settings described below are not necessary if a [PLC project template](#) [► 606] is used.

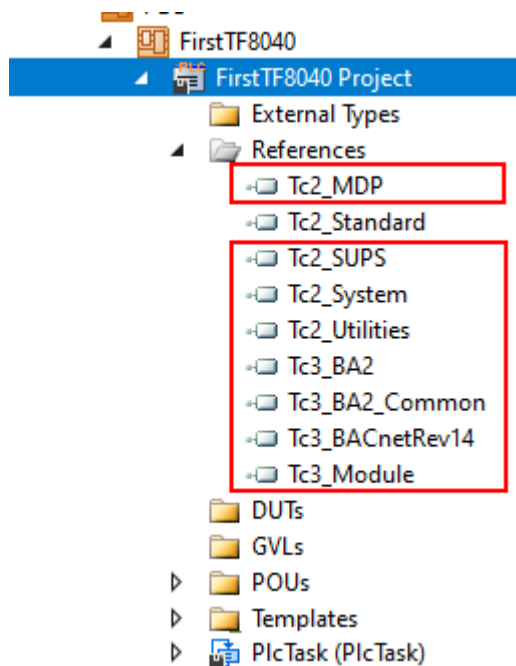
Libraries

If no template is used, the following libraries must be added to the PLC project.

- Tc3_BA2_Common
- Tc3_BACnetRev14
- Tc3_BA2



i



In the standard PLC-BA template [► 606] all necessary libraries are already loaded automatically.

I/O

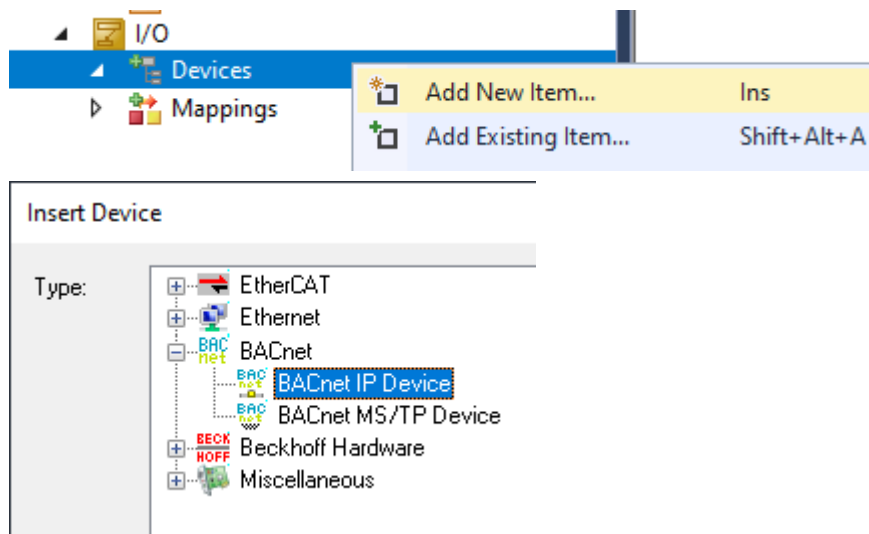
The procedure proposed here refers to the steps required to make the TF8040 function operational in combination with BACnet on the desired hardware.

i

Further steps for setting up the hardware are not be discussed in detail here.

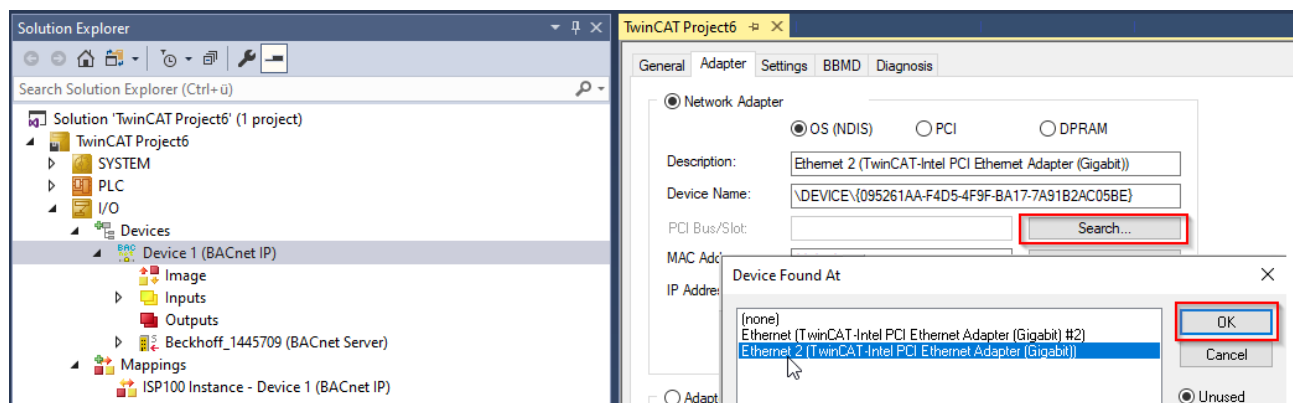
BACNet

- Add BACnet device:



- Select the appropriate network adapter:

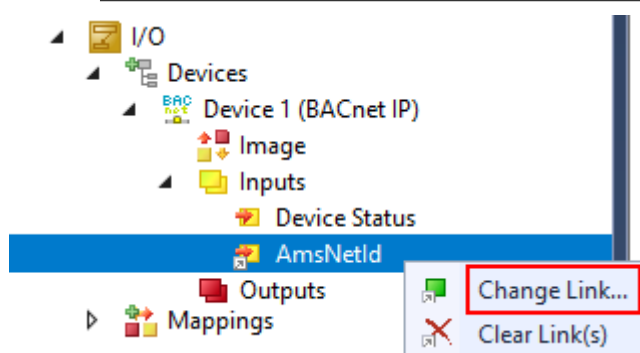
The adapter is set in the BACnet device under the Adapter tab:



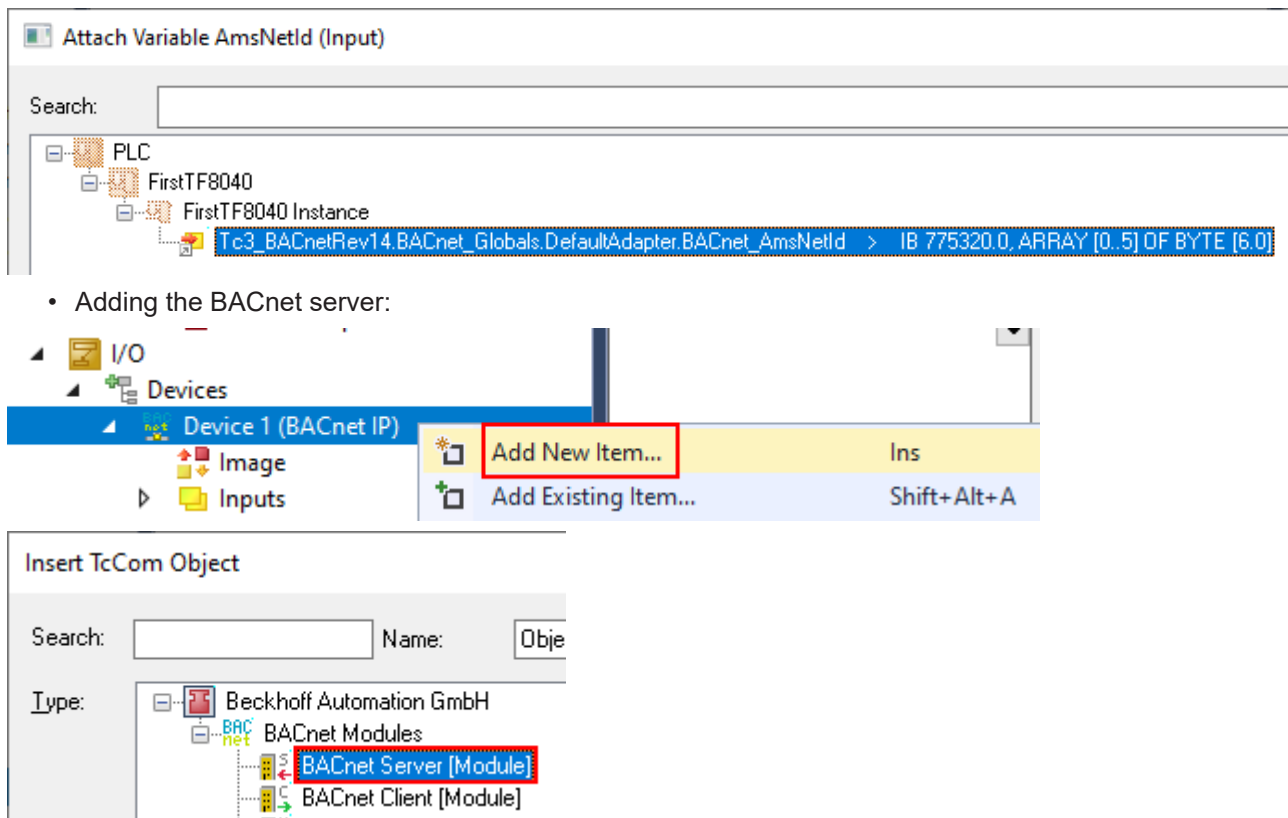
- Linking the BACnet adapter with the AMS NetID:



This step is only possible once the project has been compiled without errors.



Selection of the BACnet adapter to be used:



- Adding the BACnet server:

License

TwinCAT 3 standard licenses are tied to a unique system ID of a TwinCAT 3 license dongle (or IPC).

Standard licenses are chargeable: The license price depends on the hardware platform level.

More detailed information is described in the information system (see [Licensing](#)).

- Determine the [license status](#).
- To start it is possible to activate a [trial license](#). This unlocks all functions for 7 days.

Adapt the PLC template

Adapt the used [PLC project template](#) [[▶ 606](#)] according to the recommended [procedure](#) [[▶ 606](#)].

Continue

The PLC project is now set up and project planning can be started.

4.2 HMI

4.2.1 Starting a project

Description

The following section describes how to create and start a TcHmiBa project.



By using TcHmiBa project templates individual steps can already be prepared accordingly.

Procedure

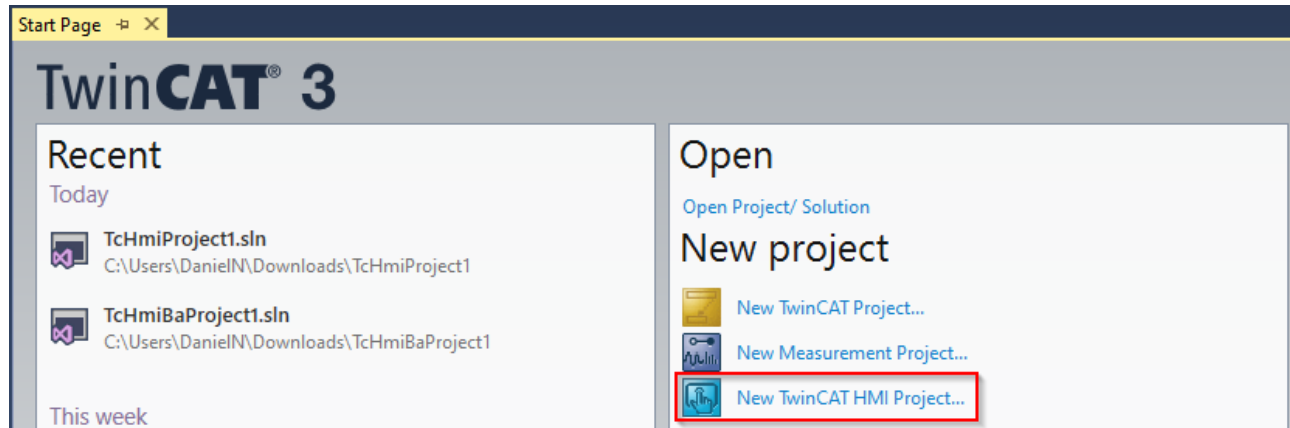
Installing the TwinCAT 3 HMI

For the creation of TcHmiBa projects the [TwinCAT 3 HMI Engineering](#) is required. Note the [system requirements](#).



Further information can be found in the documentation for [TwinCAT 3 HMI \(TE2000\)](#).

After installation, the project template for a standard HMI should be available in all development environments selected during setup (TcXaeShell, Visual Studio 2019, etc.).



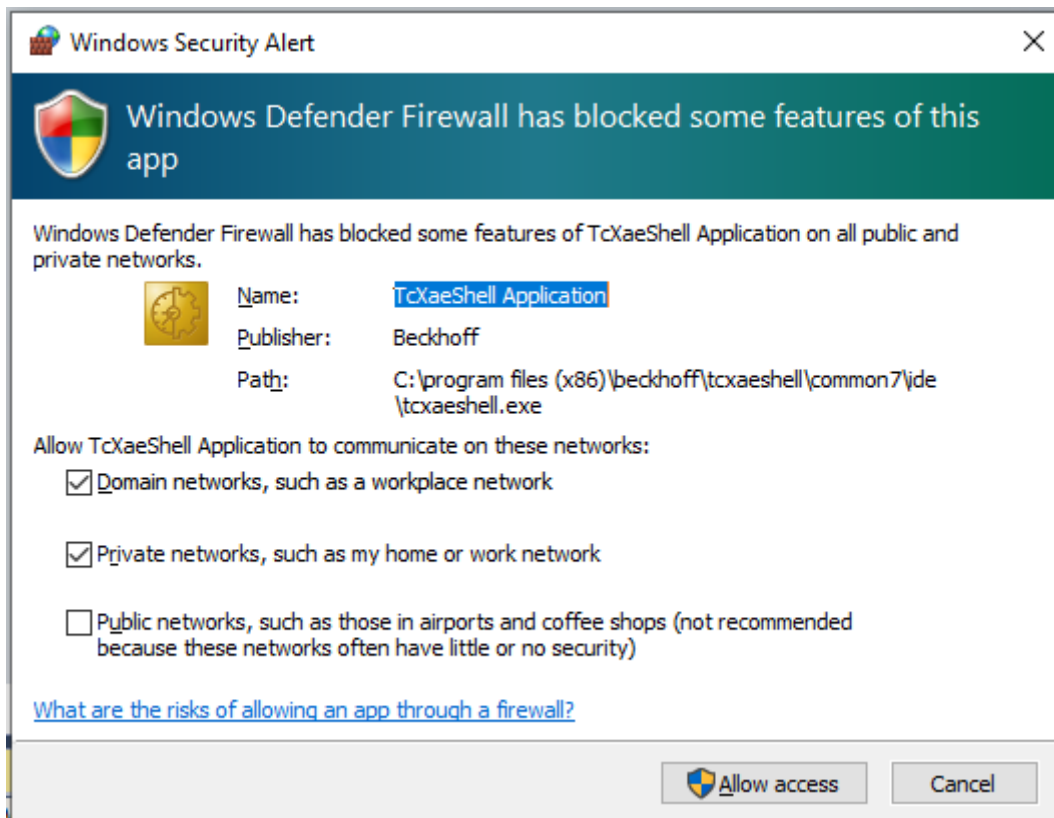
Installing TF8040

To use TcHmiBa projects, the TcHmi and TF8040 must be installed. Among other things, the TF8040 setup includes the NuGet packages, which will be discussed in more detail later.

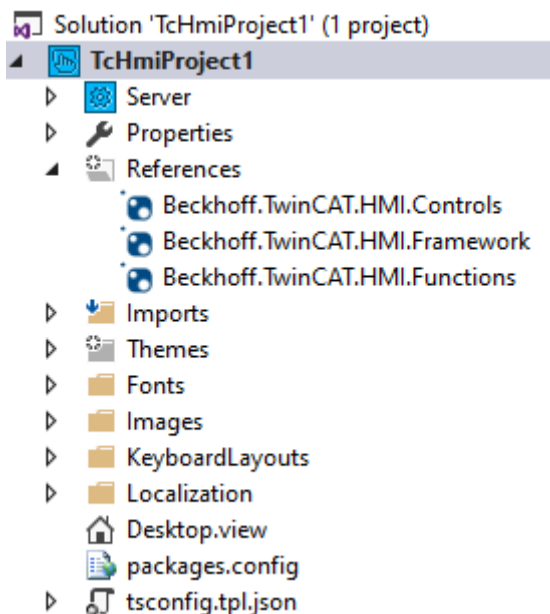
Create a new TcHmi project

Create a new **TwinCAT HMI** project on the start page or via File > New > Project under the category **TwinCAT HMI**.

During project creation there may be a Windows security alert because the engineering server is accessing the network. Please allow the access request accordingly.



The created project should now have the following structure:

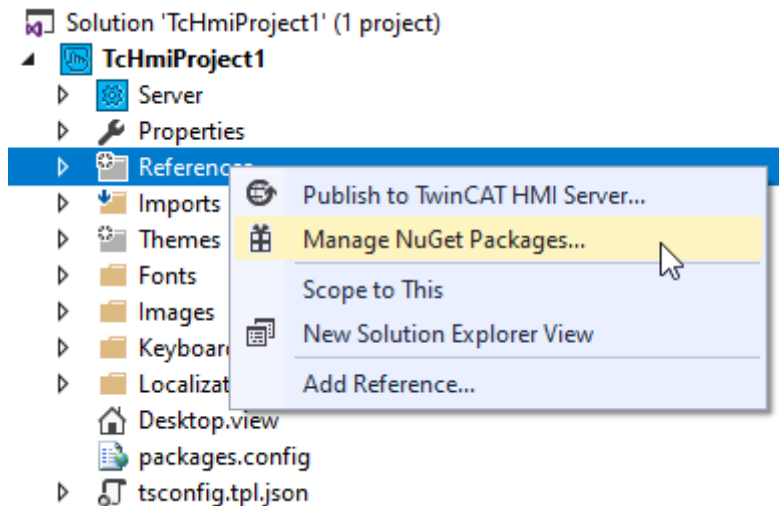


Installing NuGet packages

Specific controls for building automation are not included in the standard scope of the TcHmi toolbox.

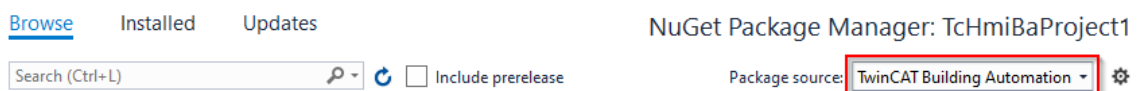
To be able to use them in the created project, the NuGet package **Beckhoff.TwinCAT.HMI.Controls** must be installed. The procedure for doing this is as follows:

- Open the NuGet package management



The correct package source **TwinCAT Building Automation** must be selected in the top right corner of the Package Manager.

The selection is only available if TF8040 including the TcHmi option (standard) has been installed.



- Install **Beckhoff.TwinCAT.HMI.BA.Controls**



Version: Latest stable Install

Options

Description

Controls for building automation HMIs.

Version:

Owner(s): Beckhoff Automation

Author(s): Beckhoff Automation

License: [View License](#)

Date published:

Project URL: https://infosys.beckhoff.com/english.php?content=../content/1033/tf8040_tc3_buildingautomation/index.html

Tags: Beckhoff TwinCAT HMI TcHmi Controls Control BA BaControls TcHmiBa TcHmiBaControls

Dependencies

native,Version=v1.12,Profile=tchmi

Beckhoff.TwinCAT.HMI.Controls (>=)

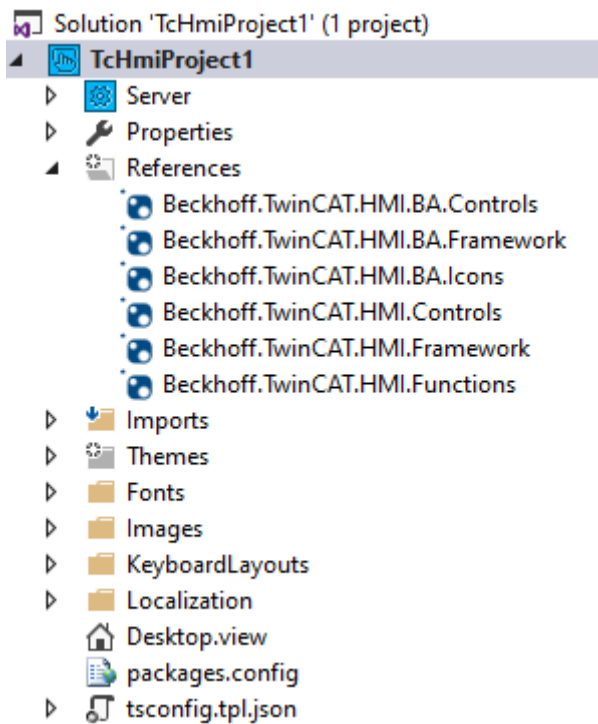
Beckhoff.TwinCAT.HMI.Framework (>=)

Beckhoff.TwinCAT.HMI.BA.Framework (>=)

Beckhoff.TwinCAT.HMI.BA.Icons (>=)

The packages [Beckhoff.TwinCAT.HMI.BA.Framework \[► 996\]](#) and [Beckhoff.TwinCAT.HMI.BA.Icons \[► 984\]](#) are also installed during the installation.

After installing the NuGet packages, the project tree should look like this.



If the *Desktop.view* and the toolbox are open, the following controls should now be available:

- ▲ BA | Building General
 - Pointer
 - BuildingInformation
 - FB_BA_WeatherStation_Draft
- ▲ BA | Common
 - Pointer
 - BulletPointList
 - Button
 - Calendar
 - Checkbox
 - ColorPicker
 - Combobox
 - DateTimeField
 - InputBox
 - Paginator
 - Scale
 - Slider
 - TabWindow
 - Textblock
 - Uilcon
- ▷ Common
- ▷ System
- ▲ BA | Plants
 - Pointer
 - Damper
 - FB_BA_AC_CoIT_02
 - FB_BA_AC_ErcPI_02
 - FB_BA_AC_ErcRot_01
 - FB_BA_AC_PreHtr
 - FB_BA_H_HtgCir01
 - Motor
 - Pump
 - SensorAnalog
 - Valve
- ▷ Shapes
- ▲ BA | Management
 - Pointer
 - EventList
 - ProjectNavigationTextual
 - Schedule
 - Trend
- ▲ BA | Room Automation
 - Pointer
 - HeatingCooling
 - Light
 - RoomControl
 - Sunblind
 - Window



Further information can be found in the respective [documentation](#) [► 822] of the individual controls.

Use

The available controls can be used to create visualizations as well as UserControls. The usage thus corresponds to the standard controls of the TcHmi.

Generic functions

If you want to use the TF8040 as a **complete solution** with all the advantages during engineering, please continue with the chapter [Generic functions](#) [► 63].

4.2.2 Generic HMI

Description

The following describes how the **generic functionalities** of TcHmiBa are used to minimize the development work for the HMI. As a prerequisite, see the chapter [Starting a project](#) [► 56].

Procedure

Please note the system requirements of the *BaSiteExtension*.

Furthermore, a PLC is required that was created with the libraries from TF8040 and is already active.



Description of required [terms](#) [► 28].

Preparing the server

Before starting to create the visualization with generic functions, the server must be prepared.

Installation BaSiteExtension

To support the generic functionalities it is necessary to install the NuGet package Beckhoff.TwinCAT.HMI.BA.BaSite.

NuGet Package Manager: TcHmiBaProject1

Package source: **TwinCAT Building Automation**

Search (Ctrl+L) ☐ Include prerelease

Beckhoff.TwinCAT.HMI.BA.BaSite by Beckhoff Automation
TwinCAT HMI is a development environment for web-based HMIs (Human Machine Interfaces). This packages provides an extension for the TwinCAT HMI Server that communicates with the SiteApi.

Beckhoff.TwinCAT.HMI.BA.Controls by Beckhoff Automation
Controls for building automation HMIs.

Beckhoff.TwinCAT.HMI.BA.Framework by Beckhoff Automation
Framework for building automation HMIs.

Beckhoff.TwinCAT.HMI.BA.Icons by Beckhoff Automation
Several icons for building automation HMIs.

Beckhoff.TwinCAT.HMI.BA.BaSite

Version:

Options

Description
TwinCAT HMI is a development environment for web-based HMIs (Human Machine Interfaces). This packages provides an extension for the TwinCAT HMI Server that communicates with the SiteApi.

Version:

Owner(s): Beckhoff Automation

Author(s): Beckhoff Automation

License: [View License](#)

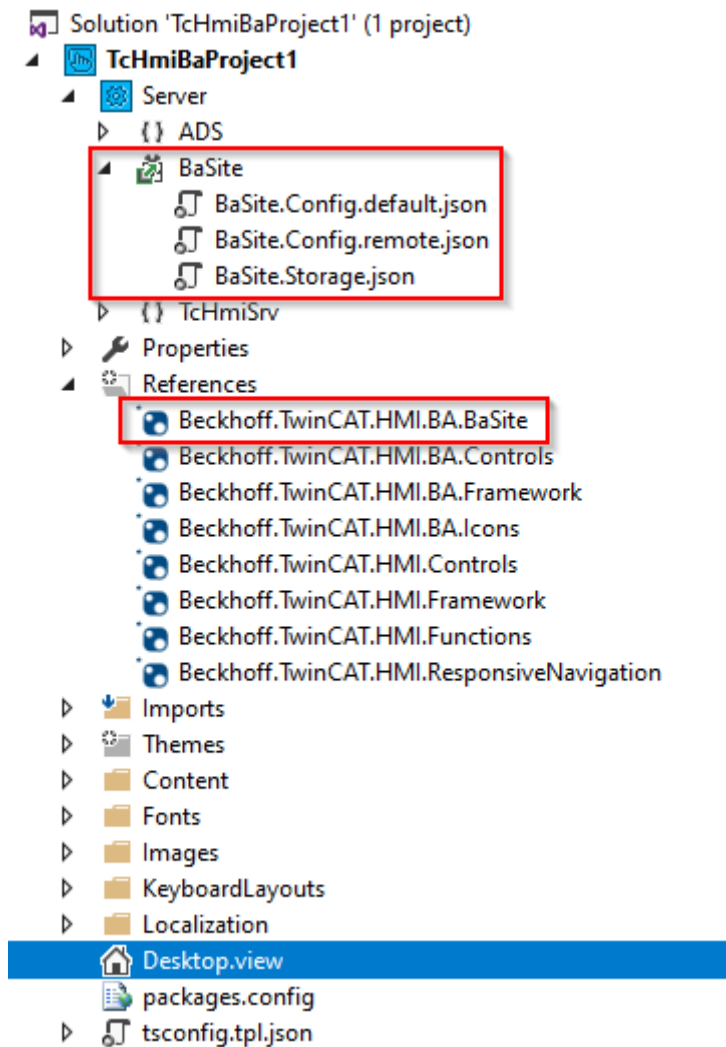
Date published: Friday, February 3, 2023 (2/3/2023)

Project URL: https://infosys.beckhoff.de/content/1033/te2000_tc3_hmi_engineering/index.html

Tags: Beckhoff TwinCAT HMI TcHmi Server Extension BA TcHmiBa BaSiteExtension BaSite TF8040

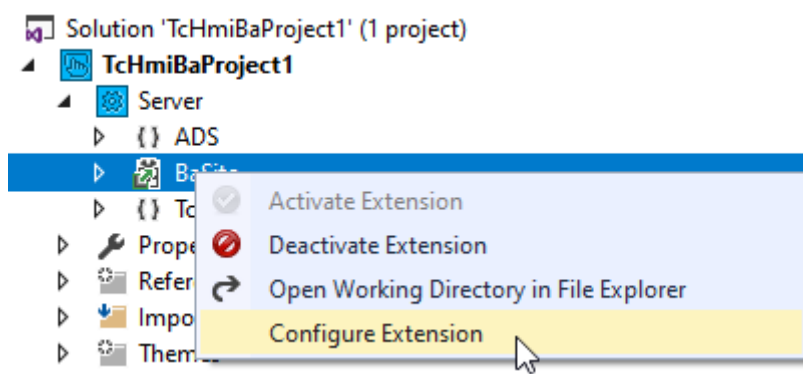
Dependencies
No dependencies

After installation the project tree should look like this:

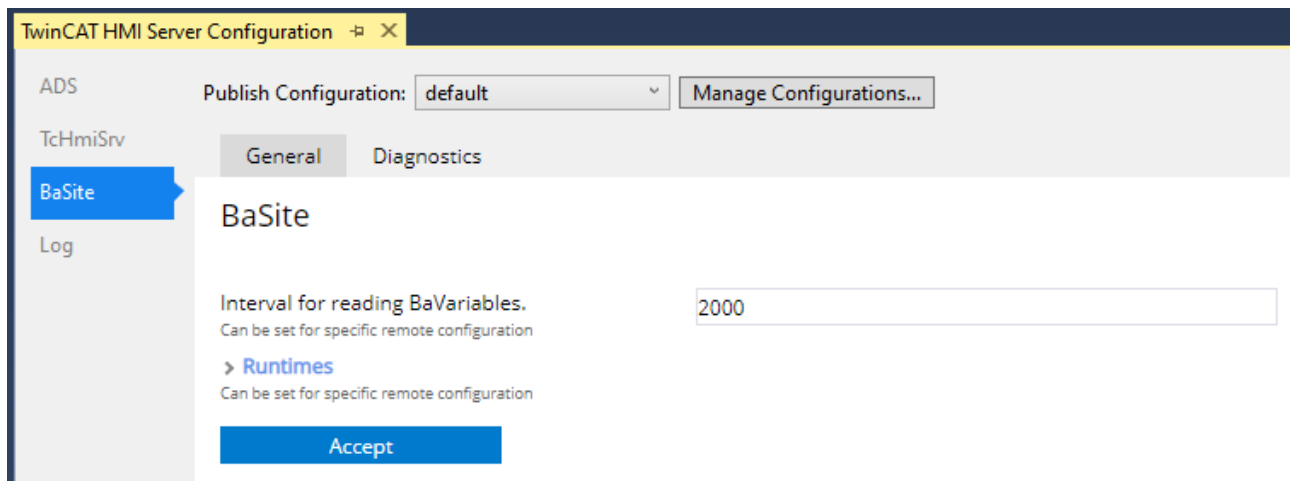


Configuration BaSiteExtension

Open configuration page of *BaSiteExtension*.



In the opened window all settings for the *BaSiteExtension* can be made.



For more information about configuration and functions, see the server extension documentation.

IFP01 is created by default in the *Runtimes* entry.

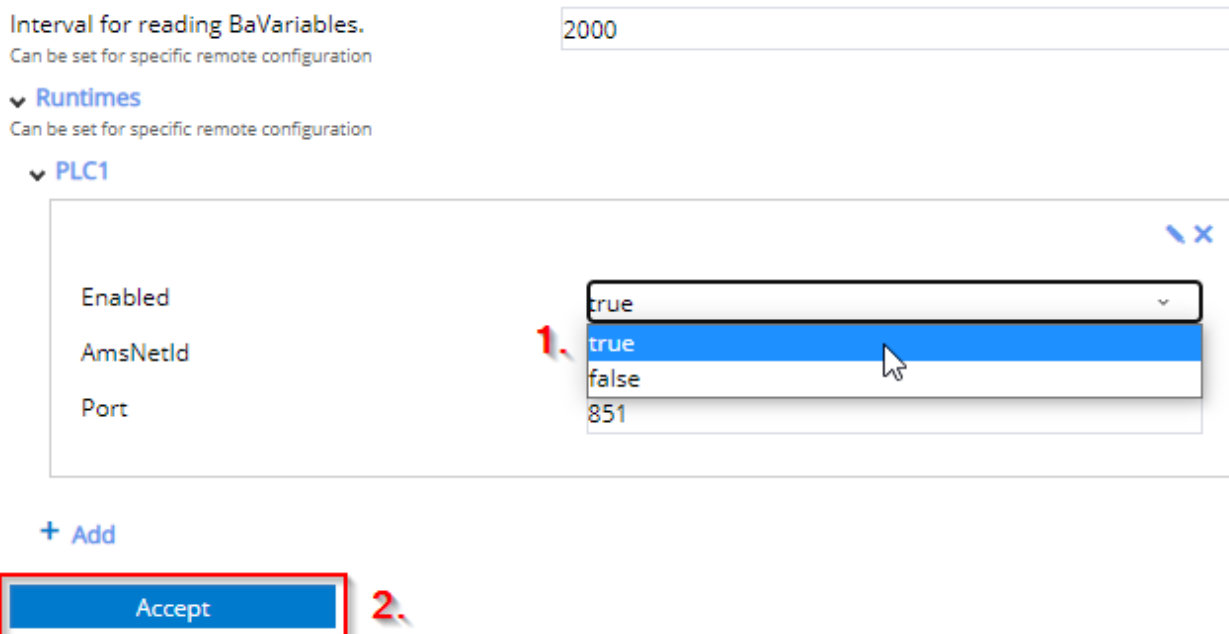
Enter the respective information for the active PLC under *AmsNetId* and *Port*. Enter the respective information for the active PLC under *AmsNetId* and *Port*.



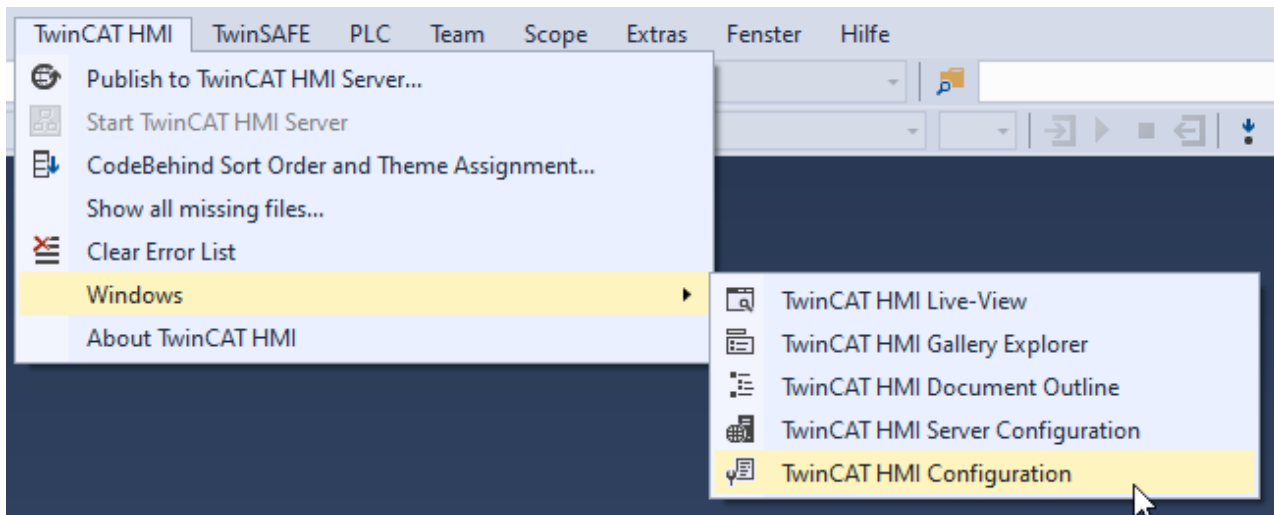
No change of the settings is required if the PLC is running locally on the development computer.

Afterwards the runtime has to be activated and the dialog has to be confirmed.

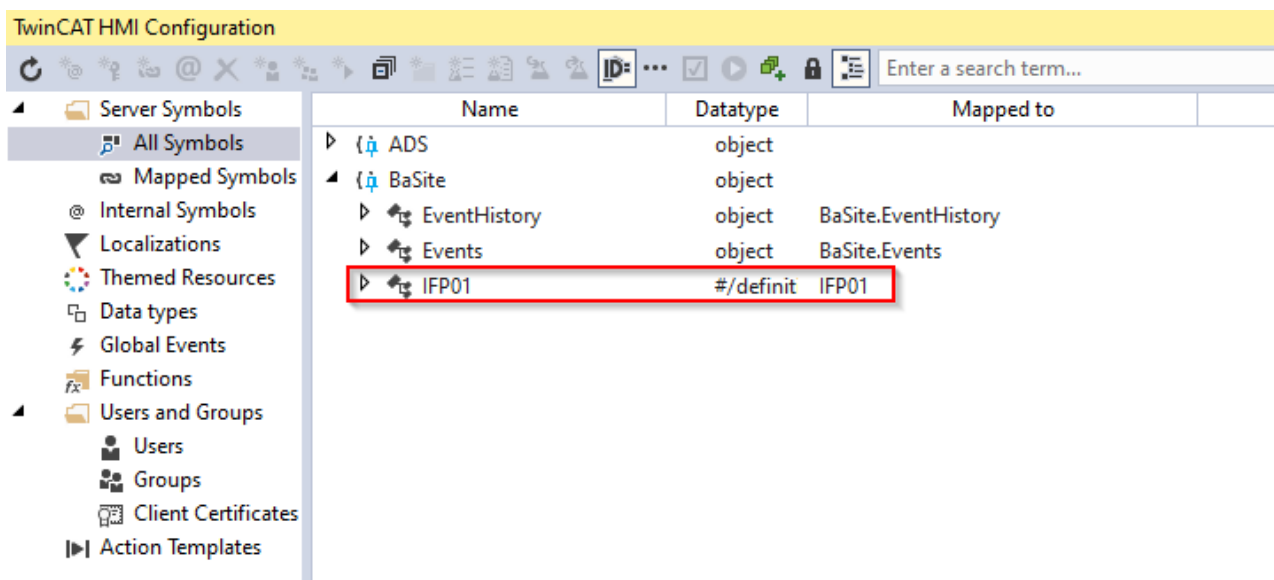
BaSite



Now in the *TwinCAT HMI Configuration* the PLC should have been successfully created in the server extension.

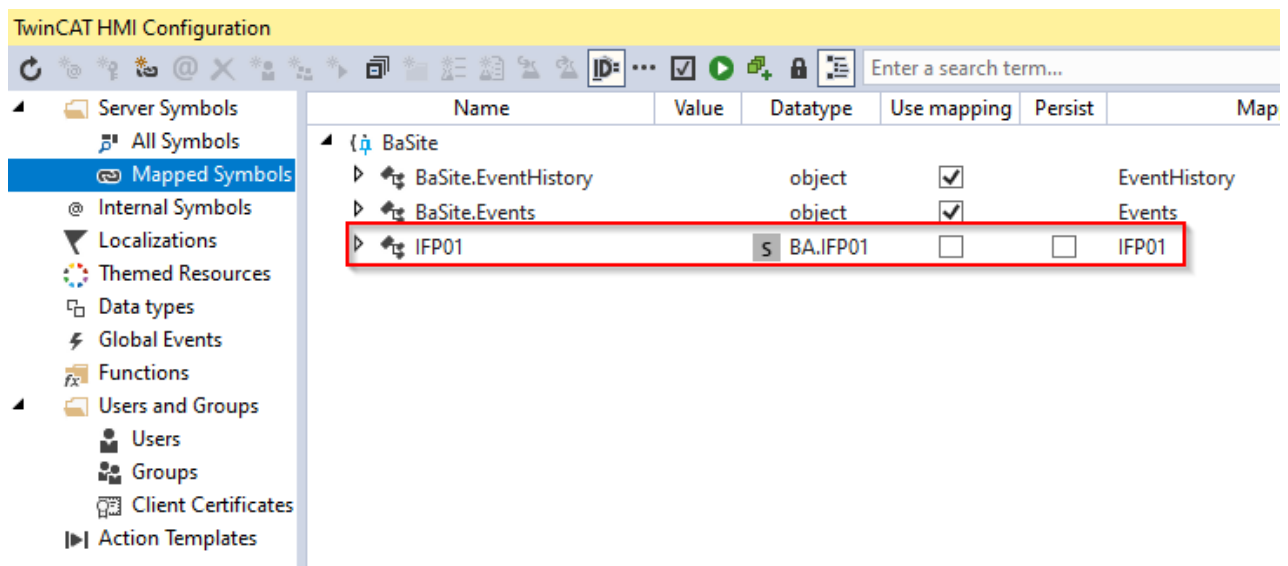


The runtime is then listed at *All Symbols* in the opened window.



If no HMI project is selected in the project tree, the display in the *TwinCAT HMI Configuration* remains empty.

A mapping for the runtime has also been created already. The mapping is listed under *Mapped Symbols*.



i Nature and source of the danger

Note the requirements for the necessary mappings for the generic functionalities.

The configuration of the server is now complete.

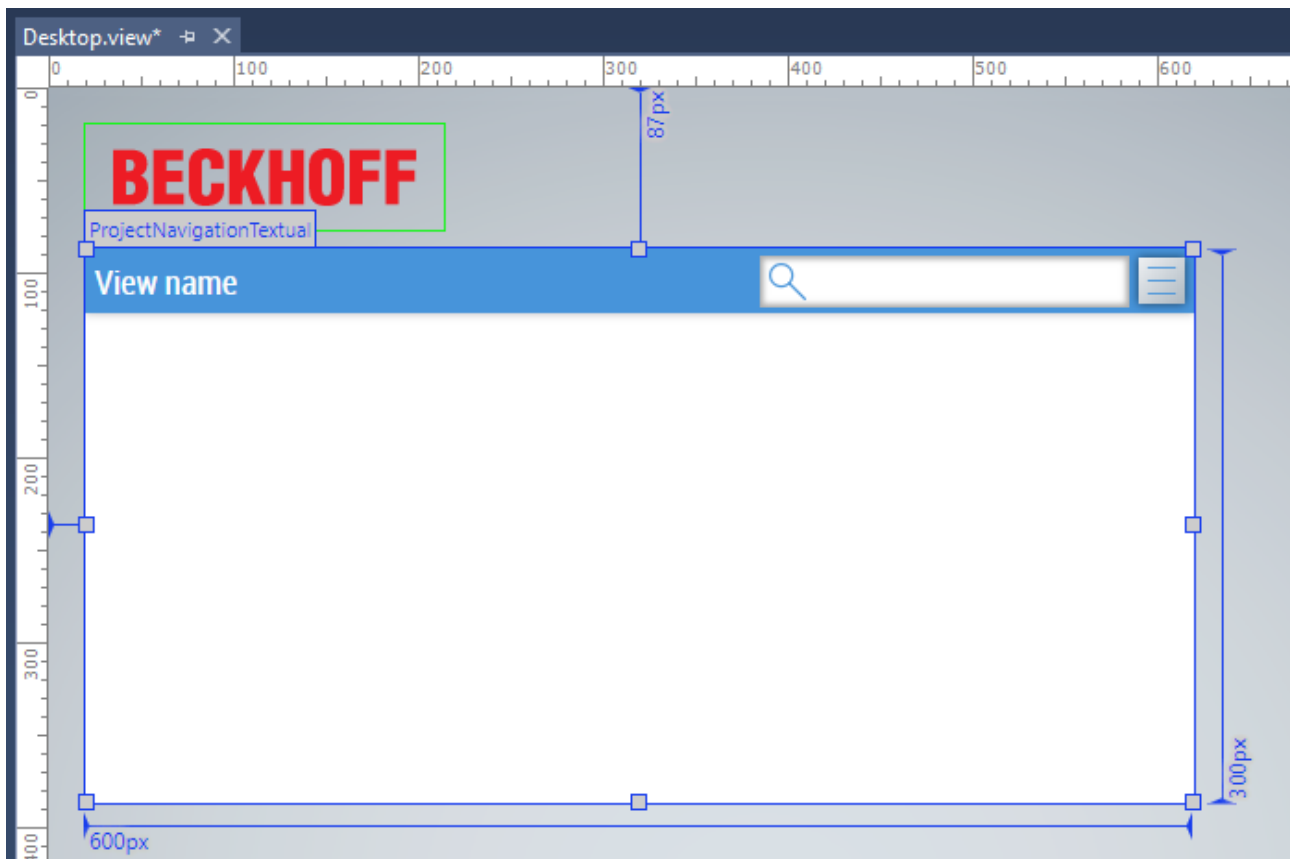
Using generic controls

The generic controls can only be used with the **TcHmiBaServerExtension**. A small selection is briefly described below.

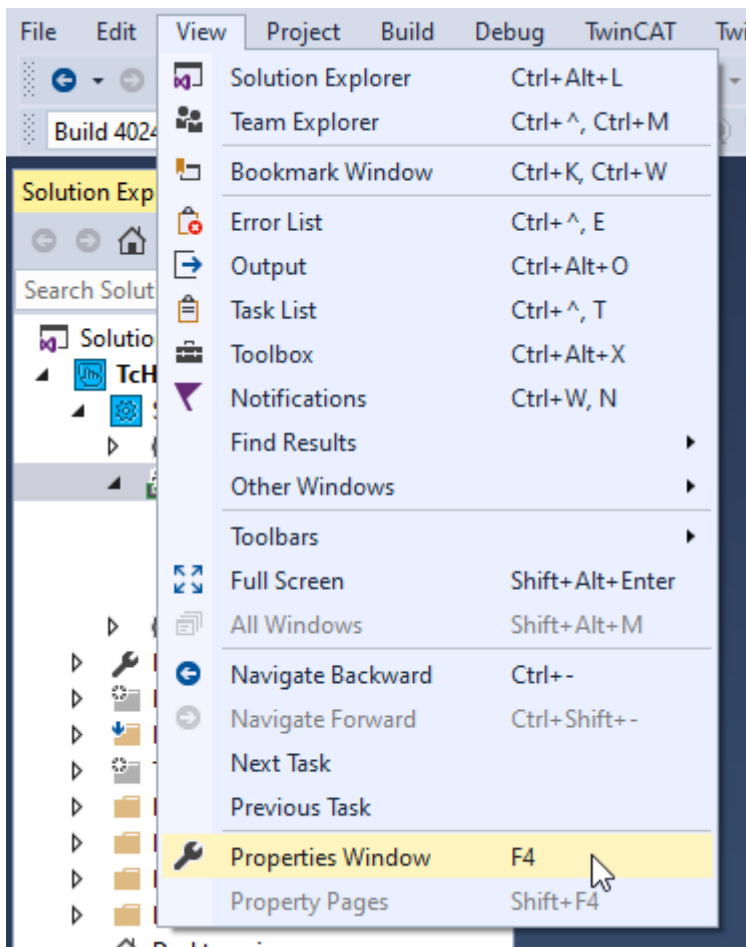
i Further information can be found in the respective documentation of the individual controls.

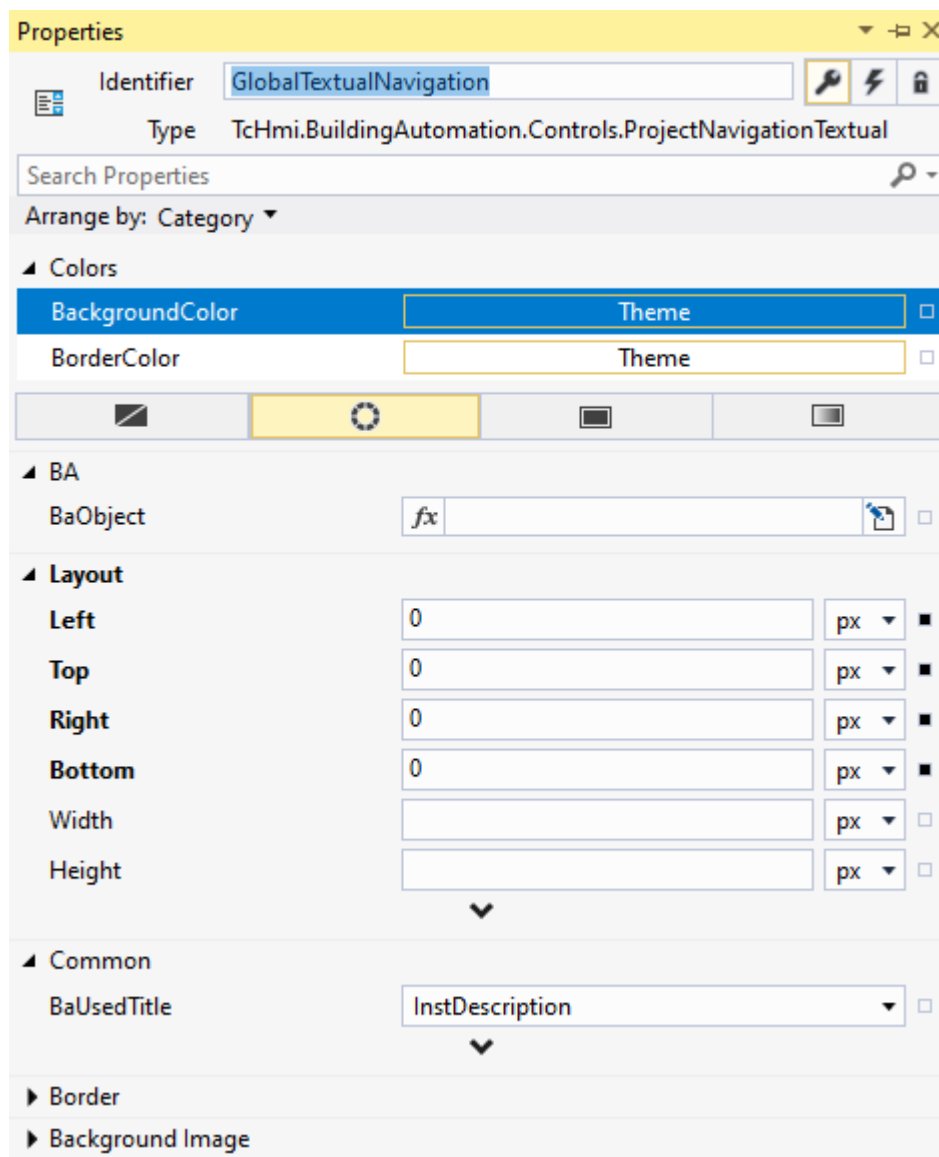
Project navigation

The quickest way to get started with the HMI is to use the ProjectNavigationTextual control. It is located in the **BA | General** category of the toolbox. If the control was placed on the *Desktop.view*, it should look like this:



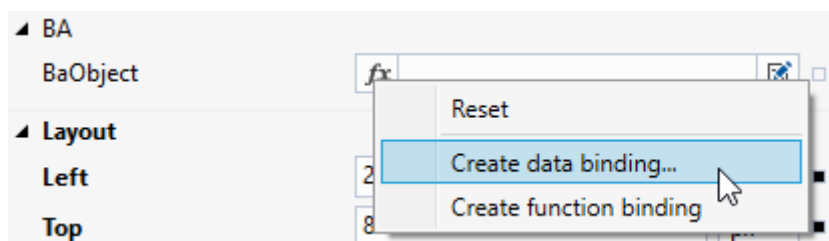
Open the Properties window.





The content of the Properties window always depends on the current selection. To see the properties of the project navigation, the control must be selected on the *Desktop.view*.

If the attribute *BaObject* is linked to the project structure (*Top* node) of the runtime, this control can be used to navigate through the entire project structure.



Select value for GlobalTextualNavigation.BaObject

✕

Quick search...

Server symbols Internal symbols Localizations Themed Resources **Mapped symbols** Controls

Name	Value	Datatype	Use mapping	Mapped from	Comment
↳ BaSite					
↳ BaSite.EventHistory		object	☒	EventHistory	
↳ BaSite.Events		object	☒	Events	
↳ IFP01		BA.IFP01	☐	IFP01	
↳ Top		BA.IFP01.ProjectStructure	☐	Top	

Settings

Symbol expression

Refresh Collapse All Unmap Symbol

2 OK Cancel

BA

BaObject

S IFP01::Top

Solution 'TcHmiProject1' (1 project)

TcHmiProject1

- Server
- Properties
- References
- Imports
- Themes
- Fonts
- Images
- KeyboardLayouts
- Localization

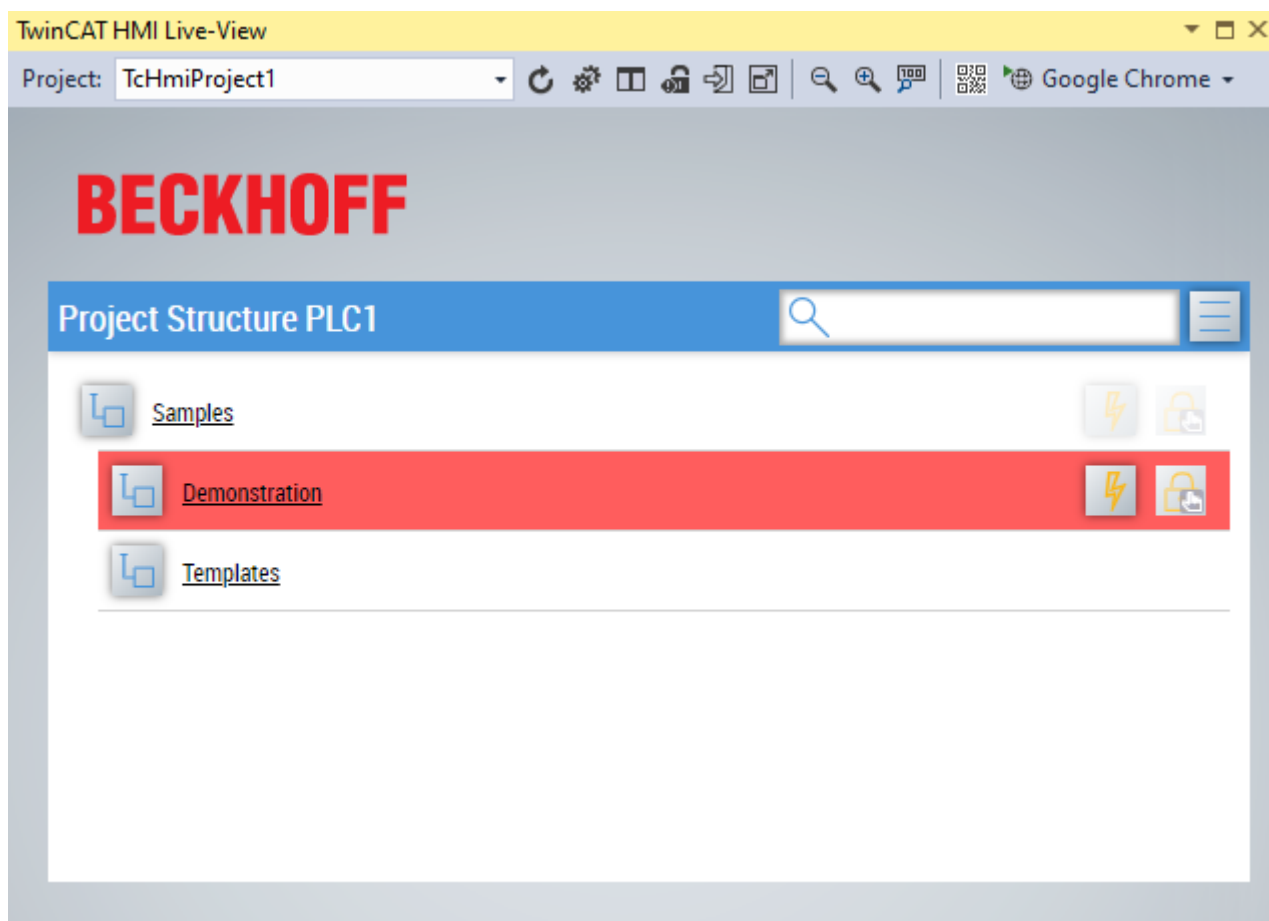
Desktop.view

packages.config

tsconfig.tpl.json

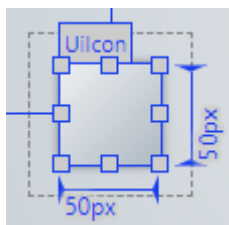
- Open
- Open With...
- Scope to This
- New Solution Explorer View
- Cut Ctrl+X
- Copy Ctrl+C
- Delete Del
- Rename
- Properties Alt+Enter
- Show in Live-View...**
- Publish this to TwinCAT HMI Server...
- Set as Start View

The Live-View should now look like this:



Uilcon

The Uilcon [► 859] can be used for various applications. It is also located in the **BA | General** category of the toolbox.

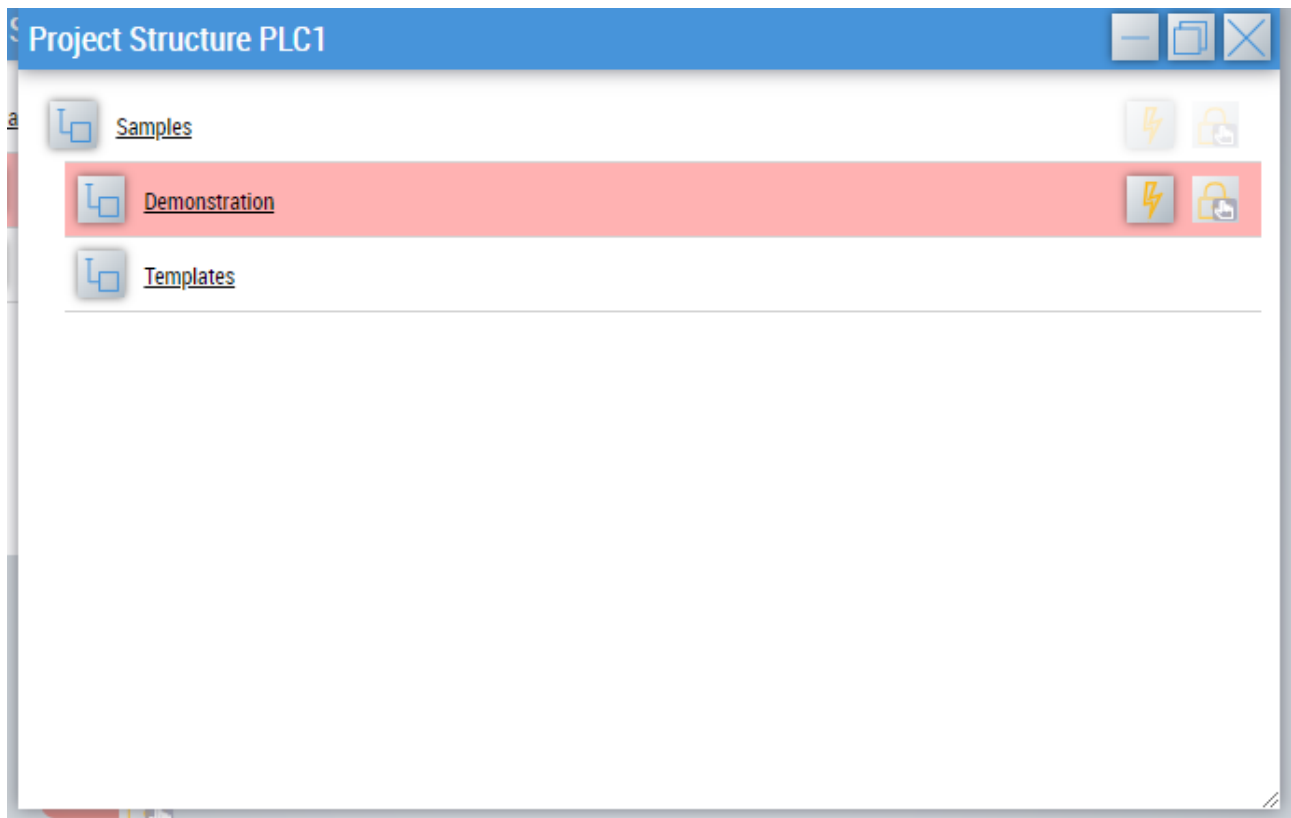


Like all controls from TcHmiBa, the *Uilcon* has the attribute *BaObject*. Any *BaView* or any *BaObject* can be linked to this. Again, the *ProjectStructure* of the runtime (see above) serves as an example.

In Live View, the *Uilcon* then looks like this:



It displays the active events of the linked view/object and enables opening of a window that contains the generic navigation from above.



The same functionalities are thus available from this window.

5 Examples

Examples of all features from TF8040.

Downloads

- TF8040 https://infosys.beckhoff.com/content/1033/TF8040_TC3_BuildingAutomation/Resources/13916972683/.zip

Contains

- Samples of PLC ([concept](#) [► 75], [templates](#) [► 88])
- Samples of HMI ([concept](#) [► 82], [templates](#) [► 89])

Configuration

PLC

Individual settings within the *TF8040-Concept-Samples-PLC-Solution* must be adapted to the hardware used.

The settings affect the Project settings and the I/O.



All necessary steps are described in [Starting a project](#) [► 50].

Individual settings that have already been made do not need to be taken into account any further.

HMI

Preparing the PLC

To run the HMI, the PLC must be configured and started.

Installing the TwinCAT HMI

To load the sample project, the [TwinCAT 3 HMI Engineering](#) must first be downloaded and installed.



Further information on the required steps can be found in the documentation for the TwinCAT 3 HMI Engineering in the section [Installation](#).

Download

Once the PLC is activated and running, the sample solution can be opened.

License development system

In order for the HMI samples to be executable, the TC3 HMI server license of the TF2000 must be licensed on the target system.

Server configuration

No further configuration is necessary if the runtime is activated on the PC on which the HMI project is also started.

If not, the configuration must be adjusted. The procedure for this is described in the tutorial [Generic HMI](#) [► 63].



Please note the general comments.

Further information

- PLC programming

5.1 Concept examples

The samples in this package show the concept of TF8040 on the part of the PLC and HMI.

The chapter presents the approaches for creating a TF8040 project. Different BaObject types and the procedure for implementing an HMI are also shown.

The samples reference the [downloads](#) [► 74].

5.1.1 PLC

The following chapters refer to the program samples within the downloads.

5.1.1.1 Demonstration

5.1.1.1.1 Objects

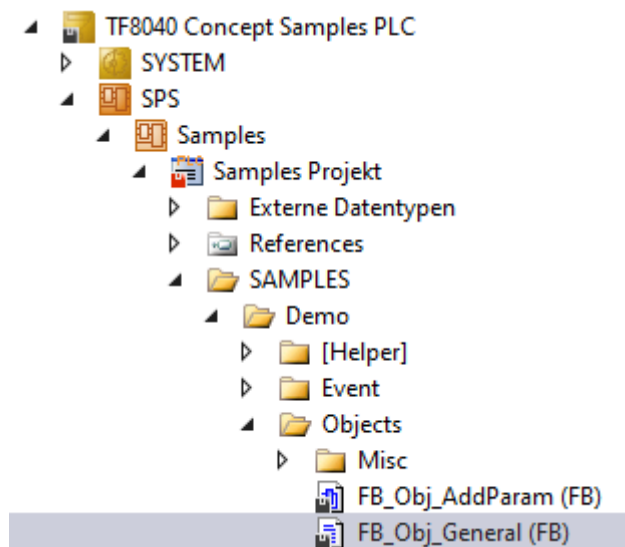
This chapter provides information on how to declare the available [object types](#) [► 125].

Furthermore, it provides application samples for initializing objects using the provided [downloads](#) [► 74].

Graphical representation of the location where the application samples from the *FB_Obj_General* can be found.

ConceptSamples/PLC/Samples/SamplesProjekt/SAMPLES/Demo/Objects

Graphical example for display in the project:



Declaration of object arrays

The declaration of multiple objects as an array is possible.

The sample shows how the parameters of all trend objects are initialized by means of declaration.


```

3  VAR CONSTANT
4      TREND_COUNT      : INT      := 5;
5  END_VAR
6  VAR
7      TrendView        : FB_BA_View := (
8          iParent
9          sDescription
10         );
11      Trends           : ARRAY[1..TREND_COUNT] OF FB_BA_Trend := [TREND_COUNT((
12          iParent
13          iLabel
14          nEventClassID
15          stStartTime
16          eLoggingType
17         ))];
18
19 TrendView();
20 FOR i := 1 TO TREND_COUNT DO
21     Trends[i]();
22 END_FOR

```

5.1.1.1.2 Reference

5.1.1.1.2.1 Publisher and Subscriber

Contains examples around the concept of referencing using publish and subscribe.

All examples also include:

- [Publisher \[► 431\]](#)
- [Object-related subscribers \[► 433\]](#)
- [Type-less subscribers \[► 433\]](#)
- [Typed subscribers \[► 434\]](#)

Example

Weather station

The distribution of information (using typed [publishers \[► 431\]](#) and [subscribers \[► 431\]](#)) is illustrated using the example of a weather station.

Use case

Building 1 has a weather station and distributes information (e.g. outdoor temperature) to all rooms in several buildings.



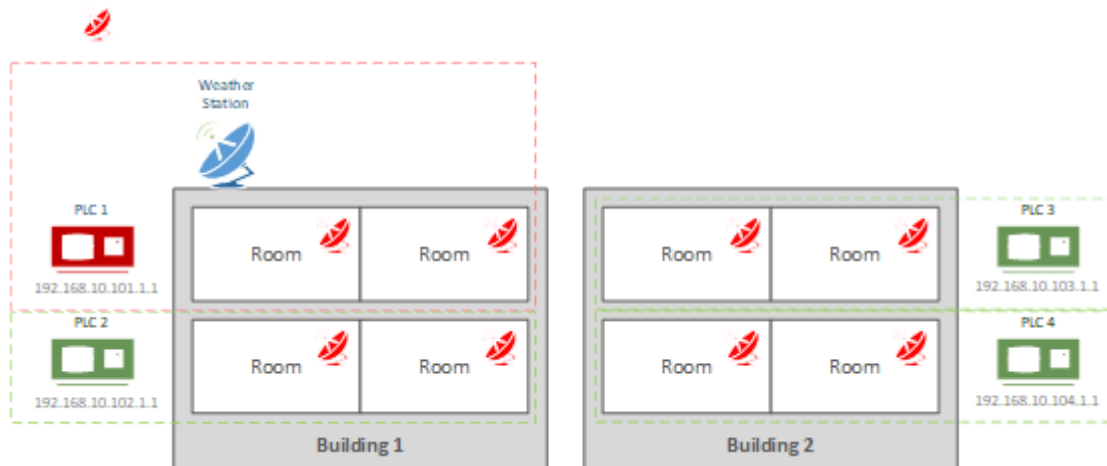
Structure

- **4 PLCs** control rooms on a total of 4 floors (2 x ground floor, 2 x upper floor).

- The **weather station** is on PLC 1.



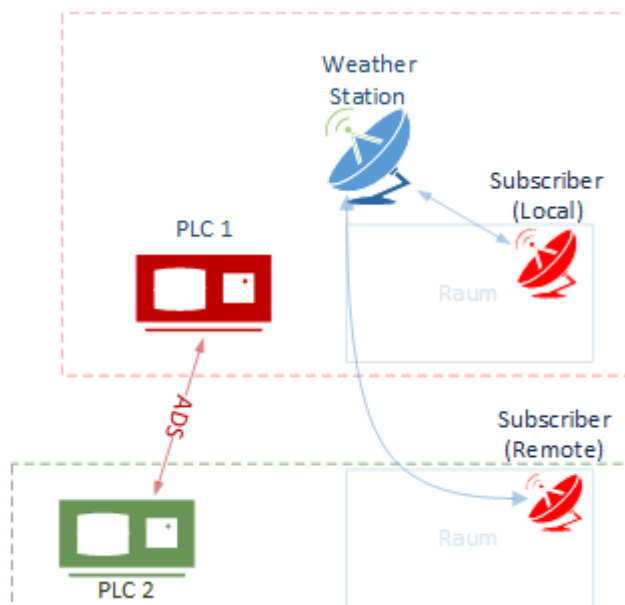
- Each room is implemented by one instance of a room template. The access to data of the weather station is established by means of a Subscriber [► 431].



Principle of operation

PLC 1 makes weather data available (publishing) through the typed publisher "*FB_BA_MyWeatherStation*" (physically connected weather station).

The rooms on the different PLCs each contain the typed subscriber of type "*FB_BA_MyWeatherStationSub*" in order to read the weather data (corresponding to local and remote subscriptions).



Implementation

- On PLC 1 the weather station is declared at any position of the PLC program:

```

4  VAR
5      MyWthSt      : FB_BA_MyWeatherStation    := (bEnPublish:= TRUE);
6  END_VAR

```

- The subject information required to initialize the [subscriber \[► 431\]](#) must exist on **all PLCs** (but usually in the [site GVL \[► 435\]](#)):

```

8 | VAR CONSTANT
9 |     WthSt_Target      : T_BA_MedString      := '192.168.10.101.1.1';
10 |     WthSt_Subject     : T_BA_MedString      := 'MyWthSt';
11 | END_VAR

```

- The declaration of the subscriber in the room template of the **respective PLC** would then look like the following:

```

WthStSub    : FB_BA_MyWeatherStationSub := (
                sSubject                := WthSt_Subject,
                sTarget                  := WthSt_Target
            );

```

- **PLC 1**

As the weather data are stored on the local PLC, the data can be read directly (within the PLC) ([Local Subscriptions \[► 431\]](#)).

- **PLC 2**

As the weather data is stored on a remote PLC, the data is read from PLC 1 via ADS ([Remote Subscriptions \[► 432\]](#)).

5.1.1.1.2.1.1 Local

This documentation describes the use of local subscribers located on the device.

Examples

Providing and reading

Providing information via [publisher \[► 431\]](#) and reading provided information via [subscriber \[► 431\]](#) (on the same device) is illustrated by implementing a pair of [publisher \[► 431\]](#) and [subscriber \[► 431\]](#) and configuring them accordingly.

Configuration

- [Enable publishing \[► 431\]](#)
- [Setting up a subscription \[► 431\]](#)

Object-related

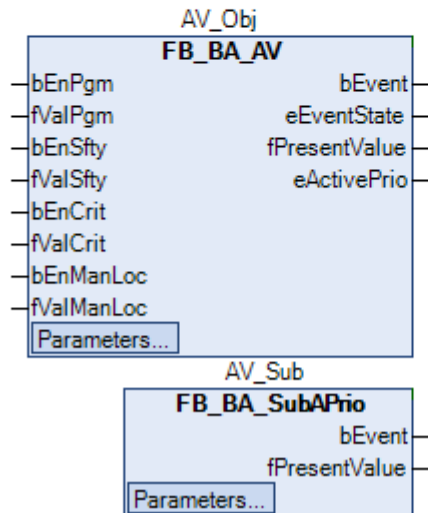
For all available object types, a sample is provided to illustrate how object type-dependent information (e.g. current value or event status) is provided and read.

Sample: publisher and subscriber of an "Analog Value" Object

```

18     AVOp_Sub      : FB_BA_SubAV_Op      := (sSubject:='AVOp_Obj');
19     AVOp_Obj      : FB_BA_AV_Op := (
20         iParent      := A,
21         iProfile      := MyProfiles.AVOp_Prfl,
22         bEnPublish    := TRUE,
23         fValueRm      := 10
24     );

```



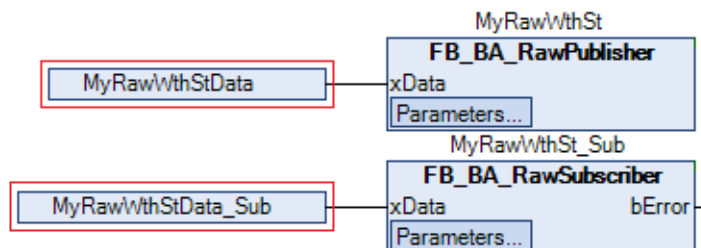
Typeless

The sample illustrates how any information (e.g. raw data in the form of a structure) is provided and read. Sample: publisher and subscriber for raw data of a weather station:

```

164     MyRawWthStData      : ST_BA_MyWeatherData;
165     MyRawWthSt           : FB_BA_RawPublisher;
166
167     MyRawWthStData_Sub   : ST_BA_MyWeatherData;
168     MyRawWthSt_Sub       : FB_BA_RawSubscriber := (sSubject:='MyRawWthSt');

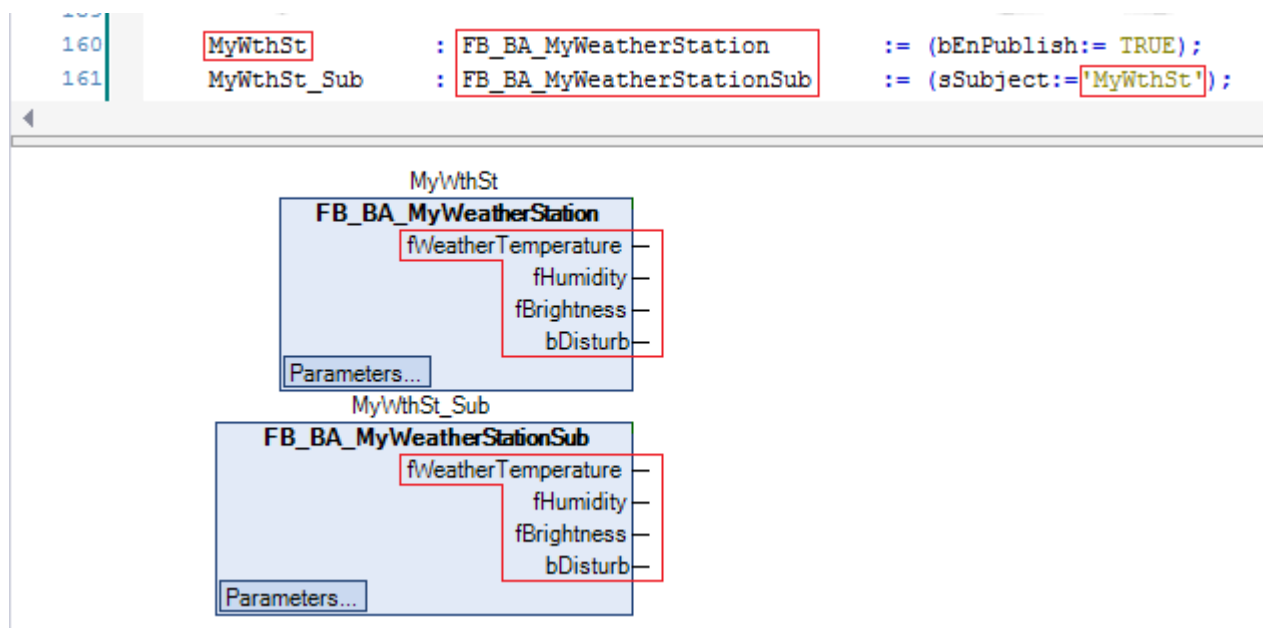
```



Typed

The sample illustrates how typed information [► 434] is provided and read by implementing custom derived publisher [► 431] and subscriber [► 431].

Sample: publisher and subscriber of the user-defined type of a weather station:

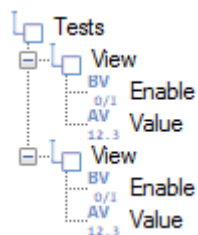


Tests

Subject path

The sample illustrates the relationship between subject information and references.

The code snippet below generates the following project structure.



Two view objects, each with two further subobjects:

```

178   Nested1a      : FB_NestedView := (
179       View      := (iParent := Tests),
180       Value     := (bEnPublish := TRUE)
181   );
182   Nested1b      : FB_NestedView := (
183       View      := (iParent := Tests),
184       Value     := (bEnPublish := TRUE)
185   );
186
187   Nested1a_AV_Sub : FB_BA_SubAV_Op := (
188       sSubject   := 'Nested1a.Value'
189   );

```

```

1  FUNCTION_BLOCK FB_NestedView
2  VAR_INPUT CONSTANT
3      View      : FB_BA_View;
4      Enable    : FB_BA_BV_Op := (
5          iParent      := View,
6          iProfile     := MyProfiles.BVOp_Prfl,
7          bValueRm     := FALSE
8      );
9      Value     : FB_BA_AV_Op := (
10         iParent      := View,
11         iProfile     := MyProfiles.AVOp_Prfl,
12         fValueRm     := 0.0
13     );
14 END_VAR

```

Since two objects with the subject *Value* exist, this information alone is no longer sufficient to uniquely identify one of these two objects!

```

Nested1a_AV_Sub : FB_BA_SubAV_Op := (
    sSubject      := 'Value'
);

```

The uniqueness is established by additionally specifying subject information of the parent.

```

Nested1a_AV_Sub : FB_BA_SubAV_Op := (
    sSubject      := 'Nested1a.Value'
);

```

5.1.1.1.2.1.2 Remote

This documentation describes how to use remote subscribers on other devices.

Examples

The samples described here refer entirely to the [samples for local subscribers](#) [► 78]:

For each local sample, there is an analogous sample of a remote subscriber.

This document is intended to illustrate the difference between [local](#) [► 431] and [remote](#) [► 432] subscribers.

Communication

[Setting up the routes](#) [► 432] is a separate step.

NOTICE

Communication

Communication between [publishers](#) [► 431] and [subscribers](#) [► 431] (on different devices) can only work if the ADS routes of both devices are set up correctly.

Application

The targets, of the [subscribers](#) [► 431] illustrated in these samples are initialized with predefined AMS NetIDs.

Here, the placeholder variable *sDevice1_NetID* is used to initialize one half, while the other half is initialized with the placeholder variable *sDevice2_NetID*:

```

36     AVOp_Sub      : FB_BA_SubAV_Op := (
37         sSubject      := 'AVOp_Obj',
38         sTarget       := sDevice1_NetID,
39         tReadInterval := T#5S
40     );
41     AI_Sub        : FB_BA_SubAI := (
42         sSubject      := 'AI_Obj',
43         sTarget       := sDevice2_NetID,
44         tReadInterval := T#5S
45     );

```

By commenting in and out and adjusting these placeholder variables accordingly, the sample program can be adapted to your own test environment:

```

7  VAR CONSTANT
8  {region 'Test Settings'}
9      // Use one (remote) device:
10     // sDevice1_NetID      : T_BA_MedString      := '192.168.10.100.1.1';
11     // sDevice2_NetID      : T_BA_MedString      := '192.168.10.100.1.1';
12
13     // Use one (loopback) device:
14     // sDevice1_NetID      : T_BA_MedString      := '127.0.0.1.1.1';
15     // sDevice2_NetID      : T_BA_MedString      := '127.0.0.1.1.1';
16
17     // Use two (remote and loopback) devices:
18     // sDevice1_NetID      : T_BA_MedString      := '192.168.2.200.1.1';
19     // sDevice2_NetID      : T_BA_MedString      := '127.0.0.1.1.1';
20
21     // Use two (remote) devices:
22     // sDevice1_NetID      : T_BA_MedString      := '192.168.10.100.1.1';
23     // sDevice2_NetID      : T_BA_MedString      := '192.168.2.200.1.1';
24 {endregion}
25 END_VAR

```

The following **test settings** are available for selection:

- A *remote* device
- A *loopback* device
- A *remote* and a *loopback* device
- Two *remote* devices

5.1.1.1.3 Optional extensions

Parameter

Application of [FB_BA_Param](#) [► 124] to programmatically adjust [global parameters](#) [► 223].

Event Subscriber

Application of Event Subscribers to subscribe to generic events.

5.1.2 HMI

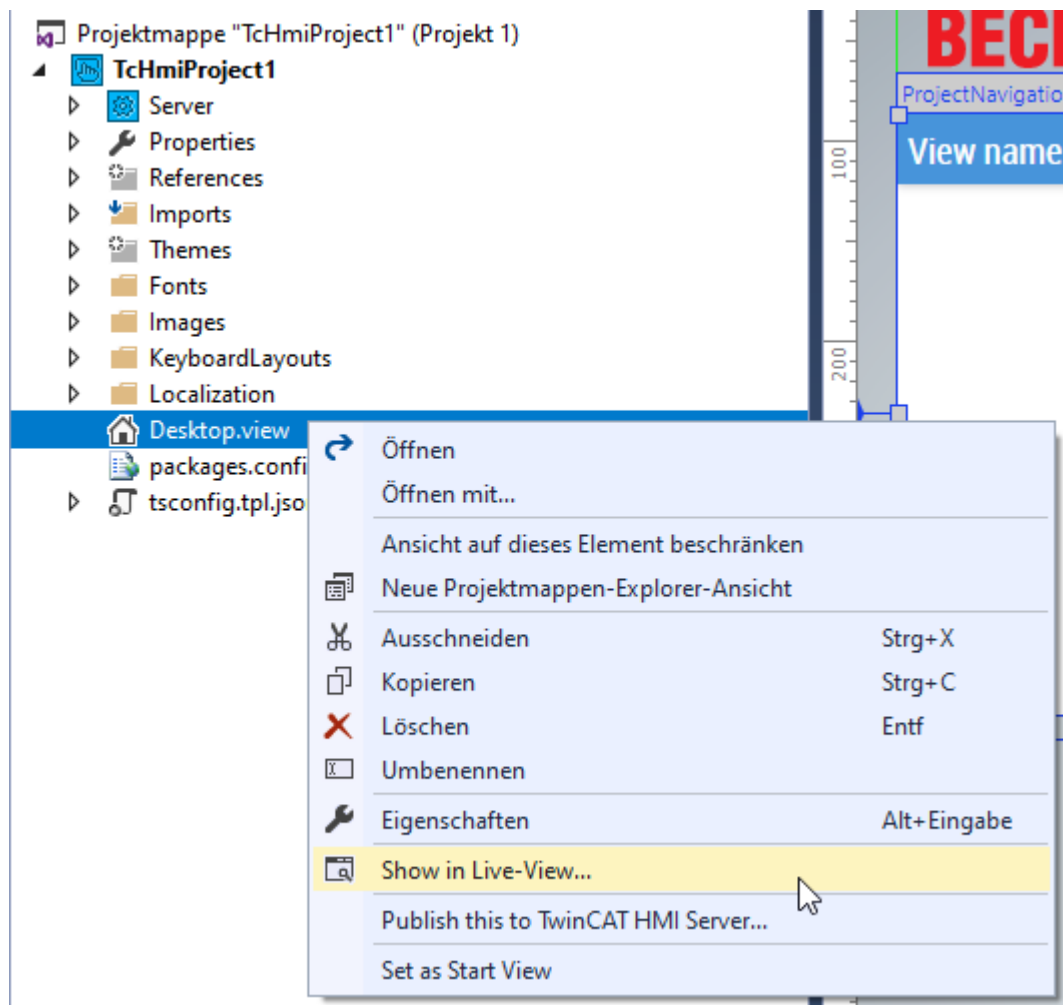
Explanation of the sample project *TF8040-Concept-Samples-HMI*.



For more information on the required steps, refer to the sample documentation in the [HMI](#) [► 74] section.

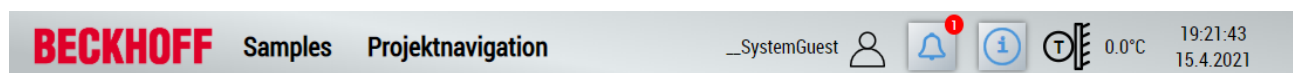
Contents

The individual sample pages of the project are described below. It is advisable to open the live view in order to be able to follow the execution more easily.



Header

The header is provided with various functions (from left to right).



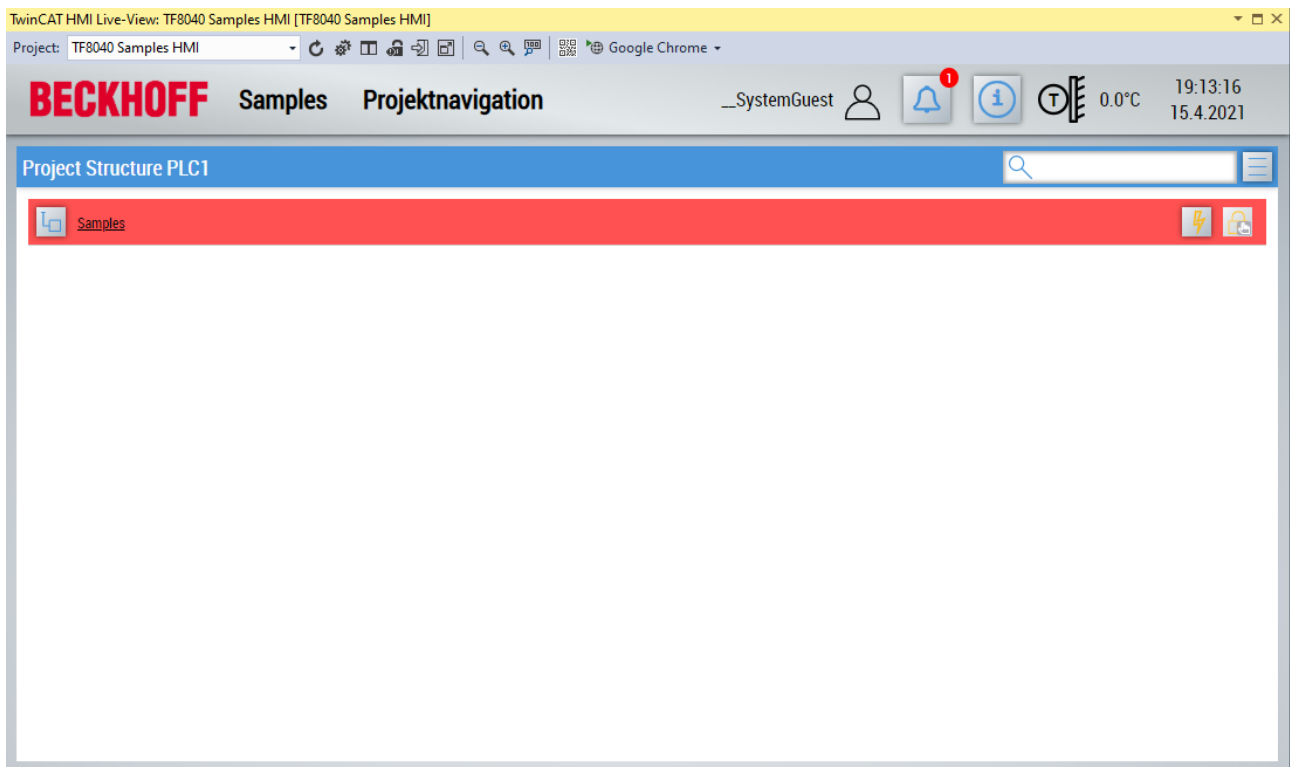
- Logo
- Responsive navigation
- User settings and further information
- Event list
- Building information
- Outdoor temperature
- Date and time



Further information on the functions can be found in the documentation for the [header](#) [► 1021].

Project navigation

The generic project navigation was defined as the start page of the visualization. The content of the Live-View should look like this after the start:



In the project navigation, you can navigate through the project structure and display the parameters of individual views or objects.



For more information on project navigation, see the documentation for [ProjectNavigationTextual](#) [► 889].

The following pages are located under the entry *Content\Samples*.

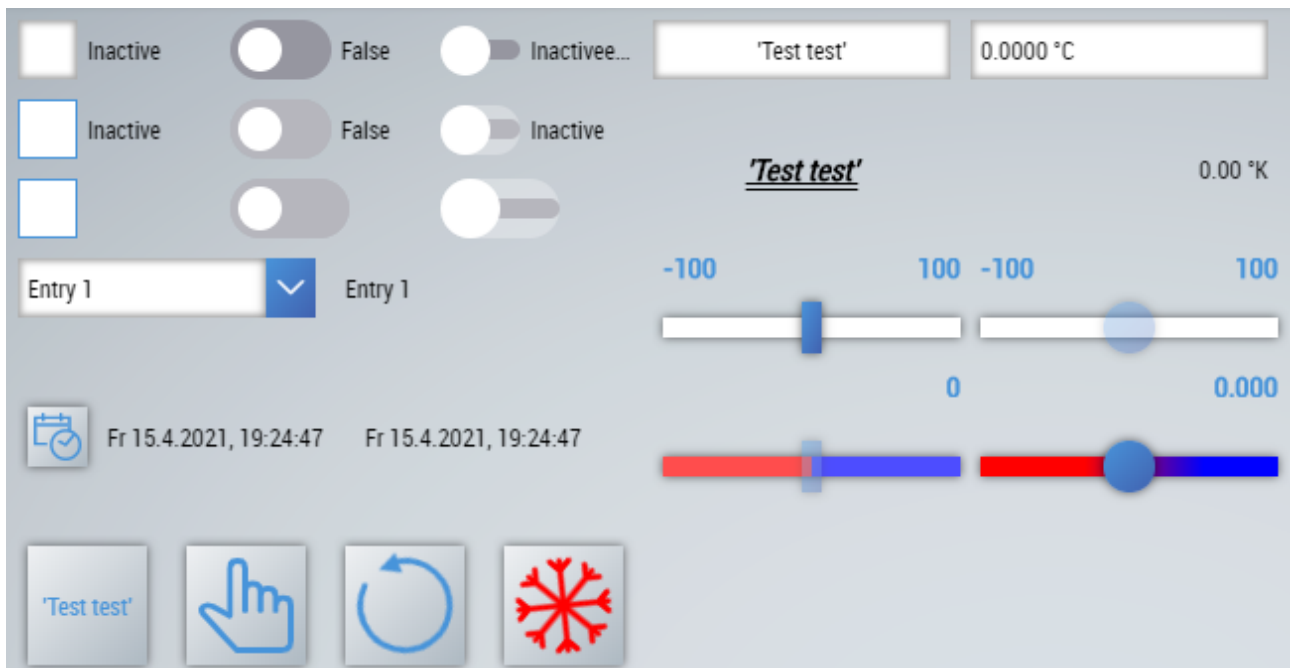
BasicComponents

This page displays all controls that are stored in the Toolbox category *BA | Base*.



The controls are not linked to variables from the PLC.

Further information can be found in the documentation for [Controls](#) [► 822].



BaObjects

On this page, a corresponding control is stored for each primitive data type (*Analog*, *Binary* and *Multistate*). The controls are connected to objects from the PLC and the values are written to the PLC and read again.

Samples - Demonstration - Objects --- General objects - A - Sample AV Setpoint	50.00 °C	
Samples - Demonstration - Objects --- General objects - A - Sample AV Display	50.00 %	
Samples - Demonstration - Objects --- General objects - B - Sample BV Setpoint	☒ Ein	
Samples - Demonstration - Objects --- General objects - B - Sample BV Display	☒ Ein	
Samples - Demonstration - Objects --- General objects - M - Sample MV Setpoint	Stufe 1 v	
Samples - Demonstration - Objects --- General objects - M - Sample MV Display	Stufe 1	

The following information is displayed per line:

- The *Description* of the object.
- The value of the object (writable or read-only).
- Button to open the project navigation of the object (only one entry is visible, because single objects are concerned).

Event

The simulation of the different event types is possible on this page.

The respective event can be activated via the checkboxes and the behavior can be observed in the event list below. A *Uilcon* to display the events is also positioned on the page. The view containing the sample events is linked to both the event list and the *Uilcon*. Therefore, no events outside the view are displayed on this page.

		TimeStamp	Device	ObjectName	InstancePath	Description
1		Di 19.4.2021, 06:21:01	PLC1	Smpl_Demo_Evt~Events++Other++CMD002	MAIN.Events.BIOthrStd	Samples - Demonstration - Event --- Events - Other - Command
2		Di 19.4.2021, 06:21:00	PLC1	Smpl_Demo_Evt~Events++Ntfy++CMD001	MAIN.Events.BINtfySmpl	Samples - Demonstration - Event --- Events - Ntfy - Command
3		Di 19.4.2021, 06:20:59	PLC1	Smpl_Demo_Evt~Events++Mntn++CMD003	MAIN.Events.BIMntnExt	Samples - Demonstration - Event --- Events - Mntn - Command
4		Di 19.4.2021, 06:20:58	PLC1	Smpl_Demo_Evt~Events++Dstb++CMD002	MAIN.Events.BIDstbStd	Samples - Demonstration - Event --- Events - Dstb - Command
5		Di 19.4.2021, 06:20:55	PLC1	Smpl_Demo_Evt~Events++Alm++CMD001	MAIN.Events.BIAImSmpl	Samples - Demonstration - Event --- Events - Alm - Command

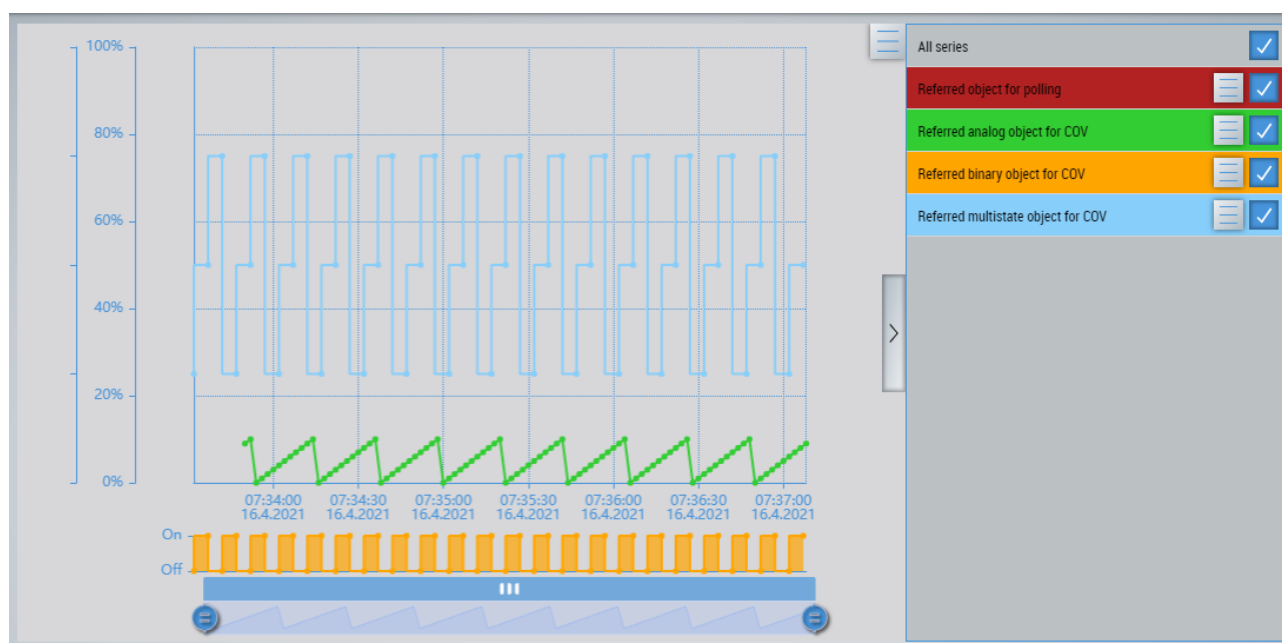
Trend

This page shows the trend control for displaying various trend curves.

In this case, the complete project structure was linked to the control. This filters the project structure for all available trends and displays them. You can select and deselect trends on the right-hand side.



For more information, see the documentation on [Trend](#) [► 897].



Schedule

The schedule displays the *current schedule* and the *weekly schedule*. The *Calendar* tab contains the entries from the linked calendar references and the local exceptions.

The buttons at the bottom of the page allow you to switch to other schedule types. In addition, the alignment and accuracy of the schedule can be set.



For more information, see the documentation on the [Schedule](#) [► 894].



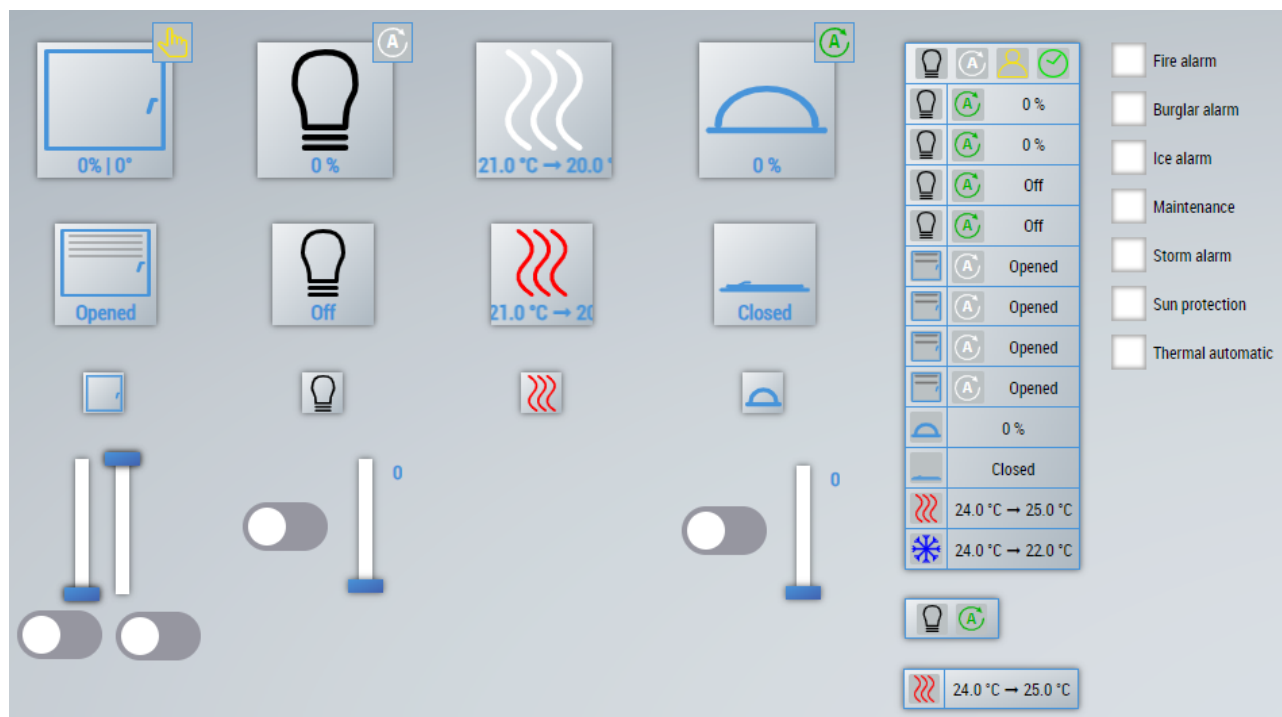
RoomAutomation

The following list shows the available controls for room automation:

Control	Description
Sunblind [► 961]	Displays and controls the position and angle of a sunblind.
Light [► 945]	Displays and controls the brightness value of a lamp (dimnable or on / off).
HeatingCooling [► 939]	Displays and controls the air conditioning of a room.
Window [► 954]	Shows and controls the position of a window (percentage or open/close).
RoomControl [► 972]	This control can combine all of the above controls.



The controls are for demonstration purposes only and are therefore not linked to variables from the PLC.

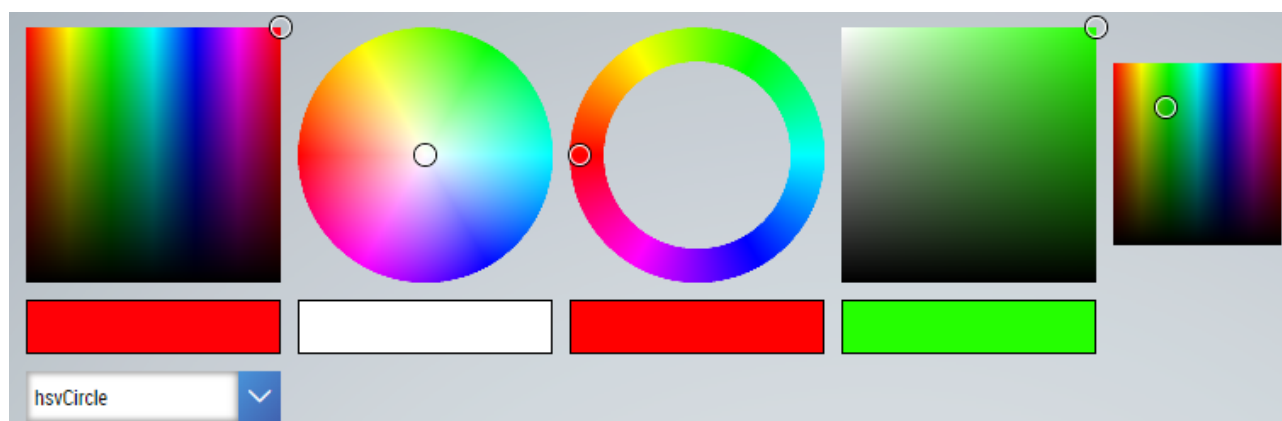


ColorPicker

On this page, ColorPicker is shown in its different versions.



Further information can be found in the documentation for the [ColorPicker](#) [► 877]



5.2 Template samples

The samples in this package are intended to show different templates in the PLC and HMI and how to use them.

5.2.1 PLC

Explanation of the procedure for starting the sample project *TF8040 Template Samples PLC* and its contents.

Procedure

The procedure can be found in the documentation for the [concept sample](#) [► 75].

5.2.2 HMI

Explanation of the sample project TF8040 *Template Samples HMI*.



For more information on the required steps, refer to the sample documentation in the [HMI \[► 74\]](#) section.

Components

The individual sample pages of the project are described below.



The open live view shows many errors. This is because objects in the PLC signal an error if no physical inputs are connected.

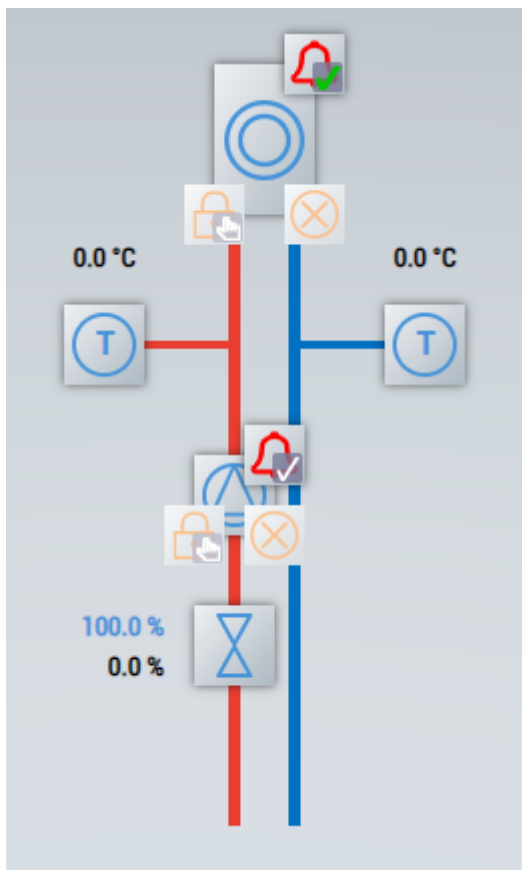
Project navigation

The [generic navigation \[► 889\]](#) is displayed on the content of this page.

Plants

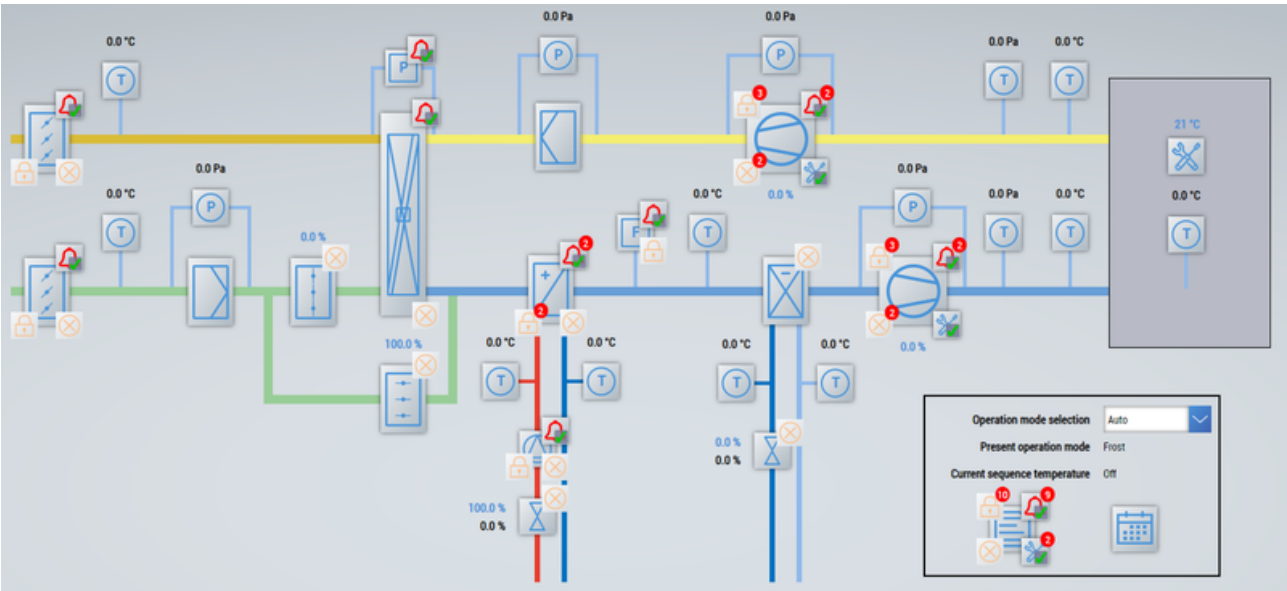
HZK01

Displays a heating circuit.



AHU01

Displays a ventilation system.



6 Programming



For new projects we strongly recommend to use the version 5 of TF8040!

6.1 PLC

6.1.1 General

General information about PLC programming with TwinCAT.

Offline

Regions

Different parameters are intended for different use cases.

For easier assignment, parameters are declared within the following regions:

Region	Description
Fixed Parameters	Parameter variables for configuration, which can be adjusted for an initialization [► 195]. After an object has been initialized, the variable is no longer designed for changes at runtime!
Variable parameters	Parameter variables for configuration, which can be adjusted arbitrarily (i.e. also at runtime).
Operational Parameters	Parameter variables for operation, which can be adjusted by the user as desired (i.e. also at runtime).
Fixed Variables	To be considered by the application as the <i>Fixed Parameters</i> region. However, variables are only provided for the integrator within the TwinCAT environment! They are not accessible from clients (e.g. Site Explorer [► 1024] or TcHmiBa [► 821]).
Output-Properties	Indicates variables that are not made externally accessible by means of VAR_OUTPUT but by a corresponding FB property.



In order not to overload a function block with respect to its interface, rarely used variables are implemented in the form of properties.

Online

Checking

The log window should be checked after loading the PLC.

If no error messages appear, the program is ready for operation. In this way, a defined initial state is established.

6.1.2 Libraries

6.1.2.1 Tc3_XBA

The PLC library Tc3_XBA is an essential element of the TwinCAT function TF8040. The most important component of this library is the base framework including the site server.

6.1.2.1.1 DUTs

6.1.2.1.1.1 Enumerations

6.1.2.1.1.1.1 Events

6.1.2.1.1.1.1.1 E_BA_EventCondition

```

TYPE E_BA_EventCondition :
(
    Invalid                := 0,

    // Separated in event-types (See "E_BA_EventType"):
    eTypeAlarm             := TO_BYTE(E_BA_EventType.eAlarm),
    eTypeDisturb           := TO_BYTE(E_BA_EventType.eDisturb),
    eTypeMaintenance       := TO_BYTE(E_BA_EventType.eMaintenance),
    eTypeNotification       := TO_BYTE(E_BA_EventType.eNotification),
    eTypeOther              := TO_BYTE(E_BA_EventType.eOther),

    // Separated in object-states (See "ST_ObjectStateFlags"):
    eFlagOverridden         := 6,
    eFlagOutOfService       := 7,
    eFlagFault              := 8,
    eFlagActiveEvent        := 9,

    // Separated in priorities (See "E_Priority"):
    ePrioLifeSafety         := 10,
    ePrioCritical           := 11,
    ePrioManualLocal        := 12,
    ePrioManualRemote       := 13,

    // Separated in lock-priorities (See "E_BA_LockPriority"):
    eLockPrioLocalMedium    := 14,
    eLockPrioLocalHigh      := 15,
    eLockPrioMedium         := 16,
    eLockPrioHigh           := 17,

    // Other:
    eEventIconDisplayed      := 18,
) BYTE;
END_TYPE

```

6.1.2.1.1.1.1.2 E_BA_AcknowledgeMode

```

TYPE E_BA_AcknowledgeMode :
(
    eSingle      := 1,
    eEntire      := 2
);
END_TYPE

```

Name	Description
eSingle	Acknowledges only the next upcoming event transition.
eEntire	Acknowledge all unacknowledged event transitions by a single acknowledgement.

6.1.2.1.1.1.1.3 E_BA_AlarmMode

Describes the alarm mode of event-enabled objects.

```

TYPE E_BA_AlarmMode :
(
    Undefined      := 0,
    UserDefined    := 1,
    eSimple        := 2,
    eStandard      := 3,
    eExtended      := 4
) BYTE;
END_TYPE

```

Name	Description
Undefined	No function
UserDefined	User-defined pattern Bit combination [TO-OFFNORMAL TO-FAULT TO-NORMAL] ACK_REQUIRED x

6.1.2.1.1.1.4 E_BA_EventIconState

```

TYPE E_BA_EventIconState :
(
    Invalid        := 0,
    eNone          := 1,
    eIndicated     := 2,
    eGoneAked      := 3,
    eGone          := 4,
    ePresentAked   := 5,
    ePresent       := 6,
) BYTE;
END_TYPE

```

Name	Description
eNone	 The event is not present.
eIndicated	 The event is not (no longer) present, but is indicated for information purposes until it is acknowledged*.
eGoneAked	 The event is not present (anymore). However, it has already been acknowledged but not yet reset**.
eGone	 An event is not present (anymore). However, it was neither acknowledged nor reset*.
ePesentAked	 An event is present and has been acknowledged.
ePresent	 An event is present.



* Only possible with alarm mode *standard*!
 ** Only possible with *extended* alarm mode!

6.1.2.1.1.1.5 E_BA_EventType

```
TYPE E_BA_EventType :
(
  Invalid      := 0,
  eAlarm       := 1,
  eDisturb     := 2,
  eMaintenance := 3,
  eNotification := 4,
  eOther       := 5,
) BYTE;
END_TYPE
```

Type	Symbol in the TwinCAT HMI	E_BA_EventType
Alarm		eAlarm
Fault		eDisturb
Maintenance		eMaintenance
Notification		eNotification
Miscellaneous		eOther

6.1.2.1.1.1.6 E_BA_ObjectStateFlags

```
TYPE E_BA_ObjectStateFlags:
(
  eOutOfService := 1,
  eOverridden   := 2,
) BYTE;
END_TYPE
```

6.1.2.1.1.1.2 Groups

6.1.2.1.1.1.2.1 E_BA_AValCalcMode

```
TYPE E_BA_AValCalcMode :  
(  
    eUndefined    := 0,  
    eMin          := 1,  
    eMax          := 2,  
    eAverage      := 3  
) BYTE;  
END_TYPE
```

6.1.2.1.1.1.2.2 E_BA_BValCalcMode

```
TYPE E_BA_BValCalcMode :  
(  
    eUndefined    := 0,  
    eAnd          := 1,  
    eOr           := 2,  
    eXOr          := 3  
) BYTE;  
END_TYPE
```

6.1.2.1.1.1.2.3 E_BA_MValCalcMode

```
TYPE E_BA_MValCalcMode :  
(  
    eUndefined    := 0,  
    eMin          := 1,  
    eMax          := 2  
) BYTE;  
END_TYPE
```

6.1.2.1.1.1.3 Priority

6.1.2.1.1.1.3.1 E_BA_LockPriority

```
TYPE E_BA_LockPriority :  
(  
    Invalid        := 0,  
    eNoLock        := 1,  
    eLocalMedium   := 2,  
    eLocalHigh     := 3,  
    eMedium        := 4,  
    eHigh          := 5  
) BYTE;  
END_TYPE
```

Name	Description
Invalid	No significance for the user.
eNolock	No locking
eLocalMedium	The event of an object triggers a switching action by means of the FB BA PlantLock [► 121] , which is located in the same level within the basic structure. The level of priority is "Medium" and generally refers to a technical fault or to the safety of a plant.
eLocalHigh	The event of an object triggers a switching action by means of the FB BA PlantLock [► 121] , which is located in the same level within the basic structure. The level of priority is "High" and generally refers to an alarm or life safety.
eMedium	The event of an object triggers a switching action. It is reported to all FB BA PlantLock [► 121] that are located in the levels of the basic frame above. The priority is "Medium" and generally refers to a technical fault or to the safety of a plant.
eHigh	The event of an object triggers a switching action. It is reported to all FB BA PlantLock [► 121] that are located in the levels of the basic frame above. The priority is "High" and generally refers to an alarm or the safety of life.

6.1.2.1.1.1.3.2 E_BA_Priority

```

TYPE E_BA_Priority :
(
  Invalid      := 0,
  eDefault     := 1,
  eProgram     := 2,
  eManualRemote := 3,
  eManualLocal := 4,
  eCritical    := 5,
  eLifeSafety  := 6,
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
eDefault	Default setting if no priority is selected.
eProgram	Control by the program.
eManualRemote	Manual override by parameters.
eManualLocal	Manual override, e.g. via switch.
eCritical	Settings for the operation of critical parts of the plant
eLifeSafety	

6.1.2.1.1.1.4 DPAD

6.1.2.1.1.1.4.1 E_BA_ConcatDPADMode

```

TYPE E_BA_ConcatDPADMode :
(
  Invalid      := 0,
  eNone        := 1,
  eEntryPoint  := 2,
  eParent      := 3,
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
eNone	✱ The event is not present.
eEntryPoint	Linking the instance to a defined entry point, e.g. plant.
eParent	Linking the instance to the parent.

6.1.2.1.1.1.4.2 E_BA_DPADMode

```

TYPE E_BA_DPADMode :
(
  Invalid           := 0,
  Undefined         := 1,
  eExclude          := 2,
  eInclude          := 3,
  eIncludeObjectName := 4,
  eIncludeDescription := 5,
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
eExclude	No application of the object name and description.
eInclude	Application of the object name and description.
eIncludeObjectName	Application of the object name.
eIncludeDescription	Application of the Description.

6.1.2.1.1.1.5 Types

6.1.2.1.1.1.5.1 E_BA_NodeType

```

TYPE E_BA_NodeType :
(
  Invalid           := 0,
  Automatic         := 1,
  {region 'Default'}
  eUnknown          := 10,
  eOther            := 11,
  eGeneral          := 12,
  {endregion}
  {region 'Location'}
  eLocation         := 13,
  eBuilding         := 14,
  eBuildingElement  := 15,
  eInformationFocus := 16,
  eControlCabinet   := 17,
  eTrade            := 18,
  eFloor            := 19,
  eRoom             := 20,
  ePlant            := 21,
  {endregion}
  {region 'Equipment'}
  eComponent        := 22,
  eAggregate        := 23,
  eFunction          := 24,
  {endregion}
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
Automatic	The system chooses a meaningful node type.

6.1.2.1.1.1.5.2 E_BA_ObjectType

```

TYPE E_BA_ObjectType :
(
  Invalid      := 0,
  Undefined    := 1,
{region 'Analog'}
  eAnalogInput  := 10,
  eAnalogOutput := 11,
  eAnalogValue  := 12,
{endregion}
{region 'Binary'}
  eBinaryInput  := 15,
  eBinaryOutput := 16,
  eBinaryValue  := 17,
{endregion}
{region 'Multistate'}
  eMultistateInput  := 20,
  eMultistateOutput := 21,
  eMultistateValue  := 22,
{endregion}
{region 'Misc'}
  eObject      := 25,
  eStructuredView := 26,
  eProject     := 27,
  eEventClass  := 28,
  eCalendar    := 29,
  eSchedule    := 30,
  eLoop        := 31,
  eTrend       := 32,
{endregion}
) BYTE;
END_TYPE

```

6.1.2.1.1.1.6 E_BA_Role

```

TYPE E_BA_Role :
(
  Undefined := 0,
  eGuest    := 1,
  eBasic    := 2,
  eAdvanced := 3,
  eExpert   := 4,
  eInternal := 5,
  eLocked   := 6
) BYTE;
END_TYPE

```

Name	Description
eGuest	Read access of the Present Value for all users.
eBasic	Read access to a few properties for a limited number of users
eAdvanced	Access for advanced users
eExpert	Access for experts
eInternal	Access for service employees
eLocked	No access

6.1.2.1.1.1.7 Communication

6.1.2.1.1.1.7.1 E_BA_ComState

```

TYPE E_BA_ComState :
(
  Invalid      := 0,
  Error        := 1,
  eUnused      := 10,
  eInitialization := 11,
  eOperation    := 12
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
Error	A problem has occurred and the object has been excluded from execution.
eUnused	The communication object is not used by any other object.
eInitialization	The object is initialized.
eOperation	The object is in operation.

6.1.2.1.1.1.8 Objects

6.1.2.1.1.1.8.1 E_BA_ObjectPurpose

```

TYPE E_BA_ObjectPurpose :
(
  Undefined      := 0,
  eInternal      := 1,
  eStructurize   := 2,
  eDescriptive   := 3,
  eGeneral       := 10,
  eOperation     := 11,
  eValue         := 12,
  eInput         := 13,
  eOutput        := 14,
  eManagement    := 20,
  eAggregate     := 21
) BYTE;
END_TYPE

```

Name	Description
eInternal	Internal management functions (e.g. EventClass objects)
eStructurize	Organization of the project structure (see Structured View Objects in TF8020)
eDescriptive	Descriptive information (e.g. project object)
eOperation	Operative value (setpoint, display value)
eValue	Value
eInput	Physical input value
eOutput	Physical output value
eManagement	Manages referenced objects (plant, control, ...)

6.1.2.1.1.1.8.2 Loop

6.1.2.1.1.1.8.2.1 E_BA_LoopSeqState

```

TYPE E_BA_LoopSeqState :
(
  Invalid      := 0,
  eNoSequence  := 1,
  eInactive    := 2,
  eWaiting     := 3,
  eActive      := 4,
  ePassed      := 5
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
eNoSequence	No sequence active
eInactive	Controller inactive
eWaiting	Waiting for tokens.
eActive	Operational (The current instance has the token.)
ePassed	The instance is at the maximum and has passed the token.

6.1.2.1.1.1.8.2.2 E_BA_SeqDirection

```

TYPE E_BA_SeqDirection :
(
  Invalid           := 0,
  eNoSequence       := 1,
  eLeft_to_Right    := 2,
  eMiddle_to_LeftOrRight := 3,
  eRight_to_Left     := 4
) BYTE;
END_TYPE

```

Name	Description
eNoSequence	No sequence active
eLeft_to_Right	The sequence starts at the loop on the left side, the token moves to the next loops.
eMiddle_to_LeftOrRight	Sequence starts in the middle, the token is passed to the next OR previous loop (depending on the control direction of the sequence).
eRight_to_Left	The sequence starts at the loop on the right side, the token moves to the previous loops.

6.1.2.1.1.1.8.2.3 E_BA_SeqState

```

TYPE E_BA_SeqState :
(
  Invalid           := 0,
  eInactive         := 1,
  eDetermineToken   := 2,
  eOperation        := 3
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
eInactive	No enabling a sequenced control.
eDetermineToken	Search of the token to determine the active loop object.
eOperation	The object is in operation.

6.1.2.1.1.1.8.3 PlantCtrl

6.1.2.1.1.1.8.3.1 E_BA_AggregateIgnoreFlags

```

TYPE E_BA_AggregateIgnoreFlags :
(
  None              := 0,
  All               := 2#1111_1111,
  Delay             := (eDelayStepDown OR eDelayStepUp),
  eProcesses        := 2#0000_0001,
  eEvents           := 2#0000_0010,
  eDelayStepDown    := 2#0000_0100,
  eDelayStepUp      := 2#0000_1000
) BYTE;
END_TYPE

```

Name	Description
eProcesses	Ignores processes of an aggregate.
eEvents	Ignore the events of an aggregate.
eDelayStepDown	Ignores a step-down delay from the aggregate.
eDelayStepUp	Ignores a step-up delay from the aggregate.

6.1.2.1.1.1.9 Parameter

6.1.2.1.1.1.9.1 E_BA_Attribute

```
TYPE E_BA_Attribute :
(
    Invalid      := 0,
    eIndex       := 1
) BYTE;
END_TYPE
```

6.1.2.1.1.1.9.2 E_BA_CommissioningState

```
TYPE E_BA_CommissioningState :
(
    Invalid      := 0,
    eUnknown     := 1,
    eChecked     := 2,
    eDefectIO    := 3,
    eDefectWiring := 4,
    eDefectDevice := 5,
    eDefectOther := 6
) BYTE;
END_TYPE
```

Name	Description
Invalid	No significance for the user.
eUnknown	State unknown
eChecked	Checked
eDefectIO	Input or output defective
eDefectWiring	Line defective
eDefectDevice	Field device defective
eDefectOther	Other defect

6.1.2.1.1.1.9.3 E_BA_ParamCOVMode

```
TYPE E_BA_ParamCOVMode :
(
    Invalid      := 0,
    eNone        := 1,
    eStandard    := 2,
    eManual      := 3,
    ePrioritized := 4
) BYTE;
END_TYPE
```

Name	Description
Invalid	No significance for the user.
eNone	COV display of the parameter is inactive.
eStandard	COV display of the parameter is evaluated periodically (depending on the load).
eManual	COV display is executed, but the changed values must be read or written manually.
ePrioritized	COV display of the parameter is evaluated in each PLC cycle.

6.1.2.1.1.1.9.4 E_BA_Parameter

Possible properties of a BACnet object.

```
TYPE E_BA_Parameter :
(
    Invalid           := 0,
    eConfigure        := 1,
    eToggleMode       := 2,
    eStepDelay        := 3,
    eCOVIncrement     := 4,

```

```

eAction                := 5,
eMinOffTime            := 6,
eMinOnTime             := 7,
eStateChangeCount      := 8,
eStateChangeTime       := 9,
eStateChangeResetPoint := 10,
eActiveTimeElapsed     := 11,
eActiveTimeResetPoint  := 12,
eInstructionText       := 13,
eAckedTransitions      := 14,
eAcknowledgeRm        := 15,
eEnPlantLock           := 16,
eAlarmValue            := 17,
eAlarmValues           := 18,
eFaultValues           := 19,
eTimeDelay             := 20,
eLowLimit              := 21,
eHighLimit             := 22,
eLimitDeadband         := 23,
eEventDetectionEnable  := 24,
eEventEnable           := 25,
eEventClassID          := 26,
eEventTransitionText   := 29,
eEventState            := 30,
eStatusFlags           := 31,
eReliability           := 32,
eEnable                := 33,
eOutOfService          := 34,
eInactiveText          := 35,
eActiveText            := 36,
eStateText             := 37,
eStateCount            := 38,
eUnit                  := 39,
ePresentValue          := 40,
eDefaultValue          := 41,
eTag                   := 42,
eAssignAsTrendReference := 43,
eDeviceType            := 44,
ePolarity              := 45,
eMappingMode           := 46,
eFeedbackMappingMode   := 47,
eFeedbackPolarity       := 48,
eOverriddenPolarity    := 49,
eScaleOffset           := 50,
eResolution            := 51,
eFeedbackValue         := 52,
eRawValue              := 53,
eRawFeedback           := 54,
eRawOverride           := 55,
eRawState              := 56,
eTerminal              := 57,
eSensor                := 58,
eAddress               := 59,
eCommissioningState    := 60,
eSymbolPath            := 61,
eSymbolName            := 62,
eInstanceID            := 63,
eObjectName            := 64,
eDescription           := 65,
eObjectType            := 66,
ePurpose               := 67,
eNodeType              := 68,
ePriorityArray          := 69,
eActivePriority         := 70,
eProjectInfo           := 71,
eOperatorInfo          := 72,
eTechnicalStaffInfo    := 73,
eEngineerInfo          := 74,
eDateList              := 75,
eEventType             := 76,
eAlarmMode             := 77,
ePriority               := 78,
eAcknowledgeRequired   := 79,
eSetpoint              := 80,
eControlledValue       := 81,
eCtrlDeviation         := 82,
eProportionalConstant  := 83,
eIntegralConstant      := 84,
eDerivativeConstant    := 85,
eDampConstant          := 86,

```

```

eOutputUnit           := 87,
eOpMode               := 88,
eNeutralZone          := 89,
eSynchronizedLoop     := 90,
eMinOutput            := 91,
eMaxOutput            := 92,
eWeek                 := 93,
eExceptionList        := 94,
eReferencedParam      := 95,
eBufferSize           := 96,
eLogBuffer            := 97,
eLoggingType          := 98,
eLogInterval          := 99,
eStartTime            := 100,
eStopTime             := 101,
eStopOnFull           := 102,
eNotificationThreshold := 103,
eRecordCount          := 104,
eTotalRecordCount     := 105
) BYTE;
END_TYPE

```

6.1.2.1.1.1.9.5 E_BA_ParamPurpose

```

TYPE E_BA_ParamPurpose :
(
  Invalid           := 0,
  eUnique           := 1,
  eIndividual       := 2,
  eIndividualSetting := 3,
  eSetting          := 4,
  eOperational      := 5,
  eState            := 6
) BYTE;
END_TYPE

```

Name	Description
Invalid	No significance for the user.
eUnique	Unique parameter value that should not be used by multiple objects.
eIndividual	Individual parameter per object instance.
eIndividualSetting	Individual setting per object instance.
eSetting	General setting

6.1.2.1.1.1.9.6 E_BA_ParamSyncMode

```

TYPE E_BA_ParamSyncMode :
(
  Invalid           := 0,
  eDefault          := 1,
  eInitialWrite     := 3,
  eStatusFlags      := 4,
  eEventTransitions := 5,
  eEventTransitionBits := 6,
  eBetaBuffered     := 7
) BYTE;
END_TYPE

```

Name	Description
eInitialWrite	The parameter is initialized by a single write command.

6.1.2.1.1.1.10 References

6.1.2.1.1.1.10.1 E_BA_AssignRefMode

Determination of an object as a reference in a related (e.g. trend) object.

```

TYPE E_BA_AssignRefMode :
(
  eNone           := 1,
  eAssignNow      := 2,

```

```
eInitByProfile      := 3
) BYTE;
END_TYPE
```

Name	Description
eAssignNow	Assign reference immediately
eInitByProfile	Assign reference when initializing the object, provided it is enabled in the associated object profile.

6.1.2.1.1.1.11 Supplements

6.1.2.1.1.1.11.1 E_BA_SupplementType

```
TYPE E_BA_SupplementType :
(
  Invalid      := 0,
  All          := 1,
  ePLC         := 10,
  eBACnet      := 11,
) BYTE;
END_TYPE
```

6.1.2.1.1.1.12 Functional

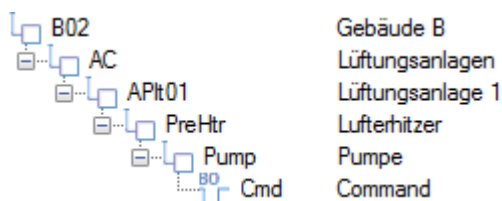
6.1.2.1.1.1.12.1 E_BA_NodeTypeTarget

The enumeration describes the relative reference to an object.

Sample

In the following project structure:

- the ventilation *Ventilation System 1* is a **system**
- the air heater *PreHtr* is an **aggregate**
- the pump *Pump* is an **aggregate**
- the command *Cmd* is a **function**



In relation to the system *Ventilation System 1*:

- the air heater *PreHtr* is the **first aggregate**
- the pump *Pump* is the **last aggregate**
- the command *Cmd* is the **function**

```
TYPE E_BA_NodeTypeTarget :
(
  Invalid          := 0,
  eFirstAggregate  := 1,
  eLastAggregate   := 2,
  eFunction        := 3,
) BYTE;
END_TYPE
```

Name	Description
Invalid	No significance for the user.
eFirstAggregate	Refers to the first aggregate in relation to an object.
eLastAggregate	Refers to the last aggregate in relation to an object.
eFunction	Refers to the function in relation to an object.

6.1.2.1.1.1.12.2 E_BA_ProjectState

```

TYPE E_BA_ProjectState :
(
  Invalid           := 0,
  Error             := 1,
  ePrepare          := 10,
  ePreInit          := 11,
  eValidateObjects  := 12,
  eReferenceInit    := 13,
  eObjectFinishInit := 14,
  eProjectInit      := 15,
  eFirstOpCycle     := 16,
  eOperation        := 17
) BYTE;
END_TYPE

```

6.1.2.1.1.1.12.3 E_BA_ObjectState

```

TYPE E_BA_ProjectState :
(
  Invalid           := 0,
  Error             := 1,
  Idle              := 2,
  ePrepare          := 10,
  ePreInit1         := 11,
  ePreInit2         := 12,
  eSynchronizeState := 13,
  eInstanceInit1    := 14,
  eInstanceInit2    := 15,
  eInstanceInit3    := 16,
  eInstanceInit4    := 17,
  eComInitDevice    := 18,
  eComInitObject    := 19,
  ePrepareSupplement := 20,
  eWaitForSupplementInit := 21,
  eSupplementInstInit := 22,
  eSupplementInit   := 23,
  ePostInit         := 23,
  eWaitForProject   := 24,
  eOperation        := 25
) BYTE;
END_TYPE

```

6.1.2.1.1.1.12.4 E_BA_ProcessSignalSource

```

TYPE E_BA_ObjectType :
(
  Invalid      := 0,
  eVarInput    := 1,
  eParameter   := 2
) BYTE;
END_TYPE

```

6.1.2.1.1.1.12.5 E_BA_SubscriberState

```

TYPE E_BA_SubscriberState :
(
  Invalid      := 0,

  eInit        := 1,
  eInitSubscription := 2,
  eReading     := 3,
  eReady       := 4,
  eSuppressing := 5,
  eError       := 6,

```

```
) BYTE;
END_TYPE
```

6.1.2.1.1.2 Types

6.1.2.1.1.2.1 *ST_BA_ObjectAttributes*

```
TYPE ST_BA_ObjectAttributes :
STRUCT
  sParent      : REFERENCE TO STRING;
  sLabel       : REFERENCE TO STRING;
  sProfile     : REFERENCE TO STRING;
  eDataClass   : E_BA_DataClass      := E_BA_DataClass.Invalid;
  ePurpose     : E_BA_ObjectPurpose := E_BA_ObjectPurpose.Undefined;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.2 *ST_BA_SubjectAttributes*

```
TYPE ST_BA_SubjectAttributes :
STRUCT
  sIdentifier   : T_BA_MedString;
  nIndex       : UDINT(1 .. 1000000) := 1;
  nHash        : DWORD;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.3 DPAD

6.1.2.1.1.2.3.1 *ST_BA_DPAD_Identifier*

```
TYPE ST_BA_DPAD_Identifier:
STRUCT
  eMode          : E_BA_DPADMode      := E_BA_DPADMode.eInclude;
  eNodeType       : E_BA_NodeType;
  nIndexDigits   : UINT               := XBA_Param.nDPAD_DefIndexDigits;
{attribute 'TcEncoding' := 'UTF-8'}
  sSeparator_ObjectName : T_BA_ShortString := XBA_Param.sDPAD_ObjectName_DefSeparator;
{attribute 'TcEncoding' := 'UTF-8'}
  sSeparator_Description : T_BA_ShortString := XBA_Param.sDPAD_Description_DefSeparator;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.4 Events

6.1.2.1.1.2.4.1 *ST_BA_EventIcon*

```
TYPE ST_BA_EventIcon :
STRUCT
  eEventType     : E_BA_EventType      := E_BA_EventType.Invalid;
  eState         : E_BA_EventIconState := E_BA_EventIconState.Invalid;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.4.2 *ST_BA_EventsPerIconImage*

```
TYPE ST_BA_EventsPerIconImage :
STRUCT
  eMostPriorisedState : E_BA_EventIconState := E_BA_EventIconState.Invalid;
  nCount              : UDINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
eMostPriorisedState	E_BA_EventIconState [► 93]	Highest priority icon state with one or more active events.
nCount	UDINT	Number of active events depending on the highest prioritized icon state.

6.1.2.1.1.2.4.3 T_BA_EventConditionFlags

```
TYPE T_BA_EventConditionFlags : ARRAY[E_BA_EventCondition.First .. E_BA_EventCondition.Last] OF BOOL; END_TYPE
```

6.1.2.1.1.2.4.4 T_BA_EventTransitions

```
TYPE T_BA_EventTransitions : ARRAY[E_BA_EventTransition.First .. E_BA_EventTransition.Last] OF BOOL; END_TYPE
```

6.1.2.1.1.2.4.5 T_BA_EventTransitionText

```
TYPE T_BA_EventTransitionText : ARRAY [E_BA_EventTransition.First .. E_BA_EventTransition.Last] OF STRING(BA_Param.nEventTransitionText_Length); END_TYPE
```

6.1.2.1.1.2.4.6 History

6.1.2.1.1.2.4.6.1 ST_BA_EventHistoryEntry

```
TYPE ST_BA_EventHistoryEntry :
STRUCT
  iObject          : I_BA_EventObject;
  nEventIncrement  : UINT;
  dtTimeStamp      : DT;
  stEvent          : ST_BA_EventIcon;
  eReliability     : E_BA_Reliability;
  stStateFlags     : ST_BA_StatusFlags;
  eLockPriority    : E_BA_LockPriority;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5 Info

6.1.2.1.1.2.5.1 ST_BA_ComStat

```
TYPE ST_BA_ComStat :
STRUCT
  nReq      : UDINT;
  nBadRsp   : UDINT;
  nGoodRsp  : UDINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
nReq	UDINT	Requests received
nBadRsp	UDINT	Requests that generated an ADS error in response.
nGoodRsp	UDINT	Requests that did not generate an ADS error in response.

6.1.2.1.1.2.5.2 ST_BA_ContactInfo

```
TYPE ST_BA_ContactInfo :
STRUCT
  {attribute 'TcEncoding':='UTF-8'}
  sName      : STRING;
  {attribute 'TcEncoding':='UTF-8'}
  sPhone     : STRING;
  {attribute 'TcEncoding':='UTF-8'}
  sMail      : STRING;
  {attribute 'TcEncoding':='UTF-8'}
```



```
sWebsite      : STRING;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5.3 ST_BA_CumObjState

```
TYPE ST_BA_CumObjState :
STRUCT
    nPendingInit      : UDINT
    nOperational       : UDINT;
    nError             : UDINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
nPendingInit	UDINT	Number of objects to be initialized.
nOperational	UDINT	Number of objects ready for operation (initialization completed).
nError	UDINT	Number of objects in error state (initialization failed).

6.1.2.1.1.2.5.4 ST_BA_DeviceInfo

```
TYPE ST_BA_DeviceInfo :
STRUCT
    sAmsNetID      : T_AmsNetId;
    dtNow           : DT;
    stSiteServer    : ST_BA_ComStat;
    stSiteClient    : ST_BA_ComStat;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5.5 ST_BA_ProjectInfo

```
TYPE ST_BA_ProjectInfo :
STRUCT
    {attribute 'TcEncoding':='UTF-8'}
    sTitle          : STRING;
    {attribute 'TcEncoding':='UTF-8'}
    sDescription     : STRING;
    {attribute 'TcEncoding':='UTF-8'}
    sLocation        : STRING;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5.6 ST_BA_RuntimeInfo

```
TYPE ST_BA_RuntimeInfo :
STRUCT
    nTaskCount       : UDINT;
    nOnlineChanges    : UDINT;
    bPersistentDataValid : BOOL;
    dtCompileTime     : DT;
    sNamespace        : STRING;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5.7 ST_BA_SupplementInfo

```
TYPE ST_BA_SupplementInfo :
STRUCT
    PLC      : ST_BA_SplInfo_PLC;
    BACnet    : ST_BA_SplInfo_BACnet;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5.8 ST_BA_TaskInfo

```
TYPE ST_BA_TaskInfo :
STRUCT
    {attribute 'parameterUnit':='ms'}
    nCycleTime : UDINT;
END_STRUCT
```

```

nCycleCount      : UDINT;
nPriority         : UINT;
nADSPort         : UINT;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.5.9 ST_BA_VersionInfo

```

TYPE ST_BA_VersionInfo :
STRUCT
{attribute 'BaGuid':='00000001-0001-0001-0150-000000000003'}
{attribute 'TcEncoding':='UTF-8'}
Tc3_XBA          : STRING(23);
{attribute 'BaGuid':='00000001-0001-0001-0151-000000000004'}
{attribute 'TcEncoding':='UTF-8'}
Tc3_BA2_Common   : STRING(23);
{attribute 'BaGuid':='00000001-0001-0001-0170-000000000004'}
{attribute 'TcEncoding':='UTF-8'}
Tc3_BACnetRev14  : STRING(23);
{attribute 'BaGuid':='00000001-0000-0000-0200-000000000200'}
{attribute 'TcEncoding':='UTF-8'}
sBACnet_Stack     : STRING(23);
{attribute 'BaGuid':='00000001-0000-0000-0201-000000000300'}
nBACnet_Revision  : DINT;
{attribute 'BaGuid':='00000001-0001-0001-0001-000000000001'}
{attribute 'TcEncoding':='UTF-8'}
Project           : STRING(23);
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.5.10 Supplements

6.1.2.1.1.2.5.10.1 ST_BA_SplInfo_BACnet

```

TYPE ST_BA_SplInfo_BACnet EXTENDS ST_BA_BaseSupplementInfo :
STRUCT
sAmsNetID        : T_AmsNetId;
tAmsPort         : T_AmsPort;
nDeviceID        : UDINT;
eDeviceStatus    : E_BACnet_DeviceStatus;
bIsOperational   : BOOL;
eErrorID         : E_BACnet_Error;
nLocalObjects    : UDINT;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.5.10.2 ST_BA_SplInfo_PLC

```

TYPE ST_BA_SplInfo_PLC EXTENDS ST_BA_BaseSupplementInfo :
STRUCT
stLocalObjects   : ST_BA_CumObjState;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.5.10.3 [Base]

6.1.2.1.1.2.5.10.3.1 ST_BA_BaseSupplementInfo

```

TYPE ST_BA_BaseSupplementInfo :
STRUCT
bIsEnabled       : BOOL;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.6 Objects

6.1.2.1.1.2.6.1 ST_BA_ObjectIdentifier

```

TYPE ST_BA_ObjectIdentifier :
STRUCT
  eObjectType      : E_BA_ObjectType := E_BA_ObjectType.Invalid;
  nInstanceID      : UDINT           := 0;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.6.2 Loop

6.1.2.1.1.2.6.2.1 ST_BA_SeqStandByInfo

```

TYPE ST_BA_SeqStandByInfo :
STRUCT
  nEnCount          : UINT;
  iFirstEn          : I_BA_LoopSeq;
  iLastEn           : I_BA_LoopSeq;
  iFirstDmd         : I_BA_LoopSeq;
  iLastDmd          : I_BA_LoopSeq;
  iLastReverse      : I_BA_LoopSeq;
  iFirstDirect      : I_BA_LoopSeq;
  nActionInvertCount : UINT;
  nDirectCount      : UINT;
  nReverseCount     : UINT;
  iActive           : I_BA_LoopSeq;
END_STRUCT
END_TYPE

```

Name	Type	Description
nEnCount	UINT	Count of released instances.
iFirstEn	I_BA_LoopSeq	Reference of the first released loop object.
iLastEn	I_BA_LoopSeq	Reference of the last released loop object.
iFirstDmd	I_BA_LoopSeq	Reference of the first requested loop object.
iLastDmd	I_BA_LoopSeq	Reference of the last requested loop object.
iLastReverse	I_BA_LoopSeq	Reference of the last loop object with indirect control direction.
iFirstDirect	I_BA_LoopSeq	Reference of the first loop object with direct control direction.
nActionInvertCount	UINT	Counter of the instances that reverse the control direction.
nDirectCount	UINT	Counter of instances with direct control direction
nReverseCount	UINT	Counter of instances with indirect control direction.
iActive	I_BA_LoopSeq	Reference with the current token.

6.1.2.1.1.2.6.3 Trend

6.1.2.1.1.2.6.3.1 ST_BA_TrendSettings

```

TYPE ST_BA_TrendSettings :
STRUCT
  bEnable          : BOOL           := FALSE;
  eLoggingType     : E_BA_LoggingType := E_BA_LoggingType.ePolled;
  {attribute 'parameterUnit':='s'}
  nLogInterval     : UDINT          := BA_Param.nTrend_DefLogInterval;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.6.4 Value

6.1.2.1.1.2.6.4.1 ST_BA_EventObjectValue

```
TYPE ST_BA_EventObjectValue EXTENDS ST_BA_ObjectValue :
STRUCT
    bEvent      : BOOL;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.6.4.2 ST_BA_ObjectValue

```
TYPE ST_BA_ObjectValue :
STRUCT
    uPresentValue : U_BA_ClassValue;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.6.4.3 ST_BA_SchedValue

```
TYPE ST_BA_SchedValue EXTENDS ST_BA_ObjectValue :
STRUCT
    uPredictedValue : U_BA_ClassValue;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.6.4.4 ST_BA_TrendValue

```
TYPE ST_BA_TrendValue EXTENDS ST_BA_EventObjectValue :
STRUCT
    nRecordCount      : UDINT;
    nTotalRecordCount : UDINT;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.7 Parameter

6.1.2.1.1.2.7.1 ST_BA_LimitParam

```
TYPE ST_BA_LimitParam :
STRUCT
    bEnable : BOOL;
    fValue  : REAL;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.7.2 ST_BA_ObjectParameter

```
TYPE ST_BA_ObjectParameter :
STRUCT
    {region 'Reference'}
    iObject      : I_BA_Object := 0;
    stObject     : ST_BA_ObjectIdentifier;
    {endregion}
    {region 'Parameter'}
    eParameter   : E_BA_Parameter := E_BA_Parameter.ePresentValue;
    {endregion}
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.7.3 ST_BA_StepDelayParam

```
TYPE ST_BA_StepDelayParam :
STRUCT
    {attribute 'parameterUnit' := 's'}
    nDown : UDINT;
    {attribute 'parameterUnit' := 's'}
    nUp   : UDINT;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.7.4 ST_BA_TimeDelayParam

```

TYPE ST_BA_TimeDelayParam :
STRUCT
  {attribute 'parameterUnit':= 's'}
  nToAbnormal      : UDINT;
  {attribute 'parameterUnit':= 's'}
  nToNormal        : UDINT;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.7.5 T_BA_MedString

```

TYPE T_BA_MedString : STRING(64); END_TYPE

```

6.1.2.1.1.2.7.6 T_BA_ShortString

```

TYPE T_BA_ShortString : STRING(12); END_TYPE

```

6.1.2.1.1.2.7.7 T_BA_SmallString

```

TYPE T_BA_SmallString : STRING(32); END_TYPE

```

6.1.2.1.1.2.7.8 Calendar**6.1.2.1.1.2.7.8.1 T_BA_CalendarDateList**

```

TYPE T_BA_CalendarDateList : ARRAY[1 .. XBA_Param.nCal_EntryCount] OF ST_BA_CalendarEntry; END_TYPE

```

6.1.2.1.1.2.7.9 Multistate**6.1.2.1.1.2.7.9.1 T_BA_StateText**

```

TYPE T_BA_StateText : STRING(XBA_Param.nStateText_Length); END_TYPE

```

6.1.2.1.1.2.7.9.2 T_BA_StateTextArray

```

TYPE T_BA_StateTextArray : ARRAY[1 .. XBA_Param.nMultistate_StateCount] OF T_BA_StateText; END_TYPE

```

6.1.2.1.1.2.7.10 References**6.1.2.1.1.2.7.10.1 ST_BA_AttributeInfo**

Describes details for defined attributes.

```

TYPE ST_BA_AttributeInfo :
STRUCT
  sShortcut      : T_BA_ShortString;
  eDataType      : E_BA_DataType := E_BA_DataType.Invalid;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.7.10.2 ST_BA_SyncInfo

```

TYPE ST_BA_SyncInfo :
STRUCT
  bChanged      : BOOL;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.7.10.3 U_BA_ParamRef

```

TYPE U_BA_ParamRef :
UNION
  fb1P      : POINTER TO FB_BA_1ParamRef;
  fb2P      : POINTER TO FB_BA_2ParamRef;

```

```

fb3P      : POINTER TO FB_BA_3ParamRef;
fb4P      : POINTER TO FB_BA_4ParamRef;
fb5P      : POINTER TO FB_BA_5ParamRef;
fbArrP    : POINTER TO FB_BA_ArrayParamRef;
fbEnP     : POINTER TO FB_BA_EnParamRef;
fbPrioP   : POINTER TO FB_BA_PrioParamRef;
END_UNION
END_TYPE

```

6.1.2.1.1.2.7.11 Scheduler

6.1.2.1.1.2.7.11.1 ST_BA_SchedCalendar

```

TYPE ST_BA_SchedCalendar :
STRUCT
  iRefCalendar      : I_BA_Object;
{region 'Internal'}
  {attribute 'conditionalshow'}
  stRefCalendarID   : ST_BA_ObjectIdentifier;
{endregion}
  aEntry            : ARRAY[1 .. XBA_Param.nSched_EntryCount] OF ST_BA_SchedEntry;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.7.11.2 ST_BA_SchedException

```

TYPE ST_BA_SchedException :
STRUCT
  eType      : E_BA_DateValChoice := E_BA_DateValChoice.Invalid;
  uDate      : U_BA_DateVal;
  aEntry     : T_BA_SchedExceptionEntryList;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.7.11.3 T_BA_SchedCalendar

```

TYPE T_BA_SchedCalendar : ARRAY[1 .. XBA_Param.nSched_CalendarCount] OF ST_BA_SchedCalendar; END_TYP
E

```

6.1.2.1.1.2.7.11.4 T_BA_SchedExceptionEntryList

```

TYPE T_BA_SchedExceptionEntryList : ARRAY[1 .. XBA_Param.nSched_EntryCount] OF ST_BA_SchedEntry; END
_TYPE

```

6.1.2.1.1.2.7.11.5 T_BA_SchedExceptionList

```

TYPE T_BA_SchedExceptionList : ARRAY[1 .. XBA_Param.nSched_ExceptionCount] OF ST_BA_SchedException;
END_TYPE

```

6.1.2.1.1.2.7.11.6 T_BA_SchedWeek

```

TYPE T_BA_SchedWeek : ARRAY[E_BA_Weekday.First .. E_BA_Weekday.Last, 1 .. XBA_Param.nSched_EntryCoun
t] OF ST_BA_SchedEntry; END_TYPE

```

6.1.2.1.1.2.7.12 Trend

6.1.2.1.1.2.7.12.1 T_BA_TrendLogBuffer

```

TYPE T_BA_TrendLogBuffer : ARRAY [1 .. XBA_Param.nTrend_BufferSize] OF ST_BA_TrendEntry; END_TYPE

```

6.1.2.1.1.2.8 Stepped

6.1.2.1.1.2.8.1 ST_BA_ActiveInfo

```

TYPE ST_BA_ActiveInfo :
STRUCT
  iReference      : I_BA_Object;

```

```

{attribute 'TcEncoding':='UTF-8'}
sSymbolName      : REFERENCE TO STRING;
{attribute 'TcEncoding':='UTF-8'}
sDescription      : REFERENCE TO T_MaxString;
iBlockingProcess  : I_BA_Process;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.8.2 ST_BA_PlantAggregateReference

```

TYPE ST_BA_PlantAggregateReference :
STRUCT
    iReference      : I_BA_Aggregate;
    aAggMode        : ARRAY [1 .. XBA_Param.nPlantCtrl_OpModeCount] OF UDINT;
{region 'Step Conditions'}
    bWaitForProcesses : BOOL := FALSE;
    bWaitForEvents    : BOOL := FALSE;
    tDelayStepDown    : TIME;
    tDelayStepUp      : TIME;
{endregion}
END_STRUCT
END_TYPE

```

Name	Type	Description
bWaitForProcesses	BOOL	Active processes that are taken into account when approaching steps.
bWaitForEvents	BOOL	Events of this (and a higher) lock priority, which are taken into account when changing steps.

6.1.2.1.1.2.8.3 ST_BA_PlantOperation

```

TYPE ST_BA_PlantOperation :
STRUCT
    aOpMode      : ARRAY [1 .. XBA_Param.nPlantCtrl_OpModeCount] OF UDINT;
    aPriority     : ARRAY [1 .. XBA_Param.nPlantCtrl_OpModeCount] OF E_BA_Priority := [ XBA_Param.nPlantCtrl_OpModeCount(E_BA_Priority.eDefault) ];
    aIgnoreFlags : ARRAY [1 .. XBA_Param.nPlantCtrl_OpModeCount] OF E_BA_AggregateIgnoreFlags := [ XBA_Param.nPlantCtrl_OpModeCount(E_BA_AggregateIgnoreFlags.None) ];
    aAggregates  : ARRAY [1 .. XBA_Param.nPlantCtrl_AggregateCount] OF ST_BA_PlantAggregateReference;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.9 Supplements

6.1.2.1.1.2.9.1 ST_BA_SupplementRef

```

TYPE ST_BA_SupplementRef :
STRUCT
    iInstance      : I_BA_SupplementObject;
    eType          : E_BA_SupplementType := E_BA_SupplementType.Invalid;
END_STRUCT
END_TYPE

```

6.1.2.1.1.2.9.2 BACnet

6.1.2.1.1.2.9.2.1 ST_BA_BACnet_PropSyncInfo

```

TYPE ST_BA_BACnet_PropSyncInfo :
STRUCT
    eProperty      : E_BACnetPropIdentifier;
    eSyncMode      : E_BA_ParamSyncMode := E_BA_ParamSyncMode.Invalid;
END_STRUCT
END_TYPE

```

6.1.2.1.2 GVLs

6.1.2.1.2.1 XBA_Globals

```

VAR_GLOBAL
{region 'General'}
    Diag                                     : FB_BA_Diagnosis;

    Top                                     : FB_BA_TopView;
{endregion}

{region 'Indicators'}
    nIncObjInitial                         : UINT    := 1;
    nIncObjActivePriority                  : UINT    := 1;
    nIncObjStatus                         : UINT    := 1;
    nIncEvent                             : UDINT   := 1;
    nIncEventConfig                       : UINT    := 1;
{endregion}
END_VAR

VAR_GLOBAL CONSTANT
{region 'Constants'}
    {region 'General'}
        guidUndefined                     : GUID      := (Data1:=16#0, Data2:=16#0, Data3:=16#0, Data4:=[16#0
, 16#0, 16#0, 16#0, 16#0, 16#0, 16#0, 16#0]);

        nNoActivePrio                     : UDINT    := 16#FFFFFFF;
    {endregion}
    {region 'Text generation'}
        sPlaceholderSign_Open             : STRING(1) := '{';
        sPlaceholderSign_Close            : STRING(1) := '}';
        sPlaceholderSign_Delimiter        : STRING(1) := ',';
        sPlaceholderSign_DenyConcat       : STRING(1) := '!';
        sPlaceholderSign_Empty            : STRING(2) := CONCAT(sPlaceholderSign_Open, sPlaceholderSign_Cl
ose);
        stUndefinedIdentifier              : ST_BA_ObjectIdentifier := (eObjectType:=E_BA_ObjectType.Undefi
ned, nInstanceID:=0);
    {endregion}
    {region 'Event'}
        //Acknowledgement
        aAckFlags_None                    : T_BA_EventTransitions := F_BA_EventTransition(FALSE, FALSE, FA
LSE);

        // Pre-defined indicator filters:
        aIndFilter_None                   : T_BA_EventConditionFlags := [ E_BA_EventCondition.Count(TRUE)
];
        aIndFilter_EvtAll                  : T_BA_EventConditionFlags := [ E_BA_EventType.Count(TRUE) ];
        aIndFilter_Evt                     : ARRAY[E_BA_EventType.First .. E_BA_EventType.Last] OF T_BA_Eve
ntConditionFlags := [
            (* eAlarm                      *) [ TRUE,  FALSE,  FALSE,  FALSE,  FALSE ],
            (* eDisturb                     *) [ FALSE,  TRUE,   FALSE,  FALSE,  FALSE ],
            (* eMaintenance                 *) [ FALSE,  FALSE,  TRUE,   FALSE,  FALSE ],
            (* eNotification                *) [ FALSE,  FALSE,  FALSE,  TRUE,   FALSE ],
            (* eOther                      *) [ FALSE,  FALSE,  FALSE,  FALSE,  TRUE ]
        ];
    {endregion}
{endregion}
END_VAR

```


Name	Type	Description
Diag	FB_BA_Diagnosis	Provides information about the project and offers information and diagnostic options: • Current system time • Set cycle times • Libraries and BACnet drivers used • Number of BACnet objects • State of the Publishers and Subscribers • Project structure output • Output of the current present and historical events.
Top	FB_BA_TopView	Root object or top object of the project structure. Collects information of all children objects.
nIncObjInitial	UINT	Indicates that properties of an object [► 28] have changed, which usually change only rarely.
nIncObjActivePriority	UINT	Indicates that the active priority of an object [► 28] has changed.
nIncObjStatus	UINT	Indicates that the state of an object [► 28] has changed.
nIncEvent	UINT	Indicates the change of an event [► 28].
nIncEventConfig	UINT	Indicates that the event configuration [► 28] of an object [► 28] has changed.
guidUndefined	GUID	Undefined system ID
nNoActivePrio	UDINT	Value of the constant indicates that no priority is active.
sPlaceholderSign_Open	STRING(1)	Start character for a placeholder.
sPlaceholderSign_Close	STRING(1)	End character for a placeholder.
sPlaceholderSign_Delimiter	STRING(1)	Separator [► 38] for placeholder attributes [► 37].
sPlaceholderSign_DenyConcat	STRING(1)	Characters [► 38] for placeholders [► 36] not to be concatenated.
sPlaceholder_Empty	STRING(2)	Empty placeholders [► 36].
stUndefinedIdentifier	ST_BA_ObjectIdentifier [► 110]	Value representing an undefined object [► 28] reference.
aAckFlags_None	T_BA_EventTransitions [► 107]	Value, which represents no active AcknowledgeRequired flags.
aIndFilter_None	T_BA_EventConditionFlags [► 107]	Value, after which no event conditions [► 92] are filtered.
aIndFilter_EvtAll	T_BA_EventConditionFlags [► 107]	Value, to filter all event conditions [► 92].
aIndFilter_Evt	T_BA_EventConditionFlags [► 107]	Value, to filter _selected event types (Grouped by event types).

6.1.2.1.2.2 XBA_Param

```

VAR_GLOBAL CONSTANT
{
  region 'Supplement-Management'
    bEnableBACnet                : BOOL := TRUE;
    nEnabledSupplements          : INT(0..2) := 1;
  }
{endregion}
{region 'Communication-Management'}
  {IF defined (BaDebug)}
    nCom_BACnetRM_IOCount       : DINT := 100;
  {ELSE}
    nCom_BACnetRM_IOCount       : DINT := 0;
  }
}

```

```

    {END_IF}
  {endregion}
  {region 'Site service'}
    bSiteServer_Enable          : BOOL := TRUE;
    nSiteServer_BufferSize      : UINT := 8192;
    nSiteServer_SessionTimeout  : TIME := T#15S;
    nSiteClient_BufferSize      : UINT := 1024;
    tSiteClient_ReadTimeout     : TIME := T#5S;
  {endregion}
  {region 'Project settings'}
    {region 'General'}
      {IF defined (BaDebug)}
        eLanguage                : E_BA_Language := E_BA_Language.eGerman;
      {ELSE}
        eLanguage                : E_BA_Language := E_BA_Language.eEnglish;
      {END_IF}

      bUtf8AutoConvert           : BOOL          := TRUE;
    {endregion}
    {region 'DPAD'}
      nDPAD_Levels               : UINT          := 10;

      nDPAD_DefIndexDigits       : UINT          := 2;
      {attribute 'TcEncoding':='UTF-8'}
      sDPAD_ObjectName_DefSeparator : T_BA_ShortString := '_';
      {attribute 'hide'}
      eDPAD_ObjectName_ManOvrConcatMode : E_BA_ConcatDPADMode := E_BA_ConcatDPADMode.eNone;
      {attribute 'TcEncoding':='UTF-8'}
      sDPAD_Description_DefSeparator : T_BA_ShortString := ' - ';
      {attribute 'hide'}
      eDPAD_Description_ManOvrConcatMode : E_BA_ConcatDPADMode := E_BA_ConcatDPADMode.eNone;
      bDPAD_Description_ExplicitIndex : BOOL      := TRUE;
    {endregion}
  {endregion}

  {region 'Event'}
    {region 'Management'}
      eEvtMgmt_AckMode           : E_BA_AcknowledgeMode := E_BA_AcknowledgeMode.eSingle

      {attribute 'TcEncoding':='UTF-8'}
      sEvtMgmt_AckMsgInternal     : STRING := 'Built-in acknowledgement.';
      {attribute 'TcEncoding':='UTF-8'}
      sEvtMgmt_AckMsgRemote      : STRING := 'Acknowledged by remote user.';
      {attribute 'TcEncoding':='UTF-8'}
      sEvtMgmt_AckMsgPLC         : STRING := 'Acknowledged by PLC.';
    {endregion}
    {region 'Alarm-Mode settings'}
      aAckFlags_Simple           : T_BA_EventTransitions := [ TRUE, TRUE, FALSE ];
      aAckFlags_Standard         : T_BA_EventTransitions := [ TRUE, TRUE, FALSE ];
      aAckFlags_Extended         : T_BA_EventTransitions := [ TRUE, TRUE, TRUE ];

      aEventEn_Simple            : T_BA_EventTransitions := [ TRUE, TRUE, FALSE ];
      aEventEn_Standard          : T_BA_EventTransitions := [ TRUE, TRUE, FALSE ];
      aEventEn_Extended          : T_BA_EventTransitions := [ TRUE, TRUE, TRUE ];
    {endregion}

    nEventHistory_EntryCount     : INT := 2048;

    nEventTransitionText_Length  : DINT := 24;
    aEventTransitionText         : T_BA_EventTransitionText := [ 'To Offnormal', 'To Fault'
, 'To Normal' ];
  {endregion}

  {region 'Event-List'}
    nEventList_EntryCount       : DINT := 512;
  {endregion}

  {region 'Parameters'}
    {region 'General'}
      nInstID_AutoGenerateOffset : UDINT := 100

      nStateText_Length          : DINT := 40;
      nTag_Length                : DINT := 8;

      aDefReadAccess             : ARRAY [E_BA_Parameter.First .. E_BA_Parameter.Last] OF E
_BA_Role := [E_BA_Parameter.Count(0)];
      aDefWriteAccess            : ARRAY [E_BA_Parameter.First .. E_BA_Parameter.Last] OF E
_BA_Role := [E_BA_Parameter.Count(0)];
    {endregion}
    {region 'Multistate'}

```

```

    nMultistate_StateCount      : DINT := 20;
  {endregion}

  {region 'Local'}
    {region 'Hardware'}
      fInput_DefResolution      : REAL := 0.1;
      fInput_DefScaleOffset     : REAL := 0;
      fOutput_DefResolution     : REAL := 0.00305185;
      fOutput_DefScaleOffset    : REAL := 0;

      eInput_DefSensor          : E_BA_MeasuringElement := E_BA_MeasuringElement.eNI1000;
    {endregion}
    {region 'Event config'}
      fDefLimitDeadband        : REAL := 0.0;
      nDefTimeDelay_ToAbnormal  : UDINT := 1;
      nDefTimeDelayAO_ToAbnormal : UDINT := 1;
      nDefTimeDelayBO_ToAbnormal : UDINT := 30;
      nDefTimeDelayMO_ToAbnormal : UDINT := 30;
    {endregion}
    {region 'Value'}
      fDefCOVIncrement         : REAL := 0.1;
    {endregion}

    {region 'Plant Control'}
      nPlantCtrl_OpModeCount    : DINT := 16;
      nPlantCtrl_AggregateCount : DINT := 16;
    {endregion}
    {region 'Sequence Link'}
      nSeqLink_RefCount         : DINT := 16;
    {endregion}
    {region 'Collector'}
      nCollect_RefCount         : DINT := 16;
    {endregion}

    {region 'Loop'}
      nLoop_DefOpMode           : E_BA_PIDMode := E_BA_PIDMode.eP1ID;
    {endregion}
    {region 'Trend'}
      nTrend_BufferSize         : UDINT := 500;
      stTrend_DefStartTime      : ST_BA_DateTime := ();
      stTrend_DefStopTime      : ST_BA_DateTime := ();
      bTrend_DefStopOnFull      : BOOL := FALSE;
      nTrend_DefLogInterval     : UDINT := 900;
      nTrend_DefNotificationThreshold : UDINT := 50;
      eTrend_DefLoggingType     : E_BA_LoggingType := E_BA_LoggingType.ePolled;
    {endregion}
    {region 'Calendar'}
      nCal_EntryCount           : DINT := 24;
    {endregion}
    {region 'Scheduler'}
      nSched_EntryCount         : DINT := 6;
      nSched_CalendarCount      : DINT := 3;
      nSched_ExceptionCount     : DINT := 24;
    {endregion}
    {endregion}
    {region 'Simulation'}
      nSim_AISen_DefDampConstant : UDINT := 20;
    {endregion}
  {endregion}
  {region 'Publish and Subscribe'}
    {region 'Subscribers'}
      tSub_ReadTolerance        : TIME := T#0S;
      bSub_ClearOnReadError     : BOOL := FALSE;
      tSub_DefReadInterval      : TIME := T#30S;
    {endregion}
  {endregion}
  {region 'Groups'}
    nGroupCmd_RefCount          : DINT := 5;
    nGroupDsp_RefCount          : DINT := 5;
    nGroupVal_RefCount          : DINT := 5;
  {endregion}
END_VAR

```

Name	Type	Description
bSiteServer_Enable	BOOL	Disables / enables the <i>Site Server</i> on the automation station.
nSiteServer_BufferSize	UINT	Defines the size of the Site Server communication buffer.
nSiteServer_SessionTimeout	TIME	Defines the maximum duration [s] of a session until it is terminated by the server after inactivity.
nSiteClient_BufferSize	UINT	Defines the size of the communication buffer from the Site Client.
tSiteClient_ReadTimeout	TIME	Defines the maximum duration [s] of read requests of the Site Client until they are aborted with a timeout.
eLanguage	E_BA_Language	Language used within the PLC, for example to format Current values.
bUtf8AutoConvert	BOOL	Automatically applies UTF-8 encoding to strings.
nDPADLevels	UINT	Sets the maximum number of levels in DPAD.
nDPAD_DefIndexDigits	UINT	Sets the number of digits for indexing levels in DPAD.
sDPAD_ObjectName_DefSeparator	T_BA_ShortString	Sets the separator for separating object names.
eDPAD_ObjectName_ManOvrConcatMode	E_BA_ConcatDPADMode [► 96]	Default to concatenate strings when an object name has been manually overwritten.
sDPAD_Description_DefSeparator	T_BA_ShortString	Sets the separator for separating descriptions.
eDPAD_Description_ManOvrConcatMode	E_BA_ConcatDPADMode [► 96]	Default for concatenate strings when a description has been manually overwritten.
bDPAD_Description_ExplicitIndex	BOOL	Conditions under which an index should be displayed in the parameter <i>Description</i> . - FALSE displays the index automatically if > 1. - TRUE displays the index only if it is explicitly defined (placeholder).
eEvtMgmt_AckMode	E_BA_AcknowledgeMode [► 92]	Behavior when acknowledging <u>events</u> [► 28].
sEvtMgmt_AckMsgInternal	STRING	Text for displaying integrated acknowledgement functions (for internal functions).
sEvtMgmt_AckMsgRemote	STRING	Text for displaying acknowledgement functions by external access.
sEvtMgmt_AckMsgPLC	STRING	Text for displaying acknowledgement functions by PLC logic.
aAckFlags_Simple	T_BA_EventTransitions [► 107]	Defines the <u>transitions</u> [► 28] to be confirmed for the "Simple" alarm mode.
aAckFlags_Standard	T_BA_EventTransitions [► 107]	Defines the <u>transitions</u> [► 28] to be confirmed for the "Standard" alarm mode.
aAckFlags_Extended	T_BA_EventTransitions [► 107]	Defines the <u>transitions</u> [► 28] to be confirmed for the "Advanced" alarm mode.
aEventEn_Simple	T_BA_EventTransitions [► 107]	Defines <u>transition states</u> [► 28] to be taken into account for the "Simple" alarm mode.
aEventEn_Standard	T_BA_EventTransitions [► 107]	Defines <u>transition states</u> [► 28] to be taken into account for the "Standard" alarm mode.
aEventEn_Extended	T_BA_EventTransitions [► 107]	Defines <u>transition states</u> [► 28] to be taken into account for the "Advanced" alarm mode.

Name	Type	Description
nEventHistory_EntryCount	INT	Maximum number of entries in the event history.
nEventTransitionText_Length	DINT	Maximum number of letters in the event transition parameter.
aEventTransitionText	T_BA_EventTransitionText [► 107]	Default value for the event transition text.
nEventList_EntryCount	DINT	Maximum number of entries displayed in event lists.
nInstID_AutoGenerateOffset	DINT	Initial value for autogeneration of instance IDs.
nStateText_Length	DINT	Maximum number of characters in the State Text parameter.
nTag_Length	DINT	Maximum amount of letters in the Tag parameter.
aDefReadAccess	E_BA_Parameter [► 101]	Possibility to customize <u>default access rights</u> [► 33] for read access to parameters.
aDefWriteAccess	E_BA_Parameter [► 101]	Possibility to customize <u>default access rights</u> [► 33] for write access to parameters.
nMultistate_StateCount	DINT	Sets the number of states for multi-state values.
fInput_DefResolution	REAL	Default value for the <i>Resolution</i> parameter of inputs.
fInput_DefScaleOffset	REAL	Default value for the <i>Offset</i> parameter of inputs.
fOutput_DefResolution	REAL	Default value for the <i>Resolution</i> parameter of outputs.
fOutput_DefScaleOffset	REAL	Default value for the <i>Offset</i> parameter of outputs.
eInput_DefSensor	E_BA_MeasuringElement	Selection of the sensor with the special input type FB_BA_AI_IOEx [► 127].
fDefLimitDeadband	REAL	Default value for the <i>Limit Deadband</i> parameter.
nDefTimeDelay_ToAbnormal	UDINT	Default value for the <i>Time Delay</i> parameter of transitions to abnormal states.
nDefTimeDelayAO_ToAbnormal	UDINT	Default value for the <i>Time Delay</i> parameter of transitions to abnormal states of analog outputs.
nDefTimeDelayBO_ToAbnormal	UDINT	Default value for the <i>Time Delay</i> parameter of transitions to abnormal states of binary outputs.
nDefTimeDelayMO_ToAbnormal	UDINT	Default value for the <i>Time Delay</i> parameter of transitions to abnormal states of multi-stage outputs.
fDefCOVIncrement	REAL	Default value for the <i>COV increment</i> parameter.
nPlantCtrl_OpModeCount	DINT	Sets the maximum number of <u>plant operation modes</u> [► 17] of FB_BA_PlantCtrl function blocks.
nPlantCtrl_AggregateCount	DINT	Sets the maximum number of aggregate references of FB_BA_PlantCtrl function blocks.
nSeqLink_RefCount	DINT	Maximum number of controller references in a sequence linker.
nCollect_RefCount	DINT	Sets the maximum number of references of FB_BA_Collector function blocks.
eLoop_DefOpMode	E_BA_PIDMode	Default value for the <i>Operation mode</i> parameter.
nTrend_BufferSize	UDINT	Number of entries in a trend buffer. Corresponds at the same time to the maximum size of the recording buffer of all trend objects!
stTrend_DefStartTime	ST_BA_DateTime	Default value for the <i>Start time</i> parameter.

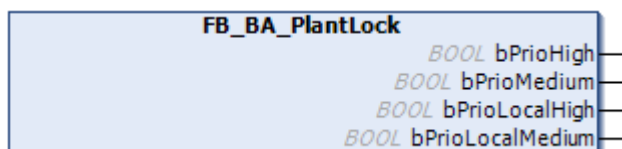
Name	Type	Description
stTrend_DefStopTime	ST_BA_DateTime	Default value for the <i>Stop time</i> parameter.
bTrend_DefStopOnFull	BOOL	Default value for the <i>Stop on full</i> parameter.
nTrend_DefLogInterval	UDINT	Default value for the <i>Logging interval</i> parameter.
nTrend_DefNotificationThreshold	UDINT	Default value for the <i>Notification threshold</i> parameter.
eTrend_DefLoggingType	E_BA_LoggingType	Default value for the <i>Logging type</i> parameter.
nCal_EntryCount	DINT	Number of entries in a calendar object.
nSched_EntryCount	DINT	Number of entries on a weekday ('T_BA_SchedWeek' [► 113]) or an exception ('ST_BA_SchedCalendar' [► 113] / 'ST_BA_SchedException' [► 113]).
nSched_CalendarCount	DINT	Number of calendar references ('T_BA_SchedCalendar' [► 113]).
nSched_ExceptionCount	DINT	Number of exceptions ('T_BA_SchedException').
nSim_AI_Sen_DefDampingConstant	UDINT	Default value for the parameter <i>Damping constant</i> .
tSub_ReadTolerance	TIME	Default value for the <i>Read tolerance</i> parameter.
bSub_ClearOnReadError	BOOL	Default value for the <i>Reset on read errors</i> parameter.
tSub_DefReadInterval	TIME	Default value for the <i>Read interval</i> parameter.
nGroupCmd_RefCount	DINT	Number of references.
nGroupDsp_RefCount	DINT	Number of references.
nGroupVal_RefCount	DINT	Number of references.

6.1.2.1.3 POU's

6.1.2.1.3.1 Function blocks

6.1.2.1.3.1.1 Events

6.1.2.1.3.1.1.1 FB_BA_PlantLock



The events of the objects often signal a relevant malfunction, which requires targeted switching operations to be performed on aggregates or plants. In large plants, there are often a large number of events, which then frequently have to be combined into a collective message.

To keep the programming effort for generating collective messages as low as possible, a *lock priority* can be determined at each event-capable object.

All four lock priorities are collected at each level of the project structure and passed up from the lowest level

of the project structure to the top level.

If required, they are displayed by the instance of a function block *FB_BA_PlantLock* and interconnected with other function blocks for the control of plant and aggregate units.

Four lock priorities are used to differentiate the object events in order to trigger different reactions.

- **Local medium**
Enables a local shutdown of [▶ 15] of medium priority.
- **Local high**
Enables a local shutdown of [▶ 15] of higher priority.
- **Medium**
Enables a higher-level shutdown of [▶ 15] of medium priority.
- **High**
Enables a higher-level shutdown of [▶ 15] of higher priority.

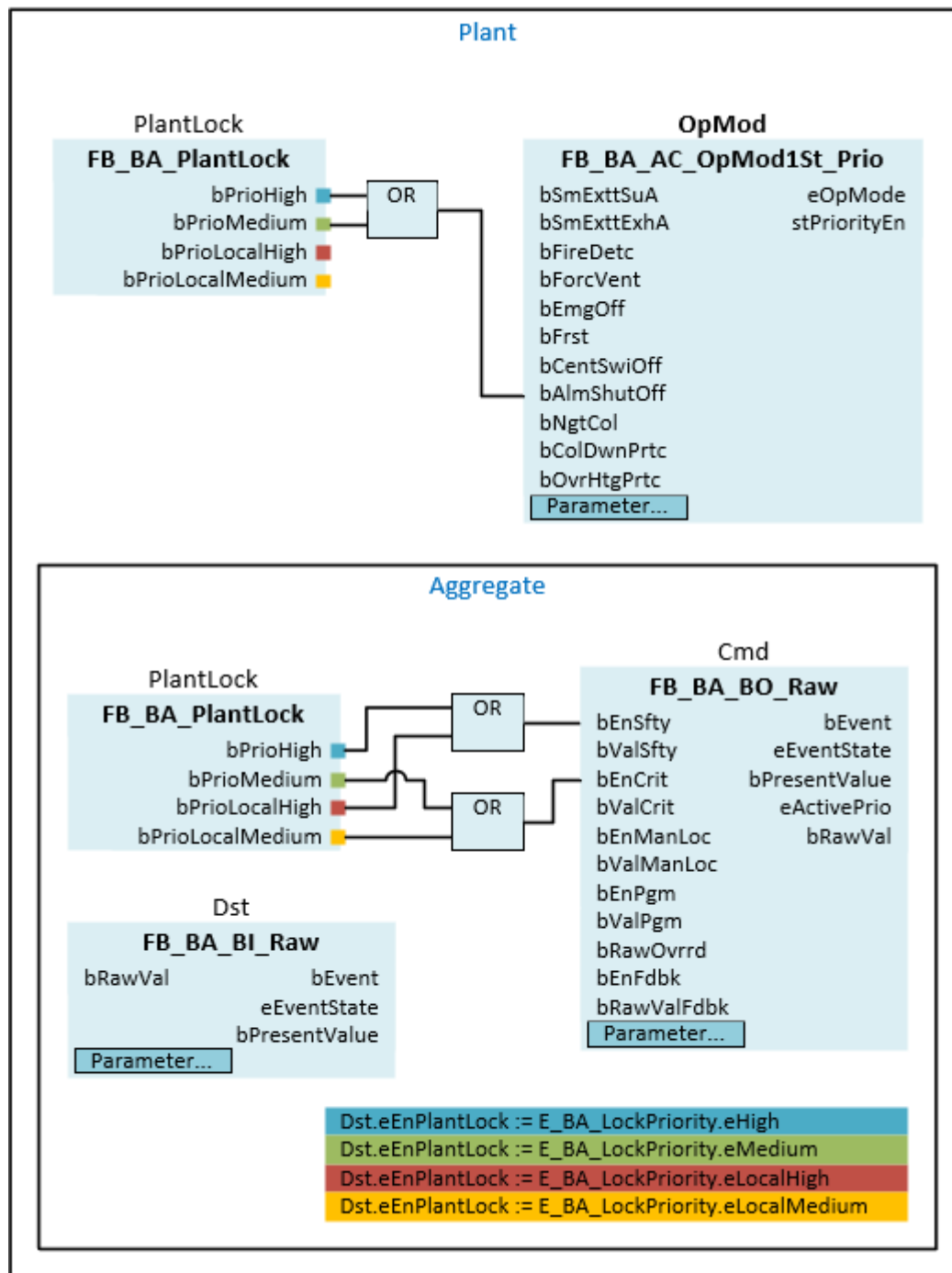
Medium priority is used for technically relevant faults that affect the safety of plants. High priority can be used for faults where the safety of people is at risk.

The local faults should be used, for example, to switch off aggregates. The non-local ones should be used to shut down equipment.

The image below is intended to explain this relationship:

The effects of different lock priorities of the fault input the DST are as follows:

Dst.eEnPlantLock	Cmd.bPresentVal	Cmd.eActivePrio	OpMode.AImShutOff
LocalMedium	False	Critical	False
LocalHigh	False	Safety	False
Medium	False	Critical	Safety
High	False	Safety	Safety



Illustration

```

FUNCTION_BLOCK FB_BA_PlantLock
VAR_OUTPUT
  bPrioHigh      : BOOL;
  bPrioMedium    : BOOL;
  bPrioLocalHigh : BOOL;
  bPrioLocalMedium : BOOL;
END_VAR
VAR
  {region 'Fixed Variables'}
    iParent : I_BA_View;
  {endregion}
END_VAR

```


🔌 Outputs

Name	Type	Description
bPrioLocalHigh	BOOL	The output signals a global event with a high priority.
bPrioMedium	BOOL	The output signals a global event with a medium priority.
bPrioLocalHigh	BOOL	The output signals a local event with a high priority.
bPrioLocalMedium	BOOL	The output signals a local event with a medium priority.

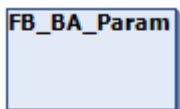
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.1.2 Additional function blocks

6.1.2.1.3.1.2.1 Settings

6.1.2.1.3.1.2.1.1 FB_BA_Param



The function block enables configuration of the properties contained in the global variable list [XBA_Param](#) [► 116].

i Existing limitations that apply to the writing of parameters are observed (see also [Regions](#) [► 91]). For example, the writing of fixed parameters at runtime is rejected. Parameters for the configuration of compiler time values **cannot** be changed at runtime!

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2 Objects

6.1.2.1.3.2.1 Operator objects

Operator objects are used to display or enter a value. For example, to display an internally calculated setpoint, or to enter a setpoint.

The following operator objects are available in the TF8040 TwinCAT Building Automation.

FB	Type	Description
FB_BA_AV_OP	AV	Analog value
FB_BA_BV	BV	Binary value
FB_BA_MV_OP	MV	Multi-state value

If the input variable *ValuePgm* of the function block is written to an operator object by the PLC, it recognizes that it is used to display a value.

When using the operator object as an input object, the value can be specified either by a BACnet client or via TwinCAT on writing to the variable *ValueRm*.

Both when used as an input or output object, the current value of the object is output at the output variable *PresentValue* for further processing.

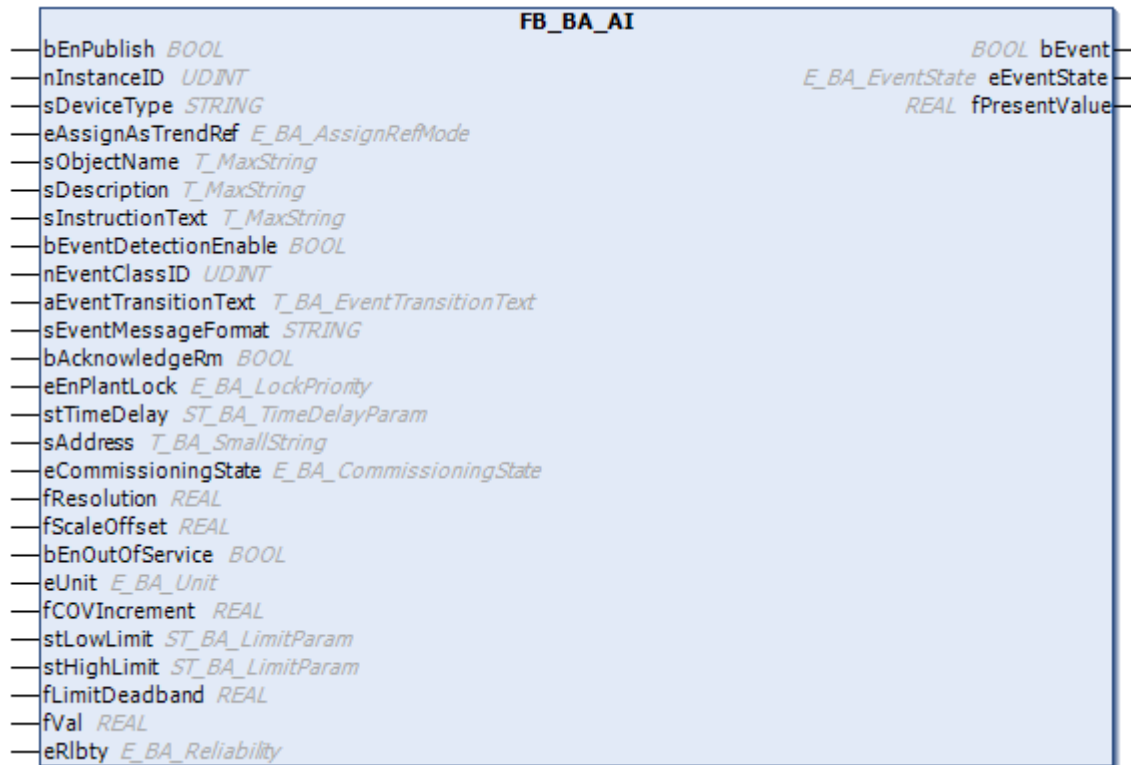
6.1.2.1.3.2.2 Local

The following chapter provides information about the local server objects.

6.1.2.1.3.2.2.1 Analog

Description of the analog function blocks.

6.1.2.1.3.2.2.1.1 FB_BA_AI



The function block FB_BA_AI represents an analog input.

It is used to map a measured value within the project structure by means of an analog input object. The measured value must already exist as a scaled value within the controller. When using this function block, the measuring signal comes, for example, from a fieldbus and not from a bus terminal.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseAI [► 175]

Illustration

```
FUNCTION_BLOCK FB_BA_AI EXTENDS FB_BA_BaseAI
VAR_INPUT
    fVal      : REAL;
    eRlbtty   : E_BA_Reliability;
END_VAR
```

🔧 Inputs

Name	Type	Description
fVal	REAL	Scaled, analog input value.
eRlbtty	E_BA_Reliability	Availability or reliability of a value.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.1.2 FB_BA_AI_IO



The function block FB_BA_AI_IO represents the object for an analog input.

It is used to map a measured value within the project structure.

The process variables for linking the function block to the bus terminal are available within the function block.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - [FB_BA_Object \[► 195\]](#)
 - [FB_BA_EventObject \[► 192\]](#)
 - [FB_BA_EventObjectEx \[► 194\]](#)
 - [FB_BA_ComEventObject \[► 191\]](#)

FB_BA_BaseAI [► 175]

Illustration

```

FUNCTION_BLOCK FB_BA_AI_IO EXTENDS FB_BA_BaseAI IMPLEMENTS I_BA_RawAI
VAR
  {region 'Raw I/O'}
    nRawState    AT %I*    : USINT;
    nRawDataIn   AT %I*    : INT;
  {endregion}
END_VAR

```

VAR

Name	Type	Description
nRawState	USINT	Variable for linking the status information of a terminal.
nRawDataIn	INT	Variable for linking the raw data of an analog input terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.1.3 FB_BA_AI_IOEx

FB_BA_AI_IOEx	
— bEnPublish <i>BOOL</i>	<i>BOOL</i> bEvent
— nInstanceID <i>UDINT</i>	<i>E_BA_EventState</i> eEventState
— sDeviceType <i>T_BA_SmallString</i>	<i>REAL</i> fPresentValue
— eAssignAsTrendRef <i>E_BA_AssignRefMode</i>	
— sObjectName <i>T_MaxString</i>	
— sDescription <i>T_MaxString</i>	
— sTag <i>STRING(BA_Param.nTag_Length)</i>	
— bEventDetectionEnable <i>BOOL</i>	
— aEventEnable <i>T_BA_EventTransitions</i>	
— nEventClassID <i>UDINT</i>	
— aEventTransitionText <i>T_BA_EventTransitionText</i>	
— bAcknowledgeRm <i>BOOL</i>	
— eEnPlantLock <i>E_BA_LockPriority</i>	
— stTimeDelay <i>ST_BA_TimeDelayParam</i>	
— sAddress <i>T_BA_SmallString</i>	
— eCommissioningState <i>E_BA_CommissioningState</i>	
— bEnOutOfService <i>BOOL</i>	
— eUnit <i>E_BA_Unit</i>	
— fCOVIncrement <i>REAL</i>	
— stLowLimit <i>ST_BA_LimitParam</i>	
— stHighLimit <i>ST_BA_LimitParam</i>	
— fLimitDeadband <i>REAL</i>	
— fResolution <i>REAL</i>	
— fScaleOffset <i>REAL</i>	
— eSensor <i>E_BA_MeasuringElement</i>	
— bConfigure <i>BOOL</i>	

The function block FB_BA_AI_IOEx represents the object of an analog input.

The process variables for linking the function block to the bus terminal are created within the function block. The function block also enables configuration of the analog input terminal. The function block FB_BA_KL32xx is used for this purpose.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA EventObject \[► 192\]](#)

[FB BA EventObjectEx \[► 194\]](#)

[FB BA ComEventObject \[► 191\]](#)

[FB BA BaseAI \[► 175\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_AI_IOEx EXTENDS FB_BA_BaseAI
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
eSensor          : E_BA_MeasuringElement := BA_Param.fInput_DefSensor;
{endregion}
END_VAR
VAR_INPUT CONSTANT
{region 'Operational Parameters'}
bConfigure       : BOOL := TRUE;
{endregion}
END_VAR
VAR
{region 'Output-Properties'}
sTerminalType    : STRING;
{endregion}
END_VAR
VAR
{region 'I/O'}
nRawState        AT %I* : USINT;
nRawDataIn       AT %I* : INT;
nRawCtrl         AT %Q* : USINT;
nRawDataOut      AT %Q* : INT;
iSrcIO           : I_BA_RawAI;
{endregion}
END_VAR
```

Inputs CONSTANT PERSISTENT

Name	Type	Description
eSensor	E_BA_MeasuringElement	This input is used to set the sensor. The setting is made by selecting the type in the enumeration.

Inputs CONSTANT

Name	Type	Description
bConfigure	BOOL	Start of the terminal configuration.

VAR

Name	Type	Description
sTerminalType	STRING	Indicates the type of bus terminal used.
nRawState	USINT	Variable for linking the status information of a terminal.
nRawDataIn	INT	Variable for linking the raw data of an analog input terminal.
nRawCtrl	USINT	Variable for linking with the control information of a terminal.
nRawDataOut	INT	Variable for linking the output value of the PLC with the process image of an output terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.1.4 FB_BA_AI_Raw



The function block FB_BA_AI_Raw represents the object of an analog input.

It has input variables for connecting the process values of the terminal.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseAI [► 175]

Illustration

```
FUNCTION_BLOCK FB_BA_AI_Raw EXTENDS FB_BA_BaseAI
VAR_INPUT
    nRawState      : USINT;
    nRawVal        : INT;
END_VAR
```

Inputs

Name	Type	Description
nRawState	USINT	Variable for linking the status information of a terminal.
nRawVal	INT	Raw value

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.1.5 FB_BA_AO



The function block FB_BA_AO is used to map a control output within the project structure. When using this function block, the control signal is not output to a terminal, but to a fieldbus, for example.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_EventObject \[► 192\]](#)

[FB_BA_EventObjectEx \[► 194\]](#)

[FB_BA_ComEventObject \[► 191\]](#)

[FB_BA_BaseAO \[► 177\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_AO EXTENDS FB_BA_BaseAO
VAR_INPUT
    eRlbtty      : E_BA_Reliability;
    bRawOvrrd    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
eRlbtty	E_BA_Reliability	Availability or reliability of a value.
bRawOvrrd	BOOL	The feedback of a switch contained in a terminal can be connected to this variable for mechanical override of an output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.1.6 FB_BA_AO_IO



The function block FB_BA_AO_IO represents an analog output object within the base framework. The variables for linking the control output to the terminal are declared within the function block.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_EventObject \[► 192\]](#)

[FB_BA_EventObjectEx \[► 194\]](#)

[FB_BA_ComEventObject \[► 191\]](#)

[FB_BA_BaseAO \[► 177\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_AO_IO EXTENDS FB_BA_BaseAO IMPLEMENTS I_BA_RawAO
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    eOverriddenPolarity    : E_BA_Polarity := E_BA_Polarity.eNormal;
{endregion}
END_VAR
VAR
{region 'Raw I/O'}
    bRawOverridden        AT %I*      : BOOL;
    nRawState              AT %I*      : USINT;
    nRawDataOut            AT %Q*      : INT;
{endregion}
END_VAR
```

Inputs CONSTANT PERSISTENT

Name	Type	Description
eOverriddenPolarity	E_BA_Polarity	Output terminals with mechanical priority operation report the state of their switches back to the controller. With this enumeration the polarity of the switch feedback can be parameterized.

VAR

Name	Type	Description
bRawOverridden	BOOL	Variable for detecting an override from the outside.
nRawState	USINT	Variable for linking the status information of a terminal.
nRawDataOut	INT	Variable for linking the output value of the PLC with the process image of an output terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.1.7 FB_BA_AO_Raw



The function block **FB_BA_AO_Raw** represents an analog output object within the base framework. The variable `nRawVal` is used to link the control value with the process image of the terminal.

It has a priority array and can therefore be commanded.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_EventObject \[► 192\]](#)

[FB_BA_EventObjectEx \[► 194\]](#)

[FB_BA_ComEventObject \[► 191\]](#)

[FB_BA_BaseAO \[► 177\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_AO_Raw EXTENDS FB_BA_BaseAO
VAR_INPUT
    bRawOverridden      : BOOL;
END_VAR
```

```
VAR_OUTPUT
  nRawVal          : INT;
END_VAR
```

Inputs

Name	Type	Description
bRawOverridden	BOOL	To this variable can be connected the feedback of a switch contained in a terminal, for the mechanical override of an output.

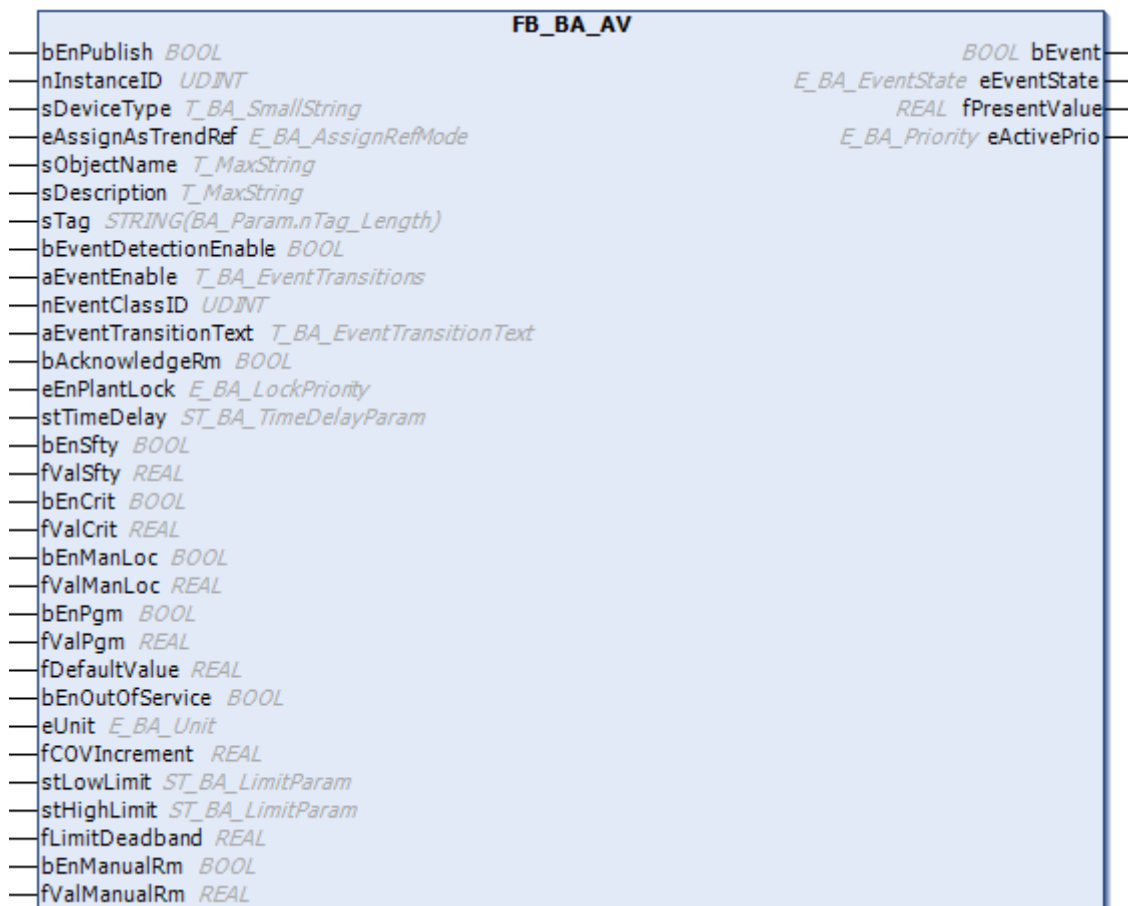
Outputs

Name	Type	Description
nRawVal	INT	The variable is used to link the raw value of an object with the process image of the input or output terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.1.8 FB_BA_AV



The function block FB_BA_AV represents an analog Value object.

It has a priority array and can be commanded.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

Illustration

```
FUNCTION_BLOCK FB_BA_AV EXTENDS FB_BA_EventObjectEx IMPLEMENTS I_BA_AnalogPrioObject, I_BA_AnyValue
VAR_INPUT
    bEnSfty          : BOOL;
    fValSfty         : REAL;
    bEnCrit          : BOOL;
    fValCrit         : REAL;
    bEnManLoc        : BOOL;
    fValManLoc       : REAL;
    bEnPgm           : BOOL;
    fValPgm          : REAL;
END_VAR
VAR_OUTPUT
    fPresentValue    : REAL;
    eActivePrio      : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        fDefaultValue      : REAL;
        bEnOutOfService    : BOOL;
        eUnit               : E_BA_Unit := E_BA_Unit.Invalid;
        fCOVIncrement      : REAL := BA_Param.fDefCOVIncrement;
        stLowLimit         : ST_BA_LimitParam;
        stHighLimit        : ST_BA_LimitParam;
        fLimitDeadband     : REAL := BA_Param.fDefLimitDeadband;
    {endregion}
    {region 'Operational Parameters'}
        bEnManualRm        : BOOL;
        fValManualRm       : REAL;
    {endregion}
END_VAR
```

🔧 Inputs

Name	Type	Description
bEnSfty	BOOL	Enabling the "Safety" priority.
fValSfty	REAL	Analog value for the "Safety" priority.
bEnCrit	BOOL	Enabling the "Critical" priority.
fValCrit	REAL	Analog value for the "Critical" priority.
bEnManLoc	BOOL	Enabling the "Manual Local" priority.
fValManLoc	REAL	Analog value for the "Manual Local" priority.
bEnPgm	BOOL	Enabling the "Program" priority.
fValPgm	REAL	Analog value for the "Program" priority.

🔧 Outputs

Name	Type	Description
fPresentValue	REAL	Current analog output value of the object.
eActivePrio	E_BA_Priority	Active priority

🚩 Inputs CONSTANT PERSISTENT

Name	Type	Description
fDefaultValue	REAL	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
eUnit	E_BA_Unit	Unit of the input or output value of an analog object.
fCOVIncrement	REAL	The variable specifies the minimum change in present value that will cause a COV notification to be issued to subscribed COV clients. This property is required if COV reporting is supported by this object.
stLowLimit	ST_BA_LimitParam [► 111]	Parameterization of the lower limit value monitoring of an analog object. The variable <i>bEnable</i> must be TRUE to enable limit value monitoring. The variable <i>fValue</i> is used to parameterize the lower limit value.
stHighLimit	ST_BA_LimitParam [► 111]	Parameterization of the upper limit value monitoring of an analog object. The variable <i>bEnable</i> must be TRUE to enable limit value monitoring. The variable <i>fValue</i> is used to parameterize the upper limit value.
fLimitDeadband	REAL	Dead band or hysteresis for the values <i>HighLimit</i> and <i>LowLimit</i> of the limit value monitoring of an analog object.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.
fValManualRm	REAL	Analog value for the "Manual Remote" priority.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.1.9 FB_BA_AV_Op



The function block FB_BA_AV_Op represents an analog Value object. It is used to display or enter an analog value.

If the input variable *fValuePgm* is linked, then the function block automatically recognizes that it is used to display the connected value. In the other case, it is used to enter a setpoint, for example.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```

FUNCTION_BLOCK FB_BA_AV_Op EXTENDS FB_BA_Object IMPLEMENTS I_BA_AnalogOpObject, I_BA_AnyValue
VAR_INPUT
    fValuePgm          : REAL;
END_VAR
VAR_OUTPUT
    fPresentValue      : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        eUnit          : E_BA_Unit := E_BA_Unit.Invalid;
        fCOVIncrement  : REAL := BA_Param.fDefCOVIncrement;
    {endregion}
    {region 'Operational Parameters'}
        fValueRm        : REAL;
    {endregion}
END_VAR
VAR
    {region 'Interface'}
        eValueSource    : E_BA_ProcessSignalSource;
    {endregion}
END_VAR

```

 Inputs

Name	Type	Description
fValuePgm	Real	Output values of the analog object for the "Program" priority.

 Outputs

Name	Type	Description
fPresentValue	REAL	Current analog output value of the object.

 Inputs CONSTANT PERSISTENT

Name	Type	Description
eUnit	E_BA_Unit	Unit of the input or output value of an analog object.
fCOVIncrement	REAL	The variable specifies the minimum change in present value that will cause a COV notification to be issued to subscribed COV clients. This property is required if COV reporting is supported by this object.
fValueRm	REAL	Variable for overwriting an analog object from the HMI.

VAR

Name	Type	Description
eValueSource	E_BA_ProcessSignalSource [► 105]	The variable indicates whether an object of the type FB_BA_..._OP serves as a display or input object. <i>eVarInput</i> = 1 The object is used to display a value. The value is passed to the object at an input within the PLC. <i>eParameter</i> = 2 The object is used to enter a parameter that can be changed by a BACnet client or the TwinCAT HMI.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.2 Binary

Below you will find the description of the binary function blocks.

6.1.2.1.3.2.2.2.1 FB_BA_BI



The function block FB_BA_BI represents a binary input object within the base framework. The variables for linking the input to the terminal are available as input variables on the function block.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseBI [► 180]

Illustration

```
FUNCTION_BLOCK FB_BA_BI EXTENDS FB_BA_BaseBI
VAR_INPUT
    bVal      : BOOL;
    eRlbtty   : E_BA_Reliability;
END_VAR
```

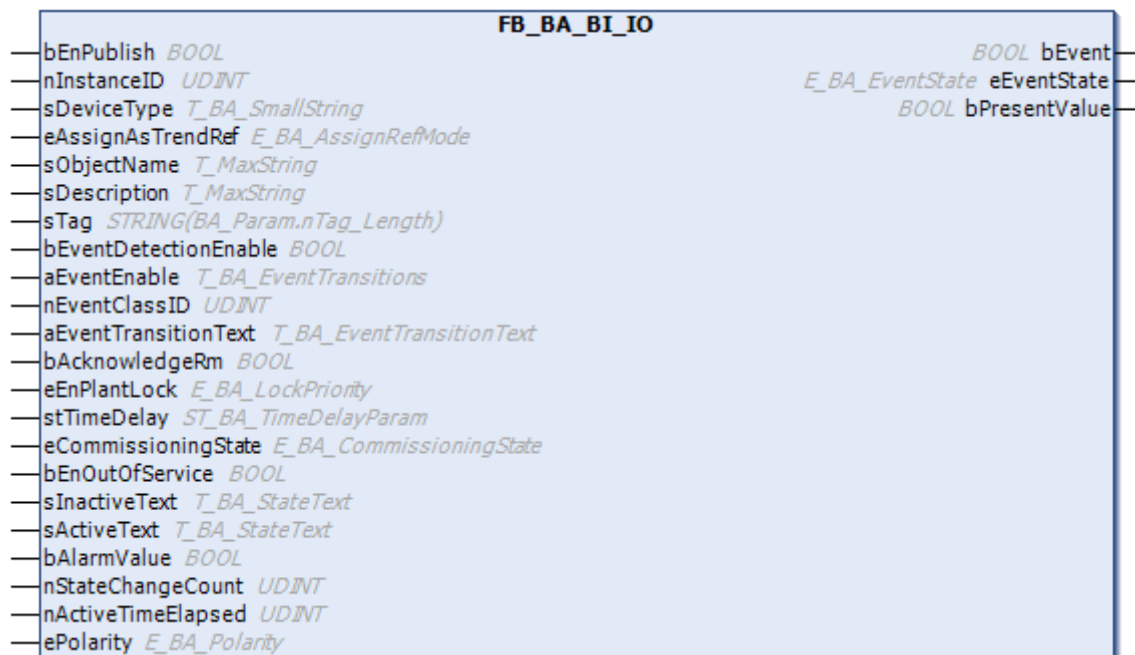
🚩 Inputs

Name	Type	Description
bVal	BOOL	Binary value
eRlbtY	E_BA_Reliability	Availability or reliability of a value.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.2 FB_BA_BI_IO



The function block FB_BA_BI_IO represents a binary input object within the base framework. The variables for linking the input to the terminal are declared within the function block.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseBI [► 180]

FB_BA_BI [► 138]

Illustration

VAR

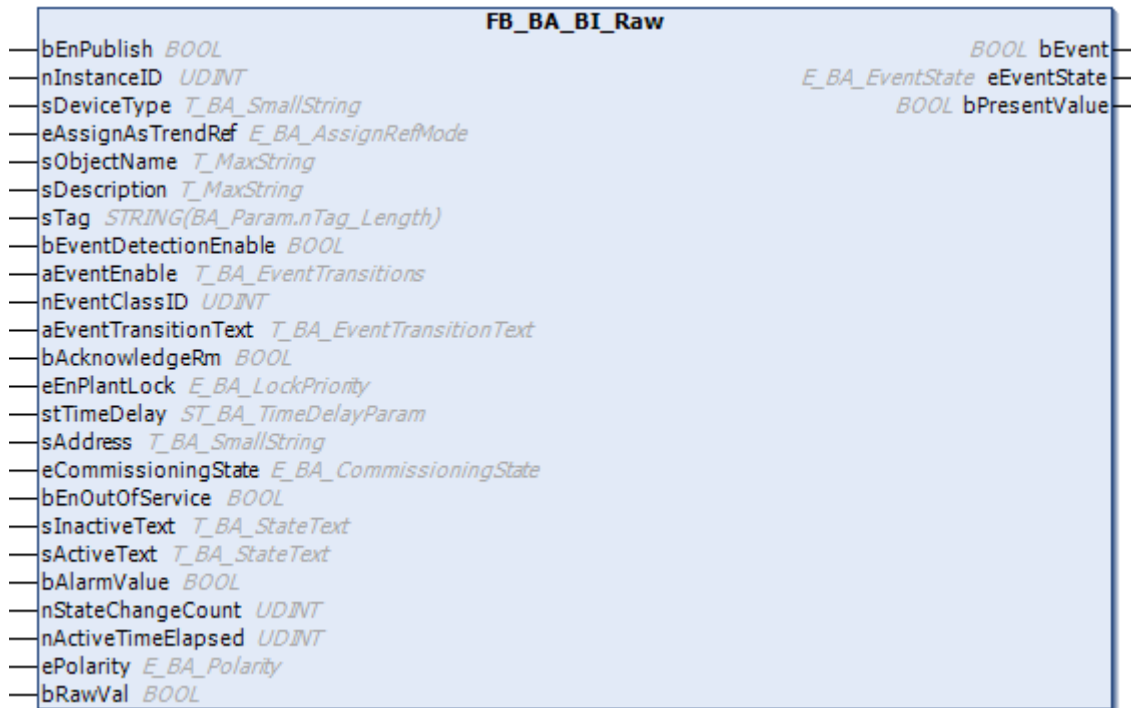
```
{region 'Raw I/O'}
  bRawVal      AT %I* : BOOL;
{endregion}
```


Name	Type	Description
bRawVal	BOOL	Variable for linking the input value to the terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.3 FB_BA_BI_Raw



The function block FB_BA_BI_Raw generates a binary input object within the basic framework. The variable *bRawVal* is used to link the binary input to the terminal.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_EventObject \[► 192\]](#)

[FB_BA_EventObjectEx \[► 194\]](#)

[FB_BA_ComEventObject \[► 191\]](#)

[FB_BA_BaseBI \[► 180\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_BI_Raw EXTENDS FB_BA_BaseBI
VAR_INPUT
    bRawVal      : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bRawVal	BOOL	Raw value

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4 FB_BA_BO



The function block FB_BA_BO represents a binary output object within the base framework. The variables for linking the control output to the terminal are available as input or output variables on the function block.

Inheritance hierarchy

```

FB_BA_Base
├── FB_BA_BasePublisher
│   └── FB_BA_Object [► 195]
    
```

[FB BA EventObject \[► 192\]](#)[FB BA EventObjectEx \[► 194\]](#)[FB BA ComEventObject \[► 191\]](#)[FB BA BaseBO \[► 182\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_BO EXTENDS FB_BA_BaseBO
VAR_INPUT
    eRlbtly          : E_BA_Reliability;
    bRawOvrrd        : BOOL;
    bEnFdbck          : BOOL;
    bRawValFdbck      : BOOL;
END_VAR
```

Inputs

Name	Type	Description
eRlbtly	E_BA_Reliability	Availability or reliability of a value.
bRawOvrrd	BOOL	The feedback of a switch contained in a terminal can be connected to this variable for mechanical override of an output.
bEnFdbck	BOOL	Variable for starting the monitoring of the feedback at the binary output.
bRawValFdbck	BOOL	The variable corresponds to the value of the current feedback.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.5 FB_BA_BO_IO



The function block FB_BA_BO_IO represents a binary output object within the base framework. The variables for linking the switching output to the terminal are declared within the function block. The feedback control of the binary output is automatically activated if the variable *bRawValFeedback* is linked to the process image of a terminal.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseBO [► 182]

Illustration

```
FUNCTION_BLOCK FB_BA_BO_IO EXTENDS FB_BA_BaseBO IMPLEMENTS I_BA_RawBO
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    eFeedbackPolarity      : E_BA_Polarity := E_BA_Polarity.eNormal;
    eOverriddenPolarity    : E_BA_Polarity := E_BA_Polarity.eNormal;
```

```

{endregion}
END_VAR
VAR
  {region 'Raw I/O'}
    bRawOverridden    AT %I*    : BOOL;
    bRawValFeedback   AT %I*    : BOOL;
    bRawVal            AT %Q*    : BOOL;
  {endregion}
END_VAR

```

Inputs Constant Persistent

Name	Type	Description
eFeedbackPolarity	E_BA_Polarity	Variable for parameterizing the polarity of the binary operating feedback of an output.
eOverriddenPolarity	E_BA_Polarity	Output terminals with mechanical priority operation report the state of their switches back to the controller. With this enumeration the polarity of the switch feedback can be parameterized.

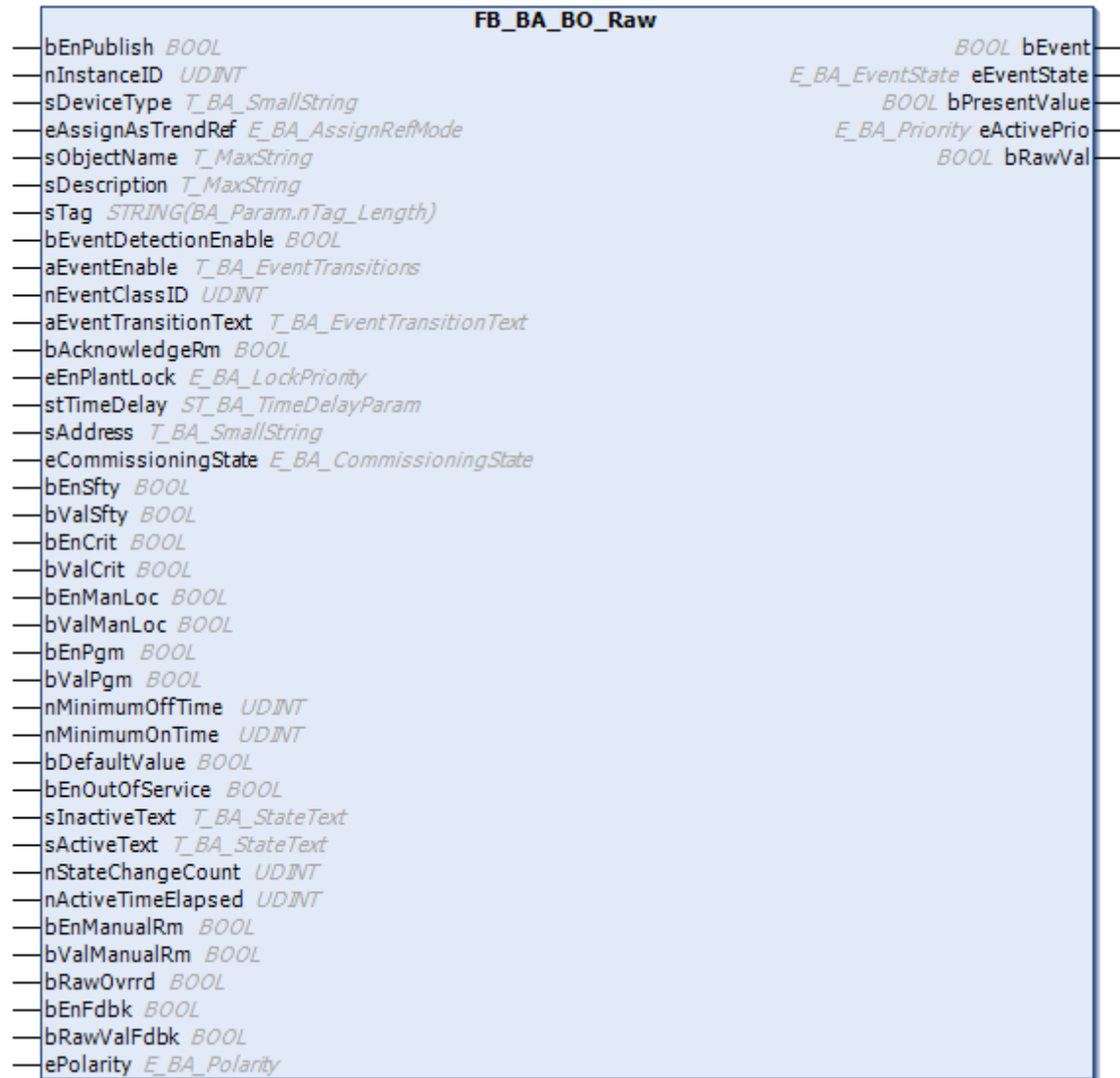
VAR

Name	Type	Description
bRawOverridden	BOOL	Variable for detecting an override from the outside.
bRawValFeedback	BOOL	Activation of feedback control after linking with the bus terminal.
bRawVal	BOOL	Variable for linking the output value to the terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6 FB_BA_BO_Raw



The function block FB_BA_BO_Raw represents a binary output object within the base framework. The variable *bRawVal* is used to link the switching command with the process image of the terminal.

The object has a priority array and can therefore be commanded.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseBO [► 182]

Illustration

```
FUNCTION_BLOCK FB_BA_BO_Raw EXTENDS FB_BA_BaseBO
VAR_INPUT
    bRawOvrrd      : BOOL;
    bEnFdbk        : BOOL;
```

```

    bRawValFdbk      : BOOL;
END_VAR
VAR_OUTPUT
    bRawVal          : BOOL;
END_VAR

```

Inputs

Name	Type	Description
bRawOvrrd	BOOL	The feedback of a switch contained in a terminal can be connected to this variable for mechanical override of an output.
bEnFdbck	BOOL	Variable for starting the monitoring of the feedback at the binary output.
bRawValFdbck	BOOL	The variable corresponds to the value of the current feedback.

Outputs

Name	Type	Description
bRawVal	BOOL	Value of the binary output for linking to the process image of the terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.7 FB_BA_BV



The function block FB_BA_BV represents a binary value object.

It has a priority array and can therefore be commanded.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA EventObject \[► 192\]](#)

[FB BA EventObjectEx \[► 194\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_BV EXTENDS FB_BA_EventObjectEx IMPLEMENTS I_BA_BinaryPrioObject, I_BA_AnyValue
VAR_INPUT
    bEnSfty          : BOOL;
    bValSfty         : BOOL;
    bEnCrit          : BOOL;
    bValCrit         : BOOL;
    bEnManLoc        : BOOL;
    bValManLoc       : BOOL;
    bEnPgm           : BOOL;
    bValPgm          : BOOL;
END_VAR
VAR_OUTPUT
    bPresentValue    : BOOL;
    eActivePrio      : E_BA_Priority;
END_VAR
```



```

VAR_INPUT CONSTANT PERSISTENT
{
  region 'Variable Parameters'
    nMinimumOffTime      : UDINT;
    nMinimumOnTime       : UDINT;
    bDefaultValue        : BOOL;
    bEnOutOfService      : BOOL;
    sInactiveText        : T_BA_StateText;
    sActiveText          : T_BA_StateText;
    bAlarmValue          : BOOL := TRUE;
    nStateChangeCount    : UDINT;
    nActiveTimeElapsed   : UDINT;
  }
endregion
{
  region 'Operational Parameters'
    bEnManualRm          : BOOL;
    bValManualRm         : BOOL;
  }
endregion
END_VAR
VAR
{
  region 'Output-Properties'
    stStateChangeTime    : ST_BA_DateTime;
    stStateChangeResetPoint : ST_BA_DateTime;
    stActiveTimeResetPoint : ST_BA_DateTime;
  }
endregion
END_VAR

```

Inputs

Name	Type	Description
bEnSfty	BOOL	Enabling the "Safety" priority.
bValSfty	BOOL	Binary value for the "Safety" priority.
bEnCrit	BOOL	Enabling the "Critical" priority.
bValCrit	BOOL	Binary value for the "Critical" priority.
bEnManLoc	BOOL	Enabling the "Manual Local" priority.
bValManLoc	BOOL	Binary value for the "Manual Local" priority.
bEnPgm	BOOL	Enabling the "Program" priority.
bValPgm	BOOL	Binary value for the "Program" priority.

Outputs

Name	Type	Description
bPresentValue	BOOL	Current binary output value of the object.
eActivePrio	E_BA_Priority	Active priority

Inputs CONSTANT PERSISTENT

Name	Type	Description
nMinimumOffTime	UDINT	Minimum time [s] in which the <i>Present_Value</i> should remain in the INACTIVE state after a write operation to <i>Present_Value</i> has assumed the INACTIVE state. This can be used to implement protection against too fast restarting.
nMinimumOnTime	UDINT	Minimum time [s] in which the <i>Present_Value</i> is to remain in the ACTIVE state after a write operation to <i>Present_Value</i> has assumed the ACTIVE state. This can be used to implement protection against premature, renewed switch-off
bDefaultValue	BOOL	Value that is assumed if all 16 priorities of the <i>Priority_Array</i> of a commandable object have no entry or are NULL.
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
sInactiveText	T_BA_StateText [► 112]	Text output when the object is inactive.
sActiveText	T_BA_StateText [► 112]	Text output when the object is active.
bAlarmValue	BOOL	Value in the event of an alarm.
nStateChangeCount	UDINT	The variable indicates how often the state of the <i>Present_Value</i> has changed since the date and the last reset.
nActiveTimeElapsed	UDINT	Time [s] in which the <i>Present_Value</i> of the object had the value ACTIVE. The time is valid from the last reset by the property <i>Time_Of_Active_Time_Reset</i> .
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.
bValManualRm	BOOL	Binary value for the "Manual Remote" priority.

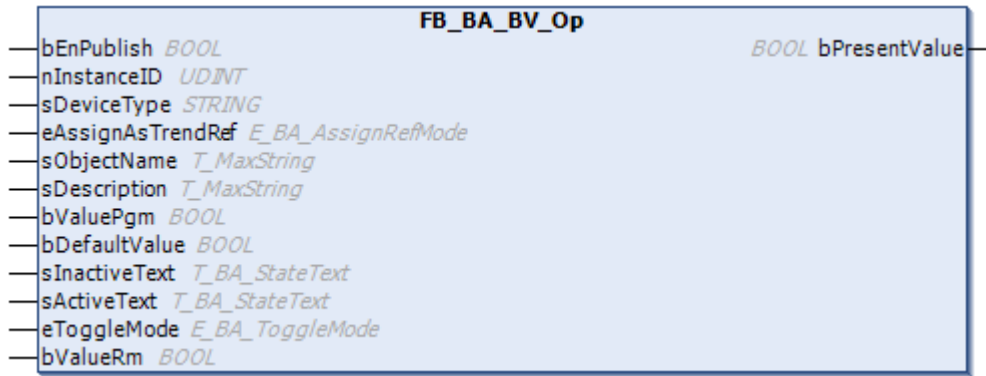
VAR

Name	Type	Description
stStateChangeTime	ST_BA_DateTime	This property shows the time of the last state change. The state change refers to the <i>Present_Value</i> of the object.
stStateChangeResetPoint	ST_BA_DateTime	Shows the date and time from which the counting of state changes started.
stActiveTimeResetPoint	ST_BA_DateTime	Indicates the time when the recording of the object's switch-on times started.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.8 FB_BA_BV_Op



The function block FB_BA_BV_Op represents a binary value object. It is used to display or input a binary value.

If the input variable *bValuePgm* is linked, then the function block automatically recognizes that it is used to display the connected value. In the other case, it is used to enter a binary value.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```
FUNCTION_BLOCK FB_BA_AV_Op EXTENDS FB_BA_Object IMPLEMENTS I_BA_AnalogOpObject, I_BA_AnyValue
VAR_INPUT
    bValuePgm      : BOOL;
END_VAR
VAR_OUTPUT
    bPresentValue  : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    eUnit          : E_BA_Unit := E_BA_Unit.Invalid;
    fCOVIncrement  : REAL := BA_Param.fDefCOVIncrement;
{endregion}
{region 'Operational Parameters'}
    fValueRm       : REAL;
{endregion};
END_VAR
VAR
{region 'Interface'}
    eValueSource    : E_BA_ProcessSignalSource := E_BA_ProcessSignalSource.Invalid;
{endregion}
END_VAR
```

Inputs

Name	Type	Description
bValuePgm	BOOL	Value of a binary object for the "Program" priority.

Outputs

Name	Type	Description
bPresentValue	BOOL	Current binary output value of the object.

🚩 Inputs CONSTANT PERSISTENT

Name	Type	Description
eUnit	E_BA_Unit	Unit of the input or output value of an analog object.
fCOVIncrement	REAL	The variable specifies the minimum change in present value that will cause a COV notification to be issued to subscribed COV clients. This property is required if COV reporting is supported by this object.
fValueRm	REAL	Variable for overwriting an analog object from the HMI.

VAR

Name	Type	Description
eValueSource	E_BA_ProcessSignalSource [► 105]	The variable indicates whether an object of the type FB_BA_..._OP serves as a display or input object. <i>eVarInput</i> = 1 The object is used to display a value. The value is passed to the object at an input within the PLC. <i>eParameter</i> = 2 The object is used to enter a parameter that can be changed by a BACnet client or the TwinCAT HMI.

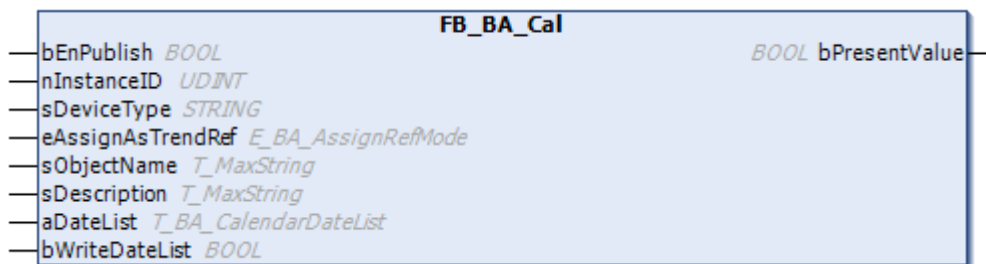
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.3 Misc

Description of the objects Calendar, Event-Class, Controller, Scheduler and Trend.

6.1.2.1.3.2.2.3.1 FB_BA_Cal



The function block FB_BA_Cal represents a calendar within the project structure.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB BA Object [► 195]

Illustration

```
FUNCTION_BLOCK FB_BA_Cal EXTENDS FB_BA_Object IMPLEMENTS I_BA_Cal
VAR_OUTPUT
    bPresentValue      : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        aDateList      : T_BA_CalendarDateList;
```

```

{endregion}
END_VAR
VAR_INPUT CONSTANT
{region 'Variable Parameters'}
  bWriteDateList : BOOL;
{endregion}
END_VAR

```

Outputs

Name	Type	Description
bPresentValue	BOOL	Current binary output value of the object.

Inputs CONSTANT PERSISTENT

Name	Type	Description
aDateList	T BA_CalendarDateList [►_112]	Date list.

Inputs CONSTANT

Name	Type	Description
bWriteDateList	BOOL	Enable to write to a date list.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.17	Tc3_BA2 from v4.8.9.0

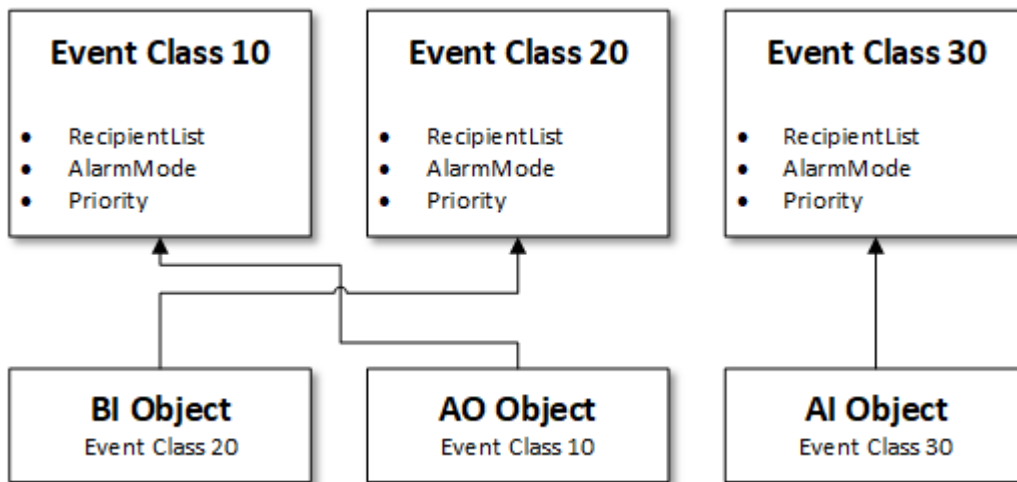
6.1.2.1.3.2.2.3.2 FB_BA_EC

FB_BA_EC	
— bEnPublish	BOOL
— nInstanceID	UDINT
— sDeviceType	STRING
— eAssignAsTrendRef	E_BA_AssignRefMode
— sObjectName	T_MaxString
— sDescription	T_MaxString
— aPriority	ARRAY[E_BA_EventTransition.First..E_BA_EventTransition.Last]OFUDINT
— eEventType	E_BA_EventType
— eAlarmMode	E_BA_AlarmMode
— aAcknowledgeRequired	T_BA_EventTransitions

The function block FB_BA_EC represents an event class (cf. notification class) within the project structure of TF8040.

The detection of an event and the object-internal reporting (intrinsic reporting) is located in the event-enabled objects. The subsequent distribution of the events to the event clients, on the other hand, is not executed in the objects, but in the event class.

Each event-enabled object is assigned an event class. One or more objects can be assigned to an event class.



The event class describes properties of an event. All objects assigned to this event class get these properties.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```

FUNCTION_BLOCK FB_BA_EC EXTENDS FB_BA_Object IMPLEMENTS I_BA_EventClass
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    aPriority          : ARRAY[E_BA_EventTransition.First .. E_BA_EventTransition.Last] OF UDINT
T;
{endregion}
{region 'Fixed Parameters'}
    eEventType         : E_BA_EventType := E_BA_EventType.eOther;
    eAlarmMode         : E_BA_AlarmMode := E_BA_AlarmMode.Invalid;
    aAcknowledgeRequired : T_BA_EventTransitions;
{endregion}
END_VAR
  
```

🔧 Inputs CONSTANT PERSISTENT

Name	Type	Description
aPriority	ARRAY [E_BA_EventTransition.First..E_BA_EventTransition.Last] OF UDINT	The variable specifies the priority with which the event class notifications are transmitted. The priorities range from 0 to 255 inclusive. A lower number means a higher priority.
eEventType	E_BA_EventType [► 94]	This parameter is used to describe an event in more detail. The type of the event also describes the representation in the TwinCAT HMI [► 821] .
eAlarmMode	E_BA_AlarmMode [► 92]	The requirements regarding the acknowledgement and resetting of events are not parameterized on the objects, but on the event classes. Three standard alarm modes are available: The acknowledgement refers to incoming alarms and the reset to outgoing alarms. The alarm mode in TwinCAT Building Automation determines whether an event (or an object) must be acknowledged and / or reset.



The default behavior of individual alarm modes can be adjusted by means of [global parameters](#) [► 116].

This may be necessary if, for example, operators require compliance with certain specifications. All settings are carefully preset.

Changes have a **significant** effect on the event behavior of an object and should be implemented with caution!

Name	Type	Description
aAcknowledgeRequired	T_BA_EventTransitions [► 107]	Acknowledgement required.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.3.3 FB_BA_Loop



The function block FB_BA_Loop represents a PID controller within the project structure of TF8040.

Functional diagram

The controller can be operated either in parallel structure or with upstream P part. This is specified by the input *eOpMode*.

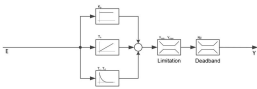
Upstream P part:

eOpMode := E_BA_PIDMode. eP1ID



Parallel structure:

eOpMode := E_BA_PIDMode. ePID



Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```
FUNCTION_BLOCK FB_BA_Loop EXTENDS FB_BA_Object IMPLEMENTS I_BA_Loop
VAR_INPUT
    bEn                : BOOL;
    fSetpoint          : REAL;
    fCtrlVal           : REAL;
    eActionPgm         : E_BA_Action;
    fMinOutputPgm      : REAL;
    fMaxOutputPgm      : REAL;
    bEnSync            : BOOL;
    fValSync           : REAL;
END_VAR
VAR_OUTPUT
    fPresentValue      : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        eOutputUnit    : E_BA_Unit := E_BA_Unit.eOther_Percent;
        fCOVIncrement  : REAL := BA_Param.fDefCOVIncrement;
        eOpMode        : E_BA_PIDMode := BA_Param.nLoop_DefOpMode;
        eActionRm       : E_BA_Action := E_BA_Action.eDirect;
        fNeutralZone    : REAL:= 0;
        fMinOutputRm    : REAL:= 0;
        fMaxOutputRm    : REAL:= 100;
        fProportionalConstant : REAL;
        fIntegralConstant : REAL;
        fDerivativeConstant : REAL := 0;
        nDampConstant   : UDINT;
        stStepDelay     : ST_BA_StepDelayParam;
    {endregion}
END_VAR
VAR_INPUT CONSTANT
    {region 'Fixed Parameters'}
        iNextSequenceRef : I_BA_Loop;
    {endregion}
    {region 'Variable Parameters'}
        iSynchronizedLoop : I_BA_Loop;
    {endregion}
END_VAR
VAR
    {region 'Output-Properties'}
        fCtrlDeviation : REAL;
    {endregion}
END_VAR
VAR
    {region 'Hardware'}
        eActionSource : E_BA_ProcessSignalSource;
        eMinMaxOutputSource : E_BA_ProcessSignalSource;
    {endregion}
END_VAR
```


Inputs

Name	Type	Description
bEn	BOOL	Activation of the function block.
fSetpoint	REAL	Setpoint
fCtrlVal	REAL	Feedback of the control value for calculating the control deviation from the setpoint.
eActionPgm	E_BA_Action	Setting the control direction.
fMinOutputPgm	REAL	Lower controller output limit.
fMaxOutputPgm	REAL	Upper controller output limit.
bEnSync	BOOL	Enable synchronization.
fValSync	REAL	Synchronization value. After a positive edge at <i>bEnSync</i> this value is written to <i>fPresentValue</i> .

Outputs

Name	Type	Description
fPresentValue	REAL	Current analog output value of the object.

Inputs CONSTANT PERSISTENT

Name	Type	Description
eOutputUnit	E_BA_Unit	Output unit
fCOVIncrement	REAL	Step size of the Present Value that triggers a COV Notification.
eOpMode	E_BA_PIDMode	Pre- or parallel-set P part.
eActionRm	E_BA_Action	Control direction
fNeutralZone	REAL	Neutral zone
fMinOutputRm	REAL	Minimum output value due to external override.
fMaxOutputRm	REAL	Maximum output value due to external override.
fProportionalConstant	REAL	Proportional constant.
fIntegralConstant	REAL	Integral constant.
fDerivativeConstant	REAL	Derivative constant.
nDampConstant	UDINT	Damping constant.
stStepDelay	<u>ST_BA_StepDelayParam</u> [► 111]	Setting the delay time for step-up or step-down.

VAR

Name	Type	Description
fCtrlDeviation	REAL	Control deviation
eActionSource	<u>E_BA_ProcessSignalSource</u> [► 105]	Definition of whether the control direction is to be treated as a variable or as a parameter.
eMinMaxOutputSource	<u>E_BA_ProcessSignalSource</u> [► 105]	Definition whether the output source is to be treated as a variable or as a parameter.

Requirements

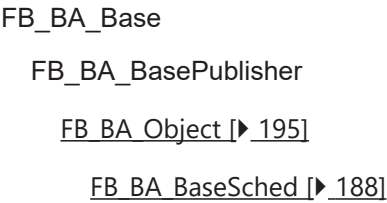
Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.3.4 FB_BA_SchedA



The function block FB_BA_SchedA represents an analog scheduler within the project structure of TF8040.

Inheritance hierarchy



Illustration

```
FUNCTION_BLOCK FB_BA_SchedA EXTENDS FB_BA_BaseSched IMPLEMENTS I_BA_SchedA
VAR_OUTPUT
    fPresentValue      : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Fixed Parameters'}
        fDefaultValue  : REAL;
    {endregion}
    {region 'Variable Parameters'}
        eUnit          : E_BA_Unit:= E_BA_Unit.Invalid;
    {endregion}
END_VAR
```

🔌 Outputs

Name	Type	Description
fPresentValue	REAL	Current analog output value of the object.

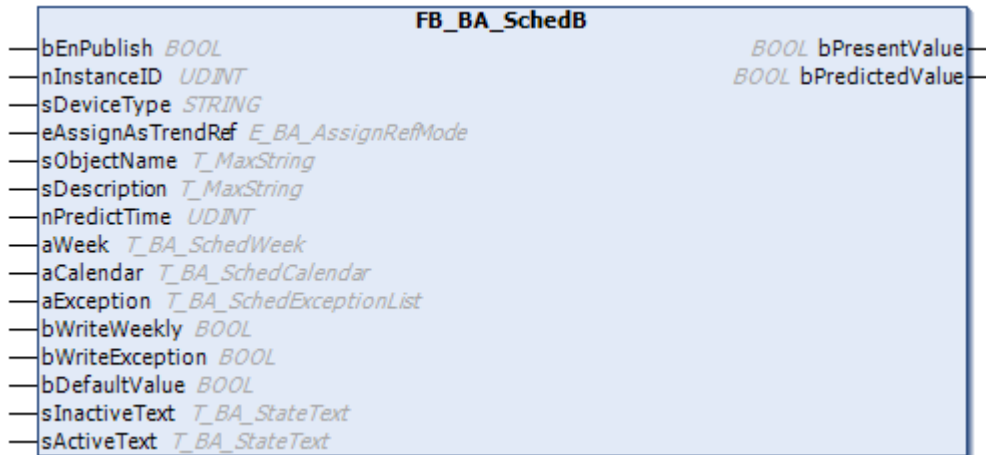
🔌 Inputs CONSTANT PERSISTENT

Name	Type	Description
fDefaultValue	REAL	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
eUnit	E_BA_Unit	Unit of the input or output value of an analog object.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.3.5 FB_BA_SchedB



The FB_BA_SchedB function block represents a binary scheduler within the TF8040 project structure.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB BA Object [► 195]

FB BA BaseSched [► 188]

Illustration

```
FUNCTION_BLOCK FB_BA_SchedB EXTENDS FB_BA_BaseSched IMPLEMENTS I_BA_SchedB
VAR_OUTPUT
    bPresentValue      : BOOL;
    bPredictedValue    : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Fixed Parameters'}
        bDefaultValue   : BOOL;
    {endregion}
    {region 'Variable Parameters'}
        sInactiveText   : T_BA_StateText;
        sActiveText     : T_BA_StateText;
    {endregion}
END_VAR
```

🔌 Outputs

Name	Type	Description
bPresentValue	BOOL	Current binary output value of the object.
bPredictedValue	BOOL	Predicted value.

🔌 Inputs CONSTANT PERSISTENT

Name	Type	Description
bDefaultValue	BOOL	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
sInactiveText	T_BA_StateText [► 112]	Text output when the object is inactive.
sActiveText	T_BA_StateText [► 112]	Text output when the object is active.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.3.6 FB_BA_SchedM



The function block FB_BA_SchedM represents a multi-state scheduler within the project structure of TF8040.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA BaseSched \[► 188\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_SchedM EXTENDS FB_BA_BaseSched IMPLEMENTS I_BA_SchedM
VAR_OUTPUT
    nPresentValue      : UDINT;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Fixed Parameters'}
        aStateText      : T_BA_StateTextArray;
        nDefaultValue    : UDINT := 1;
    {endregion}
END_VAR
VAR
    {region 'Output-Properties'}
        nStateCount      : UDINT;
    {endregion}
END_VAR
```

 Outputs

Name	Type	Description
nPresentValue	UDINT	Current value for multi-stage outputs.

🚩 Inputs CONSTANT PERSISTENT

Name	Type	Description
aStateText	T_BA_StateTextArray [► 112]	The array is used to declare the state texts of a multi-state object.
nDefaultValue	UDINT	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.

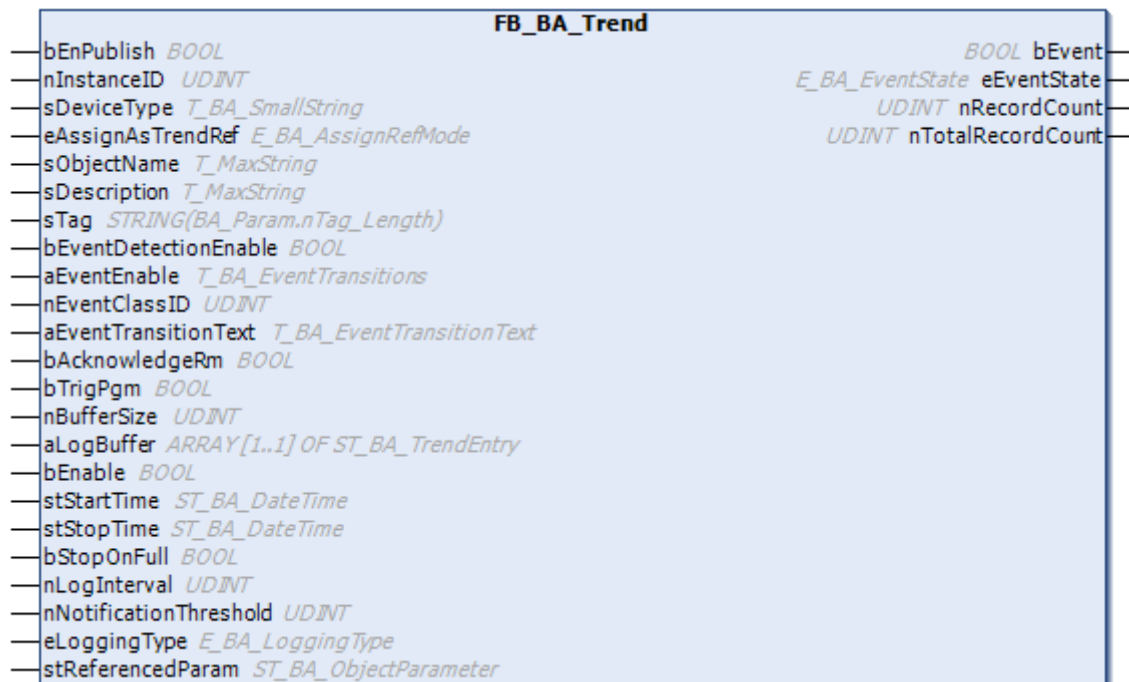
VAR

Name	Type	Description
nStateCount	UDINT	Number of states of a multi-state object.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.3.7 FB_BA_Trend



The function block FB_BA_Trend represents a trend within the project structure of TF8040.

Information about inherited elements

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_EventObject \[► 192\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_Trend EXTENDS FB_BA_EventObject IMPLEMENTS I_BA_Trend
VAR_OUTPUT
    nRecordCount          : UDINT;
```

```

    nTotalRecordCount      : UDINT;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
{region 'Fixed Parameters'}
    nBufferSize           : UDINT := BA_Param.nTrend_BufferSize;
{endregion}
{region 'Variable Parameters'}
    aLogBuffer             : T_BA_TrendLogBuffer;
    bEnable                : BOOL;
    stStartTime            : ST_BA_DateTime := BA_Param.stTrend_DefStartTime;
    stStopTime             : ST_BA_DateTime := BA_Param.stTrend_DefStopTime;
    bStopOnFull            : BOOL := BA_Param.bTrend_DefStopOnFull;
    nLogInterval           : UDINT := BA_Param.nTrend_DefLogInterval;
    nNotificationThreshold : UDINT := BA_Param.nTrend_DefNotificationThreshold;
    eLoggingType           : E_BA_LoggingType := BA_Param.eTrend_DefLoggingType;
    stReferencedParam      : ST_BA_ObjectParameter;
{endregion}
END_VAR

```

VAR_OUTPUT

Name	Type	Description
nRecordCount	UDINT	Number of records.
nTotalRecordCount	UDINT	Absolute number of records.

VAR_INPUT CONSTANT PERSISTENT

Name	Type	Description
nBufferSize	UDINT	Size of the buffer.
aLogBuffer	ST_BA_TrendEntry	Ring buffer for values with timestamp.
bEnable	BOOL	Recording enable.
stStartTime	ST_BA_DateTime	Start time.
stStopTime	ST_BA_DateTime	Stopping time.
bStopOnFull	BOOL	A TRUE stops the recording when the buffer is full.
nLogInterval	UDINT	Interval for saving the parameters.
nNotificationThreshold	UDINT	Limit value at which notifications are triggered.
eLoggingType	E_BA_LoggingType	Setting the type of saving.
stReferencedParam	ST_BA_ObjectParameter ▶ 111	Parameter object.

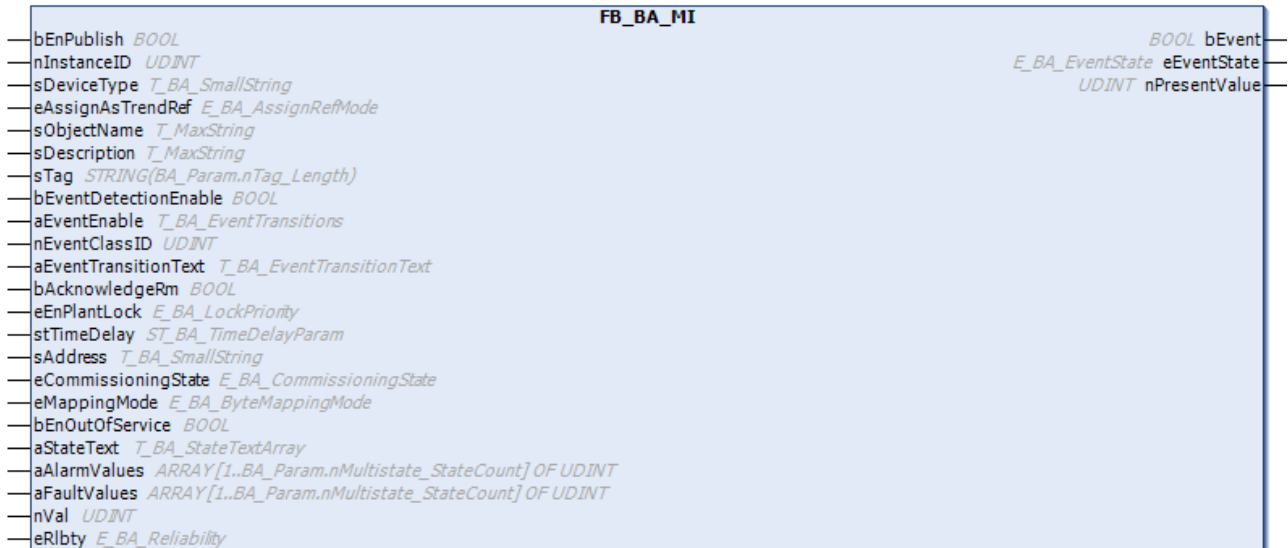
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4 Multistate

Reporting, switching and processing of multi-state values.

6.1.2.1.3.2.2.4.1 FB_BA_MI



The function block FB_BA_MI represents the Multi-State_Input object.

The Multi-State_Input object specifies an object type whose object properties represent externally visible characteristics of a data point for multiple binary inputs. The object is derived from physical or virtual data points, e.g. the limit messages of several analog inputs or analog values or from a mathematical calculation.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseMI [► 185]

Illustration

```
FUNCTION_BLOCK FB_BA_MI EXTENDS FB_BA_BaseMI
VAR_INPUT
    nVal      : UDINT := 1;
    eRlbt     : E_BA_Reliability;
END_VAR
```

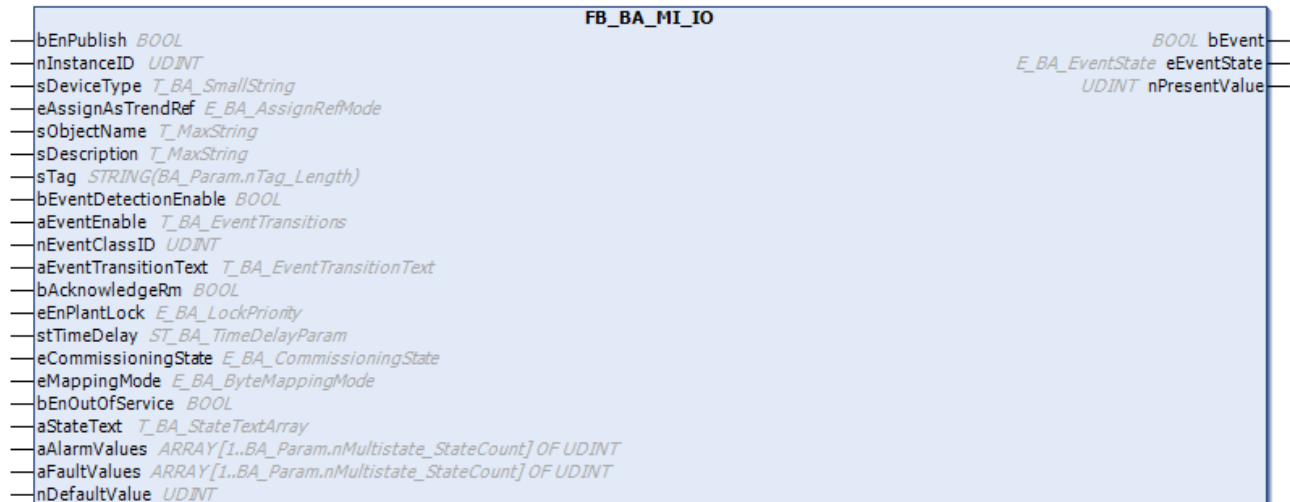
Inputs

Name	Type	Description
nVal	UDINT	Scaled, analog input value.
eRlbt	E_BA_Reliability	Availability or reliability of a value.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4.2 FB_BA_MI_IO



The function block FB_BA_MI_IO represents the Multi-State_Input object.

The Multi-State_Input object specifies an object type whose object properties represent externally visible characteristics of a data point for multiple binary inputs. The object is derived from physical or virtual data points, e.g. the limit messages of several analog inputs or analog values or from a mathematical calculation.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_EventObject \[► 192\]](#)

[FB_BA_EventObjectEx \[► 194\]](#)

[FB_BA_ComEventObject \[► 191\]](#)

[FB_BA_BaseMI \[► 185\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_MI_IO EXTENDS FB_BA_BaseMI IMPLEMENTS I_BA_RawMI
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
  nDefaultValue      : UDINT := 1;
{endregion}
END_VAR
VAR
{region 'Raw I/O'}
  stRawVal   AT %I*      : ST_BA_Byte;
{endregion}
END_VAR
```



Inputs CONSTANT PERSISTENT

Name	Type	Description
nDefaultValue	UDINT	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.

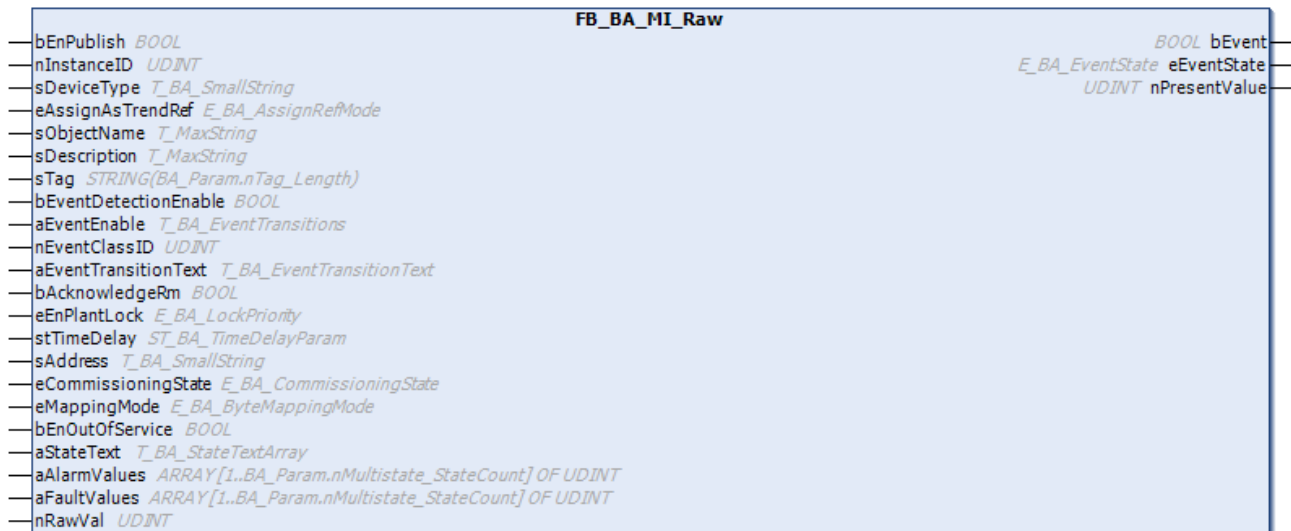
VAR

Name	Type	Description
stRawVal	ST_BA_Byte	Structure variable for linking to the process image.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4.3 FB_BA_MI_Raw



The function block FB_BA_MI_Raw represents the Multi-State_Input object.

The Multi-State_Input object specifies an object type whose object properties represent externally visible characteristics of a data point for multiple binary inputs. The object is derived from physical or virtual data points, e.g. the limit messages of several analog inputs or analog values or from a mathematical calculation.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseMI [► 185]

Illustration

```
FUNCTION_BLOCK FB_BA_MI_Raw EXTENDS FB_BA_BaseMI
VAR_INPUT
    nRawVal      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nRawVal	UDINT	Raw value

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4.4 FB_BA_MO



The function block FB_BA_MO represents the Multi-State_Output object. It has a priority array and can therefore be commanded (see Commanding objects).

The Multi-State_Output object specifies an object type whose object properties represent externally visible characteristics of a data point for multiple binary outputs, connected by arbitrary binary logic. The object is derived from physical or virtual data points, e.g. for "active/inactive" states of several binary outputs or the value of an analog output.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseMO [► 186]

Illustration

```

FUNCTION_BLOCK FB_BA_MO EXTENDS FB_BA_BaseMO
VAR_INPUT
    eRlbtly      : E_BA_Reliability;
    bRawOvrrd    : BOOL;
    bEnFdbck     : BOOL;
    nRawValFdbck : UDINT;
END_VAR

```

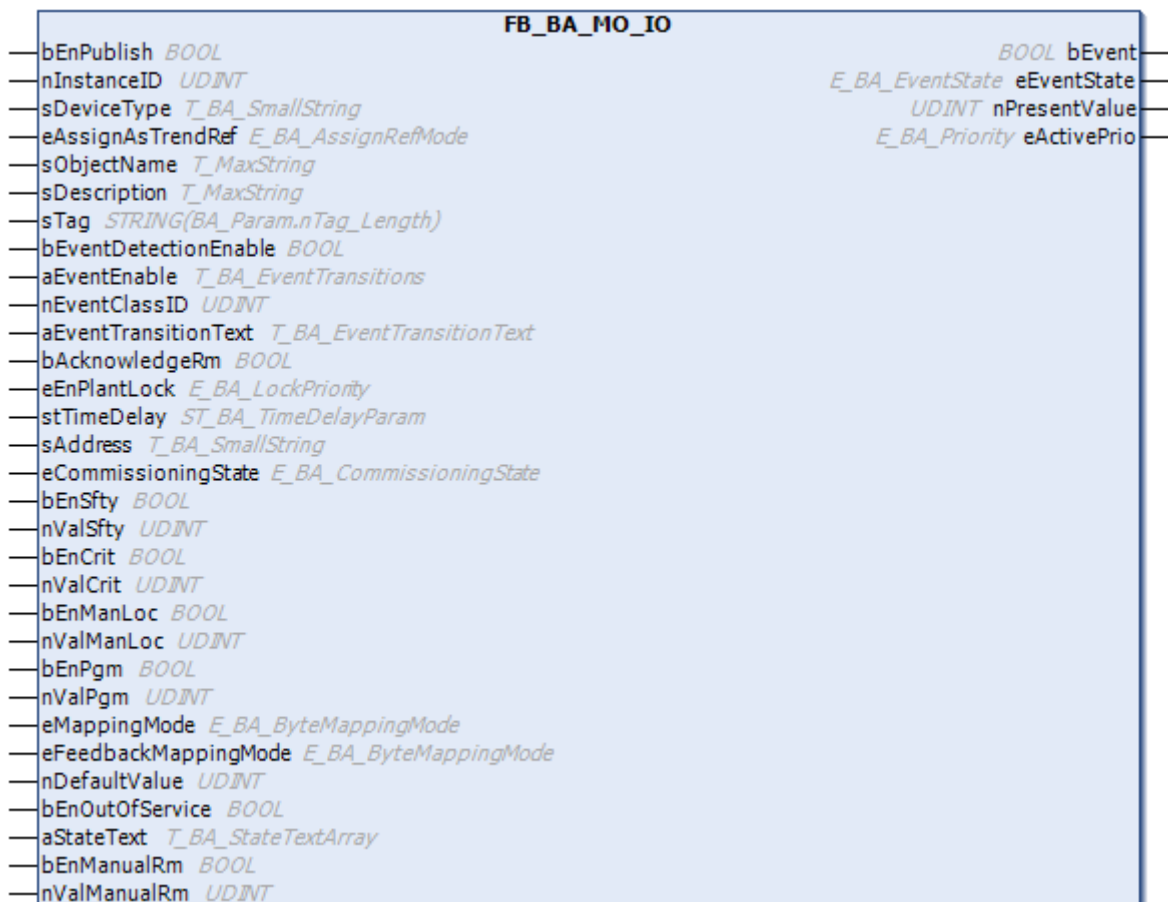
 Inputs

Name	Type	Description
eRlbtly	E_BA_Reliability	Availability or reliability of a value.
bRawOvrrd	BOOL	The feedback of a switch contained in a terminal can be connected to this variable for mechanical override of an output.
bEnFdbck	BOOL	Variable for starting the monitoring of the feedback at the binary output.
nRawValFdbck	UDINT	Raw value feedback.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4.5 FB_BA_MO_IO



The function block FB_BA_MO_IO represents the Multi-State_Output object. It has a priority array and can therefore be commanded (see Commanding objects).

The Multi-State_Output object specifies an object type whose object properties represent externally visible characteristics of a data point for multiple binary outputs, connected by arbitrary binary logic. The object is derived from physical or virtual data points, e.g. for "active/inactive" states of several binary outputs or the value of an analog output.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseMO [► 186]

Illustration

```
FUNCTION_BLOCK FB_BA_MO_IO EXTENDS FB_BA_BaseMO IMPLEMENTS I_BA_RawMO
VAR
  {region 'Raw I/O'}
    stRawVal      AT %Q*      : ST_BA_Byte;
    stRawValFeedback AT %I*      : ST_BA_Byte;
  {endregion}
END_VAR
```

VAR

Name	Type	Description
stRawVal	ST_BA_Byte	Raw value for output to a bus terminal.
stRawValFeedback	ST_BA_Byte	Feedback input of the raw value (optional).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4.6 FB_BA_MO_Raw



The function block FB_BA_MO_Raw represents the Multi-State_Output object. It has a priority array and can therefore be commanded (see Commanding objects).

The Multi-State_Output object specifies an object type whose object properties represent externally visible characteristics of a data point for multiple binary outputs, connected by arbitrary binary logic. The object is derived from physical or virtual data points, e.g. for "active/inactive" states of several binary outputs or the value of an analog output.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

FB_BA_BaseMO [► 186]

Illustration

```
FUNCTION_BLOCK FB_BA_MO_Raw EXTENDS FB_BA_BaseMO
VAR_INPUT
    bEnFeedback          : BOOL;
    nRawValFeedback      : UDINT;
END_VAR
```

```
VAR_OUTPUT
  nRawVal      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
bEnFdbck	BOOL	Variable for starting the monitoring of the feedback at the binary output.
nRawValFdbck	UDINT	Raw value feedback.

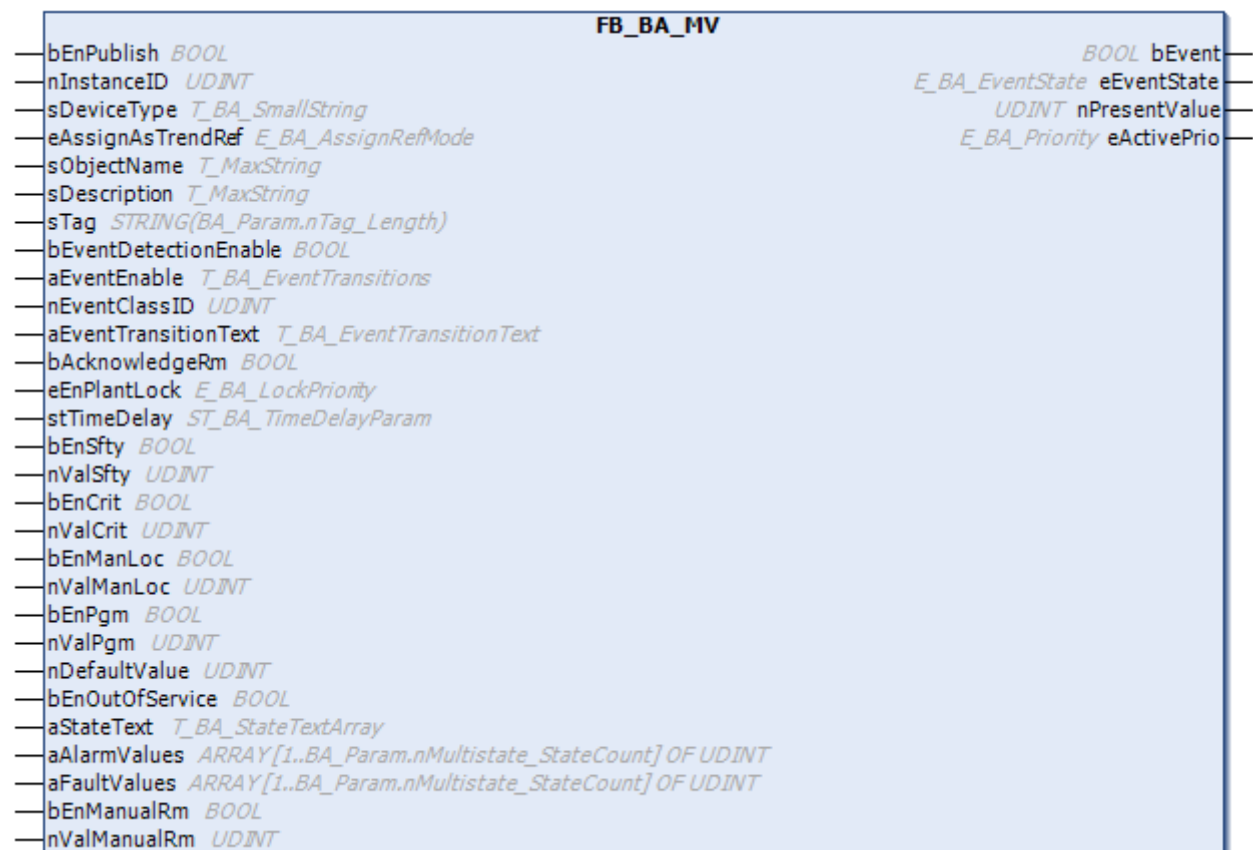
Outputs

Name	Type	Description
nRawVal	UDINT	The variable is used to link the raw value of an object with the process image of the input or output terminal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.4.7 FB_BA_MV



The function block FB_BA_MV represents the Multi-State_Value object. It has a priority array and can therefore be commanded (see Commanding objects).

The Multi-State_Value object specifies an object type whose properties represent externally visible characteristics of a virtual data point for a "Multi-State Value". The Multi-State_Value has no I/O hardware in the associated Device object, it is stored in memory.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

Illustration

```
FUNCTION_BLOCK FB_BA_MV EXTENDS FB_BA_EventObjectEx IMPLEMENTS I_BA_MultistatePrioObject, I_BA_AnyVa
lue
VAR_INPUT
    bEnSfty          : BOOL;
    nValSfty         : UDINT := 1;
    bEnCrit          : BOOL;
    nValCrit         : UDINT := 1;
    bEnManLoc        : BOOL;
    nValManLoc       : UDINT := 1;
    bEnPgm           : BOOL;
    nValPgm          : UDINT := 1;
END_VAR
VAR_OUTPUT
    nPresentValue    : UDINT := 1;
    eActivePrio      : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        nDefaultValue : UDINT:= 1;
        bEnOutOfService : BOOL;
    {endregion}
    {region 'Fixed Parameters'}
        aStateText      : T_BA_StateTextArray;
        aAlarmValues    : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
        aFaultValues    : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
    {endregion}
    {region 'Operational Parameters'}
        bEnManualRm     : BOOL;
        nValManualRm    : UDINT:= 1;
    {endregion}
END_VAR
VAR
    {region 'Output-Properties'}
        nStateCount     : UDINT;
    {endregion}
END_VAR
```

Inputs

Name	Type	Description
bEnSfty	BOOL	Enabling the "Safety" priority.
nValSfty	UDINT	Analog value for the "Safety" priority.
bEnCrit	BOOL	Enabling the "Critical" priority.
nValCrit	UDINT	Analog value for the "Critical" priority.
bEnManLoc	BOOL	Enabling the "Manual Local" priority.
nValManLoc	UDINT	Analog value for the "Manual Local" priority.
bEnPgm	BOOL	Enabling the "Program" priority.
nValPgm	UDINT	Analog value for the "Program" priority.

🔌 Outputs

Name	Type	Description
nPresentValue	UDINT	Current value for multi-stage outputs.
eActivePrio	E_BA_Priority	Active priority

🔌 Inputs CONSTANT PERSISTENT

Name	Type	Description
nDefaultValue	UDINT	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
aStateText	T_BA_StateTextArray [► 112]	The array is used to declare the state texts of a multi-state object.
aAlarmValues	ARRAY [1..BA_Param.nMultistate _StateCount] OF UDINT	Within the array the states of the multi-state object are described for which an alarm is present.
aFaultValues	ARRAY [1..BA_Param.nMultistate _StateCount] OF UDINT	Within the array the states of the multi-state object are described where an error is present.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.
nValManualRm	UDINT	Variable for writing a value to the "Manual Remote" priority.

VAR

Name	Type	Description
nStateCount	UDINT	Number of states of a multi-state object.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.4.8 FB_BA_MV_Op

FB_BA_MV_Op	
bEnPublish	BOOL
nInstanceID	UDINT
sDeviceType	T_BA_SmallString
eAssignAsTrendRef	E_BA_AssignRefMode
sObjectName	T_MaxString
sDescription	T_MaxString
sTag	STRING(BA_Param.nTag_Length)
nValuePgm	UDINT
nDefaultValue	UDINT
eToggleMode	E_BA_ToggleMode
aStateText	T_BA_StateTextArray
nValueRm	UDINT

The function block FB_BA_MV_Op represents a multi-state value object. It is used to display or input a multi-state value.

If something is connected to the input variable *nValuePgm*, then the object automatically recognizes that it is used to display a value. Otherwise it is used to enter a multi-state value.

The Multi-State_Value object specifies an object type whose properties represent externally visible characteristics of a virtual data point for a "Multi-State Value". The Multi-State_Value has no I/O hardware in the associated Device object, it is stored in memory.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_MV_Op EXTENDS FB_BA_Object IMPLEMENTS I_BA_MultistateOpObject, I_BA_AnyValue
VAR_INPUT
    nValuePgm          : UDINT;
END_VAR
VAR_OUTPUT
    nPresentValue      : UDINT := 1;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        nDefaultValue   : UDINT := 1;
        eToggleMode     : E_BA_ToggleMode := E_BA_ToggleMode.eSwitch;
    {endregion}
    {region 'Fixed Parameters'}
        aStateText      : T_BA_StateTextArray;
    {endregion}
    {region 'Operational Parameters'}
        nValueRm        : UDINT := 1;
    {endregion}
END_VAR
VAR
    {region 'Output-Properties'}
        nStateCount     : UDINT;
    {endregion}
END_VAR
VAR
    {region 'Interface'}
        eValueSource    : E_BA_ProcessSignalSource;
    {endregion}
END_VAR
```

Inputs

Name	Type	Description
nValuePgm	UDINT	Value of a multi-state object for the "Program" priority.

Outputs

Name	Type	Description
nPresentValue	UDINT	Current value for multi-stage outputs.

🚩 Inputs CONSTANT PERSISTENT

Name	Type	Description
nDefaultValue	UDINT	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
eToggleMode	E_BA_ToggleMode	Enumeration for defining how the output value <i>bPresentValue</i> of an object is generated depending on the input <i>bValuePgm</i> .
aStateText	T_BA_StateTextArray [► 112]	The array is used to declare the state texts of a multi-state object.
nValueRm	UDINT	Variable for writing an object from the HMI.

VAR

Name	Type	Description
nStateCount	UDINT	Number of states of a multi-state object.
eValueSource	E_BA_ProcessSignalSource [► 105]	The variable indicates whether an object of the type FB_BA_..._OP serves as a display or input object. <i>eVarInput</i> = 1 The object is used to display a value. The value is passed to the object at an input within the PLC. <i>eParameter</i> = 2 The object is used to enter a parameter that can be changed by a BACnet client or the TwinCAT HMI.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.5 Structurize

6.1.2.1.3.2.2.5.1 FB_BA_View

FB_BA_View	
—	bEnPublish <i>BOOL</i>
—	nInstanceID <i>UDINT</i>
—	sDeviceType <i>T_BA_SmallString</i>
—	eAssignAsTrendRef <i>E_BA_AssignRefMode</i>
—	sObjectName <i>T_MaxString</i>
—	sDescription <i>T_MaxString</i>
—	sTag <i>STRING(XBA_Param.nTag_Length)</i>
—	eNodeType <i>E_BA_NodeType</i>
—	bAcknowledgeRm <i>BOOL</i>

The function block FB_BA_View is used to create a folder within the project structure.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```
FUNCTION_BLOCK FB_BA_View EXTENDS FB_BA_Object IMPLEMENTS I_BA_View
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
```

```

    eNodeType          : E_BA_NodeType := E_BA_NodeTypeAutomatic;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Operational Parameters'}
    bAcknowledgeRm      : BOOL;
  {endregion}
END_VAR
VAR
  {region 'SubInit'}
    eDPADMode           : E_BA_DPADMode := E_BA_DPADModeUndefined;
  {endregion}
  {region 'Events'}
    fbActiveEvents       : FB_BA_EventIndicator;
  {endregion}
END_VAR

```

Inputs CONSTANT PERSISTENT

Name	Type	Description
eNodeType	E_BA_NodeType [► 97]	The variable <i>NodeType</i> describes the folder level within the project structure.

Inputs CONSTANT

Name	Type	Description
bAcknowledgeRm	BOOL	Input for local acknowledgement of the events of an object.

VAR

Name	Type	Description
eDPADMode	E_BA_DPADMode [► 97]	The variable influences the project structure integrated generation of the user address key. To create the project structure it is mandatory to establish an integrated parent / child relationship within the TwinCAT project. This means that all instances of the function block <i>FB_BA_View</i> must be told from whom they originate. However, the number of levels within the TwinCAT program or symbol path often differs from the number of levels in the user address key. Often there are more levels in the symbol path of the TwinCAT project than in the user address key. It may therefore be necessary to exclude some levels from the concatenation of texts for the generic generation of object names and description texts. This is possible with the enumeration <i>eDPADMode</i>.
fbActiveEvents	FB_BA_EventIndicator	Gives information about events of all objects of a View object.

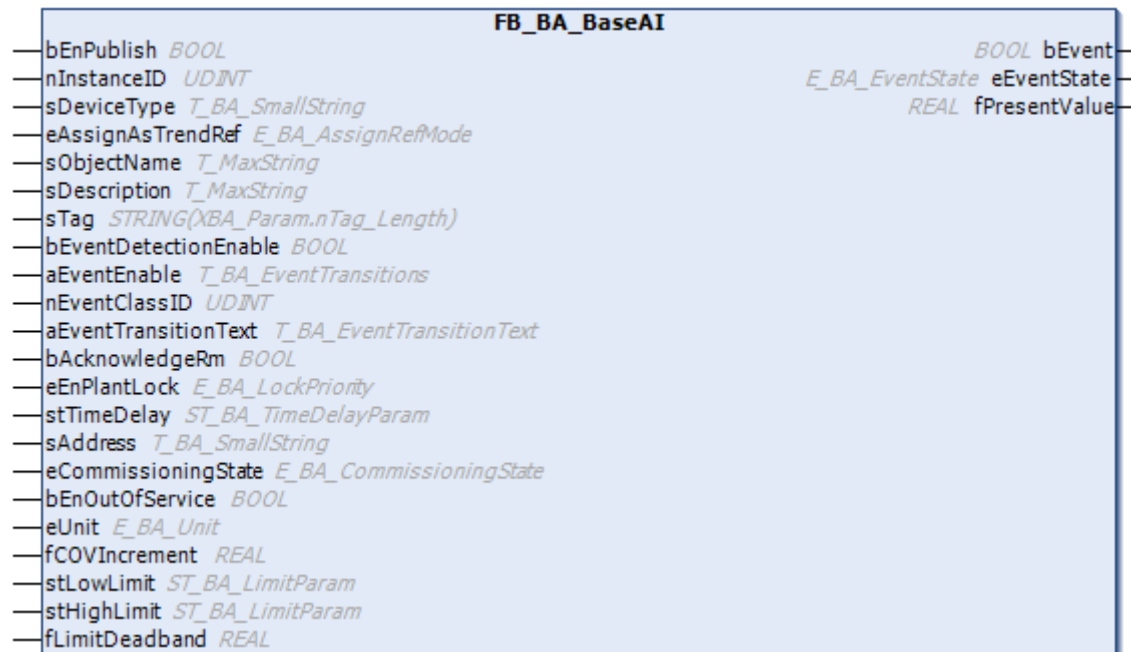
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6 Basis

These objects form the basis of the previously described function blocks. They inherit the local function blocks.

6.1.2.1.3.2.2.6.1 FB_BA_BaseAI



The function block FB_BA_BaseAI is the base of all analog input objects.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA EventObject \[► 192\]](#)

[FB BA EventObjectEx \[► 194\]](#)

[FB BA ComEventObject \[► 191\]](#)

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseAI EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_AnalogInObject, I_
BA_AnyValue
VAR_OUTPUT
    fPresentValue      : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        fResolution      : REAL := BA_Param.fInput_DefResolution;
        fScaleOffset     : REAL := BA_Param.fInput_DefScaleOffset;
        bEnOutOfService  : BOOL;
        eUnit            : E_BA_Unit := E_BA_Unit.Invalid;
        fCOVIncrement    : REAL := BA_Param.fDefCOVIncrement;
        stLowLimit       : ST_BA_LimitParam;
        stHighLimit      : ST_BA_LimitParam;
        fLimitDeadband   : REAL := BA_Param.fDefLimitDeadband;
    {endregion}
END_VAR
```

🔌 Outputs

Name	Type	Description
fPresentValue	REAL	Current analog output value of the object.

Inputs CONSTANT PERSISTENT

Name	Type	Description
fResolution [▶ 116]	REAL	Resolution of an analog signal for scaling a measured value.
fScaleOffset [▶ 116]	REAL	Scaling offset
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
eUnit	E_BA_Unit	Unit of the input or output value of an analog object.
fCOVIncrement	REAL	The variable specifies the minimum change in present value that will cause a COV notification to be issued to subscribed COV clients. This property is required if COV reporting is supported by this object.
stLowLimit	ST_BA_LimitParam [▶ 111]	Parameterization of the lower limit value monitoring of an analog object. The variable <i>bEnable</i> must be TRUE to enable limit value monitoring. The variable <i>fValue</i> is used to parameterize the lower limit value.
stHighLimit	ST_BA_LimitParam [▶ 111]	Parameterization of the upper limit value monitoring of an analog object. The variable <i>bEnable</i> must be TRUE to enable limit value monitoring. The variable <i>fValue</i> is used to parameterize the upper limit value.
fLimitDeadband	REAL	Dead band or hysteresis for the values <i>HighLimit</i> and <i>LowLimit</i> of the limit value monitoring of an analog object.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.2 FB_BA_BaseAO



The function block FB_BA_BaseAO represents the object of an analog output. It is the base of all analog outputs.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseAO EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_AnalogOutObject, I
  _BA_AnyValue
VAR_INPUT
  bEnSfty          : BOOL;
  fValSfty         : REAL;
  bEnCrit          : BOOL;
  fValCrit         : REAL;
  bEnManLoc        : BOOL;
  fValManLoc       : REAL;
  bEnPgm           : BOOL;
  fValPgm          : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue    : REAL;
```

```

    eActivePrio      : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    fResolution      : REAL := BA_Param.fOutput_DefResolution;
    fScaleOffset     : REAL := BA_Param.fOutput_DefScaleOffset;
    fDefaultValue    : REAL;
    bEnOutOfService  : BOOL;
    eUnit            : E_BA_Unit := E_BA_Unit.Invalid;
    fCOVIncrement    : REAL := BA_Param.fDefCOVIncrement;
    stLowLimit       : ST_BA_LimitParam;
    stHighLimit      : ST_BA_LimitParam;
    fLimitDeadband   : REAL := BA_Param.fDefLimitDeadband;
{endregion}
{region 'Operational Parameters'}
    bEnManualRm      : BOOL;
    fValManualRm     : REAL;
{endregion}
END_VAR

```

Inputs

Name	Type	Description
bEnSfty	BOOL	Enabling the "Safety" priority.
fValSfty	REAL	Analog value for the "Safety" priority.
bEnCrit	BOOL	Enabling the "Critical" priority.
fValCrit	REAL	Analog value for the "Critical" priority.
bEnManLoc	BOOL	Enabling the "Manual Local" priority.
fValManLoc	REAL	Analog value for the "Manual Local" priority.
bEnPgm	BOOL	Enabling the "Program" priority.
fValPgm	REAL	Analog value for the "Program" priority.

Outputs

Name	Type	Description
fPresentValue	REAL	Current analog output value of the object.
eActivePrio	E_BA_Priority	Active priority

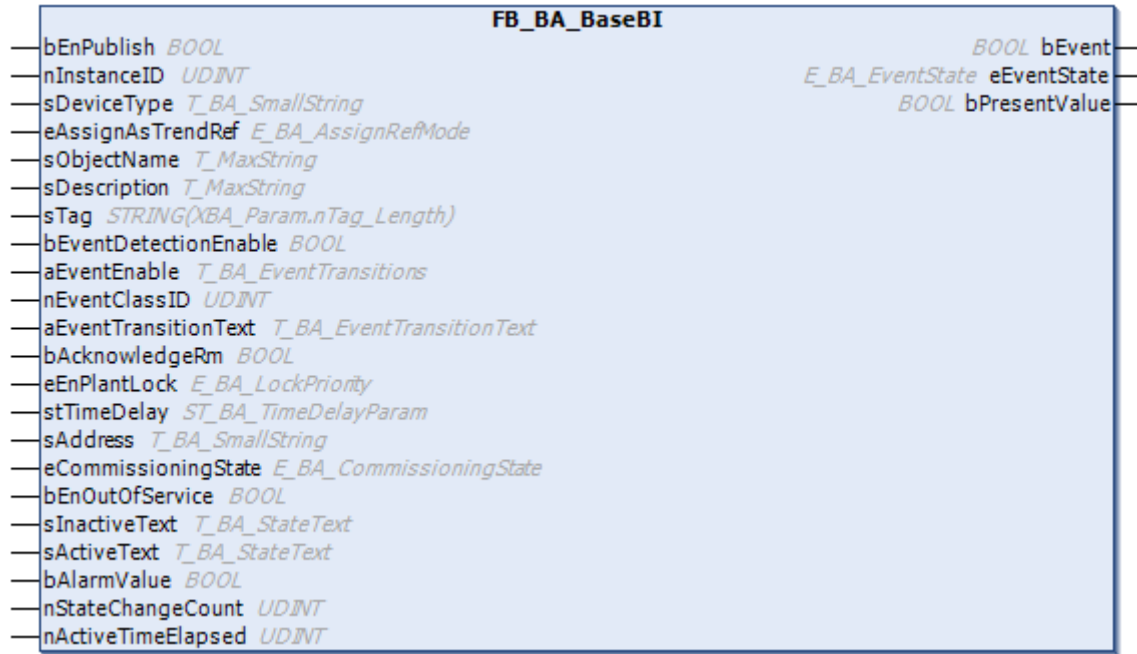
Inputs CONSTANT PERSISTENT

Name	Type	Description
fResolution [▶ 116]	REAL	Resolution of an analog signal for scaling a measured value.
fScaleOffset [▶ 116]	REAL	Scaling offset
fDefaultValue	REAL	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
eUnit	E_BA_Unit	Unit of the input or output value of an analog object.
fCOVIncrement	REAL	The variable specifies the minimum change in present value that will cause a COV notification to be issued to subscribed COV clients. This property is required if COV reporting is supported by this object.
stLowLimit	ST_BA_LimitParam ▶ 111	Parameterization of the lower limit value monitoring of an analog object. The variable <i>bEnable</i> must be TRUE to enable limit value monitoring. The variable <i>fValue</i> is used to parameterize the lower limit value.
stHighLimit	ST_BA_LimitParam ▶ 111	Parameterization of the upper limit value monitoring of an analog object. The variable <i>bEnable</i> must be TRUE to enable limit value monitoring. The variable <i>fValue</i> is used to parameterize the upper limit value.
fLimitDeadband	REAL	Dead band or hysteresis for the values <i>HighLimit</i> and <i>LowLimit</i> of the limit value monitoring of an analog object.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.
fValManualRm	REAL	Analog value for the "Manual Remote" priority.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.3 FB_BA_BaseBI



The function block FB_BA_BaseBI generates a binary input object. It is the base of all binary input objects.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseBI EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_BinaryInObject, I_
BA_AnyValue
VAR_OUTPUT
    bPresentValue          : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        bEnOutOfService    : BOOL;
        sInactiveText      : T_BA_StateText;
        sActiveText        : T_BA_StateText;
        bAlarmValue        : BOOL := TRUE;
        ePolarity           : E_BA_Polarity := E_BA_Polarity.eNormal;
        nStateChangeCount  : UDINT;
        nActiveTimeElapsed : UDINT;
    {endregion}
END_VAR
VAR
    {region 'Output-Properties'}
        stStateChangeTime  : ST_BA_DateTime;
        stStateChangeResetPoint : ST_BA_DateTime;
        stActiveTimeResetPoint : ST_BA_DateTime;
    {endregion}
END_VAR
```

🔌 Outputs

Name	Type	Description
bPresentValue	BOOL	Current binary output value of the object.

🔌 Inputs CONSTANT PERSISTENT

Name	Type	Description
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
sInactiveText	T_BA_StateText [► 112]	Text output when the object is inactive.
sActiveText	T_BA_StateText [► 112]	Text output when the object is active.
bAlarmValue	BOOL	Value in the event of an alarm.
ePolarity	E_BA_Polarity	The polarity describes the dependency between the value resulting from the evaluation of the Priority_Array and the value that is output at the output of the controller. If the polarity is normal then the result of the Priority_Array is directly forwarded to the output of the controller. With reverse polarity the output is negated.
nStateChangeCount	UDINT	The variable indicates how often the state of the <i>Present_Value</i> has changed since the date and the last reset.
nActiveTimeElapsed	UDINT	Time [s] in which the <i>Present_Value</i> of the object had the value ACTIVE. The time is valid from the last reset by the property <i>Time_Of_Active_Time_Reset</i> .

VAR

Name	Type	Description
stStateChangeTime	ST_BA_DateTime	This property shows the time of the last state change. The state change refers to the <i>Present_Value</i> of the object.
stStateChangeResetPoint	ST_BA_DateTime	Shows the date and time from which the counting of state changes started.
stActiveTimeResetPoint	ST_BA_DateTime	Indicates the time when the recording of the object's switch-on times started.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.4 FB_BA_BaseBO



The function block FB_BA_BaseBO represents a binary output object. It is the base for all other binary outputs.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_EventObject \[► 192\]](#)

[FB_BA_EventObjectEx \[► 194\]](#)

[FB_BA_ComEventObject \[► 191\]](#)

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseBO EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_BinaryOutObject, I
_BA_AnyValue
VAR_INPUT
    bEnSfty          : BOOL;
    bValSfty         : BOOL;
    bEnCrit          : BOOL;
    bValCrit         : BOOL;
    bEnManLoc        : BOOL;
    bValManLoc       : BOOL;
    bEnPgm           : BOOL;
    bValPgm          : BOOL;
END_VAR
```

```

VAR_OUTPUT
  bPresentValue          : BOOL;
  eActivePrio            : E_BA_Priority;
END_VAR

VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    nMinimumOffTime      : UDINT;
    nMinimumOnTime       : UDINT;
    bDefaultValue        : BOOL;
    bEnOutOfService      : BOOL;
    sInactiveText        : T_BA_StateText;
    sActiveText          : T_BA_StateText;
    ePolarity            : E_BA_Polarity := E_BA_Polarity.eNormal;
    nStateChangeCount    : UDINT;
    nActiveTimeElapsed   : UDINT;
  {endregion}
  {region 'Operational Parameters'}
    bEnManualRm          : BOOL;
    bValManualRm         : BOOL;
  {endregion}
END_VAR

VAR
  {region 'Output-Properties'}
    stStateChangeTime    : ST_BA_DateTime;
    stStateChangeResetPoint : ST_BA_DateTime;
    stActiveTimeResetPoint : ST_BA_DateTime;
  {endregion}
END_VAR

```

Inputs

Name	Type	Description
bEnSfty	BOOL	Enabling the "Safety" priority.
bValSfty	BOOL	Binary value for the "Safety" priority.
bEnCrit	BOOL	Enabling the "Critical" priority.
bValCrit	BOOL	Binary value for the "Critical" priority.
bEnManLoc	BOOL	Enabling the "Manual Local" priority.
bValManLoc	BOOL	Binary value for the "Manual Local" priority.
bEnPgm	BOOL	Enabling the "Program" priority.
bValPgm	BOOL	Binary value for the "Program" priority.

Outputs

Name	Type	Description
bPresentValue	BOOL	Current binary output value of the object.
eActivePrio	E_BA_Priority	Active priority

Inputs CONSTANT PERSISTENT

Name	Type	Description
nMinimumOffTime	UDINT	Minimum time [s] in which the Present_Value should remain in the INACTIVE state after a write operation to Present_Value has assumed the INACTIVE state. This can be used to implement protection against too fast restarting.
nMinimumOnTime	UDINT	Minimum time [s] in which the Present_Value is to remain in the ACTIVE state after a write operation to Present_Value has assumed the ACTIVE state. This can be used to implement protection against premature, renewed switch-off
bDefaultValue	BOOL	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
sInactiveText	T_BA_StateText ► 112	Text output when the object is inactive.
sActiveText	T_BA_StateText ► 112	Text output when the object is active.
ePolarity	E_BA_Polarity	The polarity describes the dependency between the value resulting from the evaluation of the Priority_Array and the value that is output at the output of the controller. If the polarity is normal then the result of the Priority_Array is directly forwarded to the output of the controller. With reverse polarity the output is negated.
nStateChangeCount	UDINT	The variable indicates how often the state of the <i>Present_Value</i> has changed since the date and the last reset.
nActiveTimeElapsed	UDINT	Time [s] in which the <i>Present_Value</i> of the object had the value ACTIVE. The time is valid from the last reset by the property <i>Time_Of_Active_Time_Reset</i> .
bValManualRm	BOOL	Binary value for the "Manual Remote" priority.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.

VAR

Name	Type	Description
stStateChangeTime	ST_BA_DateTime	This property shows the time of the last state change. The state change refers to the <i>Present_Value</i> of the object.
stStateChangeResetPoint	ST_BA_DateTime	Shows the date and time from which the counting of state changes started.
stActiveTimeResetPoint	ST_BA_DateTime	Indicates the time when the recording of the object's switch-on times started.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.5 FB_BA_BaseMI

FB_BA_BaseMI	
bEnPublish <i>BOOL</i>	<i>BOOL</i> bEvent
nInstanceID <i>UDINT</i>	<i>E_BA_EventState</i> eEventState
sDeviceType <i>T_BA_SmallString</i>	<i>UDINT</i> nPresentValue
eAssignAsTrendRef <i>E_BA_AssignRefMode</i>	
sObjectName <i>T_MaxString</i>	
sDescription <i>T_MaxString</i>	
sTag <i>STRING(XBA_Param.nTag_Length)</i>	
bEventDetectionEnable <i>BOOL</i>	
aEventEnable <i>T_BA_EventTransitions</i>	
nEventClassID <i>UDINT</i>	
aEventTransitionText <i>T_BA_EventTransitionText</i>	
bAcknowledgeRm <i>BOOL</i>	
eEnPlantLock <i>E_BA_LockPriority</i>	
stTimeDelay <i>ST_BA_TimeDelayParam</i>	
sAddress <i>T_BA_SmallString</i>	
eCommissioningState <i>E_BA_CommissioningState</i>	
eMappingMode <i>E_BA_ByteMappingMode</i>	
bEnOutOfService <i>BOOL</i>	
aStateText <i>T_BA_StateTextArray</i>	
aAlarmValues <i>ARRAY[1..XBA_Param.nMultistate_StateCount] OF UDINT</i>	
aFaultValues <i>ARRAY[1..XBA_Param.nMultistate_StateCount] OF UDINT</i>	

The FB_BA_BaseMI function block generates a multi-state input object. It is the base of all multi-state input objects.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA EventObject \[► 192\]](#)

[FB BA EventObjectEx \[► 194\]](#)

[FB BA ComEventObject \[► 191\]](#)

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseMI EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_MultistateInObject
, I_BA_AnyValue
VAR_OUTPUT
    nPresentValue      : UDINT := 1;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        eMappingMode      : E_BA_ByteMappingMode := E_BA_ByteMappingMode.eIndex1N;
        bEnOutOfService    : BOOL;
    {endregion}
    {region 'Fixed Parameters'}
        aStateText         : T_BA_StateTextArray;
        aAlarmValues       : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
        aFaultValues       : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
    {endregion}
END_VAR
VAR
    {region 'Output-Properties'}
        nStateCount        : UDINT;
    {endregion}
END_VAR
```

Outputs

Name	Type	Description
nPresentValue	UDINT	Analog output value.

🚩 Inputs CONSTANT PERSISTENT

Name	Type	Description
eMappingMode	E_BA_ByteMappingMode	Mode for configuring the terminal link.
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
aStateText	T_BA_StateTextArray [► 112]	The array is used to declare the state texts of a multi-state object.
aAlarmValues	ARRAY [1..BA_Param.nMultistate _StateCount] OF UDINT	Within the array the states of the multi-state object are described for which an alarm is present.
aFaultValues	ARRAY [1..BA_Param.nMultistate _StateCount] OF UDINT	Within the array the states of the multi-state object are described where an error is present.

VAR

Name	Type	Description
nStateCount	UDINT	Number of states of a multi-state object.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.6 FB_BA_BaseMO

FB_BA_BaseMO	
bEnPublish <i>BOOL</i>	<i>BOOL</i> bEvent
nInstanceID <i>UDINT</i>	<i>E_BA_EventState</i> eEventState
sDeviceType <i>T_BA_SmallString</i>	<i>UDINT</i> nPresentValue
eAssignAsTrendRef <i>E_BA_AssignRefMode</i>	<i>E_BA_Priority</i> eActivePrio
sObjectName <i>T_MaxString</i>	
sDescription <i>T_MaxString</i>	
sTag <i>STRING(XBA_Param.nTag_Length)</i>	
bEventDetectionEnable <i>BOOL</i>	
aEventEnable <i>T_BA_EventTransitions</i>	
nEventClassID <i>UDINT</i>	
aEventTransitionText <i>T_BA_EventTransitionText</i>	
bAcknowledgeRm <i>BOOL</i>	
eEnPlantLock <i>E_BA_LockPriority</i>	
stTimeDelay <i>ST_BA_TimeDelayParam</i>	
sAddress <i>T_BA_SmallString</i>	
eCommissioningState <i>E_BA_CommissioningState</i>	
bEnSfty <i>BOOL</i>	
nValSfty <i>UDINT</i>	
bEnCrit <i>BOOL</i>	
nValCrit <i>UDINT</i>	
bEnManLoc <i>BOOL</i>	
nValManLoc <i>UDINT</i>	
bEnPgm <i>BOOL</i>	
nValPgm <i>UDINT</i>	
eMappingMode <i>E_BA_ByteMappingMode</i>	
eFeedbackMappingMode <i>E_BA_ByteMappingMode</i>	
nDefaultValue <i>UDINT</i>	
bEnOutOfService <i>BOOL</i>	
aStateText <i>T_BA_StateTextArray</i>	
bEnManualRm <i>BOOL</i>	
nValManualRm <i>UDINT</i>	

The function block FB_BA_BaseMO represents a multi-level output.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

FB_BA_EventObjectEx [► 194]

FB_BA_ComEventObject [► 191]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseMO EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_MultistateOutObject, I_BA_AnyValue
VAR_INPUT
    bEnSfty          : BOOL;
    nValSfty         : UDINT := 1;
    bEnCrit          : BOOL;
    nValCrit         : UDINT := 1;
    bEnManLoc        : BOOL;
    nValManLoc       : UDINT := 1;
    bEnPgm           : BOOL;
    nValPgm          : UDINT := 1;
END_VAR
VAR_OUTPUT
    nPresentValue    : UDINT := 1;
    eActivePrio      : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    {region 'Variable Parameters'}
        eMappingMode      : E_BA_ByteMappingMode := E_BA_ByteMappingMode.eIndex1N;
        eFeedbackMappingMode : E_BA_ByteMappingMode := E_BA_ByteMappingMode.eIndex1N;
        nDefaultValue     : UDINT := 1;
        bEnOutOfService   : BOOL;
    {endregion}
    {region 'Fixed Parameters'}
        aStateText        : T_BA_StateTextArray;
    {endregion}
    {region 'Operational Parameters'}
        bEnManualRm       : BOOL;
        nValManualRm      : UDINT := 1;
    {endregion}
END_VAR
VAR
    {region 'Output-Properties'}
        nStateCount       : UDINT;
    {endregion}
END_VAR
```

🔧 Inputs

Name	Type	Description
bEnSfty	BOOL	Enabling the "Safety" priority.
nValSfty	UDINT	Analog value for the "Safety" priority.
nValCrit	UDINT	Analog value for the "Critical" priority.
bEnCrit	BOOL	Enabling the "Critical" priority.
bEnManLoc	BOOL	Enabling the "Manual Local" priority.
nValManLoc	UDINT	Analog value for the "Manual Local" priority.
bEnPgm	BOOL	Enabling the "Program" priority.
nValPgm	UDINT	Analog value for the "Program" priority.

🔌 Outputs

Name	Type	Description
nPresentValue	UDINT	Analog output value.
eActivePrio	E_BA_Priority	Active priority

🔌 Inputs **CONSTANT PERSISTENT**

Name	Type	Description
eMappingMode	E_BA_ByteMappingMode	Mode for configuring the terminal link.
eFeedbackMappingMode	E_BA_ByteMappingMode	Structure for mapping the feedback inputs.
nDefaultValue	UDINT	Value that is assumed if all 16 priorities of the Priority_Array of a commandable object have no entry or are NULL.
bEnOutOfService	BOOL	This variable sets an object out of service. It is thus OutOfService.
aStateText	T_BA_StateTextArray [► 112]	The array is used to declare the state texts of a multi-state object.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.
nValManualRm	UDINT	Variable for writing a value to the "Manual Remote" priority.

VAR

Name	Type	Description
nStateCount	UDINT	Number of states of a multi-state object.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.7 FB_BA_BaseSched

FB_BA_BaseSched	
—	bEnPublish <i>BOOL</i>
—	nInstanceID <i>UDINT</i>
—	sDeviceType <i>T_BA_SmallString</i>
—	eAssignAsTrendRef <i>E_BA_AssignRefMode</i>
—	sObjectName <i>T_MaxString</i>
—	sDescription <i>T_MaxString</i>
—	sTag <i>STRING(XBA_Param.nTag_Length)</i>
—	nPredictTime <i>UDINT</i>
—	aWeek <i>T_BA_SchedWeek</i>
—	aCalendar <i>T_BA_SchedCalendar</i>
—	aException <i>T_BA_SchedExceptionList</i>
—	bWriteWeekly <i>BOOL</i>
—	bWriteCalendar <i>BOOL</i>
—	bWriteException <i>BOOL</i>

The function block FB_BA_BaseSched forms the base for scheduler function blocks.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseSched EXTENDS FB_BA_Object IMPLEMENTS I_BA_BaseSched
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    nPredictTime      : UDINT;
    aWeek             : T_BA_SchedWeek;
    aCalendar         : T_BA_SchedCalendar;
    aException        : T_BA_SchedExceptionList;
{endregion}
END_VAR
VAR_INPUT CONSTANT
{region 'Variable Parameters'}
    bWriteWeekly      : BOOL;
    bWriteException   : BOOL;
{endregion}
END_VAR
```

🔧 Inputs CONSTANT PERSISTENT

Name	Type	Description
nPredictTime	UDINT	Calculated time value.
aWeek	T_BA_SchedWeek [► 113]	Weekly scheduler.
aCalendar	T_BA_SchedCalendar [► 113]	Calendar.
aException	T_BA_SchedExceptionList [► 113]	List of exception conditions.

🔧 Inputs CONSTANT

Name	Type	Description
bWriteWeekly	BOOL	Writes weekly.
bWriteException	BOOL	Writes time exceptions.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.8 FB_BA_BaseStateMV

FB_BA_BaseStateMV
— bEnPublish <i>BOOL</i>
— nInstanceID <i>UDINT</i>
— sDeviceType <i>T_BA_SmallString</i>
— eAssignAsTrendRef <i>E_BA_AssignRefMode</i>
— sObjectName <i>T_MaxString</i>
— sDescription <i>T_MaxString</i>
— sTag <i>STRING(XBA_Param.nTag_Length)</i>
— eStateTextDeterminationMode <i>E_BA_NodeTypeTarget</i>
— aStateText <i>T_BA_StateTextArray</i>

The function block FB_BAStateMV enables the display of states. Each referenced object is seen as *state*. It is represented as *multi-state object*.

The state texts of this *multi-state object* are automatically the names of all commanded references.

The PresentValue is automatically the highest active state, or reference.

● Abstract

The FB serves as a base (*ABSTRACT*) for providing described functionalities for inheriting FBs.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseStateMV EXTENDS FB_BA_Object IMPLEMENTS I_BA_MultistateObject
VAR_INPUT CONSTANT PERSISTENT
{region 'Fixed Parameters'}
    eStateTextDeterminationMode : E_BA_NodeTypeTarget := E_BA_NodeTypeTarget.eFunction;
    aStateText                  : T_BA_StateTextArray;
{endregion}
END_VAR
VAR
{region 'Informational'}
    stActiveInfo : ST_BA_ActiveInfo;
{endregion}
{region 'Output-Properties'}
    nPresentValue : UDINT := 1;
    nStateCount   : UDINT;
{endregion}
END_VAR
```

🔧 Inputs CONSTANT PERSISTENT

Name	Type	Description
eStateTextDeterminationMode	E_BA_NodeTypeTarget [► 104]	Defines the reference to the object whose description is displayed to represent the state.
aStateText	T_BA_StateTextArray [► 112]	The array is used to declare the state texts of a multi-state object.

VAR

Name	Type	Description
stActiveInfo	ST_BA_ActiveInfo [► 113]	Description of the object.
nPresentValue	UDINT	Current value for multi-stage outputs.
nStateCount	UDINT	Number of states of a multi-state object.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.9 FB_BA_BaseStepMV



The function block FB_BA_BaseStepMV enables the processing of steps. Each referenced object is seen as a step and can be commanded accordingly.

The *PresentValue* is automatically the current step, or the current reference, commanded by this object.



Abstract

The FB serves as a base (*ABSTRACT*) for providing described functionalities for inheriting FBs.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB BA Object [► 195]

FB BA_BaseStateMV [► 189]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseStepMV EXTENDS FB_BA_BaseStateMV
VAR_OUTPUT
    sStep      : STRING;
    bOn        : BOOL;
END_VAR
```

Outputs

Name	Type	Description
sStep	STRING	Current commanded step.
bOn	BOOL	Indicates operation.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.10 FB_BA_ComEventObject



The function block FB_BA_ComEventObject contains the basic properties or functions for recording, displaying, forwarding, acknowledging and resetting the events of an object.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB BA Object [► 195]

[FB_BA_EventObject \[► 192\]](#)

[FB_BA_EventObjectEx \[► 194\]](#)

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_ComEventObject EXTENDS FB_BA_EventObjectEx IMPLEMENTS I_BA_ComObject
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    eCommissioningState      : E_BA_CommissioningState := E_BA_CommissioningState.eUnknown;
{endregion}
END_VAR
```

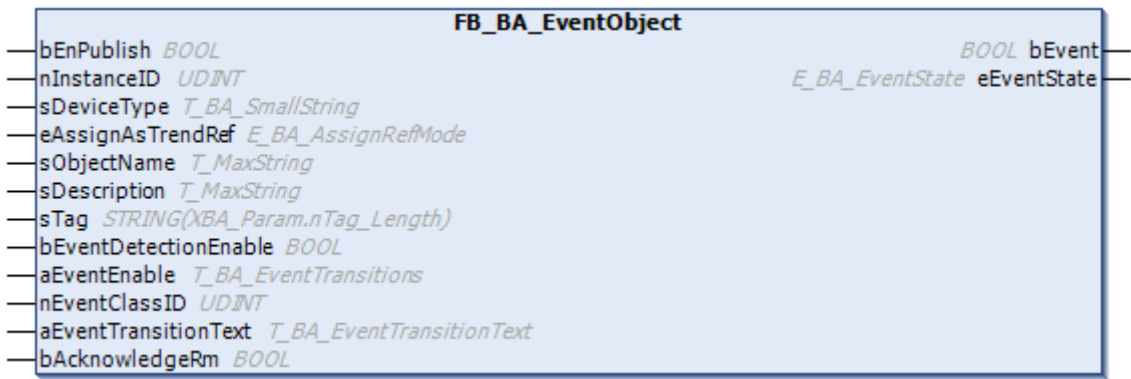
🔧 Inputs **CONSTANT PERSISTENT**

Name	Type	Description
eCommissioningState	E_BA_CommissioningState [► 101]	Parameterization of the commissioning state of an object. The settings of the commissioning state for the objects are made with the Site Explorer tool.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

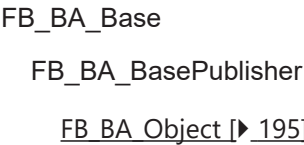
6.1.2.1.3.2.2.6.11 **FB_BA_EventObject**



The function block FB_BA_EventObject extends the basic functionalities of a BACnet object by the properties of the object-internal notification (Intrinsic Reporting). Here, notifications and alarm messages are generated by the BACnet object itself and forwarded to a notification class.

In this object the corresponding events must be enabled. This is done via the property *Event_Enable*. Each event (TO_OFFNORMAL, TO_FAULT and TO_NORMAL) can be enabled individually.

Inheritance hierarchy



Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_EventObject EXTENDS FB_BA_Object IMPLEMENTS I_BA_EventObject, I_BA_EventValue
VAR_OUTPUT
    bEvent      : BOOL;
    eEventState : E_BA_EventState;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
```

```

    bEventDetectionEnable      : BOOL := TRUE;
    nEventClassID              : UDINT;
    aEventTransitionText       : T_BA_EventTransitionText := BA_Param.aEventTransitionText;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Operational Parameters'}
    bAcknowledgeRm            : BOOL;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    eEventType                : E_BA_EventType := E_BA_EventType.Invalid;
    eAlarmMode                 : E_BA_AlarmMode := E_BA_AlarmMode.Invalid;
    eReliability                : E_BA_Reliability;
    eEventTransition            : E_BA_EventTransition := E_BA_EventTransition.Invalid;
  {endregion}
END_VAR
VAR
  {region 'Event'}
    stStateFlags               : ST_BA_StatusFlags;
    stAckedTransitions          : ST_BA_EventTransitions;
    aEventEnable                : T_BA_EventTransitions;
  {endregion}
END_VAR

```

Outputs

Name	Type	Description
bEvent	BOOL	The variable signals an abnormal event of an object. The type of event, e.g. an alarm, a maintenance message, etc., is described within the event class associated with the object (see Event).
eEventState	E_BA_EventState	The variable describes the state of an event.

Inputs CONSTANT PERSISTENT

Name	Type	Description
bEventDetectionEnable	BOOL	The variable indicates whether intrinsic reporting is enabled in the object. It must be TRUE to activate. It controls whether the events are taken into account within further processing, e.g. in event lists.
nEventClassID	UDINT	This property specifies the instance of the event class to be used for event notification distribution (see Events [► 28])
aEventTransitionText	T_BA_EventTransitionText	The array contains three strings, which are the base for the message texts of the events TO_OFFNORMAL, TO_FAULT and TO_NORMAL represent. The message texts are stored in the list <i>TxtEvent_EN</i> or <i>TxtEvent_DE</i> . Depending on the language selection, the English or German text list is installed when TF8040 is installed. The message texts are concatenated with other strings to a message in TF8040 (see BA_BACnetParam).

Inputs CONSTANT

Name	Type	Description
bAcknowledgeRm	BOOL	Input for local acknowledgement of the events of an object.

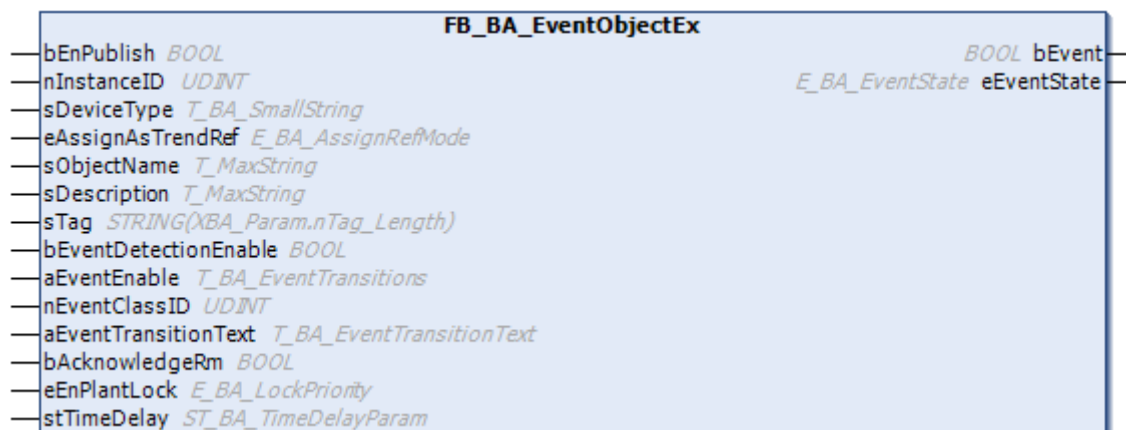
VAR

Name	Type	Description
eEventType	E_BA_EventType [► 94]	Event type.
eAlarmMode	E_BA_AlarmMode [► 92]	Definition of how to deal with an alarm.
eReliability	E_BA_Reliability	This variable indicates whether the value of the <i>Present_Value</i> property is reliable (NO_FAULT_DETECTED), otherwise it represents the reason (e.g. short circuit, missing sensor, etc.).
eEventTransition	E_BA_EventTransition	Mapping of the BACnet data type <i>BACnetEventTransitionBits</i> .
stStateFlags	ST_BA_StatusFlags	Mapping of the BACnet property <i>Status_Flags</i> .
stAckedTransitions	ST_BA_EventTransitions	Mapping of the BACnet Properties <i>Acked_Transitions</i> .
aEventEnable	T_BA_EventTransitions	Inside the array there are three event-enable bits. This allows the detection of the TO_OFFNORMAL, TO_FAULT and TO_NORMAL state changes on the object to be selectively suppressed. aEventEnable[1] = TO_OFFNORMAL aEventEnable[2] = TO_FAULT aEventEnable[3] = TO_NORMAL

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.12 FB_BA_EventObjectEx



The function block FB_BA_EventObjectEx extends the basic functionalities of a BACnet object by the property object internal notification (Intrinsic Reporting). Here, notifications and alarm messages are generated by the BACnet object itself and forwarded to a notification class.

In addition, the object is extended by the property of a time delay (*stTimeDelay*) of the event processing (*eEnPlantLock*).

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_EventObject [► 192]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_EventObjectEx EXTENDS FB_BA_EventObject IMPLEMENTS I_BA_EventObjectEx
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
    eEnPlantLock      : E_BA_LockPriority := E_BA_LockPriority.eNoLock;
    stTimeDelay       : ST_BA_TimeDelayParam := F_BA_TimeDelay(BA_Param.nDefTimeDelay_ToAbnormal, 0);
{endregion}
END_VAR
```

🔧 Inputs CONSTANT PERSISTENT

Name	Type	Description
eEnPlantLock	E_BA_LockPriority [► 95]	The enumeration can be used to parameterize various switching actions related to the event of an object. In the event of a fire damper malfunction, for example, it may be necessary to shut down the associated ventilation system (see PlantLock [► 30]).
stTimeDelay	ST_BA_TimeDelayParam [► 112]	The structure can be used to delay the object's event algorithm for the TO_OFFNORMAL and TO_NORMAL state change. In the standard PLC <i>PLC BA-Template</i> is the list BA2_Param. In it, the delay times for TO_OFFNORMAL and TO_NORMAL are pre-initialized with one second each by means of the two variables <i>nDefTimeDelay_ToAbnormal</i> and <i>nDefTimeDelay_ToNormal</i> .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.2.2.6.13 FB_BA_Object

FB_BA_Object	
— bEnPublish	BOOL
— nInstanceID	UDINT
— sDeviceType	T_BA_SmallString
— eAssignAsTrendRef	E_BA_AssignRefMode
— sObjectName	T_MaxString
— sDescription	T_MaxString
— sTag	STRING(BA_Param.nTag_Length)

The function block FB_BA_Object represents the basis for the development of further objects.

These objects must be called cyclically in the project, here the suspension of calls (e.g. within IF conditions) prevents a correct initialization.



Objects should be declared in a VAR or VAR_INPUT CONSTANT area.
Persistent declaration areas must be avoided!



Within inherited FBs the base FB must be called with "SUPER^()"

Initialization

While an object is initialized, it passes through several states in succession:

- All objects initialize until "E_BA_ObjectState.eWaitForInit"
- Then Top gives the sign to continue object initialization
- All objects continue until "E_BA_ObjectState.eWaitForProject"
- Top starts the project when all objects are set to "E_BA_ObjectState.eWaitForProject".
- All objects now go to "E_BA_ObjectState.eOperation"
- Top signals for one more cycle "E_BA_ProjectState.eFirstOpCycle"
- Then the project also goes into "E_BA_ProjectState.eOperation"

Inheritance hierarchy

FB_BA_BasePublisher

FB_BA_Object [► 195]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_Object EXTENDS FB_BA_BasePublisher IMPLEMENTS I_BA_Object, I_BACnet_ObjectOwner
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    nInstanceID      : UDINT;
    sDeviceType       : T_BA_SmallString := XBA_Globals.sPlaceholder_Empty;

    eAssignAsTrendRef : E_BA_AssignRefMode := E_BA_AssignRefMode.eInitByProfile;
  {endregion}
  {region 'Variable Parameters'}
    sObjectName       : T_MaxString := XBA_Globals.sPlaceholder_Empty;
    sDescription       : T_MaxString := XBA_Globals.sPlaceholder_Empty;
    sTag               : STRING(XBA_Param.nTag_Length);
  {endregion}
END_VAR
VAR PERSISTENT
  {region 'General'}
    sInstObjectName   : STRING;
    sInstDescription  : STRING;
  {endregion}
END_VAR
VAR
  {region 'Fixed Variables'}
    iParent           : I_BA_View;
    iLabel             : I_BA_Label;
    iProfile           : I_BA_Profile;
  {endregion}
  {region 'SubInit'}
    eObjectType        : E_BA_ObjectType := E_BA_ObjectType.eObject;
    stAttributes       : ST_BA_ObjectAttributes;
  {endregion}
END_VAR
VAR
  {region 'General'}
    nLevel             : UINT := 0; // Current hierarchy level according to root object (which has
                                // level 0)
    eState              : E_BA_ObjectState := E_BA_ObjectState.First;
    sFmtPresentValue    : STRING; // Present value in a formatted string
  {endregion}
END_VAR
```

Inputs CONSTANT PERSISTENT

Name	Type	Description
nInstanceID	UDINT	This property is a numeric code, for identifying the object. It must be unambiguous inside the device.
sDeviceType	T_BA_SmallString	This property is a text description of the physical device connected to the analog output.
eAssignAsTrendRef	E_BA_AssignRefMode	Reference mode.
sDescription	T_MaxString	This property represents a description text for the object. In the base framework of TF8040, each level can receive a substring. The substrings of the levels are concatenated by the DPAD mechanism so that the complete name of the object is created by means of this algorithm (see DPAD).
sTag	STRING	This property represents an additional description text for the object. The string <i>sTag</i> can be applied user-specific or project-specific. To enter the control cabinet device identification, for example.

Inputs PERSISTENT

Name	Type	Description
sInstObjectName	STRING	Object name of the instance.
sInstDescription	STRING	Instance description.

VAR

Name	Type	Description
iParent	I_BA_Parent	Parent object interface.
iLabel	I_BA_Label	Interface description.
iProfile	I_BA_Profile	Profile interface.
eObjectType	E_BA_ObjectType [▶ 98]	Selection of the object type.
stAttributes	ST_BA_ObjectAttributes [▶ 106]	Attribute definition.
nLevel	UINT	Current hierarchical level according to the base object (level 0).
eState	E_BA_SubscriberState [▶ 105]	Evaluation of the subscriber state.
sFmtPresentValue	STRING	Current value in a formatted string.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.3 Functions

6.1.2.1.3.3.1 Parameter

6.1.2.1.3.3.1.1 Builder

Using *builders* simplifies the initialization of complex parameters.

Operating principle

The functions offer the possibility of initializing individual values. Parameter-specific methods are provided for this purpose, such as methods for creating a [calendar](#) [► 112]:

```
F_BA_CalendarBuilder()
  .AppendDate(2020, E_BA_Month.eMarch, 10)
  .AppendDate(2020, E_BA_Month.eSeptember, 15)
  .Build()
```

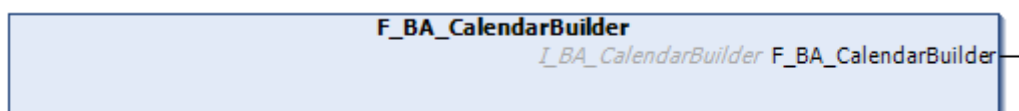
The benefits are:

- Self-explanatory notation
- Highly flexible
- Equally applicable in declaration and implementation code

See also:

[Examples](#) [► 82]

6.1.2.1.3.3.1.1.1 F_BA_CalendarBuilder



The function provides methods that simplify the initialization of [calendars](#) [► 112]:

```
Call1 : FB_BA_Cal := (
  iParent      := Cals,
  sDescription := '{Calendar 1}',
  aDateList    := F_BA_CalendarBuilder()
               .AppendDate(2021, E_BA_Month.eMarch, 10)
               .AppendDateRange(nFromYear := 2021, E_BA_Month.eMarch, 10, nToYear := 2021, E_BA_Month.eMarch, 15)
               .AppendWeekNDay(E_BA_Weekday.eMonday, E_BA_Week.Unspecified, E_BA_Month.Unspecified)
               .Build()
);
```

Methods

Name	Description
AppendDate [► 199]	Adds a date.
AppendDateRange [► 199]	Adds a date range.
AppendWeekNDay [► 200]	Adds a specific day of the week within a month.

See also:

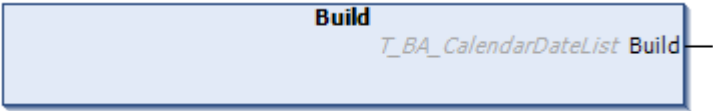
[Samples](#)

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.3.1.1.1.1 **Management**

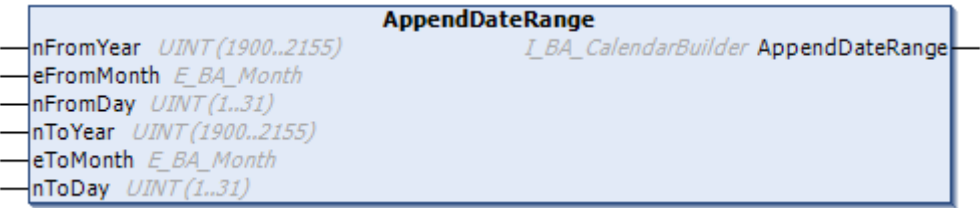
6.1.Build
2.1.
3.3.
1.1.
1.1.
1



The method outputs the contents of a date list [► 112].

6.1.2.1.3.3.1.1.1.2 **Value**

6.1.AppendDateRange
2.1.
3.3.
1.1.
1.2.
1



The method adds a defined period to the initialized calendar.

Illustration

```
METHOD AppendDateRange : I_BA_CalendarBuilder
VAR_INPUT
    nFromYear      : UINT(1900 .. 2155);
    eFromMonth     : E_BA_Month := E_BA_Month.Unspecified;
    nFromDay       : UINT(1 .. 31);
    nToYear        : UINT(1900 .. 2155);
    eToMonth       : E_BA_Month := E_BA_Month.Unspecified;
    nToDay         : UINT(1 .. 31);
END_VAR
```

 Inputs

Name	Type	Description
nFromYear	UINT	Start year from 1900 to 2155.
eFromMonth	E_BA_Month	Start month
nFromDay	UINT	Start day of a month from 1 to 31.
nToYear	UINT	End year from 1900 to 2155.
eToMonth	E_BA_Month	End month
nToDay	UINT	End day of a month from 1 to 31.

6.1.AppendDate
2.1.
3.3.
1.1.
1.2.
2

```

AppendDate
nYear  UINT(1900..2155)      I_BA_CalendarBuilder AppendDate
eMonth  E_BA_Month
nDay    UINT(1..31)

```

The method appends a date value to the initialized calendar.

Illustration

```

METHOD AppendDate : I_BA_CalendarBuilder
VAR_INPUT
  nYear      : UINT(1900 .. 2155);
  eMonth     : E_BA_Month := E_BA_Month.Unspecified;
  nDay       : UINT(1 .. 31);
END_VAR

```

Inputs

Name	Type	Description
nYear	UINT	Year from 1900 to 2155.
eMonth	E_BA_Month	Month
nDay	UINT	Day of a month from 1 to 31.

6.1.AppendWeekNDay

2.1.

3.3.

1.1.

1.2.

3

```

AppendWeekNDay
eWeekday E_BA_Weekday      I_BA_CalendarBuilder AppendWeekNDay
eWeekOfMonth E_BA_Week
eMonth E_BA_Month

```

The method appends a defined day to the initialized calendar, in a given week of a month, in a given month.

Illustration

```

METHOD AppendWeekNDay : I_BA_CalendarBuilder
VAR_INPUT
  eWeekday      : E_BA_Weekday := E_BA_Weekday.Invalid;
  eWeekOfMonth  : E_BA_Week    := E_BA_Week.Invalid;
  eMonth        : E_BA_Month   := E_BA_Month.Invalid;
END_VAR

```

Inputs

Name	Type	Description
eWeekday	E_BA_Weekday	Day of the week.
eWeekOfMonth	E_BA_Week	Week within a month.
eMonth	E_BA_Month	Month.

6.1.2.1.3.3.1.1.2 F_BA_ExceptionScheduleBuilder

```

F_BA_ExceptionScheduleBuilder
I_BA_ExceptionScheduleBuilder F_BA_ExceptionScheduleBuilder

```

The function provides methods that simplify the initialization of exception calendars:

```
Sched      : FB_BA_SchedA := (  
    sDescription      := '{Scheduler 1}',  
    aException        := F_BA_ExceptionScheduleBuilder()  
        .AppendDate(2021, E_BA_Month.eMarch, 10)  
        .AppendA(T#0H, 1.0).AppendA(T#6H, 2.0).AppendNull(T#12H)  
        .Close()  
        .Build()  
);
```

Methods

Name	Description
AppendDate [▶ 2021]	Adds a date.

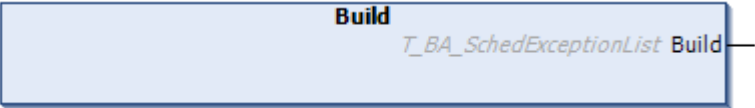
See also:
Samples

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.3.1.1.2.1 Management

6.1.Build
2.1.
3.3.
1.1.
2.1.
1



The method outputs the content [Exception calendar \[\[▶ 113\]\(#\)\]](#).

6.1.2.1.3.3.1.1.2.2 Value

6.1.AppendDateRange
2.1.
3.3.
1.1.
2.2.
1



The method adds a defined period to the initialized calendar.

Illustration

```
METHOD AppendDateRange : I_BA_CalendarBuilder  
VAR_INPUT  
    nFromYear      : UINT(1900 .. 2155);  
    eFromMonth      : E_BA_Month := E_BA_Month.Unspecified;
```

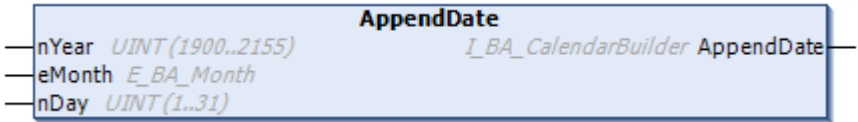
```
nFromDay      : UINT(1 .. 31);
nToYear       : UINT(1900 .. 2155);
eToMonth      : E_BA_Month := E_BA_Month.Unspecified;
nToDay        : UINT(1 .. 31);
END_VAR
```

 Inputs

Name	Type	Description
nFromYear	UINT	Start year from 1900 to 2155.
eFromMonth	E_BA_Month	Start month
nFromDay	UINT	Start day of a month from 1 to 31.
nToYear	UINT	End year from 1900 to 2155.
eToMonth	E_BA_Month	End month
nToDay	UINT	End day of a month from 1 to 31.

6.1.AppendDate

2.1.
3.3.
1.1.
2.2.
2



The method appends a date value to the initialized calendar.

Illustration

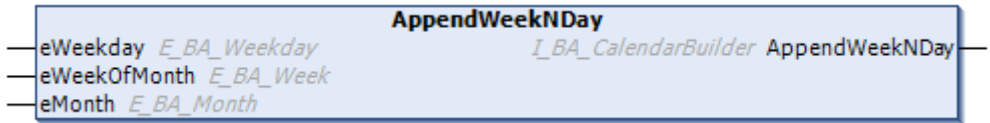
```
METHOD AppendDate : I_BA_CalendarBuilder
VAR_INPUT
  nYear      : UINT(1900 .. 2155);
  eMonth     : E_BA_Month := E_BA_Month.Unspecified;
  nDay       : UINT(1 .. 31);
END_VAR
```

 Inputs

Name	Type	Description
nYear	UINT	Year from 1900 to 2155.
eMonth	E_BA_Month	Month
nDay	UINT	Day of a month from 1 to 31.

6.1.AppendWeekNDay

2.1.
3.3.
1.1.
2.2.
3



The method appends a defined day to the initialized calendar, in a given week of a month, in a given month.

Illustration

```
METHOD AppendWeekNDay : I_BA_CalendarBuilder
VAR_INPUT
    eWeekday      : E_BA_Weekday      := E_BA_Weekday.Invalid;
    eWeekOfMonth  : E_BA_Week        := E_BA_Week.Invalid;
    eMonth        : E_BA_Month       := E_BA_Month.Invalid;
END_VAR
```

🚩 Inputs

Name	Type	Description
eWeekday	E_BA_Weekday	Day of the week.
eWeekOfMonth	E_BA_Week	Week within a month.
eMonth	E_BA_Month	Month.

6.1.2.1.3.3.1.1.3 F_BA_ParameterRolesBuilder

F_BA_ParameterRolesBuilder

I_BA_ParamRoleBuilder F_BA_ParameterRolesBuilder

The function provides methods that simplify the initialization of parameter roles:

```
Parameters      : FB_BA_Param := (
                    DefWriteAccess := F_BA_ParameterRolesBuilder()
                                    .Set(E_BA_Parameter.eEventEnable, E_BA_Role.eInternal)
                                    .Build()
                );
```

See also:

[FB_BA_Param](#) [► 124]

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.3.1.1.3.1 Management

- 6.1.Build
- 2.1.
- 3.3.
- 1.1.
- 3.1.
- 1

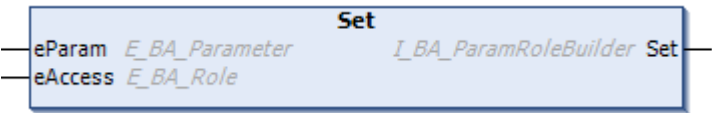
Build

ARRAY[E_BA_Parameter.First..E_BA_Parameter.Last] OF E_BA_Role Build

The method outputs the content of the possible [BACnet Properties](#) [► 101] according to the [user role](#) [► 98].

6.1.2.1.3.3.1.1.3.2 Value

- 6.1.Set
- 2.1.
- 3.3.
- 1.1.
- 3.2.
- 1



The method provided a BACnet property depending on the user role.

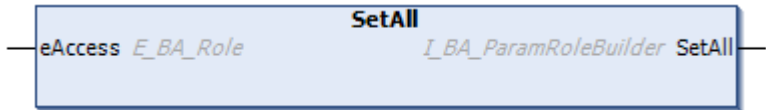
Illustration

```
METHOD Set : I_BA_ParamRoleBuilder
VAR_INPUT
    eParam      : E_BA_Parameter;
    eAccess     : E_BA_Role;
END_VAR
```

Inputs

Name	Type	Description
eParam	E BA Parameter [► 101]	BACnet property.
eAccess	E BA Role [► 98]	Definition of access rights.

- 6.1.SetAll
- 2.1.
- 3.3.
- 1.1.
- 3.2.
- 2



The method provided all BACnet properties depending on the user role.

Illustration

```
METHOD SetAll : I_BA_ParamRoleBuilder
VAR_INPUT
    eAccess     : E_BA_Role;
END_VAR
```

Inputs

Name	Type	Description
eAccess	E BA Role [► 98]	Definition of access rights.

- 6.1.SetRange
- 2.1.
- 3.3.
- 1.1.
- 3.2.
- 3



The method provided several BACnet properties depending on the user role.

Illustration

```

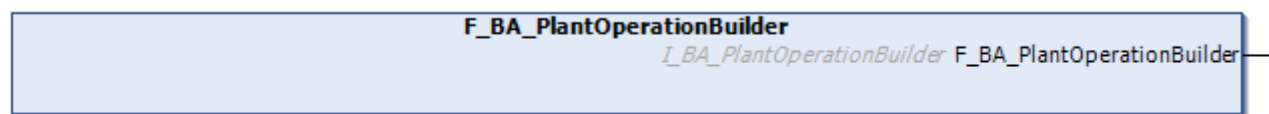
METHOD SetRange : I_BA_ParamRoleBuilder
VAR_INPUT
    eParamFrom    : E_BA_Parameter;
    eParamTo      : E_BA_Parameter;
    eAccess       : E_BA_Role;
END_VAR

```

Inputs

Name	Type	Description
eParamFrom	E_BA_Parameter [► 101]	First BACnet Property.
eParamTo	E_BA_Parameter [► 101]	Last BACnet Property.
eAccess	E_BA_Role [► 98]	Definition of access rights.

6.1.2.1.3.3.1.1.4 F_BA_PlantOperationBuilder



The function provides methods that simplify the initialization of the plant operation:

```

PlantCtrl : FB_BA_PlantCtrl := (
    stOperation      := F_BA_PlantOperationBuilder()
                    .AddAggregate(Dmpr,          FALSE, FALSE, T#0S, T#0S)
                    .AddAggregate(PreHtr,        TRUE,  FALSE, T#0S, T#0S)
                    .AddAggregate(Fan,           FALSE, FALSE, T#0S, T#10S)
                    .AddOperationMode(E_OpMod_AC.eDisturb, E_BA_Priority.eCritical, E_BA_AggregateIgnoreFlags.Delay)
                    .Close()
                    .AddOperationMode(E_OpMod_AC.eSmokeExtract, E_BA_Priority.eLifeSafety, E_BA_AggregateIgnoreFlags.All)
                    .SetAggregate(Dmpr,          100)
                    .SetAggregate(PreHtr,        E_OpMod_Binary.eOn)
                    .SetAggregate(Fan,           E_OpMod_Step2.eStep2)
                    .Close()
                    .AddOperationMode(E_OpMod_AC.eOff, E_BA_Priority.eProgram, 0)
                    .Close()
                    .AddOperationMode(E_OpMod_AC.eOn, E_BA_Priority.eProgram, 0)
                    .SetAggregate(Dmpr,          100)
                    .SetAggregate(PreHtr,        E_OpMod_Binary.eOn)
                    .SetAggregate(Fan,           E_OpMod_Step2.eStep1)
                    .Close()
                    .AddOperationMode(E_OpMod_AC.eNightCool, E_BA_Priority.eProgram, 0)
                    .SetAggregate(Dmpr,          100)
                    .SetAggregate(Fan,           E_OpMod_Step2.eStep2)
                    .Close()
                    .Build()
);

```

Methods

Name	Description
AddAggregate [► 206]	Adds an aggregate.
AddOperationMode [► 206]	Adds an operation mode.

See also:

FB_BA_PlantCtrl

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.3.1.1.4.1 Management

6.1.Build

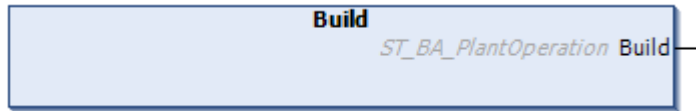
2.1.

3.3.

1.1.

4.1.

1



The method outputs the operating state [► 114] of the plant.

6.1.2.1.3.3.1.1.4.2 Value

6.1.AddAggregate

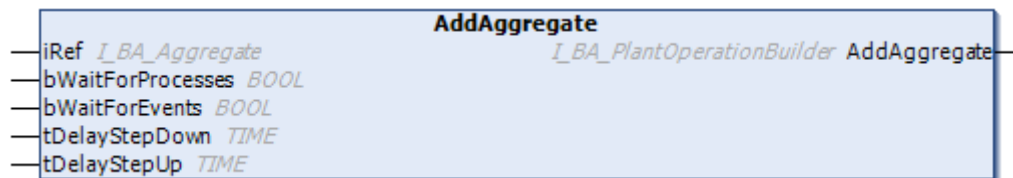
2.1.

3.3.

1.1.

4.2.

1



The method defines the behavior of aggregates.

Illustration

```
METHOD AddAggregate : I_BA_PlantOperationBuilder
VAR_INPUT
    iRef          : I_BA_Aggregate;
    bWaitForProcesses : BOOL      := FALSE;
    bWaitForEvents   : BOOL      := FALSE;
    tDelayStepDown   : TIME;
    tDelayStepUp     : TIME;
END_VAR
```

🔧 Inputs

Name	Type	Description
iRef	I_BA_Aggregate	Reference to the corresponding aggregate.
bWaitForProcesses	BOOL	Active processes are taken into account when switching.
bWaitForEvents	BOOL	Events of this (and higher) lock priority are taken into account when switching.
tDelayStepDown	TIME	Delay in step-down.
tDelayStepUp	TIME	Delay in step-up.

6.1.AddOperationMode

2.1.

3.3.

1.1.

4.2.

2

AddOperationMode

nOpMode UDINT

ePrio E_BA_Priority

eIgnore E_BA_AggregateIgnoreFlags

I_BA_PlantOperationModeBuilder

AddOperationMode

The method defines an operation mode for the plant operation.

Illustration

```
METHOD AddOperationMode : I_BA_PlantOperationModeBuilder
VAR_INPUT
    nOpMode      : UDINT;
    ePrio        : E_BA_Priority;
    eIgnore      : E_BA_AggregateIgnoreFlags := E_BA_AggregateIgnoreFlags.None;
END_VAR
```

🔧 Inputs

Name	Type	Description
nOpMode	UDINT	Operation mode.
ePrio	E_BA_Priority [► 96]	Priority
eIgnore	E_BA_AggregateIgnoreFlags [► 100]	Defines the switching conditions.

6.1.2.1.3.3.1.1.5 F_BA_ScheduleCalendarBuilder

F_BA_ScheduleCalendarBuilder

I_BA_ScheduleCalendarBuilder

F_BA_ScheduleCalendarBuilder

The function provides methods that simplify the initialization of schedules:

```
Sched : FB_BA_SchedA := (
    sDescription      := 'Scheduler 1',
    aCalendar         := F_BA_ScheduleCalendarBuilder()
                      .AppendReference(Call1)
                      .AppendA(T#0H, 1.0).AppendA(T#6H, 2.0).AppendNull(T#12H)
                      .Close()
                      .Build(),
);
```

🔧 Methods

Name	Description
AppendReference [► 208]	Adds a reference to a calendar.

See also:

[Samples](#)

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.3.1.1.5.1 Management

6.1.Build

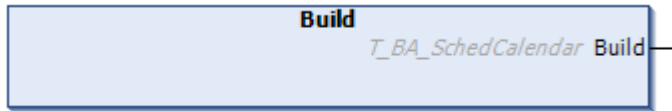
2.1.

3.3.

1.1.

5.1.

1



The method outputs the content of a [schedule](#) [► 113].

6.1.2.1.3.3.1.1.5.2 Value

6.1.AppendReference

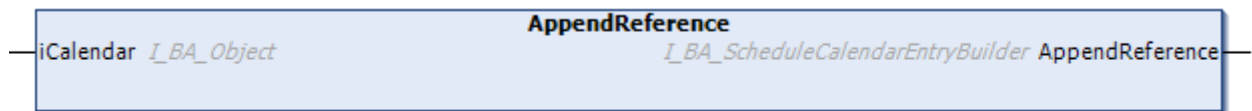
2.1.

3.3.

1.1.

5.2.

1



The method passes a calendar reference.

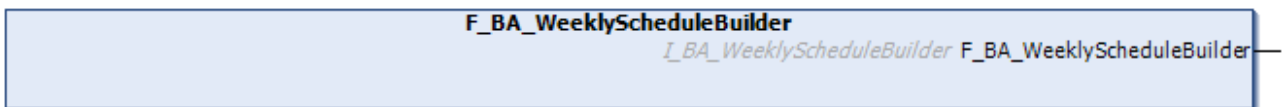
Illustration

```
METHOD AppendReference : I_BA_ScheduleCalendarEntryBuilder
VAR_INPUT
    iCalendar      : I_BA_Object;
END_VAR
```

Inputs

Name	Type	Description
iCalendar	I_BA_Object	Calendar reference.

6.1.2.1.3.3.1.1.6 F_BA_WeeklyScheduleBuilder



The function provides methods that simplify the initialization of weekly time schedules:

```
Sched    : FB_BA_SchedB := (
    sDescription      := 'Scheduler 1',
    aWeek             := F_BA_WeeklyScheduleBuilder()
                        .SetDayRange(E_BA_Weekday.eSaturday, E_BA_Weekday.eSunday)
                        .AppendB(T#0H, FALSE).AppendB(T#10H, TRUE).AppendB(T#12H, FALSE)
                        .Close()
                        .Build()
);
```

Methods

Name	Description
SetDayRange [► 210]	Sets the days when the calendar is active.

See also:

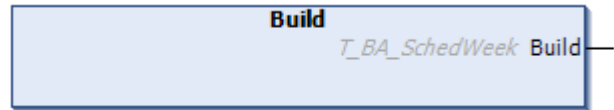
Samples

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.35	Tc3_XBA from v5.3.0.0

6.1.2.1.3.3.1.1.6.1 Management

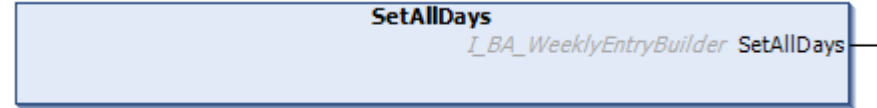
6.1.Build
2.1.
3.3.
1.1.
6.1.
1



The method outputs the content of a weekly schedule [► 113].

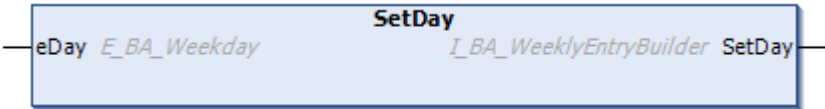
6.1.2.1.3.3.1.1.6.2 Value

6.1.SetAllDays
2.1.
3.3.
1.1.
6.2.
1



The method defines all days of the week for the weekly schedule.

6.1.SetDay
2.1.
3.3.
1.1.
6.2.
2



The method defines a specific day of the week for the weekly schedule.

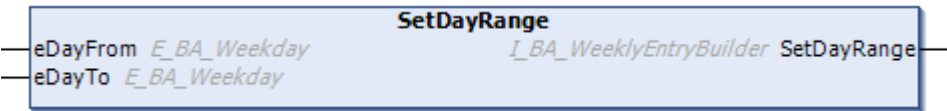
Illustration

```
METHOD SetDay : I_BA_WeeklyEntryBuilder
VAR_INPUT
    eDay      : E_BA_Weekday;
END_VAR
```

🔧 Inputs

Name	Type	Description
eDay	E_BA_Weekday	Selected day of the week.

6.1.SetDayRange
2.1.
3.3.
1.1.
6.2.
3



The method defines a period over several days for the weekly schedule.

Illustration

```
METHOD SetDayRange : I_BA_WeeklyEntryBuilder
VAR_INPUT
    eDayFrom      : E_BA_Weekday;
    eDayTo        : E_BA_Weekday;
END_VAR
```

🔧 Inputs

Name	Type	Description
eDayFrom	E_BA_Weekday	First day of the period.
eDayTo	E_BA_Weekday	Last day of the period.

- 6.1.2.2 Tc3_BA2
 - 6.1.2.2.1 DUTs
 - 6.1.2.2.1.1 Enumerations
 - 6.1.2.2.1.1.1 Room automation
 - 6.1.2.2.1.1.1.1 Heating, cooling
 - 6.1.2.2.1.1.1.1.1 E_BA_Medium

```
TYPE E_BA_Medium :
(
    Invalid           := 0,
    eNoMedium         := 1,
    eHeatMedium       := 2,
    eCoolMedium       := 3
) INT;
END_TYPE
```
 - 6.1.2.2.1.1.1.1.2 E_BA_PipeSys

```
TYPE E_BA_PipeSys :
(
    Invalid           := 0,
    e2Pipe            := 1,
```

```

    e4Pipe                := 2
) UDINT;
END_TYPE

```

6.1.2.2.1.1.1.2 Lighting

6.1.2.2.1.1.1.2.1 E_BA_LightActivationMode

The activation and deactivation of the light control functions can be done in two different ways: fully automatic or semi-automatic.

```

TYPE E_BA_LightActivationMode :
(
    eFullAutomatic,
    eSemiAutomatic
) BYTE;
END_TYPE

```

Name	Description
eFullAutomatic	Fully automatic: The light control function is activated by detected presence and deactivated by absence of presence. Button presses can also activate the function.
eSemiAutomatic	Semi-automatic: Activation of the light control function only by pressing a button; it is automatically deactivated when not occupied.

6.1.2.2.1.1.1.2.2 E_BA_LightingPrio

The enumeration contains the different priorities of the light control functions.

Not all priorities must necessarily be used in a project.

```

TYPE E_BA_LightingPrio:
(
    eNone                := 0,
    eFire                := 1,
    eCommError           := 2,
    eBurglary            := 3,
    eMaintenance         := 4,
    eManualActuator      := 5,
    eManualGroup         := 6,

    eCleaning            := 8,
    eNightWatch          := 9,

    eAllOff              := 10,
    eAllOn               := 11,
    eScene1              := 12,
    eConstantLightControl := 13,
    eAutomaticLight      := 14,
    eScene2              := 16,
    eScene3              := 17
) BYTE;
END_TYPE

```


Name	Description
eNone	The light control does not assume any function.
eFire	Fire alarm
eCommError	Behavior of the lighting in the error state, for example, communication interruption.
eBurglary	Burglary
eMaintenance	Maintenance
eManualActuator	Manual function
eManualGroup	Hand group function
eCleaning	Building cleaning
eNightWatch	Night watchman tour
eAllOff	Central off
eAllOn	Central on
eScene1	Scene 1
eConstantLightControl	Constant light control
eAutomaticLight	Automatic light
eScene2	Scene 2
eScene3	Scene 3

6.1.2.2.1.1.1.3 Sun protection

6.1.2.2.1.1.1.3.1 E_BA_PosMod

Enumerator for the definition of the positioning mode.

```

TYPE E_BA_PosMod :
(
  Invalid   := 0,
  eFix      := 1,
  eTab      := 2,
  eMaxIndc  := 3
) BYTE;
END_TYPE

```

Name	Description
eFix	The blind height is a fixed value, which is set at function block FB_BA_SunPrtc via the value <i>fFixPos</i> [%].
eTab	The height positioning takes place with the help of a table of 6 interpolation points, 4 of which are parameterizable. A blind position is then calculated from these points by linear interpolation, depending on the position of the sun (see FB_BA_BldPosEntry).
eMaxIndc	The positioning takes place with specification of the maximum desired incidence of light.

6.1.2.2.1.1.1.3.2 E_BA_ShObjType

Enumerator for selecting the shading object type.

```

TYPE E_BA_ShObjType :
(
  Invalid    := 0,
  eTetragon  := 1,
  eGlobe     := 2
) BYTE;
END_TYPE

```

Name	Description
eTetragon	Object type is a rectangle.
eGlobe	Object type is a ball.

6.1.2.2.1.1.3.3 E_BA_SunBldPrio

```

TYPE E_BA_SunBldPrio :
(
  Invalid           := 0,
  eFire             := 1,
  eStorm            := 2,
  eIce              := 3,
  eCommError        := 4,
  eBurglary         := 5,
  eMaintenance      := 6,
  eReferencing      := 7,
  eManualActuator   := 8,
  eManualGroup      := 9,
  eAllDown          := 10,
  eAllUp            := 11,
  eScene1           := 12,
  eFacadeThermoAutomatic := 13,
  eFacadeTwilightAutomatic := 14,
  eParkPosition     := 15,
  eScene2           := 16,
  eScene3           := 17,
  eSunProtection    := 18,
  eGroupThermoAuto  := 19,
  eGroupTwilightAuto := 20
) BYTE;
END_TYPE

```

Name	Description
eFire	Fire alarm
eStorm	Storm case
eIce	Icing up
eCommError	Behavior of the lighting in the error state, for example, communication interruption.
eBurglary	Burglary
eMaintenance	Maintenance
eReferencing	Central referencing
eManualActuator	Manual function
eManualGroup	Hand group function
eAllDown	Central down
eAllUp	Central up
eScene1	Scene 1
eFacadeThermoAutomatic	Facade wide thermal automatic
eFacadeTwilightAutomatic	Facade wide twilight automatic
eParkPosition	Parking position
eScene2	Scene 2
eScene3	Scene 3
eSunProtection	Sun protection in case of presence.
eGroupThermoAuto	Group or zone-wide thermal automatic.
eGroupTwilightAuto	Group or zone-wide twilight automatic.

6.1.2.2.1.1.2 Universal

6.1.2.2.1.1.2.1 E_BA_AntBlkgMode

```

TYPE E_BA_AntBlkgMedium :
(
  eOff              := 1,
  eExternalRequest  := 2,
  eOffTime          := 3
) UDINT;
END_TYPE

```

Name	Description
eOff	Deactivation of the function block
eExternalRequest	External request operation mode is active.
eOffTime	Minimum switch-off time operation mode is active.

6.1.2.2.1.1.2.2 E_BA_MdlT

Enum for the regulation of a drive or aggregate.

```
{attribute 'qualified_only'}
TYPE E_BA_MdlT :
(
  eOff      := 1,
  eOn       := 2,
  eMin      := 3,
  eMax      := 4
) UDINT;
END_TYPE
```

Name	Description
eOff	Switch off the drive.
eOn	Switching on the drive.
eMin	Minimum power step of the drive.
eMax	Maximum power step of the drive.

6.1.2.2.1.1.2.3 Control

6.1.2.2.1.1.2.3.1 E_BA_StateLoopSync

```
TYPE E_BA_StateLoopSync :
(
  eOk                := 1,
  eErrAction         := 2,
  eErrDampConstant   := 3,
  eErrDerivativeConstant := 4,
  eErrIntegralConstant := 5,
  eErrMaxOutputRm    := 6,
  eErrMinOutputRm    := 7,
  eErrNeutralZone    := 8,
  eErrOpMode         := 9,
  eErrProportionalConstant := 10
) UDINT;
END_TYPE
```

Name	Description
eErrAction	Synchronization of <i>eActionRm</i> faulty.
eErrDampConstant	Synchronization of <i>nDampConstant</i> faulty.
eErrDerivativeConstant	Synchronization of <i>fDerivativeConstant</i> faulty.
eErrIntegralConstant	Synchronization of <i>fIntegralConstant</i> faulty.
eErrMaxOutputRm	Synchronization of <i>fMaxOutputRm</i> faulty.
eErrMinOutputRm	Synchronization of <i>fMinOutputRm</i> faulty.
eErrNeutralZone	Synchronization of <i>fErrNeutralZone</i> faulty.
eErrOpMode	Synchronization of <i>eOpMode</i> faulty.
eErrProportionalConstant	Synchronization of <i>fProportionalConstant</i> faulty.

6.1.2.2.1.1.2.4 Sequence

6.1.2.2.1.1.2.4.1 E_BA_NoEnableSeqCtrl

```
TYPE E_BA_NoEnableSeqCtrl :
(
  eFirstSeqNum      := 1,
```

```

eLastSeqNum      := 2,
eFirstActionDirect := 3,
eLastActionReverse := 4,
eNextSeqCtrl     := 5,
ePreviousSeqCtrl  := 6,
eNextAction      := 7
) UDINT;
END_TYPE

```

6.1.2.2.1.1.2.4.2 E_BA_NoStartCtrlFound

```

TYPE E_BA_NoStartCtrlFound :
(
  eFirstSeqNum      := 1,
  eLastSeqNum       := 2,
  eFirstActionDirect := 3,
  eLastActionReverse := 4
) UDINT;
END_TYPE

```

6.1.2.2.1.1.2.4.3 E_BA_StateSeqLink

```

TYPE E_BA_StateSeqLink :
(
  eOff                := 1,
  eOn                 := 2,
  eMultipleDirectionActionSeqCtrl := 3,
  eNextSequenceSetpointIsSmaller := 4,
  eStartSequenceControllerNotEnabled := 5,
  eSeqNumMultiple     := 6,
  eErrorParamMaxSeqCtrl := 7,
  eNoSequenceControllerIsOperational := 8
) UDINT;
END_TYPE

```

Name	Description
eMultipleDirectionActionSeqCtrl	Multiple direction of action of the sequence controller.
eNextSequenceSetpointIsSmaller	Next sequence setpoint is smaller than its predecessor.
eStartSequenceControllerNotEnabled	The start controller <i>nNumStartCtrl</i> is not enabled. Instead, the controller with the lowest ordinal number is used.
eSeqNumMultiple	Sequence number assigned multiple times.
eErrorParamMaxSeqCtrl	Global parameter <i>nMaxSeqCtrl</i> < 1
eNoSequenceControllerIsOperational	No sequence controller is ready for operation.

6.1.2.2.1.2 Types

6.1.2.2.1.2.1 Room automation

6.1.2.2.1.2.1.1 Heating, cooling

6.1.2.2.1.2.1.1.1 ST_BA_SpRmT

Room temperature setpoints.

```

TYPE ST_BA_SpRmT :
STRUCT
  fPrtcHtg : REAL := 12.0;
  fEcoHtg  : REAL := 15.0;
  fPreCmfHtg : REAL := 19.0;
  fCmfHtg   : REAL := 21.0;
  fPrtcCol  : REAL := 40.0;
  fEcoCol   : REAL := 35.0;
  fPreCmfCol : REAL := 28.0;

```

```

    fCmfCol      : REAL := 24.0;
END_STRUCT
END_TYPE

```

The values in the structure are defined with the preset values.

Name	Type	Description
fPrtcHtg	REAL	Protection Heating
fEcoHtg	REAL	Economy Heating
fPreCmfHtg	REAL	Pre-Comfort Heating
fCmfHtg	REAL	Comfort Heating
fPrtcCol	REAL	Protection Cooling
fEcoCol	REAL	Economy Cooling
fPreCmfCol	REAL	Pre-Comfort Cooling
fCmfCol	REAL	Comfort Cooling

6.1.2.2.1.2.1.2 Lighting

6.1.2.2.1.2.1.2.1 ST_BA_Lighting

```

TYPE ST_BA_Lighting :
STRUCT
    {attribute 'parameterUnit':= '%'}
    fLgtVal      : REAL
    {attribute 'parameterUnit':= 'K'}
    fLgtT       : REAL;

    bActv       : BOOL;
    ePrio       : E_BA_LightingPrio;

    nEvtInc     : ULINT;
END_STRUCT
END_TYPE

```

Name	Type	Description
fLgtVal	REAL	Light value 0...100 %
fLgtT	REAL	Light temperature 0...1000001 K
bDimUp	BOOL	Dimming up
bDimDwn	BOOL	Dimming down
bDimMod	BOOL	Dimming mode; absolute values of <i>fLgtVal</i> are not accepted.
bActv	BOOL	Active
ePrio	E_BA_LightingPrio [► 211]	Priority of the active telegram.
nEvtInc	ULINT	Event - increment used as a timestamp.

6.1.2.2.1.2.1.3 Shading

6.1.2.2.1.2.1.3.1 ST_BA_BldPosTab

Structure of the interpolation point entries for the height adjustment of the blind.

```

TYPE ST_BA_BldPosTab:
STRUCT
    aSunElv     : ARRAY[0..5] OF REAL;
    aPos        : ARRAY[0..5] OF REAL;
    bVld        : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
aSunElv / aPos	REAL	The 6 interpolation points that are transferred, wherein the array elements 0 and 5 represent the automatically generated edge elements mentioned above.
bVld	BOOL	Validity flag for the function block FB_BA_SunPrtc. It is set to TRUE by the function block FB_BA_BldPosEntry if the data entered correspond to the validity criteria described.

6.1.2.2.1.2.1.3.2 ST_BA_FcdElem

List entry for a facade element (window).

```

TYPE ST_BA_FcdElem:
STRUCT
    fWdwWdth    : REAL;
    fWdwHght    : REAL;
    aCnr        : ARRAY [1..4] OF ST_BA_Cnr;
    nGrp        : DINT;
    bVld        : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
fWdwWdth	REAL	Width of the window.
fWdwHght	REAL	Height of the window.
aCnr	ST_BA_Cnr [► 217]	Coordinates of the window corners and information as to whether this corner point is in the shade.
nGrp	DINT	Indication of the group to which the window belongs.
bVld	BOOL	Plausibility of the entered data: <i>bVld</i> = TRUE: data are plausible.

6.1.2.2.1.2.1.3.3 ST_BA_Cnr

Information about window corners.

```

TYPE ST_BA_Cnr :
STRUCT
    fX          : REAL;
    fY          : REAL;
    bShdd       : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
fX	REAL	X-coordinate of the window (on the facade).
fY	REAL	Y-coordinate of the window (on the facade).
bShdd	BOOL	Information whether this corner point is shaded: <i>bShdd</i> = TRUE: corner point is shaded.

6.1.2.2.1.2.1.3.4 ST_BA_SunBld

Structure of the blind positioning telegram.

```

TYPE ST_BA_SunBld :
STRUCT
    fPos        : REAL;
    fAngl       : REAL;
    bManUp      : BOOL;
    bManDwn     : BOOL;
    bManMod     : BOOL;
    bActv       : BOOL;
    ePrio       : E_BA_SunBldPrio;
    nEvtInc     : UDINT;
END_STRUCT
END_TYPE

```

Name	Type	Description
fPos	REAL	Transferred blind height [%].
fAngl	REAL	Transferred slat position [°].
bManUp	BOOL	Manual command: blind up.
bManDwn	BOOL	Manual command: blind down.
bManMod	BOOL	TRUE: Manual mode is active. FALSE: Automatic mode is active.
bActv	BOOL	Priority control FB BA_SunBldPrioSwi4 [► 316], FB BA_SunBldPrioSwi8 [► 317], FB BA_SunBldTgmSel4 [► 323], FB BA_SunBldTgmSel8 [► 324] evaluated. The sun protection actuators FB BA_SunBldActr [► 307] and FB BA_RolBldActr [► 298] ignore it.
ePrio	E_BA_SunBldPrio [► 213]	Telegram priority. This enumeration is only evaluated by the priority control.
nEvtInc	EDINT	Telegram counter. With each new telegram, no matter which function block on the same controller triggers it, this counter is incremented by one. With telegrams of the same priority, the one with the higher counter "wins".

6.1.2.2.1.2.1.3.5 ST_BA_ShObj

List entry for a shading object.

```

TYPE ST_BA_ShObj :
STRUCT
  fP1x      : REAL;
  fP1y      : REAL;
  fP1z      : REAL;
  fP2x      : REAL;
  fP2y      : REAL;
  fP2z      : REAL;
  fP3x      : REAL;
  fP3y      : REAL;
  fP3z      : REAL;
  fP4x      : REAL;
  fP4y      : REAL;
  fP4z      : REAL;
  fMx       : REAL;
  fMy       : REAL;
  fMz       : REAL;
  fRads     : REAL;
  nBegMth   : USINT;
  nEndMth   : USINT;
  eType     : E_BA_ShObjType;
  bVld      : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
fP1x...fP4z	REAL	Corner coordinates. Of importance only if the element is a rectangle.
fMx...fMz	REAL	Center coordinates. Of importance only if the element is a sphere.
fRads	REAL	Radius of the ball. Of importance only if the element is a sphere.
nBegMth	USINT	Beginning of the shading period (month).
nEndMth	USINT	End of the shading period (month).
eType	E_BA_ShObjType [► 212]	Object type
bVld	BOOL	Plausibility of the data: <i>bVld</i> = TRUE: data are plausible.

Remark about the shading period:

The entries for the months may not be 0 or greater than 12, otherwise all combinations are possible.

Examples:

Start=1, End=1: shading in January.

Start=1, End=5: shading from the beginning of January to the end of May.

Start=11, End=5: shading from the beginning of November to the end of May (the following year).

6.1.2.2.1.2.1.3.6 ST_BA_SunBldScn

Table entry for a blind scene.

```
TYPE ST_BA_SunBldScn :
STRUCT
  fPos      : REAL;
  fAngl     : REAL;
END_STRUCT
END_TYPE
```

Name	Type	Description
fPos	REAL	Blind height [%]
fAngl	REAL	Slat angle [°]

6.1.2.2.1.2.2 Universal**6.1.2.2.1.2.2.1 Sequence****6.1.2.2.1.2.2.1.1 ST_BA_SeqLink**

```
TYPE ST_BA_SeqLink :
STRUCT
  arrSeqLinkData : ARRAY[1..BA_Param.nMaxSeqCtrl] OF ST_BA_SeqLinkData;
  fE              : REAL;
  nActvSeqCtrl   : UDINT;
  bSync          : BOOL;
  bEnSeqLink     : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
arrSeqLinkData	ST_BA_SeqLinkData [► 219]	Data and commands of the individual sequence controllers
fE	REAL	Control deviation Direct: $fE = fX - fW$ Indirect: $fE = fW - fX$
nActvSeqCtrl	UDINT	Number of the active sequence controller.
bSync	BOOL	Pulse - synchronization
bEnSeqLink	BOOL	Enable <i>bEn</i> of the FB_BA_SequenceLinkBase

6.1.2.2.1.2.2.1.2 ST_BA_SeqLinkData

```
TYPE ST_BA_SeqLinkData :
STRUCT
  fY      : REAL;
  fYMin   : REAL;
  fYMax   : REAL;
  fW      : REAL;
  fX      : REAL;
  fE      : REAL;
  nActvSeqCtrl : UDINT;
  nMyNum   : UDINT;
  eActn    : E_BA_Action;
  bSeqCtrlOperable : BOOL;
  bWatchdog : BOOL;
  bSeqNumMultiple : BOOL;
  bEn      : BOOL;
```



```

    bIsActvSeqCtrl      : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
fY	REAL	Control value
fYMin	REAL	Minimum control value
fYMax	REAL	Maximum control value
fW	REAL	Setpoint
fX	REAL	Actual value
fE	REAL	Control deviation Direct: $fE = fX - fW$ Indirect: $fE = fW - fX$
nActvSeqCtrl	UDINT	Number of the active sequence controller.
nMyNum	UDINT	My number in the sequence
eActn	E_BA_Action	Controller control direction
bSeqCtrlOperable	BOOL	Sequence controller is ready to operate.
bWatchdog	BOOL	Watchdog is set at each PLC cycle in each sequence controller and reset after evaluation in the sequence link.
bSeqNumMultiple	BOOL	Sequence number assigned several times in the sequence controllers.
bEn	BOOL	Enable the sequence controller.
blsActvSeqCtrl	BOOL	Active sequence controller

6.1.2.2.1.2.2.2 Aggregates

6.1.2.2.1.2.2.2.1 ST_BA_Multistate

The command structure is used to control multi-level aggregates and contains the priorities Safety, Critical and Program.

```

TYPE ST_BA_Multistate:
STRUCT
    bEnSfty      : BOOL;
    nValSfty     : UDINT;
    bEnCrit      : BOOL;
    nValCrit     : UDINT;
    bEnPgm       : BOOL;
    nValPgm      : UDINT;
END_STRUCT
END_TYPE

```

Name	Type	Description
bEnSfty	BOOL	Enable for writing the "Safety" priority.
nValSfty	UDINT	Value of the "Safety" priority to be written.
bEnCrit	BOOL	Enable for writing the "Critical" priority.
nValCrit	UDINT	Value of the "Critical" priority to be written.
bEnPgm	BOOL	Enable for writing the "Program" priority.
nValPgm	UDINT	Value of the "Program" priority to be written.

6.1.2.2.1.2.2.2.2 ST_BA_Analog

```

TYPE ST_BA_Analog :
STRUCT
    bEnSfty      : BOOL;
    fValSfty     : REAL;
    bEnCrit      : BOOL;
    fValCrit     : REAL;
    bEnPgm       : BOOL;

```

```
fValPgm      : REAL;
END_STRUCT
END_TYPE
```

Name	Type	Description
bEnSfty	BOOL	Enable for writing the "Safety" priority.
fValSfty	REAL	Value of the "Safety" priority to be written.
bEnCrit	BOOL	Enable for writing the "Critical" priority.
fValCrit	REAL	Value of the "Critical" priority to be written.
bEnPgm	BOOL	Enable for writing the "Program" priority.
fValPgm	REAL	Value of the "Program" priority to be written.

6.1.2.2.1.2.2.2.3 ST_BA_Mdlt

The command structure is used to control modulating aggregates and contains the priorities "Safety", "Critical" and "Program".

```
TYPE ST_BA_Mdlt :
STRUCT
  bEnSfty      : BOOL;
  eValSfty     : E_BA_Mdlt;
  bEnCrit     : BOOL;
  eValCrit     : E_BA_Mdlt;
  bEnPgm      : BOOL;
  eValPgm     : E_BA_Mdlt;
END_STRUCT
END_TYPE
```

Name	Type	Description
bEnSfty	BOOL	Enable for writing the "Safety" priority.
eValSfty	E_BA_Mdlt [► 214]	Enum of the "Safety" priority to be written.
bEnCrit	BOOL	Enable for writing the "Critical" priority.
eValCrit	E_BA_Mdlt [► 214]	Enum of the priority "Critical" to be written.
bEnPgm	BOOL	Enable for writing the "Program" priority.
eValPgm	E_BA_Mdlt [► 214]	Enum of priority "Program" to be written.

6.1.2.2.1.2.2.2.4 ST_BA_Binary

The command structure is used to control binary aggregates and contains the priorities Safety, Critical and Program.

```
TYPE ST_BA_Binary:
STRUCT
  bEnSfty      : BOOL;
  bValSfty     : BOOL;
  bEnCrit     : BOOL;
  bValCrit     : BOOL;
  bEnPgm      : BOOL;
  bValPgm     : BOOL;
END_STRUCT
END_TYPE
```

Name	Description

bEnSfty	BOOL	Enable for writing the "Safety" priority.
bValSfty	BOOL	Value of the "Safety" priority to be written.
bEnCrit	BOOL	Enable for writing the "Critical" priority.
bValCrit	BOOL	Value of the "Critical" priority to be written.
bEnPgm	BOOL	Enable for writing the "Program" priority.
bValPgm	BOOL	Value of the "Program" priority to be written.

6.1.2.2.1.2.2.5 ST_BA_Step

Data and command structure between the individual step sequence function blocks FB_BA_StepBinary / FB_BA_StepMdlT and the control block of the step sequence FB_BA_StepCtrlAgg16.

Illustration

```

TYPE ST_BA_Step:
STRUCT
  nStep          : UDINT;
  nRemSecOn      : UDINT;
  nRemSecOff     : UDINT;
  bEnUp          : BOOL;
  bEnDown        : BOOL;
  bRdyUp         : BOOL;
  bRdyDown       : BOOL;
  bUp            : BOOL;
  bDown          : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
nStep	UDINT	Display in which step the step sequence control is located.
nRemSecOn	UDINT	Countdown Switch-on of the next step [s]. The associated timing element is integrated in the step sequence function blocks FB_BA_StepBinary [► 420] / FB_BA_StepMdlT [► 423].
nRemSecOff	UDINT	Countdown Switch-off of the next step [s]. The associated timing element is integrated in the step sequence function blocks FB_BA_StepBinary [► 420] / FB_BA_StepMdlT [► 423].
bEnUp	BOOL	Indicates that the step has its release.
bEnDown	BOOL	The variable indicates that the active step is in the off state. After the switch-off delay <i>nRemSecOff</i> has elapsed, the step is switched off as well as <i>bEnDown</i> and the next step in the falling switch-off sequence becomes the active one.
bRdyUp	BOOL	After expiration of the start-up delay <i>nRemSecOn</i> and a TRUE at the feedback <i>bFdb</i> of the step sequence function block, <i>bRdyUp</i> is set and the next step is activated.
bRdyDown	BOOL	After the switch-off delay <i>nRemSecOff</i> has expired, <i>bRdyDown</i> is set for one cycle and the step is switched off.
bUp	BOOL	A TRUE means an increasing switch-on sequence of the aggregates from 1 to 16.
bDown	BOOL	A TRUE means a decreasing switch-off sequence from the highest, active aggregate towards 0.

6.1.2.2.1.2.2.6 ST_BA_PriorityEn

The command structure contains the releases of the plant and the priorities "Safety", "Critical" and "Program".

Illustration

```
TYPE ST_BA_PriorityEn:
STRUCT
    bPlt      : BOOL;
    bEnSfty   : BOOL;
    bEnCrit   : BOOL;
    bEnPgm    : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
bPlt	BOOL	Release plant.
bEnSfty	BOOL	Enable for writing the "Safety" priority.
bEnCrit	BOOL	Enable for writing the "Critical" priority.
bEnPgm	BOOL	Enable for writing the "Program" priority.

6.1.2.2.2 GVLs

6.1.2.2.2.1 BA_Globals

```
VAR_GLOBAL
{region 'Room-Automation'}
    nEvtIncLight    : ULINT;
    nEvtIncSunBld   : UDINT;
{endregion}
END_VAR
```

6.1.2.2.2.2 BA_Param

```
VAR_GLOBAL CONSTANT
{region 'Sun Protection'}
    nSunPrt_MaxSunBldScn    : USINT := 20;
    nSunPrt_MaxRowFcd       : UINT  := 10;
    nSunPrt_MaxColumnFcd    : UINT  := 20;
    nSunPrt_MaxShdObj       : UINT  := 20;
{attribute 'parameterUnit' := 'Byte'}
    nSunPrt_MaxDataFileSize : UDINT := 100000;
{endregion}
    {region 'Universal'}
        {region 'Sequence'}
            nMaxSeqCtrl      : USINT := 8;
        {endregion}
    {endregion}
END_VAR
```

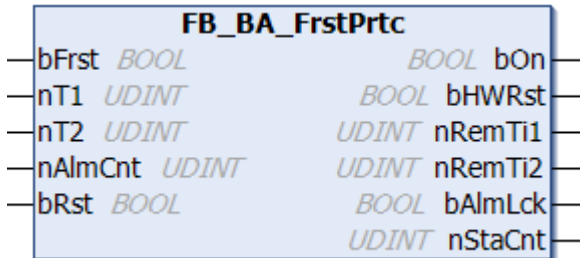
Name	Type	Description
nSunPrt_MaxSunBldScn	USINT	Maximum set of scenes that can be administered by a FB_BA_SunBldScn:
nSunPrt_MaxRowFcd	UINT	Maximum number of floors that can be monitored by the shading correction (horizontal orientation of windows).
nSunPrt_MaxColumnFcd	UINT	Maximum number of axes that can be monitored by the shading correction (vertical alignment of windows).
nSunPrt_MaxShdObj	UINT	Maximum number of shading objects that cast shadows on a facade.
nSunPrt_MaxDataFileSize	UDINT	Maximum file size [byte] for Excel lists to be read by the function blocks FB_BA_RdFcdElemLst and FB_BA_RdShdObjLst.
nMaxSeqCtrl	USINT	Maximum number of sequence controllers in a sequence.

6.1.2.2.3 POU's

6.1.2.2.3.1 Function blocks

6.1.2.2.3.1.1 Air conditioning equipment

6.1.2.2.3.1.1.1 FB_BA_FrstPrtc



The function block **FB_BA_FrstPrtc** is used for frost monitoring of a heating coil in an air conditioning system.

A frost risk is present, if the input *bFrst* is TRUE. The frost alarm must be linked in the plant program such that the plant is switched off immediately, the heater valve opens, and the heater pump is switched on.

If there is risk of frost, the output *bOn* is set, and *nT1* (seconds) is started. If the frost risk remains (*bFrst* = TRUE) after *nT1* has elapsed, *bOn* remains set. It can only be reset at input *bRst*.

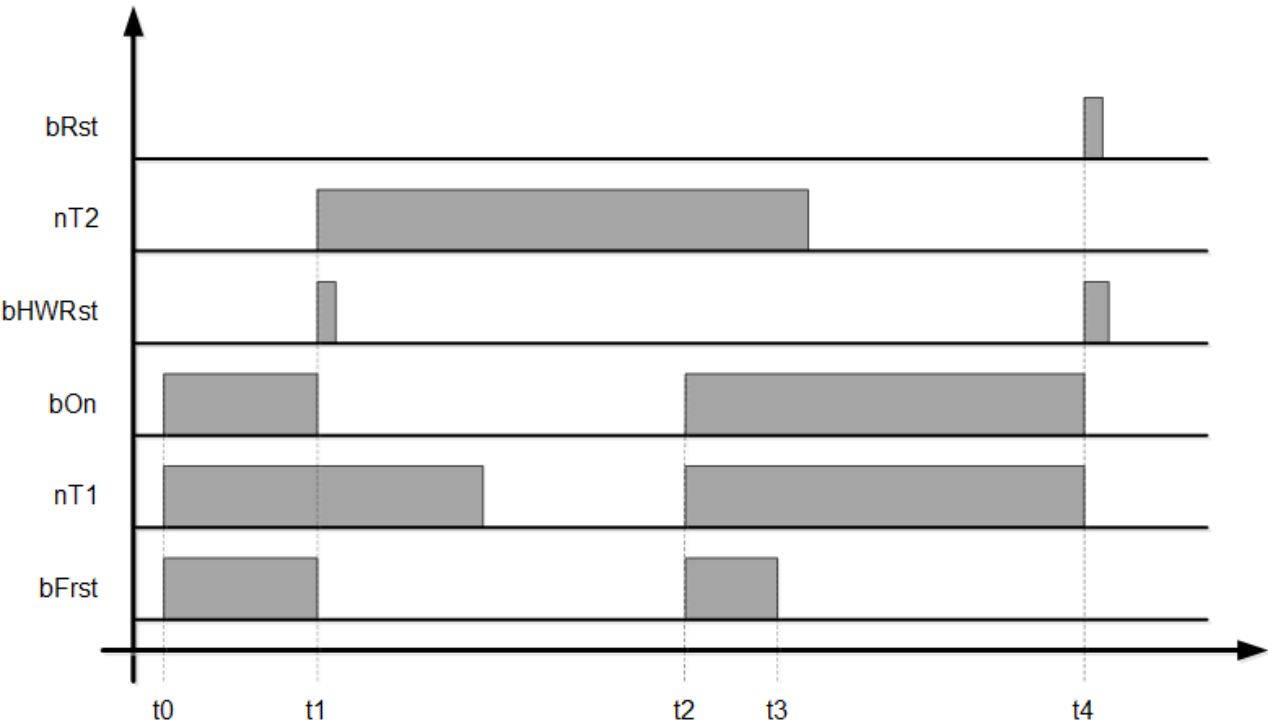
If the frost alarm ceases due to activation of the heating coil within the time *nT1* (*bFrst* = FALSE), the plant automatically restarts. For the plant restart *bOn* becomes FALSE, and at output *bHWRst* a pulse for acknowledgement of a latching circuit in the control cabinet is issued. With the restart a second monitoring period *nT2* (seconds) is initiated. If another frost alarm occurs within this period, the plant is permanently locked. *bOn* remains set until the frost alarm has been eliminated and *bRst* has been acknowledged.

In a scenario where frost alarms recur with time offsets that are greater than *nT2*, theoretically the plant would keep restarting automatically. In order to avoid this, the restarts within the function block are counted. The parameter *nAlmCnt* can be used to set the number of possible automatic restart between 0 and 4.

An acknowledgement at input *bRst* resets the alarm memory within the function block to zero.

Example:

t0 = frost alarm at input *bFrst*, alarm message at output *bOn*, start of timer T1 (*nT1* [s])
 t1 = frost alarm off, resetting of *bOn*, output of hardware pulse, start of timer T2 (*nT2* [s]), plant restart
 t2 = further frost alarm within T2, alarm message at *bOn*, start of timer T1, locking of the frost alarm
 t3 = frost alarm off.
 t4 = acknowledgement of the alarm at *bRst*, resetting of *bOn*.



🔧 Inputs

```
VAR_INPUT
    bFrst      : BOOL;
    nT1        : UDINT;
    nT2        : UDINT;
    nAlmCnt    : UDINT;
    bRst       : BOOL;
END_VAR
```

Name	Type	Description
bFrst	BOOL	Connection for frost events on the air and water side.
nT1	UDINT	Timer for restart delays [s]. Internally limited to a minimum value of 0.
nT2	UDINT	Timer monitoring time [s]. Internally limited to a minimum value of 0.
nAlmCnt	UDINT	Maximum number of automatic plant restarts without reset. Internally limited to values between 0 and 4.
bRst	BOOL	Resetting and acknowledgement of the frost alarm.

🔧 Outputs

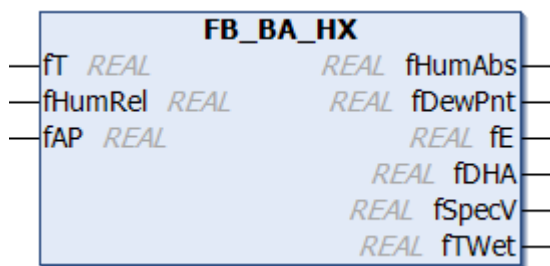
```
VAR_OUTPUT
    bOn        : BOOL;
    bHWRst     : BOOL;
    nRemTi1    : UDINT;
    nRemTi2    : UDINT;
    bAlmLck    : BOOL;
    nStaCnt    : UDINT;
END_VAR
```

Name	Type	Description
bOn	BOOL	Frost alarm active
bHWRst	BOOL	Output of a pulse for acknowledgement of the frost protection hardware
nRemTi1	UDINT	Time remaining to plant restart after frost alarm.
nRemTi2	UDINT	Remaining monitoring time.
bAlmLck	BOOL	Alarm lock - stored alarm.
nStaCnt	UDINT	Status counter – current number of unacknowledged false starts.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.1.2 FB_BA_HX



This function block FB_BA_HX is used to calculate the dew point temperature, the specific enthalpy and the absolute humidity. The temperature, the relative humidity and the barometric air pressure are required for the calculation of these parameters.

Enthalpy is a measure of the energy of a thermodynamic system.

Inputs

```
VAR_INPUT
  fT      : REAL;
  fHumRel : REAL;
  fAP     : REAL;
END_VAR
```

Name	Type	Description
fT	REAL	Temperature [°C].
fHumRel	REAL	Relative humidity [%]
fAP	REAL	Hydrostatic air pressure at 1013.25 hPa.

Outputs

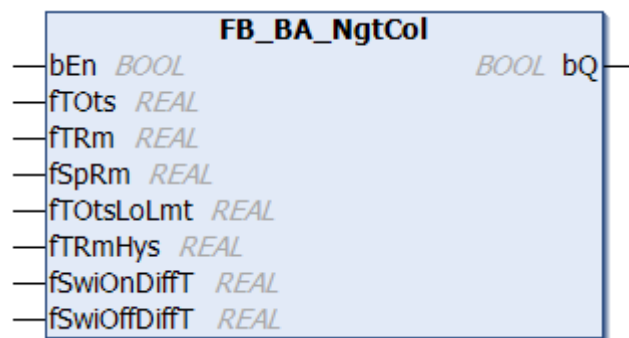
```
VAR_OUTPUT
  fHumAbs : REAL;
  fDewPnt : REAL;
  fE      : REAL;
  fDHA    : REAL;
  fSpecV  : REAL;
  fTWet   : REAL;
END_VAR
```

Name	Type	Description
fHumAbs	REAL	Absolute humidity g water per kg dry air [g/Kg].
fDewPnt	REAL	Dew point temperature [°C]
fE	REAL	Enthalpy [kJ/kg]
fDHA	REAL	Density of moist air ρ [kg mixture/m ³]
fSpecV	REAL	Specific volume [m ³ /kg]
fTWet	REAL	Wet bulb temperature [°C]

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.1.3 FB_BA_NgtCol



This function block FB_BA_NgtCol is used to cool down rooms that have been heated up during the day with cool outside air at night. The summer night cooling function serves to improve the quality of the air and to save electrical energy. Electrical energy for cooling is saved during the first hours of the next summer day.

The start conditions for the summer night cooling are defined by parameterizing the FB_BA_NgtCol function block. The function block can be used to open motor-driven windows or to switch air conditioning systems to summer night cooling mode outside their normal hours of operation.

Switching conditions

The following conditions must be met for activation of summer night cooling:

- The function block itself is enabled ($bEn = \text{TRUE}$).
- The outside temperature is not too low ($fTOts > fTOtsLoLmt$).
- The outside temperature is sufficiently low compared to the room temperature ($fTRm - fTOts > fSwiOnDiffT$).
- The room temperature is high enough to justify activating summer night cooling. $fTRm > fSpRm + fTRmHys$.

Under the following conditions the summer night cooling is disabled:

- The function block itself is disabled ($bEn = \text{FALSE}$).
- The outside temperature is too low ($fTOts < fTOtsLoLmt$).
- The outside temperature is too high compared to the room temperature ($fTRm - fTOts < fSwiOffDiffT$).
- The room temperature is lower than the setpoint. $fTRm \leq fSpRm$.

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  fTOts    : REAL;
  fTRm     : REAL;
  fSpRm    : REAL;
  fTOtsLoLmt : REAL;
```



```

fTOtsHys      : REAL;
fTRmHys       : REAL;
fSwiOnDiffT   : REAL;
fSwiOffDiffT  : REAL;
END_VAR

```

Name	Type	Description
bEn	BOOL	Enable function block.
fTOts	REAL	Outside temperature [°C]
fTRm	REAL	Room temperature [°C]
fSpRm	REAL	Room temperature setpoint
fTOtsLoLmt	REAL	Lower outside temperature limit [°C]; prevents excessive cooling.
fTOtsHys	REAL	Hysteresis for minimum outside temperature [°K]. This hysteresis, which at the lower end is internally limited to 0.5 K, is intended to prevent bQ from switching, if the outside temperature fluctuates precisely around the value of rTOtsLoLmt.
fTRmHys	REAL	Hysteresis for the room temperature [K]. This hysteresis, which at the lower end is internally limited to 0.5 K, is intended to prevent unnecessary switching of bQ, if the room temperature fluctuates precisely around the setpoint rSpRm.
fSwiOnDiffT	REAL	Difference between the room temperature and the outside temperature, from which summer night cooling is enabled [K].
fSwiOffDiffT	REAL	Difference between the room temperature and the outside temperature, from which summer night cooling is locked [K].

🔌 Outputs

```

VAR_OUTPUT
  bQ : BOOL;
END_VAR

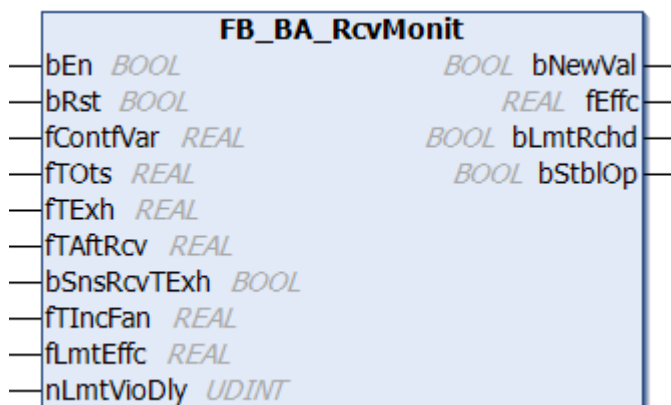
```

Name	Type	Description
bQ	BOOL	Summer night cooling On

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.1.4 FB_BA_RcvMonit



The function block FB_BA_RcvMonit is used to calculate the efficiency of an energy recovery system. The function block requires the following measured temperature values to calculate the efficiency (the so-called heat recovery coefficient):

- Outside air temperature $fTOts$
- Exhaust air temperature $fTExh$
- Air temperature of the energy recovery system in the supply air duct (alternatively: in the exhaust air duct) $fTAftRcv$

(alternative)

The function block logs the temperature values every 10 seconds and forms minutely averages from 6 consecutive values. The results are used to check whether the plant has reached a "stable" state.

- This is the case when the recorded temperatures of outside air, exhaust air and air after energy recovery are almost constant, i.e. none the 6 individual values deviate by more than 0.5 K from the respective average value.
- The temperature difference between outside air and exhaust air is at least 5 K.

If this is the case, this measuring cycle is acknowledged with a TRUE signal at output $bStbOp$, and the calculated efficiency is output at $fEffc$. If the state is not "stable", a FALSE signal appears at output $bStbOp$, and $fEffc$ is set to 0.

In any case, each measuring and analysis cycle is marked as completed with a trigger (a TRUE signal lasting one PLC cycle) at $bNewVal$.

Enable (bEn) and Reset ($bRst$)

The function block is only active if a TRUE signal is present at bEn . Otherwise its execution stops, and all outputs are set to FALSE or 0.0.

An active measuring and evaluation cycle can be terminated at any time by a TRUE signal at $bRst$. All outputs are set to FALSE or 0.0, and the measuring cycle restarts automatically.

Selection of the temperature value "after recovery" ($bSnsRcvTExh$)

A FALSE entry at $bSnsRcvTExh$ means that the temperature measurement after the heat recovery in the **supply air duct** is used for calculating the efficiency.

To use the temperature measurement after the heat recovery in the **exhaust air duct**, TRUE must be applied at $bSnsRcvTExh$.

Limit value exceeded ($fContrVar$, $fLmtEffc$, $bLmtRchd$)

A limit violation has occurred, if the calculated efficiency is less than the specified limit value $rLmtEffc$, and at the same time the control value for the heat recovery is at 100%. For this purpose, the manipulated variable must be connected to the input $fContrVar$.

The limit violation message can be delayed by an entry at $nLmtVioDly\_sec$ [s]: If the two criteria, violation and override, are present for longer than $nLmtVioDly\_sec$ [s], this is indicated by a TRUE signal at $bLmtRchd$.

A warning message that has occurred disappears if a complete measuring cycle provides "good" values or if there is a rising edge at $bRst$ or if the function block is deactivated.



This warning message only occurs if the plant is in a stable operation mode ($bStbOp=TRUE$).

Taking into account the temperature increase of the exhaust air due to the fan motor ($fTIncFan$)

It is possible that the outlet air is warmed by a fan motor, resulting in distortion of the measurement. This temperature increase can be specified through $fTIncFan$. Internally, the measured outlet air temperature is then reduced by this value.

Inputs

```

VAR_INPUT
    bEn          : BOOL;
    bRst         : BOOL;
    fContrVar    : REAL;
    fTOts        : REAL;
    fTExh        : REAL;
    fTAftRcv     : REAL;
    bSnsRcvTExh  : BOOL;
    fTIncFan     : REAL;
    fLmtEffc     : REAL;
    nLmtVioDly   : UDINT;
END_VAR

```

Name	Type	Description
bEn	BOOL	Enable function block.
bRst	BOOL	Reset - all determined values are deleted.
fContrVar	REAL	Control value for the heat recovery, i.e. the actual value.
fTOts	REAL	Outside temperature [°C]
fTExh	REAL	Exhaust air temperature [°C]
fTAftRcv	REAL	Temperature after energy recovery
bSnsRcvTExh	BOOL	Temperature at the measuring point after energy recovery: FALSE -> in the supply air duct (SupplyAir) - TRUE -> in the exhaust air duct (ExhaustAir).
fTIncFan	REAL	Temperature increase due to fan.
fLmtEffc	REAL	Limit value efficiency
nLmtVioDly	UDINT	Limit violation delay [s]. Internally limited to a minimum value of 0.

Outputs

```

VAR_OUTPUT
    bNewVal      : BOOL;
    fEffc        : REAL;
    bLmtRchd     : BOOL;
    bStblOp      : BOOL;
END_VAR

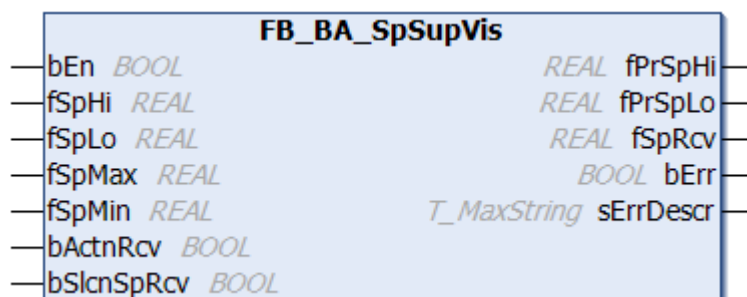
```

Name	Type	Description
bNewVal	BOOL	Output trigger for new value fEffc
fEffc	REAL	Efficiency
bLmtRchd	BOOL	Limit value reached
bStblOp	BOOL	Stable operation

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.1.5 FB_BA_SpSupVis



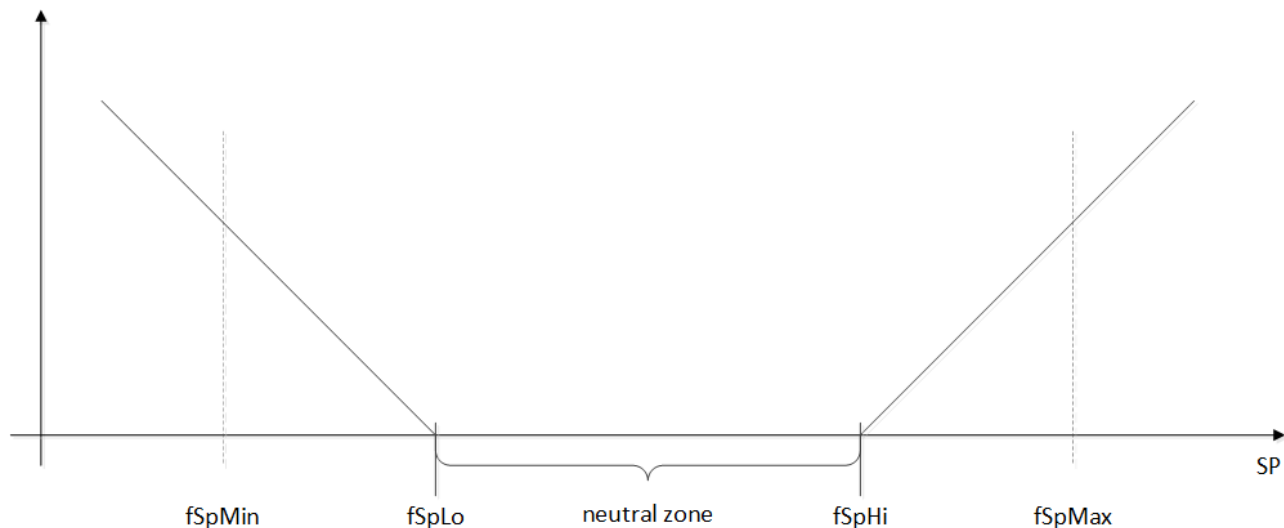
Function block FB_BA_SpSupVisdient for processing and checking the lower and upper setpoint of a supply air humidity or temperature control.

Checks and limits for the setpoints

The function block limits the setpoints. The following two tables show which parameters are checked and what the response is in the event of an error.

Checking		Action
$fSpLo > fSpHi$		Last valid values of $fSpLo$ and $fSpHi$ are used.
$fSpMin \geq fSpMax$		Last valid values of $fSpMin$ and $fSpMax$ are used.
$fSpHi > fSpMax$		$fPrSpHi = fSpMax$
$fSpLo < fSpMin$		$fPrSpLo = fSpMin$
Checking	bErr	Action
$fSpMin \geq fSpMax$	TRUE	$fSpErr = ((fSpMin + fSpMax) / 2)$
$fSpHi < fSpMin$		$fPrSpHi = fPrSpLo = fPrRcv = fSpErr$
$fSpLo > fSpMax$		

The difference between the setpoints describes an energy-neutral zone. With supply air control, no heating or cooling would take place within the neutral zone.



The checked and possibly limited setpoints are output at the function block output as $fPrSpHi$ and $fPrSpLo$ (Present Setpoint).

Setpoint for heat recovery

For a heat recovery system, the setpoint $fSpRcv$ is calculated either from the average of the upper and lower setpoints, $fSpHi$ and $fSpLo$ or as a function of the control direction of the heat recovery system.

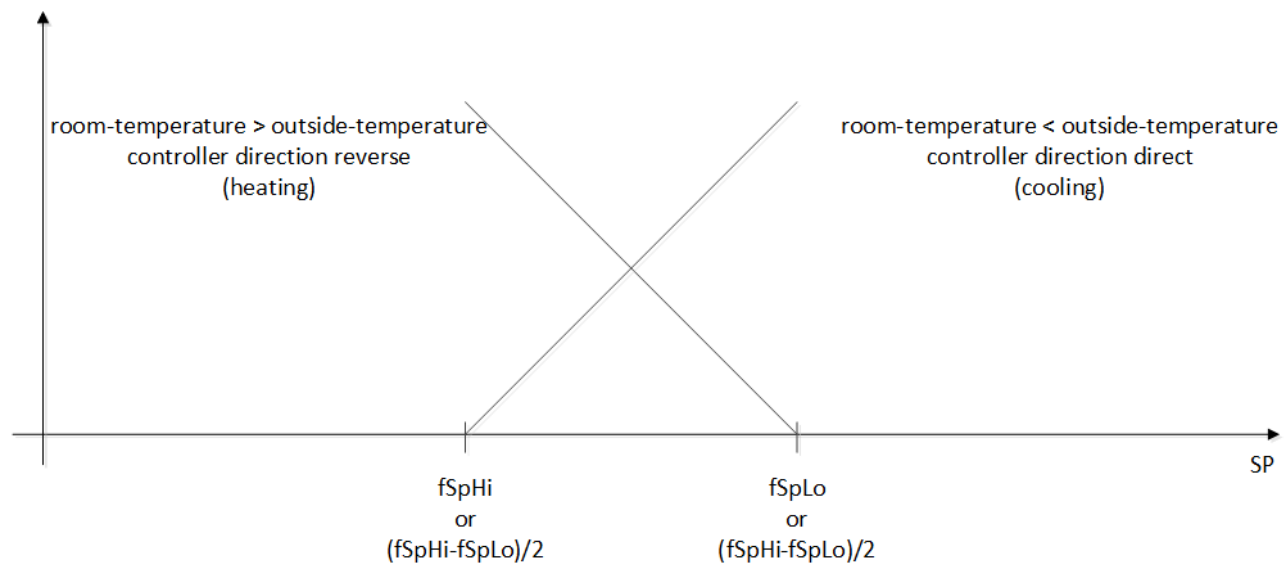
The method is defined by the input variable $bSlcnSpRcv$:

$bSlcnSpRcv$	$fSpRcv$
TRUE	Average of $fSpLo$ and $fSpHi$
FALSE	Depends on control direction, defined through input $bActRcv$

If the setpoint is defined depending on the control direction, the following applies:

$bActRcv$	Control direction	$fSpRcv$
TRUE	direct (cooling)	$fSpHi$
FALSE	indirect (heating)	$fSpLo$

Heat recovery



Inputs

```
VAR_INPUT
    bEn      : BOOL;
    fSpHi    : REAL;
    fSpLo    : REAL;
    fSpMax   : REAL;
    fSpMin   : REAL;
    bActnRcv : BOOL;
    bSlcnSpRcv : BOOL;
END_VAR
```

Name	Type	Description
bEn	BOOL	Function block enable. If bEn = FALSE, all output parameters are 0.0.
fSpHi	REAL	Upper setpoint input value to be checked.
fSpLo	REAL	Lower setpoint input value to be checked.
fSpMax	REAL	Maximum setpoint
fSpMin	REAL	Minimum setpoint.
bActnRcv	BOOL	Direction of action of the downstream heat recovery.
bSlcnSpRcv	BOOL	Setpoint selection of the downstream heat recovery system.

Outputs

```
VAR_OUTPUT
    fPrSpHi   : REAL;
    fPrSpLo   : REAL;
    fSpRcv    : REAL;
    bErr      : BOOL;
    sErrDescr : T_MAXSTRING;
END_VAR
```

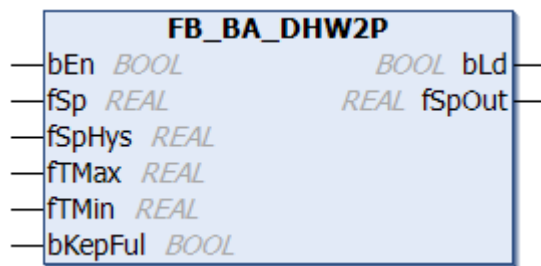
Name	Type	Description
fPrSpHi	REAL	Output value for the upper setpoint.
fPrSpLo	REAL	Output value for the lower setpoint.
fSpRcv	REAL	Output value for the resulting heat recovery setpoint.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Warning: The setpoints are not in a logical order: Either ($fSpMin \geq fSpMax$) OR ($fSpHi < fSpMin$) OR ($fSpLo > fSpMax$)

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.2 Provision of hot water**6.1.2.2.3.1.2.1 FB_BA_DHW2P**

This function block FB_BA_DHW2P controls the charging (heating) of a hot water tank via an on-off controller. Tank heating is activated at input *bEn*. If tank heating is active the output *bLd* is TRUE. The variable *fSp* is used to transfer the setpoint for the service water temperature to the function block. A minimum selection is connected to the input *fTMin*, a maximum selection of all temperature sensors of the hot water tank is connected to the input *fTMax*.

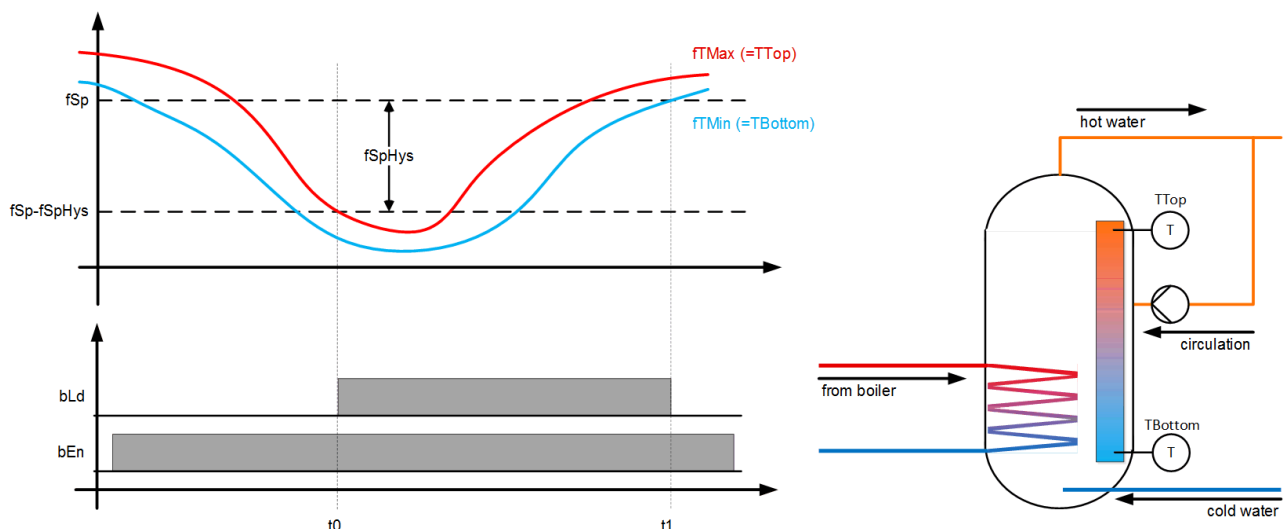
Due to the temperature stratification in the hot water tank, the top sensor is generally the one with the highest temperature and the bottom one with the lowest.

The tank can be charged in two ways via the variables *bKepFul*:

bKepFul = FALSE

Charging is requested if *fTMax* falls below the value of *fSp-fSpHys*. The charge request is disabled if *fTMin* is above the setpoint for *fSp*.

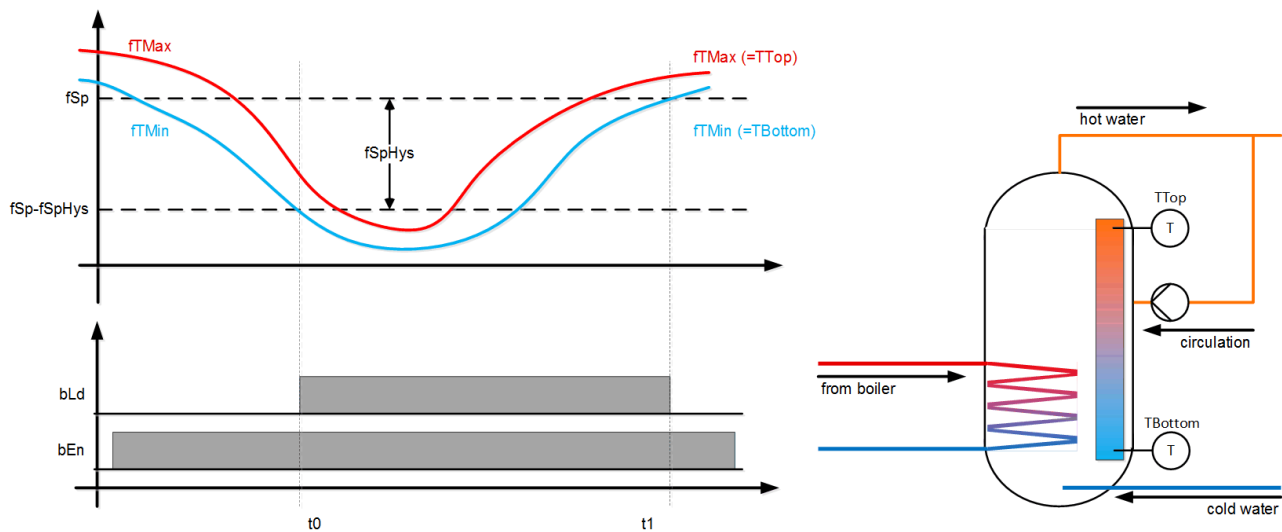
Due to the fact that the sensor at the top generally measures the highest temperature, the heating is not switched on until the hot water tank has been discharged.



bKepFul = TRUE

Charging is requested if $fTMin$ falls below the value of $fSp - fSpHys$. The charge request is disabled once $fTMin$ is above the setpoint again.

Selecting the minimum of all tank temperatures ensures that the coldest point of the tank is used for control purposes. Recharging takes place when the tank is no longer full.

**Inputs**

```
VAR_INPUT
  bEn      : BOOL;
  fSp      : REAL;
  fSpHys   : REAL;
  fTMax    : REAL;
  fTMin    : REAL;
  bKepFul  : BOOL;
END_VAR
```

Name	Type	Description
bEn	BOOL	Enable boiler charging
fSp	REAL	Service water temperature setpoint [°C].
fSpHys	REAL	Hysteresis, recommended 1°K to 5°K.
fTMax	REAL	Maximum selection of all tank temperatures [°C].
fTMin	REAL	Minimum selection of all tank temperatures [°C].
bKepFul	BOOL	Control temperature selection: FALSE = $fTMax$ is used to request bLd , $fTMin$ to switch off. TRUE = $fTMin$ alone controls the switching on/off of bLd .

Outputs

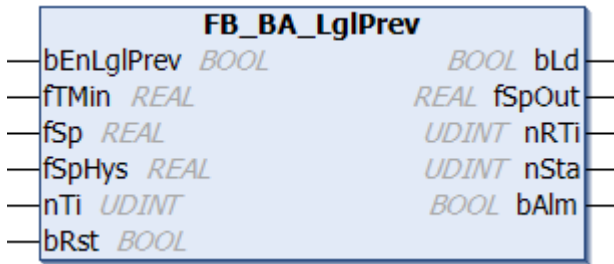
```
VAR_OUTPUT
  bLd      : BOOL;
  fSpOut   : REAL;
END_VAR
```

Name	Type	Description
bLd	BOOL	Enable charging mode.
fSpOut	REAL	Setpoint forwarding to charging circuit: $fSpOut = fSp$ (input) if the function block is enabled. $fSpOut = 0$ if the function block is not enabled.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.2.2 FB_BA_LglPrev

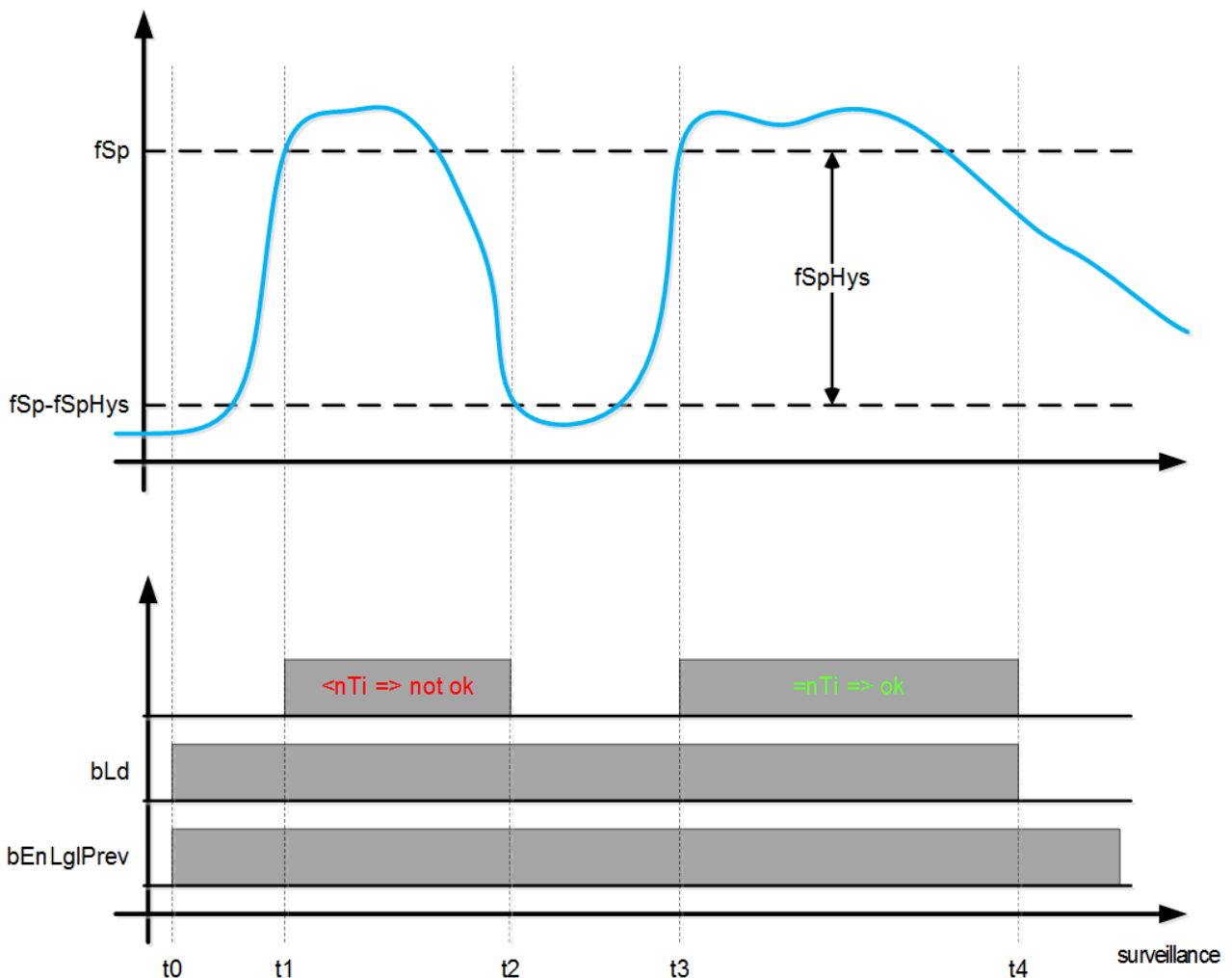


This function block FB_BA_LglPrev is used for disinfection of the service water and for killing off Legionella. Disinfection mode is activated at input *bEnLglPrev* via a timer program. It is advisable to run the disinfection at least once per week (during the night). The temperature should be at least 70 °C. The activation interval at *bEnLglPrev* must be adequately long. The output *bLd* activates the storage tank charging.

For hot water tanks with several temperature sensors, a min selection of all sensors must be connected to *fTMin*.

If *fTMin* exceeds the value of *fSp*, a monitoring timer with a time of *nTi_sec* [s] is started. If the minimum tank temperature *fTMin* remains above *fSp* - *fSpHys* while the timer is active, the tank was heated adequately. If circulation is active, the output *bLd* must be linked to enabling of the circulation pump, to ensure that the water pipe within the service water system is included in the disinfection. If the temperature has fallen below *fSp* - *fSpHys* during the disinfection process, this process must be restarted and run until the time *nTi* has fully elapsed. If disinfection is successful, output *bLd* is reset.

If the disinfection process was incomplete during the function block activation (*bEnLglPrev*), this is indicated with the output *bAlm*. The output must be reset with *bRst*.



Explanation of the diagram:

t0 Start of the legionella program and switching of output *bLd*. Heating of the hot water tank.

t1 The tank has reached the temperature fSp . The timer for the heating time is started.

t2 The minimum tank temperature has fallen below $fSp - fSpHys$. The timer for the heating time is reset.

t3 The temperature exceeds fSp again, and the heating timer is started again.

t4 The minimum tank temperature was above the limit $fSp - fSpHys$ over the period nTi ; the disinfection was successful. bLd is reset, and the hot water tank switches back to normal operation.

Inputs

```
VAR_INPUT
  bEnLglPrev : BOOL;
  fTMin      : REAL;
  fSp        : REAL;
  fSpHys     : REAL;
  nTi        : UDINT;
  bRst       : BOOL;
END_VAR
```

Name	Type	Description
bEnLglPrev	BOOL	Enabling of disinfection operation via a timer program.
fTMin	REAL	Minimum tank temperature [°C]. Minimum selection of temperature sensors at the top and bottom.
fSp	REAL	Setpoint for disinfection [°C]
fSpHys	REAL	Temperature difference [°K] lower limit; always calculated absolute.
nTi	UDINT	Monitoring period [s].
bRst	BOOL	Resetting of the legionella alarm.

Outputs

```
VAR_OUTPUT
  bLd      : BOOL;
  fSpOut   : REAL;
  nRTi     : UDINT;
  nSta     : UDINT;
  bAlm     : BOOL;
END_VAR
```

Name	Type	Description
bLd	BOOL	Anti-legionella mode active.
fSpOut	REAL	Setpoint forwarding to charging circuit: fSp (input) if the function block is activated 0 if the function block is not activated
nRTi	UDINT	Countdown timer disinfection mode.
nSta	UDINT	Disinfection program status: 1. Disinfection operation was successful. 2. Disinfection completed successfully. After the disinfection, and to reactivate legionella prevention, $bEnLglPrev$ must be FALSE. 3. Disinfection operation active. 4. Disinfection not successfully. Alarm is pending. 5. Disinfection not successfully, the alarm was acknowledged. 6. Restart of the controller or no legionella mode has been requested yet.
bAlm	BOOL	The temperature setpoint was not reached consistently over the interval nTi , so that adequate disinfection is not guaranteed.

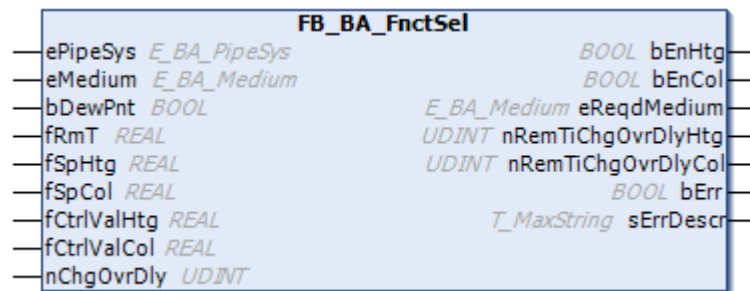
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3 Room automation

6.1.2.2.3.1.3.1 Heating, cooling

6.1.2.2.3.1.3.1.1 FB_BA_FnctSel



The function block FB_BA_FnctSel is used to enable heating or cooling operation in a room.

The type of distribution network plays a major role here:

If it is a two-pipe system, all the rooms in the system can only ever be either heated or cooled.

In a four-pipe system, on the other hand, the air conditioning of the rooms can be based on demand, i.e. one part of the rooms can be heated and the other cooled by the same system.

The function block used for each room, as already mentioned, selects its controllers, depending on which type of piping system is available:

Two-pipe network

The two-pipe system is selected when `ePipeSys.e2Pipe` is set at the input of the function block. Since all rooms served by the plant can only either be heated or cooled, the choice is specified centrally for all rooms via the input `eMedium`. If `eMedium` is FALSE, the room heating controller is selected. If the input is TRUE the cooling controller is selected. The controller enable states `bEnHtg` and `bEnCol` are always issued with a delay of `nChgOvrDly` [s]. In other words, heating cannot be enabled until the cooling enable state `bEnCol` for `nChgOvrDly` is FALSE, and vice versa. In addition to the elapsing of this changeover time, the system checks that the output from controller to be switched off is 0.0. This is done by feedback at the inputs `fCtrlValHtg` and `fCtrlValCol`. In this way, a drastic change from heating to cooling and vice versa is avoided.

Four-pipe network

The four-pipe system is selected when `ePipeSys.e4Pipe` is set at the input of the function block. In this case, the choice of controller can be different for the individual rooms as required, based on the room temperature `fRmT` and the setpoints `fSpHtg` for heating and `fSpCol` for cooling. If the room temperature exceeds the cooling setpoint, the cooling controller is enabled (`bEnCol`), if it falls below the heating setpoint, the heating controller is enabled (`bEnHtg`). If the temperature is between the two setpoints, both controllers are switched off (energy-neutral zone). Here too, the output of the controller enable states `bEnHtg` and `bEnCol` is delayed by `nChgOvrDly` [s] (see two-pipe network). In addition to the elapsing of this changeover time, the system checks that the output from controller to be switched off is 0.0. This is done by feedback at the inputs `fCtrlValHtg` and `fCtrlValCol`. In this way, a drastic change from heating to cooling and vice versa is avoided, if the changeover time is inadequate.

Dew point sensor (bDewPnt)

In both systems (two- and four-pipe) the dew point sensor has the task of deactivating cooling immediately, if required.

Program sequence

The function block can have 3 possible states:

1. Waiting for heating or cooling enable
2. Heating enable
3. Cooling enable

In the first step, the function block waits for compliance with the conditions required for heating or cooling:

Heating	Cooling
Output cooling controller = 0 ($fCtrlValCol$)	Output heating controller = 0 ($fCtrlValHtg$)
Room temperature ($fRmT$) < setpoint heating ($fSpHtg$)	Room temperature ($fRmT$) > setpoint cooling ($fSpCol$)
Cooling controller enable ($bEnCol$) is FALSE over at least the changeover time $nChgOvrDly$ [s]	Heating controller enable ($bEnHtg$) is FALSE over at least the changeover time $nChgOvrDly$ [s]
Four-pipe system is selected ($ePipeSys = E\_BA\_PipeSys.4Pipe$) or two-pipe system is selected and heating medium is available ($ePipeSys = E\_BA\_PipeSys.2Pipe$ AND $bMedium = FALSE$)	Four-pipe system is selected ($ePipeSys = E\_BA\_PipeSys.e4Pipe$) or two-pipe system is selected and cooling medium is available ($ePipeSys = E\_BA\_PipeSys.e2Pipe$ AND $bMedium = TRUE$)
	The dew point sensor does not trigger ($bDewPnt = TRUE$)

If a chain of conditions is met, the function block switches to the respective state (heating or cooling) and remains in this state until the corresponding controller issues 0 at the function block input ($fCtrlValHtg/fCtrlValCol$). This ensures that only one controller is active at any one time, even if a high heating controller output, for example, would call for a brief cooling intervention (overshoot). Heating or cooling continues until there is no longer a demand.

There are 3 exceptions, for which heating or cooling is immediately interrupted:

1. In the two-pipe system ($ePipeSys = E\_BA\_PipeSys.2Pipe$) heating is active ($bEnHtg$), but the system has been switched to cooling medium ($eMedium = E\_BA\_Medium.eCoolMedium$)
2. In the two-pipe system ($ePipeSys = E\_BA\_PipeSys.2Pipe$) cooling is active ($bEnCol$), but the system has been switched to heating medium ($eMedium = E\_BA\_Medium.eHeatMedium$)
3. The dew point sensor was triggered ($bDewPnt=TRUE$) in cooling mode (two or four-pipe system)

In these cases the heating or cooling enable states are canceled, and the plant switches to standby.

Note

If one of the two controller is enabled and the corresponding controller does not react, i.e. it remains at "0" for the time $nChgOvrDly$ [s], the function block automatically returns to the first step "Wait for heating or cooling enable".

This is an emergency function in case a temperature sensor jumps and a wrong selection is made, which then cannot be fulfilled by the selected controller. An example would be if at PLC start a sensor function outputs 0°, thus heating is selected and the sensor function then assumes a temperature value that requires the cooling controller due to a programmed PLC start delay. Without this emergency function, it would be waited in vain for the heating controller to assume a value greater than "0".

Demand message ($eReqdMedium$)

To notify the plant of the current demand for heating or cooling, a demand ID is issued at the function block output, i.e. for each room, depending on the actual and set temperature. These can be collected and evaluated centrally. The evaluation always takes place, irrespective of the network type (two- or four-pipe).

$eReqdMedium$	Medium	Room temperature
1	No medium is requested	$fRmT > fSpHtg$ AND $fRmT < fSpCol$
2	Heating medium is requested	$fRmT < fSpHtg$
3	Cooling medium is requested	$fRmT > fSpCol$

Error handling

The heating setpoint must not be greater than or equal to the cooling setpoint, since this would result in temperature range with simultaneous heating and cooling demand. However, since the function block only issues one enable state at a time (i.e. heating or cooling), the case is harmless from a plant engineering perspective. In this case only a warning message is issued (*bErr* = TRUE, *sErrDescr* = warning message); the function block does not interrupt its cycle.

Inputs

```
VAR_INPUT
  ePipeSys      : E_BA_PipeSys;
  eMedium       : E_BA_Medium;
  bDewPnt       : BOOL;
  fRmT          : REAL;
  fSpHtg        : REAL;
  fSpCol        : REAL;
  fCtrlValHtg   : REAL;
  fCtrlValCol   : REAL;
  nChgOvrDel    : UDINT;
END_VAR
```

Name	Type	Description
eParameter	E_BA_PipeSys	Pipe system (2Pipe, 4Pipe) of the plant.
eMedium	E_BA_Medium	Selection of the medium for the entire two-pipe network (NoMedium, HeatMedium, CoolMedium).
bDewPnt	BOOL	Dew point sensor: if <i>bDewPnt</i> = FALSE, then the cooling controller is locked.
fRmT	REAL	Room temperature
fSpHtg	REAL	Heating setpoint
fSpCol	REAL	Cooling setpoint
fCtrlValHtg	REAL	Current output value of the heating controller. Used internally as switching criterion from heating to cooling: <i>fCtrlValHtg</i> must be 0.
fCtrlValCol	REAL	Current output value of the cooling controller. Used internally as switching criterion from cooling to heating: <i>fCtrlValCol</i> must be 0.
nChgOvrDel	UDINT	Switchover delay [s] from heating to cooling or vice versa. Internally limited to a minimum value of 0.

Outputs

```
VAR_OUTPUT
  bEnHtg        : BOOL;
  bEnCol        : BOOL;
  eReqdMedium   : E_BA_Medium;
  nRemTiChgOvrDlyHtg : UDINT;
  nRemTiChgOvrDlyCol : UDINT;
  bErr          : BOOL;
  sErrDescr     : T_MAXSTRING;
END_VAR
```

Name	Type	Description
bEnHtg	BOOL	Heating controller enable.
bEnCol	BOOL	Cooling controller enable.
eReqdMedium	E_BA_Medium	Requested medium (see Determination of needs).
nRemTiChgOvrDlyHtg	UDINT	Countdown [s] for switchover delay from cooling to heating.
nRemTiChgOvrDlyCol	UDINT	Countdown [s] for switchover delay from heating to cooling.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description
01: Warning: The setpoint Heating is higher than or equal to the setpoint Cooling

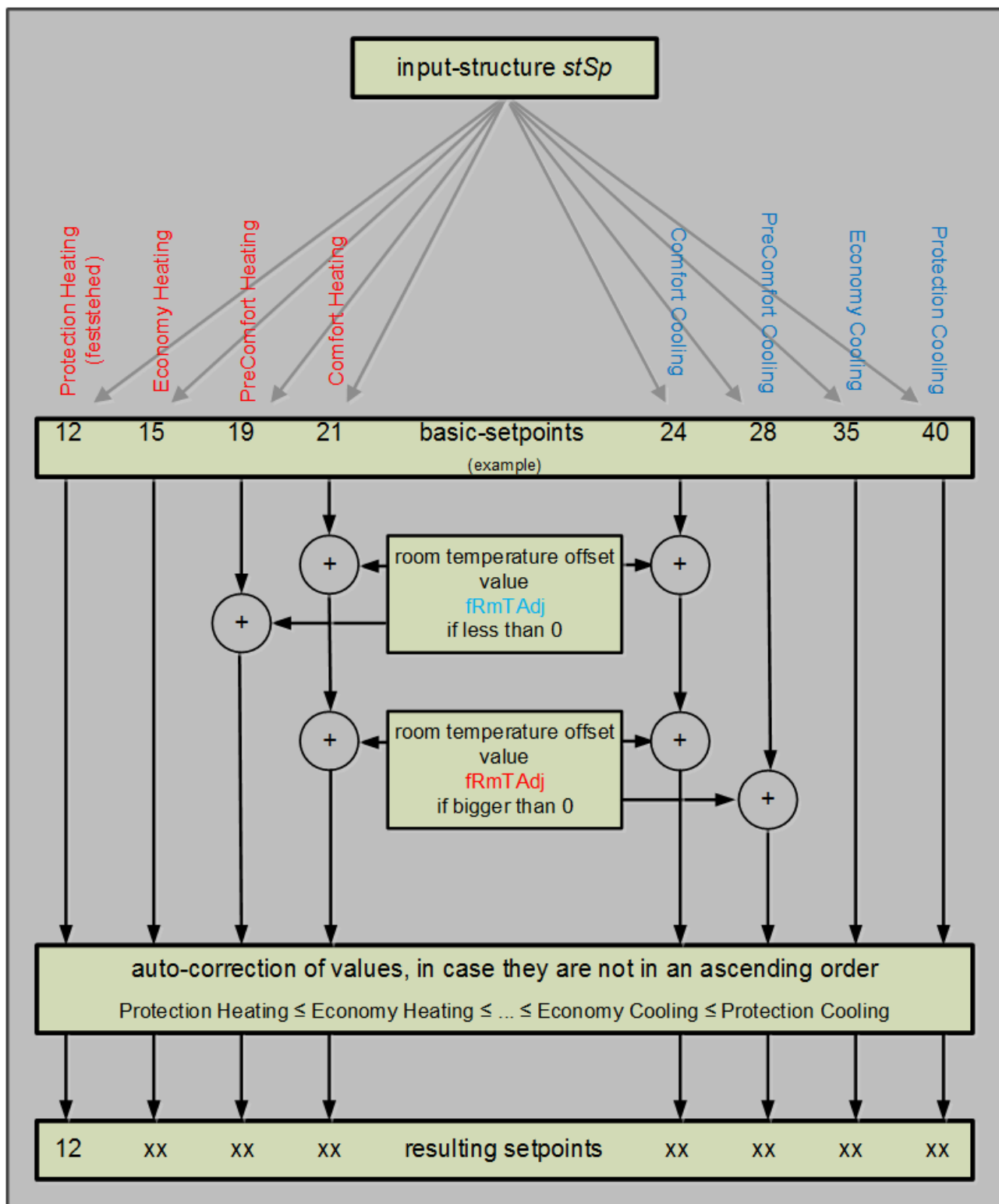
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.1.2 FB_BA_RmTAdj



The function block FB_BA_RmTAdj is used for user adjustment of the room temperature setpoint. It shifts the setpoints at the input of a function block depending on an offset *rRmTAdj*, as shown in the following diagram. At the input *fRmTAdj*, the value of a resistance potentiometer or a bus-compatible field device, for example, can be used for the setpoint correction.



If the set value *fRmTAdj* is greater than zero, room temperature heating is desired: The Comfort Heating value is raised by the value *fRmTAdj*. At the same time, the values for Comfort Cooling and PreComfort Cooling are increased. If the value *rRmTAdj* is less than zero, a lower room temperature is requested. Analogous to the heating case, the values for Comfort Cooling, Comfort Heating and PreComfort Heating are now reduced by the value *rRmTAdj*.

Auto-correction

The temperature adjustment is intended for small corrections of the values. Although it is possible to enter any input values, a heating system will only work in a meaningful manner if the setpoints have ascending values in the following order:

- Protection Heating
- Economy Heating
- Precomfort Heating
- Comfort Heating
- Comfort Cooling
- Precomfort Cooling
- Economy Cooling
- Protection Cooling

Auto-correction works according to the following principle: Starting with the value Economy Heating, the system checks whether this value is smaller than the lower value of Protection Heating. If this is the case, the value for Economy Heating is adjusted to match the value for Protection Heating. The system then checks whether the value for Precomfort Heating is less than Economy Heating and so on, until the value for Protection Cooling is compared with the value for Economy Cooling. If one or several values were corrected, this is indicated with a TRUE signal at output *bValCorr*.

Inputs

```
VAR_INPUT
  fRmTAdj    : REAL;
  stSp       : ST_BA_SpRmT;
END_VAR
```

Name	Type	Description
fRmTAdj	REAL	Room temperature offset value
stSp	ST_BA_SpRmT [► 215]	Input structure for the setpoints

Outputs

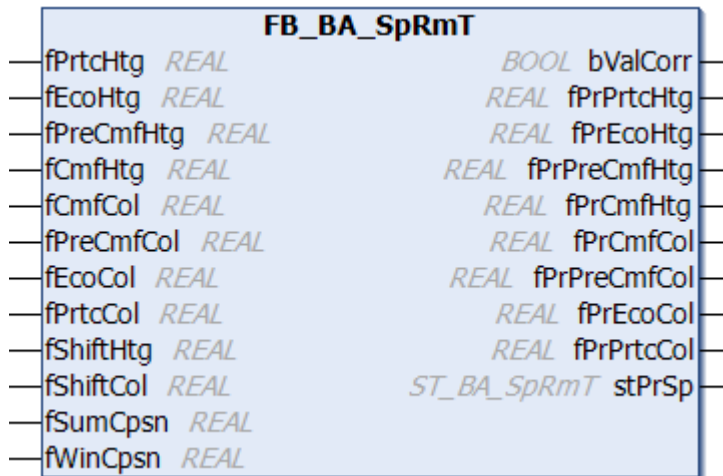
```
VAR_OUTPUT
  bValCorr    : BOOL;
  fPrPrtcHtg  : REAL;
  fPrEcoHtg   : REAL;
  fPrPreCmfHtg : REAL;
  fPrCmfHtg   : REAL;
  fPrPrtcCol  : REAL;
  fPrEcoCol   : REAL;
  fPrPreCmfCol : REAL;
  fPrCmfCol   : REAL;
  stPrSp      : ST_BA_SpRmT;
END_VAR
```

Name	Type	Description
bValCorr	BOOL	Autocorrection for the values was performed, see above.
rPrPrtcHtg	REAL	Resulting Protection Heating setpoint
rPrEcoHtg	REAL	Resulting Economy Heating setpoint.
rPrPreCmfHtg	REAL	Resulting PreComfort Heating setpoint.
rPrCmfCol	REAL	Resulting Comfort Cooling setpoint.
rPrPreCmfCol	REAL	Resulting PreComfort Cooling setpoint.
rPrEcoCol	REAL	Resulting Economy Cooling setpoint.
rPrPrtcCol	REAL	Resulting Protection Cooling setpoint
stPrSp	ST_BA_SpRmT [► 215]	Consolidated output of the resulting values in a structure.

Requirements

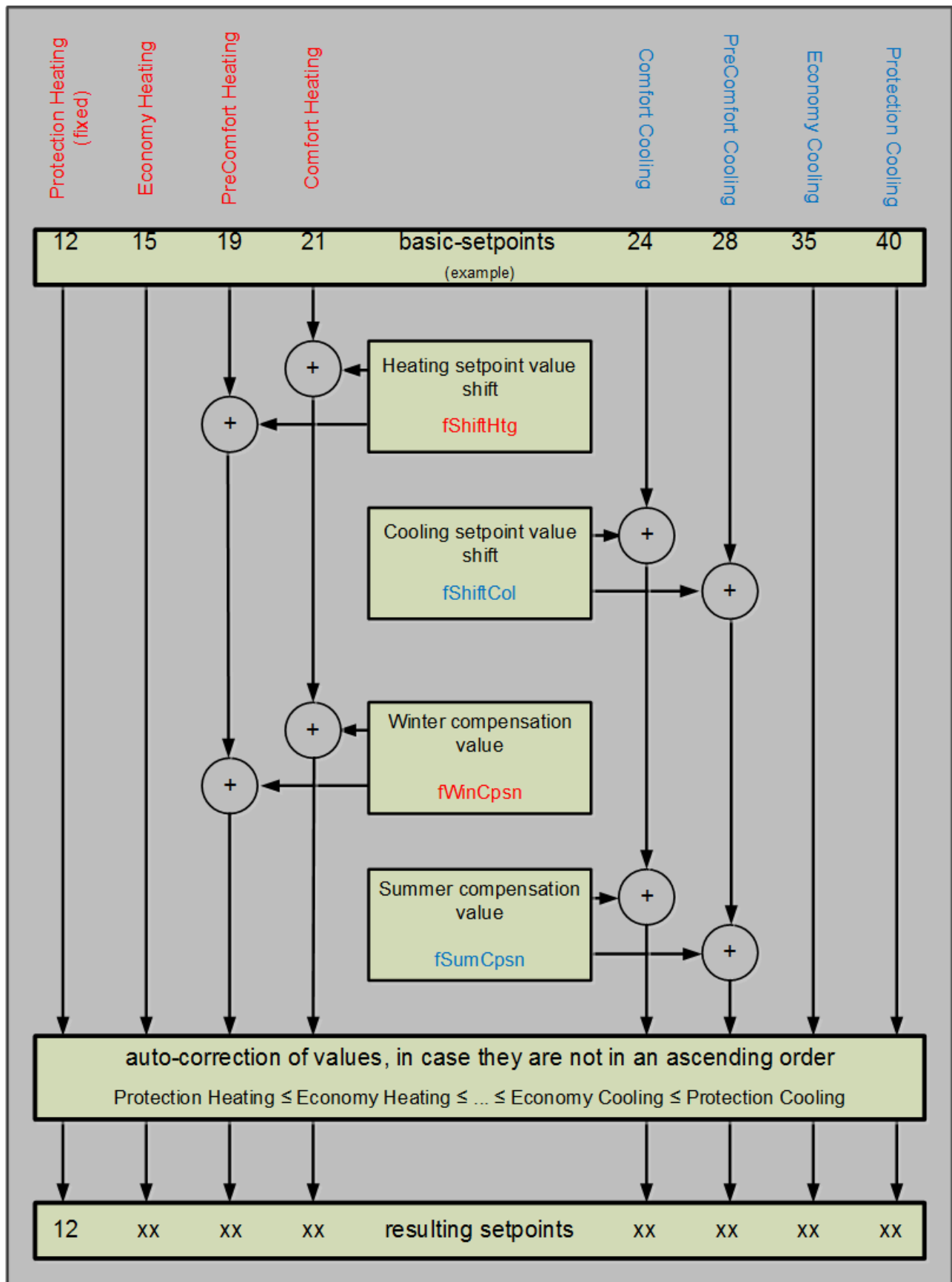
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.1.3 FB_BA_SpRmT



The function block FB_BA_SpRmT assigns setpoints for cooling and heating operation to each of the energy levels Protection, Economy, PreComfort and Comfort.

The following graphics illustrates the behavior of the function block; the entered values should be regarded as examples:



The parameter *fShiftHtg* is applied to the Comfort and Precomfort values for the heating mode as central setpoint shift. In addition, winter compensation *fWinCpsn* is applied. Similarly, the following applies for the cooling mode: The parameter *fShiftCol* is applied to the Comfort and Precomfort values. In addition, the summer compensation value *fSumCpsn* is applied.

Auto-correction

The temperature adjustment is intended for small corrections of the values. Although it is possible to enter any input values, a heating system will only work in a meaningful manner if the setpoints have ascending values in the following order:

- Protection Heating
- Economy Heating
- Precomfort Heating
- Comfort Heating
- Comfort Cooling
- Precomfort Cooling
- Economy Cooling
- Protection Cooling

Auto-correction works according to the following principle: Starting with the value Economy Heating, the system checks whether this value is smaller than the lower value of Protection Heating. If this is the case, the value for Economy Heating is adjusted to match the value for Protection Heating. The system then checks whether the value for Precomfort Heating is less than Economy Heating and so on, until the value for Protection Cooling is compared with the value for Economy Cooling. If one or several values were corrected, this is indicated with a TRUE signal at output *bValCorr*.

Inputs

```
VAR_INPUT
  fPrtcHtg    : REAL;
  fEcoHtg     : REAL;
  fPreCmfHtg  : REAL;
  fCmfHtg     : REAL;
  fCmfCol     : REAL;
  fPreCmfCol  : REAL;
  fEcoCol     : REAL;
  fPrtcCol    : REAL;
  fShiftHtg   : REAL;
  fShiftCol   : REAL;
  fSumCpsn    : REAL;
  fWrWinCpsn  : REAL;
END_VAR
```

Name	Type	Description
fPrtcHtg	REAL	Basic Protection Heating setpoint
fEcoHtg	REAL	Basic Economy Heating setpoint
fPreCmfHtg	REAL	Basic PreComfort Heating setpoint
fCmfHtg	REAL	Basic Comfort Heating setpoint
fCmfCol	REAL	Basic Comfort Cooling setpoint
fPreCmfCol	REAL	Basic PreComfort Cooling setpoint
fEcoCo	REAL	Basic Economy Cooling setpoint
fPrtcCol	REAL	Basic Protection Cooling setpoint
fShiftHtg	REAL	Heating setpoint value shift
fShiftCol	REAL	Cooling setpoint value shift
fSumCpsn	REAL	Summer compensation value
fWinCpsn	REAL	Winter compensation value

Outputs

```
VAR_OUTPUT
  bValCorr    : BOOL;
  fPrPrtcHtg  : REAL;
  fPrEcoHtg   : REAL;
  fPrPreCmfHtg : REAL;
  fPrCmfHtg   : REAL;
  fPrCmfCol   : REAL;
```

```

fPrPreCmfCol : REAL;
fPrEcoCol    : REAL;
fPrPrtcCol   : REAL;
stPrSp       : ST_BA_SpRmT;
END_VAR

```

Name	Type	Description
bValCorr	BOOL	Autocorrection for the values was performed, see above.
rPrPrtcHtg	REAL	Resulting Protection Heating setpoint
rPrEcoHtg	REAL	Resulting Economy Heating setpoint.
rPrPreCmfHtg	REAL	Resulting PreComfort Heating setpoint.
rPrCmfCol	REAL	Resulting Comfort Cooling setpoint.
rPrPreCmfCol	REAL	Resulting PreComfort Cooling setpoint.
rPrEcoCol	REAL	Resulting Economy Cooling setpoint.
rPrPrtcCol	REAL	Resulting Protection Cooling setpoint
stPrSp	ST_BA_SpRmT [► 215]	Consolidated output of the resulting values in a structure.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.2 Lighting

6.1.2.2.3.1.3.2.1 FB_BA_ConstLgtControlEx



The function block FB_BA_ConstLgtControlEx represents a constant light control with manual override.

It can be in the following operating states as described in VDI 3813 sheet 2:

- Disabled: Outputs *bControlMode* = FALSE and *bManualMode* = FALSE

- Control mode: Outputs *bControlMode* = TRUE and *bManualMode* = FALSE
- Manual override: Outputs *bControlMode* = FALSE and *bManualMode* = TRUE

In the active states (control mode, manual override), the light output value *fOut* can assume any values from 0...100 %, in the non-active state it is fixed at 0 %.

The distinction between "active" and "not active" is important when several light control functions access actuators in alternation.

This makes the function block suitable for interaction with other lighting functions in a priority selection in which the output *fOut* is regarded as a valid control value depending on the operating status outputs.

The light control function works with an input for the brightness state of a reference lamp due to the possible interaction with other light control functions. If several lighting functions control the same lamp or the same lighting group independently of each other by priority control, the function block does not know whether it is the active one. With the help of the reference input, however, it is always possible to synchronize to the current value before each switching action and to assess before a toggle action whether the light should be switched on or off in the next step.

The function block operates with an input *fRefLgtVal* for the brightness state of a reference lamp. This is important when several lighting functions control the same lamp or lighting group independently of each other, for example by priority control. In this case, the function block does not know if it is the active one. However, with the help of the reference input *fRefLgtVal* it is always possible to synchronize to the current light value before each switching action and to assess before a toggle action whether the light should be switched on or off in the next step.

If it is ensured that the function is the only one that controls the light, the input *fRefLgtVal* is not to be assigned. It is pre-initialized with the value -1, which indicates to the function block that the input is not linked.

However, this also means that it is not possible to delete an input link online without distorting the functioning of the function block!

Function

From the non-activated state, this function block operates in two steps: Short button presses at *bSwi*, *bSwiUp* or *bSwiDown* as well as rising edges at *bOn* or *bPrc* initially switch the function to control mode. However, the presence signal input *bPrc* is only active in full automatic mode.

The operation mode fully automatic or semi-automatic can be parameterized at the input (*eMode*).

Switch-on behavior

The light control is first switched to the value *fOnValCtrl*.

The switching of the light takes place with a ramp, which is specified at *nSwiTi* in seconds related to 0...100%.

If the selected light value is reached, the control remains at this value for the time *nBrtnsAdjTi* [s] so that the light sensor detects the correct value.

Already during this waiting time, the function can be switched to manual override.

Control behavior

The control now attempts to maintain this setpoint: If the daylight incidence increases, the light control value at the output *fOut* is reduced. If the total brightness becomes weaker, the light control value is increased. The change also follows a ramp, which is specified at *nRampTi* in seconds related to 0...100%. The constant light control is a simple I-controller.

To prevent the ramp block from constantly specifying new light values, a hysteresis range can be specified at the *fHys* input.

The constant light regulation controls the output up or down until the measured brightness reaches or slightly exceeds or falls below the target value. The light is then considered as adjusted. Only when the measured brightness exceeds or falls below the setpoint by *fHys*/2, it is readjusted again. The range and unit of the hysteresis depend on the light sensor.

Minimum value of the control

Changes in the lower brightness range are often perceived as annoying.

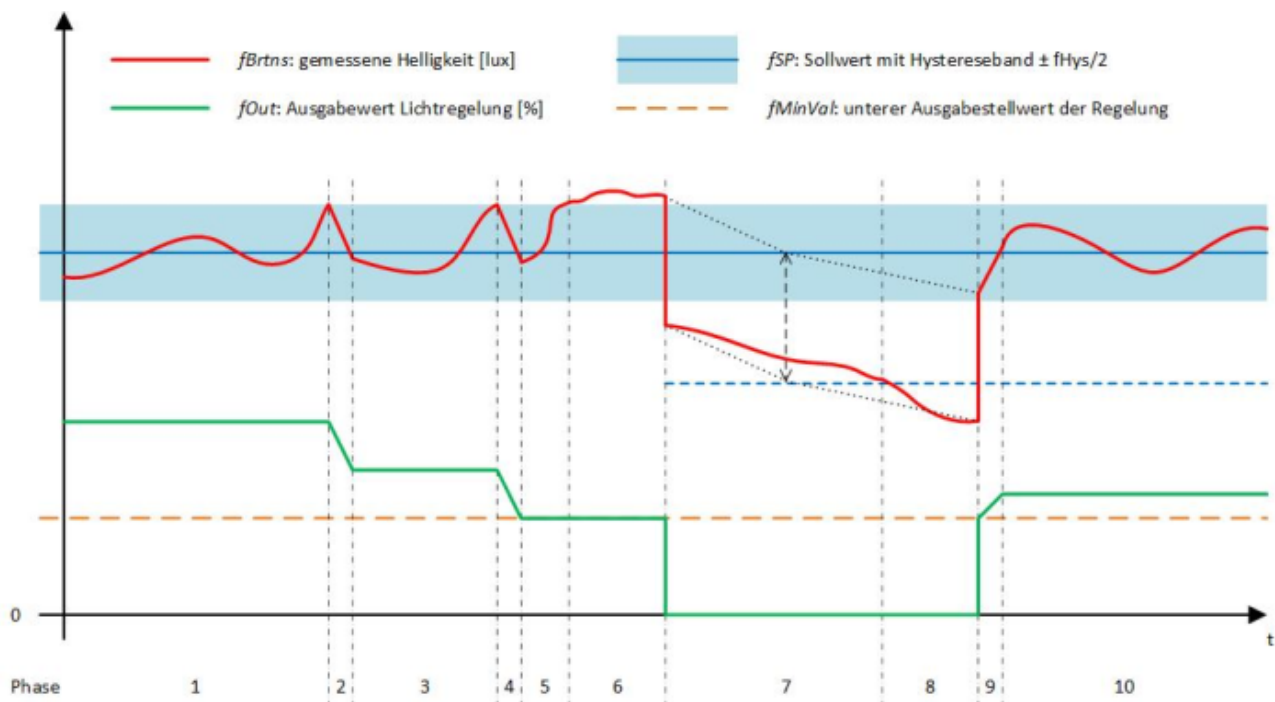
It can therefore be useful to limit the light control to a minimum control value and then switch off the light completely when the outdoor brightness is very high.

In this case, when the control has reached its minimum control value $fMinVal$, instead of reducing further it remains at the minimum value for the time $nOffDly$ [s] and then switches to the output value 0.

The measured brightness value is stored before and after switching. The difference that results is the brightness increase by switching on to the minimum value.

The new threshold for switching on again is now the control setpoint minus the brightness increase. If the measured brightness reaches or falls below this threshold value for the time $nOnDly$ [s], the light is switched on again and then controlled to the setpoint.

The following diagram is intended to illustrate the choice of threshold value:



1. The measured brightness $fBrtns$ initially moves within the tolerance limits around the setpoint fSP . Towards the end of this phase, the brightness exceeds the tolerance range.
2. The light control dims the light (control value $fOut$) until the light setpoint is reached or the light level falls just below in the PLC cycle. The measured light value $fBrtns$ can therefore be slightly below the setpoint, but this is accepted in order to avoid continuous readjustment.
3. The measured brightness $fBrtns$ is again within the tolerance limits, as in phase 1.
4. It is controlled again, this time to the minimum output control value, which brings the total measured brightness back into the tolerance range.
5. The measured brightness increases again due to outdoor brightness and exceeds the tolerance range.
6. The measured brightness $fBrtns$ remains above the tolerance range, the internal time $nOffDly$ [s] expires.
7. The light is switched off ($fOut = 0$). The difference between the measured brightness before and after switching off is recorded internally as a brightness gain by switching on the light to the minimum control value. The new threshold that must now be undershot is the control setpoint minus the brightness gain. The choice of this threshold is based on the following considerations:
 - Switching the light on again must not result in the measured brightness subsequently being above the tolerance range again, as this could result in the light continuously being switched on and off.

- This threshold is realistic, i.e. > 0 : at the start of phase 5, the outdoor brightness was already so high that the control was set to the minimum value. Before the light is switched off in phase 7, more outdoor brightness was added. In the diagram it is clear that the new threshold is reached when the increase in outdoor brightness from the start of phase 5 to the end of phase 6 has passed again – a realistic scene. To be on the safe side, however, an internal check is made not only for falling below the threshold value but also for reaching it (less than or equal to)
- 8. The threshold value is undershot for the time $nOnDly$ [s].
- 9. The light is again first switched to the minimum value and then dimmed up until the measured brightness corresponds to the setpoint value fSP .
- 10. The measured brightness $fBrtns$ is again within the tolerance limits around the setpoint fSP .

Manual override

From the control mode the following actions let the function change to the manual mode (outputs $bControlMode = FALSE$ and $bManualMode = TRUE$):

- Short button presses at the inputs $bSwi$, $bSwiUp$ and $bSwiDwn$ switch the light on and off alternately.
- With bOn and $bOff$, targeted switching on or off occurs.
- A long button press at the input $bSwi$ alternately dims the light up and down. Dimming takes place via a ramp, which is specified at $nDimTi$ in seconds related to 0...100%.
- A long button press on the input $bSwiUp$ dims the light up specifically. Dimming takes place via a ramp, which is specified at $nDimTi$ in seconds related to 0...100%.
- A long button press at the input $bSwiDwn$ dims the light down specifically. Dimming takes place via a ramp, which is specified at $nDimTi$ in seconds related to 0...100%.

"Off" in manual mode means that the overall function is still active!

A positive edge at $bRstManMod$ switches from manual override back to control mode.

Memory mode

If the memory mode is activated at the input $eMode$, the value that the function block had before the last manual switch-off is stored internally. This value is then adopted the next time the system is switched on in manual mode.

If Memory-mode is deactivated, the switch-on always takes place with the value $fOnValMan$.

Deactivating the function

From control mode or manual mode, two events cause the function to change to the deactivated state (outputs $bControlMode=FALSE$ and $bManualMode=FALSE$):

- a falling edge at the input $bPrc$.
- a TRUE signal at input $bRst$. This input is intended for central deactivation of the function block or in case there is no occupancy sensor.

On deactivation, a ramp $nPreOffRampTi$ (in seconds related to 100% to 0%) is used to dim down to a base value $fPreOffVal$. This value represents the time $nPreOffDly$ in seconds at the output before switching off. The function block is then deactivated.



If the light output value before deactivation is already smaller than the mentioned base value, the switch-off routine is skipped and switched off immediately.

Inputs

```
VAR_INPUT
  bSwi          : BOOL;
  bSwiUp        : BOOL;
  bSwiDwn       : BOOL;
  bOn           : BOOL;
  bOff          : BOOL;
  fSetValMan    : REAL;
  bSetValMan    : BOOL;
```

```
bSetCtrlMod      : BOOL;
bRstManMod       : BOOL;
bRst             : BOOL;
bPrc             : BOOL;
fBrtns          : REAL;
fSP             : REAL;
fHys            : REAL := 50;
nSwiOvrTi       : UDINT := 250;
nSwiTi          : UDINT := 2;
nDimTi          : UDINT := 5;
nRampTi         : UDINT := 60;
nPreOffRampTi   : UDINT := 10;
nBrtnsAdjTi     : UDINT := 5;
fOnValCtrl      : REAL := 50;
fOnValMan       : REAL := 100;
fPreOffVal      : REAL := 20;
nPreOffDly      : UDINT := 20;
fMinVal         : REAL;
nMinOffDly      : UDINT := 300;
nMinOnDly       : UDINT := 300;
bMemMod         : BOOL;
eLgtActMod      : E_BA_LightActivationMode;
fRefLgtVal      : REAL := -1;
END_VAR
```


Name	Type	Description
bSwi	BOOL	If the constant light control function block is not yet active, a short button press first switches the control on, then further signals cause the function block to switch to the manual mode and the following applies: short button press: on / off, long button press: alternately dimming up or down.
bSwiUp	BOOL	If the constant light control function block is not yet active, a short button press first switches the control on, then further signals cause the function block to switch to the manual function and the following applies: short button press: on / off, long button press: selective dimming up.
bSwiDwn	BOOL	If the constant light control function block is not yet active, a short button press first switches the control on, then further signals cause the function block to switch to the manual function and the following applies: short button press: on / off, long button press: selective dimming down.
bOn	BOOL	If the constant light control function block is not yet active, a TRUE signal first switches on the control, then this input only refers to switching on in manual mode.
bOff	BOOL	Switches the light off, the constant light control function block is still in manual mode. If the constant light control function block was previously in control mode, it is now in manual mode.
fSetValMan	REAL	Manual preset value. The values at input <i>fSetValMan</i> are adopted as the light output value by a rising edge at <i>bSetValMan</i> . When the function block is set, it switches directly to manual mode, regardless of whether it was previously in control mode or in the deactivated state
bSetValMan	BOOL	Adoption of the manual preset value, see <i>fSetValMan</i> .
bSetCtrlMod	BOOL	A rising edge puts the function block directly into control mode, regardless of whether it was previously in the deactivated state or in manual mode.
bRstManMod	BOOL	This input makes the constant light control function block switch back to automatic mode when it is in manual mode.
bRst	BOOL	This input switches off the constant light control function block. The shutdown is done by ramping and dwell time on a base light value, see below: <i>fPreOffVal</i> , <i>nPreOffDly</i> and <i>nPreOffRampT</i> .
bPrc	BOOL	Presence signal input. If the constant light control function block is configured in fully automatic mode, a rising edge can activate the function block via this input if it was not active before. A falling edge deactivates it. In semi-automatic mode, only a falling edge at this input deactivates the constant light control function block. "Activating" here means that the function is switched on in manual mode (outputs <i>bControlMode</i> = TRUE and <i>bManualMode</i> = FALSE). Deactivating means: the function is not active in manual or control mode (outputs <i>bControlMode</i> = FALSE and <i>bManualMode</i> = FALSE)
fBrtns	REAL	Current brightness for constant light automatic: Range and unit depend on the light sensor used.
fSP	REAL	Brightness setpoint for the constant light automatic: An adjustment with <i>fBrtnsSen</i> is aimed at. Range and unit therefore depend on the light sensor used.

Name	Type	Description
fHys	REAL	Constant light automatic: Hysteresis band. The constant light regulation controls the output up or down until the measured brightness reaches or slightly exceeds or falls below the target value. The lighting is then considered as adjusted. Only when the measured brightness exceeds or falls below the setpoint by $fHys/2$, it is readjusted again. Range and unit depend on the used light sensor, see above: <i>fBrtnsSP</i> and <i>fBrtnsSen</i> .
nSwiOvrTi	UDINT	Distinction time [ms] between short and long button press.
nSwiTi	UDINT	Ramp for the switching functions in seconds [s], related to a dimming from 0 to 100%.
nDimTi	UDINT	Ramp for the dimming functions in seconds [s], related to a dimming from 0 to 100%.
nRampTi	UDINT	Control ramp of the constant light automatic in seconds [s], related to a dimming from 0 to 100%.
nPreOffRampTi	UDINT	Ramp used to drive to a base value <i>fPreOffVal</i> before switching off.
nBrtnsAdjTi	UDINT	Waiting time in seconds [s] after switching on the light for the light sensor to detect the correct value.
fOnValCtrl	REAL	Switch-on value for control operation, this should be activated with the function previously switched off.
fOnValMan	REAL	Switch-on value of the manual function, if "Memory mode" is not selected via the parameter <i>eOperationalMode</i> .
fPreOffVal	REAL	Base value and hold time at this value before switching off the overall function. If the current light value already falls below the base value, this function is not active and it is switched off immediately
fPreOffDly	UDINT	Base value and hold time at this value before switching off the overall function. If the current light value already falls below the base value, this function is not active and it is switched off immediately
fMinVal	REAL	Constant light automatic: Minimum output value. If this value has fallen below internally (i.e. it is bright enough that no artificial light is needed), the constant light control switches the light off after <i>nOffDly</i> (in seconds) has elapsed. If the light is switched off and the control detects the need for light above the minimum value again, the control switches the light on again after <i>nOnDly</i> (in seconds) to initially <i>fMinVal</i> .
nMinOffDly	UDINT	Constant light automatic: Switch-off waiting time in seconds [s], see <i>fMinVal</i> .
nMinOnDly	UDINT	Constant light automatic: Switch-on waiting time in seconds [s], see <i>fMinVal</i> .
bMemMod	BOOL	Memory mode: When the light is switched on in manual mode, the light assumes the value that the function had before it was last switched off. If the "Memory-Mode" is not active, the switch-on value is defined via <i>fOnValMan</i> .
eLgtActMod	E_BA_LightActivationMode	Activation mode of the light control function (E_BA_LightActivationMode). - Fully automatic: The light control function is activated by detected presence and deactivated by absence of presence. Button presses can also activate the function. - Semi-automatic: Activation of the light control function only by pressing a button; it is automatically deactivated when not occupied.

Name	Type	Description
fRefLgtVal	REAL	Light value of a single light actuator representing the light zone. This input is used to assess whether in manual mode the next button press switches on or off, or from which value dimming is to take place. For automatic operation this input is important to start with the "correct" value when control is regained. This input is pre-initialized with "-1". In the event of non-assignment, this is recognized and it is assumed that the function block is the only one to be controlled. Thus, the output value <i>fOut</i> is used as the light value to assess the next switching actions.

Outputs

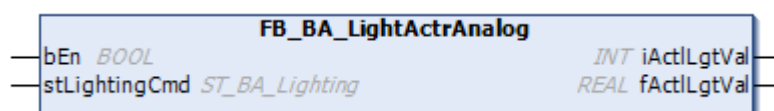
```
VAR_OUTPUT
  fOut      : REAL;
  bControlMode : BOOL;
  bManualMode : BOOL;
  bAdjusting  : BOOL;
  nRemTiMinOff : UDINT;
  nRemTiMinOn  : UDINT;
END_VAR
```

Name	Type	Description
fOut	REAL	Light output value, 0...100%
bManualMode	BOOL	Indicates whether or not the function block is working.
bControlMode	BOOL	Indicates whether the function block is operating in control mode.
bAdjusting	BOOL	The constant light control function block is in regulation mode. This signal can be used to query light sensors, which do not operate in analog mode but via communication, more frequently and thus achieve a more favorable control behavior.
nRemTiMinOff	UDINT	When the control output is at the minimum value, the timer runs down until switching to "0%". This output shows the remaining seconds until switching off.
nRemTiMinOn	UDINT	Due to the constant light regulation, the light is switched off because of sufficient ambient light, but now there is a need for light again and the timer runs down until it is switched on again. This output shows the remaining seconds until switching on.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.2.2 FB_BA_LightActrAnalog



The function block FB_BA_LightActrAnalog is used to control an analog light actuator. The output is in 0...100% and converted accordingly in 0...32767 (positive integer range). The function block is thus suitable for use with the dimmer terminals (e.g. KL2751, KL2761).

Function

If the function block is not enabled ($bEn = FALSE$) the outputs $fActlLgtVal = 0$ and $iActlLgtVal = 0$. In the enabled state the light brightness value of the input telegram ($stLightingCmd.fLgtVal$) is passed directly to the output $fActlLgtVal$. The same value multiplied by 327.65 is given to the output $iActlLgtVal$.

Inputs

```
VAR_INPUT
  bEn          : BOOL;
  stLightingCmd : ST_BA_Lighting;
END_VAR
```

Name	Type	Description
bEn	BOOL	Enabling the function block and activation of the function.
stLightingCmd	ST_BA_Lighting	Light control telegram

Outputs

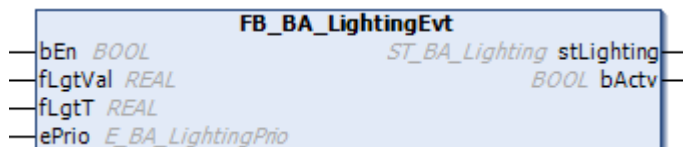
```
VAR_OUTPUT
  iActlLgtVal : INT;
  fActlLgtVal : REAL;;
END_VAR
```

Name	Type	Description
iActlLgtVal	INT	Current light value in positive integer range (0...32767).
fActlLgtVal	REAL	Current light value in percent [%].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.2.3 FB_BA_LightingEvt



The function block FB_BA_LightingEvt is used to set the light value and the color temperature at any event. It can be used, for example, to switch the lights during the night watchman tour or for building cleaning.

If the function block is enabled via the input bEn , the active flag in the light control telegram ($bActv$ in $stLighting$) is set at output $stLighting$. The values entered at the input variables $fLgtVal$ for the light value [%] and $fLgtT$ for the light temperature [K] are passed on in this telegram. If the function is no longer active by resetting bEn , the active flag in the light control telegram $stLighting$ is reset and the values for brightness and color temperature are set to "0". With a telegram selection block (e.g. FB_BA_LightingTgmSel8) a function of lower priority can take over the control by resetting.

The priority (E_BA_LightingPrio) of the output telegram can be defined via the parameter $ePrio$ (VAR_INPUT CONSTANT PERSISTENT). It is preset to $eManualActuator$.

Information about inherited elements

The function block inherits from the internal base class *FB_BA_BaseLightingEvt*, which contains a telegram counter for determining the last command sent.

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  fLgtVal  : REAL;
  fLgtT    : REAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	A TRUE signal on this input activates the function block and transfers the entered setpoints together with the active flag in the light setting telegram ST_BA_Lighting [► 216]. A FALSE signal resets the active flag again and sets the light value to zero.
fLgtVal	REAL	Light value [%]
fLgtT	REAL	Light temperature [K]

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio      : E_BA_LightingPrio := E_BA_LightingPrio.eScene1;
END_VAR
```

Name	Type	Description
ePrio	E_BA_LightingPrio	Priority E_BA_LightingPrio of the telegram, preset to <i>eScene1</i> .

Outputs

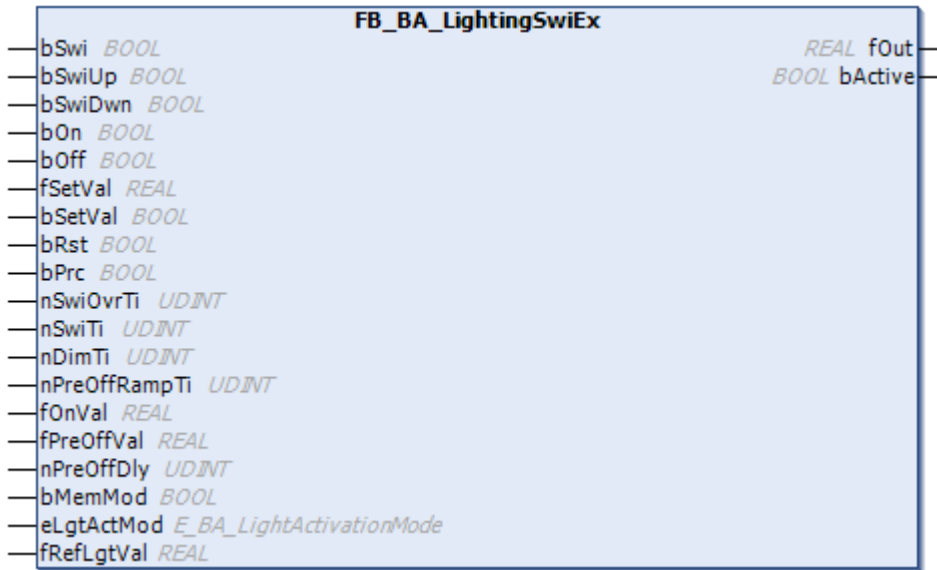
```
VAR_OUTPUT
  stLighting : ST_BA_Lighting;
  bActv      : BOOL;
END_VAR
```

Name	Type	Description
stLighting	ST_BA_Lighting	Light setting telegram
bActv	BOOL	Active

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.2.4 FB_BA_LightingSwiEx



The function block FB_BA_LightingSwiEx represents a universal switch function block. The functionality is similar to the FB_BA_ConstLgtControlEx, but without control.

The function block not only switches the light on and off, it itself can also be active (output `bActive` = TRUE) and not active (output `bActive` = FALSE and `fOut` = 0). This makes it suitable for interaction with other lighting functions in a priority selection in which the output `fOut` is considered depending on the output `bActive`.

Considering this fact, the function works with an input for the brightness state of a reference lamp. If several lighting functions control the same lamp or the same lighting group independently of each other by priority control, the function block does not know whether it is the active one. With the help of the reference input, however, it is always possible to synchronize to the current value before each switching action and to assess before a toggle action whether the light should be switched on or off in the next step.

If it is ensured that the function is the only one that controls the light, the input `fRefLgtVal` is not to be assigned. It is pre-initialized with the value -1, which indicates to the function block that the input is not linked.

However, pre-initialization also means that it is not possible to delete an input link online without distorting the function of the function block!

Switch-on behavior

From the non-activated state, short button presses at `bSwi`, `bSwiUp` or `bSwiDown`, as well as rising edges at `bOn` or `bPrc` initially activate the function and switch the light on.

However, the presence signal input `bPrc` is only active in full automatic mode.

The fully automatic or semi-automatic operation mode can be parameterized at the `eMode` input.

In the active state then applies:

- Short button presses at the inputs `bSwi`, `bSwiUp` and `bSwiDown` switch the light on and off alternately.
- With `bOn` and `bOff`, targeted switching on or off occurs.
- A long button press at the input `bSwi` alternately dims the light up and down. Dimming takes place via a ramp, which is specified at `nDimTi` in seconds related to 0...100%.
- A long button press on the input `bSwiUp` dims the light up specifically. Dimming takes place via a ramp, which is specified at `nDimTi` in seconds related to 0...100%.
- A long button press at the input `bSwiDown` dims the light down specifically. Dimming takes place via a ramp, which is specified at `nDimTi` in seconds related to 0...100%.

Memory mode

If the memory mode is activated at the input *eMode*, the value that the function block had before the last manual switch-off is stored internally. This value is then adopted the next time the system is switched on in manual mode.

If Memory-mode is deactivated, the switch-on always takes place with the value *fOnValMan*.

Deactivating the function

From the active mode two events let the function change to the deactivated state (output *bActive* = FALSE):

- a falling edge at the input *bPrc*.
- a TRUE signal at input *bRst*. This input is intended for central deactivation of the function block or in case there is no occupancy sensor.

On deactivation, a ramp *nPreOffRampTi* (in seconds related to 100% to 0%) is used to dim down to a base value *fPreOffVal*. This value represents the time *nPreOffDly* in seconds at the output before switching off. The function block is then deactivated.

Inputs

```
VAR_INPUT
  bSwi          : BOOL;
  bSwiUp        : BOOL;
  bSwiDwn       : BOOL;
  bOn           : BOOL;
  bOff          : BOOL;
  fSetVal       : REAL;
  bSetVal       : BOOL;
  bRst          : BOOL;
  bPrc          : BOOL;
  nSwiOvrTi     : UDINT := 250;
  nSwiTi        : UDINT := 2;
  nDimTi        : UDINT := 5;
  nPreOffRampTi : UDINT := 10;
  fOnVal        : REAL  := 100;
  fPreOffVal    : REAL  := 20;
  nPreOffDly    : UDINT := 20;
  eMode         : E_BA_LightControlModeEx;
  fRefLgtVal    : REAL  := -1;
END_VAR
```

Name	Type	Description
bSwi	BOOL	If the constant light control function block is not yet active, a short button press first switches the control on, then further signals cause the function block to switch to the manual mode and the following applies: short button press: on / off, long button press: alternately dimming up or down.
bSwiUp	BOOL	If the constant light control function block is not yet active, a short button press first switches the control on, then further signals cause the function block to switch to the manual function and the following applies: short button press: on / off, long button press: selective dimming up.
bSwiDwn	BOOL	If the constant light control function block is not yet active, a short button press first switches the control on, then further signals cause the function block to switch to the manual function and the following applies: short button press: on / off, long button press: selective dimming down.
bOn	BOOL	If the constant light control function block is not yet active, a TRUE signal first switches on the control, then this input only refers to switching on in manual mode.
bOff	BOOL	Switches the light off, the constant light control function block is still in manual mode. If the constant light control function block was previously in control mode, it is now in manual mode.
fSetVal	REAL	Manual preset value. The values at the input <i>fSetVal</i> are taken over as light output value by a rising edge at <i>bSetVal</i> . The function block is activated by setting if it was previously deactivated.
bSetVal	BOOL	Acceptance of the manual preset value, see <i>fSetVal</i> .
bRst	BOOL	This input switches off the constant light control function block. The shutdown is done by ramping and dwell time on a base light value, see below: <i>fPreOffVal</i> , <i>nPreOffDly</i> and <i>nPreOffRampT</i> .
bPrc	BOOL	Presence signal input. If the constant light control function block is configured in fully automatic mode, a rising edge can activate the function block via this input if it was not active before. A falling edge deactivates it. In semi-automatic mode, only a falling edge at this input deactivates the constant light control function block. "Activating" here means that the function is switched on in manual mode (outputs <i>bControlMode</i> = TRUE and <i>bManualMode</i> = FALSE). Deactivating means: the function is not active in manual or control mode (outputs <i>bControlMode</i> = FALSE and <i>bManualMode</i> = FALSE)
nSwiOvrTi	UDINT	Distinction time [ms] between short and long button press.
nSwiTl	UDINT	Ramp for the switching functions in seconds [s], related to a dimming from 0 to 100%.
nDimTi	UDINT	Ramp for the dimming functions in seconds [s], related to a dimming from 0 to 100%.
nPreOffRampTi	UDINT	Ramp used to drive to a base value <i>fPreOffVal</i> before switching off.
fOnVal	REAL	Switch-on value, if a possibility of the "Memory-Mode" is not selected via the input <i>eMode</i> (see E_BA_LightControlModeEx).
fPreOffVal	REAL	Base value and hold time at this value before switching off the overall function. If the current light value already falls below the base value, this function is not active and it is switched off immediately

Name	Type	Description
fPreOffDly	UDINT	Base value and hold time at this value before switching off the overall function. If the current light value already falls below the base value, this function is not active and it is switched off immediately
eMode	E_BA_LightControlModeEx	Operation mode. Selection from the possible combinations of semi and full automatic and activated or deactivated "Memory mode". ▪Semi-automatic: The constant light control function block is only activated by pressing a button; it is automatically deactivated when not occupied. ▪Fully automatic: The constant light control function block can be activated both by an incoming presence and by pressing a button; it is automatically deactivated when not occupied. ▪Memory mode: When the light is switched on in manual mode, the light assumes the value that the function had before it was last switched off. If the "Memory-Mode" is not active, the switch-on value is defined via <i>fOnValMan</i>.
fRefLgtVal	REAL	Light value of a single light actuator representing the light zone. This input is used to assess whether in manual mode the next button press switches on or off, or from which value dimming is to take place. For automatic operation this input is important to start with the "correct" value when control is regained. This input is pre-initialized with "-1". In the event of non-assignment, this is recognized and it is assumed that the function block is the only one to be controlled. Thus, the output value <i>fOut</i> is used as the light value to assess the next switching actions.

Outputs

```
VAR_OUTPUT
  fOut      : REAL;
  bActive   : BOOL;
END_VAR
```

Name	Type	Description
fOut	REAL	Light output value, 0...100%
bActive	BOOL	Indicates that the function block is active.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.2.5 FB_BA_LightingTgmSel8 / FB_BA_LightingTgmSel4



The function of the blocks is explained at FB_BA_LightingTgmSel8 as an example

The function blocks are used for priority control for up to 4 or up to 8 lighting control telegrams (*stLighting_Prio1 ... stLighting_Prio4*, or *stLighting_Prio1 ... stLighting_Prio8*) of type *ST_BA_Lighting*.

The active telegram with the highest priority is output at the output *stLighting*. "Active" means that the variable *bActv* is set within the structure of the positioning telegram. The priority is stored within the telegram structure as *ePrio*, whereby the lower the value of *ePrio*, the higher the priority.

For telegrams with the same priority, the last changed one (last writer wins) is valid, determined by the variable *nEvtInc*.

This function block is to be programmed in such a way that one of the applied telegrams is always active. If a telegram is not active, an empty telegram is output at the output, i.e. *fLgtVal* = 0, *fLgtT* = 0, *bDimUp* = FALSE, *bDimDwn* = FALSE, *bDimMod* = FALSE, *bActv* = FALSE.

Inputs

```
VAR_INPUT
    stLightingTgm_1    : ST_BA_Lighting;
    stLightingTgm_2    : ST_BA_Lighting;
    stLightingTgm_3    : ST_BA_Lighting;
    stLightingTgm_4    : ST_BA_Lighting;
    stLightingTgm_5    : ST_BA_Lighting;
    stLightingTgm_6    : ST_BA_Lighting;
    stLightingTgm_7    : ST_BA_Lighting;
    stLightingTgm_8    : ST_BA_Lighting;
END_VAR
```

Name	Type	Description
stLightingTgm_N	ST_BA_Lighting	Telegram inputs

Outputs

```
VAR_OUTPUT
    stLighting          : ST_BA_Lighting;
    nNumActvTgm         : UINT;
    ePrioActvTgm        : E_BA_LightingPrio;
END_VAR
```

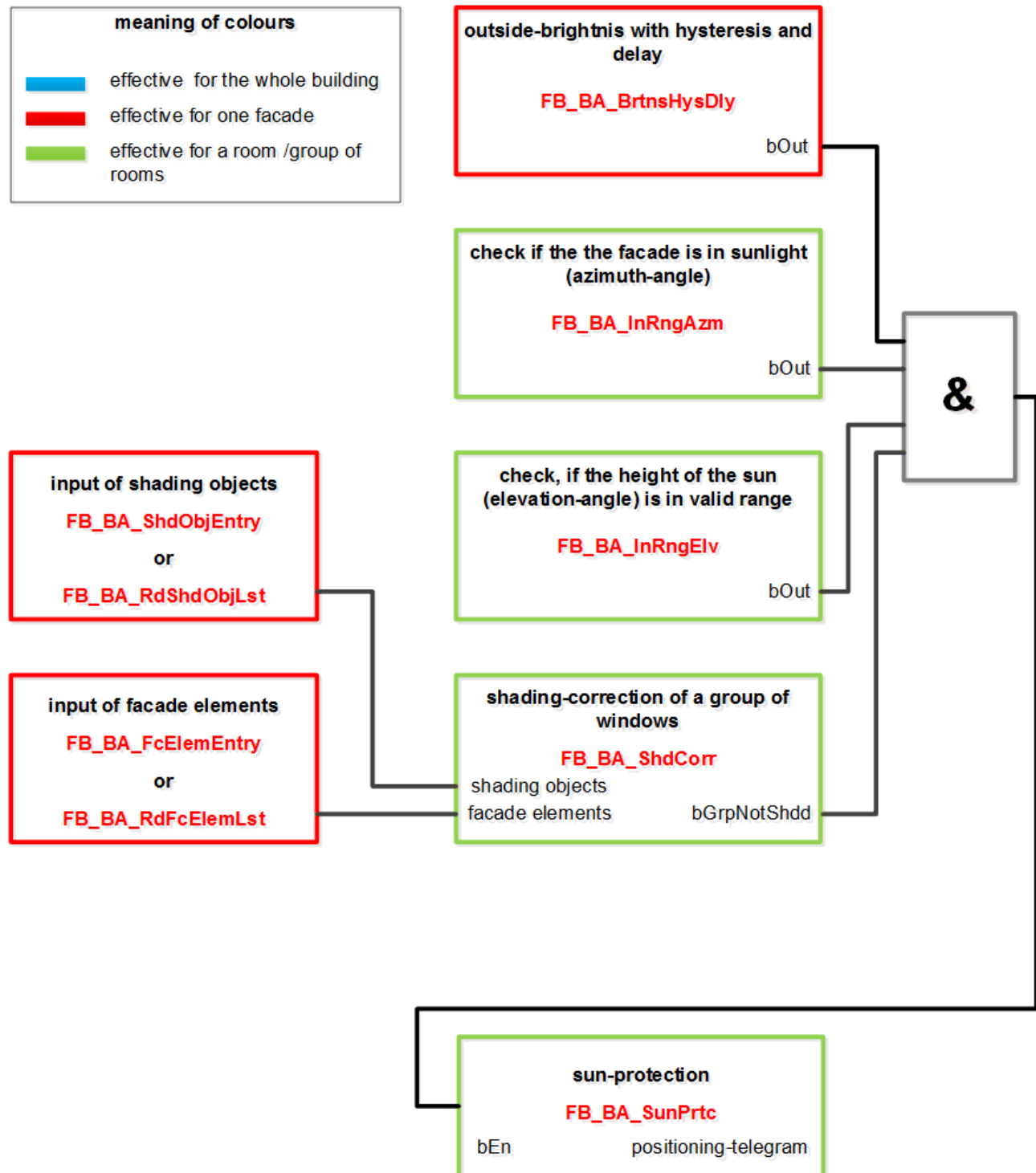
Name	Type	Description
stLighting	ST_BA_Lighting	Resulting telegram
nNumActvTgm	UINT	Indicates which input results, e.g. if <i>stLightingTgm_3</i> is passed, <i>nNumActvTgm</i> = 3. If <i>nNumActvTgm</i> = 0, no telegram is active.
ePrioActvTgm	E_BA_LightingPrio	This output indicates the priority of the active telegram.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3 Sun protection

6.1.2.2.3.1.3.3.1 Overview of shading correction



6.1.2.2.3.1.3.3.2 Shading correction: Basic principles and definitions

The shading correction can be used in conjunction with the automatic sun function or lamella setpoint tracing. The function checks whether a window or a window group that is assigned to a room, for example, is temporarily placed in the shade by surrounding buildings or parts of its own building. Sun protection for windows that stand in the shadow of surrounding buildings or trees is not necessary and may even be disturbing under certain circumstances. On the basis of data entered regarding the facade and its

surroundings, the shading correction determines which parts of the front are in the shade. Hence, it is then possible to decide whether the sun protection should be active for individual windows or window groups. Apart from the current position of the sun, the shading of the individual windows depends on three things:

- the orientation of the facade
- the position of the windows
- the positioning of the shading objects

The following illustrations are intended to describe these interrelationships and to present the parameters to be entered.

Orientation of the facade

Observation from above

A two-dimensional coordinate system is required for observing the shadow cast on the facade, which is why the x- and y-axis were placed on the facade. The zero point is thereby at the bottom left on the base, as if one were regarding the facade from the front. For the calculation of the shading objects the Z component is then also added. Its axis points from away the facade and has the same zero point as the X and Y axis.

In the northern hemisphere, the horizontal sun position (azimuth angle) is determined from the north direction by definition. The facade orientation is likewise related to north, wherein the following applies to the line of sight from a window in the facade:

Line of sight	Facade orientation
North	$\beta=0^\circ$
East	$\beta=90^\circ$
South	$\beta=180^\circ$
West	$\beta=270^\circ$

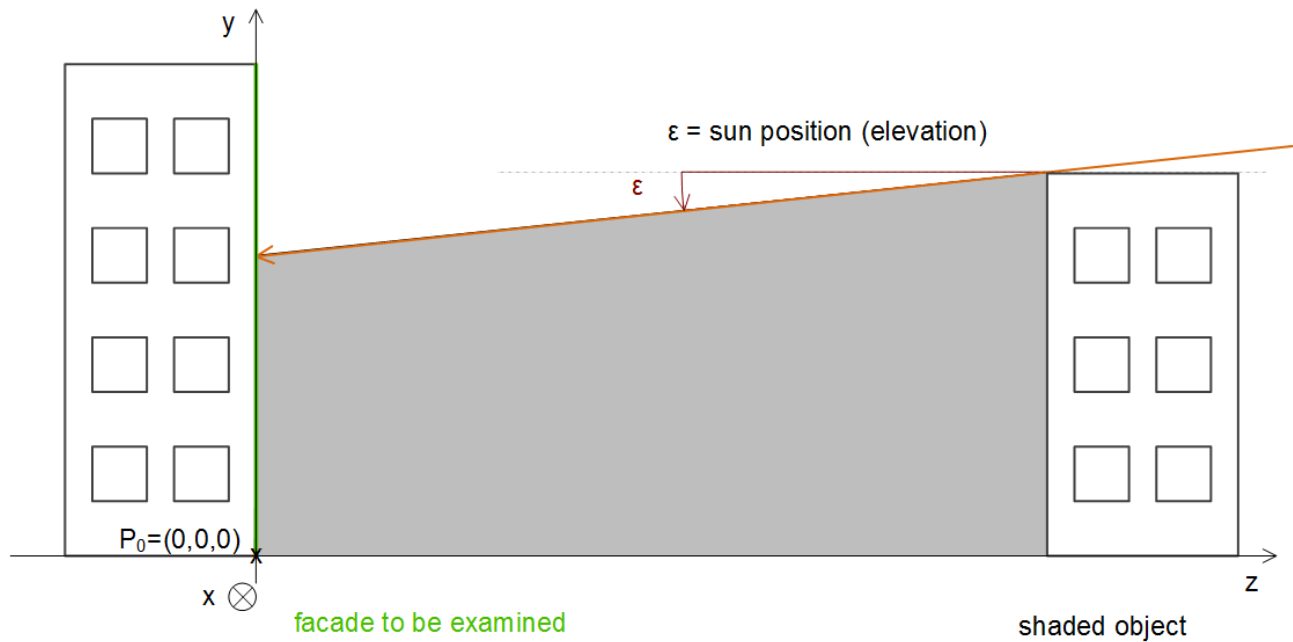
In the southern hemisphere the sun path is reversed: Although it also rises in the east, at midday it is in the north. The facade orientation is adjusted to this path:

Line of sight	Facade orientation
South	$\beta=0^\circ$
East	$\beta=90^\circ$
North	$\beta=180^\circ$
West	$\beta=270^\circ$

For convenience, the other explanations refer to the northern hemisphere. The calculations for the southern hemisphere are analogous. When the function block `FB_BA_ShdCorr` [► 498] (shading correction) is parameterized they are activated through a boolean input, *bSouth*

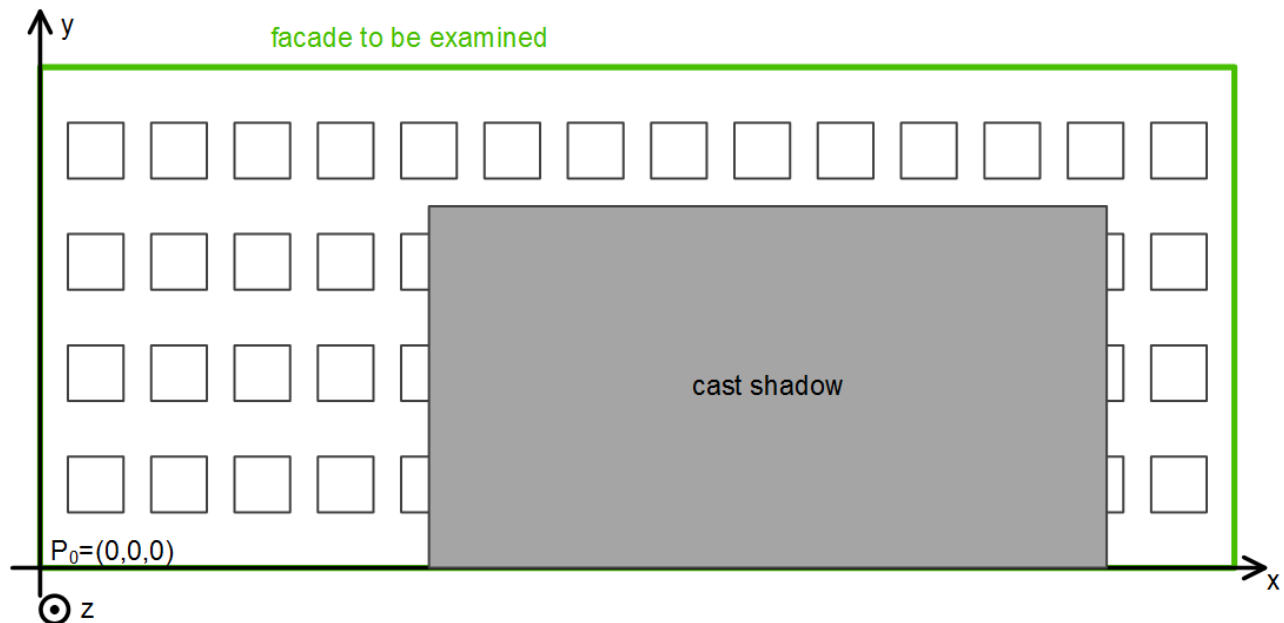
The following two illustrations are intended to further clarify the position of the point of origin P_0 as well as the orientation of the coordinate system:

Observation from the side



The angle of elevation (height of the sun) can be represented using this illustration: by definition this is 0° at sunrise (horizontal incidence of light) and can reach maximally 90° , but this applies only to places within the Tropic of Cancer and the Tropic of Capricorn.

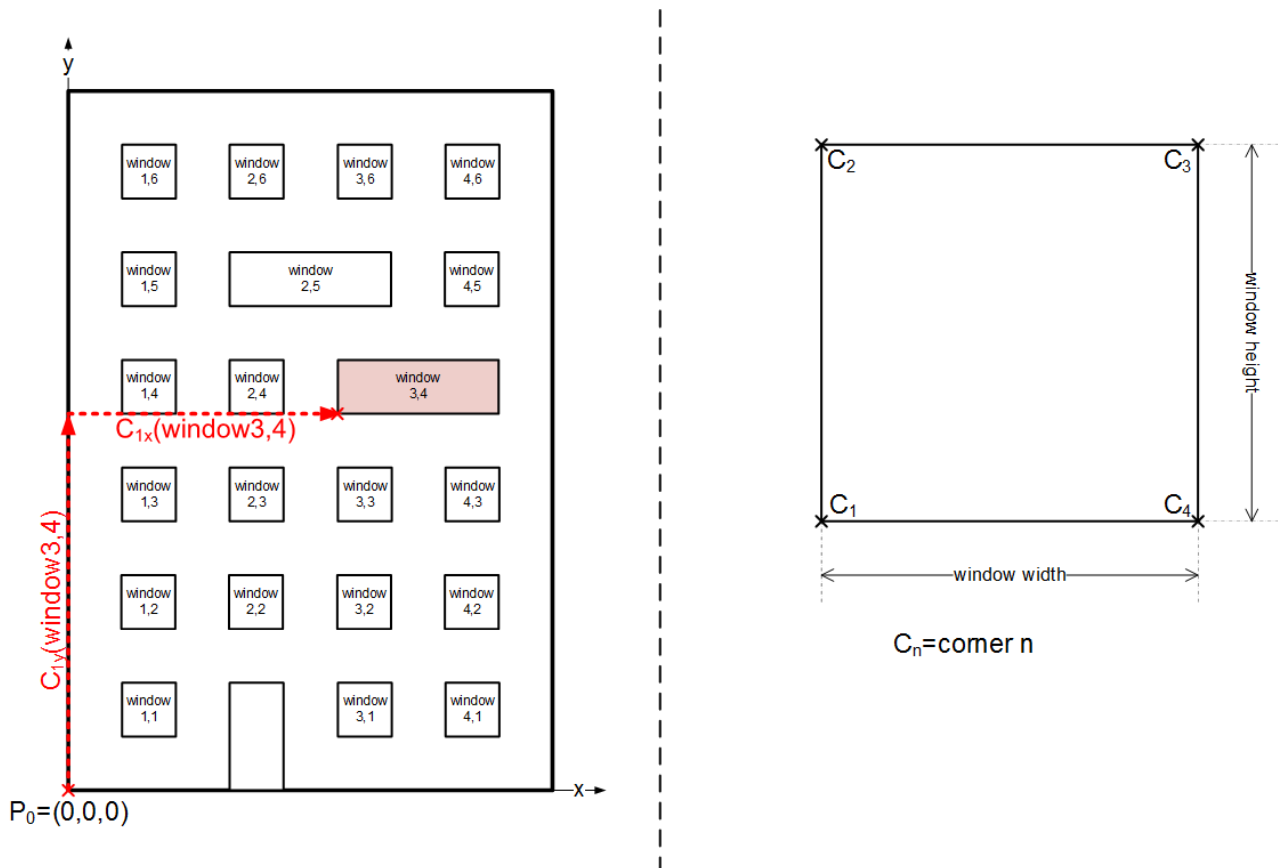
Observation from the front



Here, the position of the point of origin, P_0 , at the bottom left base point of the facade is once more very clear. Beyond that the X-Y orientation is illustrated, which is important later for the entry of the window elements.

Position of the windows

The position of the windows is defined by the specification of their bottom left corner in relation to the facade coordinate system. Since a window lies flat on the facade, the entry is restricted to the X and Y coordinates.



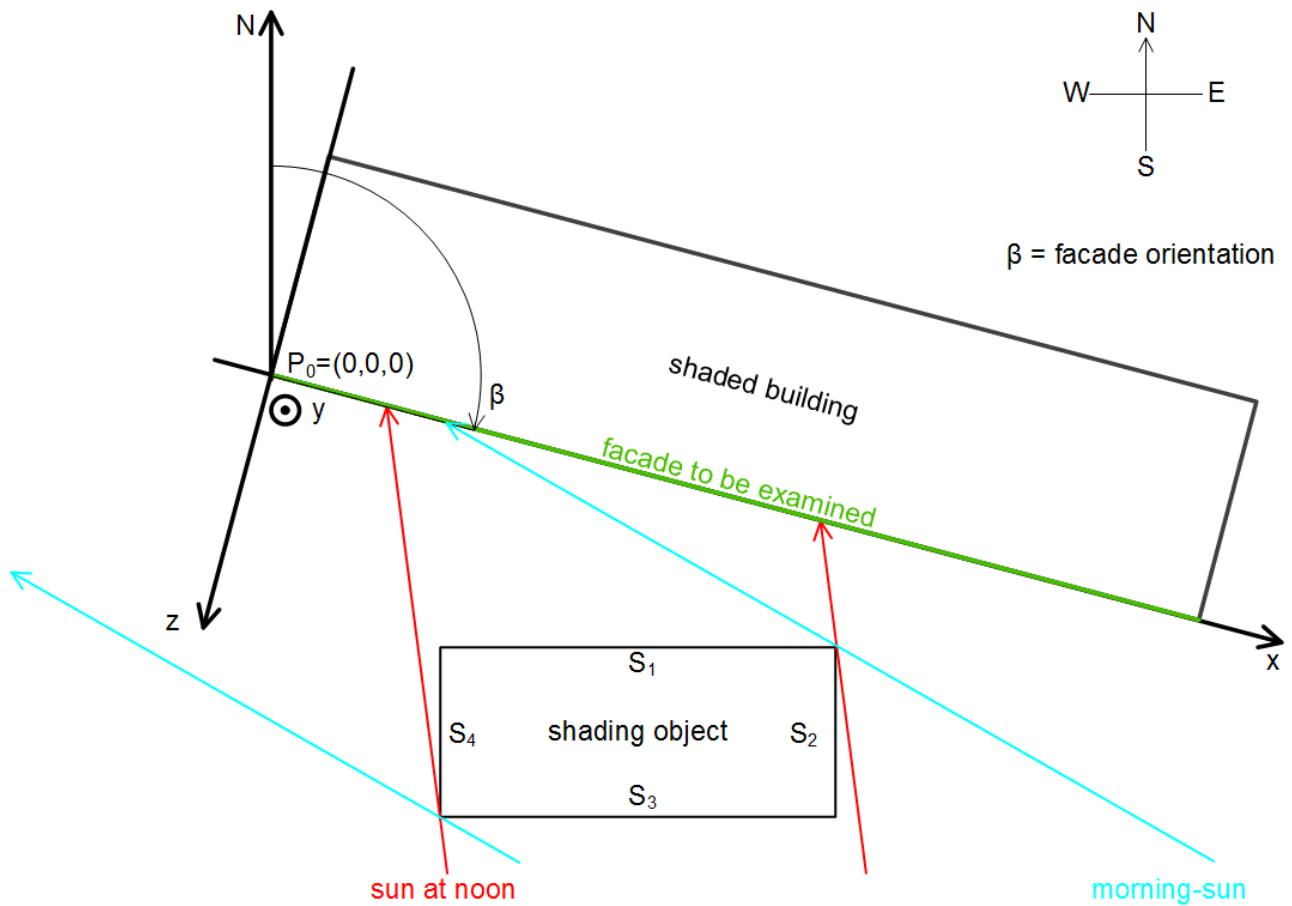
In addition, the window width and the window height have to be specified.

The position of each window corner on the facade is determined internally from the values entered. A window is considered to be in the shade if all corners lie in the shade.

Positioning of the shading objects

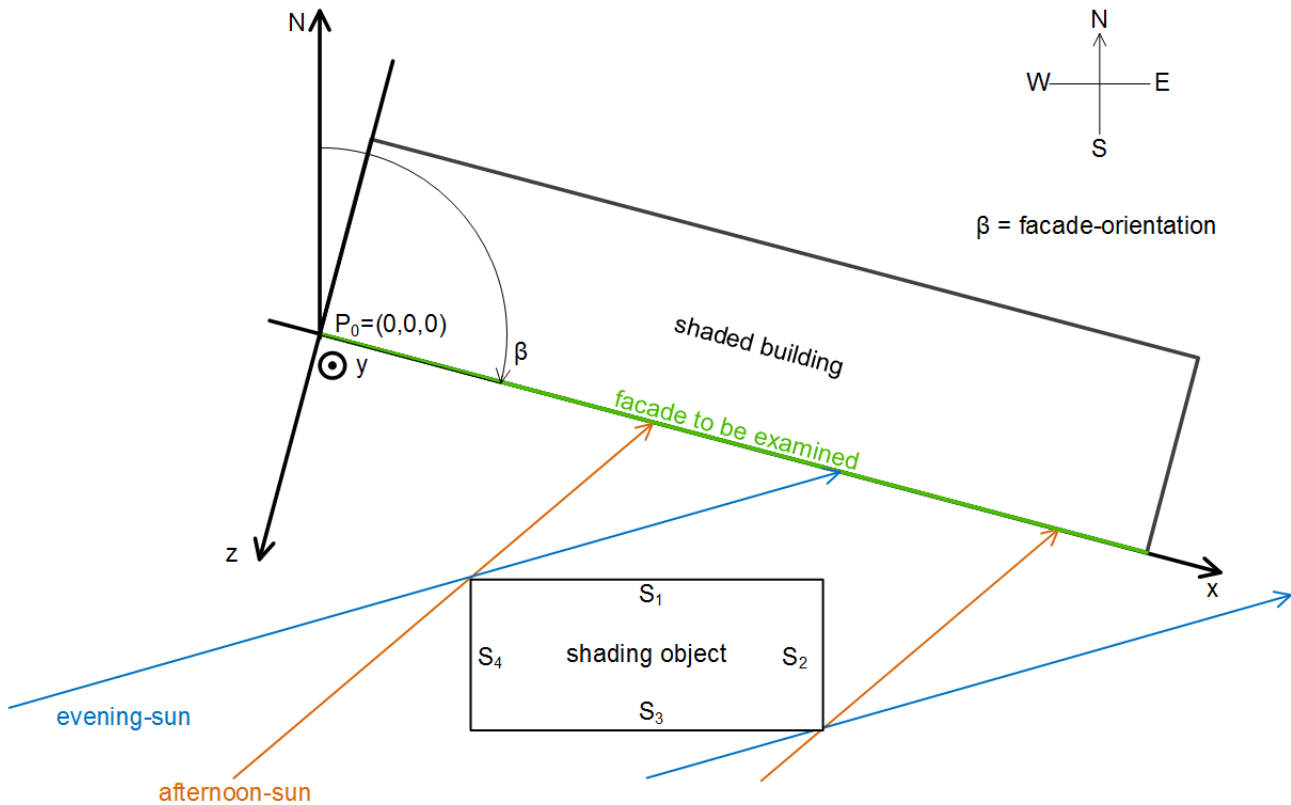
When describing the shading objects, distinction is made between angular objects (building, column) and objects that are approximately spherical (e.g. trees). Angular objects can be subdivided into square shadow-casting facades according to their shadows, noting which cast the main shadow throughout the day:

Morning/noon

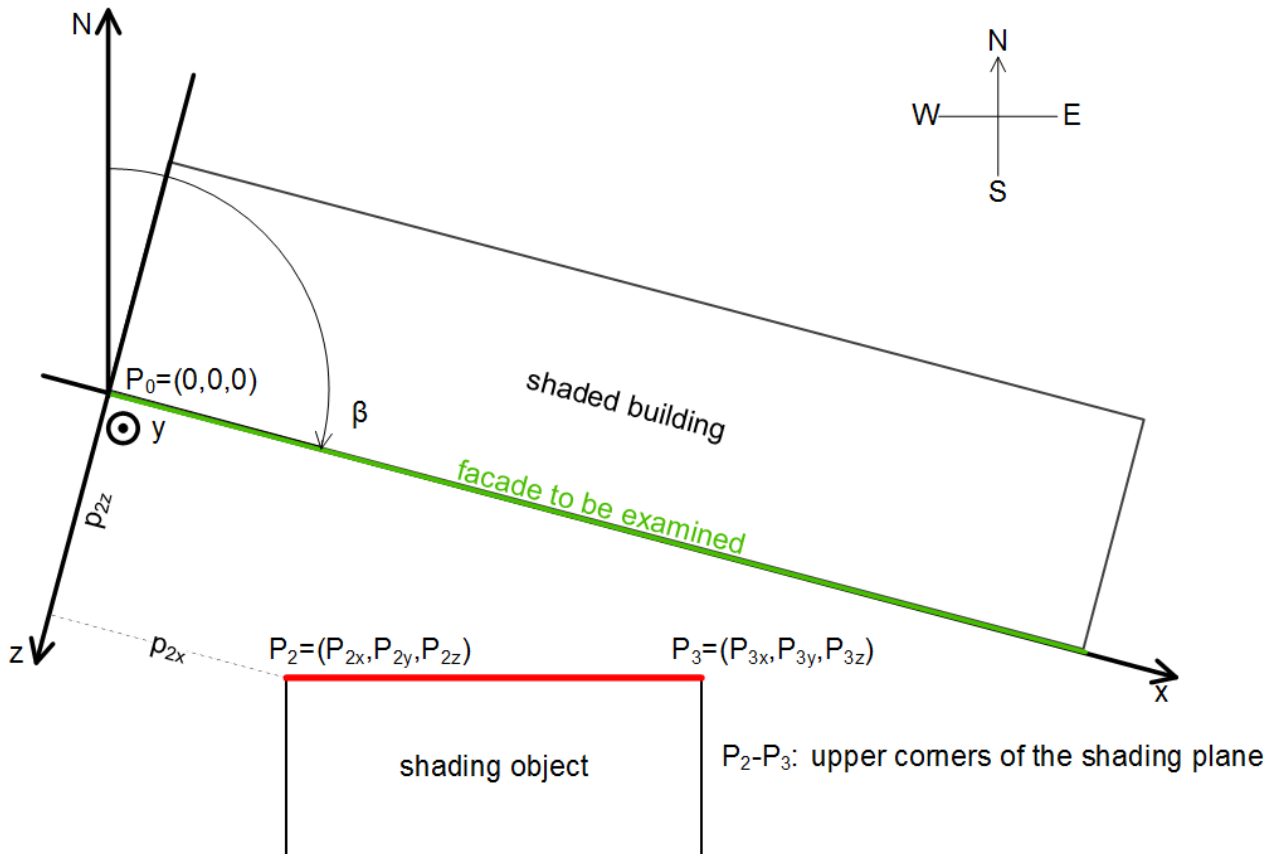


In the morning and around noon, the shadow is mainly cast by the sides S_1 and S_4 . S_2 and S_3 do not have to be considered, unless they are higher.

Afternoon/evening



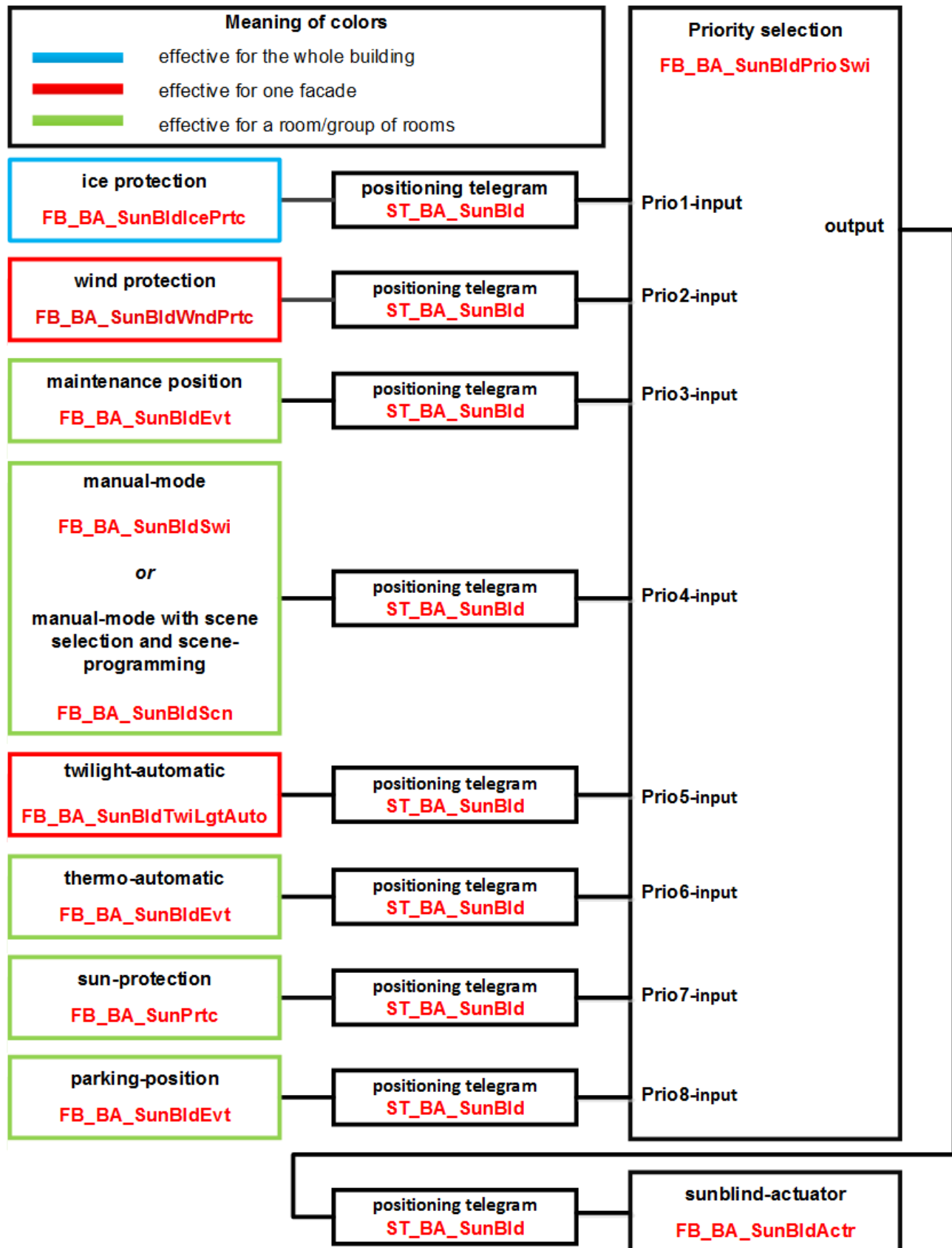
In the afternoon and evening, the total shade can be determined solely through S_1 and S_2 . In this case it is therefore sufficient to specify S_1 , S_2 and S_4 as shadow casters. The entry is made on the basis of the four corners or their coordinates in relation to the zero point of the facade:



In this sketch only the upper points, P_2 and P_3 , are illustrated due to the plan view. The lower point P_1 lies underneath P_2 and P_4 lies underneath P_3 .

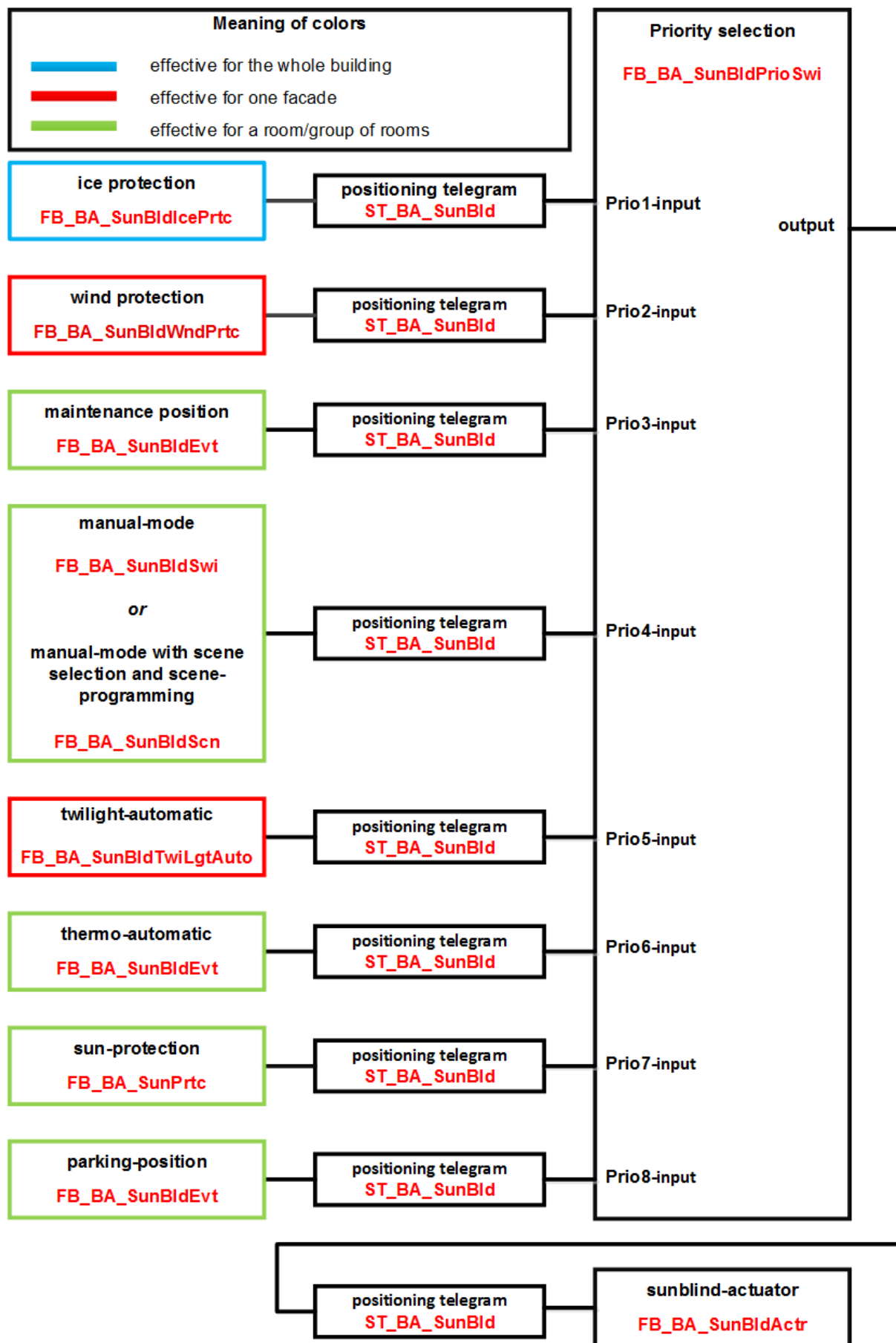
The input of shadow-casting ball elements is done by entering the center of the ball and its radius:

Ball elements



A "classification" of the ball element as in the case of the angular building is of course unnecessary, since the shadow cast by a ball changes only its direction, but not its size.

6.1.2.2.3.1.3.3.3 Overview of automatic sun protection



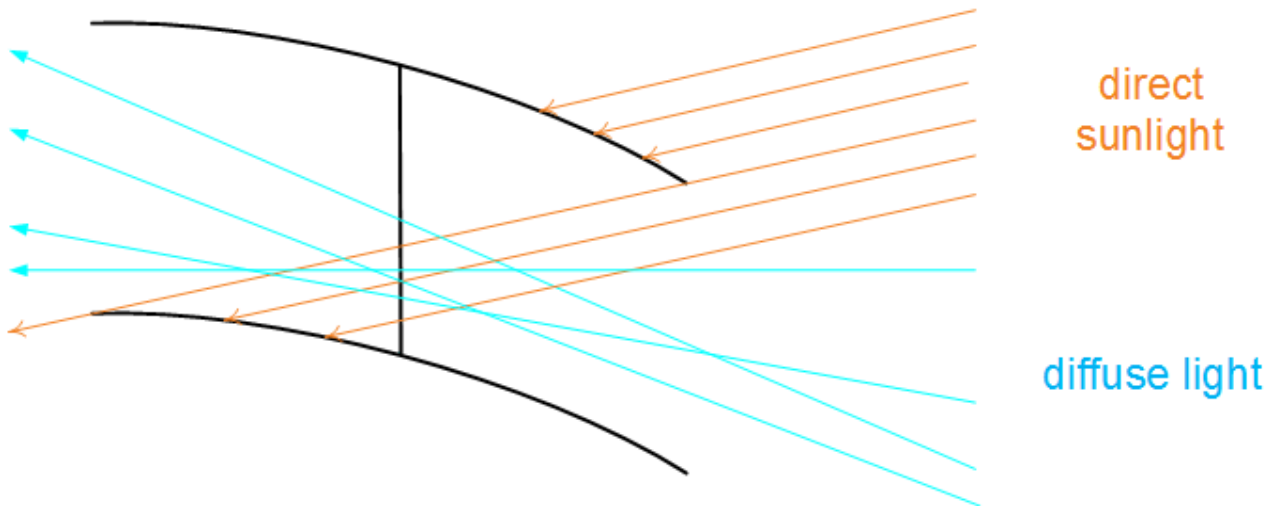
6.1.2.2.3.1.3.3.4 Sun protection: Basic principles and definitions

The direct incidence of daylight is regarded as disturbing by persons in rooms. On the other hand, however, people perceive natural light to be more pleasant in comparison with artificial light. Two options for glare protection are to be presented here:

- Slat adjustment
- Height adjustment

Slat adjustment

A louvered blind that can be adjusted offers the option of intelligent sun protection here. The position of the slats is cyclically adapted to the current position of the sun, so that no direct daylight enters through the blinds, but as much diffuse daylight can be utilized as possible.



The illustration shows that diffuse light can still enter from underneath, whereas no further direct daylight, or theoretically only a single ray, can enter. The following parameters are necessary for the calculation of the slat angle:

- the current height of the sun (angle of elevation)
- the sun position, i.e. the azimuth angle
- the facade orientation
- the slat width
- the slat spacing

Effective elevation angle

If the blind is viewed in section as above, the angle of incidence does not depend solely on the solar altitude (elevation), but also on the direction of the sun:

- If the facade orientation and the sun position (azimuth) are the same, i.e. the sunlight falls directly onto the facade, the effective light incidence angle is the same as the current elevation angle.
- However, if the sunlight falls at an angle onto the facade as seen from the sun direction, the effective angle is larger for the same angle of elevation.

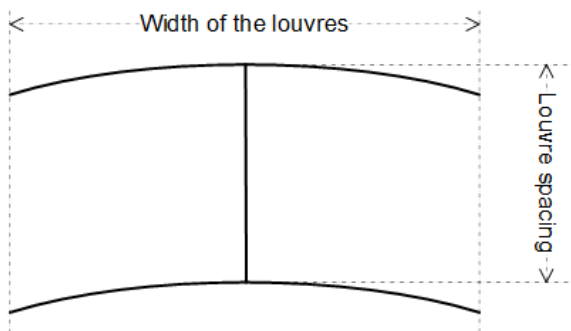
This relationship can easily be illustrated with a set square positioned upright on the table: Viewed directly from the side you can see a triangle with two 45° angles and one 90° angle. If the triangle is rotated, the side on the table appears to become shorter and the two original 45° angles change. The triangle appears to be getting steeper.

We therefore refer to the "effective elevation angle", which describes the proportion of light that falls directly onto the blind.

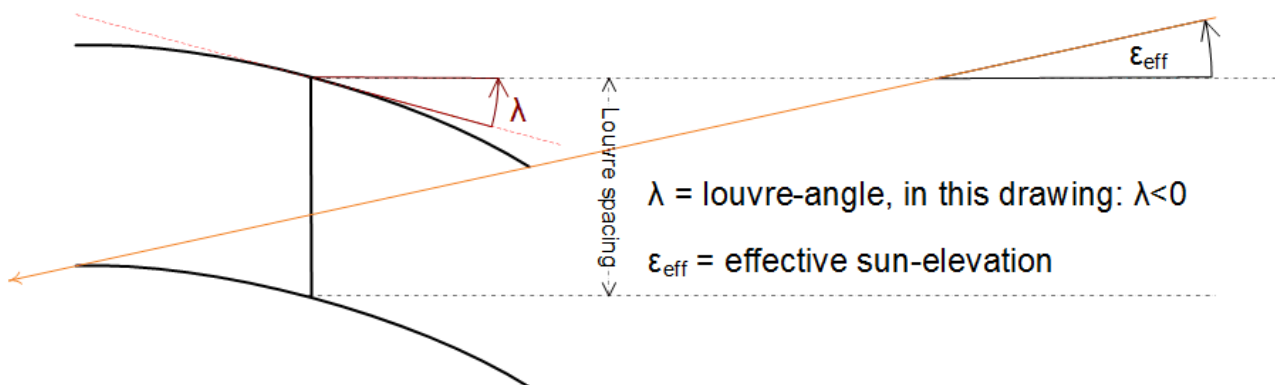
The following three images illustrate the relationship between the effective elevation angle and the blind dimensions, and how the resulting slat angle λ changes during the day:

louvre-angle

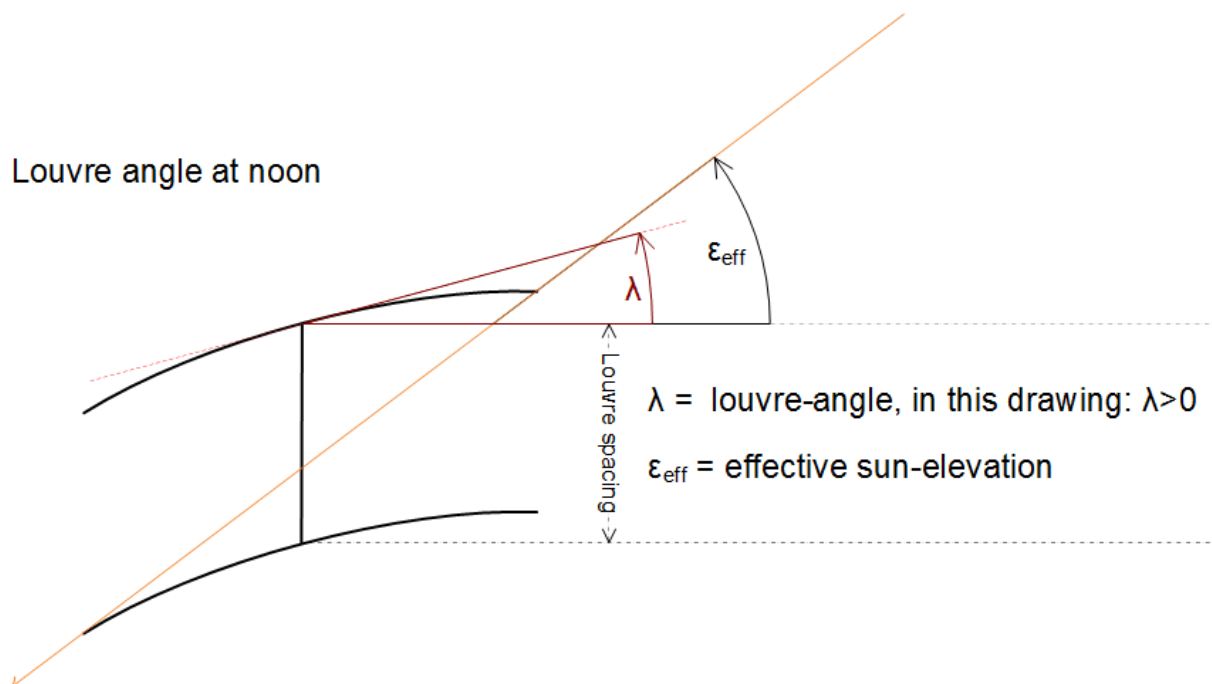
louvre at an angle of $\lambda=0$



Louvre-angle in the morning and in the evening

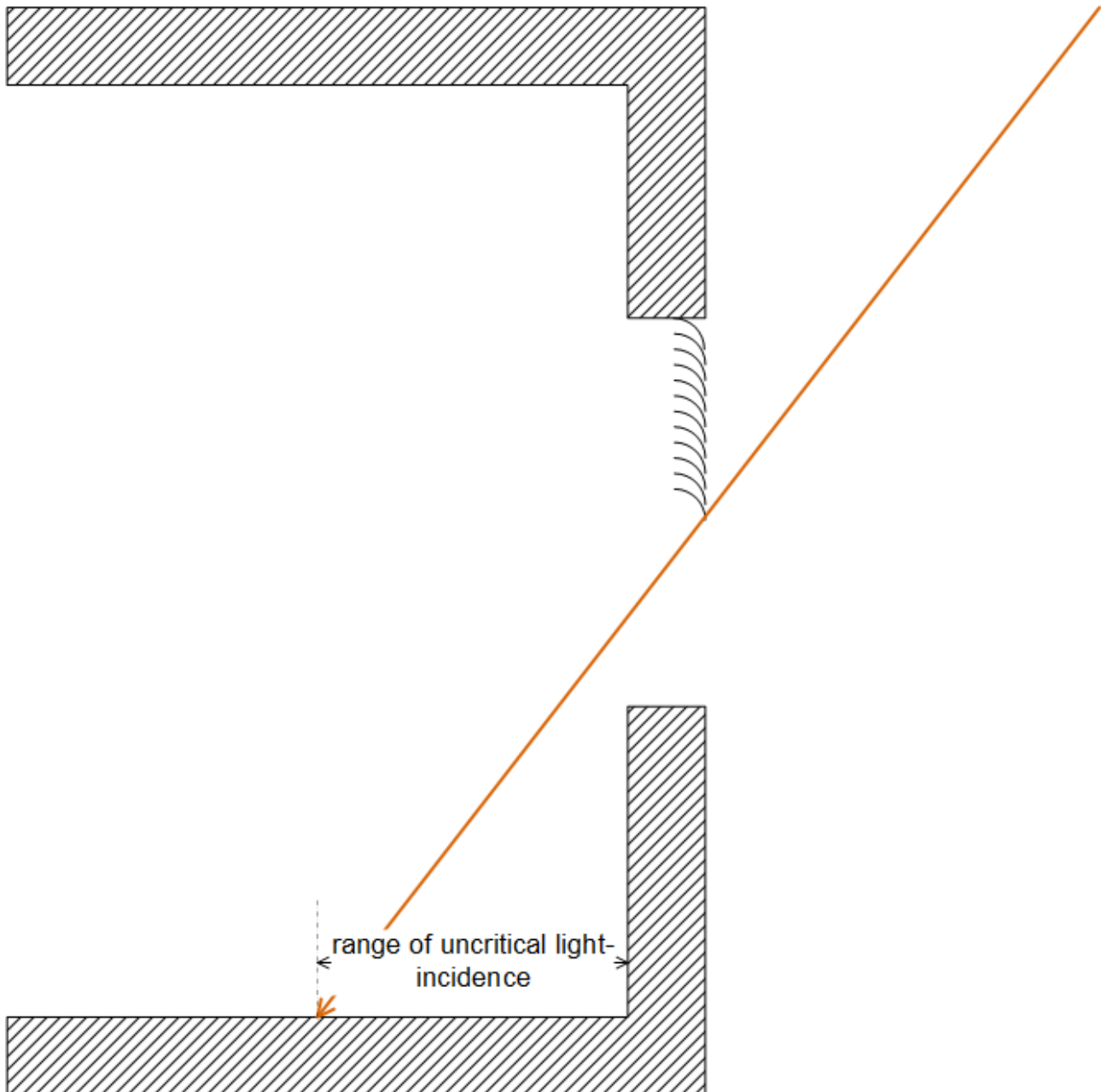


Louvre angle at noon



Height adjustment

With a high position of the sun at midday, the direct rays of sunlight do not penetrate into the full depth of the room. If direct rays of sunlight in the area of the window sill are regarded as uncritical, the height of the sun protection can be adapted automatically in such a way that the rays of sunlight only ever penetrate into the room up to an uncritical depth.

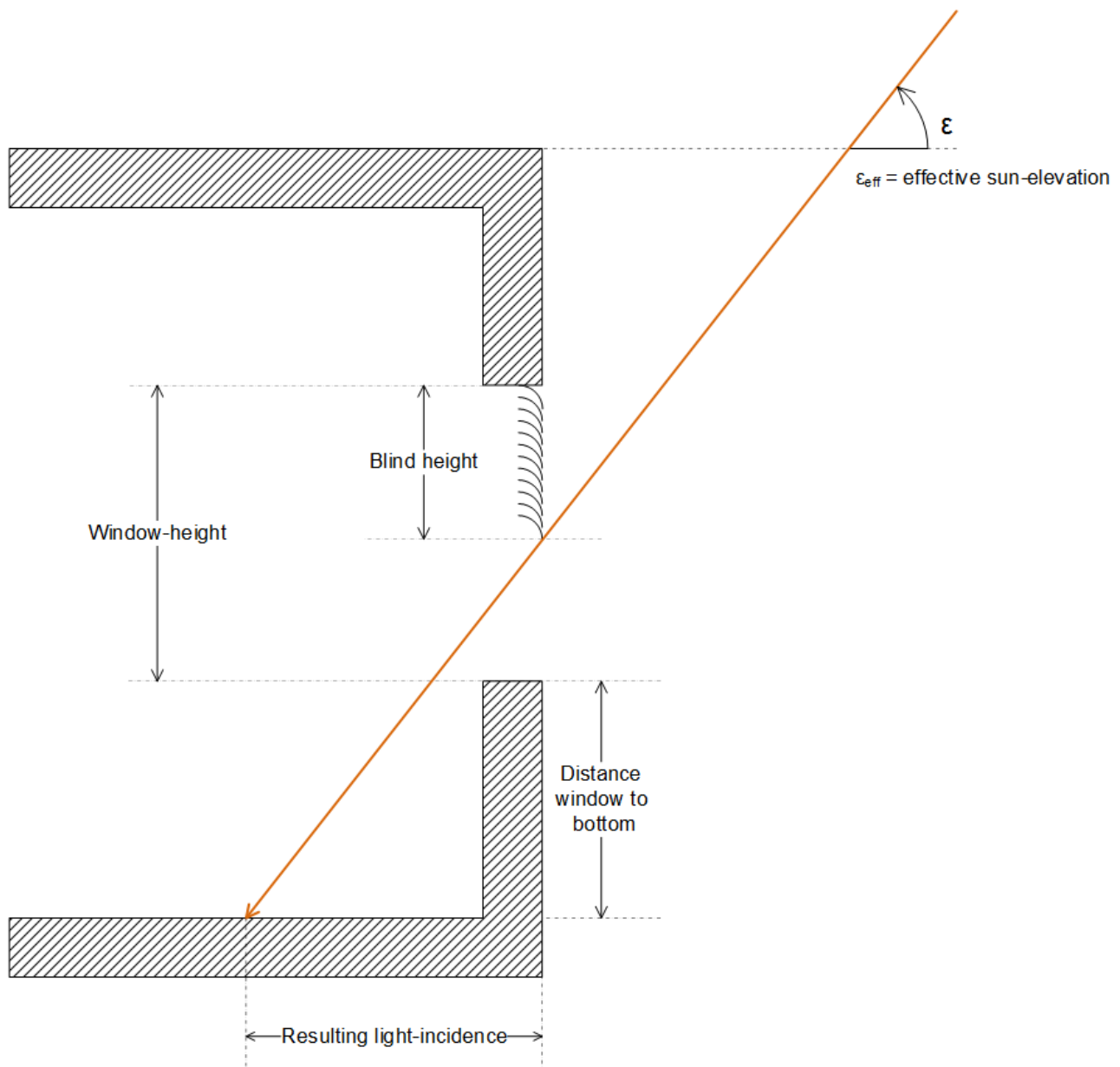


In order to be able to calculate at any time the appropriate blind height that guarantees that the incidence of sunlight does not exceed a certain value, the following values are necessary.

Required for the calculation of the respective blind height:

- Height of the sun (elevation)
- Window height
- Distance between the window and the floor

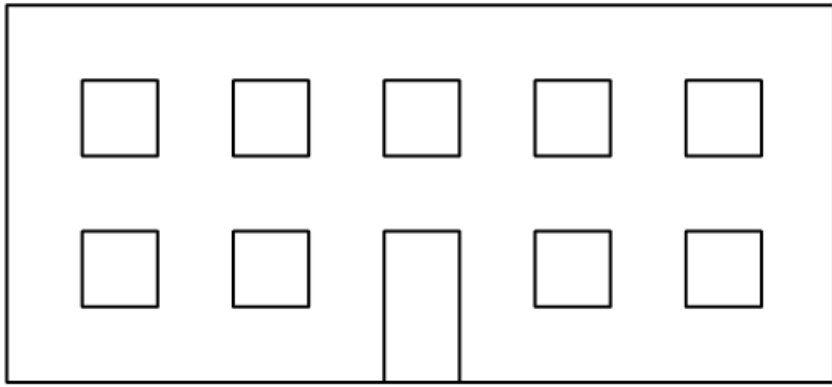
The following illustration shows where these parameters are to be classified:



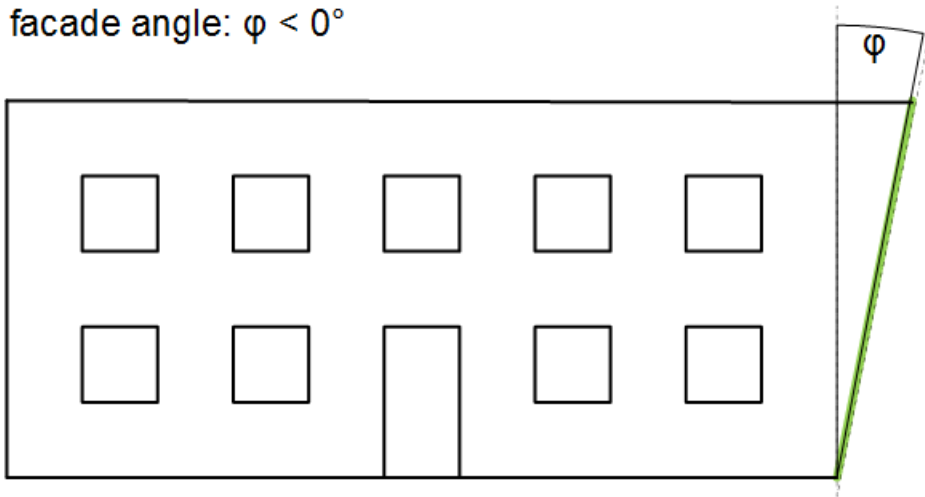
Influence of the facade inclination

In both of the methods of sun protection described, it was assumed that the facade and thus the windows are perpendicular to the ground. In the case of an inclined facade, however, the incidence of light changes such that this influence will also be taken into account. The facade inclination is defined as follows:

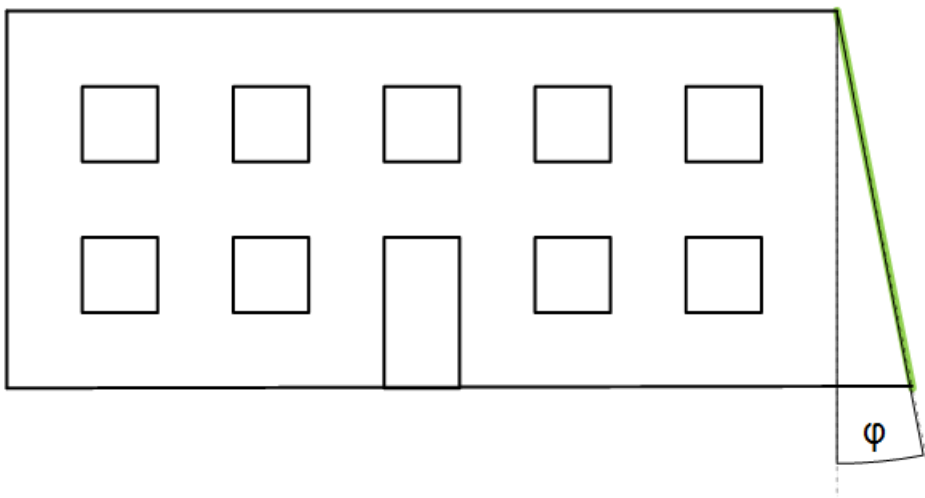
facade angle: $\varphi = 0^\circ$



facade angle: $\varphi < 0^\circ$



facade angle: $\varphi > 0^\circ$



6.1.2.2.3.1.3.3.5 List of shading elements

The data of all shading objects (building components, trees, etc.) per facade are stored in a field of structure elements of type `ST_BA_ShdObj` [► 218] within the program.

The shading correction `FB_BA_ShdCorr` [► 299] reads the information from this list. The management function block `FB_BA_ShdObjEntry` [► 303] reads and writes it as input/output variable.

It is therefore advisable to declare this list globally:

```
VAR_GLOBAL
    aShdObj : ARRAY[1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

The variable `nSunPrt_MaxShdObj` represents the upper limit of the available elements and is defined as a global constant within the program library:

```
VAR_GLOBAL CONSTANT
    nSunPrt_MaxShdObj : UINT := 20;
END_VAR
```

6.1.2.2.3.1.3.3.6 List of facade elements

The data of all windows (facade elements) per facade are saved within the program in a field of structure elements of the type `ST_BA_FcdElem` [► 217].

The management function block `FB_BA_FcdElemEntry` [► 278] and the shading correction `FB_BA_ShdCorr` [► 299] read and write to this list (the latter sets the shading information); they access this field as input/output variables.

It is therefore advisable to declare this list globally:

```
VAR_GLOBAL
    aFcdElem : ARRAY[1..BA_Param.nSunPrt_MaxRowFcd, 1..BA_Param.nSunPrt_MaxColumnFcd] OF ST_BA_FcdElem;
END_VAR
```

The variables `nSunPrt_MaxColumnFcd` and `nSunPrt_MaxRowFcd` define the upper limit of the available elements and are declared as global constants within the program library:

```
VAR_GLOBAL CONSTANT
    nSunPrt_MaxRowFcd : UINT := 10;
    nSunPrt_MaxColumnFcd : UINT := 20;
END_VAR
```

6.1.2.2.3.1.3.3.7 FB_BA_BldPosEntry



The function block `FB_BA_BldPosEntry` serves for the input of interpolation points for the function block `FB_BARSunProtectionEx` [► 329], if this should be operated in the height positioning mode with the help of a table (see `E_BARPosMode`).

In addition to the operation modes "Fixed blind height" and "Maximum light incidence", the function block `FB_BA_SunPrtc` [► 329] also offers the possibility to control the blind height in relation to the sun elevation by means of table entries. By entering several interpolation points, the blind height relative to the respective sun position is calculated by linear interpolation. However, since incorrectly entered values can lead to

malfunctions in [FB_BA_SunPrtc](#) [► 329], this function block is to be preceded by the function block [FB_BA_BldPosEntry](#). Four interpolation points can be parameterized on this function block, whereby a missing entry is evaluated as a zero entry.

The function block does not sort the values entered independently, but instead ensures that the positions of the sun entered in the respective interpolation points are entered in ascending order. Unintentional erroneous entries are noticed faster as a result.

The values chosen for *fSunElv1* ... *fSunElv4* must be unique, for example, the following situation must be avoided:

[*fSunElv1* = 10 ; *fPos1* = 50] and simultaneously [*fSunElv2* = 10 ; *fPos2* = 30].

This would mean that there would be two different target values for one and the same value, which does not allow a unique functional correlation to be established.

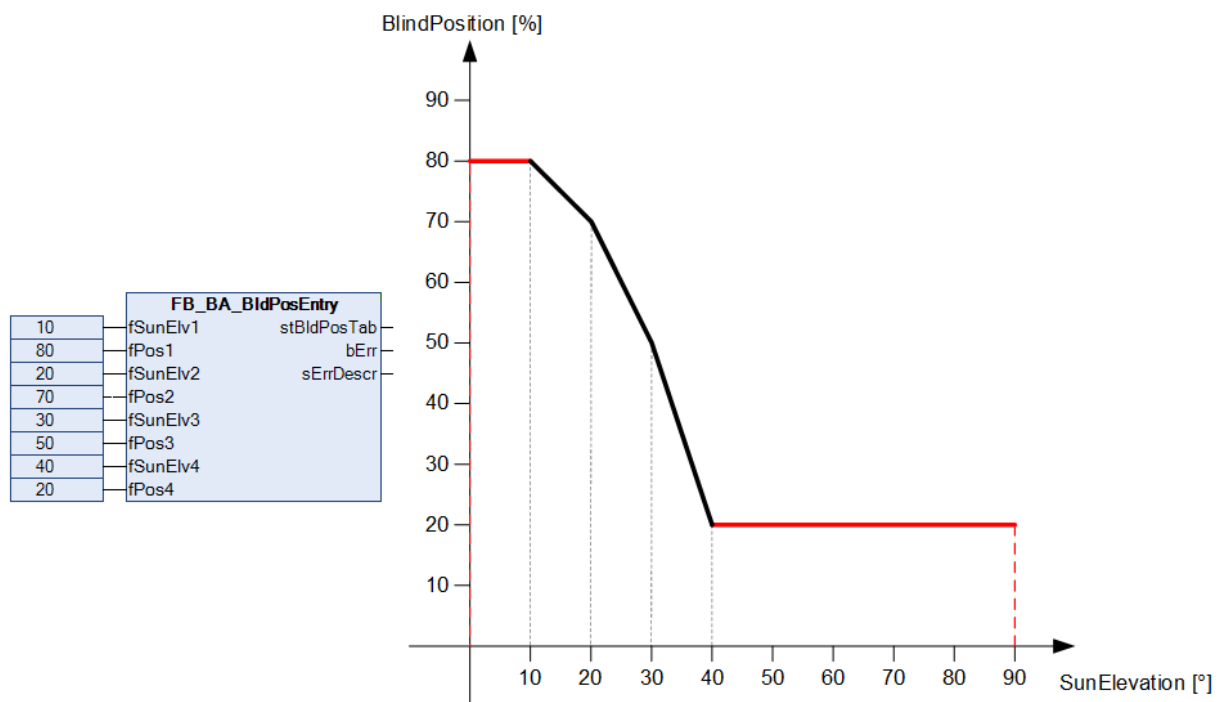
In addition, the entries for the position of the sun and blind height must lie within the valid range.

Mathematically this means that the following conditions must be satisfied:

- $fSunElv1 < fSunElv2 < fSunElv3 < fSunElv4$ - (values ascending and not equal)
- $0 \leq fSunElv \leq 90$ (["°"] - scope source values)
- $0 \leq fPos \leq 100$ (in percent - scope target values)

The function block checks the entered values for these conditions and issues an error message if they are not met. In addition, the control value *bVld* of [ST_BA_BldPosTab](#) [► 216] is set to FALSE.

Furthermore, the function block independently ensures that the boundary areas are filled out: Internally, another interpolation point is set at *fSunElv* = 0 with *fPos1* and another one above *fSunElv4* at *fSunElv* = 90 with *fPos4*. This ensures that a sensible target value exists for all valid input values $0 \leq fSunElv \leq 90$ **without** the user having to assign an entry for *fSunElv* = 0 and *fSunElv* = 90:



The actual number of interpolation points transferred to the function block [FB_BA_SunPrtc](#) [► 329] thus increases to 6, see [ST_BA_BldPosTab](#) [► 216].

The interpolation of the values takes place in the glare protection function block.

Inputs

```
VAR_INPUT
  fSunElv1 : REAL;
  fPos1    : REAL;
  fSunElv2 : REAL;
  fPos2    : REAL;
  fSunElv3 : REAL;
  fPos3    : REAL;
```



```
fSunElv4 : REAL;
fPos4    : REAL;
END_VAR
```

Name	Type	Description
fSunElv1	REAL	Position of the sun at the first interpolation point [°] (0...90).
fPos1	REAL	Blind position (degree of closure) at the first interpolation point [%] (0...100).
fSunElv2	REAL	Position of the sun at the second interpolation point [°] (0...90).
fPos2	REAL	Blind position (degree of closure) at the second interpolation point [%] (0...100).
fSunElv3	REAL	Position of the sun at the third interpolation point [°] (0...90).
fPos3	REAL	Blind position (degree of closure) at the third interpolation point [%] (0...100).
fSunElv4	REAL	Position of the sun at the fourth interpolation point [°] (0...90).
fPos4	REAL	Blind position (degree of closure) at the fourth interpolation point [%] (0...100).

Outputs

```
VAR_OUTPUT
  stBldPosTab : ST_BA_BldPosTab;
  bErr        : BOOL;
  sErrDescr   : T_MAXSTRING;
END_VAR
```

Name	Type	Description
stBldPosTab	ST_BA_BldPosTab [► 216]	Transfer structure of the interpolation points
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Error: The x-values (elevation values) in the table are either not listed in ascending order, or they are duplicated.

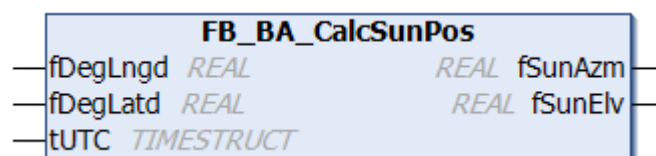
02: Error: An elevation value that was entered is outside the valid range of 0°...90°.

03: Error: A position value that was entered is outside the valid range of 0%...100%.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

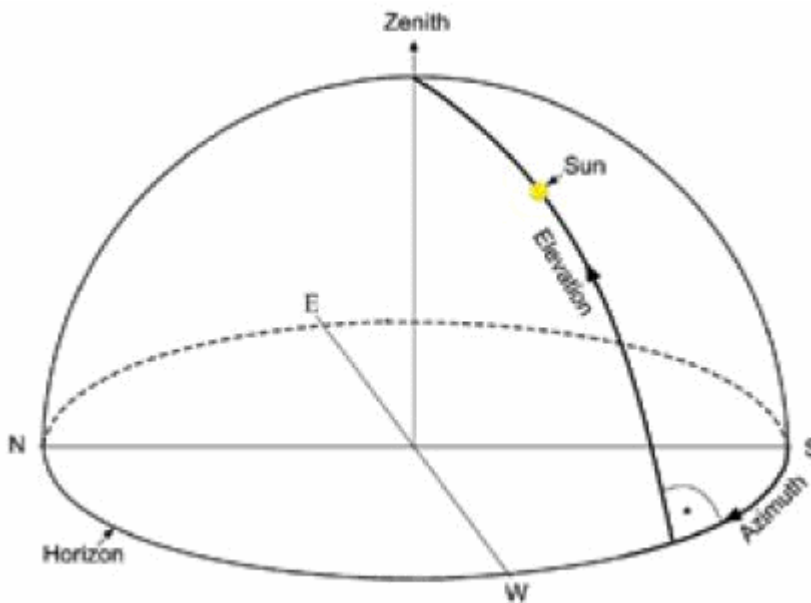
6.1.2.2.3.1.3.3.8 FB_BA_CalcSunPos



The function block FB_BA_CalcSunPos is used to calculate the position of the sun by specifying the date, time, longitude and latitude.

The position of the sun for a given point in time can be calculated according to common methods with a defined accuracy. The present function block is sufficient for applications with moderate requirements. As the basis for this, the SUNAE algorithm was used, which represents a favorable compromise between accuracy and computing effort.

The position of the sun at a fixed observation point is normally determined by specifying two angles. One angle indicates the height above the horizon; 0° means that the sun is in the horizontal plane of the location; a value of 90° means that the sun is perpendicular to the observer. The other angle indicates the direction at which the sun is positioned. The SUNAE algorithm is used to distinguish whether the observer is standing on the northern hemisphere (longitude > 0 degrees) or on the southern hemisphere (longitude < 0 degrees) of the earth. If the observation point is in the northern hemisphere, then a value of 0° is assigned for the northern sun direction and it then runs in the clockwise direction around the compass, i.e. 90° is east, 180° is south, 270° is west etc. If the point of observation is in the southern hemisphere, then 0° corresponds to the southern direction and it then runs in the counter clockwise direction, i.e. 90° is east, 180° is north, 270° is west etc.



The time has to be specified as UTC, Universal Time Coordinated (previously referred to as GMT, Greenwich Mean Time).

The latitude is the northerly or southerly distance of a location on the Earth's surface from the equator, in degrees $[\circ]$. The latitude can assume values between 0° (at the equator) and $\pm 90^\circ$ (at the poles). A positive sign thereby indicates a northern direction and a negative sign a southern direction. The longitude is an angle that can assume values up to $\pm 180^\circ$ starting from the prime meridian 0° (an artificially determined North-South line). A positive sign indicates a longitude in an eastern direction and a negative sign in a western direction. Examples:

Location	Longitude	Latitude
Sydney, Australia	151.2°	-33.9°
New York, USA	-74.0°	40.7°
London, England	-0.1°	51.5°
Moscow, Russia	37.6°	55.7°
Beijing, China	116.3°	39.9°
Dubai, United Arab Emirates	55.3°	25.4°
Rio de Janeiro, Brazil	-43.2°	-22.9°
Hawaii, USA	-155.8°	20.2°
Verl, Germany	8.5°	51.9°

If the function block *FB_BA_CalcSunPos* returns a negative value for the sun elevation *fSunElv*, the sun is invisible. This can be used to determine sunrise and sunset.

Inputs

```
VAR_INPUT
    fDegLngd    : REAL;
    fDegLatd    : REAL;
    tUTC        : TIMESTRUCT;
END_VAR
```

Name	Type	Description
fDegLngd	REAL	Longitude [°]
fDegLatd	REAL	Latitude [°]
tUTC	TIMESTRUCT	Current time as Coordinated Universal Time. The function block FB_BA_GetTime [► 335] can be used to read this time from a target system.

Outputs

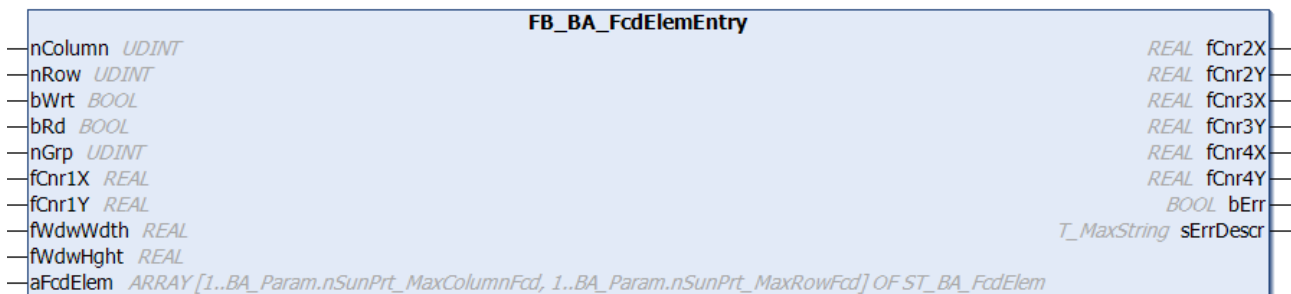
```
VAR_OUTPUT
    fSunAzm     : REAL;
    fSunElv     : REAL;
END_VAR
```

Name	Type	Description
fSunAzm	REAL	Sun direction (northern hemisphere: 0° north ... 90° east ... 180° south ... 270° west ... / southern hemisphere: 0° south ... 90° east ... 180° north ... 270° west ...).
fSunElv	REAL	Sun elevation (0° horizontal ... 90° perpendicular).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.9 FB_BA_FcdElemEntry



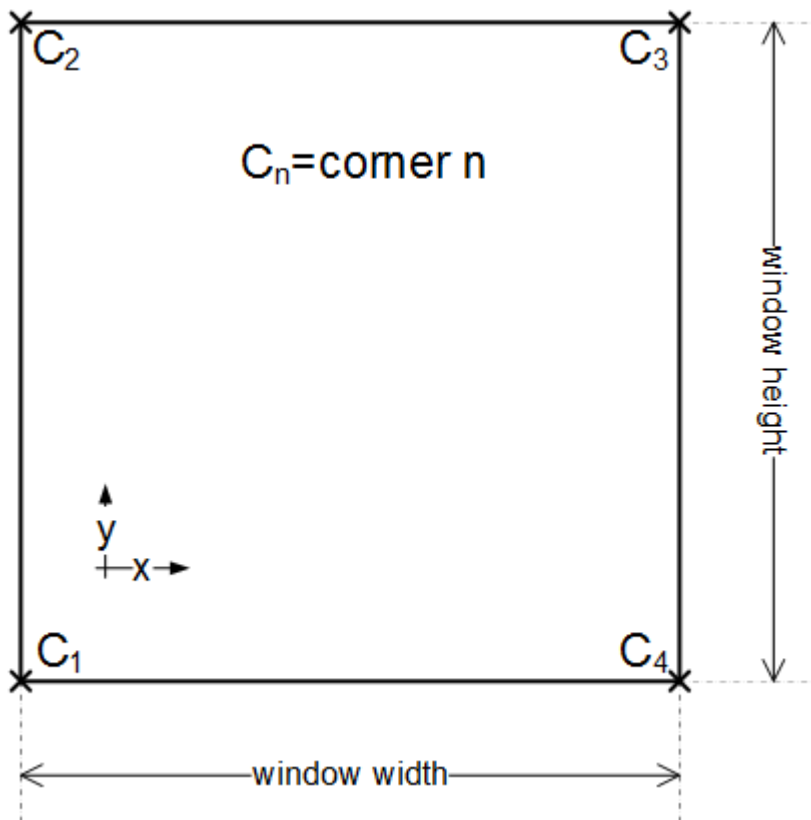
The function block **FB_BA_FcdElemEntry** is used to manage all facade elements (windows) of a facade, which is globally stored in a [List of facade elements](#) [► 274]. It is intended to facilitate inputting element information - not least with regard to using the TC3 PLC HMI. A schematic representation of the objects with description of the coordinates is shown in [Shading correction: principles and definitions](#) [► 261].

The facade elements are declared in the global variables as a two-dimensional field above the window columns and rows:

```
VAR_GLOBAL
    aFcdElem : ARRAY[1..Param.nSunPrt_MaxColumnFcd, 1..Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Each individual element *arrFcdElem[x,y]* carries the information for one facade element ([ST_BA_FcdElem](#) [► 217]). The information includes the group assignment, the dimensions (width, height) and the coordinates of the corners. The function block thereby accesses this field directly via the IN-OUT variable *aFcdElem*.

Note: The fact that the coordinates of corners C2 to C4 are output values arises from the fact that they are formed from the input parameters and are to be available for use in a visualization:



All data in [m]!

$fCnr2X = fCnr1X$
 $fCnr2Y = fCnr1Y + fWdwHght$ (window height)
 $fCnr3X = fCnr1X + fWdwWdth$ (window width)
 $fCnr3Y = fCnr2Y$
 $fCnr4X = fCnr1X + fWdwWdth$ (window width)
 $fCnr4Y = fCnr1Y$

The function block is used in three steps:

- Read
- Change
- Write

Read

With the entries at $nColumn$ and $nRow$ the corresponding element is selected from the list, $aFcdElem[nColumn, nRow]$. A rising edge on bRd reads the following data from the list element:

- nGrp group membership,
- $fCnr1X$ x-coordinate of corner point 1 [m]
- $fCnr1Y$ y-coordinate of corner point 1 [m]
- $fWdwWdth$ window width [m]
- $fWdwHght$ window height [m]

These are then assigned to the corresponding input variables of the function block, which uses them to calculate the coordinates of corners C_2 - C_4 as output variables in accordance with the correlation described above. It is important here that the input values are not overwritten in the reading step. Hence, all values can initially be displayed in a visualization.

Change

In a next program step the listed input values can then be changed. The values entered are constantly checked for plausibility. The output *bErr* indicates whether the values are valid (*bErr*=FALSE). If the values are invalid, a corresponding error message is issued at output *sErrDescr*. See also "Error (*bErr*=TRUE)" below.

Write

With a positive edge at *bWrt* the parameterized data are written into the field of the array *aFcdElem* dependent on *nRow* and *nColumn*, regardless of whether they represent valid values or not. The element structure ST_BA_FcdElem [► 217] therefore also contains a plausibility bit *bVld*, which forwards precisely this information to the function block FB_BA_Shdcorr [► 299] to prevent miscalculations.

This approach is to be regarded only as a proposal. It is naturally also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on *bWrt*.

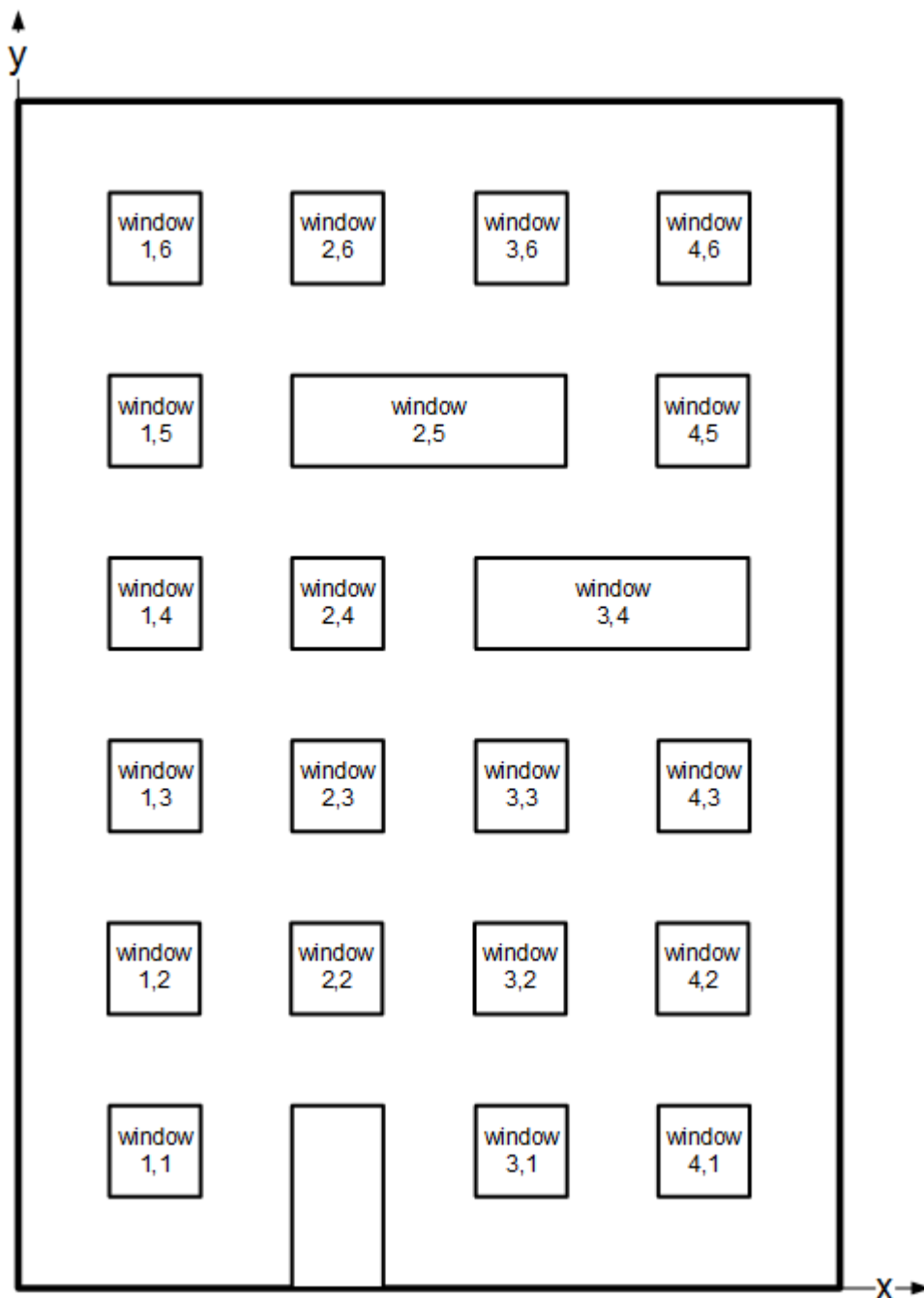
Error (*bErr*=TRUE)

The function block FB_BA_Shdcorr [► 299], which assesses whether all windows in a group are shaded, will only perform its task if all windows in the examined group have valid entries. This means:

- *nGrp* must be greater than 0
- *fCnr1X* must be greater than or equal to 0.0
- *fCnr1Y* must be greater than or equal to 0.0
- *fWdwWidth* must be greater than 0
- *fWdwHght* must be greater than 0

If one of these criteria is not met, it is interpreted as incorrect input, and the error output *bErr* is set at the function block output of FB_BA_FcdElemEntry. Within the window element ST_BA_FcdElem [► 217] the plausibility bit *bVld* is set to FALSE.

If, on the other hand, **all** entries of a facade element are zero, it is regarded as a valid, intentionally omitted facade element:



In the case of a facade of 6x4 windows, the elements window (2.1), window (3.5) and window (4.4) would be empty elements here.

Inputs

```
VAR_INPUT
  nColumn   : UDINT;
  nRow      : UDINT;
  bWrt      : BOOL;
  bRd       : BOOL;
  nGrp      : UDINT;
  fCnr1X    : REAL;
  fCnr1Y    : REAL;
  fWdwWdth  : REAL;
  fWdwHght  : REAL;
END_VAR
```

Name	Type	Description
nColumn	UDINT	Column index of the selected component on the facade. This refers to the selection of a field element of the array stored in the IN-OUT variable <i>aFcdElem</i> .
nRow	UDINT	ditto row index. <i>nRow</i> and <i>nColumn</i> must not be zero! This is due to the field definition, which always starts with 1; see above.
bRd	BOOL	With a positive edge at this input, the information of the selected element, <i>aFcdElem[nColumn, nRow]</i> is read into the function block and assigned to the input variables <i>nGrp</i> to <i>fWdwHght</i> . The resulting output variables are <i>fCnr2X</i> to <i>fCnr4Y</i> . If data are already present on the inputs <i>nGrp</i> to <i>fWdwHght</i> at time of reading, then the data previously read are immediately overwritten with these data.
bWrt	BOOL	A positive edge writes the entered as well as calculated values into the selected field element <i>aFcdElem[nColumn, nRow]</i> .
nGrp	UDINT	Group membership. Internally limited to a minimum value of 0.
fCnr1X	REAL	X-coordinate of corner point 1 [m]
fCnr1Y	REAL	Y-coordinate of corner point 1 [m]
fWdwWdth	REAL	Window width [m]
fWdwHght	REAL	Window height [m]

Outputs

```
VAR_OUTPUT
  fCnr2X   : REAL;
  fCnr2Y   : REAL;
  fCnr3X   : REAL;
  fCnr3Y   : REAL;
  fCnr4X   : REAL;
  fCnr4Y   : REAL;
  bErr     : BOOL;
  sErrDesc : T_MAXSTRING;
END_VAR
```

Name	Type	Description
fCnr2X	REAL	X-coordinate determined for corner point 2 of the window [m] (see function description [► 278])
fCnr2Y	REAL	Y-coordinate determined for corner point 2 of the window [m] (see function description [► 278])
fCnr3X	REAL	X-coordinate determined for corner point 3 of the window [m] (see function description [► 278])
fCnr3Y	REAL	Y-coordinate determined for corner point 3 of the window [m] (see function description [► 278])
fCnr4X	REAL	X-coordinate determined for corner point 4 of the window [m] (see function description [► 278])
fCnr4Y	REAL	Y-coordinate determined for corner point 4 of the window [m] (see function description [► 278])
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description
01: Error: Index error! nColumn and/or nRow are outside the permissible limits 1... nSunPrt_MaxColumnFcd or 1... nSunPrt_MaxRowFcd. See list of facade elements.
02: Error: The group index is 0, but at the same time another entry of the facade element is not zero. Only if all entries of a facade element are zero is it considered to be a valid, deliberately omitted facade component, otherwise it is interpreted as an incorrect entry. Note: Group entries less than zero are internally limited to zero.
03: Error: The X-component of the first corner point (Corner1) is less than zero.
04: Error: The Y-component of the first corner point (Corner1) is less than zero.
05: Error: The window width is less than or equal to zero.
06: Error: The window height is less than or equal to zero.

 / Inputs/Outputs

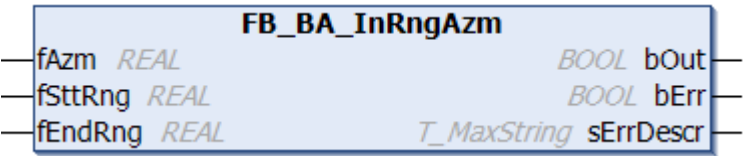
```
VAR_IN_OUT
aFcdElem : ARRAY[1..Param.nSunPrt_MaxColumnFcd, 1..Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Name	Type	Description
aFcdElem	ST_BA_FcdElem	List of facade elements ▶ 274

Requirements

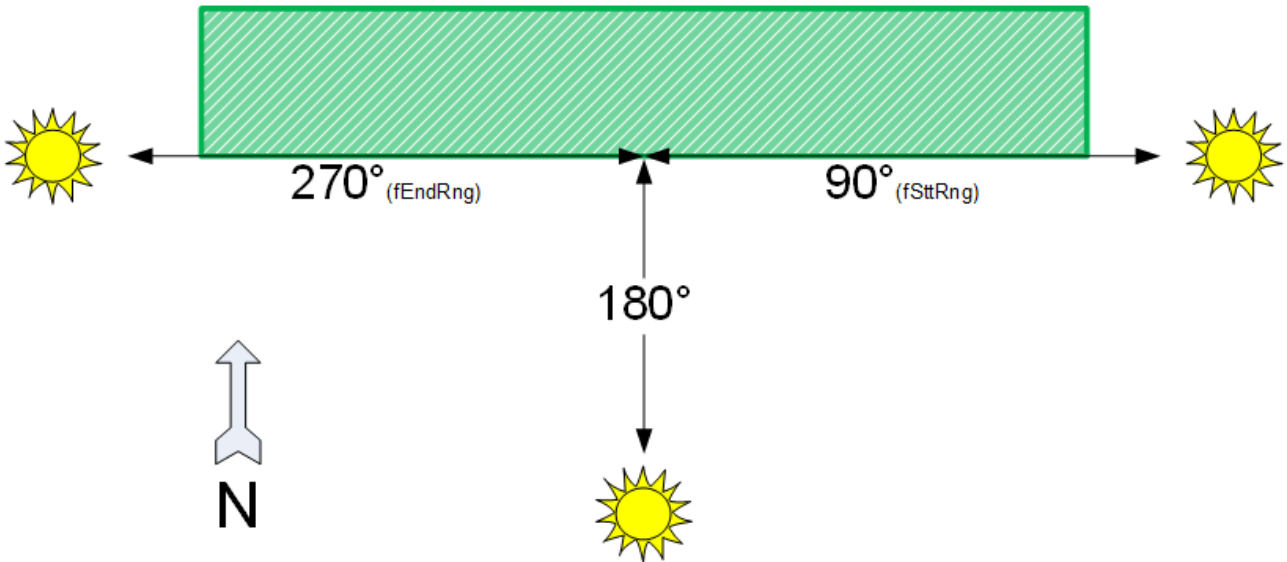
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.10 FB_BA_InRngAzm

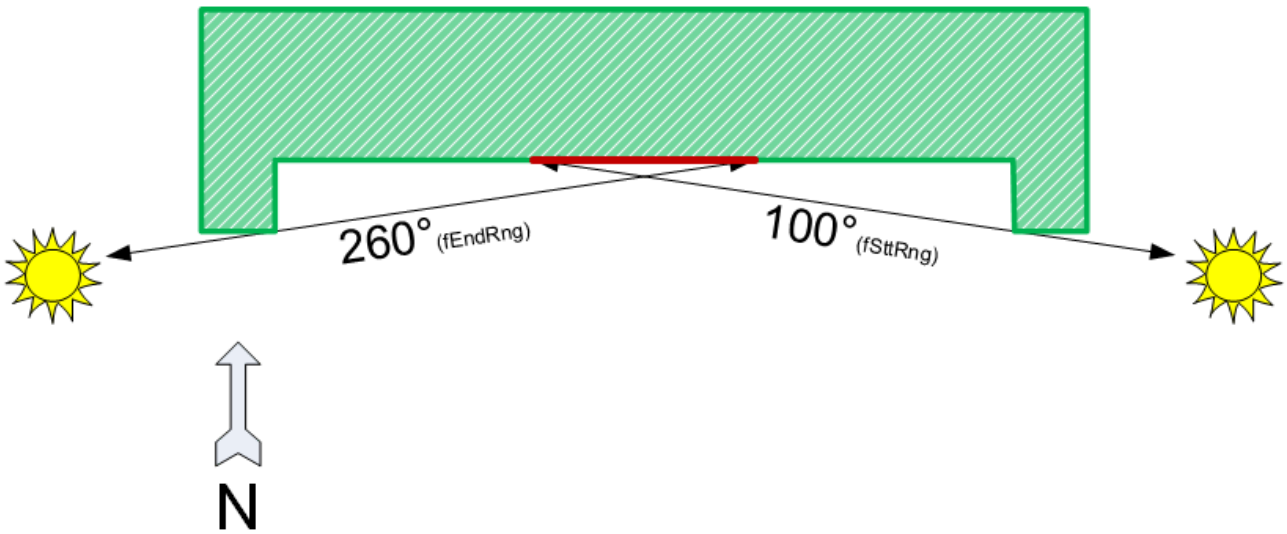


This function block FB_BA_InRngAzm checks whether the current azimuth angle (horizontal position of the sun) lies within the limits entered. As can be seen in the [overview \[▶ 268\]\(#\)](#), the function block provides an additionally evaluation as to whether the sun protection of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

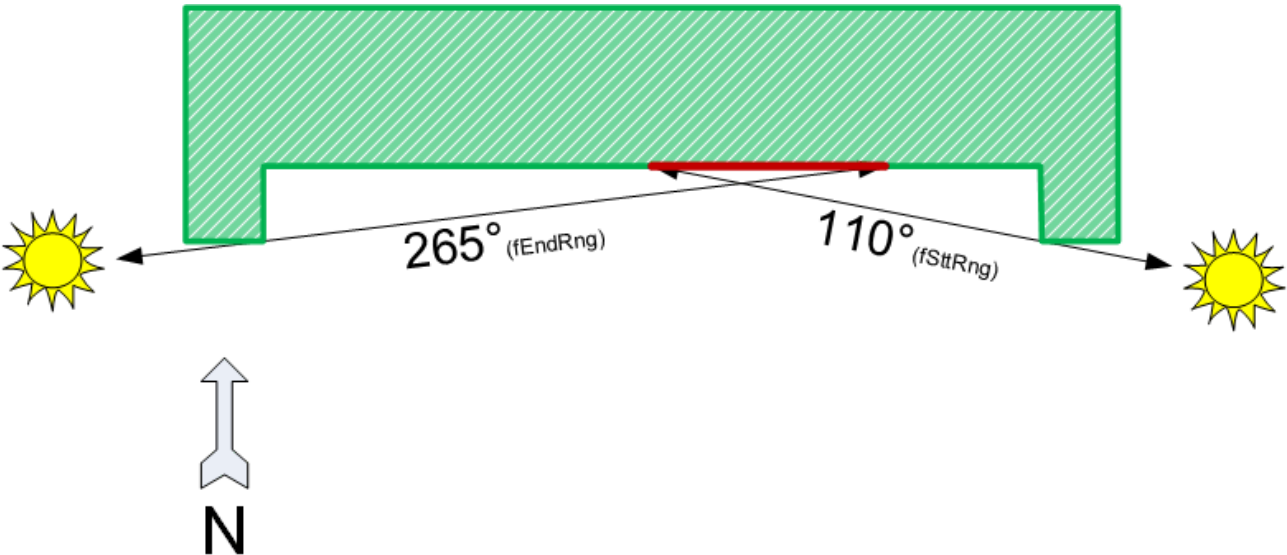
The sun incidence azimuth angle on a smooth facade will always be *facade orientation-90°... facade orientation+90°*.



If the facade has lateral projections, however, this range is limited. This limitation can be checked with the help of this function block. However, the position of the window group on the facade also plays a role. If it lies centrally, this gives rise to the following situation (the values are only examples):



The values change for a group at the edge:



The beginning of the range *fSttRng* may thereby be larger than the end *fEndRng*; it is then regarded beyond 0° :

Example

fAzm	10.0°
fSttRng	280.0°
fEndRng	20.0°
bOut	TRUE

However, the range regarded may not be greater than 180° or equal to 0° – this would be unrealistic. Such entries result in an error on the output *bErr* – the test output *bOut* is then additionally set to FALSE.

Inputs

```
VAR_INPUT
    fAzm      : REAL;
    fSttRng   : REAL;
    fEndRng   : REAL;
END_VAR
```

Name	Type	Description
fAzm	REAL	Current azimuth angle
fSttRng	REAL	Start of range [°]
fEndRng	REAL	End of range [°].

Outputs

```
VAR_OUTPUT
    bOut      : BOOL;
    bErr      : BOOL;
    sErrDescr : T_MAXSTRING;
END_VAR
```

Name	Type	Description
bOut	BOOL	The facade element is in the sun if the output is TRUE.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

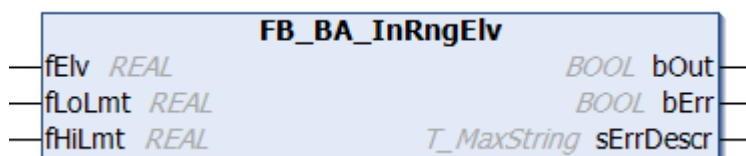
01: Error: *fSttRng* or *fEndRng* less than 0° or greater than 360°.

02: Error: The difference between *fSttRng* and *fEndRng* is greater than 180°. This range is too large for analyzing the insolation on a facade.

Requirements

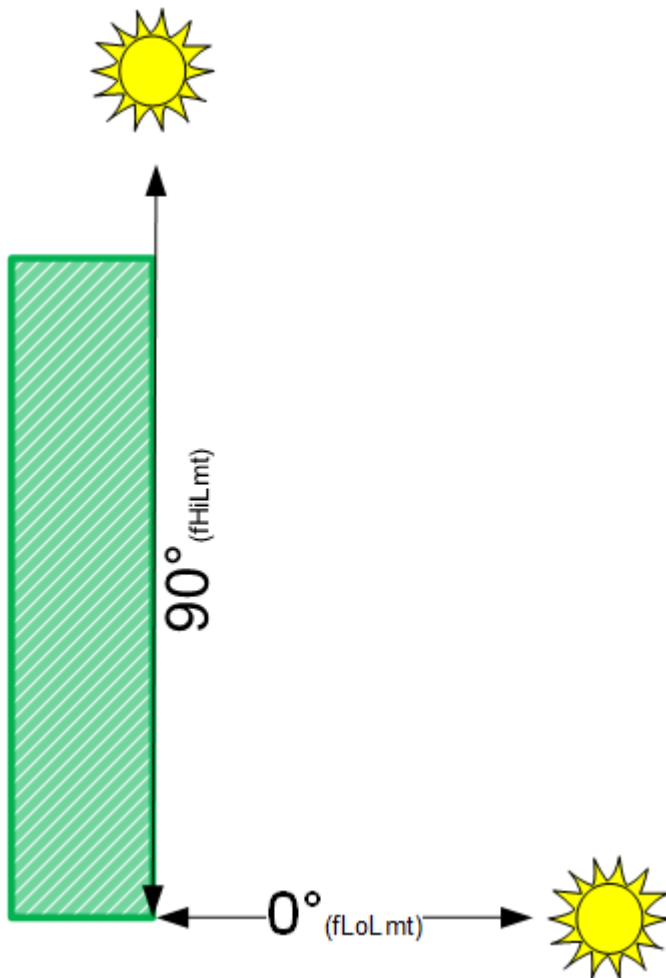
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.11 FB_BA_InRngElv

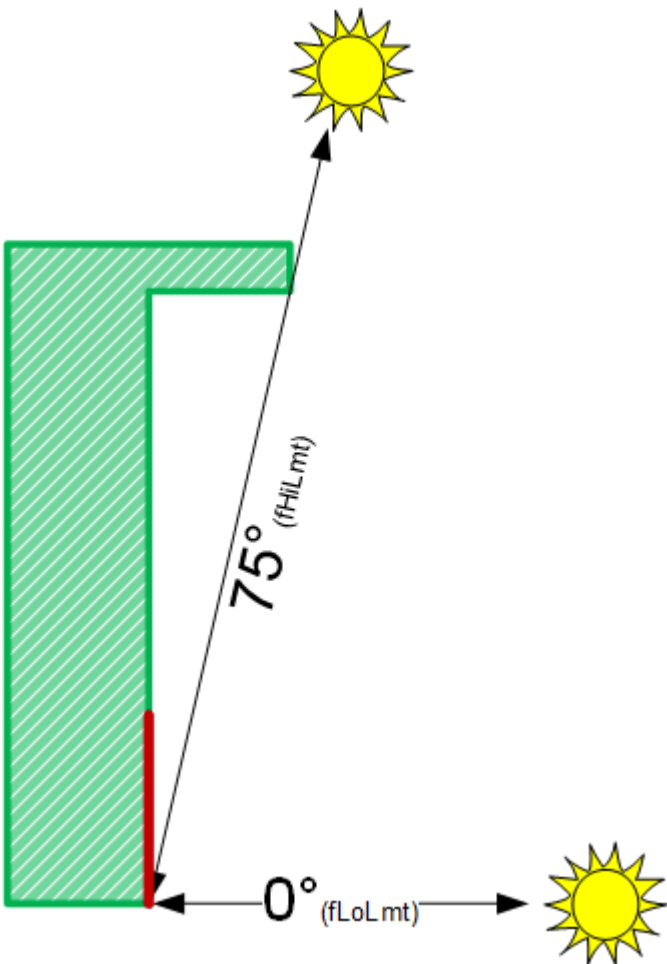


This function block FB_BA_InRngElv checks whether the current elevation angle (vertical position of the sun) lies within the limits entered. As can be seen in the [overview \[► 268\]](#), the function block provides an additionally evaluation as to whether the sun protection of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

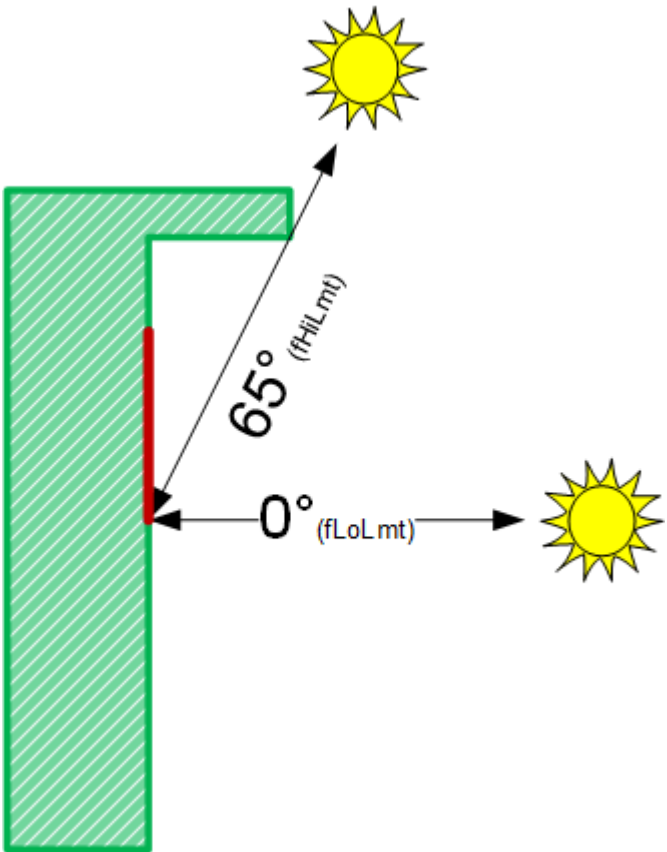
A normal vertical facade is irradiated by the sun at an angle of elevation of 0° to maximally 90°.



If the facade has projections, however, this range is limited. This limitation can be checked with the help of this function block. However, the position of the window group on the facade also plays a role. If it lies in the lower range, this gives rise to the following situation (the values are only examples):



The values change for a group below the projection:



The lower observation limit, *fLoLmt*, may thereby not be greater than or equal to the upper limit, *fHiLmt*. Such entries result in an error on the output *bErr* – the test output *bOut* is then additionally set to FALSE.

Inputs

```
VAR_INPUT
    fElv      : REAL;
    fLoLmt    : REAL;
    fHiLmt    : REAL;
END_VAR
```

Name	Type	Description
fElv	REAL	Current elevation angle
fLoLmt	REAL	Lower limit value [°]
fHiLmt	REAL	Upper limit value [°]

Outputs

```
VAR_OUTPUT
    bOut      : BOOL;
    bErr      : BOOL;
    sErrDescr : T_MAXSTRING;
END_VAR
```

Name	Type	Description
bOut	BOOL	The facade element is in the sun if the output is TRUE.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Error: *fHiLmt* less than or equal to *fLoLmt*.

02: Error: *fLoLmt* is less than 0° or *fHiLmt* is greater than 90°.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.12 FB_BA_RdFcdElemLst

FB_BA_RdFcdElemLst			
— bStt	BOOL		BOOL bBusy
— sDataFile	STRING		UDINT nAmtSetsRd
— sLogFile	STRING		BOOL bErr
— tNetId	T_AmsNetId		T_MaxString sErrDescr
— aFcdElem	ARRAY[1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem		BOOL bErrDataSet

With the help of the function block FB_BA_RdFcdElemLst, data for facade elements (windows) can be imported from a pre-defined Excel table in csv format into the List of facade elements [► 274]. In addition the imported data are checked for plausibility and errors are written to a log file.

The following example shows the Excel table with the entries of the window elements.

All text fields are freely writable. Important are the fields marked in green. Each row there identifies a data set.

The following rules are to be observed:

- A data set must always start with a '@'.
- The indices *IndexColumn* and *IndexRow* must lie within the defined limits, see List of facade elements [► 274]. These indices directly describe the facade element in the list *aFcdElem* to which the data from the set are saved.
- Window width and window height must be greater than zero

- The corner coordinates P1x and P1y must be greater than or equal to zero.
- Each window element must be assigned to a group 1...255.
- For system-related reasons the total size of the table may not exceed 65534 bytes.
- This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)".

It is not necessary to describe all window elements that would be possible by definition or declaration. Before the new list is read in, the function block deletes the entire old list in the program. All elements that are not described by entries in the Excel table then have pure zero entries and are thus marked as non-existent and also non-evaluable, since the function block for shading correction, FB_BA_ShdCorr [► 299], does not accept elements with the group entry '0'.

EN_FacadeElements.xls										
	A	B	C	D	E	F	G	H	I	J
1	Number	Description		IndexColumn	IndexRow	Window-Width	Window-Height	P1x	P1y	Group
2				(Axis)	(Floor)	[m]	[m]	[m]	[m]	
3		Text								
4	1	Description	@	1	1	1,2	1,3	1,5	1	2
5	2	Description	@	0	1	1,2	1,3	2,7	1	2
6	3	Description	@	3	1	1,2	1,3	4,4	1	2
7	4	Description	@	4	1	1,2	1,3	6,1	1	2
8	5	Description	@	5	1	1,2	1,3	7,8	1	2
9	6	Description	@	6	1	1,2	1,3	9,5	1	2
10	7	Description	@	7	1	1,2	1,3	11,2	1	2
11	8	Description	@	8	1	1,2	1,3	12,9	1	2
12	9	Description	@	9	1	1,2	1,3	14,6	1	2
13	10	Description	@	10	1	1,2	1,3	16,3	1	2
14	11	Description	@	1	1	1,2	1,3	1,5	4	3
15	12	Description	@	0	1	1,2	1,3	2,7	4	3
16	13	Description	@	3	1	1,2	1,3	4,4	4	3
17	14	Description	@	4	1	1,2	1,3	6,1	4	3
18	15	Description	@	5	1	1,2	1,3	7,8	4	3
19	16	Description	@	6	1	1,2	1,3	9,5	4	3
20	17	Description	@	7	1	1,2	1,3	11,2	4	3
21	18	Description	@	8	1	1,2	1,3	12,9	4	3
22	19	Description	@	9	1	1,2	1,3	14,6	4	3
23	20	Description	@	10	1	1,2	1,3	16,3	4	3
24	21	Description	@	1	1	1,2	1,3	1,5	7	4
25	22	Description	@	0	1	1,2	1,3	2,7	7	4
26	23	Description	@	3	1	1,2	1,3	4,4	7	4
27	24	Description	@	4	1	1,2	1,3	6,1	7	4
28	25	Description	@	5	1	1,2	1,3	7,8	7	4
29	26	Description	@	6	1	1,2	1,3	9,5	7	4
30	27	Description	@	7	1	1,2	1,3	11,2	7	4
31	28	Description	@	8	1	1,2	1,3	12,9	7	4
32	29	Description	@	9	1	1,2	1,3	14,6	7	4
33	30	Description	@	10	1	1,2	1,3	16,3	7	4
34	31	Description	@	1	1	1,2	1,3	1,5	10	5
35	32	Description	@	0	1	1,2	1,3	2,7	10	5
36	33	Description	@	3	1	1,2	1,3	4,4	10	5
37	34	Description	@	4	1	1,2	1,3	6,1	10	5
38	35	Description	@	5	1	1,2	1,3	7,8	10	5
39	36	Description	@	6	1	1,2	1,3	9,5	10	5
40	37	Description	@	7	1	1,2	1,3	11,2	10	5
41	38	Description	@	8	1	1,2	1,3	12,9	10	5
42	39	Description	@	9	1	1,2	1,3	14,6	10	5
43	40	Description	@	10	1	1,2	1,3	16,3	10	5
44										

Log file

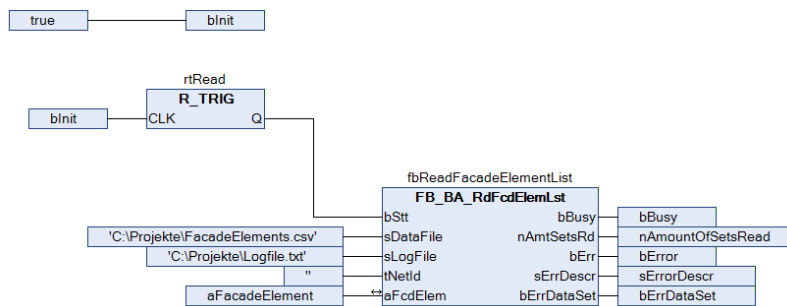
Each time the reading function block is restarted, the log file is rewritten and the old contents are deleted. If there is no log file, it will be automatically created first. The log file then contains either an OK message or a list of all errors that have occurred. Errors connected with the opening, writing or closing of the log file itself cannot be written at the same time. Therefore, always note the output *sErrDescr* of the reading function block that indicates the last error code. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.

Program sample

```

1 PROGRAM ReadFacadeElements
2 VAR
3     bInit          : BOOL;
4     rtRead         : R_TRIG;
5     fbReadFacadeElementList : FB_BA_RdFcdElemLst;
6     aFacadeElement : ARRAY [1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;
7
8     bBusy          : BOOL;
9     nAmountOfSetsRead : UDINT;
10    bError         : BOOL;
11    sErrorDescr    : T_MaxString;
12    bErrDataSet    : BOOL;
13 END_VAR

```



In this sample the variable *bInit* is initially set to TRUE when the PLC starts. Hence, the input *bStt* on the function block *fbReadFacadeElementList* receives a once-only rising edge that triggers the reading process. The file *FacadeElements.csv* is read, which is located in the folder *C:\Projekte*. The log file "Logfile.txt" is then saved in the same folder. If this log file does not yet exist it will be created, otherwise the existing contents are overwritten. Reading and writing take place on the same computer on which the PLC is located. This is defined by the input *tNetId* = " (=local). All data are written to the list *aFcdElem* declared in the program. The output *bBusy* is set to TRUE during reading and writing. The last file handling error that occurred is displayed at *sErrDescr*; *bErr* is then TRUE. If an error is detected in the data set, this is displayed at *bErrDataSet* and described in more detail in the log file. The number of found and read data rows is displayed at *nAmtSetsRd* for verification purposes.

The errors marked were "built into" the following Excel list. This gives rise to the log file shown:

	A	B	C	D	E	F	G	H	I	J	K
1	Number	Description		IndexColumn (Axis)	IndexRow (Floor)	Window wid [m]	Window hei [m]	P1x [m]	P1y [m]	Group	
2											
3		Text									
4	1	Description @		1	1	1,2	1,3	1,5	1,4	2	
5	2	Description @		0	1	1,2	1,3	2,7	1	2	
6	3	Description @		3	1	1,2	1,3	4,4	1	2	
7	4	Description @		4	-1	1,2	1,3	6,1	1	2	
8	5	Description @		5	1	1,2	1,3	7,8	1	2	
9	6	Description @		6	1	0	1,3	9,5	1	2	
10	7	Description @		7	1	1,2	1,3	11,2	1	2	
11	8	Description @		8	1	1,2	1,3	12,9	1	2	
12	9	Description @		9	1	1,2	1,3	14,6	1	2	
13	10	Description @		10	1	1,2	1,3	16,3	1	5	
14	11	Description @		1	2	1,2	1,3	1	-1	5	
15	12	Description @		2	2	1,2	1,3	2,7	3	5	
16	13	Description @		3	2	1,2	1,3	4,4	4	5	
17	14	Description @		4	2	1,2	1,3	4,4	4	5	
18	15	Description @		5	2	1,2	0	7,8	4	5	
19	16	Description @		6	2	1,2	1,3	9,5	4	5	
20	17	Description @		7	2	1,2	1,3	11,2	4	5	
21	18	Description @		8	2	1,2	1,3	12,9	4	5	
22	19	Description @		9	2	1,2	1,3	14,6	4	3	
23	20	Description @		10	2	1,2	1,3	16,3	4	3	
24											
25	31	Description @		1	3	1,2	1,3	1	7	3	
26	32	Description @		2	3	1,2	1,3	-1	6	3	
27	33	Description @		3	3	1,2	1,3	4,4	7	3	
28	34	Description @		4	3	1,2	1,3	6,1	7	0	
29	35	Description @		5	3	1,2	1,3	7,8	7	3	
30	36	Description @		6	3	1,2	1,3	9,5	7	3	
31	37	Description @		7	3	1,2	1,3	11,2	7	3	
32	38	Description @		8	3	1,2	1,3	12,9	7	7	
33	39	Description @		9	3	1,2	1,3	14,6	7	7	
34	40	Description @		10	3	1,2	1,3	16,3	7	7	

LogFacade.txt - Editor

Datei Bearbeiten Format Ansicht ?

Index-Error in Data-Set #2

Index-Error in Data-Set #4

Validation-Error in Data-Set #6, Errordescription:05: Error: The window width is less than or equal to zero.

Validation-Error in Data-Set #11, Errordescription:04: Error: The Y-component of the first corner (Corner1) is less than zero.

Validation-Error in Data-Set #15, Errordescription:06: Error: The window height is less than or equal to zero.

Validation-Error in Data-Set #22, Errordescription:03: Error: The X-component of the first corner (Corner1) is less than zero.

Validation-Error in Data-Set #24, Errordescription:02: Error: The group index is 0, but at the same time another entry of the facade element is not zero. See manual for this FB.

The first error is in data set 2 and is an index error, since "0" is not permitted.

The next error in data set 6 was found after validation of the data with the internally used function block **FB BA_ShdObjEntry** [► 303] and allocated an error description. The third and the fourth errors likewise occurred after the internal validation.



Important here it that the data set numbers (in this case 22 and 24) do not go by the numbers entered in the list, but by the actual sequential numbers: only 30 data sets were read in here.

Inputs

```
VAR_INPUT
  bStt      : BOOL;
  sDataFile : STRING;
  sLogFile  : STRING;
  tNetId    : T_AmsNetId;
END_VAR
```


Name	Type	Description
bStt	BOOL	A TRUE edge on this input starts the reading process.
sDataFile	SRING	Contains the path and file name for the data file to be opened. This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)". If the file is opened with a simple text editor, the values must be separated by semicolons. Example of an entry: <i>sDataFile:= 'C:\Projekte\FacadeElements.csv'</i>
sLogFile	STRING	ditto log file for the accumulating errors. This file is overwritten each time the function block is activated, so that only current errors are contained.
tNetId	T_AmsNetId	A string can be entered here with the AMS Net ID of the TwinCAT computer on which the files are to be written/read. If it is to be run on the local computer, an empty string can be entered.



The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

Inputs/Outputs

```
VAR_IN_OUT
  aFcdElem : ARRAY[1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem
;
END_VAR
```

Name	Type	Description
aFcdElem	ST_BA_FcdElem	List of facade elements [► 274]

Outputs

```
VAR_OUTPUT
  bBusy      : BOOL;
  nAmtSetsRd : UDINT;
  bErr       : BOOL;
  sErrDescr  : T_MAXSTRING;
  bErrDataSet : BOOL;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is TRUE as long as elements are being read from the file.
aAmtSetsRd	UDINT	Number of data sets read
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

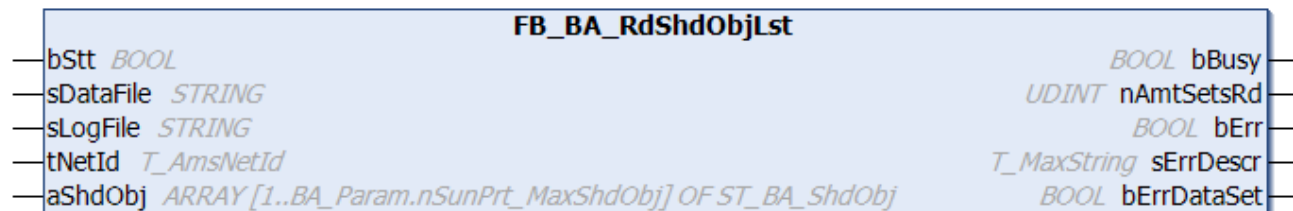
Error description
01: File handling error: Opening the log file - the ADS error number is stated.
02: File handling error: Opening the data file - the ADS error number is stated.
03: File handling error: Reading the data file - the ADS error number is stated.
04: Error: During reading of the data file it was determined that the file is too large (number of bytes larger than nMaxDataFileSize)
05: File handling error: Writing the log file - the ADS error number is stated.
06: File handling error: Closing the data file - the ADS error number is stated.
07: File handling error: Writing to the log file (OK message if no errors were detected) - the ADS error number is stated.
08: File handling error: Closing the log file - the ADS error number is stated.

Name	Type	Description
bErrDataSet	BOOL	This output is set to TRUE, if the read data sets are faulty. Further details are entered in the log file.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.13 FB_BA_RdShdObjLst



With the help of the function block FB_BA_RdShdObjLst, data for shading objects can be imported from a pre-defined Excel table in csv format into the [list of shading objects \[► 274\]](#). In addition the imported data are checked for plausibility and errors are written to a log file.

The following example shows the Excel table with the entries of the window elements.

All text fields are freely writable. The fields marked in green are important; each line in these fields identifies a data set. The columns G to J have a different meaning depending on whether the type rectangle or sphere is concerned. The columns K to M are to be left empty in the case of spheres. With regard to the rectangle coordinates, only the relevant data are entered and the remainder are internally calculated (see [FB_BA_ShObjEntry \[► 303\]](#)).

The following rules are to be observed:

- A data set must always start with a '@'.
 - The month entries must not be 0 and not be greater than 12, all other combinations are possible.
- Examples:**
- Start=1, End=1: Shading in January.
 - Start=1, End=5: Shading from the beginning of January to the end of May.
 - Start=11, End=5: Shading from the beginning of November to the end of May (of the following year).
- Window width and window height must be greater than zero
 - The z-coordinates P1z and P3z or Mz must be greater than zero.
 - The radius must be greater than zero.
 - For system-related reasons the total size of the table may not exceed 65534 bytes.
 - This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)".

It is not necessary to describe all shading objects that are possible per facade. Only those contained in the list ultimately take effect.

DE_ShadingObjects.csv													
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Number	Description		Type	Begin	End	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Tetragon	(Month)	(Month)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Globe									
4		Text											
5	1	Description	@	0	1	2	-94,75	0	36,06	11	-70,71	11	68,59
6	2	Description	@	0	1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Description	@	0	1	2	62,23	0	0	14,47	62,23	14,47	8
8	4	Description	@	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Description	@	0	1	2	46	0	13	14,47	46	14,47	38,89
10	6	Description	@	0	1	2	0	0	14	9	35	9	14
11	7	Description	@	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Description	@	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Description	@	0	1	2	27,8	0	14	9,8	35	9,8	14
14													
15	10	Description	@	1	1	2	27	15	40	6			
16	11	Description	@	1	1	2	38	15	36	6			
17	12	Description	@	1	1	2	-14	4	4	1,5			
18	13	Description	@	1	1	2	-6,5	6	6	3,2			
19	14	Description	@	1	1	2	-7	9	6	1,2			
20	15	Description	@	1	1	2	-1	6	8	3,2			
21	16	Description	@	1	1	2	-1	9	8	1,2			

Log file

Each time the reading function block is restarted, the log file is rewritten and the old contents are deleted. If there is no log file, it will be automatically created first. The log file then contains either an OK message or a list of all errors that have occurred. Errors connected with the opening, writing or closing of the log file itself cannot be written at the same time. Therefore, always note the output *sErrDescr* of the reading function block that indicates the last error code. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.

Program sample

```
PROGRAM ReadShadingObjects
```

```
VAR
```

```

    bInit          :   BOOL;
    rtRead         :   R_TRIG;
    fbReadShadingObjects :   FB_BA_RdShdObjLst;
    aShadingObject :   ARRAY [1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShObj;

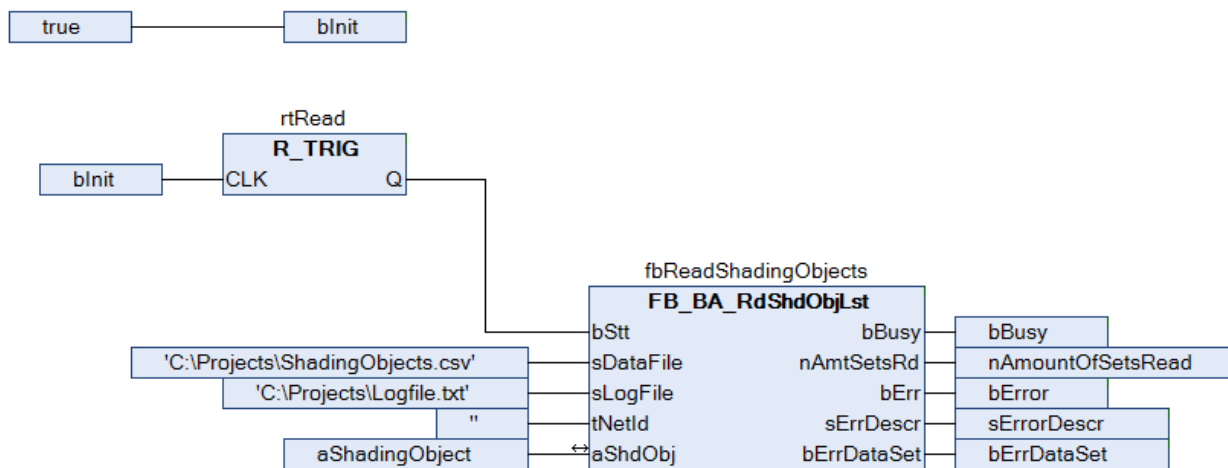
```

```

    bBusy          :   BOOL;
    nAmountOfSetsRead :   UDINT;
    bError         :   BOOL;
    sErrorDescr    :   T_MaxString;
    bErrDataSet    :   BOOL;

```

```
END_VAR
```



In this sample the variable *bInit* is initially set to TRUE when the PLC starts. Hence, the input *bStt* on the function block *fbReadShadingObjects* receives a once-only rising edge that triggers the reading process. The file *ShadingObjects.csv* is read, which is located in the folder *C:\Projekte*. The log file *Logfile.txt* is then saved in the same folder. If this log file does not yet exist it will be created, otherwise the existing contents are overwritten. Reading and writing take place on the same computer on which the PLC is located. This is defined by the input *tNetID* = "" (=local). All data are written to the list *aShdObj* declared in the program. The output *bBusy* is set to TRUE during reading and writing. The last file handling error that occurred is displayed at *sErrDescr*; *bErr* is then TRUE. If an error is detected in the data set, this is displayed at *bErrDataSet* and described in more detail in the log file. The number of found and read data rows is displayed at *nAmtSetsRd* for verification purposes.

The errors marked were built into the following Excel list. This gives rise to the log file shown:

A	B	C	D	E	F	G	H	I	J	K	L	M
1	Number	Description	Type	Begin	End	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2			0 - Tetragon	(Month)	(Month)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3			1 - Globe									
4		Text										
5	1	Description @	2	1	2	-94,75	-4	36,06	11	-70,71	11	68,59
6	2	Description @	0	-1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Description @	0	1	13	62,23	0	0	14,47	62,23	14,47	8
8	4	Description @	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Description @	0	1	2	46	0	-13	14,47	46	14,47	38,89
10	6	Description @	0	1	2	0	0	14	9	35	9	-14
11	7	Description @	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Description @	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Description @	0	1	2	27,8	0	14	9,8	35	9,8	14
14												
15	11	Description @	1	1	2	27	15	-40	6			
16	12	Description @	1	1	2	38	15	36	6			
17	13	Description @	1	1	2	-14	4	4	1,5			
18	14	Description @	1	1	2	-6,5	6	6	3,2			
19	15	Description @	1	1	2	-7	9	6	1,2			
20	16	Description @	1	1	2	-1	6	8	3,2			
21	17	Description @	1	1	2	-1	9	8	1,2			

LogShading.txt - Editor
 Datei Bearbeiten Format Ansicht ?
 Type-Error in Data-Set #1
 Validation-Error in Data-Set #2, Errordescription:15: Error: Month input error.
 Validation-Error in Data-Set #3, Errordescription:15: Error: Month input error.
 Validation-Error in Data-Set #5, Errordescription:04: Error: The z-component of P1 is less than zero. This corner would thus lie behind the facade.
 Validation-Error in Data-Set #6, Errordescription:05: Error: The z-component of P3 is less than zero. This corner would thus lie behind the facade.
 Validation-Error in Data-Set #10, Errordescription:13: Error: The z-component of the ball centre is less than zero. This point would thus lie behind the facade.

The first error is in data set 3 and is a type error, since "2" is not defined.

The next error in data set 6 was found after validation of the data with the internally used function block `FB_BA_ShdObjEntry` [303] and allocated an error description. The third error likewise occurred after the internal validation.



Important here it that the data set number (in this case 11) does not go by the numbers entered in the list, but by the actual sequential number: only 16 data sets were read in here.

Inputs

```

VAR_INPUT
  bStt      : BOOL;
  sDataFile : STRING;
  sLogFile  : STRING;
  tNetId    : T_AmsNetId;
END_VAR

```

Name	Type	Description
bStt	BOOL	A TRUE edge on this input starts the reading process.
sDataFile	SRING	Contains the path and file name for the data file to be opened. This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)". If the file is opened with a simple text editor, the values must be separated by semicolons. Example of an entry: <code>sDataFile:= 'C:\Projekte\FacadeElements.csv'</code>
sLogFile	STRING	ditto log file for the accumulating errors. This file is overwritten each time the function block is activated, so that only current errors are contained.
tNetId	T_AmsNetId	A string can be entered here with the AMS Net ID of the TwinCAT computer on which the files are to be written/read. If it is to be run on the local computer, an empty string can be entered.



The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

Inputs/Outputs

```
VAR_IN_OUT
  aShdObj: ARRAY[1..BA_Param. nSunPrt_MaxShdObj] OF ST_BA_ShObj;
END_VAR
```

Name	Type	Description
aShdObj	ST_BA_ShObj	List of shading objects [▶ 274]

Outputs

```
VAR_OUTPUT
  bBusy      : BOOL;
  nAmtSetsRd : UDINT;
  bErr       : BOOL;
  sErrDescr  : T_MAXSTRING;
  bErrDataSet : BOOL;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is TRUE as long as elements are being read from the file.
aAmtSetsRd	UDINT	Number of data sets read
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

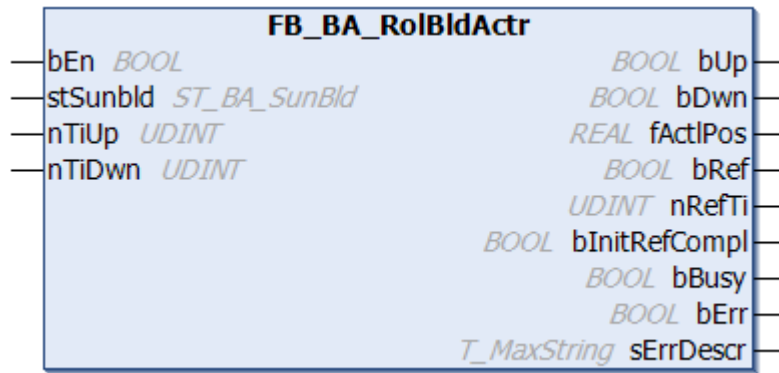
01: File handling error: Opening the log file - the ADS error number is stated.
02: File handling error: Opening the data file - the ADS error number is stated.
03: File handling error: Reading the data file - the ADS error number is stated.
04: Error: During reading of the data file it was determined that the file is too large (number of bytes larger than nMaxDataFileSize)
05: File handling error: Writing the log file - the ADS error number is stated.
06: File handling error: Closing the data file - the ADS error number is stated.
07: File handling error: Writing to the log file (OK message if no errors were detected) - the ADS error number is stated.
08: File handling error: Closing the log file - the ADS error number is stated.

Name	Type	Description
bErrDataSet	BOOL	This output is set to TRUE, if the read data sets are faulty. Further details are entered in the log file.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.14 FB_BA_RolBldActr



The function block `FB_BA_RolBldActr` is used to position a roller blind via two outputs: Up and Down. The roller blind can be driven to any desired position with the positioning telegram `stSunBld` [► 217]. In addition, the positioning telegram `stSunBld` [► 217] also contains manual commands with which the roller blind can be moved individually to certain positions. These manual commands are controlled by the function block `FB_BA_SunBldSwi` [► 321].

The current height position is not read by an additional encoder, but is determined internally by the runtime of the roller blind.

The two different runtime parameters `nTiUp` (runtime roller blind up [ms]) and `nTiDwn` (runtime roller blind down [ms]) take account of the different travel characteristics.

The function block fundamentally controls the roller blind via the information from the positioning telegram `stSunBld` [► 217]. If automatic mode is active (`bManMod=FALSE`), the current position is always approached, and changes are immediately taken into account. In manual mode (`bManMod=TRUE`) the commands `bManUp` and `bManDwn` control the roller blind.

Referencing

Safe referencing refers to a situation when the roller blind is upwards-controlled for longer than its complete travel-up time. The position is then always "0". Since a roller blind positioning without encoder is always error-prone by nature, it is important to reference automatically as often as possible: every time the position "0" is to be approached, the roller blind first moves up normally with continuous position calculation. Once the calculated position value 0% is reached, the output `bUp` continues to be held for the complete travel-up time + 5s.

For reasons of flexibility there are now two possibilities to interrupt the referencing procedure: until the calculated 0% position is reached, a change in position continues to be assumed and executed. Once this 0% position is reached, the roller blind can still be moved with the manual "down" command. These two sensible restrictions make it necessary for the user to ensure that the roller blind is referenced safely whenever possible.

After a system restart, the function block executes a reference run. Completion of the initial referencing is indicated through a TRUE signal at output `bInitRefCmpl`. The initial referencing can also be terminated through a manual "down" command.

Inputs

```
VAR_INPUT
    bEn          : BOOL;
    stSunBld     : ST_BA_Sunblind;
    nTiUp        : UDINT;
    nTiDwn       : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	Enable input for the function block. As long as this input is TRUE, the actuator function block accepts and executes commands as described above. A FALSE signal on this input resets the control outputs <i>bUp</i> and <i>bDwn</i> and the function block remains in a state of rest.
stSunBld	ST_BA_SunBld [► 217]	Positioning telegram
nTiUp	UDINT	Complete time for driving up [ms]
nTiDwn	UDINT	Complete time for driving down [ms]

Outputs

```

VAR_OUTPUT
  bUp      : BOOL;
  bDwn     : BOOL;
  fActlPos : REAL;
  bRef     : BOOL;
  nRefTi   : UDINT;
  bInitRefCompl : BOOL;
  bBusy    : BOOL;
  bErr     : BOOL;
  sErrDescr : T_MAXSTRING
END_VAR

```

Name	Type	Description
bUp	BOOL	Roller blind control output up
bDwn	BOOL	Roller blind control output down
fActlPos	REAL	Current position in percent.
bRef	BOOL	The roller blind is in referencing mode, i.e. the output <i>bUp</i> is set for the complete travel-up time + 5s. Only a manual "down" command can move the roller blind in the opposite direction and terminate this mode.
nRefTi	UDINT	Referencing countdown display [s]
bInitRefCompl	BOOL	Initial referencing process complete
bBusy	BOOL	A positioning or a referencing procedure is in progress.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Error: The total travel-up or travel-down time (nTiUp / nTiDwn) is zero.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.15 FB_BA_ShdCorr

FB_BA_ShdCorr	
tTiActl <i>TIMESTRUCT</i>	BOOL bGrpNotShdd
fFcdOrtn <i>REAL</i>	BOOL bFcdSunlit
fAzm <i>REAL</i>	BOOL bErr
fElv <i>REAL</i>	T_MaxString sErrDescr
nGrpID <i>UDINT</i>	
bSouth <i>BOOL</i>	
aShdObj <i>ARRAY [1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj</i>	
aFcdElem <i>ARRAY [1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem</i>	

The function block FB_BA_ShdCorr is used to assess the shading of a group of windows on a facade.

The function block *FB_BA_Shdcorr* calculates whether a window group lies in the shadow of surrounding objects. The result, which is output at the output *bGrpNotShdd*, can be used to assess whether sun protection makes sense for this window group.

The function block thereby accesses two lists, which are to be defined:

- The parameters that describe the shading elements that are relevant to the facade on which the window group is located. This [list of shading objects](#) [► 274] is used as input variable *aShdObj* for the function block, since the information is read only.
- The data of the elements (window) of the facade in which the group to be regarded is located. This [list of facade elements](#) [► 274] is accessed via the IN/OUT variable *aFcdElem*, since not only the window coordinates are read, but the function block *FB_BA_Shdcorr* also stores the shading information for each window corner in this list. In this way, the information can also be used in other parts of the application program.

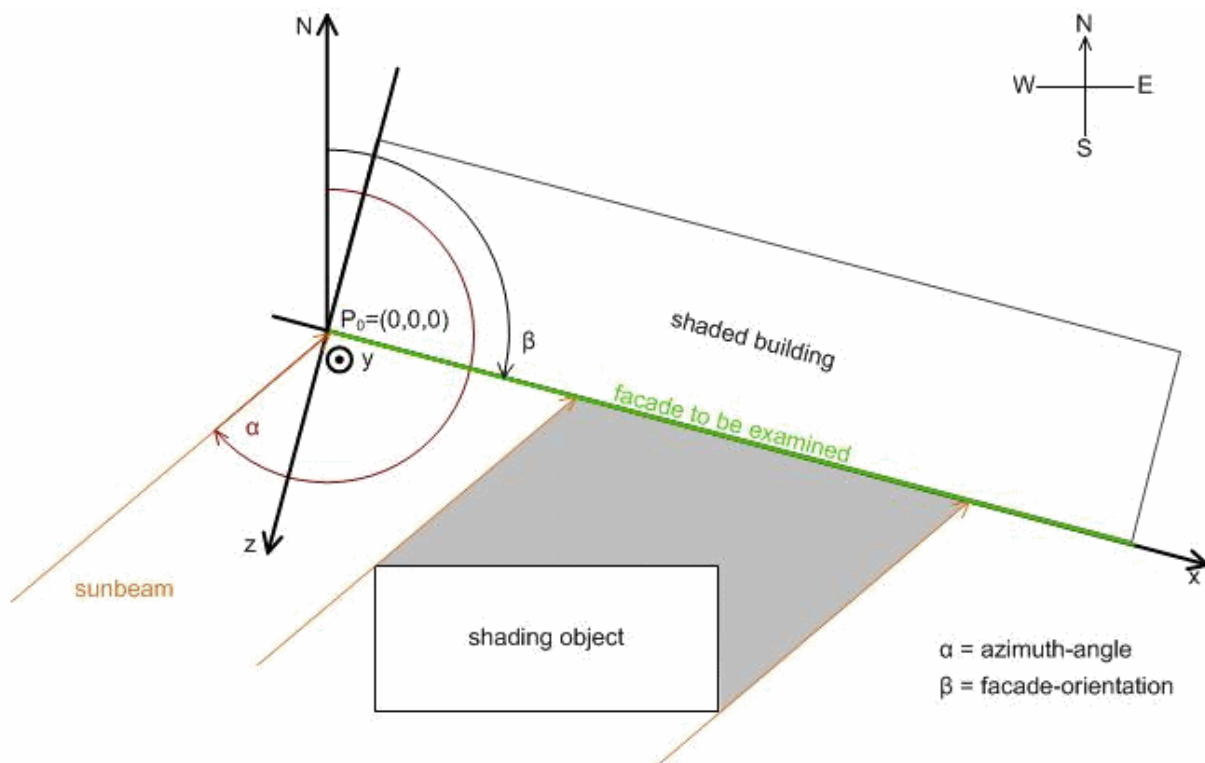
On the basis of the facade orientation (*fFcdOrtn*), the direction of the sun (*fAzsm*) and the sun elevation (*fElv*), a calculation can be performed for each corner of a window to check whether this lies in a shaded area. A window group is considered to be completely shaded if all corners are shaded.

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):

Viewing direction	Facade orientation
North	$\beta=0^\circ$
East	$\beta=90^\circ$
South	$\beta=180^\circ$
West	$\beta=270^\circ$

The function block performs its calculations only if the sun is actually shining on the facade. Considering the drawing presented in the introduction, this is the case if:

Facade orientation < azimuth angle < facade orientation + 180°

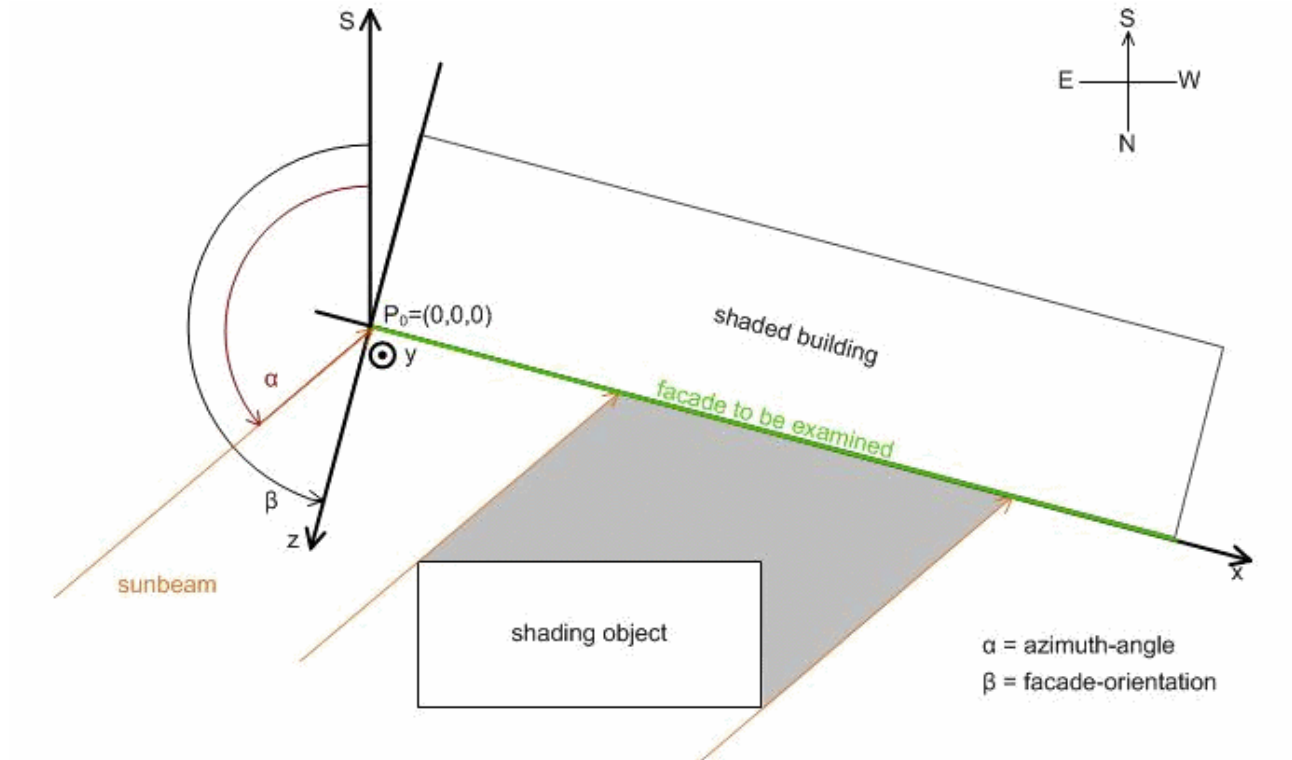


In addition, a calculation is also not required, if the sun has not yet risen, i.e. the sun elevation is below 0°. In both cases the output *bFcdSunlit* is set to FALSE.

The situation is different for the southern hemisphere. The following applies to the facade orientation (looking out the window):

Viewing direction	Facade orientation
South	$\beta=0^\circ$
East	$\beta=90^\circ$
North	$\beta=180^\circ$
West	$\beta=270^\circ$

The internal calculation or the relationship between facade and sunbeam also changes:



To distinguish between the situation in the northern and southern hemisphere, set the input parameter *bSouth* to FALSE (northern hemisphere) or TRUE (southern hemisphere)

Inputs

```
VAR_INPUT
  tTiAct1    : TIMESTRUCT;
  fFcdOrtn   : REAL;
  fAzm       : REAL;
  fElv       : REAL;
  nGrpID     : DINT;
  bSouth     : BOOL;
  aShdObj    : ARRAY[1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShObj;
END_VAR
```

Name	Type	Description
tTiActI	TIMESTRUCT	Input of the current time - local time in this case, since this time takes into account the shaded months. If the UTC time (or GMT) is used, the month may change in the middle of the day, depending on the location on the earth.
fFcdOrtn	REAL	Facade orientation, see illustration above.
fAzm	REAL	Direction of the sun at the time of observation [°]
fElv	REAL	Sun elevation at the time of observation [°]
nGrpID	UDINT	Window group regarded. The group 0 is reserved here for unused window elements (see FB BA_FcdElemEntry [► 278]). A 0-entry would lead to an error output (bErr=TRUE). The function block is then not executed any further and <i>bGrpNotShdd</i> is set to FALSE.
bSouth	BOOL	FALSE: Calculations refer to conditions in the northern hemisphere - TRUE: in the southern hemisphere
aShdObj	ST_BA_ShddObj	List of shading objects [► 274].

/ Inputs/Outputs

```
VAR_IN_OUT
  aFcdElem : ARRAY[1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Name	Type	Description
aFcdElem	ST_BA_FcdElem	List of facade elements [► 274]

Outputs

```
VAR_OUTPUT
  bOut      : BOOL;
  bErr      : BOOL;
  sErrDescr : T_MAXSTRING;
END_VAR
```

Name	Type	Description
bOut	BOOL	The facade element is in the sun if the output is TRUE.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

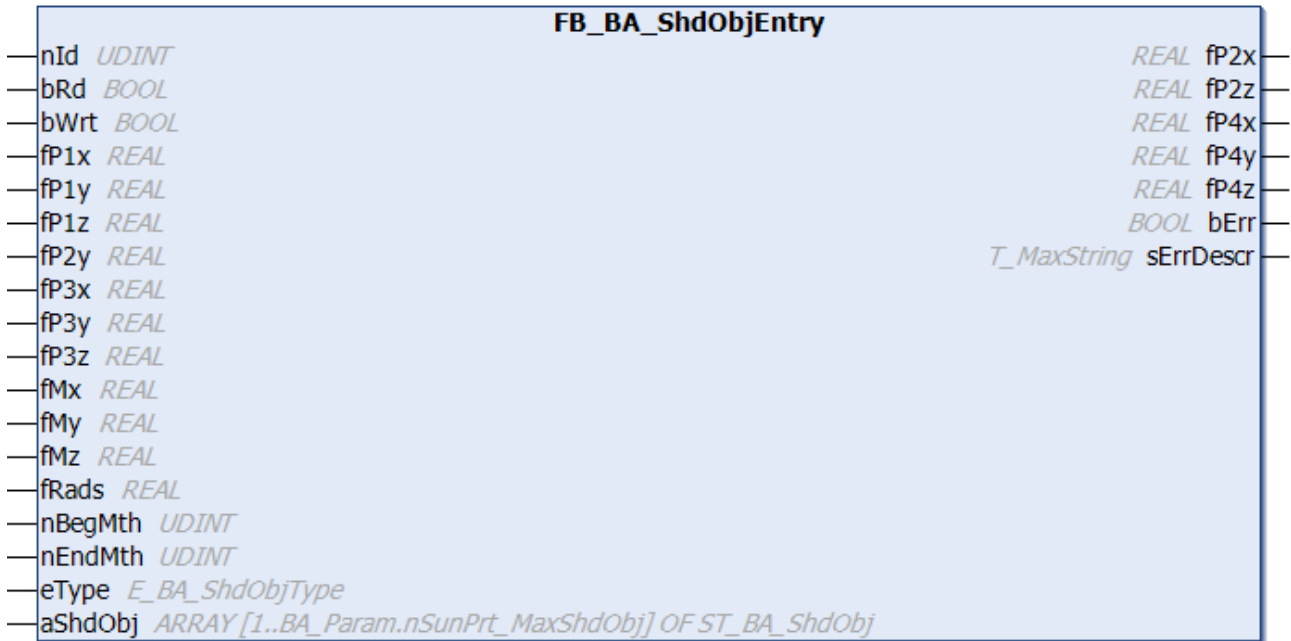
01: Error: *fHiLmt* less than or equal to *fLoLmt*.

02: Error: *fLoLmt* is less than 0° or *fHiLmt* is greater than 90°.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.16 FB_BA_ShObjEntry



The function block FB_BA_ShObjEntry serves for the administration of all shading elements in a facade, which are globally saved in a [list of shading elements](#) [► 274]. It is intended to facilitate the input of the element information - also with regard to the use of the visualization. A schematic representation of the objects with description of the coordinates is shown in [Shading correction: principles and definitions](#) [► 261].

The shading elements are declared in the global variables:

```
VAR_GLOBAL
    arrShdObj : ARRAY[1..BA_Param.nSunPrt.nMaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Each individual element *aShdObj[1]* to *aShdObj[nMaxShdObj]* carries the information for one shading element (ST_BA_ShdObj [► 218]). This information consists of the selected type of shading (rectangle or sphere) and the respectively associated coordinates. For a rectangle, these are the corner points (*fP1x*, *fP1y*, *fP1z*), (*fP2x*, *fP2y*, *fP2z*), (*fP3x*, *fP3y*, *fP3z*) and (*fP4x*, *fP4y*, *fP4z*) for a sphere this are the center point (*fMx*, *fMy*, *fMz*) and the radius *fRads*. In addition, the phase of the shading can be defined via the inputs *nBegMth* and *nEndMth*, which is important in the case of objects such as trees that bear no foliage in winter.

The function block thereby directly accesses the field of this information via the IN-OUT variable *aShdObj*.

Note: The fact that the rectangle coordinates *fP2x*, *fP2z*, *fP4x*, *fP4y* and *fP4z* are output values results from the fact that they are formed from the input parameters:

fP2x = *fP1x*; *fP2z* = *fP1z*; *fP4x* = *fP3x*; *fP4y* = *fP1y*; *fP4z* = *fP3z*;

That limits the input of a rectangle to the extent that the lateral edges stand vertically on the floor (*fP2x* = *fP1x* and *fP4x* = *fP3x*), that the rectangle has no inclination (*fP2z* = *fP1z* and *fP4z* = *fP3z*) and can only have a different height "upwards", i.e. in the positive y-direction (*fP4y* = *fP1y*).

The function block is used in three steps:

- Read
- Change
- Write

Read

Selection of the element from the list *aShdObj[nId]* is based on the entry at *nId*. A rising edge on *bRd* reads the data. These values are assigned to the input and output variables of the function block. These are the input values *fP1x*, *fP1y*, *fP1z*, *fP2y*, *fP3x*, *fP3y*, *fP3z*, *fMx*, *fMy*, *fMz*, *fRads*, the object enumerator *eType* and the output values *fP2x*, *fP2z*, *fP4x*, *fP4y* and *fP4z*. It is important here that the input values are not overwritten in the reading step. Hence, all values can initially be displayed in a visualization.

Change

In a next program step the listed input values can then be changed. If a rectangle is preselected at input `eType` via the value `"eObjectTypeTetragon"`, the output values `fP2x`, `fP2z`, `fP4x`, `fP4y`, and `fP4z` result from the rectangle coordinates that were entered (see above).

The values entered are constantly checked for plausibility. The output `bErr` indicates whether the values are valid (`bErr=FALSE`). If the value is invalid, a corresponding error message is output at output `sErrDescr`. If a rectangle is defined, only the inputs `fP1x`, `fP1y`, `fP1z`, `fP2y`, `fP3x`, `fP3y` and `fP3z` must be written to, the inputs `fMx`, `fMy`, `fMz` and `fRads` need not be linked. For a sphere definition, only `fMx`, `fMy`, `fMz` and `fRads` have to be described; the rectangle coordinates can remain unlinked

Write

The parameterized data are written to the list element with the index `nId` upon a positive edge on `bWrt`, regardless of whether they represent valid values or not. The element structure `ST_BA_ShdObj` [► 218] therefore contains a plausibility bit `bVld`, which forwards precisely this information to the function block `FB_BA_ShdCorr` [► 299] to prevent miscalculations.

This approach is to be regarded only as a proposal. It is also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on `bWrt`.

Inputs

```
VAR_INPUT
  nId      : UDINT;
  bRd      : BOOL;
  bWrt     : BOOL;
  fPlx     : REAL;
  fPly     : REAL;
  fPlz     : REAL;
  fP2y     : REAL;
  fP3x     : REAL;
  fP3y     : REAL;
  fP3z     : REAL;
  fMx      : REAL;
  fMy      : REAL;
  fMz      : REAL;
  fRads    : REAL;
  nBegMth  : UDINT;
  nEndMth  : UDINT;
  eType    : E_BA_ShdObjType;
END_VAR
```

Name	Type	Description
nId	UDINT	Index of the selected element. This refers to the selection of a field element of the array stored in the IN-OUT variable <i>aShdObj</i> . The variable <i>nId</i> must not be zero! This is due to the field definition, which starts with 1. However, an incorrect input is recognized and displayed as such at <i>bErr/sErrDescr</i> .
bRd	BOOL	The information of the selected element, <i>aShdObj[nId]</i> , is read into the function block with a positive edge at this input and assigned to the input variables <i>fP1x</i> to <i>eType</i> and the output variables <i>fP2x</i> to <i>fP4z</i> . If at this time data have already been applied to the inputs <i>fP1x</i> to <i>eType</i> , the previously read data are immediately overwritten with these.
bWrt	BOOL	A positive edge writes the values applied to inputs <i>fP1x</i> to <i>eType</i> and the values determined and assigned to outputs <i>fP2x</i> to <i>fP4z</i> to the selected field element <i>aShdObj[nId]</i> .
fP1x	REAL	X-coordinate of point 1 of the shading element (rectangle) [m].
fP1y	REAL	Y-coordinate of point 1 of the shading element (rectangle) [m].
fP1z	REAL	Z-coordinate of point 1 of the shading element (rectangle) [m].
fP2y	REAL	Y-coordinate of point 2 of the shading element (rectangle) [m].
fP3x	REAL	X-coordinate of point 3 of the shading element (rectangle) [m].
fP3y	REAL	Y-coordinate of point 3 of the shading element (rectangle) [m].
fP3z	REAL	Z-coordinate of point 3 of the shading element (rectangle) [m].
fMx	REAL	X-coordinate of the center of the shading element (ball) [m].
fMy	REAL	Y-coordinate of the center of the shading element (ball) [m].
fMz	REAL	Z-coordinate of the center of the shading element (ball) [m].
fRads	REAL	Radius of the shading element (ball) [m].
nBegMth	UDINT	Beginning of the shading period (month).
nEndMth	UDINT	End of the shading period (month).
eType	E_BA_ShObjType	Selected element type: rectangle or sphere.

Remark about the shading period:

The month entries must not be 0 and not be greater than 12, all other combinations are possible.

Examples:

Start=1, End=1: shading in January.

Start=1, End=5: shading from the beginning of January to the end of May.

Start=11, End=5: shading from the beginning of November to the end of May (of the following year).

/ Inputs/Outputs

```
VAR_IN_OUT
  aShdObj: ARRAY[1..BA_Param. nSunPrt_MaxShdObj] OF ST_BA_ShObj;
END_VAR
```

Name	Type	Description
aShdObj	ST_BA_ShObj	List of shading objects [▶ 274]

Outputs

```

VAR_OUTPUT
  fP2x      : REAL;
  fP2z      : REAL;
  fP4x      : REAL;
  fP4y      : REAL;
  fP4z      : REAL;
  bErr      : BOOL;
  sErrDescr : T_MAXSTRING;
END_VAR

```

Name	Type	Description
fP2x	REAL	Determined X-coordinate of point 2 of the shading element (rectangle) [m] (see "Note [► 303]" above).
fP2Z	REAL	Determined Z-coordinate of point 2 of the shading element (rectangle) [m] (see "Note [► 303]" above).
fP4x	REAL	Determined X-coordinate of point 4 of the shading element (rectangle) [m] (see "Note [► 303]" above).
fP4y	REAL	Determined Y-coordinate of point 4 of the shading element (rectangle) [m] (see "Note [► 303]" above).
fP4z	REAL	Determined Z-coordinate of point 4 of the shading element (rectangle) [m] (see "Note [► 303]" above).
bErr	BOOL	Contains the error description.
sErrDescr	T_MAXSTRING	Contains the error description.

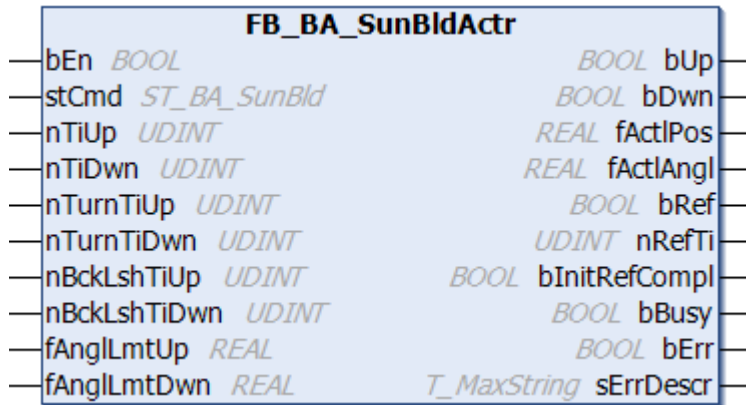
Error description

01: Error: The input <i>nId</i> is outside the permissible limits 1... <i>nMaxShdObj</i> .
02 Error: The sum of the angles of the rectangle is not 360°. This means that the corners are not in the order P1, P2, P3 and P4 but rather P1, P3, P2 and P4. This results in a crossed-over rectangle.
03: Error: The corners of the rectangle are not in the same plane.
04: Error: The z-component of P1 is less than zero. This corner would thus lie behind the facade.
05: Error: The z-component of P3 is less than zero. This corner would thus lie behind the facade.
06: Error: P1 is equal to P2. The object entered is thus not a rectangle.
07: Error: P1 is equal to P3. The object entered is thus not a rectangle.
08: Error: P1 is equal to P4. The object entered is thus not a rectangle.
09: Error: P2 is equal to P3. The object entered is thus not a rectangle.
10: Error: P2 is equal to P4. The object entered is thus not a rectangle.
11: Error: P3 is equal to P4. The object entered is thus not a rectangle.
12: Error: The radius entered is zero.
13: Error: The z-component of the ball center is less than zero. This point would thus lie behind the facade.
14: Error: Error object type <i>eType</i> - neither rectangle nor ball.
15: Error: Month input error.

Requirements

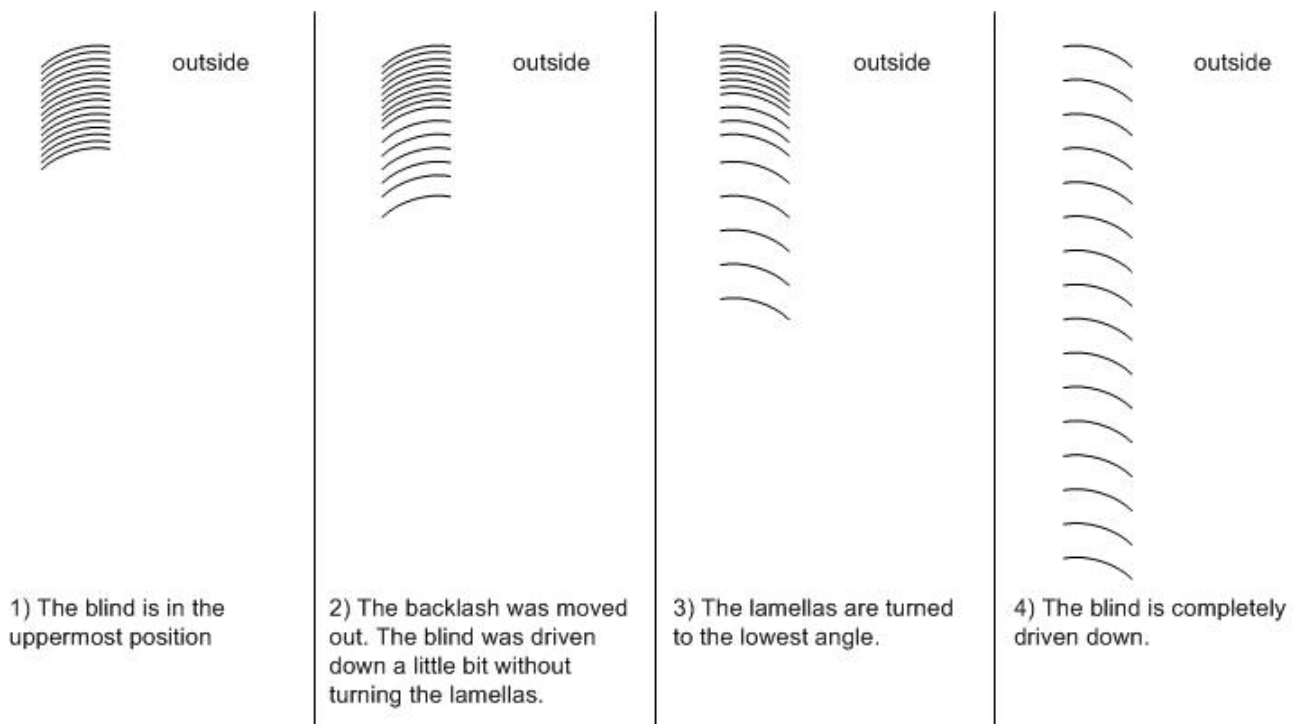
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.17 FB_BA_SunBldActr



The function block FB_BA_SunBldActr is used to position a blind via two outputs: Up and Down. The blind can be driven to any desired (height) position and slat angle via the positioning telegram [stSunBld \[► 217\]](#). In addition, the positioning telegram [stSunBld \[► 217\]](#) also contains manual commands with which the blind can be moved individually to certain positions. These manual commands are controlled by the function block [FB_BA_SunBldSwi \[► 321\]](#).

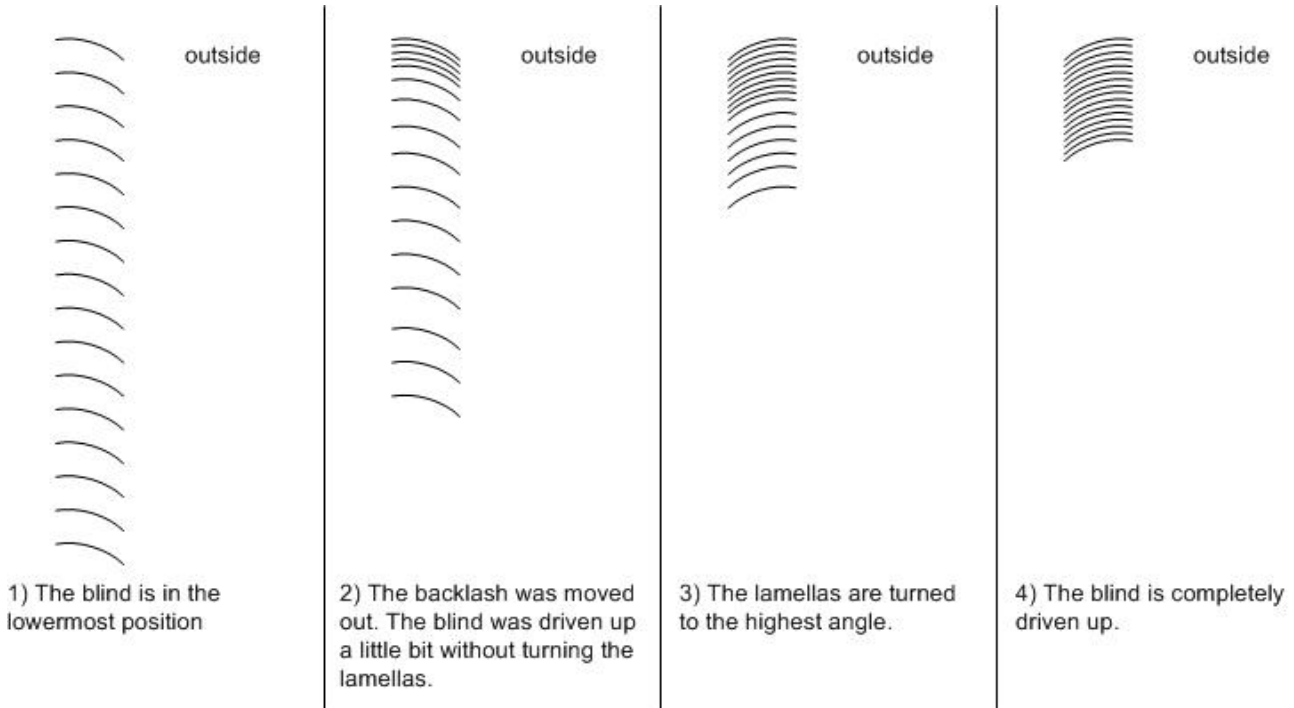
The current height position and the slat angle are not read in by an additional encoder, but determined internally by the travel time of the blind. The calculation is based on the following travel profile (regarded from the highest and lowest position of the blind):

Downward travel profile:**More detailed explanations of the terms "backlash" and "turning" are given here in the downward movement:**

The blind normally describes its downward movement with the slat low point directed outwards, as in fig. 3). If the blind is in an initial position with the low point directed inwards (i.e. after the conclusion of an upward movement), then a certain time elapses after a new downward movement begins before the slats start to turn from the "inward low point" to the "outward low point". During this time the slat angle does not change; the blind only drives downward (fig.1 and fig. 2). This time is an important parameter for the movement calculation and is entered in the function block under *nBckLshTiDwn* [ms]. Since it is not known at any point after a blind movement of any length whether backlash has already taken effect, the backlash of the

downward movement or its travel time can be measured most reliably if the blind was first raised fully. A further important parameter is the time interval of the subsequent turning of the slats from the "Inward low point" to the "Outward low point". This time should be entered as *nTurnTiDwn* [ms] at the function block.

Upward travel profile:



More detailed explanations of the terms "backlash" and "turning" are given here in the upward movement:

The circumstances are similar to the downward movement described above: The blind normally describes its upward movement with the slat low point directed inwards, as in fig. 3).

If the blind is in an initial position with the low point directed outwards (i.e. after the conclusion of a downward movement), then a certain time elapses after a new upward movement begins before the slats start to turn from the "Outward low point" to the "Inward low point". During this time the slat angle does not change; the blind only drives upward (fig. 1 and fig. 2). Also this time is an important parameter for the movement calculation and is entered in the function block under *nBckLshTiUp* [ms]. Since it is not known at any point after a blind movement of any length whether backlash has already taken effect, the backlash of the upward movement or its travel time can be measured most reliably if the blind was first driven fully downward. A further important parameter is the time interval of the subsequent turning of the slats from the "Outward low point" to the "Inward low point". This time should be entered as *nTurnTiUp* [ms] at the function block.

Parameterization

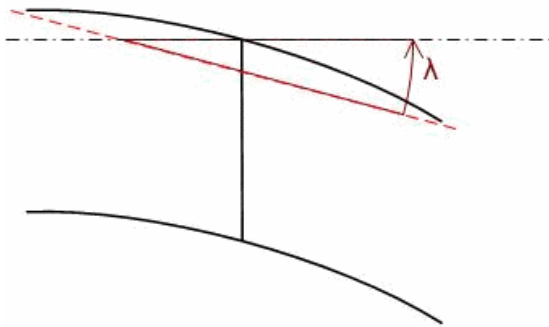
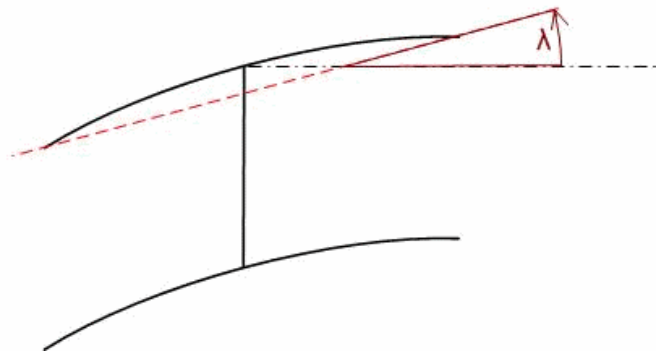
For the calculation of the (height) position and the slat angle, the following times now have to be determined for both the upward and downward movement:

- the backlash duration (*nBckLshTiUp* / *nBckLshTiDwn* [ms])
- the turning duration (*nTurnTiUp* / *nTurnTiDwn* [ms])
- the total travel time (*nTiUp* / *nTiDwn* [ms])

Furthermore the following are required for the calculation:

- the highest slat angle after turning upwards (*fAnglLmtUp* [°])
- the lowest slat angle after turning downwards (*fAnglLmtDwn* [°])

The slat angle λ is defined by a notional straight line through the end points of the slat to the horizontal.

louvre angle $\lambda < 0$ louvre angle $\lambda > 0$ 

Functioning

As a rule, the function block controls the blind based on the information from the positioning telegram `stSunBld` [► 602]. If automatic mode is active (`bManMod = FALSE`), then the current position and slat angle are always driven to, wherein changes are immediately accounted for. The height positioning takes priority: First the entered height and afterwards the slat angle are driven to. For reasons of the simplicity the position error due to the angle movement is disregarded. In manual mode (`bManMod = TRUE`), the blind is controlled by the commands `bManUp` and `bManDwn`.

An automatic movement command is triggered whenever a change from manual to automatic mode occurs.

Referencing

Secure referencing is ensured if the blind is driven upward for longer than its complete drive-up time. The position is then in any case "0" and the louvre angle is at its maximum. Since blind positioning without an encoder is naturally always susceptible to error, it is important to automatically reference as often as possible: each time the "0" position is to be driven to (the angle is unimportant), the blind initially drives upward quite normally with continuous position calculation. Once the calculated position value 0% is reached, the output `bUp` continues to be held for the complete travel-up time + 5 s.

For reasons of flexibility, there are two ways to interrupt the referencing process: Until the calculated 0% position is reached, a change in position continues to be assumed and executed. Once this 0% position is reached, the blind can still be moved with the manual "travel-down" command. These two sensible limitations make it necessary for the user to ensure that the blind is securely referenced as often as possible.

After a system restart, the function block executes a reference run. Completion of the initial referencing is indicated through a TRUE signal at output `blnitRefCmpl`. The initial referencing can also be terminated through a manual "travel-down" command.

Target accuracy

Since the function block determines the blind position solely via travel times, the cycle time of the PLC task plays a crucial role for positioning accuracy. If the switching time for a slat angle range of -70° to 10° is 1 second, for example, the accuracy at a cycle time of 50 ms is $\pm 4^\circ$.

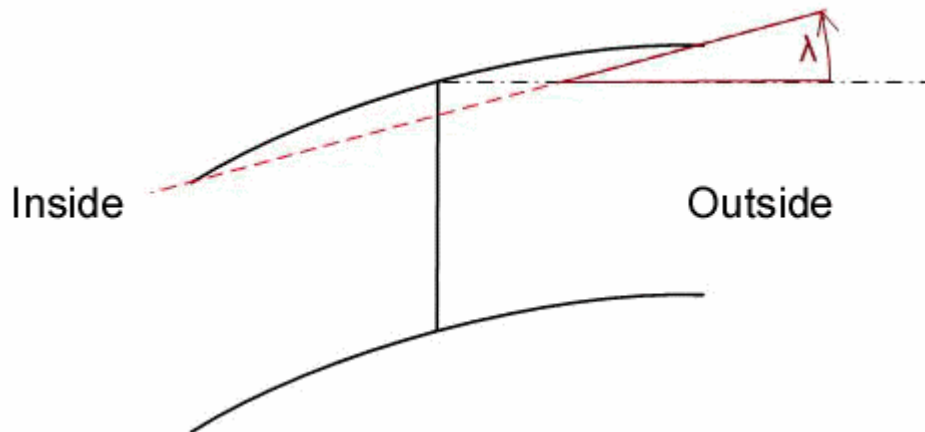
Inputs

```
VAR_INPUT
    bEn          : BOOL;
    stCmd        : ST_BA_SunBld;
    nTiUp        : UDINT;
    nTiDwn       : UDINT;
    nTurnTiUp    : UDINT;
    nTurnTiDwn   : UDINT;
    nBckLshTiUp  : UDINT;
    nBckLshTiDwn : UDINT;
    fAnglLmtUp   : REAL;
    fAnglLmtDwn  : REAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	Enable input for the function block. As long as this input is TRUE, the actuator function block accepts and executes commands as described above. A FALSE signal on this input resets the control outputs <i>bUp</i> and <i>bDwn</i> and the function block remains in a state of rest.
stCmd	ST_BA_SunBld [► 217]	Positioning telegram
nTiUp	UDINT	Complete time for driving up [ms]
nTiDwn	UDINT	Complete time for driving down [ms].
nTurnTiUp	UDINT	Time for turning the slats in the upward direction [ms].
nTurnTiDwn	UDINT	Time for turning the slats in the downward direction [ms].
nBckLshTiUp	UDINT	Time to traverse the backlash in the upward direction [ms]. This input is internally limited to a minimum value of 0.
nBckLshTiDwn	UDINT	Time to traverse the backlash in the downward direction [ms]. This input is internally limited to a minimum value of 0.
fAnglLmtUp	REAL	Highest position of the slats [°].

This position is reached once the blind has moved to the top position.

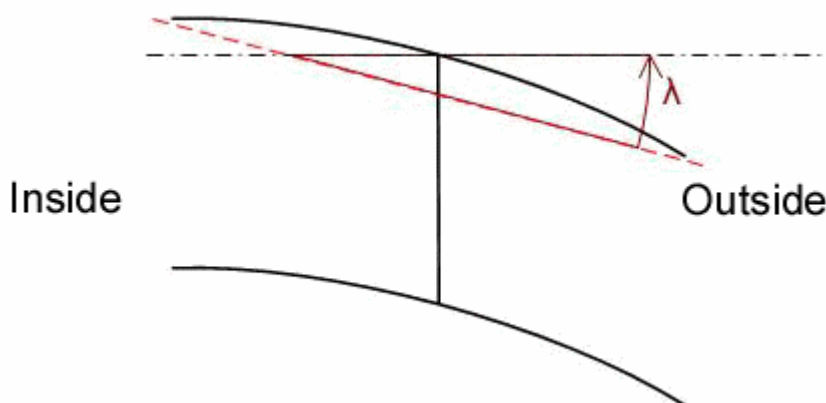
The slat angle λ , as defined above, is then typically greater than zero.



Name	Type	Description
fAnglLmtDwn	REAL	Lowest position of the slats [°].

This position is reached once the blind has moved to the bottom position.

The slat angle λ , as defined above, is then typically less than zero.



🔌 Outputs

```

VAR_OUTPUT
  bUp          : BOOL;
  bDwn         : BOOL;
  fActlPos     : REAL;
  fActlAngl    : REAL;
  bRef         : BOOL;
  nRefTi       : UDINT;
  bInitRefCompl : BOOL;
  bBusy        : BOOL;
  bErr         : BOOL;
  sErrDesc     : T_MAXSTRING;
END_VAR

```

Name	Type	Description
bUp	BOOL	Control output for blind up
bDwn	BOOL	Control output for blind down
fActlPos	REAL	Current position in percent
fActlAngl	REAL	Current slat angle [°]
bRef	BOOL	The blind is in referencing mode, i.e. the output <i>bUp</i> is set for the complete travel-up time + 5s. Only a manual "down" command can move the blind in the opposite direction and terminate this mode.
nRefTi	UDINT	Referencing countdown display [s]
bInitRefCompl	BOOL	Initial referencing process complete.
bBusy	BOOL	A positioning or a referencing procedure is in progress.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Error: Up/down timer = 0.

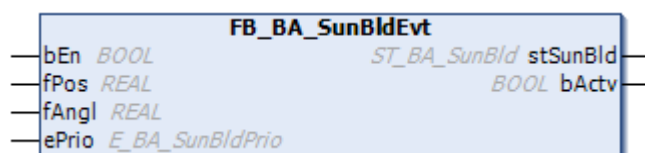
02: Error: Turning timer = 0.

03: Error: Slat angle limits: the upper limit is less than or equal to the lower limit (*fAnglLmtUp* <= *fAnglLmtDwn*).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.18 FB_BA_SunBldEvt



The function block FB_BA_SunBldEvt is used for position and angle specification at any event. It can be used, for example, in order to drive to a parking position or to drive the blind upward for maintenance.

The function is enabled via the input *bEn*. If this is the case, the active flag in the positioning telegram (*bActv* in *stSunBld*) is set at output *stSunBld* [► 217] and the values entered at the In-Out variables *fPos* for the blind height [%] and *fAngl* for the slat angle [°] are passed on in this telegram. If the function is no longer active due to the resetting of *bEn*, then the active flag in the positioning telegram *stSunBld* [► 217] is reset and the positions for height and angle are set to "0". The priority function block (e.g. FB_BA_SunBldPrioSwi4 [► 512]) enables a function with lower priority to take over the control by resetting.

Inputs

```
VAR_INPUT
    bEn      : BOOL;
    fPos     : REAL;
    fAngl    : REAL;
    ePrio    : E_BA_SunBldPrio := E_BA_SunBldPrio.eSunProtection;
END_VAR
```

Name	Type	Description
bEn	BOOL	A TRUE signal on this input activates the function block and transfers the entered setpoints together with the active flag in the positioning telegram ST_BA_SunBld [► 217] . A FALSE signal resets the active flag again and sets position and angle to zero.
fPos	REAL	Height position of the blind [%] in case of activation.
fAngl	REAL	Slat angle of the blind [°] in case of activation
ePrio	E_BA_SunBldPrio	Priority of the active telegram

Outputs

```
VAR_OUTPUT
    stSunBld : ST_BA_SunBld;
    bActv    : BOOL;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output structure of the blind positions.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram ST_BA_SunBld [► 217] and is solely used to indicate whether the function block sends an active telegram.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.19 FB_BA_SunBldIcePrtc



The function block `FB_BA_SunBldIcePrtc` deals with direction-independent anti-icing.

The weather protection has the highest priority in the blind controller (see [overview \[► 268\]](#)) and is intended to ensure that the blind is not damaged by ice or wind.

Impending icing up is detected when the measured outside temperature *fOtsT* falls below the frost limit value *fFrstT* while at the same time rain is detected on *bRainSns*. This event is saved internally and remains active until it is ensured that the ice has melted again. In addition, the outside temperature must have exceeded the frost limit value for the entered deicing time *nDeiceTi* [s]. For safety reasons the icing event is persistently saved, i.e. also beyond a PLC failure. Thus, if the controller fails during the icing up or deicing period, the blind is considered to be newly iced up when then the controller restarts and the deicing timer starts from the beginning again.

If there is a risk of icing, the blind is moved to the protection position specified by *fPosProt* (height position [%]) and *fAnglProt* (slat angle [°]).

Inputs

```
VAR_INPUT
    bEn          : BOOL;
    fOtsT        : REAL;
    bRainSns     : BOOL;
    fFrstT       : REAL;
    nDeiceTi     : UDINT;
    fPosProt     : REAL;
    fAnglProt    : REAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	A TRUE signal on this input activates the function block and transfers the entered setpoints together with the active flag in the positioning telegram <i>ST_BA_SunBld</i> [► 217]. A FALSE signal resets the active flag again and sets position and angle to zero. <i>bActv</i> is FALSE. This means that another function takes over control of the blind via the priority controller.
fOtsT		Outside temperature [°C]
bRainSns		Input for a rain sensor.
fFrstT		Icing up temperature limit value [°] Celsius. This value may not be greater than 0. Otherwise an error is output.
nDeiceTi		Time until the deicing of the blind after icing up [s]. After that the ice alarm is reset.
fPosProt		Height position of the blind [%] in the case of protection.
fAnglProt		Slat angle of the blind [°] in the case of protection.

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
    ePrio      : E_BA_SunBldPrio := E_BA_SunBldPrio.eIce;
END_VAR
```

Name	Type	Description
ePrio	E_BA_SunBldPrio	Priority of the active program

Outputs

```
VAR_OUTPUT
    stSunBld      : ST_BA_SunBld;
    bActv         : BOOL;
    bIceAlm       : BOOL;
    nRemTiIceAlm  : UDINT;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output structure of the blind positions.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram <i>ST_BA_SunBld</i> [► 217] and is solely used to indicate whether the function block sends an active telegram.
bIceAlm	BOOL	Indicates the icing up alarm.
nRemTilceAlm	UDINT	In the case of impending icing up (<i>bIceAlm</i> = TRUE), this second counter is set to the deicing time. As soon as the temperature lies above the frost point entered (<i>fFrstT</i>), the remaining number of seconds until the 'all-clear' signal is given (<i>bIceAlm</i> = FALSE) is displayed here. This output is 0 as long as no countdown of the time is taking place.

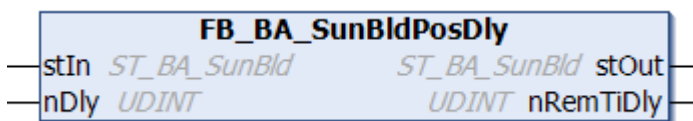


If an error occurs, this automatic control is disabled and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see [Overview](#) [► 268]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.20 FB_BA_SunBldPosDly



This function block *FB_BA_SunBldPosDly* delays changes in position based on automatic commands.

If an event, e.g. weather protection, results in too many blind drives being started at the same time, fuses may be triggered by motor starting current peaks. It is therefore advisable to start the blind drives slightly staggered, in order to avoid excessive total current values.

This function block relays automatic commands from the input telegram *stIn* [► 217] to the output telegram *stOut* [► 217] with a delay. A distinction is made between three cases

1. the blind position *rPos* has changed in automatic mode (*bManMode* = FALSE in telegram *stIn*)
2. the slat angle *rAngl* has changed in automatic mode (*bManMode* = FALSE in telegram *stIn*)
3. manual mode has just been exited, i.e. automatic mode has just become active (falling edge *bManMode* in telegram *stIn*)

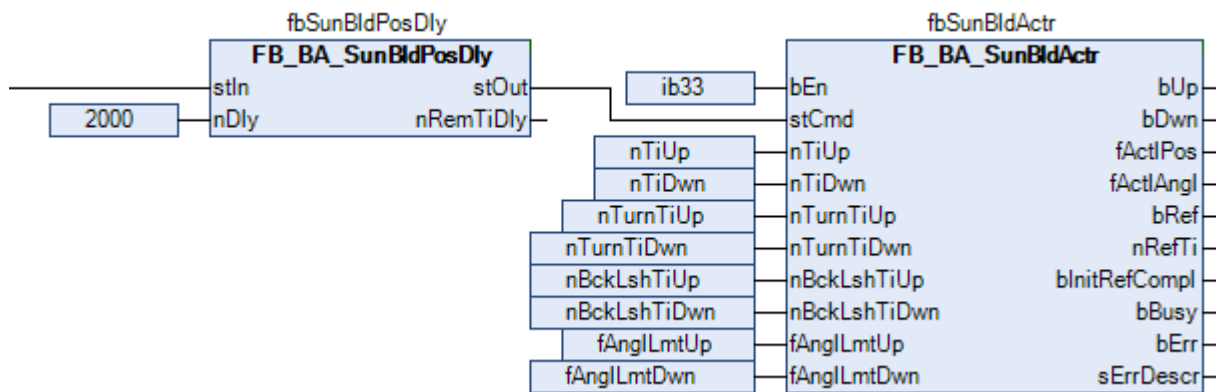
The output telegram *stOut* is always a direct copy of the input telegram *stIn*. In these three cases, however, the output telegram *stOut* is fixed for the time of *nDly* [ms].

This ensures that the blind controlled via the function block *FB_BA_SunBldActr* [► 307] is kept at its position during the delay period. Each further change based on the criteria mentioned above within the delay time restarts the timer.

However, a change to manual in the input telegram (*bManMode* = TRUE) cancels the delay timer immediately. The (manual) telegram is passed on without delay. In this way, **only** automatic telegrams are delayed.

Application

Preferably directly before the blind actuator function block:



Inputs

```
VAR_INPUT
    stIn      : ST_BA_Sunblind;
    nDly      : UDINT;
END_VAR
```

Name	Type	Description
stIn	ST_BA_SunBld [► 217]	Input positioning telegram.
nDly	UDINT	Delay time of the active bit in the positioning telegram [ms].

Outputs

```
VAR_OUTPUT
    stOut      : ST_BA_Sunblind;
    nRemTiDly  : UDINT;
END_VAR
```

Name	Type	Description
stOut	ST_BA_SunBld [► 217]	Output positioning telegram.
nRemTiDly	UDINT	Display output for elapsed delay time [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.21 FB_BA_SunBldPosHMI



The function block FB_BA_SunBldPosHMI is used for position and angle specification via a user interface. With each rising edge at *bStart* an active positioning telegram with the target position *fPos* and the slat angle *fAngl* is output at *stSunBld*.

The input *bRstManFnct* is used to reset the output telegram. The telegram is described as follows:

```
stSunBld.bActv := FALSE;
stSunBld.fPos  := 0.0;
stSunBld.fAngl := 0.0;
```

The input *bRstManFnct* has a static effect: as long as it is TRUE, the output telegram is set to the values mentioned above.

In addition, the function block output *bActv* is set to FALSE. This is to indicate whether an active telegram is output.

Inputs

```
VAR_INPUT
    fPos      : REAL;
    fAngl     : REAL;
    bStart    : BOOL;
    bRstManFnct : BOOL;
END_VAR
```

Name	Type	Description
fPos	REAL	Input of the target position
fAngl	REAL	Input of the slat angle
bStart	BOOL	A rising edge at this input outputs an active telegram with the entered target position <i>fPos</i> and the slat angle <i>fAngl</i> .
bRstManFnct	BOOL	A TRUE at this input deletes the telegram at the output (<i>stSunBld.bActv</i> = FALSE, <i>bActv</i> = FALSE, <i>stSunBld.fPos</i> = 0, <i>stSunBld.fAngl</i> = 0) and no new values are passed through. The output <i>bActv</i> then also goes to FALSE.

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
    ePrio : E_BA_SunBldPrio := E_BA_SunBldPrio.eManualActuator;
END_VAR
```

Name	Type	Description
ePrio	E_BA_SunBldPrio	Priority of the output telegram.

Outputs

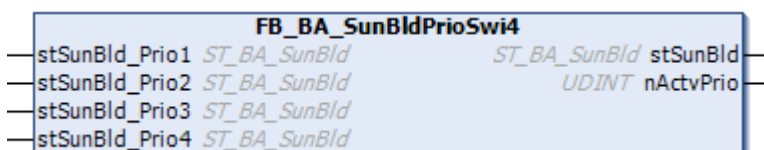
```
VAR_OUTPUT
    stSunBld : ST_BA_SunBld;
    bActv    : BOOL;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram <u>ST_BA_SunBld [► 217]</u> and is solely used to indicate whether the function block sends an active telegram.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.22 FB_BA_SunBldPrioSwi4



The function block FB_BA_SunBldPrioSwi4 is used for priority control for up to 4 positioning telegrams (*stSunBld_Prio1* ... *stSunBld_Prio4*) of type ST_BA_SunBld [► 217] from different control function blocks.

The telegram on *stSunBld_Prio1* has the highest priority and that on *stSunBld_Prio4* the lowest. The active telegram with the highest priority is output at *stSunBld*. Active means that the variable *bActv* is set within the structure of the positioning telegram.

This function block is to be programmed in such a way that one of the applied telegrams is always active. If no telegram is active, an empty telegram is output, i.e. *rPos* = 0, *rAngl* = 0, *bManUp* = FALSE, *bManDwn* = FALSE, *bManMod* = FALSE, *bActv* = FALSE. Since the blind function block [FB_BA_SunBldActr](#) [► 307] or the roller blind function block [FB_BA_RolBldActr](#) [► 298] does not take account of the flag *bActv*, this telegram would be interpreted as movement command to position "0", i.e. fully open. The absence of an active telegram therefore does not represent a safety risk for the blind.

Inputs

```
VAR_INPUT
    stSunBld_Prio1 : ST_BA_SunBld;
    stSunBld_Prio2 : ST_BA_SunBld;
    stSunBld_Prio3 : ST_BA_SunBld;
    stSunBld_Prio4 : ST_BA_SunBld;
END_VAR
```

Name	Type	Description
stSunBld_PrioN	ST_BA_SunBld [► 217]	Positioning telegrams available for selection. <i>stSunBld_Prio1</i> has the highest priority and <i>stSunBld_Prio4</i> the lowest.

Outputs

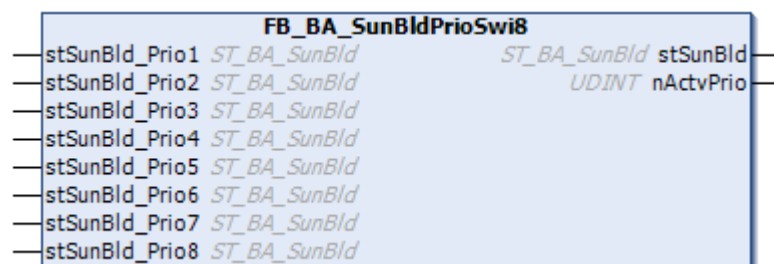
```
VAR_OUTPUT
    stSunBld : ST_BA_SunBld;
    nActvPrio : UDINT;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
nActvPrio	UDINT	Active positioning telegram. If none is active, "0" is output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.23 FB_BA_SunBldPrioSwi8



The function block [FB_BA_SunBldPrioSwi8](#) is used for priority control for up to 8 positioning telegrams (*stSunBld_Prio1* ... *stSunBld_Prio8*) of type [ST_BA_SunBld](#) [► 217] from different control function blocks.

The telegram on *stSunBld_Prio1* has the highest priority and that on *stSunBld_Prio8* the lowest. The active telegram with the highest priority is output at *stSunBld*. Active means that the variable *bActv* is set within the structure of the positioning telegram.

This function block is to be programmed in such a way that one of the applied telegrams is always active. If no telegram is active, an empty telegram is output, i.e. *rPos* = 0, *rAngl* = 0, *bManUp* = FALSE, *bManDwn* = FALSE, *bManMod* = FALSE, *bActv* = FALSE. Since the blind function block [FB_BA_SunBldActr](#) [► 307] or the

roller blind function block **FB BA RolBldActr** [► 298] does not take account of the flag *bActv*, this telegram would be interpreted as movement command to position "0", i.e. fully open. The absence of an active telegram therefore does not represent a safety risk for the blind.

Inputs

```
VAR_INPUT
  stSunBld_Prio1 : ST_BA_SunBld;
  stSunBld_Prio2 : ST_BA_SunBld;
  stSunBld_Prio3 : ST_BA_SunBld;
  stSunBld_Prio4 : ST_BA_SunBld;
  stSunBld_Prio5 : ST_BA_SunBld;
  stSunBld_Prio6 : ST_BA_SunBld;
  stSunBld_Prio7 : ST_BA_SunBld;
  stSunBld_Prio8 : ST_BA_SunBld;
END_VAR
```

Name	Type	Description
stSunBld_PrioN	ST_BA_SunBld [► 217]	Positioning telegrams available for selection. <i>stSunBld_Prio1</i> has the highest priority and <i>stSunBld_Prio8</i> the lowest.

Outputs

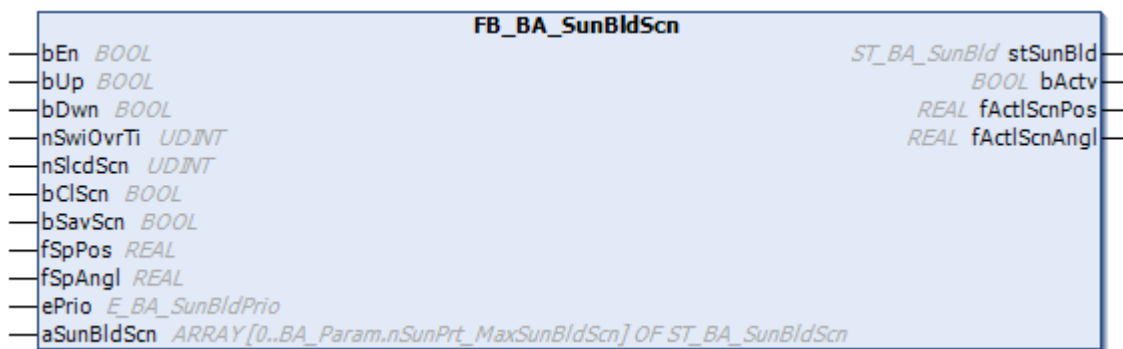
```
VAR_OUTPUT
  stSunBld : ST_BA_SunBld;
  nActvPrio : UDINT;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
nActvPrio	UDINT	Active positioning telegram. If none is active, "0" is output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.24 FB_BA_SunBldScn



The function block **FB_BA_SunBldScn** represents an extension of the manual operation **FB BA SunBldSw** [► 321] by a scene memory and a call function. The blind actuator **FB BA SunBldActr** [► 307] or the roller blind actuator **FB BA RolBldActr** [► 298] can thus be controlled in manual operation mode and can also drive directly to previously saved positions (scenes). Up to 21 scenes can be saved.

Operation

In manual mode, the function block controls the blind function block **FB BA SunBldActr** [► 307] or the roller blind function block **FB BA RolBldActr** [► 298] via the command inputs *bUp* and *bDwn*; *bUp* has priority. The commands are passed on to the respective commands *bManUp* and *bManDwn* of the positioning telegram

► 217]. If a command input is activated for longer than the entered time $nSwiOvrTi$ [ms], the corresponding control command latches. Activating a command input again clears this latching.

A rising edge at $bSavScn$ saves the current position and the slat angle in the scene selected at $nSlcdScn$. This procedure is possible at any time, even during active positioning. With $bClScn$ the selected scene is called up, i.e. the stored values of position and angle are driven to.

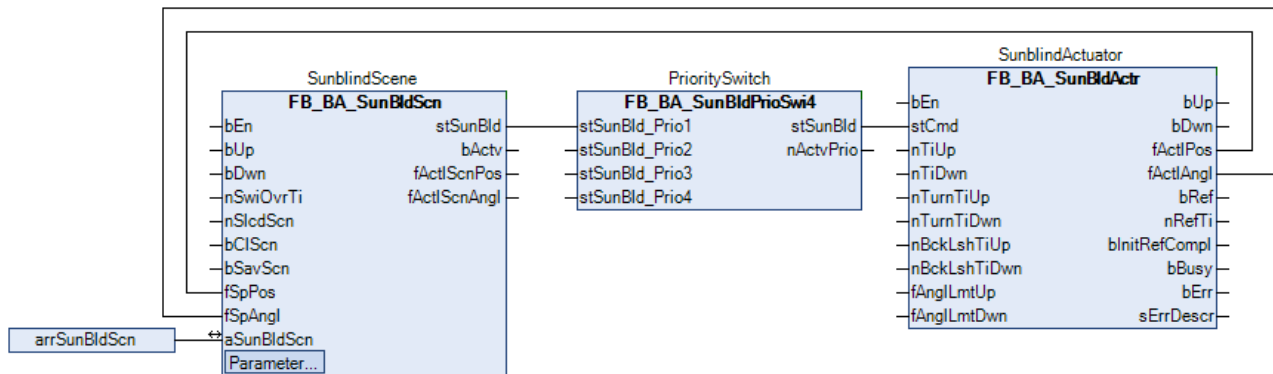
If the function block is activated by input $bEn = TRUE$, bit $bActv$ is set immediately in the positioning telegram. In this way the function block signals its priority over lower priorities at a priority switch (see FB_BA_SunBldTgmSel4 or FB_BA_SunBldTgmSel8). If the command "Call Scene" is not active ($bClScn = TRUE$), the bit $bManMod$ is also set in the positioning telegram to notify the connected actuator function blocks that they should respond to manual commands.

If the function block is deactivated by $bEn = FALSE$, both bits, $bActv$ and $bManMod$, are set to FALSE again.

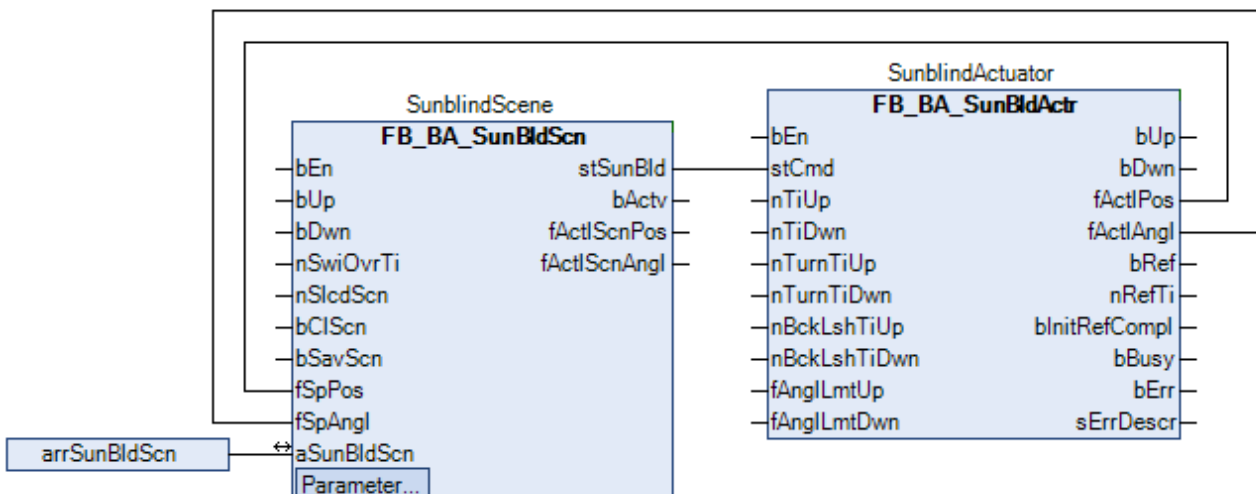
Linking to the blind function block

Like the "normal" manual mode function block FB_BA_SunBldSwi [► 321], the scene selection function block can be connected either via an upstream priority control FB_BA_SunBldPrioSwi4 or FB_BA_SunBldPrioSwi8, or directly via the blind function block. The connection is established via the positioning telegram ST_BA_Sunbld [► 217]. Furthermore the scene function block requires the current positions from the blind function block for the reference blind:

Use of a priority controller:



Direct connection:



Inputs

```
VAR_INPUT
  bEn      : BOOL;
  bUp      : BOOL;
  bDwn     : BOOL;
  nSwiOvrTi : UDINT;
```

```

nSlcdScn      : UDINT;
bClScn        : BOOL;
bSavScn       : BOOL;
fSpPos        : REAL;
fSpAngl       : REAL;
END_VAR

```

Name	Type	Description
bEn	BOOL	The function block has no function if this input is FALSE. 0 is output for the position and the angle in the positioning telegram <u>ST_BA_Sunbld</u> [► 217] - <i>bManMod</i> and <i>bActv</i> are set to FALSE. For a connection with priority controller this means that another functionality takes over control of the blind. Conversely, a direct connection allows the blind to drive directly to the 0 position, i.e. fully up, since the actuator function block does not evaluate the bit <i>bActv</i> itself.
bUp	BOOL	Command input for blind up.
bDwn	BOOL	Command input for blind down.
nSwiOvrTi	UDINT	Time [ms] until the corresponding manual command in the positioning telegram <u>ST_BA_Sunbld</u> [► 217] switches to latching mode, if the command input is activated permanently. Internally limited to a minimum value of 0.
nSlcdScn	UDINT	Selected scene which should either be saved (<i>bSavScn</i>) or called (<i>bClScn</i>). Internally limited to a minimum value from 0 to <u>BA_Param.nSunPrt_MaxSunBldScn</u>
bClScn	BOOL	Call selected scene
bSavScn	BOOL	Save selected scene
fSpPos	REAL	Set position [%] that is to be saved in the selected scene. This must be linked to the actual position of the actuator function block <u>FB_BA_SunBldActr</u> or <u>FB_BA_RolBldActr</u> of the reference blind/roller blind, in order to be able to save a position that was previously approached manually. Internally limited to values between 0 and 100.
fSpAngl	REAL	ditto slat angle [°]

Inputs CONSTANT PERSISTENT

```

VAR_INPUT CONSTANT PERSISTENT
  ePrio      : E_BA_SunBldPrio := E_BA_SunBldPrio.eScene1;
END_VAR

```

Name	Type	Description
ePrio	E_BA_SunBldPrio	Priority of the active program

/ Inputs/Outputs

```

VAR_IN_OUT
  aSunBldScn : ARRAY[0..BA_Param.nSunPrt_MaxSunBldScn] OF ST_BA_SunBldScn;
END_VAR

```

Name	Type	Description
aSunBldScn	<u>ST_BA_SunBldScn</u> [► 219]	Table with the scene entries

Outputs

```

VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  bActv         : BOOL;
  fActlScnPos   : REAL;
  fActlScnAngl  : REAL;
END_VAR

```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram <u>ST_BA_SunBld</u> [► 217] and is solely used to indicate whether the function block sends an active telegram.
fActlScnPos	REAL	Indicates the saved relative blind height position [%] for the currently selected scene.
fActlScnAngl	REAL	ditto slat angle [°]

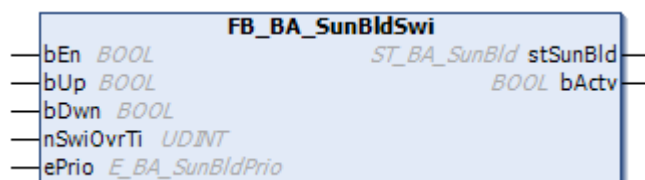


If an error occurs, this automatic control is disabled and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see [Overview](#) [► 268]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.25 FB_BA_SunBldSwi



With the help of the function block FB_BA_SunBldSwi the blind actuator FB_BA_SunBldActr [► 307] or the roller blind actuator FB_BA_RolBldActr [► 298] can be controlled in manual operation mode. The connection takes place via the positioning telegram ST_BA_Sunbld [► 217] either directly or with an additional priority control.

Operation

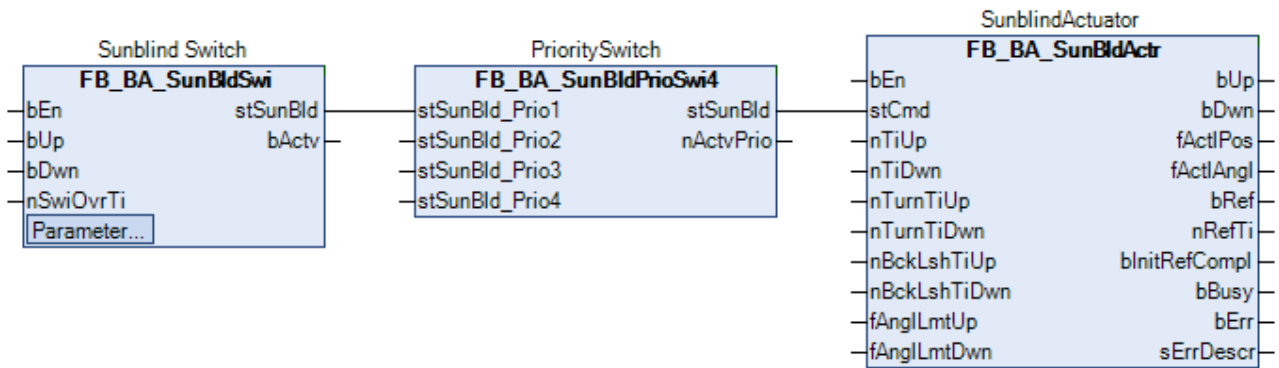
In manual mode, the function block controls the blind function block FB_BA_SunBldActr [► 307] or the roller shutter function block FB_BA_RolBldActr [► 298] via the command inputs *bUp* and *bDwn*; *bUp* has priority. The commands are passed on to the respective commands *bManUp* and *bManDwn* of the positioning telegram. If a command input is activated that is longer than the entered time *nSwiOvrTi* [ms], then the corresponding control command latches. Activating a command input again releases this latch. If the function block is activated by input *bEn* = TRUE, bit *bActv* is set immediately in the positioning telegram. The function block uses this to indicate its priority over low priorities at the priority switch (see). At the same time, the bit *bManMod* is set in the positioning telegram to notify the connected actuator function blocks that they should respond to manual commands.

If the function block is deactivated by *bEn* = FALSE, both bits, *bActv* and *bManMod*, are set to FALSE again.

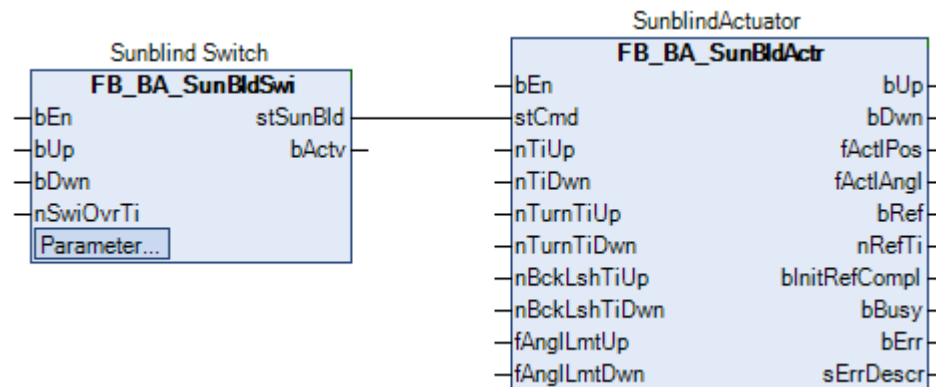
Linking to the blind function block

The manual mode function block can be connected either via an upstream priority control FB_BA_... or directly at the blind function block. The connection is established via the positioning telegram ST_BA_Sunbld [► 217].

Use of a priority controller:



Direct connection:



Inputs

```

VAR_INPUT
    bEn          : BOOL;
    bUp          : BOOL;
    bDwn         : BOOL;
    nSwiOvrTi    : UDINT;
END_VAR

```

Name	Type	Description
bEn	BOOL	The function block has no function if this input is FALSE. 0 is output for the position and the angle in the positioning telegram ST_BA_Sunbld [► 217] - <i>bManMod</i> and <i>bActv</i> are set to FALSE. For a connection with priority controller this means that another functionality takes over control of the blind. Conversely, a direct connection allows the blind to drive directly to the 0 position, i.e. fully up, since the actuator function block does not evaluate the bit <i>bActv</i> itself.
bUp	BOOL	Command input for blind up.
bDwn	BOOL	Command input for blind down.
nSwiOvrTi	UDINT	Time [ms] until the corresponding manual command in the positioning telegram ST_BA_Sunbld [► 217] switches to latching mode, if the command input is activated permanently. Internally limited to a minimum value of 0.

Inputs CONSTANT PERSISTENT

```

VAR_INPUT CONSTANT PERSISTENT
    ePrio      : E_BA_SunBldPrio := E_BA_SunBldPrio.eManualActuator;
END_VAR

```


Name	Type	Description
ePrio	E_BA_SunBldPrio	Priority of the active program

🔌 Outputs

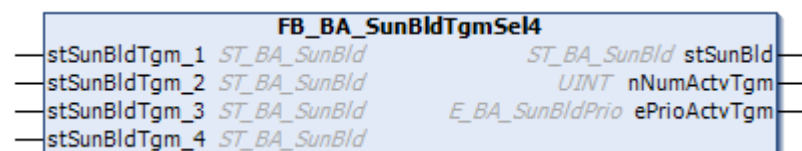
```
VAR_OUTPUT
    stSunBld : ST_BA_SunBld;
    bActv    : BOOL;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram <u>ST_BA_SunBld [► 217]</u> and is solely used to indicate whether the function block sends an active telegram.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.26 FB_BA_SunBldTgmSel4



The function block FB_BA_SunBldTgmSel4 is used for priority control for up to 4 positioning telegrams (*stSunBld_Prio1 ... stSunBld_Prio4*) of type ST_BA_SunBld [► 217] from different control function blocks.

The active telegram with the highest priority is output at *stSunBld*. Active means that the variable *bActv* is set within the structure of the positioning telegram. The priority is stored within the telegram structure as *ePrio*. The smaller the value of *ePrio*, the higher the priority.

For telegrams with the same priority, the last changed one (last writer wins) is valid, determined by the variable *nEvtInc*.

This function block is to be programmed in such a way that one of the applied telegrams is always active. If a telegram is not active, an empty telegram is output at the output:

```
fPos    = 0,
fAngl   = 0,
bManUp  = FALSE,
bManDwn = FALSE,
bManMod = FALSE,
bActv   = FALSE;
```

Since the blind function block FB_BA_SunBldActr [► 307] or the roller blind function block FB_BA_RolBldActr [► 298] does not take account of the flag *bActv*, this telegram would be interpreted as movement command to position "0", i.e. fully open. The absence of an active telegram therefore does not represent a safety risk for the blind.

🔌 Inputs

```
VAR_INPUT
    stSunBldTgm_1 : ST_BA_SunBld;
    stSunBldTgm_2 : ST_BA_SunBld;
    stSunBldTgm_3 : ST_BA_SunBld;
    stSunBldTgm_4 : ST_BA_SunBld;
END_VAR
```


Name	Type	Description
stSunBldTgm_1... stSunBldTgm_4	ST_BA_SunBld [► 217]	Telegram inputs

🔌 Outputs

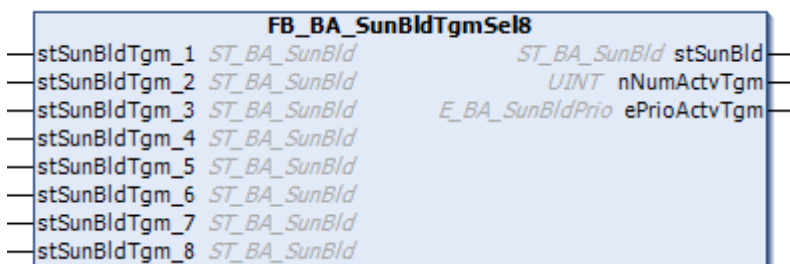
```
VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  nNumActvTgm   : UINT;
  ePrioActvTgm  : E_BA_SunBldPrio;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
nNumActvTgm	UINT	Indicates which input is valid, e.g. if <i>stSunBldTgm_3</i> is passed, <i>nNumActvTgm</i> = 3. If <i>nNumActvTgm</i> = 0, no telegram is active.
ePrioActvTgm	E_BA_SunBldPrio	This output indicates the priority of the active telegram.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.27 FB_BA_SunBldTgmSel8



The function block **FB_BA_SunBldTgmSel8** is used for priority control for up to 8 positioning telegrams (*stSunBld_Prio1* ... *stSunBld_Prio8*) of type **ST_BA_SunBld** [► 217] from different control function blocks.

The active telegram with the highest priority is output at *stSunBld*. Active means that the variable *bActv* is set within the structure of the positioning telegram. The priority is stored within the telegram structure as *ePrio*. The smaller the value of *ePrio*, the higher the priority.

For telegrams with the same priority, the last changed one (last writer wins) is valid, determined by the variable *nEvtInc*.

This function block is to be programmed in such a way that one of the applied telegrams is always active. If a telegram is not active, an empty telegram is output at the output:

```
fPos    = 0,
fAngl   = 0,
bManUp  = FALSE,
bManDwn = FALSE,
bManMod = FALSE,
bActv   = FALSE;
```

Since the blind function block **FB_BA_SunBldActr** [► 307] or the roller blind function block **FB_BA_RolBldActr** [► 298] does not take account of the flag *bActv*, this telegram would be interpreted as movement command to position "0", i.e. fully open. The absence of an active telegram therefore does not represent a safety risk for the blind.

Inputs

```
VAR_INPUT
    stSunBldTgm_1 : ST_BA_SunBld;
    stSunBldTgm_2 : ST_BA_SunBld;
    stSunBldTgm_3 : ST_BA_SunBld;
    stSunBldTgm_4 : ST_BA_SunBld;
    stSunBldTgm_5 : ST_BA_SunBld;
    stSunBldTgm_6 : ST_BA_SunBld;
    stSunBldTgm_7 : ST_BA_SunBld;
    stSunBldTgm_8 : ST_BA_SunBld;
END_VAR
```

Name	Type	Description
stSunBldTgm_1... stSunBldTgm_8	ST_BA_SunBld [► 217]	Telegram inputs

Outputs

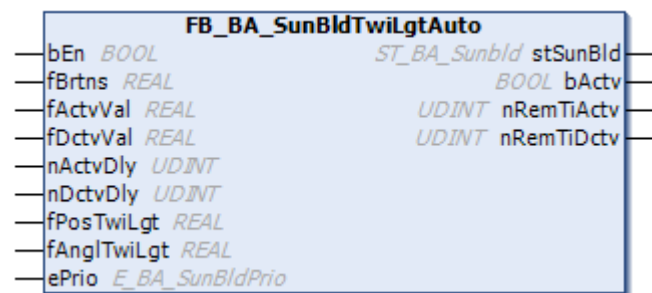
```
VAR_OUTPUT
    stSunBld : ST_BA_SunBld;
    nNumActvTgm : UINT;
    ePrioActvTgm : E_BA_SunBldPrio;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
nNumActvTgm	UINT	Indicates which input is valid, e.g. if <i>stSunBldTgm_3</i> is passed, <i>nNumActvTgm</i> = 3. If <i>nNumActvTgm</i> = 0, no telegram is active.
ePrioActvTgm	E_BA_SunBldPrio	This output indicates the priority of the active telegram.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.28 FB_BA_SunBldTwiLgtAuto



The function block FB_BA_SunBldTwiLgtAuto controls the blind when the outdoor brightness has fallen below a limit value.

The twilight automatic works with a value and a time hysteresis: If the outdoor brightness value *fBrtns* [lx] falls below the value *fActvVal* [lx] for the time *nActvDly* [s], then the function block is active and will provide the blind positions *fPosTwiLgt* (height [%]) and *fAnglTwiLgt* (slat angle [°]) specified at the input variables at the output in the positioning telegram *ST_BA_Sunbld* [► 217]. Conversely, if the outdoor brightness exceeds the value *fDctvVal* [lx] for the time *nDctvDly* [s], the automatic function is no longer active. The active flag in the positioning telegram *ST_BA_Sunbld* [► 217] is reset and the positions for height and angle are set to "0". A function with a lower priority can then take over control.

Inputs

```
VAR_INPUT
  bEn          : BOOL;
  fBrtns       : REAL;
  fActvVal     : REAL;
  fDctvVal     : REAL;
  nActvDly     : UDINT;
  nDctvDly     : UDINT;
  fPosTwilgt   : REAL;
  fAnglTwilgt  : REAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	The function block has no function if this input is FALSE. 0 is output for the position and the angle in the positioning telegram <u>ST_BA_SunBld</u> [► 217] - <i>bManMod</i> and <i>bActv</i> are set to FALSE. This means that another function takes over control of the blind via the priority controller. Conversely, a direct connection allows the blind to drive directly to the 0 position, i.e. fully up, since the actuator function block does not evaluate the bit <i>bActv</i> itself.
fBrtns	REAL	Outdoor brightness [lx].
fActvVal	REAL	Activation limit value [lx]. The value <i>fActvVal</i> is internally limited to values from 0 to <i>rDctvVal</i> .
fDctvVal	REAL	Deactivation limit value [lx]. Internally limited to a minimum value of 0.
nActvDly	UDINT	Activation delay [s]. Internally limited to a minimum value of 0.
nDctvDly	UDINT	Deactivation delay [s]. Internally limited to a minimum value of 0.
fPosTwilgt	REAL	Vertical position of the blind [%] if the twilight automatic is active. Internally limited to values between 0 and 100.
fAnglTwilgt	REAL	Slat angle of the blind [°] if the twilight automatic is active

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio       : E_BA_SunBldPrio := E_BA_SunBldPrio.eGroupTwilightAuto;
END_VAR
```

Name	Type	Description
ePrio	E_BA_SunBldPrio	Priority of the active program

Outputs

```
VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  bActv         : BOOL;
  nRemTiActv    : UDINT;
  nRemTiDctv    : UDINT;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram <i>ST_BA_SunBld</i> [► 217] and is solely used to indicate whether the function block sends an active telegram.
nRemTiActv	UDINT	Shows the time remaining [s] after falling below the switch value <i>fActvVal</i> until automatic mode is activated. This output is 0 as long as no countdown of the time is taking place.
nRemTiDctv	UDINT	Shows the time remaining [s] after exceeding of the switch value <i>fDctvVal</i> until automatic mode is disabled. This output is 0 as long as no countdown of the time is taking place.

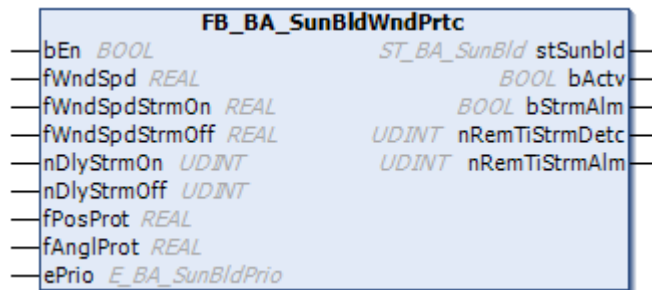
i If an error occurs, this automatic control is disabled and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see [Overview](#) [► 268]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

i If an error occurs, this automatic control is disabled and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see [Overview](#) [► 268]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.29 FB_BA_SunBldWndPrtc



The function block *FB_BA_SunBldWndPrtc* deals with the direction-dependent wind protection.

The weather protection has the highest priority in the blind controller (see [overview](#) [► 268]) and is intended to ensure that the blind is not damaged by ice or wind.

If the measured wind speed lies above the value *fWndSpdStrmOn* for the time *nDlyStrmOn* [s], then it is assumed that a storm is directly impending. Only if the wind speed falls below the value *fWndSpdStrmOff* for the time *nDlyStrmOff* [s] is the storm considered to have abated and the driving of the blind considered to be safe. For safety reasons the storm event is also persistently saved. Thus, if the controller fails during a storm, the sequence timer is started again from the beginning when the controller is restarted.

In wind hazard cases, the blind is moved to the protection position specified by *fPosProt* (height position [%]) and *fAnglProt* (slat angle [°]).

Inputs

```
VAR_INPUT
  bEn          : BOOL;
  fWndSpd      : REAL;
  fWndSpdStrmOn : REAL;
  fWndSpdStrmOff : REAL;
  nDlyStrmOn   : UDINT;
  nDlyStrmOff  : UDINT;
  fPosProt     : REAL;
  fAnglProt    : REAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	The function block has no function if this input is FALSE. 0 is output for the position and the angle in the positioning telegram ST_BA_SunBld [► 217] - <i>bManMod</i> and <i>bActv</i> are set to FALSE. For a connection with priority controller this means that another functionality takes over control of the blind.
fWndSpd	REAL	Wind speed. The unit of the entry is arbitrary, but it is important that no value is smaller than 0 and that the values become larger with increasing speed.
fWndSpdStrmOn	REAL	Wind speed limit value for the activation of the storm alarm. This value may not be smaller than 0 and must lie above the value for the deactivation. Otherwise an error is output. The unit of the entry must be the same as that of the input <i>fWndSpd</i> . A value greater than this limit value triggers the alarm after the entered time <i>nDlyStrmOn</i> .
fWndSpdStrmOff	REAL	Wind speed limit value for the deactivation of the storm alarm. This value may be not smaller than 0 and must lie below the value for the activation. Otherwise an error is output. The unit of the entry must be the same as that of the input <i>fWndSpd</i> . A value smaller than or equal to this limit value resets the alarm after the entered time <i>nDlyStrmOff</i> .
nDlyStrmOn	UDINT	Time delay until the storm alarm is triggered [s].
nDlyStrmOff	UDINT	Time delay until the storm alarm is reset [s].
fPosProt	REAL	Height position of the blind [%] in the case of protection.
fAnglProt	REAL	Slat angle of the blind [°] in the case of protection

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio      : E_BA_SunBldPrio := E_BA_SunBldPrio.eStorm;
END_VAR
```

Name	Type	Description
ePrio	E_BA_SunBldPrio	Priority of the active program

Outputs

```
VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  bActv         : BOOL;
  bStrmAlm      : BOOL;
  nRemTiStrmDetc : UDINT;
  nRemTiStrmAlm  : UDINT;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram <i>ST_BA_SunBld</i> [► 217] and is solely used to indicate whether the function block sends an active telegram.
bStrmAlm	BOOL	Indicates the storm alarm
nRemTiStrmDetc	UDINT	In an uncritical case this second counter constantly indicates the alarm delay time <i>nDlyStrmOn</i> . If the measured wind speed <i>fWndSpd</i> is above the activation limit value <i>fWndSpdStrmOn</i> , the seconds to the alarm are counted down. This output is 0 as long as no countdown of the time is taking place.
nRemTiStrmAlm	UDINT	As soon as the storm alarm is initiated, this second counter initially constantly indicates the deactivation time delay of the storm alarm <i>nDlyStrmOff</i> . If the measured wind speed <i>fWndSpd</i> falls below the deactivation limit value <i>fWndSpdStrmOff</i> , the seconds to the all-clear signal (<i>bStrmAlm</i> =FALSE) are counted down. This output is 0 as long as no countdown of the time is taking place

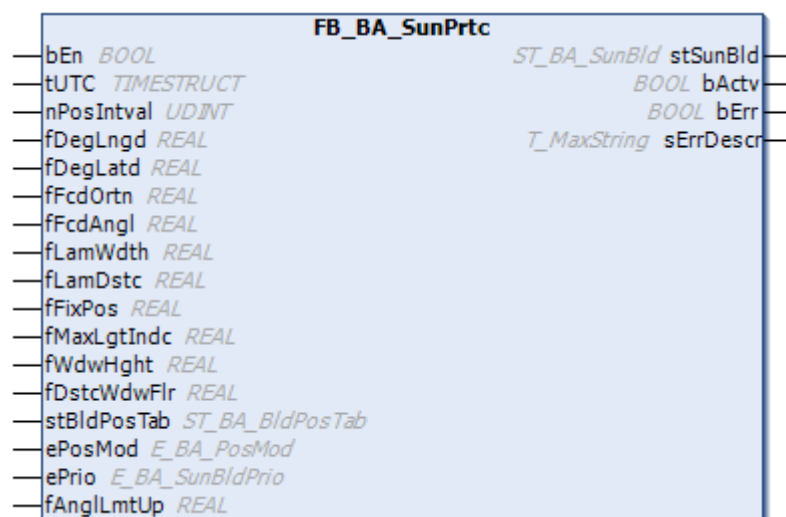


If an error occurs, this automatic control is disabled and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see [Overview](#) [► 268]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.30 FB_BA_SunPrtc



The function block *FB_BA_SunPrtc* is used for glare protection with the aid of a slatted blind.

Glare protection is realized through variation of the slat angle and positioning of the blind height.

The slat angle is set as a function of the sun position such that direct glare is prevented, while letting as much natural light through as possible.

Three different operation modes are available for varying the blind height.

1. When sun protection is active, the blind moves to a fixed height. The height value is specified with the variable *fFixPos*.
2. The blind position is varied as a function of the position of the sun. The position is specified in the table (ST_BA_BldPosTab [► 216]). See also description of FB_BA_BldPosEntry [► 474].
3. The high of the blind is calculated based on the window geometry such that the sun's rays reach a specified depth in the room. The incidence depth of the sun's rays is defined with the variable *fMaxLgtIndc*.

In order to avoid excessive repositioning of the slat angle, the variable *nPosIntval* [min] can be used to specify a time interval, within which the slat angle is not adjusted. In order to avoid glare, the angle is always changed sufficiently for the respective time interval.

The following conditions must be met for positioning the blind and setting the slat angle.

- 1. The input *bEn* must be TRUE.
- 2. The sun must have risen. (elevation > 0)
- 3. The function block is parameterized correctly (*bErr* = False)

Inputs

```
VAR_INPUT
  bEn          : BOOL;
  tUTC         : TIMESTRUCT;
  nPosIntval   : UDINT;
  fDegLngd     : REAL;
  fDegLatd     : REAL;
  fFcdOrtn     : REAL;
  fFcdAngl     : REAL;
  fLamWdth     : REAL;
  fLamDstc     : REAL;
  fFixPos      : REAL;
  fMaxLgtIndc  : REAL;
  fWdwHght     : REAL;
  fDstcWdwFlr  : REAL;
  stBldPosTab  : ST_BA_BldPosTab;
  ePosMod      : E_BA_PosMod := E_BA_PosMod.eFix;
END_VAR
```

Name	Type	Description
bEn	BOOL	If this input is set to FALSE, the positioning is inactive, i.e. the active bit (<i>bActv</i>) is reset in the positioning structure <i>stSunBld</i> of the type <i>ST_BA_SunBld</i> [► 217] and the function block itself remains in a standstill mode. If on the other hand the function block is activated, then the active bit is TRUE and the function block outputs its control values (<i>fPos</i> , <i>fAngl</i>) in the positioning structure at the appropriate times.
stUTC	TIMESTRUCT	Input of current time as UTC - Coordinated Universal Time (previously referred to as GMT, Greenwich Mean Time) (see TIMESTRUCT). The function block <i>FB_BA_GetTime</i> [► 531] can be used to read this time from a target system.



A jump of more than 300 seconds leads to immediate repositioning, if the blind is in the sun and glare protection is active, based on the above criteria. This functionality was added to ensure a reproducible program execution.

Name	Type	Description
nPosIntval	UDINT	Positioning interval in minutes - time between two blind position outputs. Valid range: 1 min...720 min.
fDegLngd	REAL	Longitude [°]. Valid range: - 180°...180°.
fDegLatd	REAL	Latitude [°]. Valid range: - 90°...90°.
fFcdOrtn	REAL	Facade orientation [°]:

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):

Viewing direction	Facade orientation
North	$\beta=0^\circ$
East	$\beta=90^\circ$
South	$\beta=180^\circ$
West	$\beta=270^\circ$

The following applies for the southern hemisphere:

Viewing direction	Facade orientation
South	$\beta=0^\circ$
East	$\beta=90^\circ$
North	$\beta=180^\circ$
West	$\beta=270^\circ$

Name	Type	Description
fCcdAngl	REAL	Facade inclination [°] (see Facade inclination [► 272]).
fLamWdth	REAL	Width of the slats in mm (see sketch [► 269]).
fLamDstc	REAL	Slat spacing in mm (see sketch [► 269]).
fFixPos	REAL	Fixed (constant) blind height [0...100%]. Applies if <i>ePosMod</i> = <i>ePosModFix</i> (see E_BA_PosMod [► 598]).
fMaxLgtIndc	REAL	Maximum desired light incidence in mm measured from the outside of the wall (see height adjustment [► 471]). The parameters <i>fWdwHght</i> and <i>fDstcWdwFlr</i> are used to calculate how high the blinds must be, depending on the position of the sun, such that the incidence of light does not exceed the value <i>fMaxLgtIndc</i> . Applies if <i>ePosMod</i> = <i>ePosModeMaxIncidence</i> (see E_BA_PosMod).
fWdwHght	REAL	Window height in mm for the calculation of the blind height if the mode "maximum desired incidence of light" is selected.
fDstcWdwFlr	REAL	Distance between the floor and the window sill in mm for the calculation of the blind height if the mode "maximum desired incidence of light" is selected.
stBldPosTab	ST_BA_BldPosTab	Table of 6 interpolation points, 4 of which are parameterizable, from which a blind position is then given in relation to the position of the sun by linear interpolation. Applies if <i>ePosMod</i> = <i>ePosModFix</i> (see E_BA_PosMod). For a more detailed description please refer to FB_BA_BldPosEntry [► 474].
ePosMo	E_BA_PosMod	Selection of the positioning mode

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
    ePrio      : E_BA_SunBldPrio := E_BA_SunBldPrio.eSunProtection;
    fAnglLmtUp : REAL;
END_VAR
```

Name	Type	Description
ePrio	E_BA_SunBldPrio	Priority of the active program
fAnglLmtUp	REAL	Slats do not need to be opened above 0°.

Outputs

```

VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  bActv         : BOOL;
  bErr          : BOOL;
  sErrorDescr   : T_MAXSTRING;
END_VAR

```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Output telegram, for the position and angle of the slat.
bActv	BOOL	Corresponds to the boolean value <i>bActv</i> in the blind telegram ST_BA_SunBld [► 217] and is solely used to indicate whether the function block sends an active telegram.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Error: The duration of the positioning interval is less than or equal to zero, or it exceeds 720 min.
02: Error: The value entered for the longitude is not in the valid range of -180°...180°.
03: Error: The value entered for the latitude is not in the valid range of -90°...90°.
04: Error: The value entered for the facade inclination <i>fFcdAngl</i> is outside the valid range of -90°..90°.
05: Error: The value for the slat spacing (<i>fLamDstc</i>) is greater than or equal to the value for the slat width (<i>fLamWdth</i>). This does not represent a "valid" blind, since the slats cannot close fully. Mathematically, this would lead to errors.
06: Error: The value entered for the slat width <i>fLamWdth</i> is zero.
07: Error: The value entered for the slat spacing <i>fLamDstc</i> is zero.
08: Error: The value entered for the fixed blind height (<i>fFixPos</i>) is greater than 100 or less than 0. At the same time, positioning "fixed blind height" is selected - <i>ePosMod</i> = <i>ePosModFix</i> .
09: Error: The "Values valid" bit (<i>bVld</i>) in the <i>stBldPosTab</i> positioning table is not set - invalid values, see FB_BA_BldPosEntry. At the same time, "Table" positioning is selected – <i>ePosMod</i> = <i>ePosModTab</i> .
10: Error: The value entered for the maximum required light incidence <i>fMaxLgtIndc</i> is less than or equal to zero. At the same time, "maximum light incidence" is selected – <i>ePosMod</i> = <i>ePosModMaxIndc</i> .
11: Error: The value entered for the window height <i>fWdwHght</i> is less than or equal to zero. At the same time, "maximum light incidence" is selected – <i>ePosMod</i> = <i>ePosModMaxIndc</i> .
12: Error: The distance between lower window edge and floor <i>fDstcWdwFlr</i> that was entered is less than zero. At the same time, "maximum light incidence" is selected – <i>ePosMod</i> = <i>ePosModMaxIndc</i> .
13: Error: An invalid positioning mode is entered at input <i>ePosMod</i> .



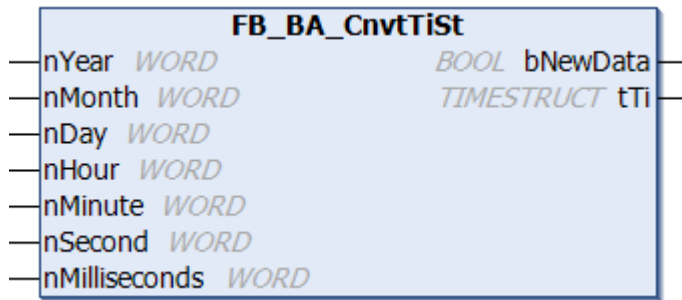
If an error occurs, this automatic control is disabled and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see [Overview \[► 268\]](#)) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.4 System

6.1.2.2.3.1.4.1 FB_BA_CnvtTiSt



The function block FB_BA_CnvtTiSt can be used to combine the individual components of a time structure into a single structure.



The function block does not check for incorrect entries, such as an hour entry of 99. It makes sense to check this in the connected function blocks, which have to check the time structure in any case. The limit values are shown as part of the variable explanations.

Inputs

```
VAR_INPUT
    nYear      : WORD;
    nMonth     : WORD;
    nDay       : WORD;
    nHour      : WORD;
    nMinute    : WORD;
    nSecond    : WORD;
    nMilliseconds : WORD;
END_VAR
```

Name	Type	Description
nYear	WORD	The year (1970...2106).
nMonth	WORD	The month (1...12).
nDay	WORD	The day of the month (1...31).
nHour	WORD	The hour (0...23).
nMinute	WORD	The minutes (0...59).
nSecond	WORD	The seconds (0...59).
nMilliseconds	WORD	The milliseconds (0...999).

Outputs

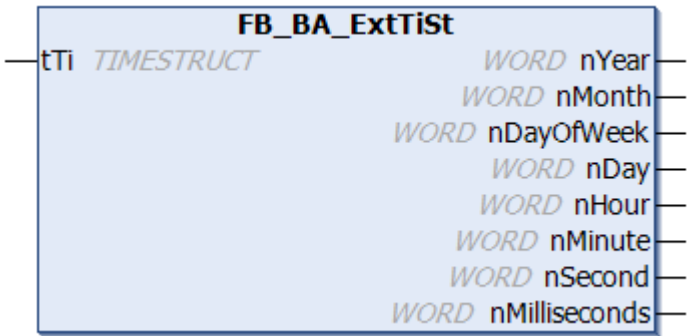
```
VAR_OUTPUT
    bNewData : BOOL;
    stTi     : TIMESTRUCT;
END_VAR
```

Name	Type	Description
bNewData	BOOL	The output is TRUE in the cycle in which the input variables have changed.
stTi	TIMESTRUCT	Output time structure

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.4.2 FB_BA_ExtTiSt



The function block FB_BA_ExtTiSt resolves a time structure into the different components, so that it can be used for time conditions, for example.

 Inputs

```
VAR_INPUT
    tTi      : Timestruct;
END_VAR
```

Name	Type	Description
tTi	Timestruct	Input time structure

 VAR_OUTPUT

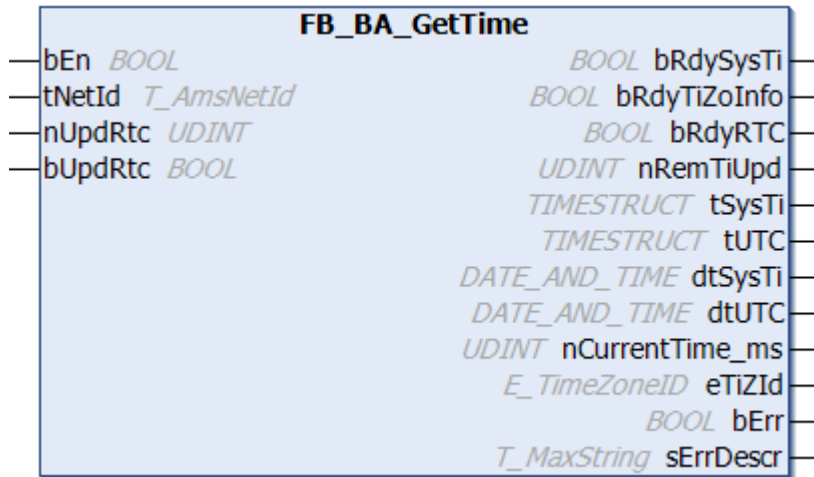
```
VAR_OUTPUT
    nYear      : WORD;
    nMonth     : WORD;
    nDayOfWeek : WORD;
    nDay       : WORD;
    nHour      : WORD;
    nMinute    : WORD;
    nSecond    : WORD;
    nMilliseconds : WORD;
END_VAR
```

Name	Type	Description
nYear	WORD	The year (1970...2106).
nMonth	WORD	The month (1...12).
nDayOfWeek	WORD	The day of the week (0 (Sun)...0 (Sat)).
nDay	WORD	The day of the month (1...31).
nHour	WORD	The hour (0...23).
nMinute	WORD	The minutes (0...59).
nSecond	WORD	The seconds (0...59).
nMilliseconds	WORD	The milliseconds (0...999).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.4.3 FB_BA_GetTime



The function block FB_BA_GetTime can be used to implement an internal clock (Real Time Clock RTC) in the TwinCAT PLC. When the function block is enabled via *bEn*, the RTC clock is initialized with the current NT system time. One system cycle of the CPU is used to calculate the current RTC time. The function block must be called once per PLC cycle in order for the current time to be calculated. Internally, an instance of the function blocks NT_GetTime, FB_GetTimeZoneInformation and RTC_EX2 is called in the function block. The time is output at the outputs *tSysTi* for the read system time and *tUtcTi* for the Coordinated Universal Time (UTC). This is determined internally from the system time and the time zone. If the system time and/or the time zone was entered incorrectly, the UTC time will also be wrong.

The system time is read cyclically via the timer to be set (*nUpdRTC* [sec]); it is used to synchronize the internal RTC clock. The time information (time zone, time shift relative to UTC, summer/winter time) is read in the same cycle. The output *nRemTiUpd* indicates the seconds remaining to the next read cycle. The time structures that are output, *dtSysTi* and *dtUtc*, can be resolved with the aid of the function block FB_BA_ExtTiSt into the components day, month, hour, minute etc.

Information on the read/wait cycle

During the read cycle, the outputs *bRdySysTi* and *bRdyTiZoInfo* change to FALSE, and the enumerator *eTiZId* shows 0 = *eTimeZoneID_Unknown*. If the read operation was successful, the outputs switch back to TRUE or show the respective information for summer or winter time, if available. If the read operation was unsuccessful - internally the system waits for a response for 5 seconds - the outputs remain at FALSE or 0, and another wait cycle is started before the next read cycle. Although the internal RTC clock is not synchronized in the event of an error and may still show the right time, the time information may be wrong, and therefore also the UTC time. Errors during the read cycle will, any case, show up in *bErr* and *sErrDescr*. The countdown output *nRemTiUpd* is not restarted until the wait cycle starts.

Inputs

```
VAR_INPUT
    bEn          : BOOL;
    tNetId       : T_AmsNetId;;
    nUpdRtc      : UDINT;
    bUpdRtc      : BOOL;
END_VAR
```

Name	Type	Description
bEn	BOOL	Enables the function block. If <i>bEn</i> = TRUE, then the RTC clock is initialized with the NT system time.
tNetId	T_AmsNetId	This parameter can be used to specify the AmsNetID of the TwinCAT computer, whose NT system time is to be read as timebase. If it is to be run on the local computer, an empty string can be entered.
nUpdRtc	UDINT	Time specification [s] with which the RTC clock is regularly synchronized with the NT system time. Internally this value is limited to a minimum of 5 seconds, in order to ensure correct processing of the internal function blocks.
bUpdRtc	BOOL	In parallel with the time <i>nUpdRtc</i> , the RTC clock can be synchronized via a positive edge at this input

Outputs

```

VAR_OUTPUT
  bRdySysTi      : BOOL;
  bRdyTiZoInfo   : BOOL;
  bRdyRTC        : BOOL;
  nRemTiUpd      : UDINT;
  tSysTi         : TIMESTRUCT;
  tUTC           : TIMESTRUCT;
  dtSysTi        : DT;
  dtUTC          : DT;
  nCurrentTime_ms : UDINT
  eTiZId         : E_TimeZoneID;
  bErr           : BOOL;
  sErrDescr      : T_MAXSTRING;
END_VAR

```

Name	Type	Description
bRdySysTi	BOOL	The system time was read successfully from the target system
bRdyTiZoInfo	BOOL	The additional time information (time zone, time shift relative to UTC and summer/winter time) was read successfully.
bRdyRTC	BOOL	This output is set if the function block has been initialized at least once. If this output is set, then the values for date, time and milliseconds at the outputs are valid.
nRemTiUpd	UDINT	Countdown to next synchronization/update of the time information.
stSysTi	TIMESTRUCT	System time of the read target system. The time structure can be resolved with the aid of the function block FB_BA_ExtTiSt into its components: day, month, hour, minute etc. Info: If the function block is not enabled (<i>bEn</i> = FALSE), the output <i>stSysTi</i> and its subelements (day month, etc.) show 0.
tUTC	TIMESTRUCT	Coordinated Universal Time. This is determined internally from the system time and the time information read from the target system. The time structure can be resolved with the aid of the function block FB_BA_ExtTiSt into its components: day, month, hour, minute etc. Info: If the function block is not enabled (<i>bEn</i> =FALSE), the output <i>tUTC</i> and its subelements (day month, etc.) show 0.
dtSysTi / dtUTC	DT	Same as <i>stSysTi</i> / <i>stUTC</i> , but in DATE-AND-TIME format: year-month-day-hour-minute-seconds. Info: If the function block is not enabled (<i>bEn</i> = FALSE), the outputs <i>dtSysTi</i> and <i>dtUTC</i> each display DT#1970-01-01-00:00, since this is the lower limit and it corresponds to the zeros in the structure representation of <i>dtSysTi</i> / <i>dtUTC</i> .
nCurrentTime	UDINT	Current time of day [ms].
eTiZId	E_TimeZoneID	Enumerator for summer/winter time information.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Warning: ADS error when reading the time (NT_GetTime). The ADS error number is stated.

02: Warning: ADS error when reading the time zone information (FB_GetTimeZoneInformation). The ADS error number is stated.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.4.4 FB_BA_SetTime

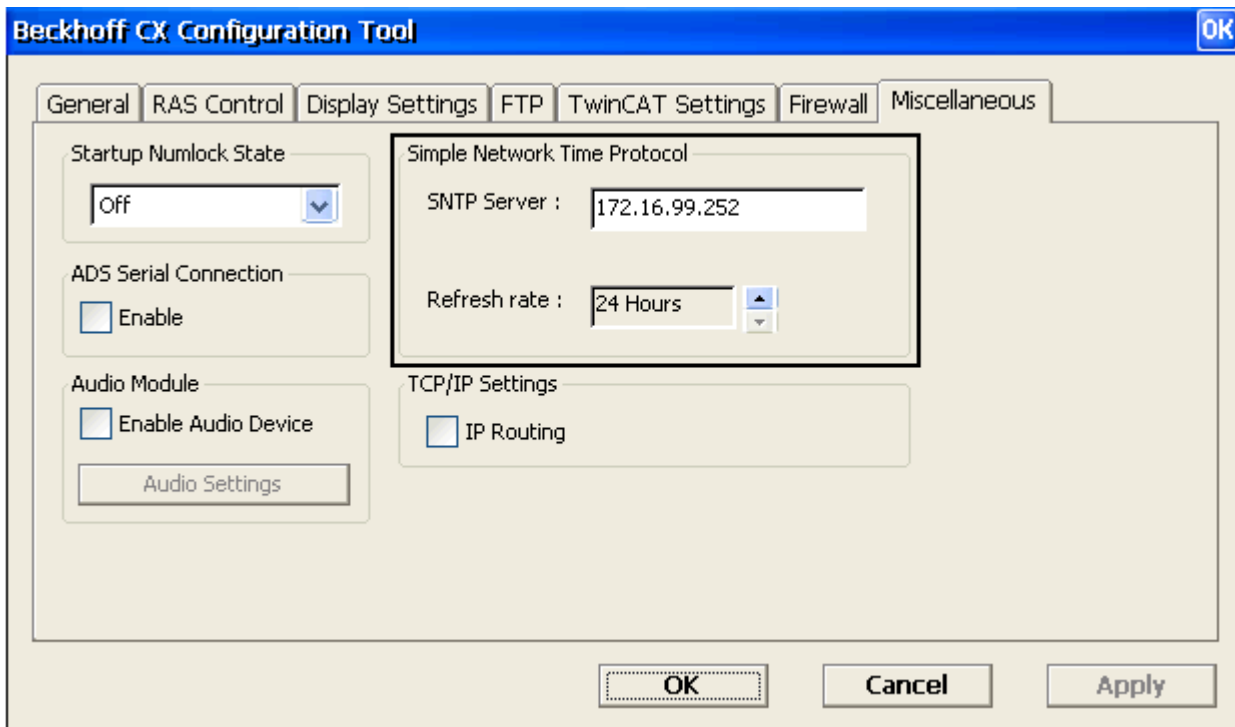


The function block FB_BA_SetTime can be used to set the local NT system time and the date for a TwinCAT system (the local NT system time is shown in the taskbar). The system time is specified via the structure *tSysTi*.

Internally, an instance of the function block NT_SetLocalTime from the TcUtilities library is called in the function block.



The local NT system time can also be synchronized with a reference time with the aid of the SNTP protocol. For further information please refer to the Beckhoff Information System under: Beckhoff Information System > Embedded PC > Operating systems > CE > SNTP: Simple Network Time Protocol



Inputs

```
VAR_INPUT
  bSet      : BOOL;
  tNetId    : T_AmsNetId;
  tSysTi    : TIMESTRUCT;
  nTiOut    : UDINT;
END_VAR
```

Name	Type	Description
bSet	BOOL	Activation of the function block with a rising edge.
tNetId	T_AmsNetId	This parameter can be used to specify the AmsNetID of the TwinCAT computer, whose local NT system time is to be set. If applicable, an empty string <i>sNetId</i> := ""; can be specified for the local computer.
tSysTi	TIMESTRUCT	Structure with the new local NT system time. If the time is not available as structure, it is advisable to use the function block FB_BA_CnvtTiSt [► 333], which brings the subvariables of date and time in a structure together.
nTiOut	UDINT	Indicates the timeout time [s], which must not be exceeded during execution.

🚀 Outputs

```
VAR_OUTPUT
  bBusy      : BOOL;
  bErr       : BOOL;
  sErrDescr  : T_MAXSTRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	If the function block is activated via a rising edge at <i>bSet</i> , this output is set and remains set until feedback occurs.
bErr	BOOL	This output is set to TRUE, if either the system time to be transferred is incorrect or an ADS error occurs during the transfer.
sErrDescr	T_MAXSTRING	Contains the error description

Error description

01: Error: Error range exceeded year

02: Error: Error range exceeded month

03: Error: Error range exceeded day of the month

04: Error: Error range exceeded hour

05: Error: Error range exceeded minute

06: Error: Error range exceeded second

07: Error: Error range exceeded millisecond

08: Warning: An ADS error occurred while setting the time (NT_SetLocalTime). The ADS error number is stated.

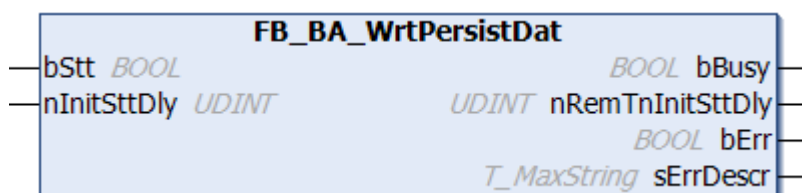
Time specification limits

The time structure *stUtcTi* that was created is internally checked for limits (see TIMESTRUCT)

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.4.5 FB_BA_WrtPersistDat



When activated, the function block FB_BA_WrtPersistDat first saves the persistent data in the Port_xxx.bootdata file. It is not necessary to explicitly specify the port or runtime system at which the PLC is located; this is determined internally. Once the data has been written, the content of the file Port_xxx.bootdata is copied to the backup file Port_xxx.bootdata-old. Thus both files are always synchronized. In case the original file with the persistent data is not readable, the backup copy, which is then read, contains the same data.



In any case, the checkmark for "Clear Invalid Persistent Data" must be removed (see Description of persistent data handling under TwinCAT3).

The function block can be started in two ways:

Via a positive edge at input *bStt*, if the function block is not in the set start-up phase.

Initially once the start-up phase is completed after a reset or TwinCAT restart. The duration is set at *nInitSttDly* in seconds. If "0" is entered there, the duration of the start-up phase is 0 and an initial execution of the function block is skipped.

No commands are accepted at *bStt* during the start-up phase.

If errors occur while reading, writing, opening or closing the files, this is indicated by a corresponding error message at *bErr/sErrDescr*. After an internally fixed waiting time of two seconds, the function block automatically attempts to execute the command (read, write, open or close) again.

It is therefore advisable to keep an eye on the error outputs or to evaluate them.

It is also important to note whether the backup file for the persistent data was loaded during the TwinCAT restart or after a reset. This indicates that the original file cannot be read and that the memory card of the controller is defective. It can be queried for each runtime system with the boolean assignment of *TwinCAT_SystemInfoVarList._AppInfo.OldBootData* (see *PlcAppSystemInfo*).

Sample in ST:

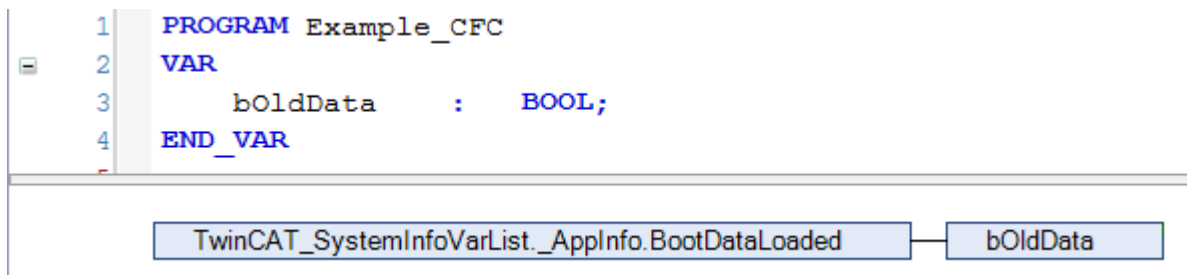
```

1  PROGRAM Example_ST
2  VAR
3      bOldData      :   BOOL;
4  END_VAR

1  bOldData:=TwinCAT_SystemInfoVarList._AppInfo.OldBootData;

```

Sample in CFC:



NOTICE

File handle conflicts

Make sure that only this function block and only one instance of it accesses the persistent data. If several function blocks open a file and do not close it again, unforeseen file handle conflicts can occur which cannot be intercepted. The persistent data will then no longer be updated in the xxx.bootdata file.

Description of persistent data handling under TwinCAT 3

TwinCAT saves the persistent data for each runtime system in a file during each orderly shutdown, i.e. when switching from Run to Config or Stop mode.

The file name consists of the ADS port name of the runtime system with the file extension .bootdata, e.g.: *Port_851.bootdata* and is stored in the TwinCAT directory under *TwinCAT\3.1\Boot\PLC*.

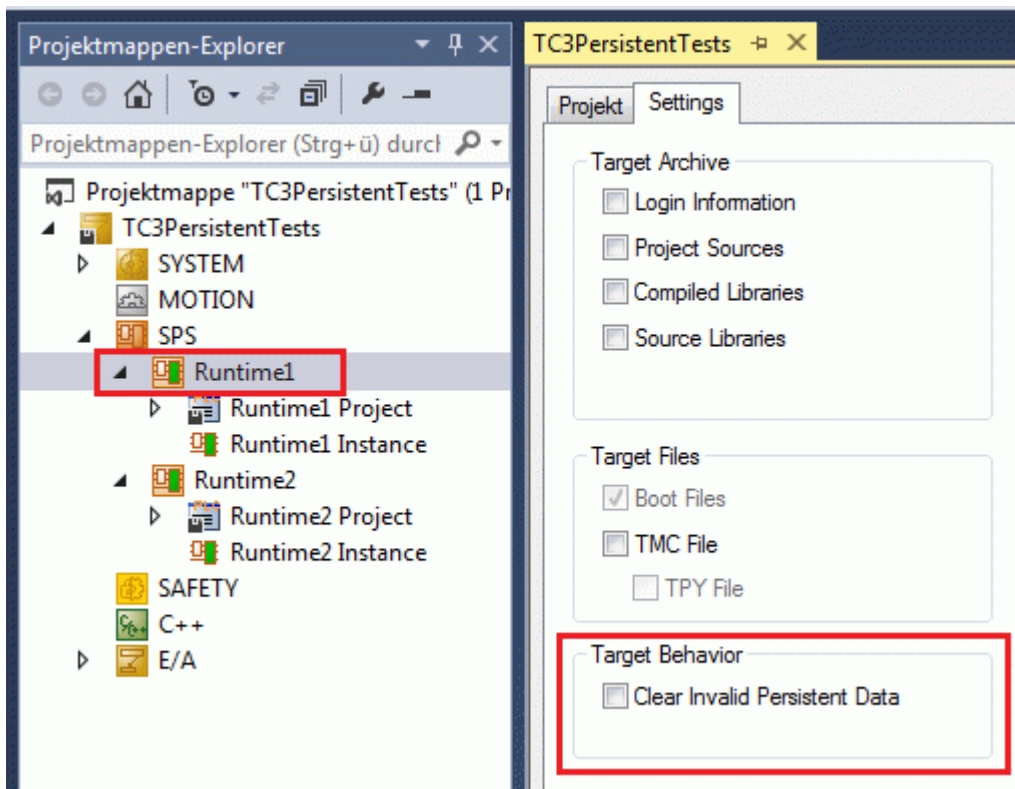
When the system is restarted, i.e. when switching to run mode, this file is read and then saved as *Port_xxx.bootdata-old*.

If the file *Port_xxx.bootdata-old* already exists, it is overwritten.

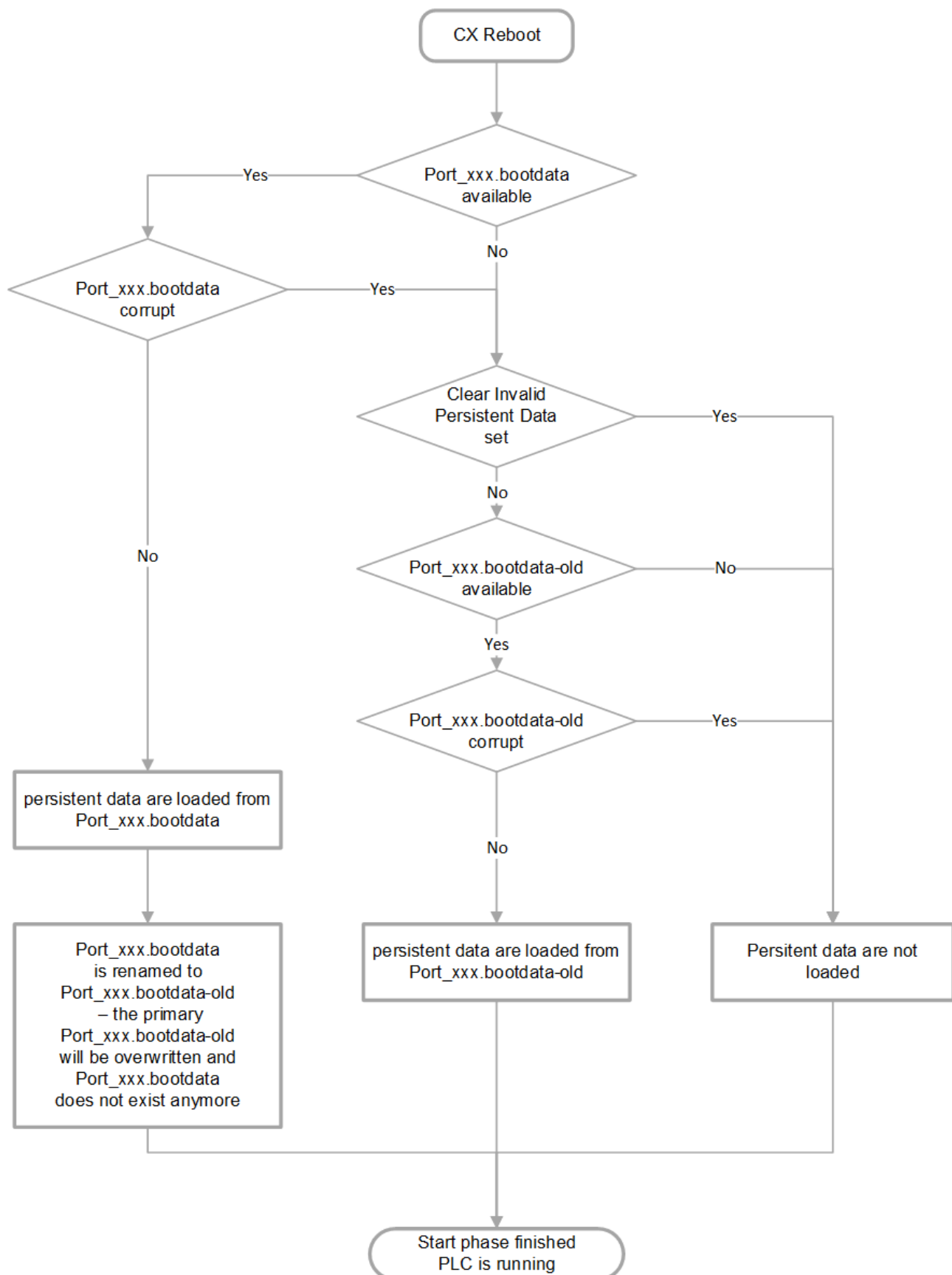
The original file *Port_xxx.bootdata* then no longer exists. It is created again automatically when switching to Stop mode or by the function block *FB_WritePersistentData* from the *TC2_Uilities* library.

This behavior applies to each runtime system; each system has its own files with persistent data.

If the file is defective when the TwinCAT system is restarted, the system automatically accesses the backup file *Port_xxx.bootdata-old*. However, this behavior only applies if the **Clear Invalid Persistent Data** checkmark is unchecked in the runtime settings. If it is checked and the original file is defective, no data will be read.



The *Port_xxx.bootdata-old* backup file is also used when the controller is de-energized. In this case, too, the current persistent data is not stored in *Port_xxx.bootdata*. When the system is restarted, only the old data is available, unless a more up-to-date file was created by the function block FB_WritePersistDat before the system was switched off.



Inputs

```

VAR_INPUT
  bStt          : BOOL;
  nInitSttDly   : UDINT;
END_VAR
  
```

Name	Type	Description
bStt	BOOL	A rising edge at this input starts the function block if it is not in the start-up phase.
nInitSttDly	UDINT	Start-up phase after a reset or TwinCAT restart. The duration is set in seconds. Once the start-up phase has elapsed, the function block is automatically started once. No commands are accepted for <i>bStt</i> during the start-up phase. If "0" is set at <i>nInitSttDly</i> , the start-up phase is skipped. This input is preconfigured with 10 s.

Outputs

```

VAR_OUTPUT
  bBusy          : BOOL;
  nRemTiInitSttDly : UDINT;
  bErr           : BOOL;
  sErrDescr      : T_MaxString;
END_VAR

```

Name	Type	Description
bBusy	BOOL	The function block is being executed.
nRemTiInitSttDly	UDINT	Countdown of the set start-up phase.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.
sErrDescr	T_MAXSTRING	Contains the error description.

Error description

01: Error: The number of the ADS port issued by the PLC is "0".

02: Warning: Error when writing the persistent data via the internal function block *FB_WritePersistentData*. Additionally its error number.

03: Warning: Error when opening the backup file (xxx.bootdata-old) via the internal function block *FB_FileOpen*. Additionally its error number.

04: Warning: Error when reading the original file (xxx.bootdata) via the internal function block *FB_FileRead*. Additionally its error number.

05: Warning: Error when writing the backup file (xxx.bootdata-old) via the internal function block *FB_FileWrite*. Additionally its error number.

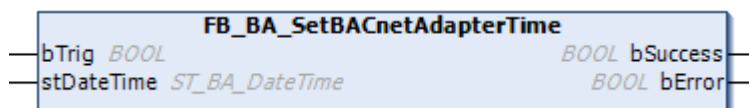
06: Warning: Error when closing the original file (xxx.bootdata) via the internal function block *FB_FileClose*. Additionally its error number.

07: Warning: Error when closing the backup file (xxx.bootdata-old) via the internal function block *FB_FileClose*. Additionally its error number.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.4.6 FB_BA_SetBACnetAdapterTime



The function block can be used to set the local BACnet system time of a TwinCAT system.

Illustration

```

FUNCTION_BLOCK FB_BA_SetBACnetAdapterTime
VAR_INPUT
  bTrig      : BOOL;
  stDateTime : ST_BA_DateTime;

```

```

END_VAR
VAR_OUTPUT
    bSuccess      : BOOL;
    bError        : BOOL;
END_VAR

```

Inputs

Name	Type	Description
bTrig	BOOL	On a rising edge the content of the stDateTime structure is written to the local BACnet system time of the TwinCAT system.
stDateTime	ST_BA_DateTime	Time specification which is written to the local BACnet system time.

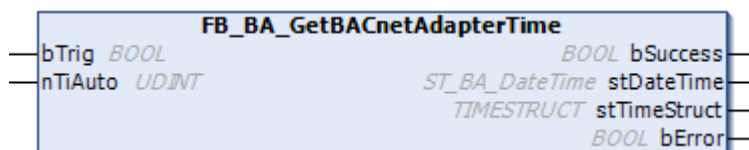
Outputs

Name	Type	Description
bSuccess	BOOL	The time specification <i>stDateTime</i> was successfully written to the target system. The variable is TRUE for one cycle.
bError	BOOL	Error while writing the BACnet system time.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.4.7 FB_BA_GetBACnetAdapterTime



The function block can be used to determine the local BACnet system time of a TwinCAT system.

Internally, an internal software clock is implemented by means of the function block "RTC_EX2" (Extended Real Time Clock). This software clock is initialized with the BACnet system time and output by the two structures stDateTime and stTimeStruct.

The function block should be called in every PLC cycle, so that the current time can be calculated.

Illustration

```

FUNCTION_BLOCK FB_BA_GetBACnetAdapterTime
VAR_INPUT
    bTrig      : BOOL;
    {attribute 'parameterUnit' := 'ms'}
    nTiAuto    : UDINT;
END_VAR
VAR_OUTPUT
    bSuccess      : BOOL;
    stDateTime     : ST_BA_DateTime;
    stTimeStruct   : TIMESTRUCT;
    bError        : BOOL;
END_VAR

```

🔧 Inputs

Name	Type	Description
bTrig	BOOL	A rising edge at this input determines the local BACnet system time of a TwinCAT system.
nTiAuto	UDINT	Based on this time information, the local BACnet system time of a TwinCAT system is regularly determined automatically. The time specification must be greater than or equal to 500ms.

🔌 Outputs

Name	Type	Description
bSuccess	BOOL	The system time was read successfully from the target system. The variable is TRUE for one cycle.
stDateTime	ST_BA_DateTime	Current BACnet system time of a TwinCAT system.
stTimeStruct	<u>TIMESTRUCT</u>	Current BACnet system time of a TwinCAT system.
bError	BOOL	Error while reading the BACnet system time.

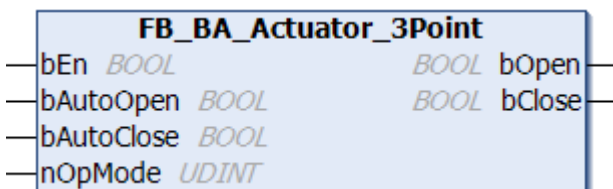
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5 Universal

6.1.2.2.3.1.5.1 Actuators

6.1.2.2.3.1.5.1.1 FB_BA_Actuator_3Point



The function block FB_BA_Actuator_3Point is used to control a 3-point actuator, e.g. a 3-point damper or a 3-point valve.

The command for opening the actuator is connected to output *bOpen*.

The command for closing the actuator is connected to output *bClose*.

In automatic mode (*nOpMode* = 0) the control commands of *bCmdOpen* and *bCmdClose* are forwarded directly to the outputs *bOpen* and *bClose*.

The *nOpMode* input is used to determine the operation mode of the 3-point actuator:

- 0 = Automatic
- 1 = Stop (*bOpen* = *bClose* = FALSE)
- 2 = Close
- 3 = Open

Inputs

```
VAR_INPUT
    bEn       : BOOL;
    bAutoOpen : BOOL;
    bAutoClose : BOOL;
    nOpMode   : UDINT
END_VAR
```

Name	Type	Description
bEn	BOOL	General enable of the function block.
bAutoOpen	BOOL	Command to open the actuator
bAutoClose	BOOL	Command to close the actuator.
nOpMode	UDINT	Select operation mode (0 = Automatic, 1 = Stop (bOpen = bClose = FALSE), 2 = Close, 3 = Open).

Outputs

```
VAR_OUTPUT
    bOpen : BOOL;
    bClose : BOOL;
END_VAR
```

Name	Type	Description
bOpen	BOOL	Open control output
bClose	BOOL	Close control output

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.1.2 FB_BA_Anlg3Pnt



The function block FB_BA_Anlg3Pnt is intended for control of three-point actuators for valves or dampers. A continuous control signal for positioning an actuator is converted into the binary commands for opening and closing.

If the deviation between the position setpoint fIn and the calculated actual position value $fPos$ of the actuator is greater than the set threshold $fHys / 2$, the function block starts to correct the position by switching the outputs $bOpn$ or $bCls$, depending on the amount of the control deviation:

	bOpn	bCls
$fIn - fPos > fHys / 2$	TRUE	FALSE
$fIn - fPos < - fHys / 2$	FALSE	TRUE

If the function block reaches an end position $fOut = 0$ or $fOut = 100$ through a corresponding input value fIn , the corresponding switching output remains permanently set in order to safely reach this end position at the valve or damper:

	bOpn	bCls
fOut = 0	FALSE	permanently TRUE
fOut = 100	permanently TRUE	FALSE

Any deactivation of the continuous signal must be implemented by the user through external programming.

The input *fIn* is automatically limited internally to the range of 0...100 %.

This also applies to the inputs *fHys* and *fRefVal*. The travel times *nTiCls* as well as *nTiOpn* are both limited downwards to 10 (milliseconds).

A rising edge at *bRef* triggers a referencing command (setting the calculated actual position to *fRefVal*).

If the drive has limit switches, these can also be detected directly by means of a digital input and used for referencing at *bRef*.

Inputs

```
VAR_INPUT
    fIn      : REAL;
    fHys     : REAL;
    nTiCls   : UDINT;
    nTiOpn   : UDINT;
    bRef     : BOOL;
    fRefVal  : REAL;
    bCloseInit : BOOL;
END_VAR
```

Name	Type	Description
fIn	REAL	Setpoint for the actuator position [0...100 %]. Internally limited to values between 0 and 100.
fHys	REAL	Hysteresis for the actuator position [0...100 %]. Internally limited to values between 0 and 100.
nTiCls	UDINT	Run time of the actuator from open to closed [ms]. Internally limited to values between 0 and 100.
nTiOpn	UDINT	Run time of the actuator from closed to open [ms]. Internally limited to values between 0 and 100.
bRef	BOOL	Edge references the internal position memory of the drive to value of <i>fRefVal</i> [0...100 %].
fRefVal	REAL	Value for referencing the actuator with <i>bRef</i> [0...100 %]. Internally limited to values between 0 and 100.
bCloseInit	BOOL	If this input is TRUE, output <i>bCls</i> is TRUE for the time <i>udiTiOpn_ms</i> .

Outputs

```
VAR_OUTPUT
    bCls     : BOOL;
    bOpn     : BOOL;
    fPos     : REAL;
END_VAR
```

Name	Type	Description
bCls	BOOL	Output for closing the actuator.
bOpn	BOOL	Output for opening the actuator.
fPos	REAL	Current calculated actuator position [0...100 %].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.1.3 FB_BA_AntBlkg



This function block FB_BA_AntBlkg prevents blocking of pumps or actuators after prolonged idle periods by issuing a switch-on pulse.

Generally, a pulse output only occurs if the function block at *bEn* is enabled.

The maximum idle period before such a pulse is issued is determined by the value of the variable *nOffMin*. For logging the idle time, the input *bFdb* must be linked to the operating feedback from the aggregate. The length of the pulse is parameterized with the variable *nImplLngt*. For this function the operation mode *E_BA_AntBlkgMode.eOffTime* must be set.

The input *bExtReq* should be used if the blocking protection pulse is to be issued cyclically based on a schedule, rather than depending on the idle times. A rising edge at *bExtReq* immediately triggers output of a pulse to *bQ*. For this function the operation mode *E_BA_AntBlkgMode.eExternalRequest* must be set.

Inputs

```
VAR_INPUT
    bEn          : BOOL;
    bFdb         : BOOL;
    bExtReq      : BOOL;
    bLock        : BOOL;
    nOffMin      : UDINT;
    nImplLngt    : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	<i>bEn</i> is the general enable of the function block. If <i>bEn</i> is FALSE, the message output <i>bQ</i> is also FALSE.
bFdb	BOOL	Input for connecting the feedback signal of a motor or valve. This input is only considered in the operation mode <i>E_BA_AntBlkgMode.eOffTime</i> .
bExtReq	BOOL	Active in the <i>E_BA_AntBlkgMode.eExternalRequest</i> operation mode. External request for a pulse, for example from a schedule. With a rising edge the blocking protection pulse is started.
bLock	BOOL	Active in the operation modes <i>E_BA_AntBlkgMode.eExternalRequest</i> or <i>E_BA_AntBlkgMode.eOffTime</i> . To prevent that e.g. the pump and the valve of a heater get a pulse at the same time, the output of the pulse is always suppressed until <i>bLock</i> is FALSE again. If <i>bLock</i> becomes TRUE during the output of a blocking protection pulse, then the blocking protection pulse is interrupted. After <i>bLock</i> is FALSE again, the blocking protection pulse is restarted.
nOffMin	UDINT	Minimum switch-off time of the actuator without movement of the motor or valve [s].
nImplLngt	UDINT	Length of the blocking protection pulse [s] at <i>bQ</i> .

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
    eMode      : E_BA_AntBlkgMode := E_BA_AntBlkgMode.eOffTime;
END_VAR
```

Name	Type	Description
eMode	E_BA_AntBlkgMode	Input that specifies the operation mode of the function block

E_BA_AntBlkgMode.eOff - the operation mode *eOff* is similar to the input *bEn*. If this operation mode is active, the pulse output *bQ* is FALSE.

E_BA_AntBlkgMode.eExternalRequest - External request for a pulse, for example from a schedule. With a rising edge the blocking protection pulse is started.

E_BA_AntBlkgMode.eOffTime - Minimum switch-off time of the actuator without movement of the motor or valve (*bFdb* = FALSE). After the timer has expired, the blocking protection pulse is started.

Outputs

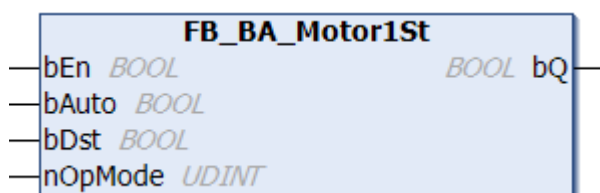
```
VAR_OUTPUT
    bQ          : BOOL;
    nRemOffMin   : UDINT;
    nRemImplLngt : UDINT;
END_VAR
```

Name	Type	Description
bQ	BOOL	Output for the blocking protection pulse.
nRemOffMin	UDINT	Remaining time [s] before the next pulse is issued in the absence of movement.
nRemImplLngt	UDINT	Remaining residual time [s] of the pulse at <i>bQ</i> .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.1.4 FB_BA_Motor1St



The function block *FB_BA_Motor1St* is used to control 1-step motors.

The input *bEn* is used for enabling the function block.

The input *nOpMode* is used to set the operation mode of the motor:

- 0 = Automatic
- 1 = Manual off
- 2 = Manual on

In automatic mode (*nOpMode* = 0) the motor can be operated via input *bAuto* (*bAuto* = *bQ* = TRUE).

The collection of all possible malfunctions of a motor is connected to *bDst*.

Inputs

```
VAR_INPUT
    bEn      : BOOL;
    bAuto    : BOOL;
    bDst     : BOOL;
    nOpMode  : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	Enable motor.
bAutoOpen	BOOL	Actuator request in automatic mode (<i>nOpMode</i> = 0).
bDst	BOOL	Input for collecting the possible motor malfunctions.
nOpMode	UDINT	Selection of the operation mode (0 = Automatic, 1 = Manual off, 2 = Manual on).

Outputs

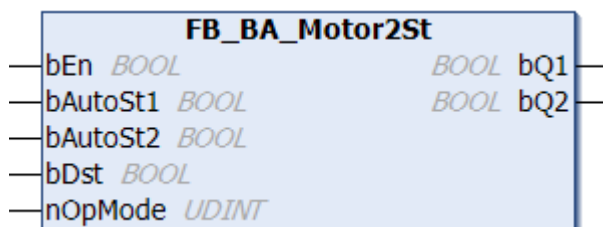
```
VAR_OUTPUT
    bQ : BOOL;
END_VAR
```

Name	Type	Description
bQ	BOOL	Control output

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.1.5 FB_BA_Motor2St



The function block FB_BA_Motor2St is used to control 2-step motors.

The input *bEn* is used for enabling the function block.

The input *nOpMode* is used to set the operation mode of the motor:

- 0 = Automatic
- 1 = Manual off
- 2 = Manual step 1
- 3 = Manual step 2

In automatic mode (*nOpMode* = 0) the desired step can be set via the inputs *bAutoSt1* (step 1) and *bAutoSt2* (step 2).

The collection of all possible malfunctions of a motor is connected to *bDst*.

Inputs

```
VAR_INPUT
    bEn      : BOOL;
    bAutoSt1 : BOOL;
    bAutoSt2 : BOOL;
```

```

bDst      : BOOL;
nOpMode   : UDINT;
END_VAR

```

Name	Type	Description
bEn	BOOL	Enable motor
bAutoSt1	BOOL	Request of the actuator to step 1 in automatic mode (<i>nOpMode</i> = 0).
bAutoSt2	BOOL	Request of the actuator to step 2 in automatic mode (<i>nOpMode</i> = 0).
bDst	BOOL	Input for collecting the possible motor malfunctions.
nOpMode	UDINT	Selection of the operation mode (0 = Automatic, 1 = Manual off, 2 = Manual step 1, 3 = Manual step 2).

Outputs

```

VAR_OUTPUT
bQ1 : BOOL;
bQ2 : BOOL;
END_VAR

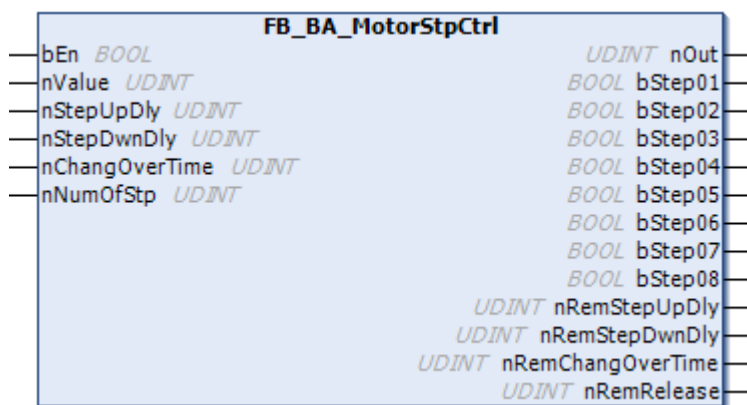
```

Name	Type	Description
bQ1	BOOL	Control output step 1
bQ2	BOOL	Control output step 2

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

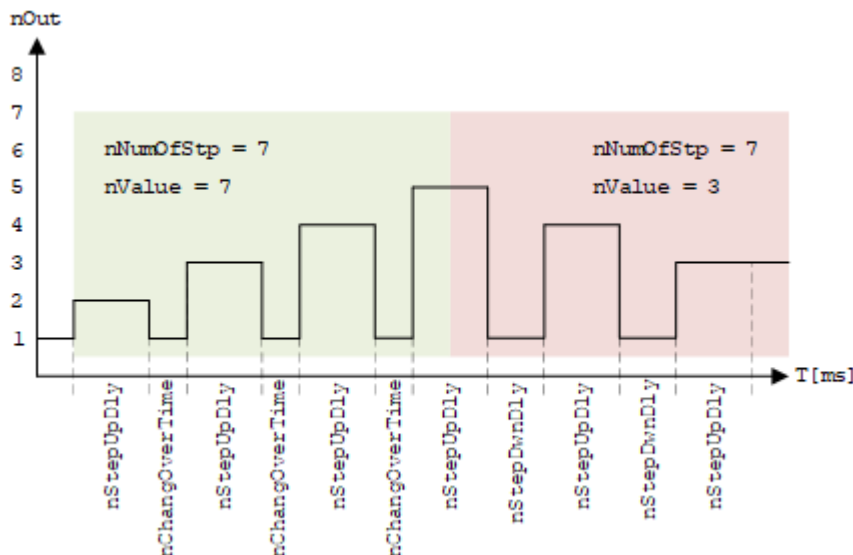
6.1.2.2.3.1.5.1.6 FB_BA_MotorStpCtrl



The function block **FB_BA_MotorStpCtrl** is used to control multi-stage drives. The program always starts at level 1 and, depending on the requirements of *nValue*, switches step by step to the next higher level. Switching up and down into the individual steps is influenced by the 3 time specifications *nStepUpDly*, *nStepDwnDly* and *nChangOverTime*.

In case of a restart or by removing the enable, a restart of the drive is inhibited for the time period of (*nOut* * *nStepDwnDly*). This time course is indicated by the output variable *nRemRelease*. Internally the last active state of *nOut* is persistently stored for the calculation of the blocking time.

Example



Error handling

The limitation of the input value $nNumOfStp$ is monitored and corrected internally.

$nNumOfStp < 1$ is adjusted to the value 1 and a detailed description is output via the *ErrorDescription* property.

$nNumOfStp > 9$ is adjusted to the value 9 and a detailed description is output via the *ErrorDescription* property.

In addition, a warning message is output in the Error List window of the TwinCAT programming tool.

The limitation of the input value $nValue$ is monitored and corrected internally.

$nValue < 1$ is adjusted to the value 1 and via the property *ErrorDescription* a detailed description is output.

$nValue > 9$ is adjusted to the value 9 and via the property *ErrorDescription* a detailed description is output.

In addition, a warning message is output in the Error List window of the TwinCAT programming tool.

Inputs

```
VAR_INPUT
    bEn      : BOOL;
    nValue   : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	General enable of the function block. If <i>bEn</i> is FALSE, then all output variables are FALSE or have the value 0.
nValue	UDINT	Step to be controlled from 1 to 9.

nValue	Request
1	Off
2	bStep01
3	bStep02
4	bStep03
5	bStep04
6	bStep05
7	bStep06
8	bStep07
9	bStep08

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
    nStepUpDly      : UDINT := 3000;
    nStepDwnDly     : UDINT := 1000;
    nChangOverTime  : UDINT := 100;
    nNumOfStp       : UDINT := 4;
END_VAR
```

Name	Type	Description
nStepUpDly	UDINT	Minimum switch-on time of the respective step [ms].
nStepDwnDly	UDINT	Switch-back time or switch-off time of the steps [ms].
nChangOverTime	UDINT	Time delay for the changeover phase when switching up [ms] between the steps in order to protect the motor windings. During this time, all outputs are FALSE and $nOut = 1$.
nNumOfStp	UDINT	Input of the number of steps required. The input is limited to a range from 1 to 9.

Outputs

```
VAR_OUTPUT
    nOut          : UDINT;
    bStep01       : BOOL;
    bStep02       : BOOL;
    bStep03       : BOOL;
    bStep04       : BOOL;
    bStep05       : BOOL;
    bStep06       : BOOL;
    bStep07       : BOOL;
    bStep08       : BOOL;
    nRemStepUpDly : UDINT;
    nRemStepDwnDly : UDINT;
    nRemChangOverTime : UDINT;
    nRemRelease   : UDINT;
END_VAR
```

Name	Type	Description
nOut	UDINT	Output of the currently valid step from 1 to 9.
bStep0N	BOOL	Output of step N depending on the stepped output signal $nOut$.
nRemStepUpDly	UDINT	Remaining time of the minimum switch-on time of the respective step [ms].
nRemStepDwnDly	UDINT	Remaining time of the switch-back time or switch-off time of the steps [ms].
RemChangOverTime	UDINT	Remaining time of the changeover time when switching up and down [ms].
nRemRelease	UDINT	Remaining time of the internal blocking of the function block after a restart or by removing the enable bEn [ms].

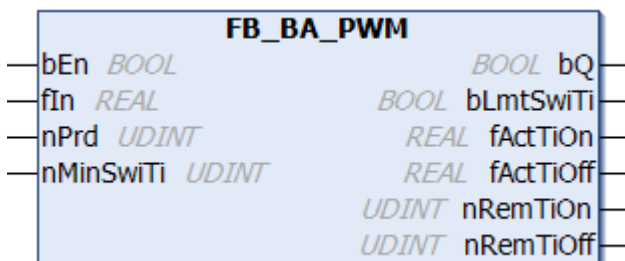
 Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [► 352] .
NumberOfSteps	UDINT	Get	Outputs the current or corrected value of the number of steps. In the error case of <i>nNumOfStp</i> there is a detailed description of the error at error handling [► 352] .
Step	STRING	Get	Output of the current step depending on <i>nOut</i> .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.1.7 FB_BA_PWM



The function block FB_BA_PWM calculates from an analog input signal *rln* (0...100 %, **internally fixed**) and the period duration *nPrd* [s] a switch-on and a switch-off time *fActTiOn* and *fActTiOff* [s].

The following relationships apply:

- 100% at the input of a switch-on time *fActTiOn* of the total period *nPrd* and a switch-off time *fActTiOff* of 0 s
- 0 % at the input of a switch-on time *fActTiOn* of 0 s and a switch-off time *fActTiOff* of the total period duration *nPrd*.

In addition, there is the possibility to limit the switching time downwards via *nMinSwiTi* [s] to avoid damage to drives by too short actuating pulses. This behavior is only valid for $0 > fln > 100!$

If *fln* = 0 or 100, the output *bQ* remains deleted or set. After the period time has elapsed, the current input signal is evaluated again. If it is still set to 0 or 100, there is no change of state of *bQ*.

Switching characteristics

1. A FALSE signal at input *bEn* disables the function block and sets *bQ* to FALSE. Only the switch-on and switch-off times are continuously calculated and displayed at the outputs *fActTiOn* / *fActTiOff* [s].
2. A rising edge at input *bEn* enables the function block: It will initially jump to a decision step. Depending on the previous state of the switching output *bQ*, the switching step is now accessed. However, if the input *fln* is set to 0, an immediate jump occurs to the Off step (*bQ*=FALSE), or to the On step if *fln*=100 (*bQ*=TRUE), irrespective of the previous state of *bQ*. The minimum switching time is deactivated for these two cases.
3. A countdown timer with the current calculated starting value runs in the respective active step (ON or OFF), which is based on the pulse/pause ratio. The on- or off-step is completed with the calculated time, irrespective of whether the pulse/pause ratio changes in the meantime. The respective countdown is displayed at the outputs *nRemTiOn* / *nRemTiOff* in full seconds.
4. Completion of the on- or off-step is followed by a jump back to the decision step (point 2).

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  fIn      : REAL;
  nPrd     : UDINT;
  nMinSwiTi : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	Activation of pulse width modulation.
fIn	REAL	Input signal, internally limited to 0...100%.
nPrd	UDINT	Period time[s]. Internally limited to a minimum value of 0.
nMinSwiTi	UDINT	Minimum switch-on time [s], to avoid too short pulses. Internally limited to values between 0 and <i>nPrd</i> .

Outputs

```
VAR_OUTPUT
  bQ      : BOOL;
  bLmtSwiTi : BOOL;
  fActTiOn : REAL;
  fActTiOff : REAL;
  nRemTiOn : UDINT;
  nRemTiOff : UDINT;
END_VAR
```

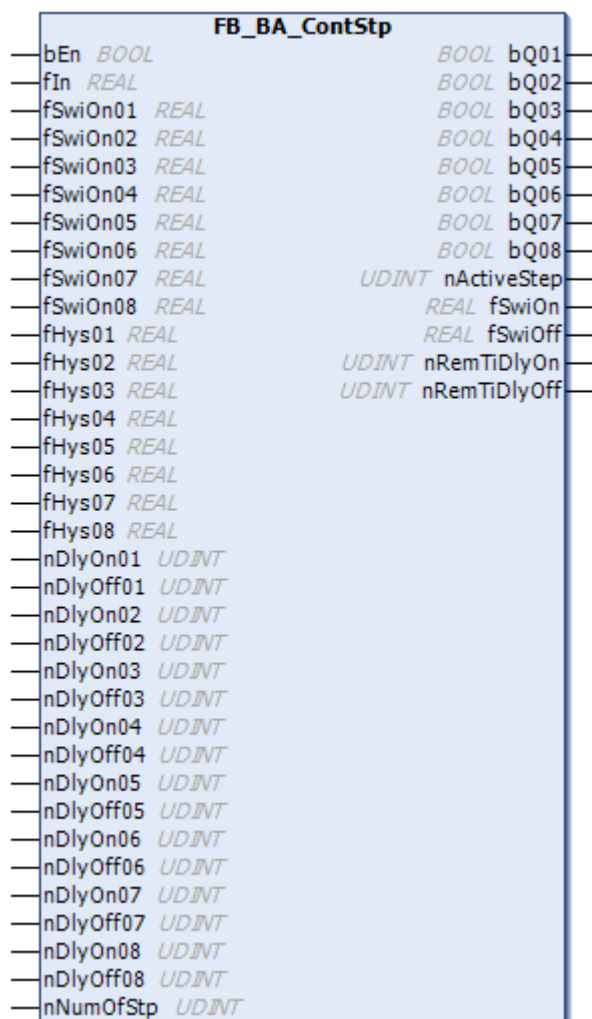
Name	Type	Description
bQ	BOOL	PWM output.
bLmtSwiTi	BOOL	Information output to indicate that the input signal is so low that the minimum switch-on time is used as limit.
fActTiOn	REAL	Information output: calculated switch-on time [s].
fActTiOff	REAL	Information output: calculated switch-off time [s].
nRemTiOn	UDINT	Switch-on timer countdown [s].
nRemTiOff	UDINT	Switch-off timer countdown [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

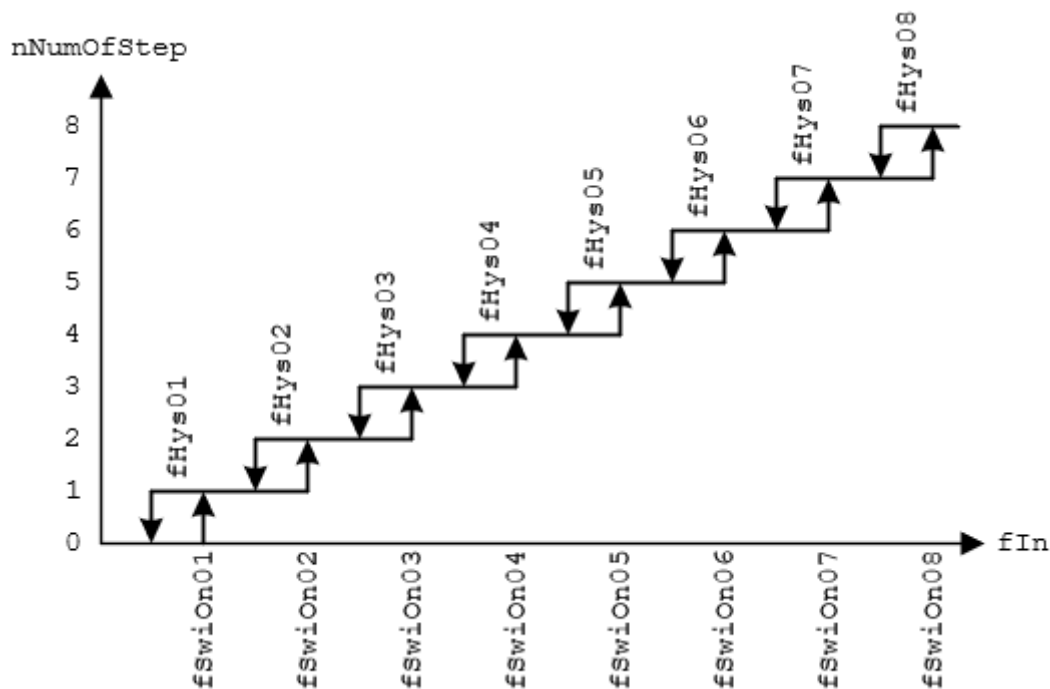
6.1.2.2.3.1.5.2 Control

6.1.2.2.3.1.5.2.1 FB_BA_ContStp



The function block FB_BA_ContStp determines the resulting control steps of a multi-level aggregate, depending on the continuous input signal *fIn*.

nActiveStep	nNumOfStep	fSwiOn	fSwiOff	nRemTiDlyOn	nRemTiDlyOff	bQ01	bQ02	bQ03	bQ04	bQ05	bQ06	bQ07	bQ08
0	0	fSwiOn01	fSwiOn01 - fHys01	nDlyOn01		FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	> = 1	fSwiOn02	fSwiOn01 - fHys01	nDlyOn02	nDlyOf01	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	> = 2	fSwiOn03	fSwiOn02 - fHys02	nDlyOn03	nDlyOf02	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	> = 3	fSwiOn04	fSwiOn03 - fHys03	nDlyOn04	nDlyOf03	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
4	> = 4	fSwiOn05	fSwiOn04 - fHys04	nDlyOn05	nDlyOf04	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
5	> = 5	fSwiOn06	fSwiOn05 - fHys05	nDlyOn06	nDlyOf05	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
6	> = 6	fSwiOn07	fSwiOn06 - fHys06	nDlyOn07	nDlyOf06	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
7	> = 7	fSwiOn08	fSwiOn07 - fHys07	nDlyOn08	nDlyOf07	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
8	8	fSwiOn08	fSwiOn08 - fHys08		nDlyOf08	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE



Inputs

```

VAR_INPUT
    bEn      : BOOL;
    fIn      : REAL;
    fSwiOn01 : REAL;

```

```

fSwiOn02    : REAL;
fSwiOn03    : REAL;
fSwiOn04    : REAL;
fSwiOn05    : REAL;
fSwiOn06    : REAL;
fSwiOn07    : REAL;
fSwiOn08    : REAL;
END_VAR

```

Name	Type	Description
bEn	BOOL	General enable of the function block. If <i>bEn</i> is FALSE, all message outputs <i>bQ0N</i> are also FALSE.
fIn	REAL	Continuous input value from which the switching states are derived.
fSwiOn0N	REAL	Switch-on point step 0N.

Inputs CONSTANT PERSISTENT

```

VAR_INPUT CONSTANT PERSISTENT
fHys01      : REAL := 5;
fHys02      : REAL := 5;
fHys03      : REAL := 5;
fHys04      : REAL := 5;
fHys05      : REAL := 5;
fHys06      : REAL := 5;
fHys07      : REAL := 5;
fHys08      : REAL := 5;
nDlyOn01    : UDINT;
nDlyOff01   : UDINT;
nDlyOn02    : UDINT;
nDlyOff02   : UDINT;
nDlyOn03    : UDINT;
nDlyOff03   : UDINT;
nDlyOn04    : UDINT;
nDlyOff04   : UDINT;
nDlyOn05    : UDINT;
nDlyOff05   : UDINT;
nDlyOn06    : UDINT;
nDlyOff06   : UDINT;
nDlyOn07    : UDINT;
nDlyOff07   : UDINT;
nDlyOn08    : UDINT;
nDlyOff08   : UDINT;
nNumOfStp   : UDINT := 4;
END_VAR

```

Name	Type	Description
fHys0N	REAL	Absolute value hysteresis step 0N.
nDlyOn0N	UDINT	Start-up delay step 0N.
nDlyOff0N	UDINT	Switch-off delay step 0N.
nNumOfStp	UDINT	Input of the number of steps required. The input is limited to a range from 0 to 8.

Outputs

```

VAR_OUTPUT
bQ01        : BOOL;
bQ02        : BOOL;
bQ03        : BOOL;
bQ04        : BOOL;
bQ05        : BOOL;
bQ06        : BOOL;
bQ07        : BOOL;
bQ08        : BOOL;
nActiveStep  : UDINT;
fSwiOn       : REAL;
fSwiOff      : REAL;
nRemTiDlyOn  : UDINT;
nRemTiDlyOff : UDINT;
END_VAR

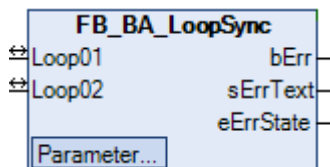
```

Name	Type	Description
bQ0N	BOOL	Shows state step 0N. The step can only be active when the preceding steps are TRUE. TRUE = ON; FALSE = OFF
nActiveStep	UDINT	Indicates how many steps are switched on.
fSwiOn	REAL	Indicates the next switch-on point.
fSwiOff	REAL	Indicates the next switch-off point.
nRemTiDlyOn	UDINT	If the switch-on point for switching to the next step is reached, the remaining time of the start-up delay is displayed here.
nRemTiDlyOff	UDINT	If the switch-off point for switching down to the next step is reached, the remaining time of the switch-off delay is displayed here.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.2.2 FB_BA_LoopSync



The function block is used for parameter synchronization for 2 controllers of type FB_BA_Loop [► 154]. This function block can be used, for example, to calibrate master controllers that are used to generate the supply air setpoints for an air conditioning system.

The following parameters of the FB_BA_Loop [► 154] are synchronized:

- eOpMode
- eActionRm
- fProportionalConstant
- fIntegralConstant
- fDerivativeConstant
- fMaxOutputRm
- fMinOutputRm
- nDampConstant
- fNeutralZone

Error detection

The error messages listed below are detected by the *FB_BA_LoopSync*.

The error messages are output in the TwinCAT 3 development environment in the "Error list" window. This can be activated under the menu item View.

The error texts are output via the property *ErrText* and the output *sErrText*.

In addition, the messages are displayed by the enum *eErrState*.

Error messages

Message text German	Message text English	Explanation
'Synchronisation <i>eActionRm</i> fehlerhaft'	'Synchronization <i>eActionRm</i> faulty'	Possibly the wrong synchronization can be triggered by the input <i>eActionPgm</i> . This input overwrites the VAR_INPUT CONSTANT variable <i>eActionRm</i> to be monitored when occupied.
'Synchronisation <i>nDampConstant</i> fehlerhaft'	'Synchronization <i>nDampConstant</i> faulty'	
'Synchronisation <i>fDerivativeConstant</i> fehlerhaft'	'Synchronization <i>fDerivativeConstant</i> faulty'	
'Synchronisation <i>fIntegralConstant</i> fehlerhaft'	'Synchronization <i>fIntegralConstant</i> faulty'	
'Synchronisation <i>fMaxOutputRm</i> fehlerhaft'	'Synchronization <i>fMaxOutputRm</i> faulty'	Possibly the wrong synchronization can be triggered by the input <i>fMaxOutputPgm</i> . This input overwrites the VAR_INPUT CONSTANT variable <i>fMaxOutputRm</i> to be monitored when occupied.
'Synchronisation <i>fMinOutputRm</i> fehlerhaft'	'Synchronization <i>fMinOutputRm</i> faulty'	Possibly the wrong synchronization can be triggered by the input <i>fMinOutputPgm</i> . This input overwrites the VAR_INPUT CONSTANT variable <i>fMinOutputRm</i> to be monitored when occupied.
'Synchronisation <i>fNeutralZone</i> fehlerhaft'	'Synchronization <i>fNeutralZone</i> faulty'	
'Synchronisation <i>eOpMode</i> fehlerhaft'	'Synchronization <i>eOpMode</i> faulty'	
'Synchronisation <i>fProportionalConstant</i> fehlerhaft'	'Synchronization <i>fProportionalConstant</i> faulty'	

Illustration

```

FUNCTION_BLOCK FB_BA_LoopSync
VAR_OUTPUT
    bErr      : BOOL;
    sErrText  : T_MaxString;
    eErrState : E_BA_StateLoopSync;
END_VAR
VAR_IN_OUT
    Loop01    : FB_BA_Loop;
    Loop02    : FB_BA_Loop;
END_VAR

```

 Outputs

Name	Type	Description
bErr	BOOL	The output indicates when an error has occurred during synchronization.
sErrText	T_MaxString	The variable shows the state of synchronization in <u>text form</u> [► 360].
eErrState	E_BA_StateLoopSync [► 214]	The enumeration shows the state of the synchronization.

🔧 / 📡 Inputs Outputs

Name	Type	Description
Loop1	FB_BA_Loop [► 154]	Reference to controller no. 1 of the parameter adjustment.
Loop2	FB_BA_Loop [► 154]	Reference to controller no. 2 of the parameter adjustment.

📄 Properties

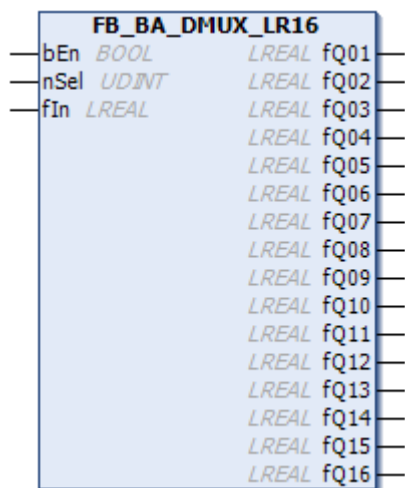
Name	Type	Access	Description
ErrText	T_MaxString	Get	The property <i>ErrText</i> displays the error texts from <i>sErrText</i> .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3 General control functions

6.1.2.2.3.1.5.3.1 FB_BA_DMUX_XX



Demultiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different output values (4, 8, 12 and 16), but they all have the same functionality. The function block FB_BA_DMUX_LR16 is described as an example.

The function block FB_BA_DMUX_XX outputs in the activated state (*bEn*= TRUE) the value at input *fIn* to the output *fQ01*..*fQ16* whose number is entered at input *nSel*. All other outputs are set to 0 (for boolean demultiplexers to FALSE).

Example:

Inputs	Outputs
bEn = TRUE	fQ01 = 0.0
nSel = 5	fQ02 = 0.0
fIn = 32.5	fQ03 = 0.0
	fQ04 = 0.0
	fQ05 = 32.5
	fQ06 = 0.0
	fQ07 = 0.0
	fQ08 = 0.0
	fQ09 = 0.0
	fQ10 = 0.0
	fQ11 = 0.0
	fQ12 = 0.0
	fQ13 = 0.0
	fQ14 = 0.0
	fQ15 = 0.0
	fQ16 = 0.0

If the value entered at *nSel* is greater than the number of outputs, the value of *fIn* is output at the "highest" output:

Inputs	Outputs
bEn = TRUE	fQ01 = 0.0
nSel = 25	fQ02 = 0.0
fIn = 32.5	fQ03 = 0.0
	fQ04 = 0.0
	fQ05 = 0.0
	fQ06 = 0.0
	fQ07 = 0.0
	fQ08 = 0.0
	fQ09 = 0.0
	fQ10 = 0.0
	fQ11 = 0.0
	fQ12 = 0.0
	fQ13 = 0.0
	fQ14 = 0.0
	fQ15 = 0.0
	fQ16 = 32.5

If *bEn* = FALSE, 0.0 is output at all outputs, or FALSE for boolean demultiplexers.

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  nSel     : UDINT;
  fIn      : LREAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	Activation of the block function
nSel	UDINT	Number of the output <i>fQ01...fQ16</i> , which is to assume the value of input <i>fIn</i> .
fIn	LREAL	Value to be output.

 Outputs

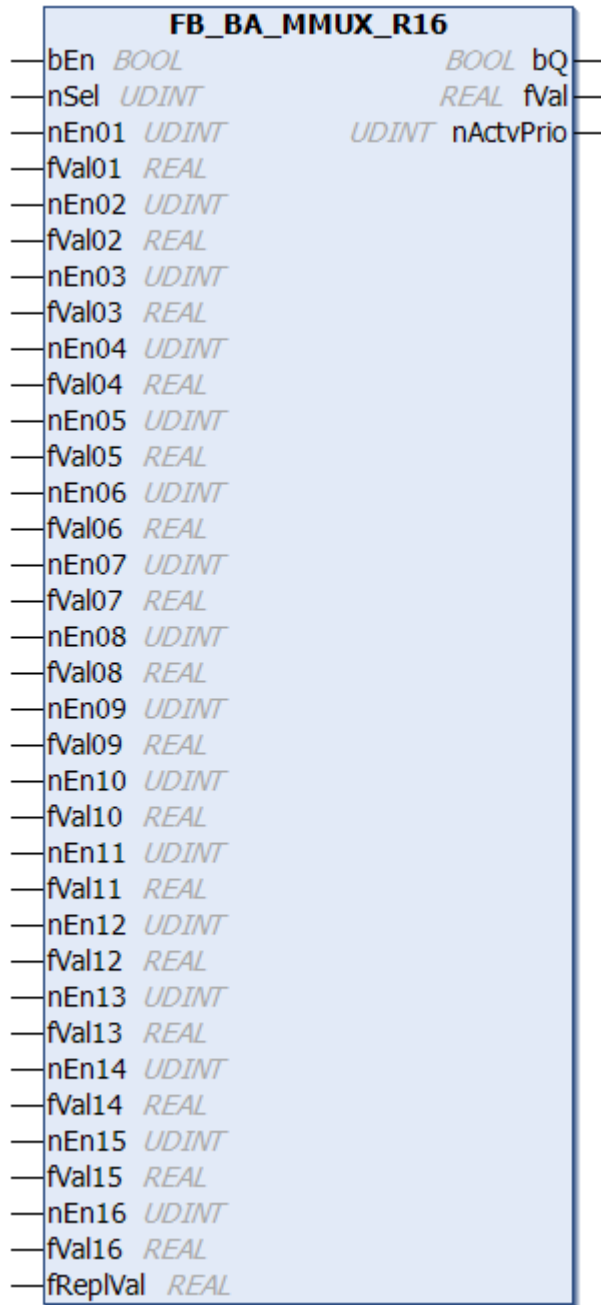
```
VAR_OUTPUT
  fQ01 : LREAL;
  fQ02 : LREAL;
  fQ03 : LREAL;
  fQ04 : LREAL;
  fQ05 : LREAL;
  fQ06 : LREAL;
  fQ07 : LREAL;
  fQ08 : LREAL;
  fQ09 : LREAL;
  fQ10 : LREAL;
  fQ11 : LREAL;
  fQ12 : LREAL;
  fQ13 : LREAL;
  fQ14 : LREAL;
  fQ15 : LREAL;
  fQ16 : LREAL;
END_VAR
```

Name	Type	Description
fQ01...fQ16	LREAL	Value outputs

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.2 FB_BA_MMUX_XX



The function block FB_BA_MMUX_XX activates an input value on the output, depending on a selector and the corresponding input selector condition.

Multiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different input values (4, 8, 12, 16 and 24), but they all have the same functionality. The function block FB_BA_MMUX_R16 is described as an example.

The function block switches one of the input values *rValxx* to the output *fVal* in the activated state (*bEn*=TRUE) depending on a selector *nSel* and the corresponding input selector condition *nEnxx*. If several input selector conditions *nEn01...nEn16* are equal and the selector *nSel* matches a condition, then the input value *fVal01...fVal16* of the lowest active selector condition is switched to the output *fVal*. *nEn01* is the lowest, *nEn16* the highest selector condition.

The output variable *bQ* indicates that the selector *nSel* matches an input selector condition *nEnxx*. The output variable *nActvPrio* indicates the active selector condition. If no selector condition is active, *fReplVal* is output to *fVal*. *bQ* is then FALSE and *nActvPrio* indicates a 255.

Example:

Inputs		Output	
Variable	Value	Variable	Value
bEn	TRUE	bQ	TRUE
nSel	5	fVal	1.123
nEn01	4	nActvPrio	7
fVal01	123		
nEn02			
fVal02			
nEn03	3		
fVal03	321		
nEn04			
fVal04			
nEn05	8		
fVal05	345		
nEn06			
fVal06			
nEn07	5		
fVal07	1.123		
nEn08			
fVal08			
nEn09	5		
fVal09	5.4321		
nEn10			
fVal10			
nEn11			
fVal11			
nEn12			
fVal12			
nEn13			
fVal13			
nEn14			
fVal14			
nEn15			
fVal15			
nEn16			
fVal16			
fReplVal			

If no active priority is present, then the value of the global constant BA_Globals.nNoActivePrio is output at the output *nActvPrio*.

Inputs

```

VAR_INPUT
  bEn      : BOOL;
  nSel     : UDINT;
  nEn01    : UDINT := BA_Globals.nNoActvPrio;
  fVal01   : REAL;
  nEn02    : UDINT := BA_Globals.nNoActvPrio;
  fVal02   : REAL;
  nEn03    : UDINT := BA_Globals.nNoActvPrio;
  fVal03   : REAL;
  nEn04    : UDINT := BA_Globals.nNoActvPrio;
  fVal04   : REAL;
  nEn05    : UDINT := BA_Globals.nNoActvPrio;
  fVal05   : REAL;

```

```

nEn06      : UDINT := BA_Globals.nNoActvPrio;
fVal06     : REAL;
nEn07      : UDINT := BA_Globals.nNoActvPrio;
fVal07     : REAL;
nEn08      : UDINT := BA_Globals.nNoActvPrio;
fVal08     : REAL;
nEn09      : UDINT := BA_Globals.nNoActvPrio;
fVal09     : REAL;
nEn10      : UDINT := BA_Globals.nNoActvPrio;
fVal10     : REAL;
nEn11      : UDINT := BA_Globals.nNoActvPrio;
fVal11     : REAL;
nEn12      : UDINT := BA_Globals.nNoActvPrio;
fVal12     : REAL;
nEn13      : UDINT := BA_Globals.nNoActvPrio;
fVal13     : REAL;
nEn14      : UDINT := BA_Globals.nNoActvPrio;
fVal14     : REAL;
nEn15      : UDINT := BA_Globals.nNoActvPrio;
fVal15     : REAL;
nEn16      : UDINT := BA_Globals.nNoActvPrio;
fVal16     : REAL;
fReplVal   : REAL;
END_VAR

```

Name	Type	Description
bEn	BOOL	Activation of the block function
nSel	UDINT	Selector. Internally limited to values between 0 and 4294967294.
nEn01...nEn16	UDINT	Input values to select from.
fReplVal	REAL	Substitute value, if no input selector condition is active.

Outputs

```

VAR_OUTPUT
  bQ      : BOOL;
  fVal    : REAL;
  nActvPrio : UDINT;
END_VAR

```

Name	Type	Description
bQ	BOOL	TRUE if the selector <i>nSel</i> matches an input selector condition <i>nEnxx</i> .
fVal	REAL	Value of the selected input selector condition.
nActvPrio	UDINT	Indicates which input selector condition is active. If no active priority is present, then the value of the global constant BA_Globals.nNoActivePrio is output at the output <i>nActvPrio</i> .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.3 FB_BA_MUX_XX



Multiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different input values (4, 8, 12 and 16), but they all have the same functionality. The function block FB_BA_MUX_LR16 is described as an example.

The function block FB_BA_MUX_XX outputs in the activated state ($bEn=TRUE$) that input value $fIn01..fIn16$ at output fQ whose number is entered at input $nSel$.

Example:

Inputs	Output
$bEn = TRUE$	$fQ = 16.5$
$nSel = 5$	
$fIn01 = 15.9$	
$fIn02 = 32.5$	
$fIn03 = 17.4$	
$fIn04 = 5.84$	
$fIn05 = 9.56$	
$fIn06 = 16.5$	
$fIn07 = 32,781$	
$fIn08 = 25.4$	
$fIn09 = 44.5$	
$fIn10 = 66.1$	
$fIn11 = 45.5$	
$fIn12 = 83.3$	
$fIn13 = 54.56$	
$fIn14 = 33.8$	
$fIn15 = 98.5$	
$fIn16 = 71.3$	

If the entered value at $nSel$ is greater than the number of inputs, the "highest" input is output at fQ :

Inputs	Output
bEn = TRUE	fQ = 2.3
nSel = 25	
fIn01 = 15.9	
fIn02 = 32.5	
fIn03 = 17.4	
fIn04 = 5.84	
fIn05 = 9.56	
fIn06 = 16.5	
fIn07 = 32,781	
fIn08 = 25.4	
fIn09 = 44.5	
fIn10 = 66.1	
fIn11 = 45.5	
fIn12 = 83.3	
fIn13 = 54.56	
fIn14 = 33.8	
fIn15 = 98.5	
fIn16 = 71.3	

If *bEn*=FALSE, 0.0 is output at output *fQ* or FALSE for boolean multiplexers.

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  nSel     : UDINT;
  fIn01    : LREAL;
  fIn02    : LREAL;
  fIn03    : LREAL;
  fIn04    : LREAL;
  fIn05    : LREAL;
  fIn06    : LREAL;
  fIn07    : LREAL;
  fIn08    : LREAL;
  fIn09    : LREAL;
  fIn10    : LREAL;
  fIn11    : LREAL;
  fIn12    : LREAL;
  fIn13    : LREAL;
  fIn14    : LREAL;
  fIn15    : LREAL;
  fIn16    : LREAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	Activation of the block function
nSel	UDINT	Number of the input, whose value is to be output at <i>fQ</i> .
f01...f16	LREAL	Input values to select from.

Outputs

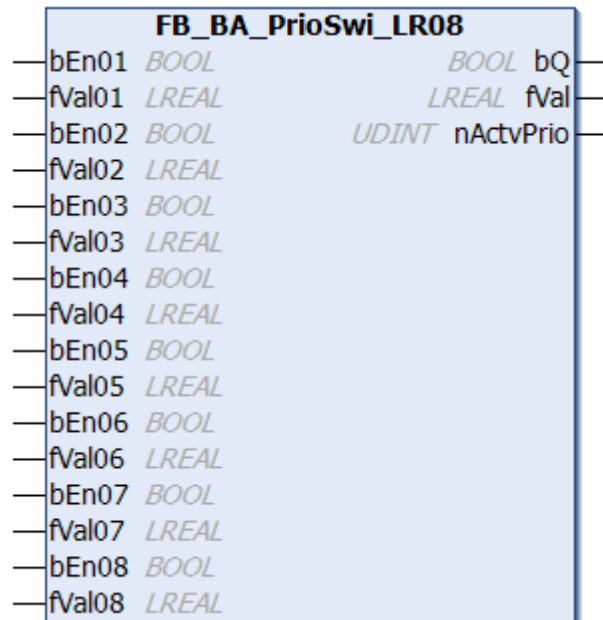
```
VAR_OUTPUT
  fQ      : LREAL;
END_VAR
```

Name	Type	Description
fQ	LREAL	Value of the selected input.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.4 FB_BA_PrioSwi_XX



The priority switches exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different output values (4, 8, 12 and 16 or 24), but they all have the same functionality. The function block FB_BA_PrioSwi_LR08 is described as an example.

Priority switches are available for selecting different values. At output *fVal* the value with the highest priority is applied whose input *bEnxx* is TRUE.

Example:

Inputs			Outputs		
bEn01	FALSE		bQ	TRUE	
fVal01		32.5	fVal		5.84
bEn02	FALSE		nActvPrio		3
fVal02		17.4			
bEn03	TRUE				
fVal03		5.84			
bEn04	TRUE				
fVal04		9.56			
bEn05	FALSE				
fVal05		16.5			
bEn06	TRUE				
fVal06		32.781			
bEn07	FALSE				
fVal07		25.4			
bEn08	TRUE				
fVal08		44.5			

If none of the priorities is enabled, the output *bQ* switches to FALSE. 0 is output at the outputs *fVal* and *nActvPrio*. For a boolean priority switch, FALSE is then output at *bVal*.

Inputs			Outputs		
bEn01	FALSE		bQ	FALSE	
fVal01		32.5	fVal		0.0
bEn02	FALSE		nActvPrio		0
fVal02		17.4			
bEn03	FALSE				
fVal03		5.84			
bEn04	FALSE				
fVal04		9.56			
bEn05	FALSE				
fVal05		16.5			
bEn06	FALSE				
fVal06		32.781			
bEn07	FALSE				
fVal07		25.4			
bEn08	FALSE				
fVal08		44.5			

If no active priority is present, then the value of the global constant `nNoActivePrio` [► 115] is output at the output `nActvPrio`.

Inputs

```
VAR_INPUT
  bEn01   : BOOL;
  fVal01  : LREAL;
  bEn02   : BOOL;
  fVal02  : LREAL;
  bEn03   : BOOL;
  fVal03  : LREAL;
  bEn04   : BOOL;
  fVal04  : LREAL;
  bEn05   : BOOL;
  fVal05  : LREAL;
  bEn06   : BOOL;
  fVal06  : LREAL;
  bEn07   : BOOL;
  fVal07  : LREAL;
  bEn08   : BOOL;
  fVal08  : LREAL;
END_VAR
```

Name	Type	Description
bEn01...bEn08	BOOL	Enabling the priority value
fVal01...fVal08	LREAL	Priority value

Outputs

```
VAR_OUTPUT
  bQ      : BOOL;
  fVal    : LREAL;
  nActvPrio : UDINT;
END_VAR
```

Name	Type	Description
bQ	BOOL	Output to indicate whether a priority is enabled.
fVal	LREAL	Output of the value of the current (highest) priority that is enabled.
nActvPrio	UDINT	Current (highest) priority that is enabled.

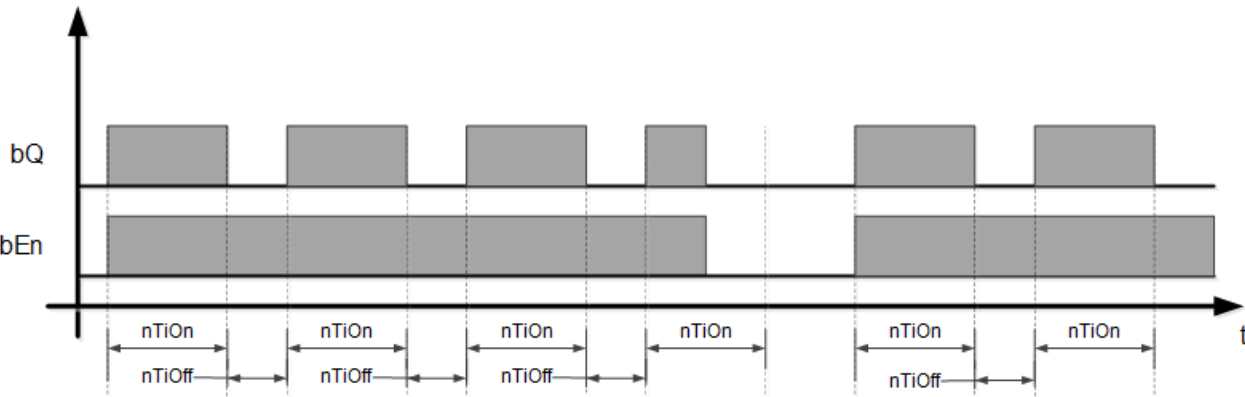
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.5 FB_BA_Blink



The function block FB_BA_Blink is an oscillator with adjustable pulse and pause time, *nTiOn* and *nTiOff* [ms]. It is enabled with a TRUE signal at *bEn* and starts with the pulse phase.



nTiNextSwi is a countdown [s] to the next change of *bQ*.

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  nTiOn    : UDINT;
  nTiOff   : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	Function block enable
nTiOn	UDINT	Pulse time [ms]
nTiOff	UDINT	Pause time [ms]

Outputs

```
VAR_OUTPUT
  bQ      : BOOL;
  nTiNextSwi : UDINT;
END_VAR
```

Name	Type	Description
bQ	BOOL	Oscillator output
nTiNextSwi	UDINT	Countdown to next change of <i>bQ</i> [s]

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.6 FB_BA_FIFO04



The function block FB_BA_FIFO04 enables sequential control of up to four units, with automatic switching of the switch-on sequence based on operating hours.

The function block is available in two versions: for a sequence of four or eight [► 373] units.

Units with fewer operating hours take precedence in the sequence over units with more operating hours.

A rising edge at *bChg* forces a sequence change. The units with the fewest operating hours are set to the top of the FIFO and thus given priority for switching on.

In the sequence only units are entered, which are enabled at inputs *bEn01*..*bEn04*. *nNum* indicates the number of requested units.

The operating hours of the units are entered at inputs *nActvTi01* to *nActvTi04*. If all these inputs are set to a constant value of zero, the sequence change is controlled cyclically, depending on *bChg*.

The first unit is removed from the FIFO, the other units are advanced, and the first unit is appended at the end of the FIFO again. As a result is an alternating sequence of units.

Inputs

```

VAR_INPUT
    bEn      : BOOL;
    nNum     : UDINT;
    bChg     : BOOL;
    bEn01    : BOOL;
    bEn02    : BOOL;
    bEn03    : BOOL;
    bEn04    : BOOL;
    nActvTi01 : UDINT;
    nActvTi02 : UDINT;
    nActvTi03 : UDINT;
    nActvTi04 : UDINT;
END_VAR
  
```

Name	Type	Description
bEn	BOOL	Function block enable
nNum	UDINT	Number of aggregates
bChg	BOOL	Force sequence change
bEn01...bEn04	BOOL	Enable aggregate 1...enable aggregate 4.
nActvTi01...nActvTi04	UDINT	Operating hours aggregate 1...operating hours aggregate 4.

Outputs

```

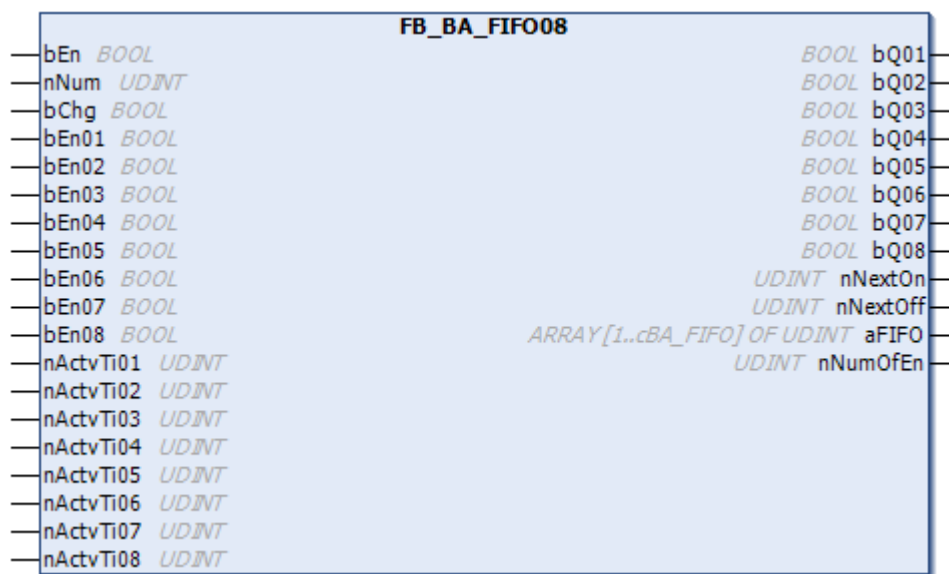
VAR_OUTPUT
    bQ01      : BOOL;
    bQ02      : BOOL;
    bQ03      : BOOL;
    bQ04      : BOOL;
    nNextOn   : UDINT;
    nNextOff  : UDINT;
    aFIFO     : ARRAY [1..4] OF UDINT;
    nNumOfEn  : UDINT;
END_VAR
  
```

Name	Type	Description
bQ01...bQ04	BOOL	Switches aggregate 1...4.
nNextOn	UDINT	Number of the aggregate that is switched on next.
nNextOff	UDINT	Number of the aggregate which will be switched off next.
aFIFO	UDINT	FIFO buffer as a field.
nNumOfEn	UDINT	Number of devices, depending on the individual enable states.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.7 FB_BA_FIFO08



The function block FB_BA_FIFO08 enables a sequence control of up to eight aggregates with automatic change of the switch-on sequence according to operating hours.

The function block is available in two versions: for a sequence of four [► 372] and of eight aggregates.

Aggregates with fewer operating hours take precedence in the sequence over units with more operating hours.

A rising edge at *bChg* forces a sequence change. The aggregates with the lowest operating hours are set to the front in the FIFO and thus switched on with priority.

In the following, only aggregates are entered which are enabled at the inputs *bEn01...bEn08*. *nNum* specifies the number of requested aggregates.

The operating hours of the aggregates are entered at the inputs *nActvTi01* to *nActvTi08*. If these inputs are all constantly set to zero, the sequence change is only cyclically controlled depending on *bChg*.

In this case, the first aggregate always falls out of the FIFO, the others are moved forwards, and the first aggregate is attached again at the end of the FIFO. As a result is an alternating sequence of aggregates.

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  nNum     : UDINT;
  bChg     : BOOL;
  bEn01    : BOOL;
  bEn02    : BOOL;
  bEn03    : BOOL;
  bEn04    : BOOL;
  bEn05    : BOOL;
  bEn06    : BOOL;
```

```

bEn07      : BOOL;
bEn08      : BOOL;
nActvTi01  : UDINT;
nActvTi02  : UDINT;
nActvTi03  : UDINT;
nActvTi04  : UDINT;
nActvTi05  : UDINT;
nActvTi06  : UDINT;
nActvTi07  : UDINT;
nActvTi08  : UDINT;

```

END_VAR

Name	Type	Description
bEn	BOOL	Function block enable
nNum	UDINT	Number of aggregates
bChg	BOOL	Force sequence change
bEn01...bEn08	BOOL	Enable aggregate 1...enable aggregate 8.
nActvTi01...nActvTi08	UDINT	Operating hours aggregate 1...operating hours aggregate 8.

Outputs

```

VAR_OUTPUT
bQ01      : BOOL;
bQ02      : BOOL;
bQ03      : BOOL;
bQ04      : BOOL;
bQ05      : BOOL;
bQ06      : BOOL;
bQ07      : BOOL;
bQ08      : BOOL;
nNextOn   : UDINT;
nNextOff  : UDINT;
aFIFO     : ARRAY [1..8] OF UDINT;
nNumOfEn  : UDINT;

```

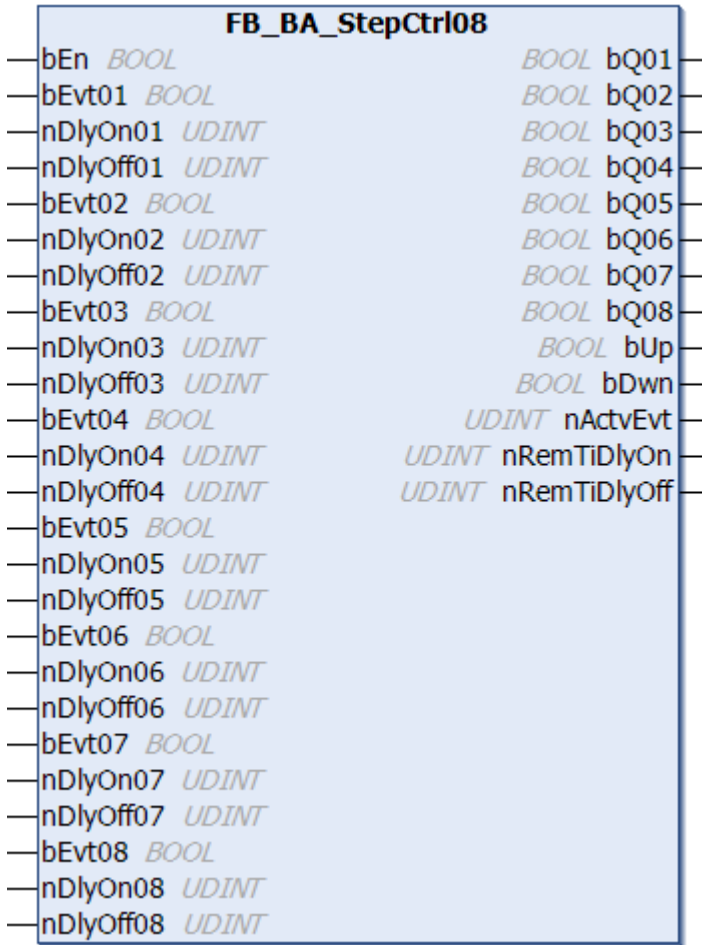
END_VAR

Name	Type	Description
bQ01...bQ08	BOOL	Switches aggregate 1...8.
nNextOn	UDINT	Number of the aggregate that is switched on next.
nNextOff	UDINT	Number of the aggregate which will be switched off next.
aFIFO	UDINT	FIFO buffer as a field.
nNumOfEn	UDINT	Number of devices, depending on the individual enable states.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.8 FB_BA_StepCtrl08

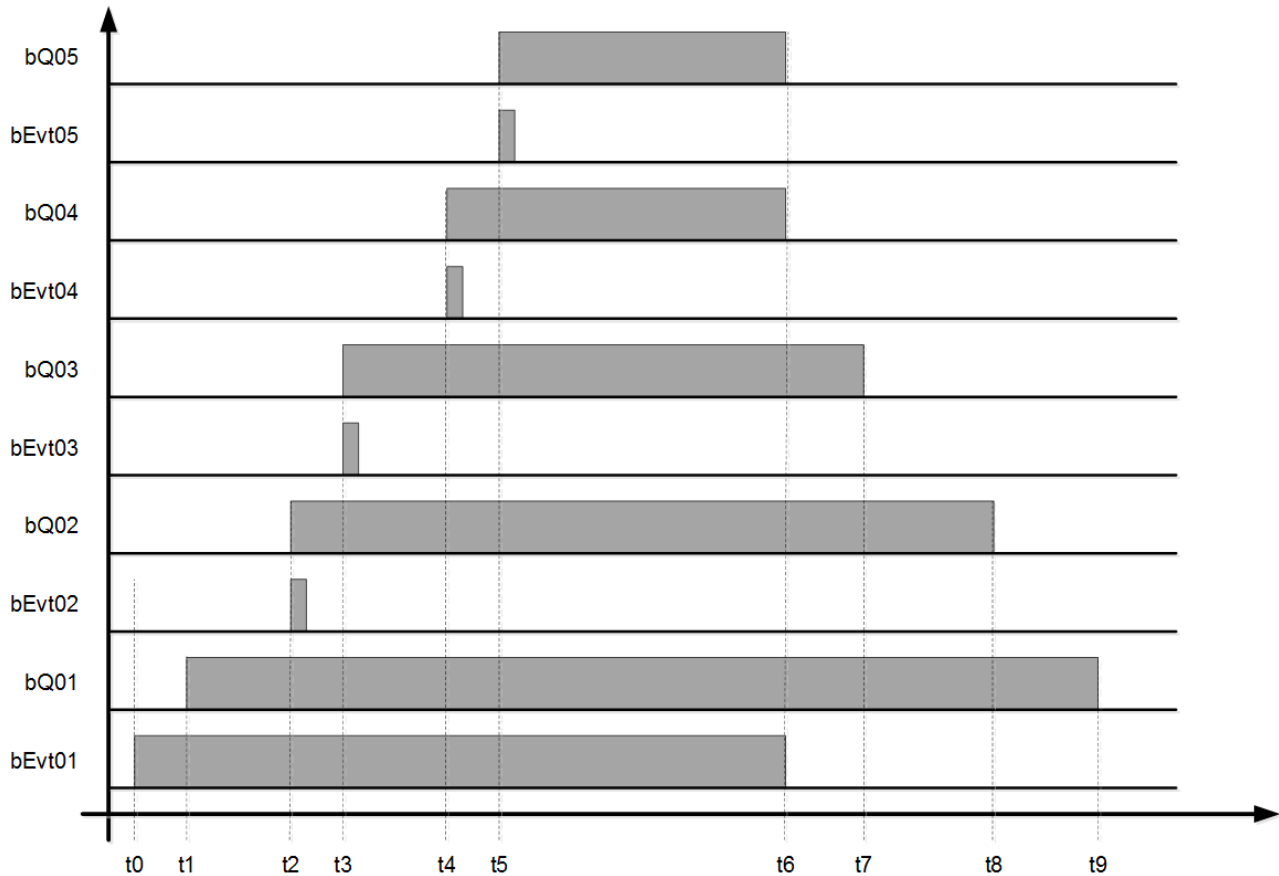


The function block FB_BA_StepCtrl08 is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs from *bQ01* to *bQ08* are set to FALSE. The control sequence starts at input *bEvt01*. Once the timer *nDlyOn01* (see Parameters) has elapsed, the corresponding output *bQ01* is set. Further steps are activated after a rising edge at the inputs *bEvt02* to *bEvt08*, in each case delayed via the timers *nDlyOn02* to *nDlyOn08*. If *bEvt01* becomes FALSE once the control chain is up and running, the control sequence switches back in reverse order. The switching off of the outputs is delayed by the timers *nDlyOff01* to *nDlyOff08*, see parameters.

The outputs *bUp* and *bDwn* indicate whether the control chain is in ascending or descending state. The variable *nActvEvt* indicates the current step of the control chain. "0" means the step sequence is not active.

The output *nRemTiDlyOn* indicates the time remaining to the next step during up-switching of the control chain. The output *nRemTiDlyOff* indicates the time remaining to the next lower step during down-switching of the control chain.

Sample



- t0 step sequence switch-on
- t1 switch on step 1 $nDlyOn01 = t1 - t0$
- t2 event enable step 2, switch on step 2, $nDlyOn02 = 0$
- t3 event enable step 3, switch on step 3, $nDlyOn03 = 0$
- t4 event enable step 4, switch on step 4, $nDlyOn04 = 0$
- t5 event enable step 5, switch on step 5, $nDlyOn05 = 0$
- t6 disable the step sequence, disable step 5, disable step 4; $nDlyOff05 = 0$, $nDlyOff04 = 0$
- t7 switch off step 3, $nDlyOff03 = t7 - t6$
- t8 switch off step 2, $nDlyOff02 = t8 - t7$
- t9 switch off step 1, $nDlyOff01 = t9 - t8$

Inputs

```

VAR_INPUT
  bEn          : BOOL;
  bEvt01       : BOOL;
  nDlyOn01     : UDINT;
  nDlyOff01    : UDINT;
  bEvt02       : BOOL;
  nDlyOn02     : UDINT;
  nDlyOff02    : UDINT;
  bEvt03       : BOOL;
  nDlyOn03     : UDINT;
  nDlyOff03    : UDINT;
  bEvt04       : BOOL;
  nDlyOn04     : UDINT;
  nDlyOff04    : UDINT;
  bEvt05       : BOOL;
  nDlyOn05     : UDINT;
  nDlyOff05    : UDINT;
  bEvt06       : BOOL;
  nDlyOn06     : UDINT;
  nDlyOff06    : UDINT;

```

```

bEvt07      : BOOL;
nDlyOn07    : UDINT;
nDlyOff07   : UDINT;
bEvt08      : BOOL;
nDlyOn08    : UDINT;
nDlyOff08   : UDINT;
END_VAR

```

Name	Type	Description
bEn	BOOL	Function block enable
bEvt01...08	BOOL	Switch-on command for steps 1 to 8.
nDlyOn01...08	UDINT	Start-up delay for output <i>bQ01...08</i> [s]
nDlyOff01...08	UDINT	Switch-off delay for output <i>bQ01...08</i> [s]

Outputs

```

VAR_OUTPUT
bQ01      : BOOL;
bQ02      : BOOL;
bQ03      : BOOL;
bQ04      : BOOL;
bQ05      : BOOL;
bQ06      : BOOL;
bQ07      : BOOL;
bQ08      : BOOL;
bUp       : BOOL;
bDwn      : BOOL;
nActvEvt  : UDINT;
nRemTiDlyOn : UDINT;
nRemTiDlyOff : UDINT;
END_VAR

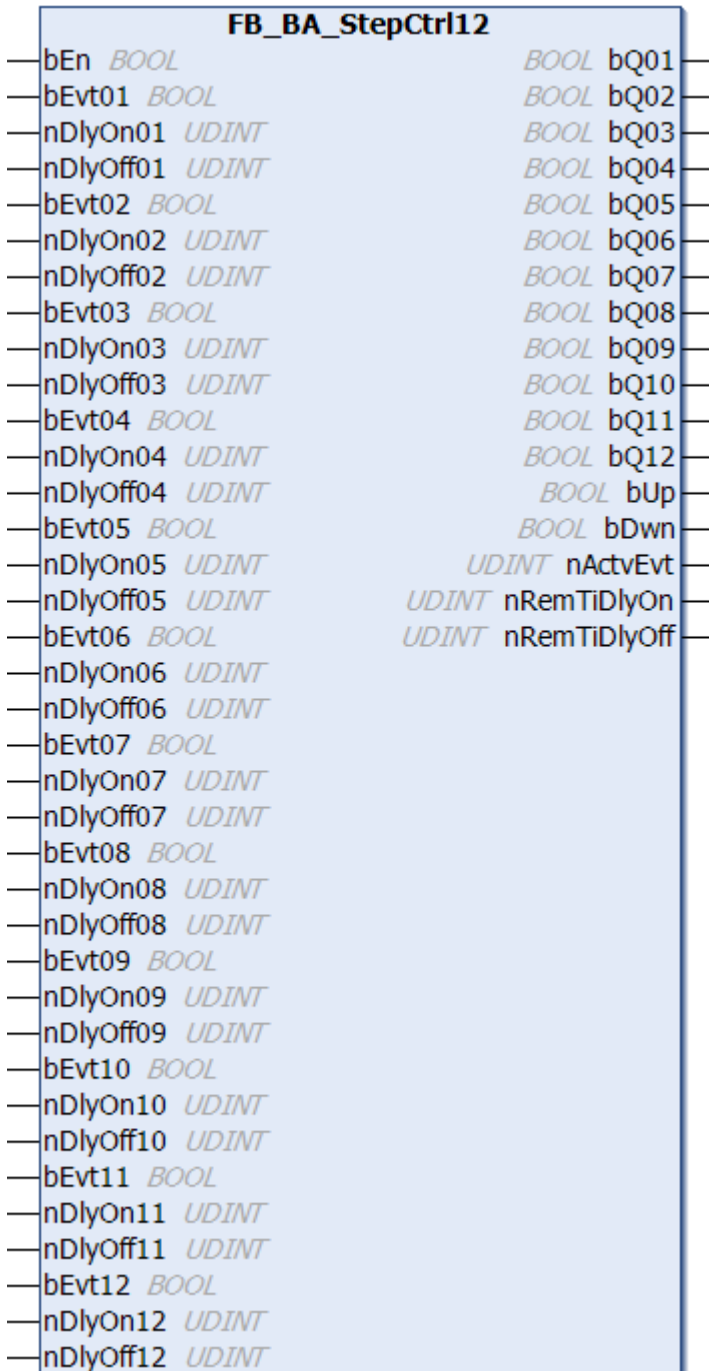
```

Name	Type	Description
bQ01...bQ08	BOOL	Step 1 to 8 On
bUp	BOOL	Control chain is in ascending state.
bDwn	BOOL	Control chain is in descending state.
nActvEvt	UDINT	Active step, display 0...8, "0" means not active step sequence.
nRemTiDlyOn	UDINT	Time remaining to up-switching to the next step [s].
nRemTiDlyOff	UDINT	Time remaining to down-switching to the previous step [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.9 FB_BA_StepCtrl12

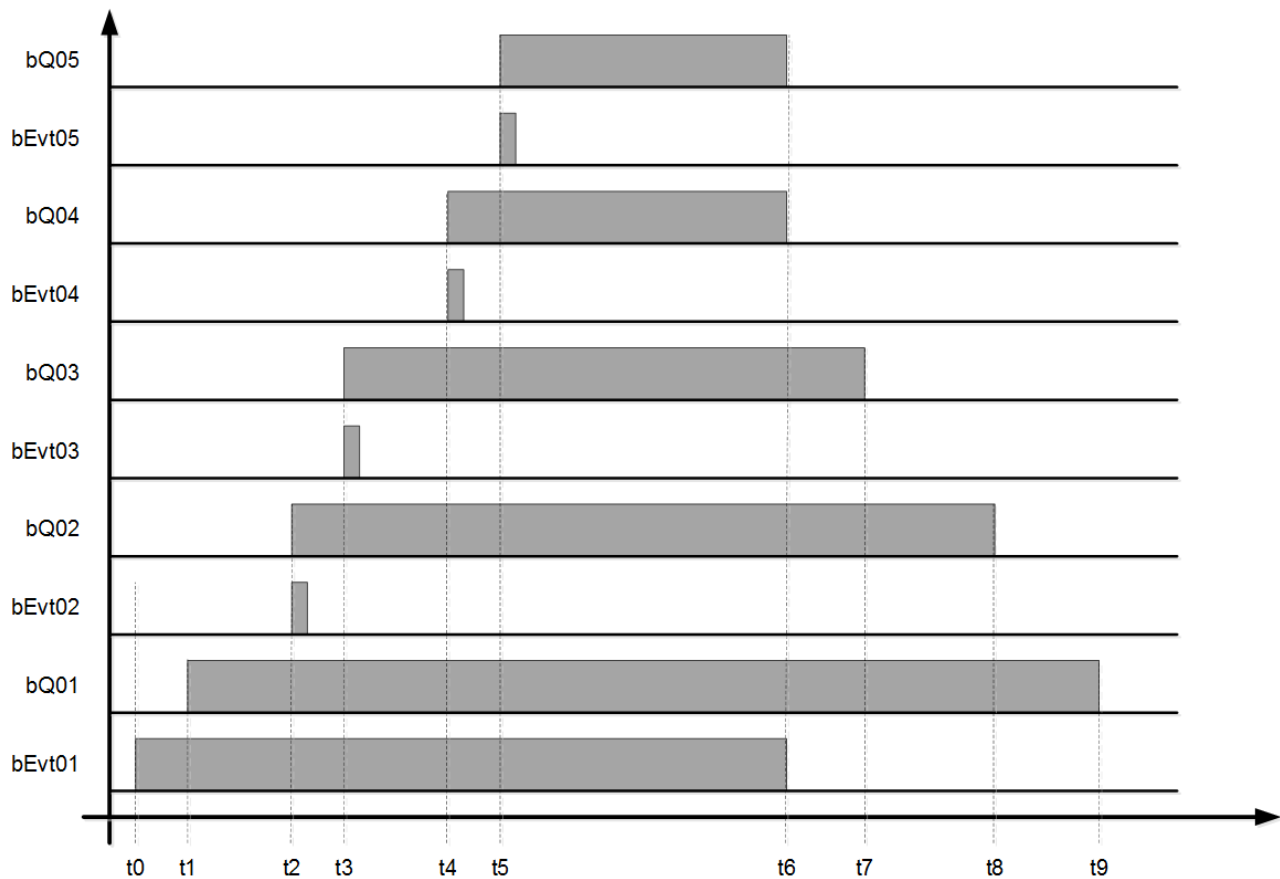


The function block FB_BA_StepCtrl12 is used for output sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs from *bQ01* to *bQ12* are set to FALSE. The control sequence starts at input *bEvt01*. Once the timer *nDlyOn01* (see Parameters) has elapsed, the corresponding output *bQ01* is set. Further steps are activated after a rising edge at the inputs *bEvt02* to *bEvt12*, in each case delayed via the timers *nDlyOn02* to *nDlyOn12*. If *bEvt01* becomes FALSE once the control chain is up and running, the control sequence switches back in reverse order. The switching off of the outputs is delayed by the timers *nDlyOff01* to *nDlyOff12*, see parameters.

The outputs *bUp* and *bDwn* indicate whether the control chain is in ascending or descending state. The variable *nActvEvt* indicates the current step of the control chain. "0" means the step sequence is not active.

The output *nRemTiDlyOn* indicates the time remaining to the next step during up-switching of the control chain. The output *nRemTiDlyOff* indicates the time remaining to the next lower step during down-switching of the control chain.

Sample



- t0 step sequence switch-on
- t1 switch on step 1 $nDlyOn01 = t1 - t0$
- t2 event enable step 2, switch on step 2, $nDlyOn02 = 0$
- t3 event enable step 3, switch on step 3, $nDlyOn03 = 0$
- t4 event enable step 4, switch on step 4, $nDlyOn04 = 0$
- t5 event enable step 5, switch on step 5, $nDlyOn05 = 0$
- t6 disable the step sequence, disable step 5, disable step 4; $nDlyOff05 = 0$, $nDlyOff04 = 0$
- t7 switch off step 3, $nDlyOff03 = t7 - t6$
- t8 switch off step 2, $nDlyOff02 = t8 - t7$
- t9 switch off step 1, $nDlyOff01 = t9 - t8$

Inputs

```

VAR_INPUT
  bEn          : BOOL;
  bEvt01       : BOOL;
  nDlyOn01     : UDINT;
  nDlyOff01    : UDINT;
  bEvt02       : BOOL;
  nDlyOn02     : UDINT;
  nDlyOff02    : UDINT;
  bEvt03       : BOOL;
  nDlyOn03     : UDINT;
  nDlyOff03    : UDINT;
  bEvt04       : BOOL;
  nDlyOn04     : UDINT;
  nDlyOff04    : UDINT;
  bEvt05       : BOOL;
  nDlyOn05     : UDINT;
  nDlyOff05    : UDINT;
  bEvt06       : BOOL;
  nDlyOn06     : UDINT;
  nDlyOff06    : UDINT;

```



```

bEvt07      : BOOL;
nDlyOn07    : UDINT;
nDlyOff07   : UDINT;
bEvt08      : BOOL;
nDlyOn08    : UDINT;
nDlyOff08   : UDINT;
bEvt09      : BOOL;
nDlyOn09    : UDINT;
nDlyOff09   : UDINT;
bEvt10      : BOOL;
nDlyOn10    : UDINT;
nDlyOff10   : UDINT;
bEvt11      : BOOL;
nDlyOn11    : UDINT;
nDlyOff11   : UDINT;
bEvt12      : BOOL;
nDlyOn12    : UDINT;
nDlyOff12   : UDINT;
END_VAR

```

Name	Type	Description
bEn	BOOL	Function block enable
bEvt01...012	BOOL	Switch-on command for steps 1 to 12.
nDlyOn01...12	UDINT	Start-up delay for output <i>bQ01...12</i> [s]
nDlyOff01...12	UDINT	Switch-off delay for output <i>bQ01...12</i> [s]

Outputs

```

VAR_OUTPUT
bQ01      : BOOL;
bQ02      : BOOL;
bQ03      : BOOL;
bQ04      : BOOL;
bQ05      : BOOL;
bQ06      : BOOL;
bQ07      : BOOL;
bQ08      : BOOL;
bQ09      : BOOL;
bQ10      : BOOL;
bQ11      : BOOL;
bQ12      : BOOL;
bUp       : BOOL;
bDwn      : BOOL;
nActvEvt  : UDINT;
nRemTiDlyOn : UDINT;
nRemTiDlyOff : UDINT;
END_VAR

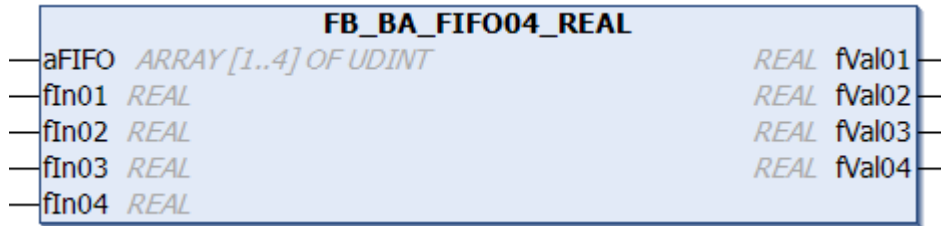
```

Name	Type	Description
bQ01...bQ12	BOOL	Step 1 to 12 On
bUp	BOOL	Control chain is in ascending state.
bDwn	BOOL	Control chain is in descending state.
nActvEvt	UDINT	Active step, display 0...12, "0" means not active step sequence.
nRemTiDlyOn	UDINT	Time remaining to up-switching to the next step [s].
nRemTiDlyOff	UDINT	Time remaining to down-switching to the previous step [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.10 FB_BA_FIFO04_XX



The function block FB_BA_FIFO04_XX is used to evaluate the FiFo memory from FB_BA_FIFO04 [► 372]. The inputs are linked according to the FIFO table to the corresponding outputs of the function block FB_BA_FIFO04_BOOL or FB_BA_FIFO04_REAL.

Example:

In the sample the array contains: 4,3,1,2,0,0,0,0. The following result is output in FB_BA_FIFO04_REAL:

fIn01 on output fVal04

fIn02 on output fVal03

fIn03 on output fVal01

fIn04 on output fVal02

Inputs

```

VAR_INPUT
    aFIFO      : Array [1..4] OF UDINT;
    fIn01...fIn04 : REAL;
END_VAR

```

Name	Type	Description
aFIFO	UDINT	Contains the assignment table with a maximum of eight values. The first value indicates where the first input was copied to, the second value where the second input was copied to and so on. No assignment takes place with "0".
fIn01...fIn04	REAL	Setpoints to be linked.

Outputs

```

VAR_OUTPUT
    fVal01...fVal04 : REAL;
END_VAR

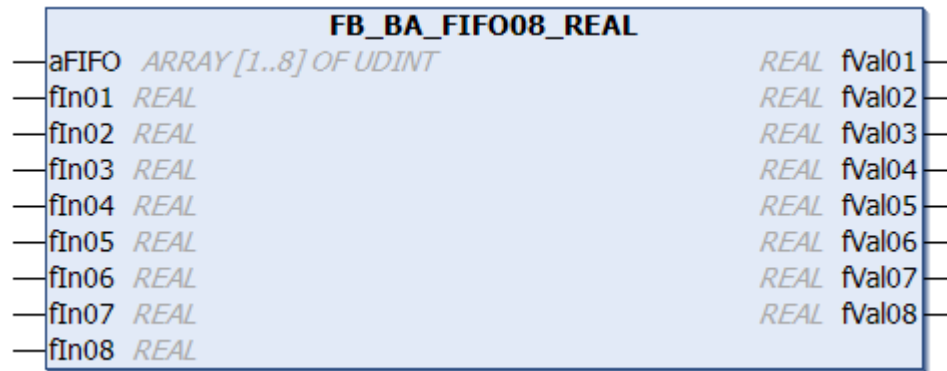
```

Name	Type	Description
fVal01...fVal04	REAL	Actuator setpoint, input value linked according to FIFO table.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.3.11 FB_BA_FIFO08_XX



The function block FB_BA_FIFO08_XX is used to evaluate the FiFo memory from FB_BA_FIFO08 [► 373]. The inputs are linked according to the FIFO table to the corresponding outputs of the function block FB_BA_FIFO08_BOOL or FB_BA_FIFO08_REAL.

Example:

In the sample the array contains: 4,3,1,2,0,0,0,0. The following result is output in FB_BA_FIFO08_REAL:

fIn01 on output fVal04

fIn02 on output fVal03

fIn03 on output fVal01

fIn04 on output fVal02

Inputs

```

VAR_INPUT
  aFIFO      : Array [1..8] OF UDINT;
  fIn01...fIn08 : REAL;
END_VAR
  
```

Name	Type	Description
aFIFO	UDINT	Contains the assignment table with a maximum of eight values. The first value indicates where the first input was copied to, the second value where the second input was copied to and so on. No assignment takes place with "0".
fIn01...fIn08	REAL	Setpoints to be linked.

Outputs

```

VAR_OUTPUT
  fVal01...fVal08 : REAL;
END_VAR
  
```

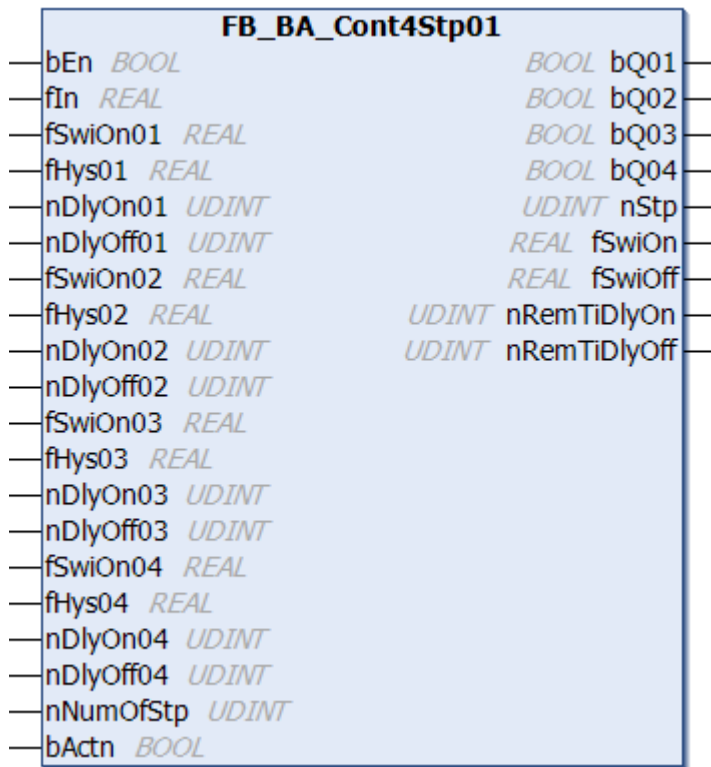
Name	Type	Description
fVal01...fVal08	REAL	Actuator setpoint, input value linked according to FIFO table.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.4 Hysteresis, 2-point control

6.1.2.2.3.1.5.4.1 FB_BA_Cont4Stp01

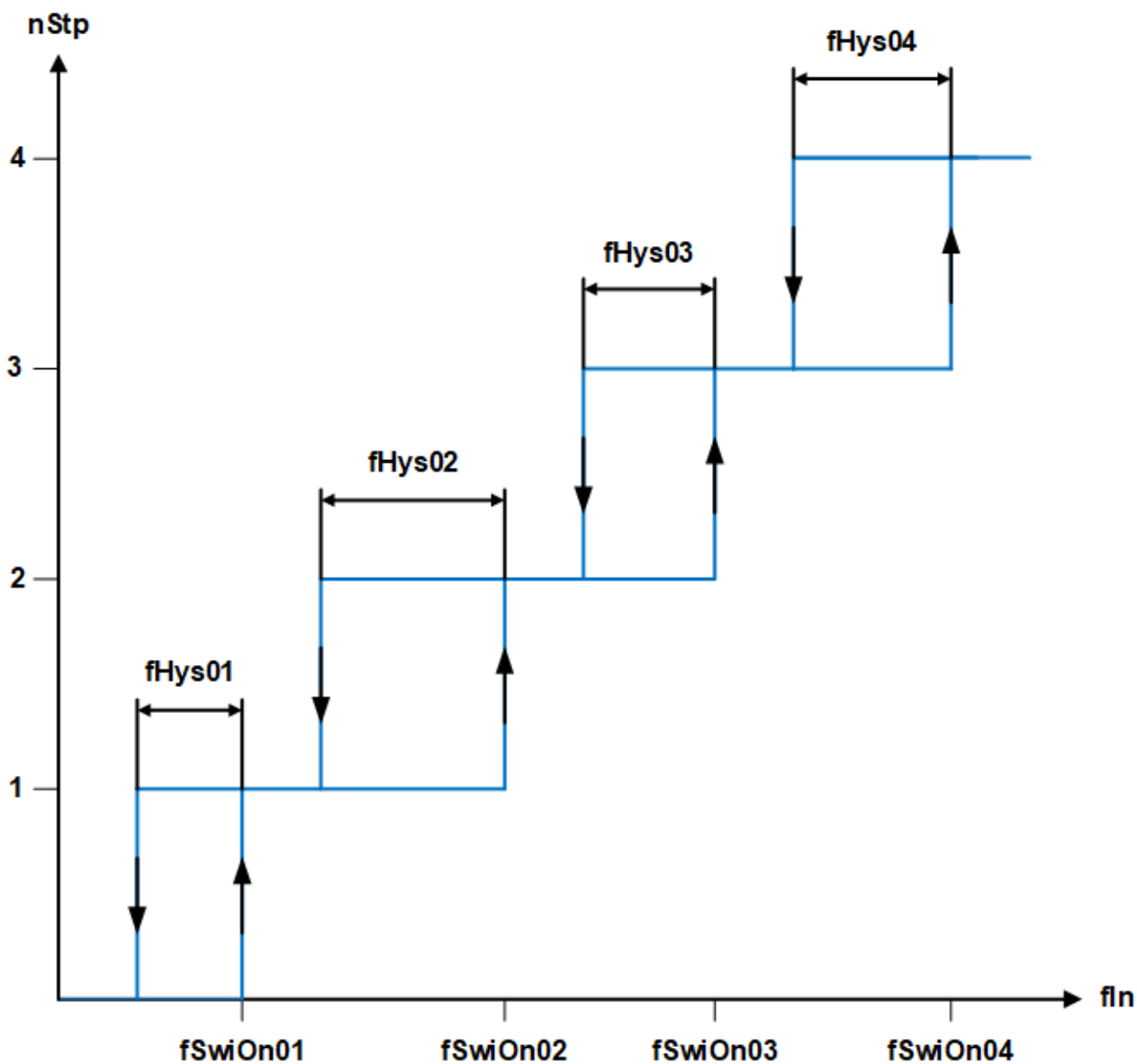


The function block FB_BA_Cont4Stp01 determines the resulting control steps of a multi-stage aggregate depending on the input signal.

Four switch-on thresholds and four hystereses can be parameterized.

Diagram 01

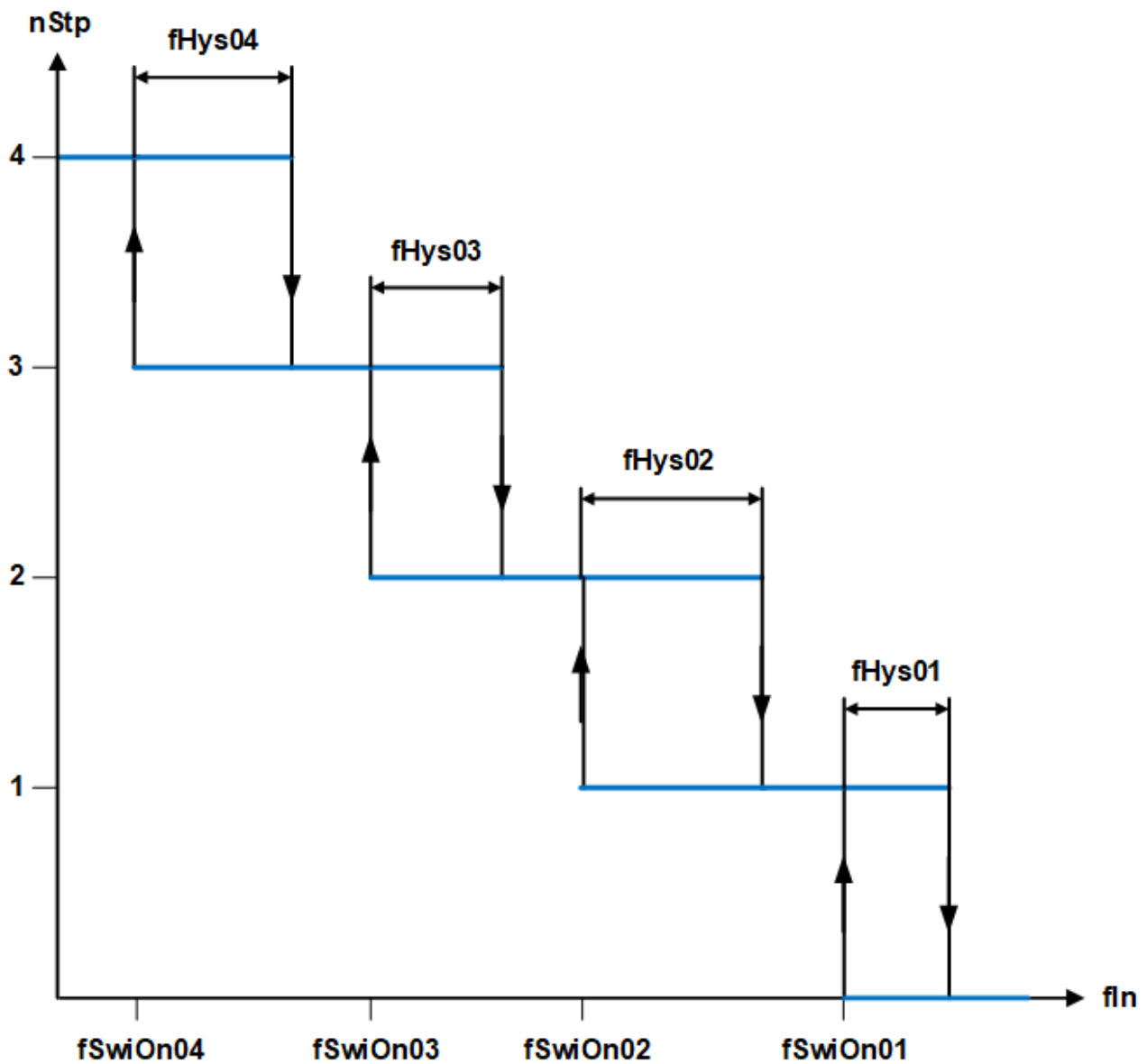
Control direction of parameter *bActn* = FALSE = Reverse = Heating



nStp	nNumOf-Stp	fSwiOn	fSwiOff	nRemTiDlyOn	nRemTiDlyOff	bQ01	bQ02	bQ03	bQ04
0	0	fSwiOn01	fSwiOn01 - fHys01	nDlyOn0 1	0	FALSE	FALSE	FALSE	FALSE
1	≥ 1	fSwiOn02	fSwiOn01 - fHys01	nDlyOn0 2	nDlyOff0 1	TRUE	FALSE	FALSE	FALSE
2	≥ 2	fSwiOn03	fSwiOn02 - fHys02	nDlyOn0 3	nDlyOff0 2	TRUE	TRUE	FALSE	FALSE
3	≥ 3	fSwiOn04	fSwiOn03 - fHys03	nDlyOn0 4	nDlyOff0 3	TRUE	TRUE	TRUE	FALSE
4	≥ 4	fSwiOn04	fSwiOn04 - fHys04	0	nDlyOff0 4	TRUE	TRUE	TRUE	TRUE

Diagram 02

Control direction parameter $bActn = \text{TRUE} = \text{Direct} = \text{Cooling}$



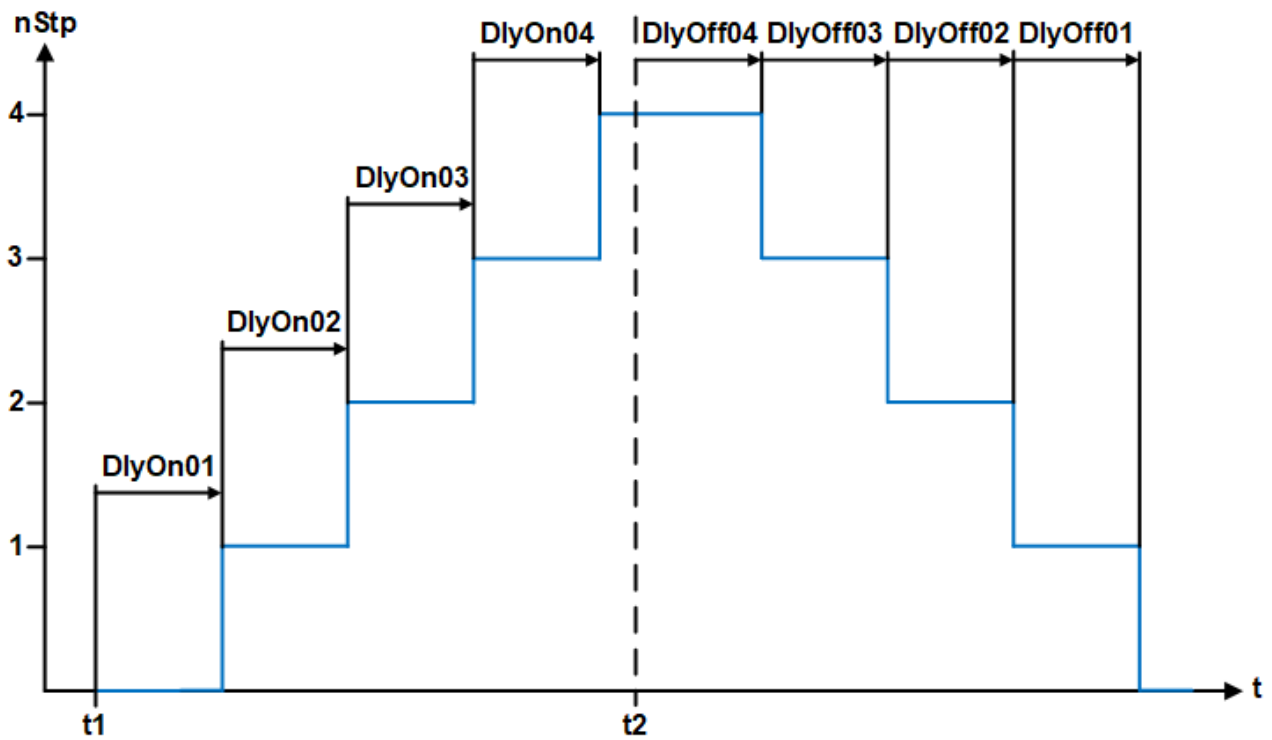
nStp	nNumOf-Stp	fSwiOn	fSwiOff	nRemTiDlyOn	nRemTiDlyOff	bQ01	bQ02	bQ03	bQ04
0	0	fSwiOn01	fSwiOn01 + fHys01	nDlyOn0 1	0	FALSE	FALSE	FALSE	FALSE
1	≥ 1	fSwiOn02	fSwiOn01 + fHys01	nDlyOn0 2	nDlyOff0 1	TRUE	FALSE	FALSE	FALSE
2	≥ 2	fSwiOn03	fSwiOn02 + fHys02	nDlyOn0 3	nDlyOff0 2	TRUE	TRUE	FALSE	FALSE
3	≥ 3	fSwiOn04	fSwiOn03 + fHys03	nDlyOn0 4	nDlyOff0 3	TRUE	TRUE	TRUE	FALSE
4	4	fSwiOn04	fSwiOn04 + fHys04	0	nDlyOff0 4	TRUE	TRUE	TRUE	TRUE

Diagram 03

Timing of the switch-on and switch-off delays

At time $t1$ fln jumps from $fSwiOn01$ to $fSwiOn04$

At time $t2$ fln jumps from $fSwiOn04$ to $fSwiOn01 - fHys01$



Inputs

```

VAR_INPUT
  bEn          : BOOL;
  fIn          : REAL;
  fSwiOn01     : REAL;
  fHys01       : REAL;
  nDlyOn01     : UDINT;
  nDlyOff01    : UDINT;
  fSwiOn02     : REAL;
  fHys02       : REAL;
  nDlyOn02     : UDINT;
  nDlyOff02    : UDINT;
  fSwiOn03     : REAL;
  fHys03       : REAL;
  nDlyOn03     : UDINT;
  nDlyOff03    : UDINT;
  fSwiOn04     : REAL;
  fHys04       : REAL;
  nDlyOn04     : UDINT;
  nDlyOff04    : UDINT;
  nNumOfStp    : UDINT;
  bActn        : BOOL;
END_VAR

```

Name	Type	Description
bEn	BOOL	General enable of the function block. If <i>bEn</i> is FALSE, all outputs are set to 0.
fIn	REAL	Input value, from which the switching state is derived.
fSwiOn01	REAL	Switch-on point step 01
fHys01	REAL	Absolute value hysteresis step 01
nDlyOn01	UDINT	Start-up delay step 01
nDlyOff01	UDINT	Switch-off delay step 01
fSwiOn02	REAL	Switch-on point step 02
fHys02	REAL	Absolute value hysteresis step 02
nDlyOn02	UDINT	Start-up delay step 02
nDlyOff02	UDINT	Switch-off delay step 02
fSwiOn03	REAL	Switch-on point step 03
fHys03	REAL	Absolute value hysteresis step 03
nDlyOn03	UDINT	Start-up delay step 03
nDlyOff03	UDINT	Switch-off delay step 03
fSwiOn04	REAL	Switch-on point step 04
fHys04	REAL	Absolute value hysteresis step 04
nDlyOn04	UDINT	Start-up delay step 04
nDlyOff04	UDINT	Switch-off delay step 04
nNumOfStp	UDINT	Input of the number of steps required. The input is limited from 0 to 4.
bActn	BOOL	Input variable with which the control direction of the step switch is determined. TRUE = Direct = Cooling; FALSE = Reverse = Heating

Outputs

```

VAR_OUTPUT
  bQ01      : BOOL;
  bQ02      : BOOL;
  bQ03      : BOOL;
  bQ04      : BOOL;
  nStp      : UDINT;
  fSwiOn    : REAL;
  fSwiOff   : REAL;
  nRemTiDlyOn : UDINT;
  nRemTiDlyOff : UDINT;
END_VAR

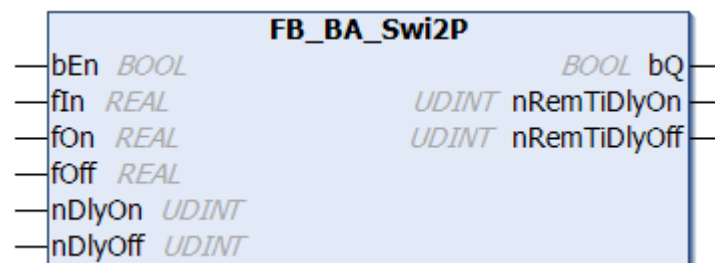
```


Name	Type	Description
bQ01	BOOL	Display of status step 01 TRUE = ON; FALSE = OFF nStp >= 1
bQ02	BOOL	Display of status step 02 TRUE = ON; FALSE = OFF nStp >= 2
bQ03	BOOL	Display of status step 03 TRUE = ON; FALSE = OFF nStp >= 3
bQ04	BOOL	Display of status step 04 TRUE = ON; FALSE = OFF nStp >= 4
nStp	UDINT	Shows the current step of the step switch
fSwiOn	REAL	Shows the next switch-on point
fSwiOff	REAL	Shows the next switch-off point
nRemTiDlyOn	UDINT	If the switch-on point for switching to the next level is met, the progress of the switch-on delay time is displayed here.
nRemTiDlyOff	UDINT	If the switch-off point for switching down to the next level is met, the progress of the switch-off delay time is displayed here.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.4.2 FB_BA_Swi2P

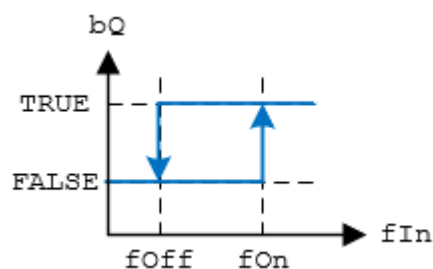


The function block FB_BA_Swi2P is a two-point switch with one switch-on point and one switch-off point.

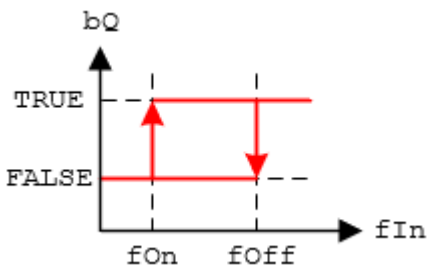
A binary output signal is generated based on a comparison of the input signal *fIn* with the switch-on point *fOn* and the switch-off point *fOff*.

A general function block enable can be implemented at input *bEn*. If *bEn* is FALSE, the output *bQ* is also FALSE. The control direction of the function block depends on the relative position of the switch-on/switch-off points.

If the switch-on point is greater than the switch-off point, the control direction is direct/synchronous (cooling mode).



If the switch-off point is greater than the switch-on point, the control direction is indirect/reversed (heating mode).



🔧 Inputs

```
VAR_INPUT
  bEn      : BOOL;
  fIn      : REAL;
  fOn      : REAL;
  fOff     : REAL;
  nDlyOn   : UDINT;
  nDlyOff  : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	General enable of the function block.
fIn	REAL	Input value
fOn	REAL	Switch-on point
fOff	REAL	Switch-off point
nDlyOn	UDINT	Start-up delay [s]
nDlyOff	UDINT	Switch-off delay [s]

🔧 Outputs

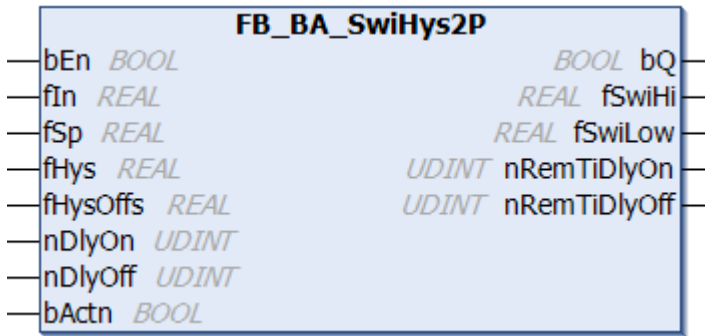
```
VAR_OUTPUT
  bQ      : BOOL;
  nRemTiDlyOn : UDINT;
  nRemTiDlyOff : UDINT;
END_VAR
```

Name	Type	Description
bQ	BOOL	Calculated output value of the characteristic curve.
nRemTiDlyOn	UDINT	Remaining time of the start-up delay [s].
nRemTiDlyOff	UDINT	Remaining time of the switch-off delay [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.4.3 FB_BA_SwiHys2P



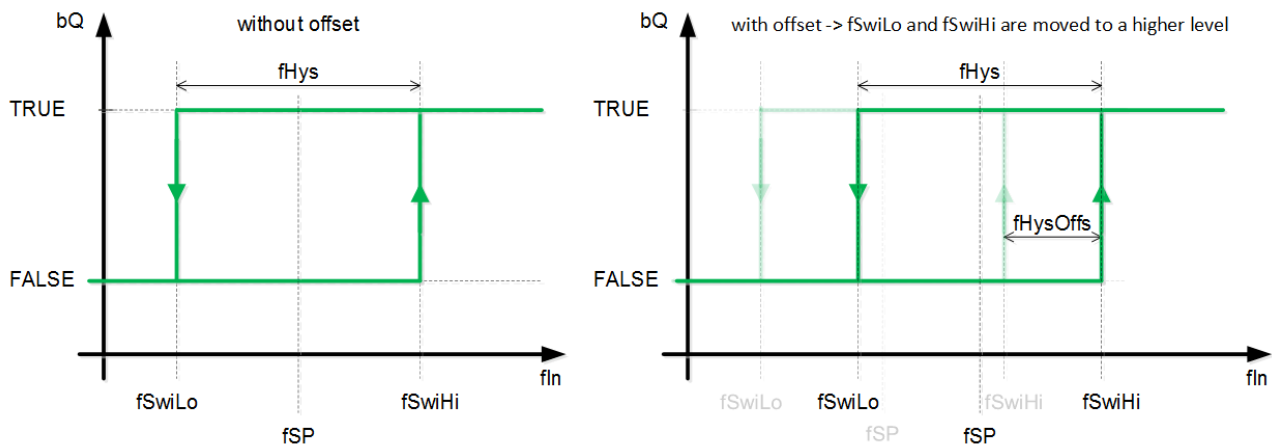
The function block FB_BA_SwiHys2P is a two-point switch with adjustable hysteresis and hysteresis offset.

A general function block enable can be implemented at input *bEn*. If the function block is locked, the output *bQ* is FALSE. The setpoint for the two-point switch is connected at input *fSp*. The control direction of the function block depends on the input variable *bActn*.

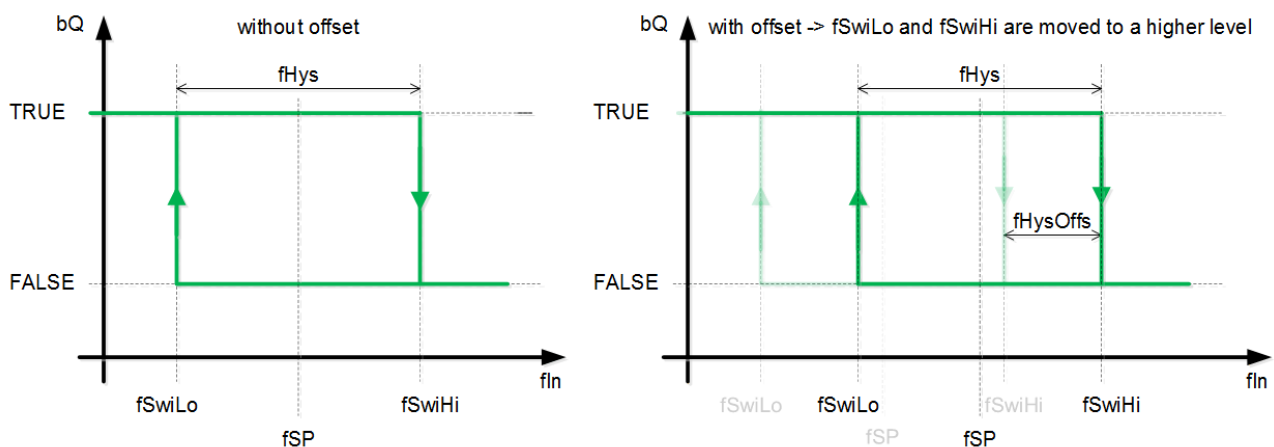
The active switching points result from the setpoint, the hysteresis and the hysteresis offset. They are output at *fSwiHi* and *fSwiLow*.

- The upper switching point is given by $fSp + fHys/2 + fHysOffs$.
- The lower switching point is given by $fSp - fHys/2 + fHysOffs$.

If *bActn* TRUE, the result is direct/synchronous control direction (cooling mode).



If *bActn* is FALSE, the result is indirect/reversed control direction (heating mode).



Inputs

```
VAR_INPUT
  bEn      : BOOL;
  rIn      : REAL;
  rSp      : REAL;
  rHys     : REAL;
  rHysOffs : REAL;
  udiDlyOn_sec : UDINT;
  udiDlyOff_sec : UDINT;
  bActn    : BOOL;
END_VAR
```

Name	Type	Description
bEn	BOOL	General enable of the function block.
fIn	REAL	Input value
fSp	REAL	Setpoint input
fHys	REAL	Hysteresis
fHysOffs	REAL	Hysteresis offset
nDlyOn	UDINT	Start-up delay [s]
nDlyOff	UDINT	Switch-off delay [s]
bActn	BOOL	Control direction

Outputs

```
VAR_OUTPUT
  bQ      : BOOL;
  fSwiHi  : REAL;
  fSwiLow : REAL;
  nRemTiDlyOn : UDINT;
  nRemTiDlyOff : UDINT;
END_VAR
```

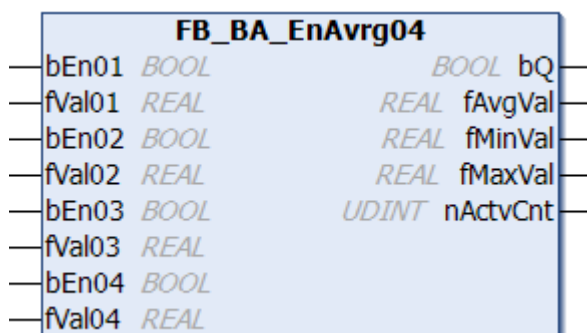
Name	Type	Description
bQ	BOOL	Output
fSwiHi	REAL	Upper switching point
fSwiLow	REAL	Lower switching point
nRemTiDlyOn	UDINT	Remaining time of the start-up delay [s].
nRemTiDlyOff	UDINT	Remaining time of the switch-off delay [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5 Mathematical functions

6.1.2.2.3.1.5.5.1 FB_BA_EnAvrg0X



The function block calculates the arithmetic average from the enabled input values. The function block is available for the variants of 2, 4 and 8 input values. The following documentation refers to the *FB_BA_EnAvrg04*.

Inputs

```
VAR_INPUT
    bEn01...bEn04      : BOOL;
    fVal01...fVal04    : REAL;
END_VAR
```

Name	Type	Description
bEn01...bEn04	BOOL	General enable of an average calculation. If <i>bEn0x</i> = FALSE, the corresponding input value is not included for averaging.
fVal01...fVal04	REAL	The values from which the average value is to be calculated are applied to the variables <i>fVal0</i> to <i>fVal0x</i> .

Outputs

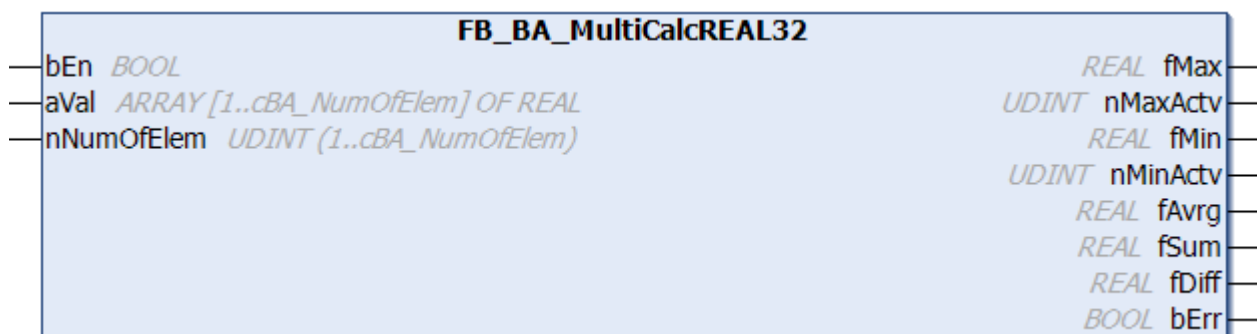
```
VAR_OUTPUT
    fAvgVal      : REAL;
    fMinVal      : REAL;
    fMaxVal      : REAL;
    nActvCnt     : UDINT;
END_VAR
```

Name	Type	Description
fAvgVal	REAL	Calculated arithmetic average.
fMinVal	REAL	Smallest input value.
fMaxVal	REAL	Largest input value.
nActvCnt	UDINT	Number of input values considered for averaging.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.2 FB_BA_MultiCalcREAL32



The multi-calculation function block *FB_BA_MultiCalcREAL32* exists for the variable type REAL.

In enabled state (*bEn* = TRUE), the function block determines the following from the input values *aVal*:

- the maximum value of all inputs *fMax*
- the input at which this maximum value is applied *nMaxActv*
- the minimum value of all inputs *fMin*
- the input at which this minimum value is applied *nMinActv*
- the average of all inputs *fAvg*

- the sum of all inputs *fSum*
- the difference between the maximum and minimum value *fDiff*

If not all inputs are to be calculated, the number can be limited by an entry at *nNumOfElem*: with *nNumOfElem* = 6, for example, the calculations are performed only for the first six entries of *aVal*. An entry greater than 32 is automatically limited to 32, an entry less than 1 is automatically limited to 1.

Sample:

Inputs	Output
bEn = TRUE	fMax = 32
aVal[1] = 32	nMaxActv = 1
aVal[2] = 17	fMin = 5
aVal[3] = 5	nMinActv = 3
aVal[4] = 9	fAvg = 18.5
aVal[5] = 16	fSum = 111
aVal[6] = 32	fDiff = 27
aVal[7] = 25	
aVal[8] = 44	
nNumOfElem = 6	

If *bEn* = FALSE, 0 is output at all outputs.

Inputs

```
VAR_INPUT
  bEn      : BOOL;
  aVal     : ARRAY [1..??] of REAL;
  nNumOfElem : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	Activation of the block function.
aVal	REAL	Field with the values to be calculated.
nNumOfElem	UDINT	Number of input values to be used for the calculation.

Outputs

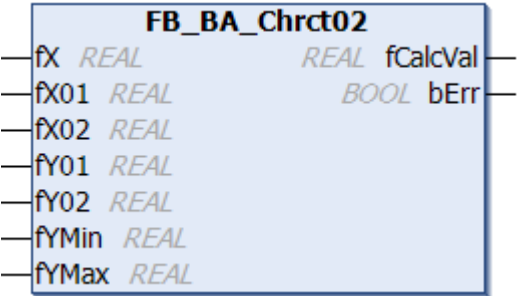
```
VAR_OUTPUT
  fMax      : REAL;
  nMaxActv  : UDINT;
  fMin      : REAL;
  nMinActv  : UDINT;
  fAvg      : REAL;
  fSum      : REAL;
  fDiff     : REAL;
  bErr      : BOOL;
END_VAR
```

Name	Type	Description
fMax	REAL	Maximum value of all inputs.
nMaxActv	UDINT	Input at which the maximum value is present.
fMin	REAL	Minimum value of all inputs.
nMinActv	UDINT	Input at which the minimum value is present.
fAvg	REAL	Average value of all inputs
fSum	REAL	Sum of all inputs
fDiff	REAL	Difference between maximum and minimum value.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.

Requirements

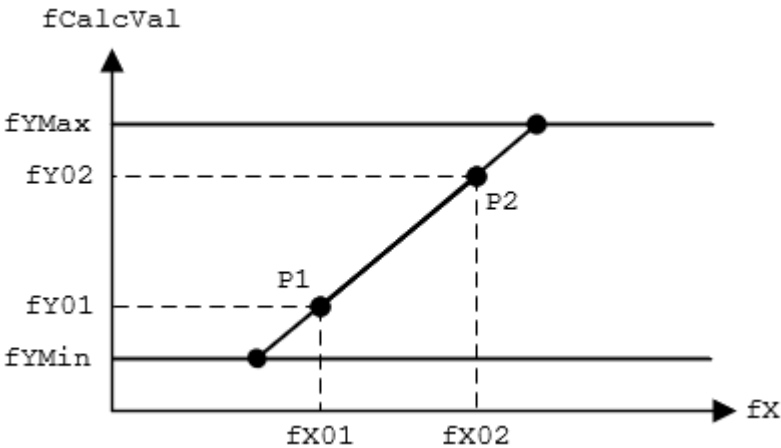
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.3 FB_BA_Chrc02



The function block FB_BA_Chrc02 represents a linear interpolation with 2 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [fX01/fY01] and [fX02/fY02].

The calculated output value *fCalcVal* is limited by *fYMin* or *fYMax*.



Error handling

The input values for *fX0[n+1]* must always be greater than from *fX0[n]*.

In case of an error the variable *bErr* indicates this. The property *ErrorDescription* gives out a detailed description.

The input value for *fYMin* must not be greater than *fYMax*.

🔧 Inputs

```
VAR_INPUT
  fX      : REAL;
  fX01    : REAL;
  fX02    : REAL;
  fY01    : REAL;
  fY02    : REAL;
  fYMin   : REAL;
  fYMax   : REAL;
END_VAR
```

Name	Type	Description
fX	REAL	Input value of the characteristic curve.
fX0N	REAL	X-value for interpolation point PN.
fY0N	REAL	Y-value for interpolation point PN.
fYMin	REAL	Lower limit of <i>fCalcVal</i> . The input for the lower limit is limited to the global parameter <i>BACmn_Global.fMinReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).
fYMax	REAL	Upper limit of <i>fCalcVal</i> . The input for the upper limit is limited to the global parameter <i>BACmn_Global.fMaxReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).

Outputs

```
VAR_OUTPUT
    fCalcVal    : REAL;
    bErr        : BOOL;
END_VAR
```

Name	Type	Description
fCalcVal	REAL	Calculated output value of the characteristic curve.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.

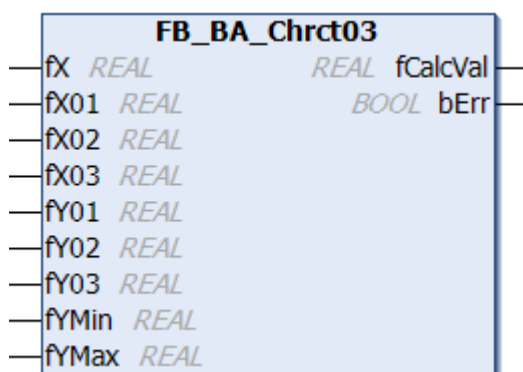
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [► 394].

Requirements

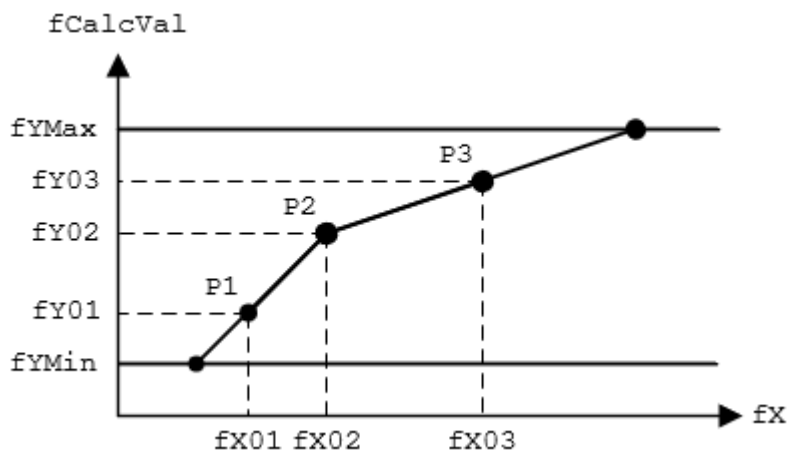
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.4 FB_BA_Chrtc03



The function block FB_BA_Chrtc03 represents a linear interpolation with 3 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [fX01/fY01], [fX02/fY02] and [fX03/fY03].

The calculated output value *fCalcVal* is limited by *fYMin* or *fYMax*.



Error handling

The input values for $fx0[n+1]$ must always be greater than from $fx0[n]$.

In case of an error the variable *bErr* indicates this. The property *ErrorDescription* gives out a detailed description.

The input value for *fYMin* must not be greater than *fYMax*.

Inputs

```
VAR_INPUT
  fX      : REAL;
  fX01    : REAL;
  fX02    : REAL;
  fX03    : REAL;
  fY01    : REAL;
  fY02    : REAL;
  fY03    : REAL;
  fYMin   : REAL;
  fYMax   : REAL;
END_VAR
```

Name	Type	Description
fX	REAL	Input value of the characteristic curve.
fX0N	REAL	X-value for interpolation point PN.
fY0N	REAL	Y-value for interpolation point PN.
fYMin	REAL	Lower limit of <i>fCalcVal</i> . The input for the lower limit is limited to the global parameter <i>BACmn_Global.fMinReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).
fYMax	REAL	Upper limit of <i>fCalcVal</i> . The input for the upper limit is limited to the global parameter <i>BACmn_Global.fMaxReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).

Outputs

```
VAR_OUTPUT
  fCalcVal : REAL;
  bErr     : BOOL;
END_VAR
```

Name	Type	Description
fCalcVal	REAL	Calculated output value of the characteristic curve.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.

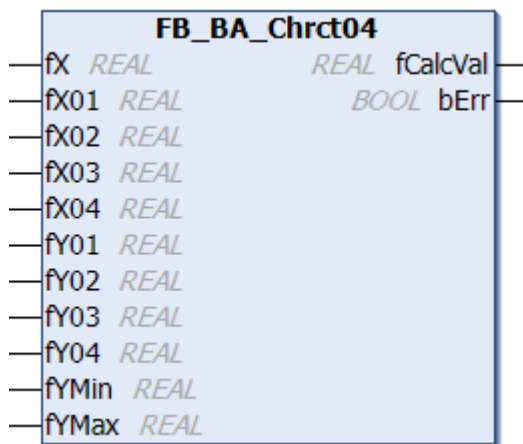
 Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [► 396].

Requirements

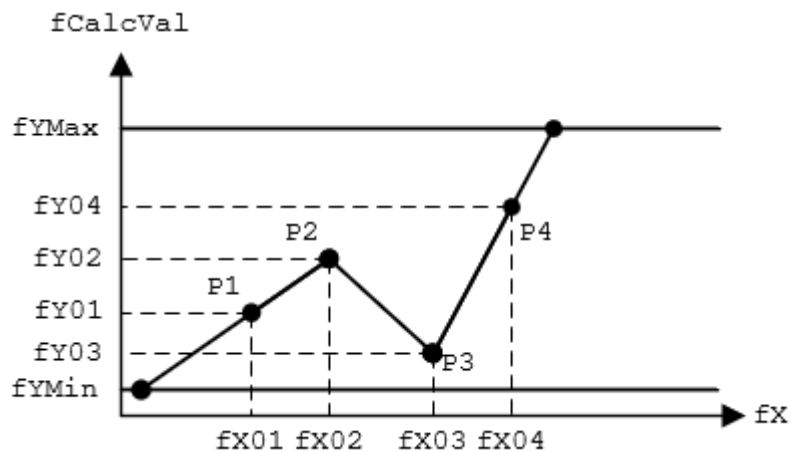
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.5 FB_BA_Chrc04



The function block FB_BA_Chrc04 represents a linear interpolation with 4 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points $[fX01/fY01]$, $[fX02/fY02]$, $[fX03/fY03]$ and $[fX04/fY04]$.

The calculated output value $fCalcVal$ is limited by $fYMin$ or $fYMax$.



Error handling

The input values for $fX0[n+1]$ must always be greater than from $fX0[n]$.

In case of an error the variable $bErr$ indicates this. The property *ErrorDescription* gives out a detailed description.

The input value for $fYMin$ must not be greater than $fYMax$.

Inputs

```
VAR_INPUT
  fX      : REAL;
  fX01    : REAL;
  fX02    : REAL;
  fX03    : REAL;
  fX04    : REAL;
  fY01    : REAL;
  fY02    : REAL;
  fY03    : REAL;
  fY04    : REAL;
  fYMin   : REAL;
  fYMax   : REAL;
END_VAR
```

Name	Type	Description
fX	REAL	Input value of the characteristic curve.
fX0N	REAL	X-value for interpolation point PN.
fY0N	REAL	Y-value for interpolation point PN.
fYMin	REAL	Lower limit of <i>fCalcVal</i> . The input for the lower limit is limited to the global parameter <i>BACmn_Global.fMinReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).
fYMax	REAL	Upper limit of <i>fCalcVal</i> . The input for the upper limit is limited to the global parameter <i>BACmn_Global.fMaxReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).

Outputs

```
VAR_OUTPUT
  fCalcVal : REAL;
  bErr     : BOOL;
END_VAR
```

Name	Type	Description
fCalcVal	REAL	Calculated output value of the characteristic curve.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.

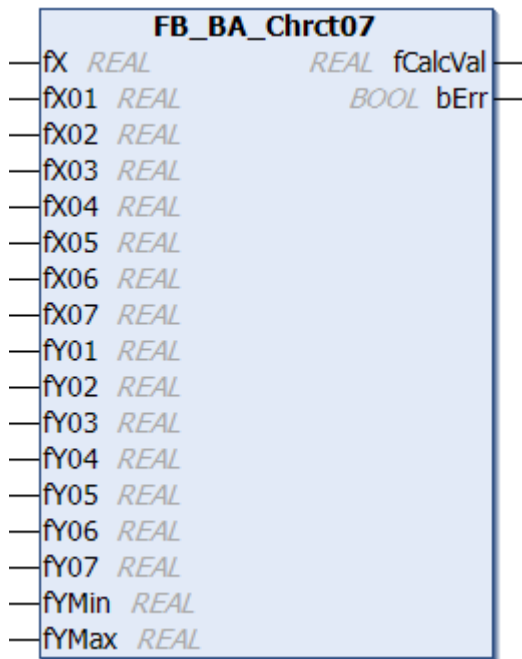
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [► 397].

Requirements

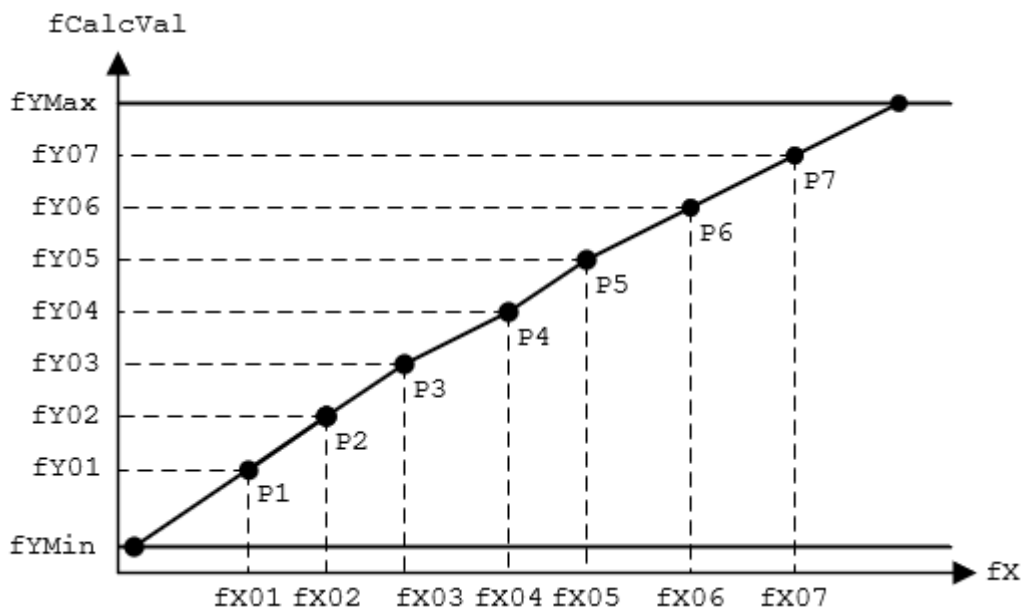
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.6 FB_BA_Chrc07



The function block FB_BA_Chrc07 represents a linear interpolation with 7 interpolation points and can be used to generate a characteristic curve. The characteristic is determined by the interpolation points [fx01/fy01], [fx02/fy02], [fx03/fy03] ... [fx07/fy07].

The calculated output value *fCalcVal* is limited by *fyMin* or *fyMax*.

**Error handling**

The input values for *fx0[n+1]* must always be greater than from *fx0[n]*.

In case of an error the variable *bErr* indicates this. The property *ErrorDescription* gives out a detailed description.

The input value for *fyMin* must not be greater than *fyMax*.

Inputs

```
VAR_INPUT
  fX      : REAL;
  fX01    : REAL;
  fX02    : REAL;
  fX03    : REAL;
  fX04    : REAL;
  fX05    : REAL;
  fX06    : REAL;
  fX07    : REAL;
  fY01    : REAL;
  fY02    : REAL;
  fY03    : REAL;
  fY04    : REAL;
  fY05    : REAL;
  fY06    : REAL;
  fY07    : REAL;
  fYMin   : REAL;
  fYMax   : REAL;
END_VAR
```

Name	Type	Description
fX	REAL	Input value of the characteristic curve.
fX0N	REAL	X-value for interpolation point PN.
fY0N	REAL	Y-value for interpolation point PN.
fYMin	REAL	Lower limit of <i>fCalcVal</i> . The input for the lower limit is limited to the global parameter <i>BACmn_Global.fMinReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).
fYMax	REAL	Upper limit of <i>fCalcVal</i> . The input for the upper limit is limited to the global parameter <i>BACmn_Global.fMaxReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).

Outputs

```
VAR_OUTPUT
  fCalcVal : REAL;
  bErr     : BOOL;
END_VAR
```

Name	Type	Description
fCalcVal	REAL	Calculated output value of the characteristic curve.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.

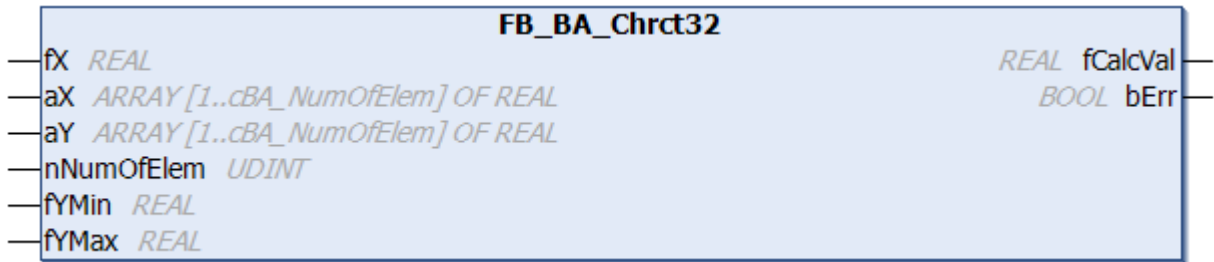
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [► 399] .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

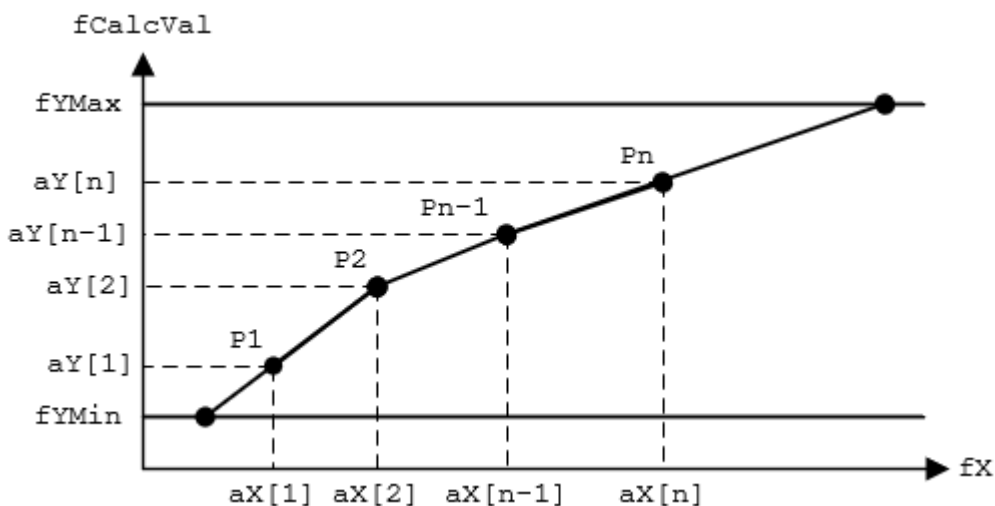
6.1.2.2.3.1.5.5.7 FB_BA_Chrc32



The function block FB_BA_Chrc32 represents a linear interpolation with 32 interpolation points and can be used to generate a characteristic curve. In contrast to the "smaller" interpolation function blocks [FB BA Chrc02](#) [▶ 394], [FB BA Chrc04](#) [▶ 397] and [FB BA Chrc07](#) [▶ 399], and in the interest of clarity, the interpolation points are determined via field variables $[aX[1]/aY[1]]$ to $[aX[n]/aY[n]]$.

The input variable *nNumOfElem* determines the number of interpolation points from the field ranges *aX/aY*.

The calculated output value *fCalcVal* is limited by *fYMin* or *fYMax*.



Error handling

The input values for $aX[n+1]$ must always be greater than from $aX[n]$.

In case of an error the variable *bErr* indicates this. The property *ErrorDescription* gives out a detailed description.

The input value for *fYMin* must not be greater than *fYMax*.

In case of an error the variable *bErr* indicates this. The property *ErrorDescription* gives out a detailed description.

Inputs

```

VAR_INPUT
  fX      : REAL;
  aX      : ARRAY [1..cBA_NumOfElem] OF REAL;
  aY      : ARRAY [1..cBA_NumOfElem] OF REAL;
  nNumOfElem : DINT (2..32);
  fYMin    : REAL;
  fYMax    : REAL;
END_VAR

```

Name	Type	Description
fX	REAL	Input value of the characteristic curve.
aX	REAL	X-value for the interpolation points.
aY	REAL	Y-value for the interpolation points.
nNumOfElem	DINT	Number of interpolation points. Internally limited to the range 2 ... 32.
fYMin	REAL	Lower limit of <i>fCalcVal</i> . The input for the lower limit is limited to the global parameter <i>BACmn_Global.fMinReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).
fYMax	REAL	Upper limit of <i>fCalcVal</i> . The input for the upper limit is limited to the global parameter <i>BACmn_Global.fMaxReal</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).

Outputs

```
VAR_OUTPUT
    fCalcVal    : REAL;
    bErr        : BOOL;
END_VAR
```

Name	Type	Description
fCalcVal	REAL	Calculated output value of the characteristic curve.
bErr	BOOL	This output is switched to TRUE if the parameters entered are erroneous.

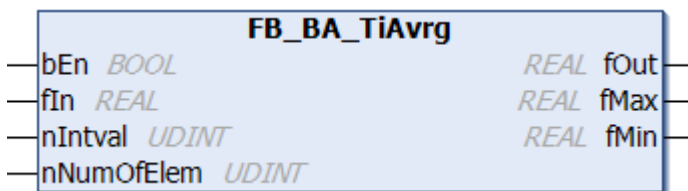
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [► 399] .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.8 FB_BA_TiAavg



The function block FB_BA_TiAavg calculates the average from a sequence of past input values.

The function acquires input values *fIn* at defined time intervals *nIntval* and calculates the average of the acquired values. After each time interval, a new average value *fOut* is available.

The values to be averaged are written to a FIFO memory. This memory is limited to a maximum of 512 entries *nNumOfElem*. After the time interval *nIntval* has elapsed, a new value is written to the memory and then the average value is formed. If the entries of the FIFO memory = *nNumOfElem*, the oldest entry is deleted so that a new value is included for the average calculation.

Interval	nNumOfElem	fln	FIFO memory	fOut	fMax	fMin
1	5	2	2 / 1	2	2	2
2	5	4	(4+2) / 2	3	4	2
3	5	5	(5+4+2) / 3	3.667	5	2
4	5	6	(6+5+4+2) / 4	4.25	6	2
5	5	7	(7+6+5+4+2) / 5	4.8	7	2
6	5	9	(9+7+6+5+4) / 5	6.2	8	2
7	5	14	(14+9+7+6+5) / 5	8.2	14	2
8	5	1	(1+14+9+7+6) / 5	7.4	14	1
9	5	-4	(-4+1+14+9+7) / 5	5.4	14	-4
10	5	-12	((-12)+(-4)+1+14+9) / 5	1.6	14	-12

Inputs

```
VAR_INPUT
    bEn      : BOOL;
    fIn      : REAL;
    nIntVal  : UDINT;
    nNumOfElem : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	Enables the function block. FALSE means that the output variables have the value 0 and the content of the FIFO memory is deleted.
fln	REAL	Field with the values to be calculated.
nIntVal	UDINT	Time interval [s] for writing new values into the FIFO. Internally limited to a value between 1 and 2147483.
nNumOfElem	UDINT	Size of the FIFO buffer. A change resets the previous averaging. Internally limited to a value between 1 and 512.

Outputs

```
Ausgänge
    fOut      : REAL;
    fMax      : REAL;
    fMin      : REAL;
END_VAR
```

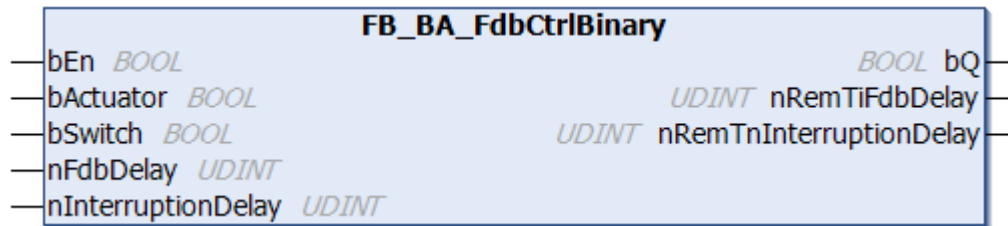
Name	Type	Description
fOut	REAL	Calculated average
fMax	REAL	Largest value in the FIFO buffer
fMin	REAL	Smallest value in the FIFO buffer

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.6 Monitoring functions

6.1.2.2.3.1.5.6.1 FB_BA_FdbCtrlBinary



The function block FB_BA_FdbCtrlBinary is used for feedback monitoring of an actuator by means of digital feedback. Application examples of the function block are, for example, an operation feedback monitoring, a process feedback monitoring or the run monitoring of a drive by means of limit switches.

The function block monitors a limit switch in two steps, for example. Step 1 includes monitoring while the actuator is moving. Step 2 includes monitoring of the opened state of the actuator.

Step 1: If a TRUE is present at the input **bActuator**, then the travel time of the actuator *nFdbDelay*/*nRemTiFdbDelay* expires. If no TRUE is present at input *bSwitch* within this time, this is indicated via output *bQ*. If *bSwitch* becomes TRUE within the time *nFdbDelay*, step 1 is complete and step 2 becomes active.

Step 2: *bActuator* and *bSwitch* are TRUE. If *bActuator* becomes FALSE, step 1 becomes active again. If *bSwitch* becomes FALSE and within the time *nInterruptionDelay*/*nRemTiInterruptionDelay* is not TRUE again, this is indicated via the output *bQ*.

In addition, the idle position of the actuator is monitored. That is, if *bActuator* is not active and the feedback signal *bSwitch* indicates a TRUE, then after the time *nFdbDelay* has elapsed, the output *bQ* is set to TRUE.

bQ signals that the monitoring of the feedback signal has a fault.

Inputs

```

VAR_INPUT
    bEn          : BOOL;
    bActuator    : BOOL;
    bSwitch      : BOOL;
    nFdbDelay    : UDINT;
    nInterruptionDelay : UDINT;
END_VAR

```

Name	Type	Description
bEn	BOOL	General enable of the function block. If <i>bEn</i> is FALSE, the message output <i>bQ</i> is also FALSE.
bActuator	BOOL	The switching actuator output of the aggregate to be monitored is connected to this input.
bSwitch	BOOL	Used to connect the feedback signal, e.g. of a differential pressure switch, flow monitor or limit switch.
nFdbDelay	UDINT	Response delay [s] of the monitoring function when the actuator is started. The input of the time is limited to the global parameter <i>BAComn_Global.udiMaxSecInMilli</i> from the Tc3_BA2_Common library (see BAComn_Global).
nInterruptionDelay	UDINT	Delay time (e.g. of a limit switch) in the open state in [s]. The input of the time is limited to the global parameter <i>BAComn_Global.udiMaxSecInMilli</i> from the Tc3_BA2_Common library, see (BAComn_Global).

🔌 Outputs

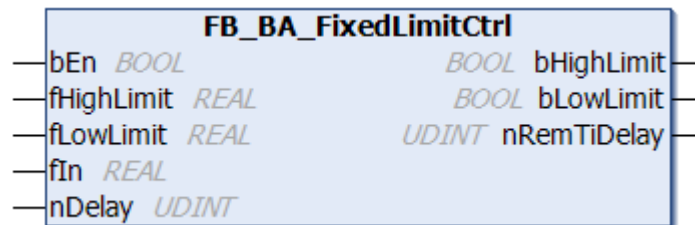
```
VAR_OUTPUT
    bQ          : BOOL;
    nRemTiFdbDelay : UDINT;
    nRemTiInterruptionDelay : UDINT;
END_VAR
```

Name	Type	Description
bQ	BOOL	bQ is used to signal that the monitoring of the feedback signal indicates a fault.
nRemTiFdbDelay	UDINT	Remaining time [s] until output <i>bQ</i> is set. The default comes from <i>nFdbDelay</i> .
nRemTiInterruptionDelay	UDINT	Remaining time [s] until output <i>bQ</i> is set. The default comes from <i>nInterruptionDelay</i> .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.6.2 FB_BA_FixedLimitCtrl



This function block **FB_BA_FixedLimitCtrl** is used to monitor a fixed value.

A tolerance range is defined around the value *fIn* to be monitored.

The tolerance range results from an upper limit *fHighLimit* and a lower limit *fLowLimit*.

If the value *fIn* exceeds the upper limit value of the tolerance range, then the output *bHighLimit* is set. A response delay of the output *bHighLimit* can be parameterized with the time variable *nDelay*.

If the value *fIn* falls below the lower limit value of the tolerance range, then the output *bLowLimit* is set. A response delay of the output *bLowLimit* can be parameterized with the time variable *nDelay*.

If *fLowLimit* > *fHighLimit*, then internally *fLowLimit* is corrected to *fHighLimit*.

🔌 Inputs

```
VAR_INPUT
    bEn          : BOOL;
    fHighLimit   : REAL := 32;
    fLowLimit    : REAL := 16;
    fIn          : REAL;
    nDelay       : UDINT;
END_VAR
```

Name	Type	Description
bEn	BOOL	General enable of the function block. If <i>bEn</i> is FALSE, the message outputs <i>bHighLimit</i> and <i>bLowLimit</i> are also FALSE.
fHighLimit	REAL	Default upper limit.
fLowLimit	REAL	Default lower limit.
fIn	REAL	Input value to be monitored.
nDelay	UDINT	Response delay [s] of the outputs <i>bHighLimit</i> / <i>bLowLimit</i> . The input of the time is limited to the global parameter <i>BACmn_Global.udiMaxSecInMilli</i> from the Tc3_BA2_Common library (see <i>BACmn_Global</i>).

Outputs

```

VAR_OUTPUT
  bHighLimit      : BOOL;
  bLowLimit       : BOOL;
  nRemTiDelay     : UDINT;
END_VAR

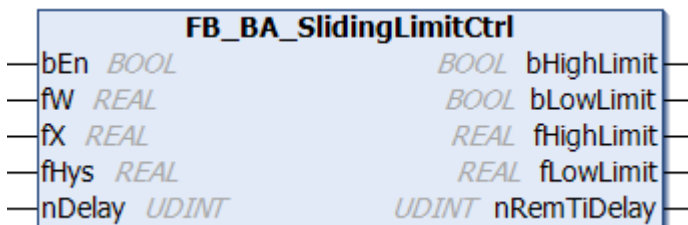
```

Name	Type	Description
bHighLimit	BOOL	Upper limit value reached.
bLowLimit	BOOL	Lower limit value reached.
nRemTiDelay	UDINT	Remaining time after exceeding a limit value until one of the outputs <i>bHighLimit</i> or <i>bLowLimit</i> is set.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

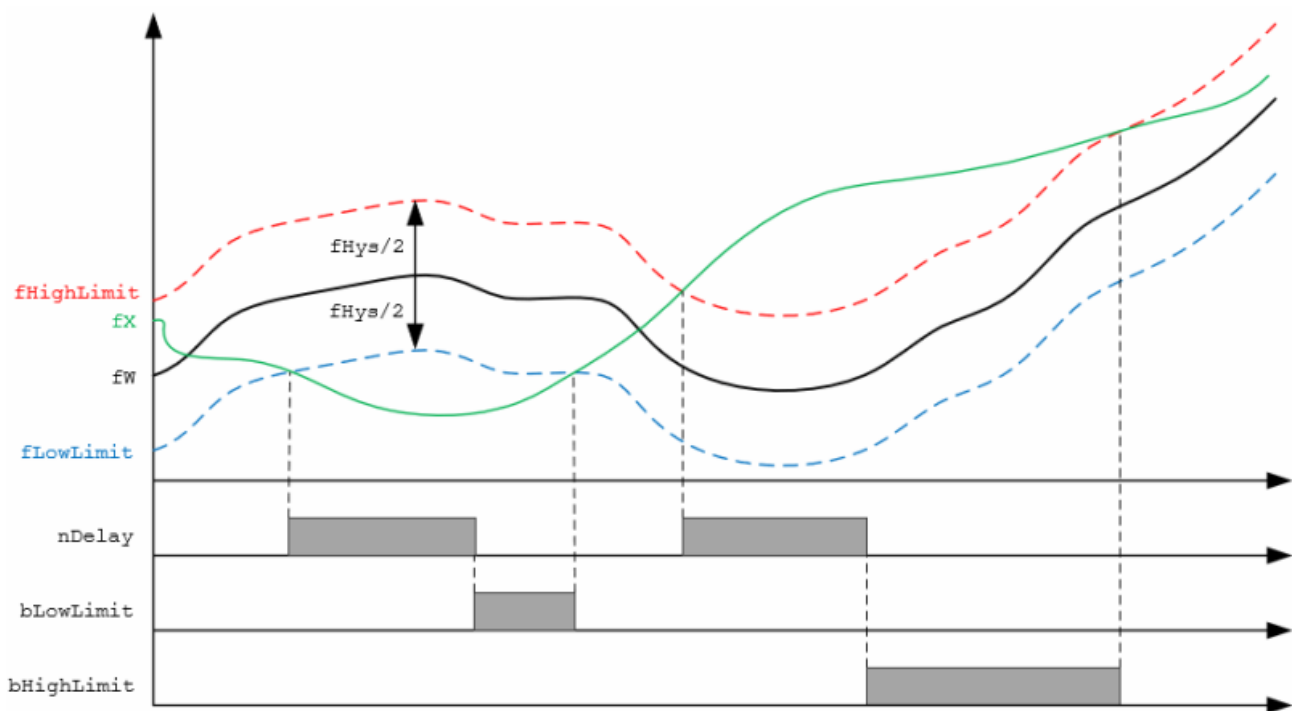
6.1.2.2.3.1.5.6.3 FB_BA_SlidingLimitCtrl



The function block *FB_BA_SlidingLimitCtrl* is used to monitor a sliding setpoint. The input *bEn* is used for general enabling of the function block.

To check the function of a control system, the actual value is compared with the setpoint of the controlled system.

If the deviation between the setpoint and the actual value is within the tolerance range *fHys*, then the control system is OK. If the actual value deviates from the setpoint by an amount outside this tolerance range over an extended period, the timer *nDelay* is started. After the timer has expired, if the control deviation remains, either the output *bLowLimit* or *bHighLimit* TRUE of the function block outputs a message.



Inputs

```

VAR_INPUT
    bEn      : BOOL;
    fW       : REAL;
    fX       : REAL;
    fHys     : REAL;
    nDelay   : UDINT;
ENDVAR

```

Name	Type	Description
bEn	BOOL	General enable of the function block. If <i>bEn</i> is FALSE, the message outputs <i>bHighLimit</i> and <i>bLowLimit</i> are also FALSE.
fW	REAL	Setpoint
fX	REAL	Actual value
fHys	REAL	Hysteresis
nDelay	UDINT	Response delay [s] of the outputs <i>bHighLimit</i> / <i>bLowLimit</i> . The input of the time is limited to the global parameter <i>BAComn_Global.udiMaxSecInMilli</i> from the Tc3_BA2_Common library (see <i>BAComn_Global</i>).

Outputs

```

VAR_OUTPUT
    bHighLimit : BOOL;
    bLowLimit  : BOOL;
    fHighLimit : REAL;
    fLowLimit  : REAL;
    nRemTiDelay : UDINT;
END_VAR

```

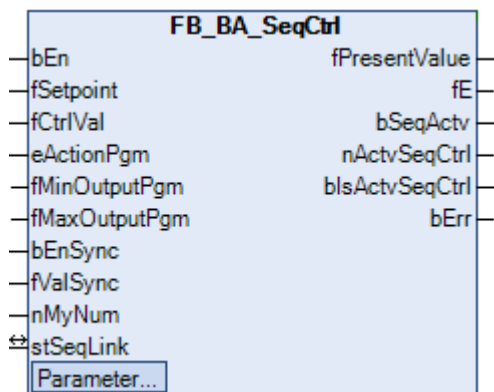
Name	Type	Description
bHighLimit	BOOL	Upper limit value reached.
bLowLimit	BOOL	Lower limit value reached.
fHighLimit	REAL	Output of the upper limit value. $fHighLimit = fW + (fHys / 2)$
fLowLimit	REAL	Output of the lower limit value. $fLowLimit = fW - (fHys / 2)$
nRemTiDelay	UDINT	Remaining time after exceeding a limit value until one of the outputs <i>bHighLimit</i> or <i>bLowLimit</i> is set. The default comes from <i>nDelay</i> .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.7 Sequence

6.1.2.2.3.1.5.7.1 FB_BA_SeqCtrl



PID sequence controller as part of a sequence.

On receipt of the enable *bEn* = TRUE, the higher-level control block *FB_BA_SequenceLinkBase* [► 411] is informed that the sequence controller is ready for operation for sequence control.

The data exchange between the sequence controllers *FB_BA_SeqCtrl* and the control block *FB_BA_SequenceLinkBase* [► 411] takes place via the structure *stSeqLink* [► 219].

Output value fPresentValue

The function block determines the resulting value at the output *fPresentValue* depending on the enables *bEn* and *stSequenceLink.bEnSeqLink*, the control direction *eActionPgm* / *eActionRm* and the sequence number *nActvSeqCtrl* / *nMyNum*.

bEn	stSe- quenceLink.bEnSe- qLink	eActionPgm/eAc- tionRm	nActvSeqCtrl/ nMyNum	fPresentValue
TRUE	FALSE	E_BA_Action.eReve rse		fMaxOutputPgm / fMaxOutputRm
TRUE	FALSE	E_BA_Action.eDirec t		fMinOutputPgm / fMinOutputRm
FALSE	FALSE			0
TRUE	TRUE	E_BA_Action.eReve rse	nActvSeqCtrl > nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eReve rse	nActvSeqCtrl < nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE	E_BA_Action.eDirec t	nActvSeqCtrl < nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eDirec t	nActvSeqCtrl > nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE		nActvSeqCtrl = nMyNum	FB_BA_Loop.fPrese ntValue

Error detection

The error messages listed below are detected by *FB_BA_SeqCtrl*.

The x in the text messages stands for a numerical specification of a sequence controller.

The error messages are output in the "Error list" window in the Tc3 development environment. This can be activated under the menu item View.

The error texts are output via the properties *ErrorParamMaxSeqCtrl* and *ErrorSeqNumMultiple*.



Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1

This message is the only one that disables sequence control.

German	English
Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1	Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1
Sequenznummer nMyNum = x mehrfach vergeben	Sequence number nMyNum = x assigned multiple times

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_Loop [► 154]

Illustration

```
FUNCTION_BLOCK FB_BA_SeqCtrl EXTENDS FB_BA_Loop
VAR_INPUT
    nMyNum          : UDINT;
END_VAR
VAR_OUTPUT
    fE              : REAL;
    bSeqActv        : BOOL;
    nActvSeqCtrl    : UDINT;
    bIsActvSeqCtrl  : BOOL;
    bErr            : BOOL;
END_VAR
```

```

VAR_IN_OUT
  stSeqLink      : ST_BA_SeqLink;
END_VAR

```

Inputs

Name	Type	Description
nMyNum	UDINT	

Outputs

Name	Type	Description
fE	REAL	Displays the control deviation of the sequence controller. This depends on the control direction of the sequence controller. E_BA_Action.eDirect -> fE = fX-fW E_BA_Action.eReverse -> fE = fW-fX
bSeqActv	BOOL	The sequence controller is implemented in the control sequence.
nActvSeqCtrl	UDINT	Number of the active sequence controller.
bIsActvSeqCtrl	BOOL	Indicates that the sequence controller is the active one in the sequence control. The property <i>IsActvSeqCtrl</i> also displays this message.
bErr	BOOL	Indicates that an error has been detected. More detailed information is shown in the properties <i>ErrorParamMaxSeqCtrl</i> and <i>SeqNumMultiple</i> . Further explanations on error analysis can be found at Error detection [► 409] .

/ Inputs Outputs

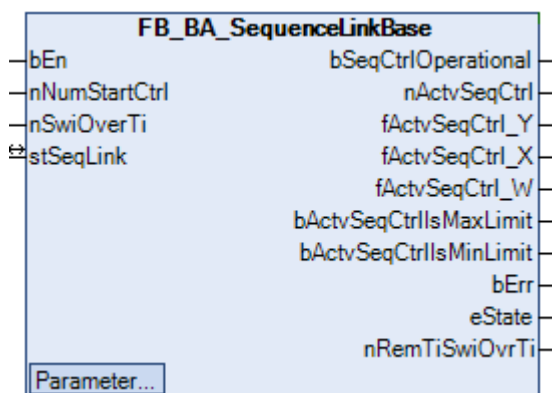
Name	Type	Description
stSeqLink	ST_BA_SeqLink [► 219]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [► 408] and the function block <i>FB_BA_SequenceLinkBase</i> .

 Properties

Name	Type	Access	Description
ActvSeqCtrl	UDINT	Get	The property shows the number of the active sequence controller.
ErrorParamMaxSeqCtrl	T_MaxString	Get	The property displays the following text in case of error: "Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1", see BA_Param.nMaxSeqCtrl [► 223]. The global parameter must be adjusted in any case, because due to the wrong parameterization the sequence stops.
SeqNumMultiple	T_MaxString	Get	The property displays the following text in case of error: "Sequence number nMyNum = x assigned multiple times". A check of the displayed sequence number is necessary because it has been assigned several times in the sequence.
IsActvSeqCtrl	BOOL	Get	The sequence controller is the active one in the sequence control.

Requirements

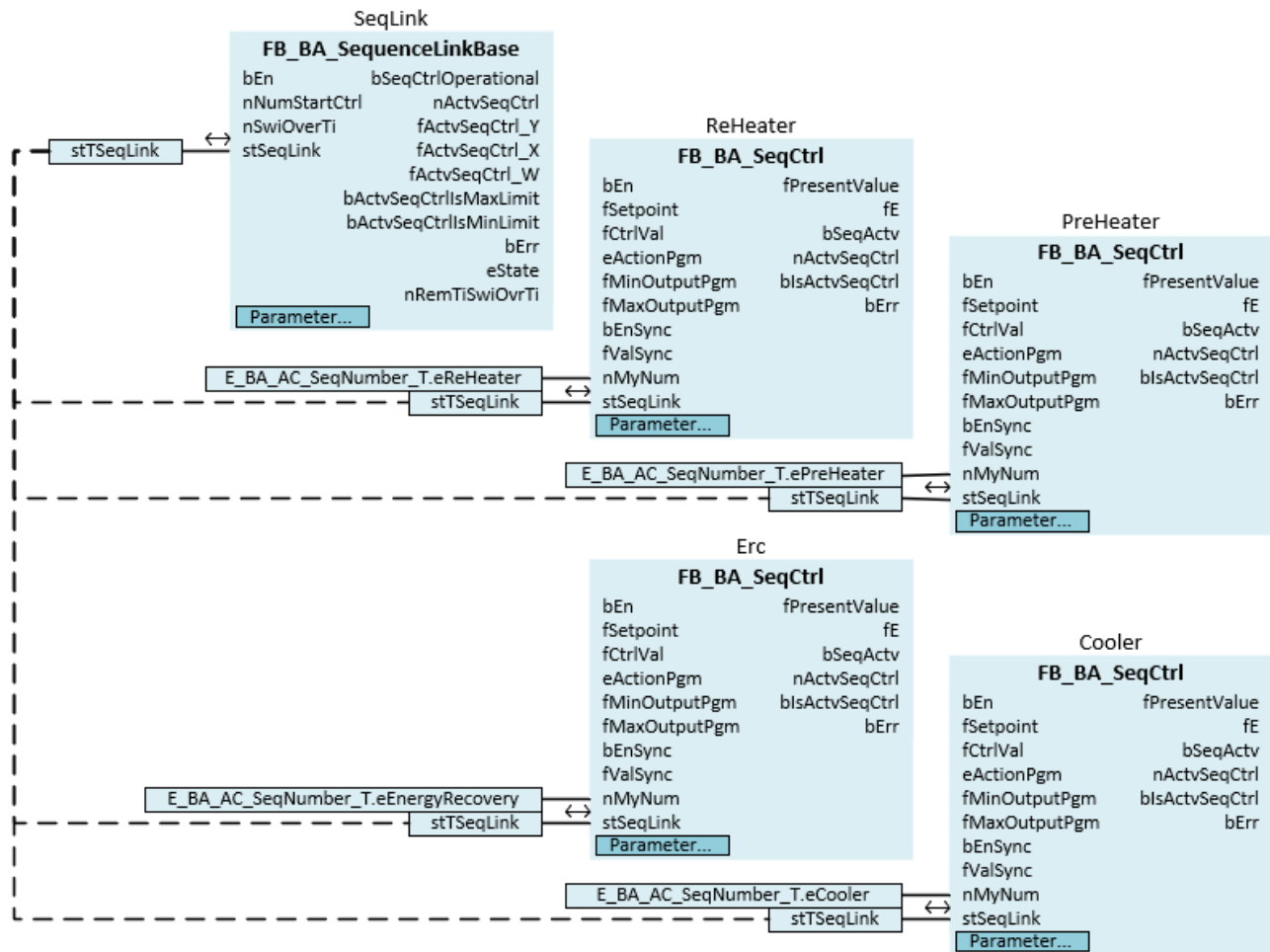
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.7.2 FB_BA_SequenceLinkBase

This function block represents the higher-level connection of a sequence control.

Data exchange sequence control

The data exchange between the sequence controllers [FB_BA_SeqCtrl \[► 408\]](#) and the [FB_BA_SequenceLinkBase](#) takes place via the structure [stSeqLink \[► 219\]](#).



Start behavior of the sequence

A TRUE signal at input *bEn* activates the entire sequence control. The function block will initially activate the sequence controller that is known at *nNumStartCtrl*. All other sequence controller base their output value [► 408] on the ranking of the active controller.

If the sequence controller specified at *nNumStartCtrl* is not ready for operation, the enum *eNoStartCtrlFound* [► 215] defines the procedure to find a new start controller. This ensures that the sequence always starts.

Switching in the sequence

If a sequence controller reaches its maximum or minimum value, the system switches to the next controller in the sequence depending on the control direction, provided that the actual value falls below or exceeds the setpoint of the next controller.

4 cases are distinguished:

- The still active controller has a direct control direction (cooling, *E_BA_Action.eDirect*) and is at its maximum value: The next higher controller is selected in the ranking order if the actual value exceeds the setpoint of this controller.
- The still active controller has a direct control direction (cooling, *E_BA_Action.eDirect*) and is at its minimum value: The next lower controller in the ranking order is selected if the actual value falls below the setpoint of this controller.
- The still active controller has an indirect control direction (heating, *E_BA_Action.eReverse*) and is at its maximum value: The next lower controller in the ranking order is selected if the actual value falls below the setpoint of this controller.
- The still active controller has an indirect control direction (heating, *E_BA_Action.eReverse*) and is at its minimum value: The next higher controller is selected in the ranking order if the actual value exceeds the setpoint of this controller.

Switching to another sequence controller can be delayed by specifying the time *nSwiOverTi*.

Operational readiness of the sequence controllers

If a controller loses its operational readiness in the sequence (missing enable *bEn*), it is no longer available for the entire sequence.

If this is not the previously active controller, a temperature change may occur, depending on which control value this controller has output, which is compensated by the controller sequence, if possible.

However, if it is the active sequence controller, the enumeration *eNoEnableSeqCtrl* [► 214] defines the procedure to find a new, active sequence controller.

If a sequence controller obtains its operational readiness in the running process, the controller is added to the sequence. Depending on the control direction *E_BA_Action* and its own sequence number *nMyNum* it will be placed in the controller sequence and output its minimum or maximum value. The resulting temperature change is compensated by the controller sequence, if possible.

Error detection

The error messages listed below are detected by *FB_BA_SequenceLinkBase*.

The x in the text messages stands for a numerical specification of a sequence controller.

The error messages are output in the "Error list" window in the Tc3 development environment. This can be activated under the menu item View.

Output of the error texts via the property *StateText*.

The enumeration *eState* [► 215] additionally displays the messages.



Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1

This message is the only one that disables sequence control.

German	English
Der nächste Sequenzsollwert ist kleiner als sein Vorgänger: x	Next sequence setpoint is smaller than its predecessor: x
Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1	Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1
Mehrfache Wirkrichtung "eActn" in der Sequenzsteuerung	Multiple direction of action "eActn" of the sequence controller
Startsequenzregler ist nicht freigegeben: nNumStartCtrl = x	Start sequence controller is not enabled: nNumStartCtrl = x
Kein Sequenzregler ist betriebsbereit	No sequence controller is operational
Sequenznummer mehrfach vergeben: x	Sequence number assigned multiple times: x

Illustration

```

FUNCTION_BLOCK FB_BA_SequenceLinkBase
VAR_INPUT
    bEn                : BOOL;
    nNumStartCtrl      : UDINT;
    {attribute 'parameterUnit':= 's'}
    nSwiOverTi         : UDINT(0..1800);
END_VAR
VAR_OUTPUT
    bSeqCtrlOperational : BOOL;
    nActvSeqCtrl        : UDINT;
    fActvSeqCtrl_Y      : REAL;
    fActvSeqCtrl_X      : REAL;
    fActvSeqCtrl_W      : REAL;
    bActvSeqCtrlIsMaxLimit : BOOL;
    bActvSeqCtrlIsMinLimit : BOOL;
    bErr                : BOOL;
    eState              : E_BA_StateSeqLink;
    {attribute 'parameterUnit':= 's'}

```

```

    nRemTiSwiOvrTi      : UDINT;
END_VAR
VAR_IN_OUT
    stSeqLink           : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
    fHys                : REAL := 0.2;
    eNoStartCtrlFound   : E_BA_NoStartCtrlFound := E_BA_NoStartCtrlFound.eLastActi
onReverse;
    eNoEnableSeqCtrl    : E_BA_NoEnableSeqCtrl := E_BA_NoEnableSeqCtrl.eNextAction
;
END_VAR

```

Inputs

Name	Type	Description
bEn	BOOL	Sequence control is activated when the function block is enabled. However, the sequence controllers must first be enabled.
nNumStartCtrl	UDINT	Ordinal number of the sequence controller, which should be the start controller when the sequence is activated. An internal limitation allows only values from 1 - BA Param.nMaxSeqCtrl [► 223] .
nSwiOverTi	UDINT	Switching to another sequence controller can be delayed by the time specification. An internal limitation only allows values from 0 - 1800 seconds.

Outputs

Name	Type	Description
bSeqCtrlOperational	BOOL	The sequence control is ready to operate or is in operation.
nActvSeqCtrl	UDINT	The output variable shows the number of the active sequence controller.
fActvSeqCtrl_Y	REAL	Control value of the active sequence controller.
fActvSeqCtrl_X	REAL	Actual value of the active sequence controller.
fActvSeqCtrl_W	REAL	Setpoint of the active sequence controller.
bActvSeqCtrlIsMaxLimit	BOOL	The upper output limit of the active sequence controller has been reached. If it has not yet been advanced to the next sequence controller, <i>bActvSeqCtrlIsMaxLimit</i> is not set to FALSE until <i>fActvSeqCtrl_Y</i> has fallen below (<i>fActvSeqCtrl_Y</i> - <i>fHys</i>).
bActvSeqCtrlIsMinLimit	BOOL	The lower output limit of the active sequence controller has been reached. If it has not yet been advanced to the next sequence controller, <i>bActvSeqCtrlIsMinLimit</i> is not set to FALSE until <i>fActvSeqCtrl_Y</i> has risen above (<i>fActvSeqCtrl_Y</i> + <i>fHys</i>).
bErr	BOOL	Indicates that an error has been detected. More detailed information is provided by the property <i>StateText</i> . In addition, the enum <i>eState</i> [► 215] displays further information. Further explanations on error analysis can be found at Error detection [► 413] .
eState	E_BA_StateSeqLink [► 215]	The enumeration shows the state of the <i>FB_BA_SequenceLinkBase</i> and the sequence control.
nRemTiSwiOvrTi	UDINT	Countdown of switching to the next sequence controller [s].

 / Inputs Outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [► 219]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [► 408] and the function block <i>FB_BA_SequenceLinkBase</i> .

 Inputs CONSTANT

Name	Type	Description
fHys	REAL	This hysteresis value sets a hysteresis around the upper and lower output limit <i>bActvSeqCtrlsMaxLimit</i> / <i>bActvSeqCtrlsMinLimit</i> of the active sequence controller. An internal limitation only allows values from 0 - 2.
eNoStartCtrlFound	E_BA_NoStartCtrlFound [► 215]	If during the running process the active sequence controller loses its operational readiness, then the enumeration defines the procedure to find a new, active sequence controller.
eNoEnableSeqCtrl	E_BA_NoEnableSeqCtrl [► 214]	If the start controller at the input <i>nNumStartCtrl</i> is not available or enabled, the enumeration defines the procedure to find a new start controller.

 Properties

Name	Type	Access	Description
FirstActvSeqCtrl	UDINT	Get	The property shows the number of the first sequence controller ready for operation.
FirstActvSeqCtrlWithActionDirect	UDINT	Get	The property shows the number of the first sequence controller ready for operation with the control direction <i>E_BA_Action.eDirect</i> .
LastActvSeqCtrl	UDINT	Get	The property shows the number of the last sequence controller ready for operation.
LastActvSeqCtrlWithActionReverse	UDINT	Get	The property shows the number of the last sequence controller ready for operation with the control direction <i>E_BA_Action.eReverse</i> .
NextActionActvSeqCtrl	UDINT	Get	Depending on the control direction of the active sequence controller <i>nActvSeqCtrl</i> the next sequence controller ready for operation is displayed here. Cooling/Direct control direction: next sequence controller ready for operation with the Cooling/Direct control direction (after 5 would come 6 or higher). If none is found, then <i>E_BA_NoEnableSeqCtrl.eLastSeqNum</i> is implemented. Heating/reverse control direction: previous operational sequence controller with heating/reverse control direction (after 5 would come 4 or lower). If none is found, then <i>E_BA_NoEnableSeqCtrl.eFirstSeqNum</i> is implemented.
NextActvSeqCtrl	UDINT	Get	The property shows the number of the next sequence controller ready for operation after the last active one (after 5 would come 6 or higher).
PreviousActvSeqCtrl	UDINT	Get	The property shows the number of the previous sequence controller ready for operation after the last active one (after 5 would come 4 or lower).
ActvSeqCtrl	UDINT	Get	The property shows the number of the active sequence controller, see <i>nActvSeqCtrl</i> .
ActvSeqCtrl_ControlValue	REAL	Get	The property shows the control value of the active sequence controller, see <i>fActvSeqCtrl_Y</i> .
ActvSeqCtrl_IsMaxLimit	BOOL	Get	The property indicates that the upper output limit of the active sequence controller has been reached, see <i>bActvSeqCtrlIsMaxLimit</i> .
ActvSeqCtrl_IsMinLimit	BOOL	Get	The property indicates that the lower output limit of the active sequence controller has been reached, see <i>bActvSeqCtrlIsMinLimit</i> .
ErrorParamMaxSeqCtrl	T_MaxString	Get	The property displays the following text in case of error: "Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1", see BA_Param.nMaxSeqCtrl [► 223]. The global parameter must be adjusted in any case, because due to the wrong parameterization the sequence stops.
ErrorStartSeqCtrlNotOperable	T_MaxString	Get	The property displays the following text in case of error: "Start sequence controller is not enabled: nNumStartCtrl = x". A check of the start sequence controller, which was specified with the input variable <i>nNumStartCtrl</i> , is required.

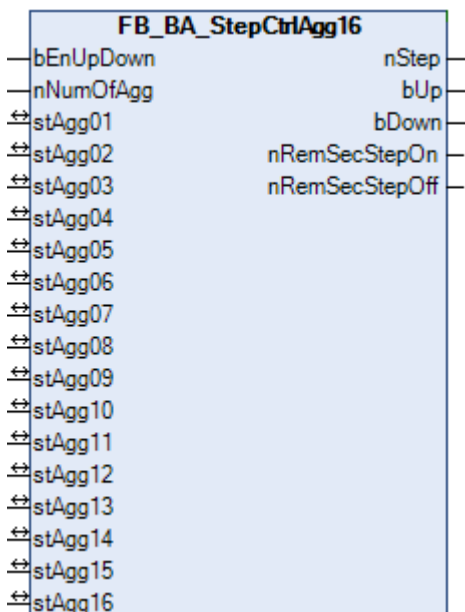
Name	Type	Access	Description
MultiDirectionActionSeqCtrl	T_MaxString	Get	The property displays the following text in case of error: Multiple direction of action "eActn" of the sequence controller: x. Multiple direction of action of the sequence controller. It always looks from the sequence controller with the smallest number to the next largest one. It is possible to change the direction of action.
NextSeqSplsSmaller	T_MaxString	Get	The property displays the following text in case of error: "Next sequence setpoint is smaller than its predecessor: x". A check of the setpoints of the sequence controllers according to ascending order is required.
NoSeqCtrlIsOperational	T_MaxString	Get	The property displays the following text in case of error: "No sequence controller is operational". A check of the operational readiness of the sequence controllers is required.
SeqNumMulti	T_MaxString	Get	The property displays the following text in case of error: "Sequence number assigned multiple times: x". A check of the displayed sequence number is necessary because it has been assigned several times in the sequence.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.8 StepControl

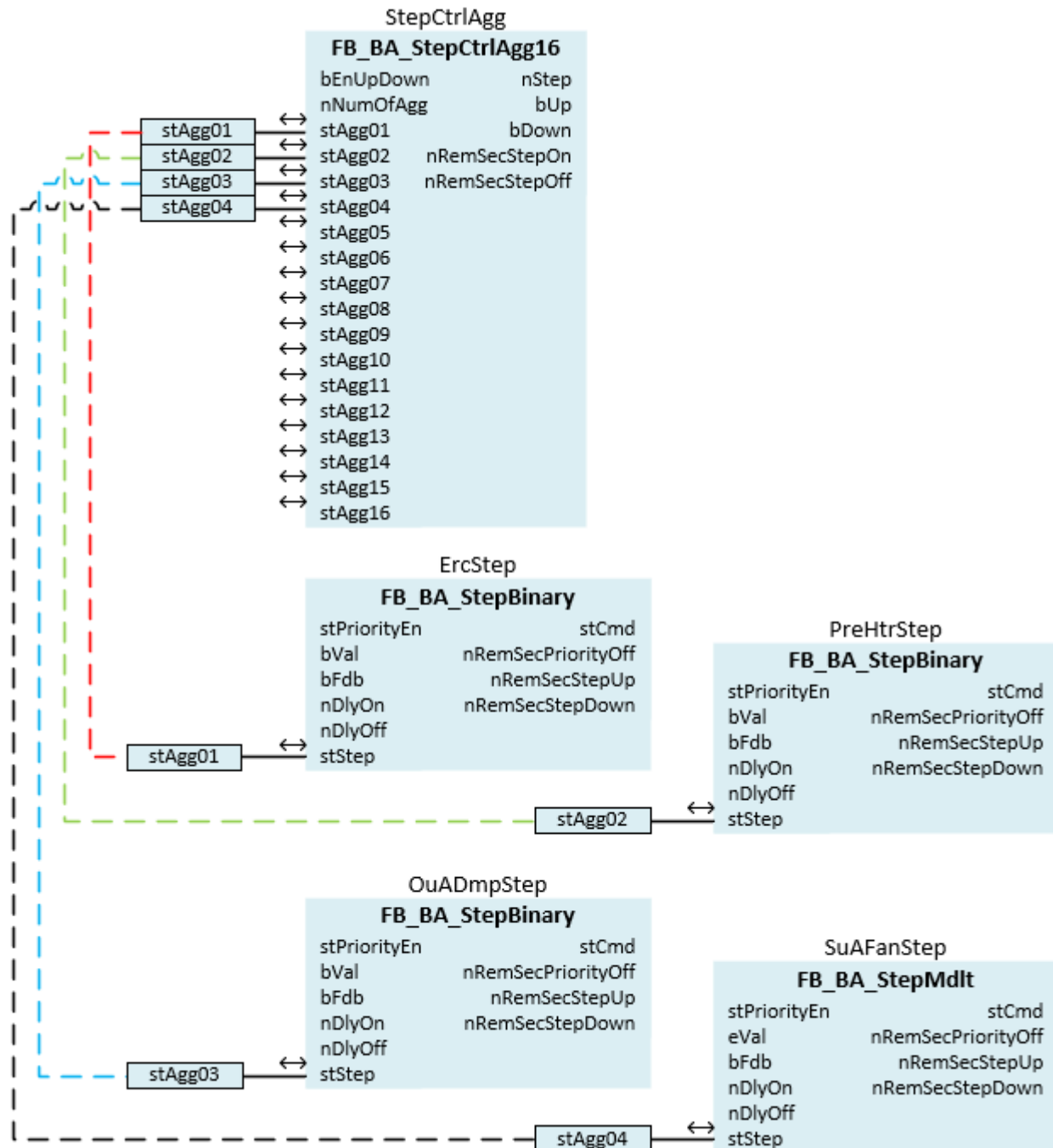
6.1.2.2.3.1.5.8.1 FB_BA_StepCtrlAgg16



The function block FB_BA_StepCtrlAgg16 represents the higher-level control unit of a step sequence control of aggregates.

The control unit *FB_BA_StepCtrlAgg16* can be used to sequentially switch on (stAgg01 > stAgg02 > stAgg03 > stAgg04 ... stAgg16) or switch off (stAgg16 > stAgg15 > stAgg14 > stAgg13 ... stAgg01) individual aggregates of a plant in a specific order.

Data exchange between the receive blocks of the step sequence control (*FB_BA_StepBinary* [► 420], *FB_BA_StepMdlIt* [► 423]) and the control unit *FB_BA_StepCtrlAgg16* takes place via the data and command structures *stAggxx*.



Illustration

```
FUNCTION_BLOCK FB_BA_StepCtrlAgg16
VAR_INPUT
    bEnUpDown      : BOOL;
    nNumOfAgg      : UDINT;
END_VAR
VAR_OUTPUT
    nStep          : UDINT;
    bUp            : BOOL;
    bDown         : BOOL;
    nRemSecStepOn  : UDINT;
    nRemSecStepOff : UDINT;
```



```

END_VAR
VAR_IN_OUT
    stAgg01-16      : ST_BA_Step;
END_VAR

```

🔌 Inputs

Name	Type	Description
bEnUpDown	BOOL	A rising edge means an increasing switch-on sequence of the aggregates from 1 to 16. FALSE means a decreasing switch-off sequence from the highest, active aggregate towards 0. 0 means that no aggregate of the step sequence control is active.
nNumOfAgg	UDINT	Input of the number of aggregates of the step sequence control. A limit only allows values from 1 - 16.

🔌 Outputs

Name	Type	Description
nStep	UDINT	Display in which step the step sequence control is located. 0 means that no aggregate of the step sequence control is active.
bUp	BOOL	Indicates that the sequence of the step sequence control is in the rising state.
bDown	BOOL	Indicates that the sequence of the step sequence control is in the falling state.
nRemSecStepUp	UDINT	Countdown switching to the next higher level [s].
nRemSecStepDown	UDINT	Countdown switching to the next lower level [s].

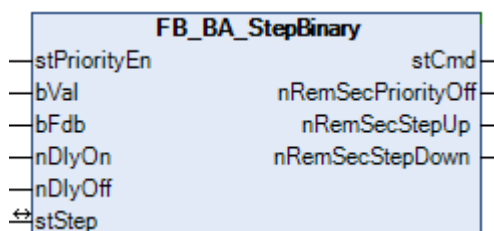
🔌 / 🔌 Inputs Outputs

Name	Type	Description
stAgg01-16	ST_BA_Step [▶ 222]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [▶ 408] and the function block FB_BA_SequenceLinkBase .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.8.2 FB_BA_StepBinary



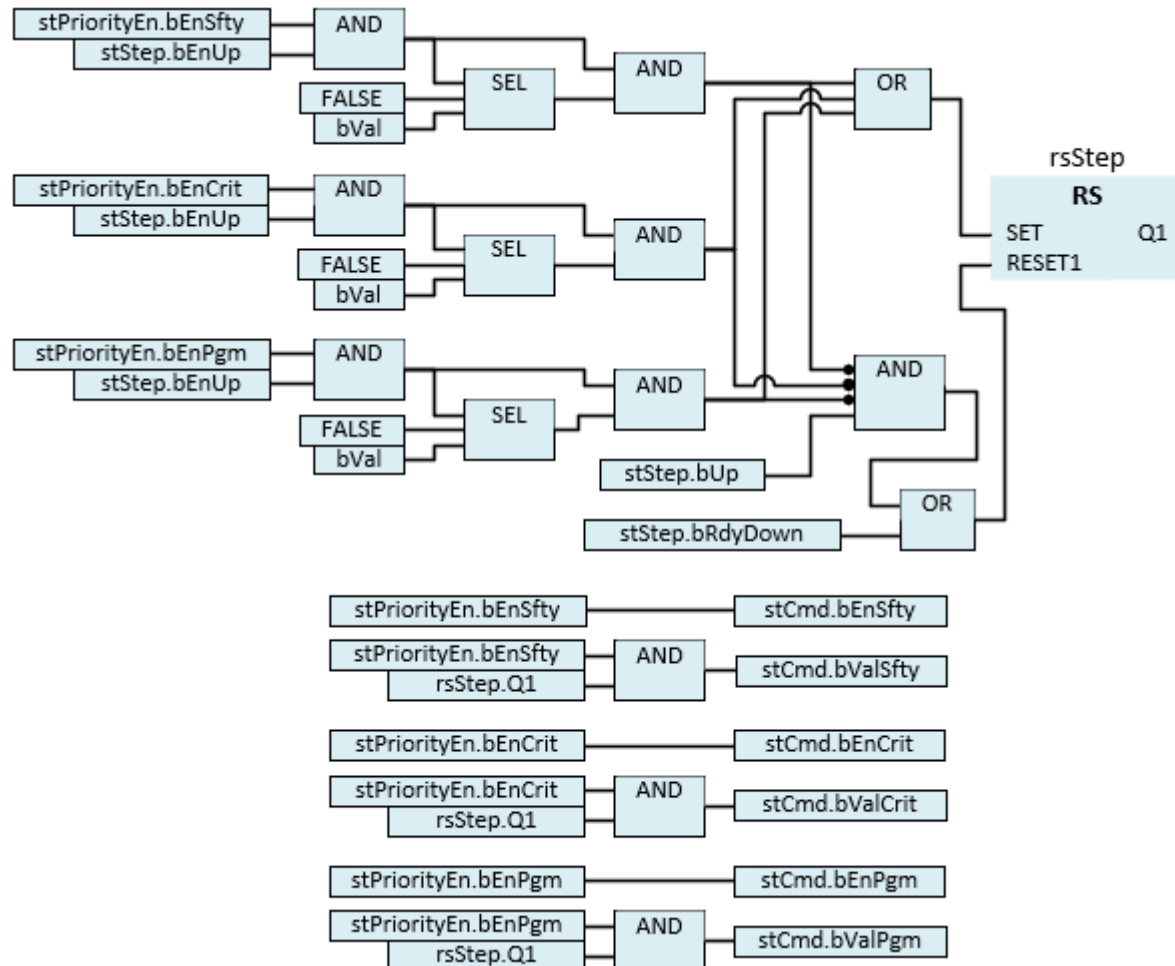
The function block is part of a step sequence control.

It communicates via the data and command structure *stStep* with the higher-level control unit of the step sequence control *FB BA StepCtrlAgg16* [► 418].

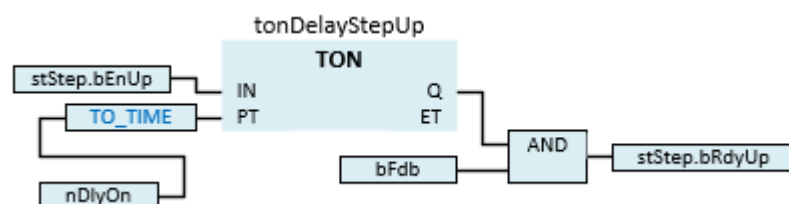
The command structure *stPriorityEn* contains the plant enable and the priorities.

The command structure *stCmd* controls the connected aggregate.

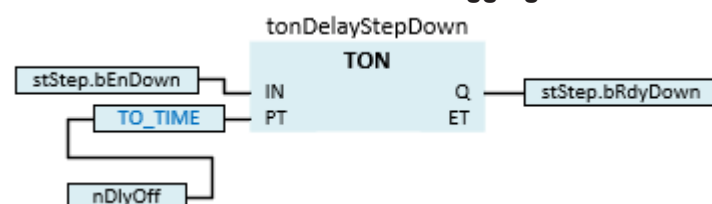
Switch-on conditions of the connected aggregate



Step enabling condition within the step sequence



Switch-off conditions of the active aggregate within the step sequence



Illustration

```

FUNCTION_BLOCK FB_BA_StepBinary
VAR_INPUT
    stPriorityEn      : ST_BA_PriorityEn;
    bVal              : BOOL;
    bFdb              : BOOL;
    nDlyOn            : UDINT;
    nDlyOff           : UDINT;
END_VAR
VAR_OUTPUT
    stCmd             : ST_BA_Binary;
    nRemSecStepUp     : UDINT;
    nRemSecStepDown   : UDINT;
END_VAR
VAR_IN_OUT
    stStep            : ST_BA_Step;
END_VAR

```

Inputs

Name	Type	Description
stPriorityEn	ST_BA_PriorityEn  223	The command structure includes the plant enable and the priorities "Safety", "Critical" and "Program".
bVal	BOOL	Switch-on value for the connected aggregate.
bFdb	BOOL	Feedback of the connected aggregate. The feedback is required for switching to the next step. <i>bFdb</i> is only considered if the aggregate is the active one in the step sequence control.
nDlyOn	UDINT	Time specification of start-up delay [s]. The time specification is required for switching to the next step.
nDlyOff	UDINT	Time specification of switch-off delay [s]. The time specification is needed to switch off the active step.

Outputs

Name	Type	Description
stCmd	ST_BA_Binary  221	The command structure <i>stCmd</i> transmits the enables and switching values of the priorities to the connected aggregate.
nRemSecStepUp	UDINT	Countdown switching to the next higher level [s].
nRemSecStepDown	UDINT	Countdown switching to the next lower level [s].

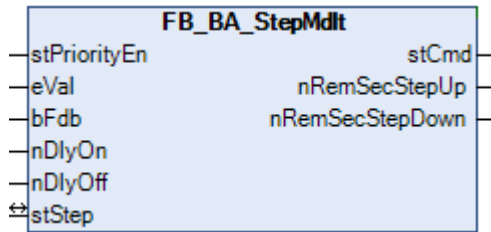
/ Inputs Outputs

Name	Type	Description
stStep	ST_BA_Step  222	Data and command structure between the step sequence function block <i>FB_BA_StepBinary</i> and the control block of the step sequence FB_BA_StepCtrlAgg16  418 .

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.8.3 FB_BA_StepMdl



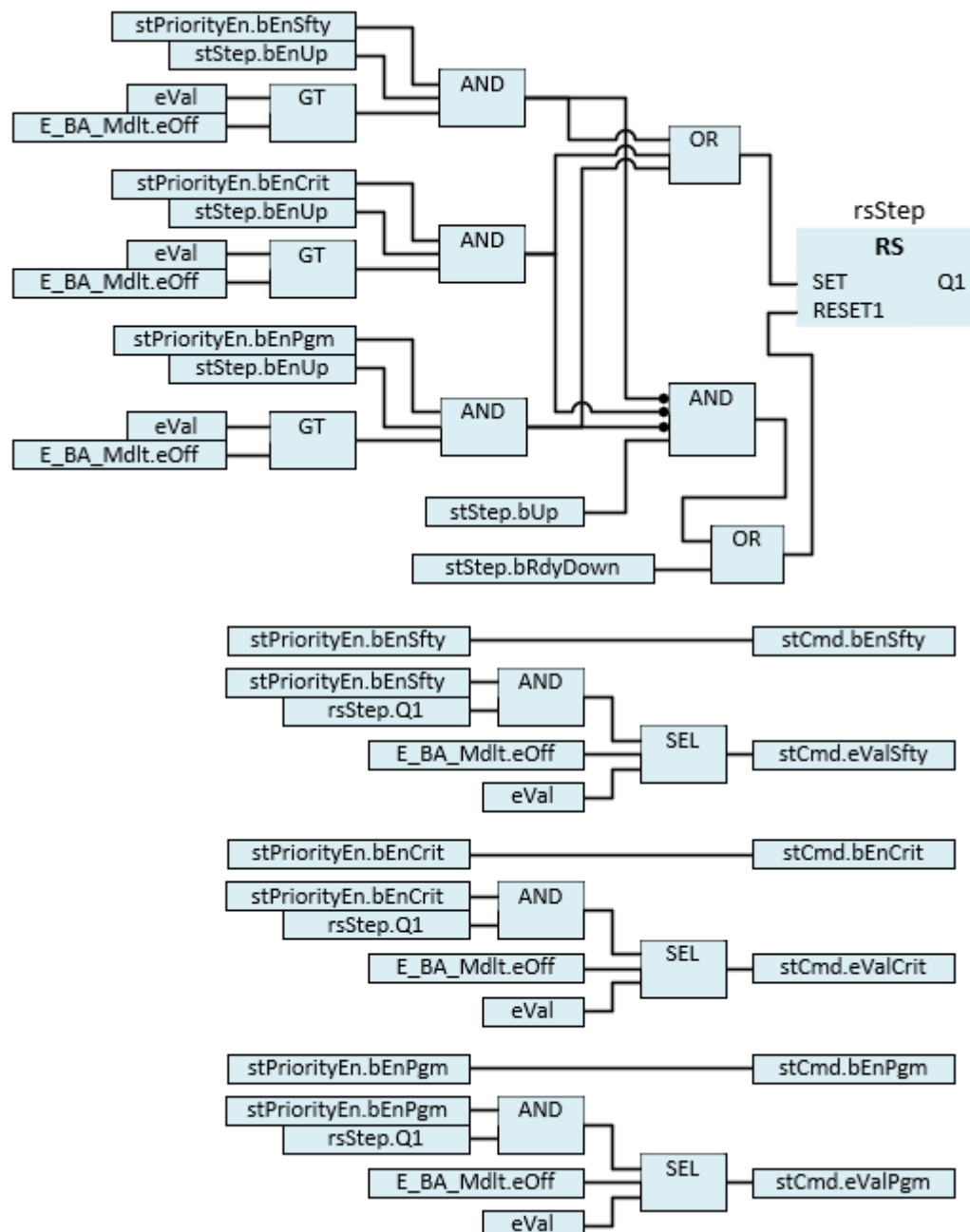
The function block is part of a step sequence control.

It communicates via the data and command structure *stStep* with the higher-level control unit of the step sequence control *FB_BA_StepCtrlAgg16* [► 418].

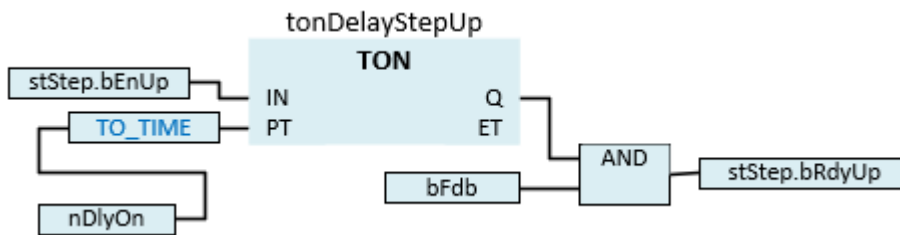
The command structure *stPriorityEn* contains the plant enable and the priorities.

The command structure *stCmd* controls the connected aggregate.

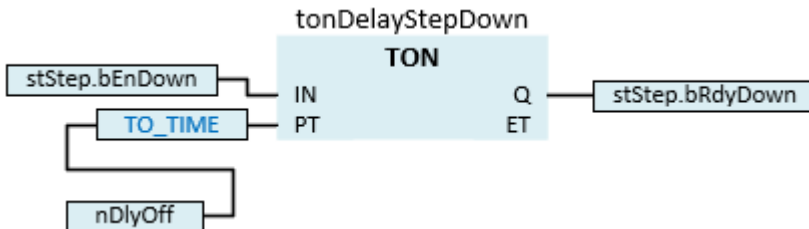
Switch-on condition of the connected aggregate



Step enabling condition within the step sequence



Switch-off condition of the active aggregate within the step sequence



Inputs

Name	Type	Description
stPriorityEn	ST_BA_PriorityEn [► 223]	The command structure includes the plant enable and the priorities "Safety", "Critical" and "Program".
eVal	E_BA_Mdlt [► 214]	Switch-on value for the connected aggregate.
bFdb	BOOL	Feedback of the connected aggregate. The feedback is required for switching to the next step. <i>bFdb</i> is only considered if the aggregate is the active one in the step sequence control.
nDlyOn	UDINT	Time specification of start-up delay [s]. The time specification is required for switching to the next step.
nDlyOff	UDINT	Time specification of switch-off delay [s]. The time specification is needed to switch off the active step.

Outputs

Name	Type	Description
stCmd	ST_BA_Mdlt [► 221]	The command structure <i>stCmd</i> transmits the enables and switching values of the priorities to the connected aggregate.
nRemSecStepUp	UDINT	Countdown switching to the next higher level [s].
nRemSecStepDown	UDINT	Countdown switching to the next lower level [s].

Inputs Outputs

Name	Type	Description
stStep	ST_BA_Step [► 222]	Data and command structure between the step sequence function block <i>FB_BA_StepBinary</i> and the control block of the step sequence <i>FB_BA_StepCtrlAgg16</i> [► 418].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.4 Workflows

6.1.2.2.4.1 Referencing

6.1.2.2.4.1.1 Collectors

FB_BA_Collector collects values and states from one or more referenced [objects \[► 28\]](#) of uniform types.

In this way, different values and states (such as heat requests or operation mode collection of plants and aggregates) can be collected.

Programming

Use cases

Depending on the data type of the referenced [objects \[► 28\]](#) different states and values are output at the function block output:

- **Active**

Consideration of at least one referenced [object \[► 28\]](#) due to an active value.

- **ActiveCount**

Number of active referenced [objects \[► 28\]](#).

- **MinVal** (Separate for *Analog* and *Multistate*)

Smallest value of all active referenced [objects \[► 28\]](#).

- **MaxVal** (Separate for *Analog* and *Multistate*)

Largest value of all active referenced [objects \[► 28\]](#).

- **AvgVal**

Average value of all active referenced [objects \[► 28\]](#).

Depending on the configuration of the object references, the FB operates under one of the following control modes:

- **Analog, Binary, Multistate**

Collects values from *active* [objects \[► 28\]](#) of the respective type.

An [object \[► 28\]](#) is active as soon as the following condition applies:

```
PresentValue <> DefaultValue
```

- **En, EnAnalog, EnBinary, EnMultistate**

Collects values from *enabled* [objects \[► 28\]](#) of the respective type.

An [object \[► 28\]](#) is active as soon as the *enable reference* has the value TRUE. The *value reference* is now taken into account in the calculation of the function block outputs.

Initialization

Variant 1) Direct referencing

```

{region 'Analog'}
    ColAV_Op1      : FB_BA_AV_Op;
    ColAV_Op2      : FB_BA_AV_Op;
    ColAV_Collect    : FB_BA_Collector      := (iReferences := [ ColAV_Op1, ColAV_Op2 ]);
{endregion}
{region 'Binary'}
    ColBV_1        : FB_BA_BV;
    ColBV_Op2      : FB_BA_BV_Op;
    ColBV_Collect    : FB_BA_Collector      := (iReferences := [ ColBV_1, ColBV_Op2 ]);
{endregion}
{region 'Multistate'}
    ColMV_1        : FB_BA_MV;
    ColMV_2        : FB_BA_MV;
    ColMV_Collect    : FB_BA_Collector      := (iReferences := [ ColMV_1, ColMV_2 ]);
{endregion}

```

Variant 2) Indirect referencing

```

ColEn_View1      : FB_BA_View;
    En1          : FB_BA_BV_Op      := (iParent := ColEn_View1);
    Val1         : FB_BA_AV_Op      := (iParent := ColEn_View1);
ColEn_View2      : FB_BA_View;
    En01         : FB_BA_BV_Op      := (iParent := ColEn_View2);
    Val01        : FB_BA_AV_Op      := (iParent := ColEn_View2);

ColEn_CollectA     : FB_BA_Collector := (
    iReferences      := [ ColEn_View1, ColEn_View2 ],
    sRefSubjectEn    := 'En',
    sRefSubjectVal    := 'Val'
);

```

The subject references of the referenced [objects](#) [► 28] are mapped (depending on the desired *control mode*) by the corresponding variables *sRefSubjectEn* and *sRefSubjectVal*.

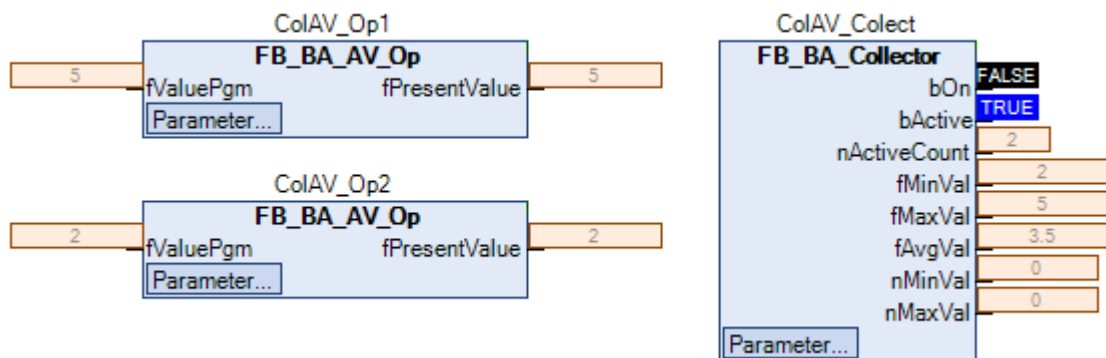
Samples

Variant 1) Direct referencing

```

{region 'Analog'}
    ColAV_Op1      : FB_BA_AV_Op;
    ColAV_Op2      : FB_BA_AV_Op;
    ColAV_Collect    : FB_BA_Collector      := (iReferences := [ ColAV_Op1, ColAV_Op2 ]);
{endregion}

```



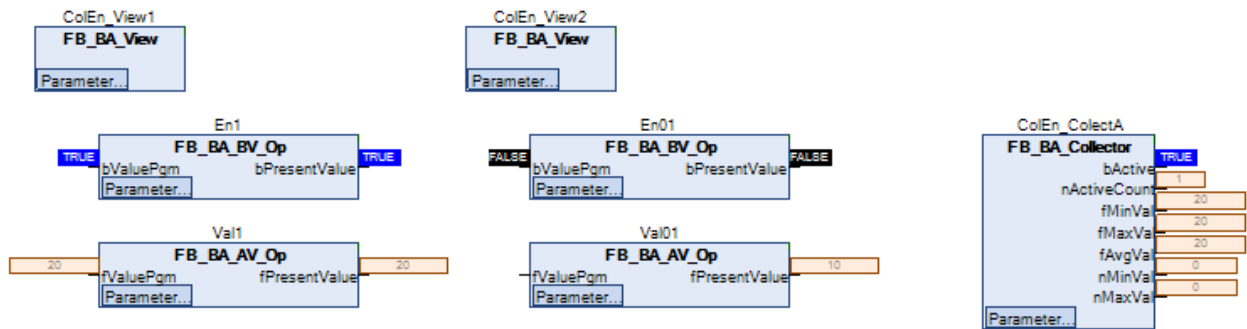
Variant 2) Indirect referencing

```

ColEn_View1      : FB_BA_View;
  En1             : FB_BA_BV_Op      := (iParent := ColEn_View1);
  Val1           : FB_BA_AV_Op      := (iParent := ColEn_View1);
ColEn_View2      : FB_BA_View;
  En01           : FB_BA_BV_Op      := (iParent := ColEn_View2);
  Val01          : FB_BA_AV_Op      := (iParent := ColEn_View2);

ColEn_CollectA    : FB_BA_Collector := (
  iReferences      := [ ColEn_View1, ColEn_View2 ],
  sRefSubjectEn    := 'En',
  sRefSubjectVal   := 'Val'
);

```



Further information

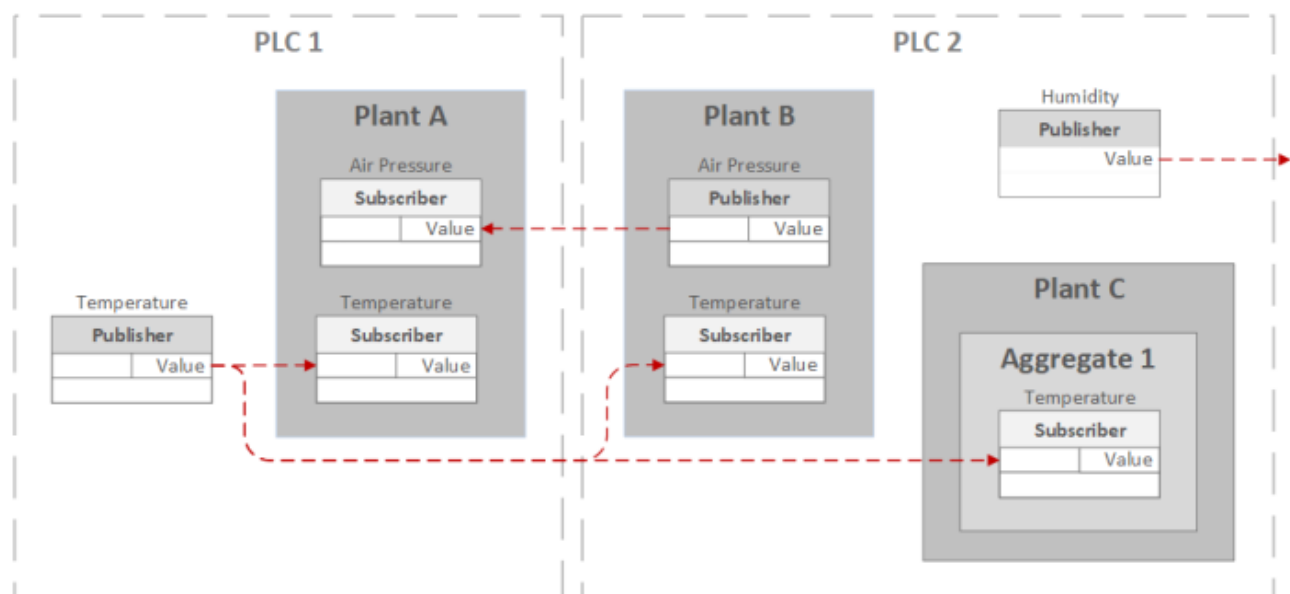
- Error messages

6.1.2.2.4.1.2 Publisher and Subscriber

Publisher and Subscriber provide a way to easily exchange information.

- Publishers provide information.
- Subscribers subscribe to information from a referenced publisher in order to make it available in a specific program section (e.g. template).

Regardless of where Publisher and Subscriber are implemented (e.g. on different PLCs), information is exchanged in a uniform manner.



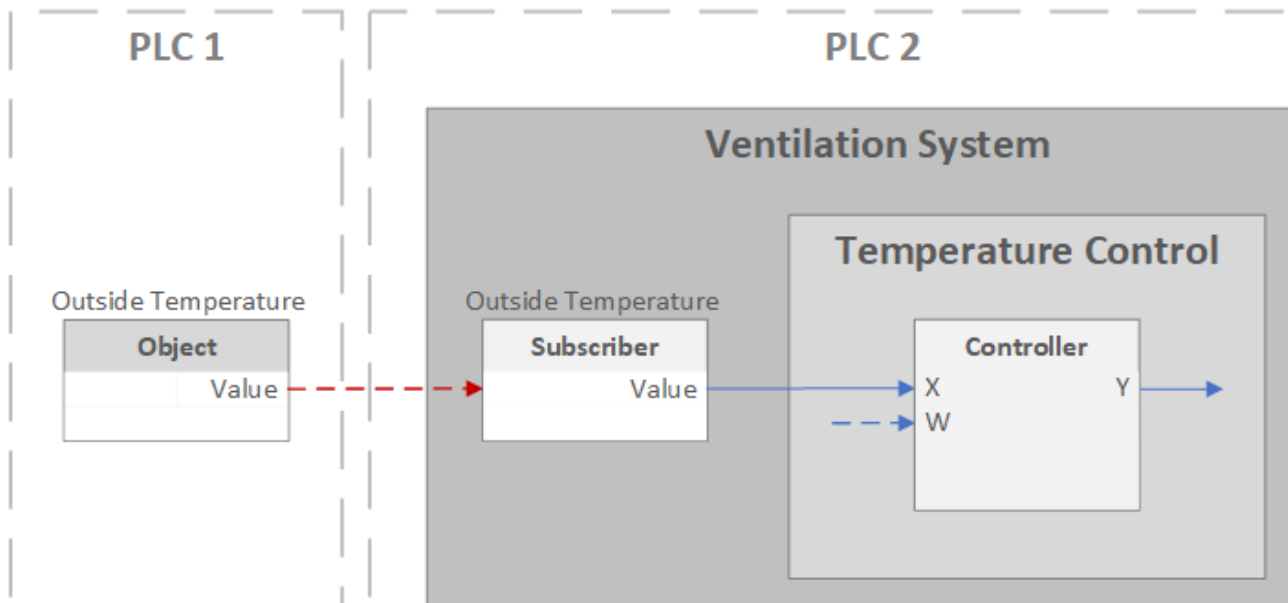
Identification

The **connection between object and its application purpose** is known to the integrator when he sees the object's designation.

This connection is not directly established in the PLC, which exclusively processes the program code.

In order to get access to information of a specific publisher from any place, it is necessary to make the publisher uniquely identifiable.

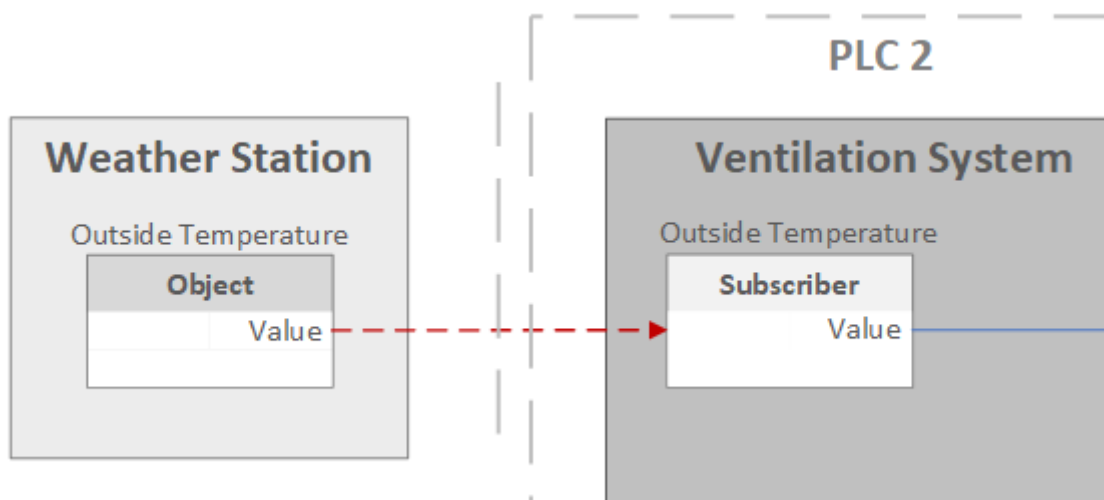
This is achieved by uniform designation of publishers or objects.



Example: Subscribing to a specific (uniquely designated) object - the outdoor temperature.

By adhering to uniform designations, independent development is made possible.

Thus, the ventilation system illustrated above would also be able to subscribe to values of the object *Outdoor temperature* if this were not a simple sensor but a weather station, which however provides an object with the same designation.



For the subscriber it is irrelevant where the outdoor temperature is located - but it must be available.

Programming

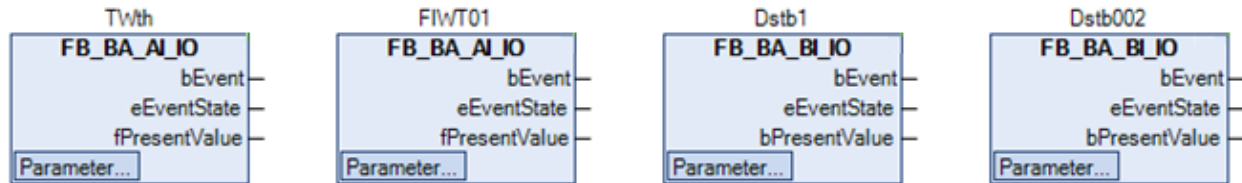
Subject information

Subject information describes an (arbitrary) *publisher* or *object*.

The *publisher* thus becomes a (very concrete) *subject* that can be uniquely identified.

The symbol name of a *publisher* is used to determine *identifier* and *index*.

In order to be able to carry out comparison operations more efficiently, a hash is formed via the identifier.



Mapping of different objects with partly the same subject information.

Subject information is generated from the *symbol names* of the objects as follows:

Symbol name	Identifier	Index
TWth	TWth	1
Flwt	Dstb1	1
Dstb1	Dstb1	1
Dstb002	Dstb002	2

Now the following options are available:

- **Referencing**

Referencing specific objects using *subject information*.

- **Queries**

Search plants for specific objects.

```
MyPlant.QueryObjectSbj('Pu', 2)
```

- **Case distinctions**

Iterate over all objects in a plant to perform *subject-dependent* operations.

```
WHILE (MyPlant.IterateObjects(_iChild)) DO
  CASE (_iChild.SubjectInfo.sIdentifier) OF
    'TWth': // ...
    'Dstb': // ...
  ELSE
    // ...
  END_CASE
END_WHILE
```

From this principle, use cases such as the following are possible:

- An HMI or **operator workstation** searches the PLC for plants and aggregates to automatically **generate plant graphics**.
- A **generic navigation** through project structures and systems in applications (e.g. the Terminal Explorer).

Special cases

The following special characters are not allowed if they are **at the end** of the identifier (in this case they are ignored or truncated):

_ -

The following overview compares the *symbol name* with the expected *subject information*:

Symbol name	Identifier	Index
ISP_10	ISP	10
_Building05	_Building	5

Subject references

In order to refer to a Publisher, or Object (example: reference in a Collector or Subscriber), a *reference* is expressed in an appropriate syntax:

Use cases

The use cases listed here illustrate the application of *references* using subscribers. The same applies to other references (e.g. in [Collectors](#) [► 425]).

- **Simple Referencing**
A publisher with unique subject information is referenced.

```
TF1_Sub      : FB_BA_SubAI := (
                sSubject      := 'TF1',
            );
```

- Referencing **the same** subjects
If several publishers with the same subject identifier are subscribed to, their indices must be defined:

```
TF101       : FB_BA_SubAI := (
                bEnPublish      := TRUE
            );
TF102       : FB_BA_SubAI := (
                bEnPublish      := TRUE
            );
```

```
TF1_Sub      : FB_BA_SubAI := (
                sSubject        := 'TF12'
            );
```

- Referencing the **same, nested** subjects
If several publishers with the same subject information are subscribed to, their *subject paths* must be defined:

```
1  FUNCTION_BLOCK FB_BA_Pump EXTENDS FB_BA_View
2  VAR_INPUT CONSTANT
3      Dstb      : FB_BA_BI_IO := (
4                  iParent      := THIS^,
5                  bEnPublish    := TRUE
6              );
7  END_VAR

Pump1        : FB_BA_Pump;
Pump2        : FB_BA_Pump;

Dstb2_Sub    : FB_BA_SubBI := (
                sSubject        := 'Pump2.Dstb'
            );
```

The subject *Dstb* would not be clear enough in this case.



Subject paths always refer to the *symbol path* of a Publisher, **not** to its *object path* within the project structure.



Subject paths should be *relative*. Paths that are deeper than what is necessary to establish uniqueness should be avoided, since their resolution requires computing capacity.

Publisher



Every object is automatically also a publisher.

Enable publishing

In order to allow *publishing* (the *sharing* of information) for one or more *subscribers*, this must first be *enabled*.

- Enable

```
TF1      : FB_BA_AI := (  
          bEnPublish := TRUE  
        );
```

Example: Enable for the current value of a flow temperature.



The *publishing* must be enabled separately for individual *publishers*.

Subscriber

Setting up a subscription

The *subscription* (connection from a *subscriber* to a publisher) can be established if the subject reference of the *subscriber* matches the subject information of the publisher.



Requirements for a valid *subscription* are:

- The **data type** of the *subscriber* and the subscribed publisher are identical.
 - The **data** of the *subscriber* and the subscribed publisher have the same size.
-



Practically, any number of *subscribers* can be created.

The more *subscriptions* are created, the more computing power is used for the subscription.

For more information, refer to [Best Practice \[► 435\]](#).

Local Subscriptions

Before *subscribing* (the *subscribing* of information) to a publisher is possible, the access to its information must be enabled.

The *subscriber* then only needs the subject information of the other side or the subscribed publisher.

- Subject information

```
TF1_Sub   : FB_BA_SubAI := (  
          sSubject    := 'TF1'  
        );
```

Remote Subscriptions

The procedure for setting up a *subscription* to publishers on remote ADS devices is the same as for the local subscription.

Only the following information must be provided in addition:

- Target system

```
TF1_Sub      : FB_BA_SubAI := (
                sSubject      := 'TF1'
            );
```

Furthermore, it is possible to define the following (optional) parameters:

- Read interval
- Read tolerance
- Reset in case of read errors

Example: declaration with optional parameters:

```
TF1_Sub      : FB_BA_SubAI := (
                sSubject      := 'TF1',
                sTarget       := '192.168.10.100.1.1:852',
                tReadInterval := T#5S,
                tReadTolerance := T#20S,
                bClearOnReadError := TRUE
            );
```



No FB properties of object-related *subscribers* are supported.

These are exclusively for **local subscribers**.

ADS routes

Creating routes for ADS devices that communicate with each other is a separate step.

An integrator first implements the subscriptions and loads the control systems, although the communication (due to missing routes) **does not** yet work.

These routes can then be set up via the Terminal Explorer (see Device communication). After setting up the routes, all subscriptions will automatically (without any further steps) change from the fault state to the operating state.

Read errors

In the event of read errors (e.g. interruption of the physical connection), the affected *Subscriber* switches to the error state *Error*.



With each further failed reading attempt, the set read interval is temporarily increased by a factor of 1 (up to a maximum of 10) to conserve the network load and reduce error messages.

States

The state of a *subscriber* is indicated by the variable eState:

- *Init*
The *subscription* is initialized.

- *Ready*
The *subscriber* is ready (Initialization was completed successfully).
- *Error*
An error has occurred.

States that a subscriber can assume once a publisher is subscribed to on the local device:

- *Ready*
Data is read cyclically by the subscribed publisher in this state.

States that a subscriber can assume once a publisher is subscribed to on a remote device:

- *Init Subscription*
The subscribed publisher is queried on the remote ADS device.
- *Ready*
The subscriber is idle between individual read operations.

Suppressing

A read error has occurred which is suppressed for a set time.



No error state is displayed during *suppression*.

Error messages

At runtime, different error messages can be output in the log.

Use cases

Depending on the use case [typeless](#) [► 433] or [typed](#) [► 434] *publisher* and *subscriber* can be used.



[Typeless](#) [► 433] and [typed](#) [► 434] *publishers* / *subscribers* can be applied mixed.

For example, a derivative of FB_BA_Publisher can be used to share data, while an FB_BA_RawSubscriber is used to read this data.

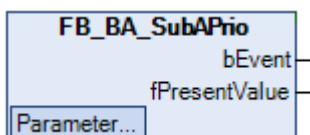
Object-related

Object-related subscribers are useful when data is to be read from standard objects.

For all object types there are corresponding prefabricated *subscribers*.

Thus, a subscriber type is compatible with one or more object types.

Example: subscriber that can be applied to all analog objects with priority array:



Typeless

Typeless *publishers* / *subscribers* are useful if data can be *released* or *subscribed* without any processing or preparation (e.g. program logic for formatting raw data to be sent or received).

Application

- **Publisher**

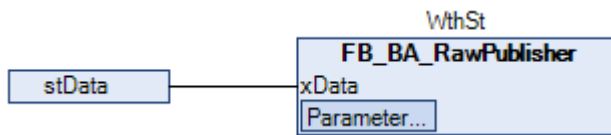
Declaration: data to be released and raw publisher:

```

2  VAR
3      stData      : ST_BA_MyWeatherData;
4      WthSt       : FB_BA_RawPublisher;
5  END_VAR

```

Implementation: assignment of data to be released to the publisher:



- **Suscriber**

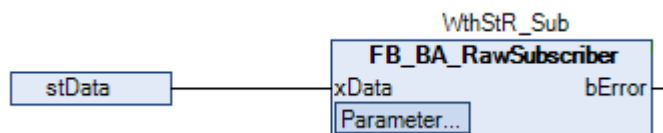
Declaration: data to be received and raw subscriber:

```

3      stData      : ST_BA_MyWeatherData;
4      WthSt_Sub   : FB_BA_RawSubscriber := (
5          sSubject                                     := 'WthSt'
6      );

```

Implementation: assignment of data to be received to the subscriber:



Typed

Typed *publishers* / *subscribers* are useful when data:

- should be processed in an object-oriented manner and encapsulated in an FB.
- are read out (e.g. with linked I/O terminals) before *release*.
- are prepared before the *release*, or after the *subscription* (e.g. by means of appropriate program logic).
- should be validated before *release*, or after *subscription*.

Implementation

The implementation of one's own publishers and subscribers is illustrated with the example of a weather station.

- **Publisher**
 - Derive from FB_BA_Publisher.
 - During initialization, call the inherited method *OnInit()*:

```

FB_BA_MyWeatherStation -> X
1 FUNCTION_BLOCK FB_BA_MyWeatherStation EXTENDS FB_BA_Publisher IMPLEMENTS I_BA_MyWeatherStation
2 VAR
3     {region 'Internal'}
4         eState      : (eInit:=1, eOperation:=2, eError:=3) UDINT := 1;
5         stData       : ST_BA_MyWeatherData;
6     {endregion}
7 END_VAR

1 CASE (eState) OF
2     eInit:
3         IF (OnInit()) THEN
4             eState := eOperation;
5         ELSE
6             eState := eError;
7         END_IF

```

- Override inherited method *GetData()* to make released data retrievable:

```

FB_BA_MyWeatherStation (FB)
  [Protected] Events
  Management
  Properties
  GetData
  Properties
  FB_init

FB_BA_MyWeatherStation.GetData -> X
1 // Returns published data.
2 METHOD GetData
3 VAR_OUTPUT
4     pData      : PVOID;           // Pointer to published data.
5     nSize      : DINT;           // Size of published data.
6 END_VAR

1 pData := ADR(stData);
2 nSize := SIZEOF(stData);

```

• Subscriber

- Derive FB_BA_Subscriber.
- Override inherited method *GetData()* to write subscribed data:

```

FB_BA_MyWeatherStationSub (FB)
  [Protected] Events
  Management
  GetData
  FB_init

FB_BA_MyWeatherStationSub.GetData -> X
1 // Returns subscription-data (For storing read data).
2 METHOD GetData
3 VAR_OUTPUT
4     pData      : PVOID;           // Pointer to subscribed data.
5     nSize      : DINT;           // Size of subscribed data.
6 END_VAR

1 pData := ADR(stData);
2 nSize := SIZEOF(stData);

```

Best practice

Site GVL

All devices on a site should use a GVL with *site-wide information*.

In this way, all devices can uniformly access the appropriately released information.

```

VAR_GLOBAL
    {region 'Devices'}
        Dev_Isp01      : T_BA_MedString := '192.168.10.101.1.1';
        Dev_Isp02      : T_BA_MedString := '192.168.10.102.1.1';
        Dev_Isp03      : T_BA_MedString := '127.168.0.3.1.1';
    {endregion}
    {region 'Subscriptions'}
        WthT_Subject    : T_BA_MedString := 'WthT';
        WthT_Target     : T_BA_MedString := NETID_B_ISP02;
        WthSt_Subject   : T_BA_MedString := 'WthSt';
        WthSt_Target    : T_BA_MedString := NETID_B_ISP02;
        Spt_Subject     : T_BA_MedString := 'Setp';
        Spt_Target      : T_BA_MedString := NETID_B_ISP01;
    {endregion}
END_VAR

```

Example: Site GVL with known devices and subscriptions.

- **Device 1:** provides general setpoints *Setp*.

- **Device 2:** provides data of the weather station *WthSt* (additionally the outdoor temperature *WthT*).

The subscription is independent of the device on which it is implemented.

This is because the correctness of the subject information is ensured in the *site GVL*, from which it is then extracted.

Sample: subscriber on any device that subscribes to weather data:

```
WthSt    : FB_BA_WeatherStationSub := (  
    sSubject      := SiteGVL.WthSt_Subject,  
    sTarget       := SiteGVL.WthSt_Target  
);
```



To ensure that all devices are using the same site GVL version, one of the following procedures is recommended:

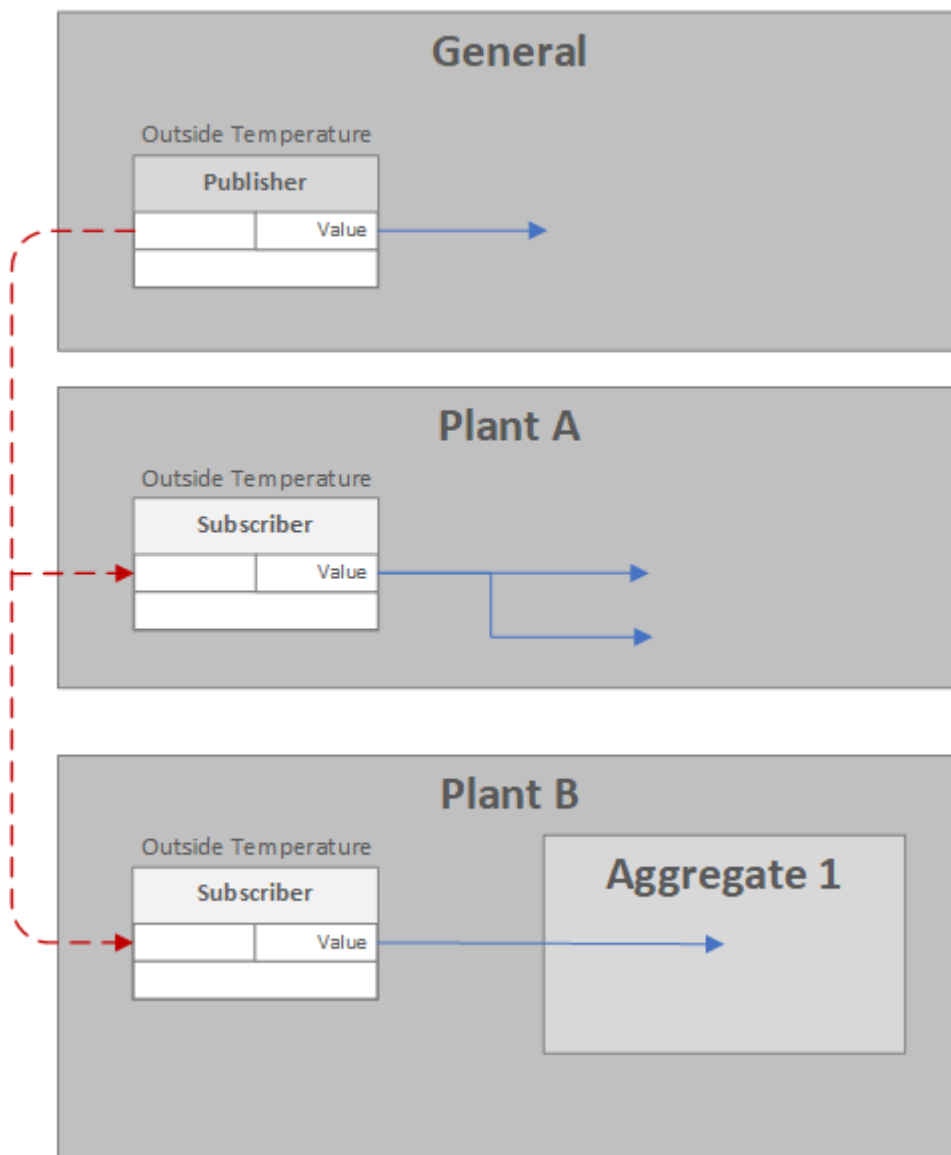
- **Project library**
Outsource GVL to TwinCAT library, which is then integrated in all devices.
 - **Template Repository**
Keep GVL up-to-date on all devices using [Template Repository](#) [► 1066].
-

Subscriptions

You should create as few subscribers as possible (e.g. for reasons of clarity or performance), but as many as necessary.

It therefore makes sense to declare subscribers only at plant level. Subscribed data can then be passed (e.g. using *VAR_IN_OUT*) to lower-level or nested program sections (such as aggregates).

Example: subscribing to the outdoor temperature within the (same) "Plant" level; passing on to lower program sections:



Example

Publisher and subscriber [► 76] (Also includes the code sample *weather station*)

- Local [► 78]
- Remote [► 81]

6.1.2.3 Archive

6.1.2.3.1 Tc2_BA

This library was imported from TwinCAT 2 because of its proven functionalities from the TS8040 supplement. Therefore, the documentation is identical, except for the function blocks concerning BACnet (see Tc_BA).

The BACnet specific function blocks can now be found in the library *Tc2_BACnetRev12*.

6.1.2.3.2 Tc3_BA



This library is used for the maintenance and servicing of existing projects. For new projects please use the library *Tc3_BA2*.

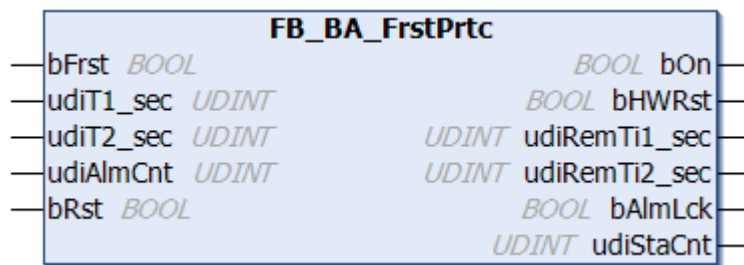
6.1.2.3.2.1 POU's

6.1.2.3.2.1.1 Air conditioning equipment

Function blocks

Name	Description
FB_BA_FrstPrtc [► 438]	Monitoring of frost alarm and emergency heating
FB_BA_HX [► 440]	Calculation of dew point temperature, specific enthalpy and absolute humidity
FB_BA_NgtCol [► 440]	Summer night cooling
FB_BA_RcvMonit [► 442]	Function block for calculating the efficiency of an energy recovery system
FB_BA_SPSupvis [► 444]	Function block for processing and checking the lower and upper setpoint of a supply air humidity or temperature control

6.1.2.3.2.1.1.1 FB_BA_FrstPrtc



The function block is used for frost monitoring of a heating coil in an air conditioning system.

A frost risk is present, if the input `bFrst` is TRUE. The frost alarm must be linked in the plant program such that the plant is switched off immediately, the heater valve opens, and the heater pump is switched on.

If there is risk of frost, the output `bOn` is set, and `udiT1_sec` (seconds) is started. If the frost risk remains (`bFrst`=TRUE) after `udiT1_sec` has elapsed, `bOn` remains set. It can only be reset at input `bRst`.

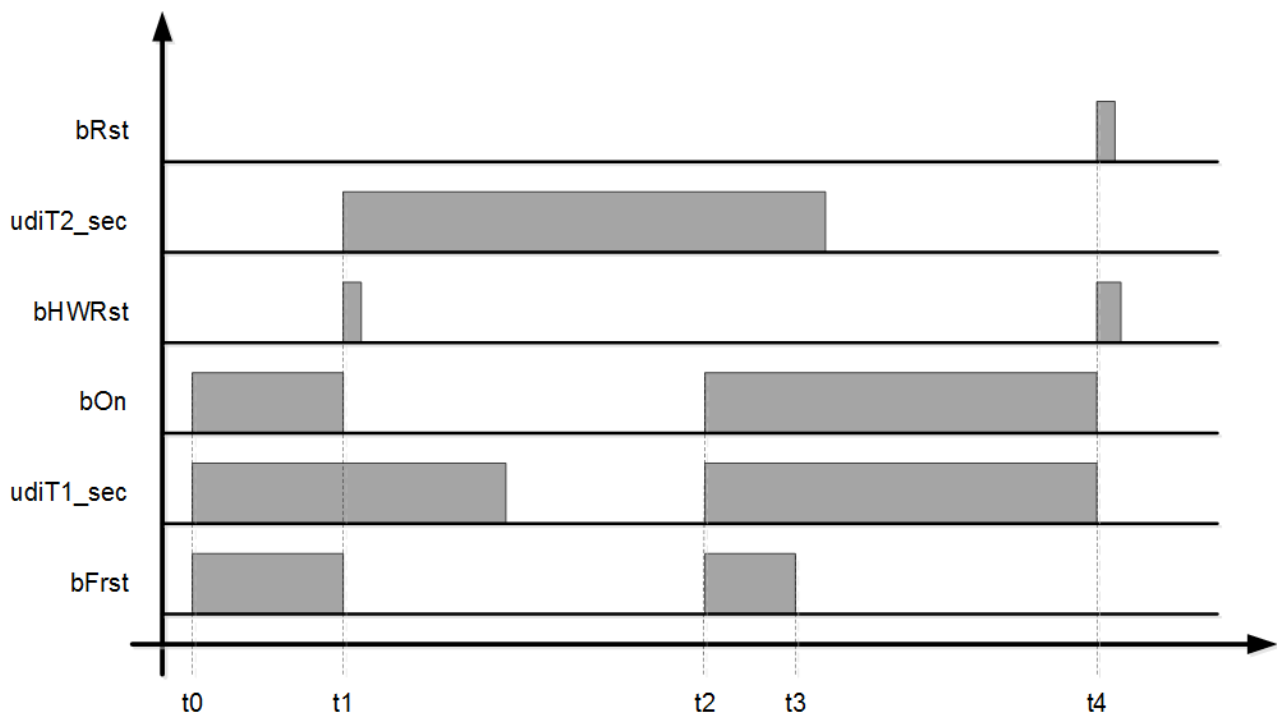
If the frost alarm ceases due to activation of the heating coil within the time `udiT1_sec` (`bFrst`=FALSE), the plant automatically restarts. For the plant restart `bOn` becomes FALSE, and at output `bHWRst` a pulse for acknowledgement of a latching circuit in the control cabinet is issued. With the restart a second monitoring period `udiT2_sec` (seconds) is initiated. If another frost alarm occurs within this period, the plant is permanently locked. `bOn` remains set until the frost alarm has been eliminated and `bRst` has been acknowledged.

In a scenario where frost alarms recur with time offsets that are greater than `udiT2_sec`, theoretically the plant would keep restarting automatically. In order to avoid this, the restarts within the function block are counted. The parameter `udiAlmCnt` can be used to set the number of possible automatic restart between 0 and 4.

An acknowledgement at input `bRst` resets the alarm memory within the function block to zero.

Sample:

`t0` = frost alarm at input `bFrst`, alarm message at output `bOn`, start of timer T1 (`udiT1_sec` [s])
`t1` = frost alarm off, resetting of `bOn`, output of hardware pulse, start of timer T2 (`udiT2_sec` [s]), plant restart
`t2` = further frost alarm within T2, alarm message at `bOn`, start of timer T1, locking of the frost alarm
`t3` = frost alarm off.
`t4` = acknowledgement of the alarm at `bRst`, resetting of `bOn`.



VAR_INPUT

```

bFrst      : BOOL;
udiT1_sec  : UDINT;
udiT2_sec  : UDINT;
udiAlmCnt  : UDINT;
bRst       : BOOL;

```

bFrst: Connection for frost events on the air and water side.

udiT1_sec: Timer for restart delays [s]. Internally limited to a minimum value of 0.

udiT2_sec: Timer monitoring time [s]. Internally limited to a minimum value of 0.

udiAlmCnt: Maximum number of automatic plant restarts without reset. Internally limited to values between 0 and 4.

bRst: Resetting and acknowledgement of the frost alarm.

VAR_OUTPUT

```

bOn         : BOOL;
bHWRst      : BOOL;
udiRemTi1_sec : UDINT;
udiRemTi2_sec : UDINT;
bAlmLck     : BOOL;
udiStaCnt   : UDINT;

```

bOn: Frost alarm active.

bHWRst: Output of a pulse for acknowledgement of the frost protection hardware.

udiRemTi1_sec: Time remaining to plant restart after frost alarm.

udiRemTi2_sec: Remaining monitoring time.

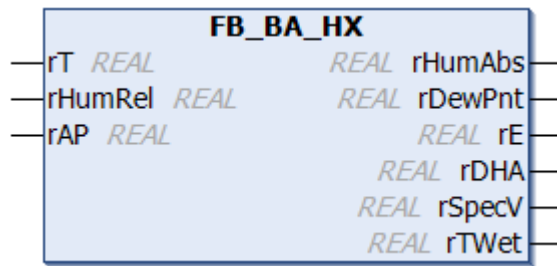
bAlmLck: Alarm lock - stored alarm.

udiStaCnt: Status counter – current number of unacknowledged false starts.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.1.2 FB_BA_HX



This function block is used to calculate the dew point temperature, the specific enthalpy and the absolute humidity. The temperature, the relative humidity and the barometric pressure are required for calculating these parameters.

The enthalpy is a measure for the energy of a thermodynamic system.

VAR_INPUT

```
rT      : REAL;
rHumRel : REAL;
rAP     : REAL;
```

rT: Temperature [°C].

rHumRel: Relative humidity [%].

rAP: Hydrostatic air pressure at 1013.25 hPa.

VAR_OUTPUT

```
lrHumAbs : LREAL;
lrDewPnt : LREAL;
lrE      : LREAL;
lrDHA    : LREAL;
lrSpecV  : LREAL;
lrTWet   : LREAL;
```

lrHumAbs: Absolute humidity g water per kg dry air [g/Kg].

lrDewPnt: Dew point temperature [°C].

lrE: Enthalpy [kJ/kg].

lrDHA: Density of moist air ρ [kg mixture/m³].

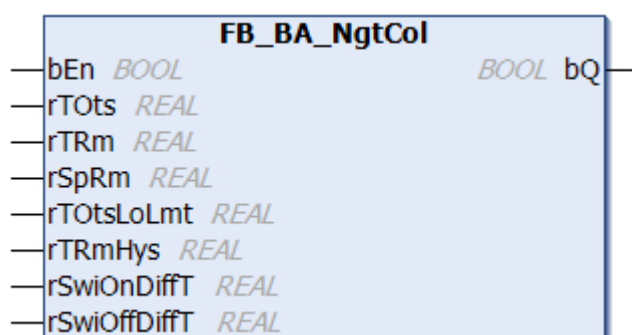
lrSpecV: Specific volume [m³/kg].

lrTWet: Wet bulb temperature [°C].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.1.3 FB_BA_NgtCol



With this function block, rooms that were heated up on the day before can be cooled down during the night using cool outside air. The summer night cooling function serves to improve the quality of the air and to save electrical energy. Electrical energy for cooling is saved during the first hours of the next summer day.

The start conditions for the summer night cooling are defined by parameterizing the *FB_BA_NgtCol* function block. The function block can be used to open motor-driven windows or to switch air conditioning systems to summer night cooling mode outside their normal hours of operation.

The following conditions must be met for activation of summer night cooling:

- The function block itself is enabled ($bEn=TRUE$).
- The outside temperature is not too low ($rTOts > rTOtsLoLmt$).
- The outside temperature is sufficiently low compared with the room temperature ($rTRm - rTOts > rSwiOnDiffT$).
- The room temperature is high enough to justify activating summer night cooling. $rTRm > rSpRm + rTRmHys$.

Under the following conditions the summer night cooling is disabled:

- The function block itself is disabled ($bEn = FALSE$).
- The outside temperature is too low ($rTOts < rTOtsLoLmt$).
- The outside temperature is too high compared with the room temperature ($rTRm - rTOts < rSwiOffDiffT$).
- The room temperature is lower than the setpoint. $rTRm \leq rSpRm$.

VAR_INPUT

```
bEn      : BOOL;
rTOts    : REAL;
rTRm     : REAL;
rSpRm    : REAL;
rTOtsLoLmt : REAL;
rTOtsHys : REAL;
rTRmHys  : REAL;
rSwiOnDiffT : REAL;
rSwiOffDiffT : REAL;
```

bEn: Enable function block.

rTOts: Outside temperature [°C].

rTRm: Outside temperature [°C].

rSpRm: Room temperature setpoint.

rTOtsLoLmt: Lower outside temperature limit [°C]; prevents excessive cooling.

rTOtsHys: Hysteresis for minimum outside temperature [K]. This hysteresis, which at the lower end is internally limited to 0.5 K, is intended to prevent jitter in *bQ*, if the outside temperature fluctuates precisely around the value of *rTOtsLoLmt*.

rTRmHys: Hysteresis for the room temperature [K]. This hysteresis, which at the lower end is internally limited to 0.5 K, is intended to prevent unnecessary fluctuation of *bQ*, if the room temperature fluctuates precisely around the setpoint *rSpRm*.

rSwiOnDiffT: Difference between the room temperature and the outside temperature, from which summer night cooling is enabled [K].

rSwiOffDiffT: Difference between the room temperature and the outside temperature, from which summer night cooling is locked [K].

VAR_OUTPUT

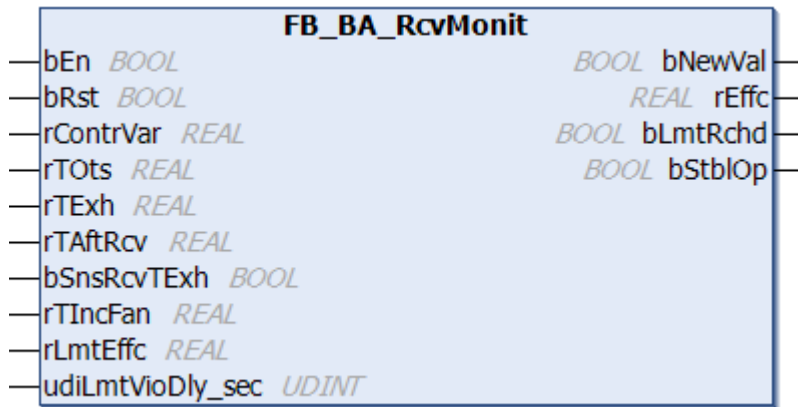
```
bQ : BOOL;
```

bQ: Summer night cooling on.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

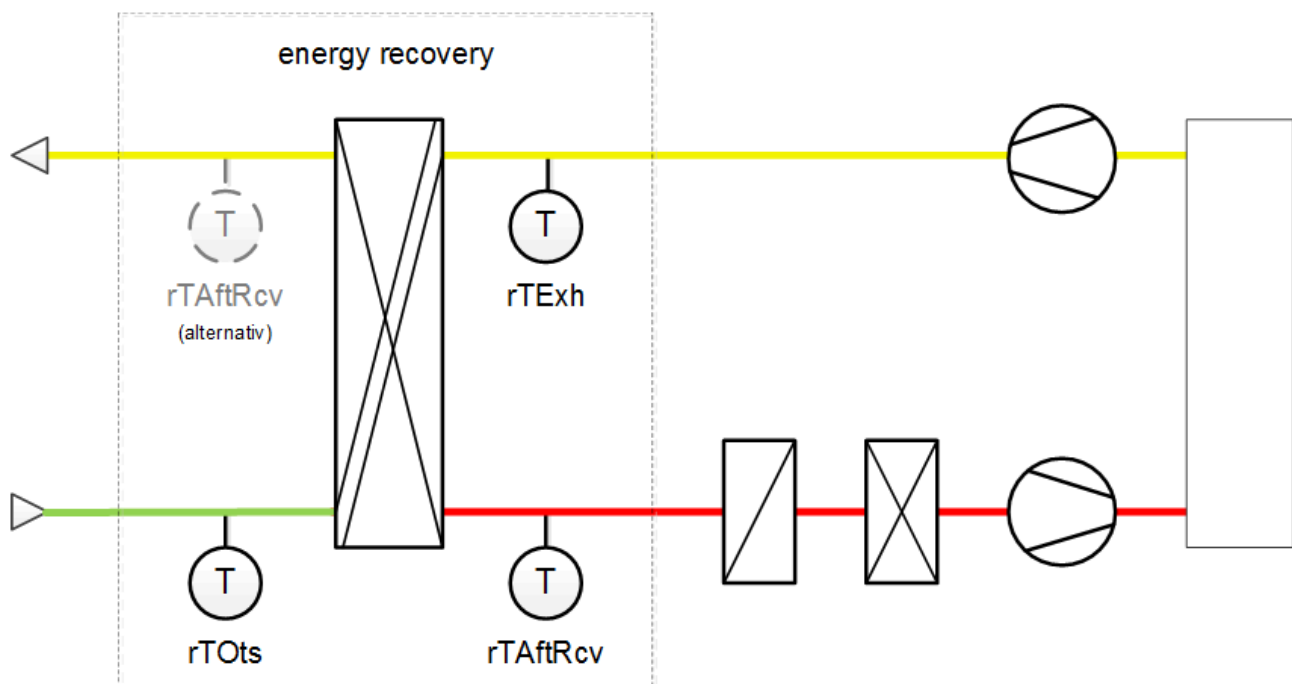
6.1.2.3.2.1.1.4 FB_BA_RcvMonit



The function block is used for calculating the efficiency of an energy recovery system.

The function block requires the following measured temperature values for calculating the efficiency (heat recovery rate):

- Outside air temperature *rTOts*
- Exhaust air temperature *rTExh*
- Air temperature of the energy recovery system in the inlet air duct (alternatively: in the outlet air duct) *rTAftRcv*



The function block logs the temperature values every 10 seconds and forms minutely averages from 6 consecutive values. The results are used to check whether the plant has reached a "stable" state.

- This is the case when the recorded temperatures of outside air, exhaust air and air after energy recovery are almost constant, i.e. none the 6 individual values deviate by more than 0.5 K from the respective mean value.
- The temperature difference between outside air and exhaust air is at least 5 K.

If this is the case, this measuring cycle is acknowledged with a TRUE signal at output *bStbOp*, and the calculated efficiency is output at *rEffc*. If the state is not "stable", a FALSE signal appears at output *bStbOp*, and *rEffc* is set to 0.

In any case, each measuring and analysis cycle is marked as completed with a trigger (a TRUE signal lasting one PLC cycle) at *bNewVal*.

Enable (*bEn*) and Reset (*bRst*)

The function block is only active if a TRUE signal is present at *bEn*. Otherwise its execution stops, and all outputs are set to FALSE or 0.0.

An active measuring and evaluation cycle can be terminated at any time by a TRUE signal at *bRst*. All outputs are set to FALSE or 0.0, and the measuring cycle restarts automatically.

Selection of the temperature value "after recovery" (*bSnsRcvTExh*)

A FALSE entry at *bSnsRcvTExh* means that the temperature measurement after the heat recovery in the **supply air duct** is used for calculating the efficiency.

To use the temperature measurement after the heat recovery in the **exhaust air duct**, TRUE must be applied at *bSnsRcvTExh*.

Limit violation (*rContrVar*, *rLmtEffc*, *bLmtRchd*)

A limit violation has occurred, if the calculated efficiency is less than the specified limit value *rLmtEffc*, and at the same time the control value for the heat recovery is at 100%. To this end the control value must be linked to input *rContrVar*.

The limit violation message can be delayed by an entry at *udiLmtVioDly_sec* [s]: If the two criteria, violation and override, are met for longer than *udiLmtVioDly_sec* [s], this is indicated with a TRUE signal at *bLmtRchd*.

A warning message, which may have occurred, is canceled if a complete measuring cycle provides "good" values, or with a rising edge at *bRst* or deactivation of the function block.



This warning message only occurs if the plant is in a stable operating mode (*bStbOp*=TRUE).

Taking into account the temperature increase of the outlet air due to the fan motor (*rTIncFan*)

It is possible that the outlet air is warmed by a fan motor, resulting in distortion of the measurement. This temperature increase can be specified through *rTIncFan*. Internally, the measured outlet air temperature is then reduced by this value.

VAR_INPUT

```

bEn          : BOOL;
bRst         : BOOL;
rContrVar    : REAL;
rTOts        : REAL;
rTExh        : REAL;
rTAftRcv     : REAL;
bSnsRcvTExh  : BOOL;
rTIncFan     : REAL;
rLmtEffc     : REAL;
udiLmtVioDly_sec : DINT;

```

bEn: Function block enable.

bRst: Reset - all determined values are deleted.

rContrVar: Control value for the heat recovery, i.e. the actual value.

rTOts: Outside temperature.

rTExh: Exhaust air temperature.

rTAftRcv: Temperature after energy recovery.

bSnsRcvTExh: Temperature at the measuring point after energy recovery: FALSE -> in inlet air duct (SupplyAir) - TRUE -> in outlet air duct (ExhaustAir).

rTIncFan: Temperature increase due to fan.

rLmtEffc: Limit value efficiency.

udiLmtVioDly_sec: Limit violation delay [s]. Internally limited to a minimum value of 0.

VAR_OUTPUT

```
bNewVal      : BOOL;
rEffc        : REAL;
bLmtRchd     : BOOL;
bStblOp      : BOOL;
```

bNewVal: Output trigger for new value *rEffc*.

rEffc: Efficiency

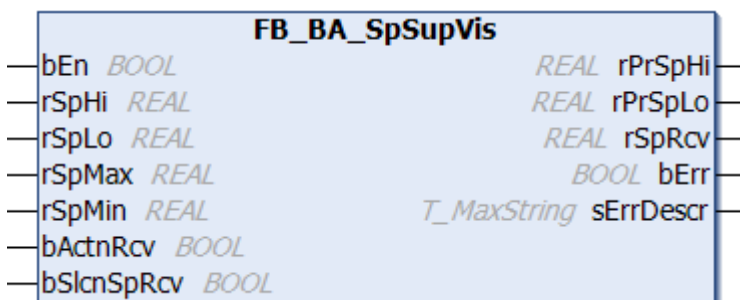
bLmtRchd: Limit value reached

bStblOp: Stable operation.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.1.5 FB_BA_SpSupvis



Function block for processing and checking the lower and upper setpoint of an inlet air humidity or temperature control.

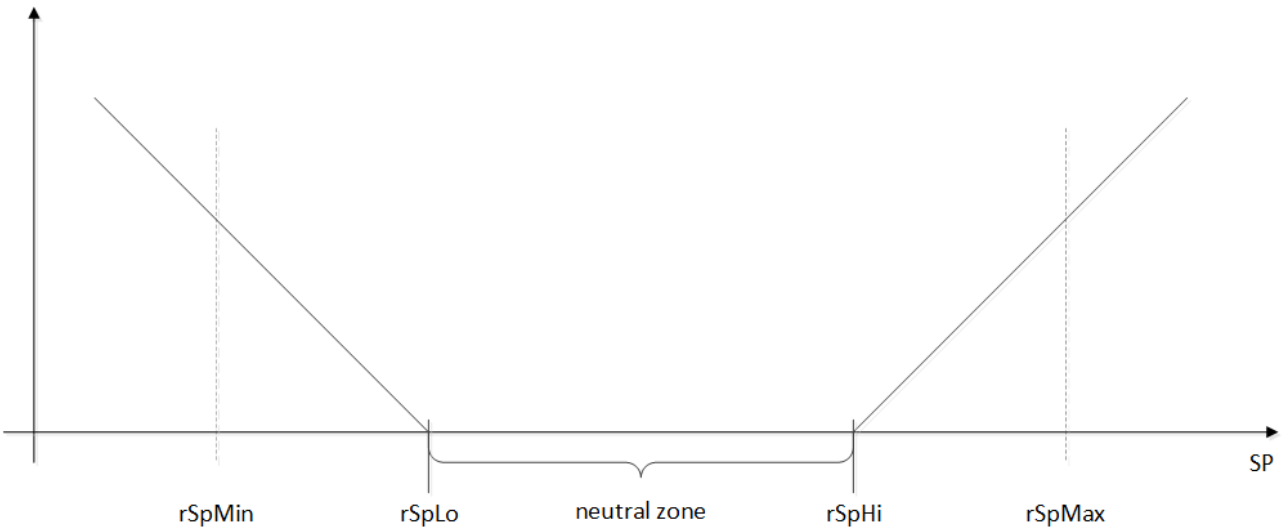
Checking and limitation of the setpoints

The function block limits the setpoints. The following two tables show which parameters are checked and what the response is in the event of an error.

Checking	Action
$rSpLo > rSpHi$	last valid values of <i>rSpLo</i> and <i>rSpHi</i> are used
$rSpMin \geq rSpMax$	last valid values of <i>rSpMin</i> and <i>rSpMax</i> are used
$rSpHi > rSpMax$	$rPrSpHi = rSpMax$
$rSpLo < rSpMin$	$rPrSpLo = rSpMin$

Checking	bErr	Action
$rSpMin \geq rSpMax$	TRUE	$rSpErr = ((rSpMin + rSpMax) / 2)$
$rSpHi < rSpMin$		$rPrSpHi = rPrSpLo = rPrRcv = rSpErr$
$rSpLo > rSpMax$		

The difference between the setpoints describes an energy-neutral zone. With inlet air control, no heating or cooling would take place within the neutral zone.



The checked and, if necessary, limited setpoints are output at the function block output as *rPrSpHi* and *rPrSpLo* (Present Setpoint).

Setpoint for heat recovery

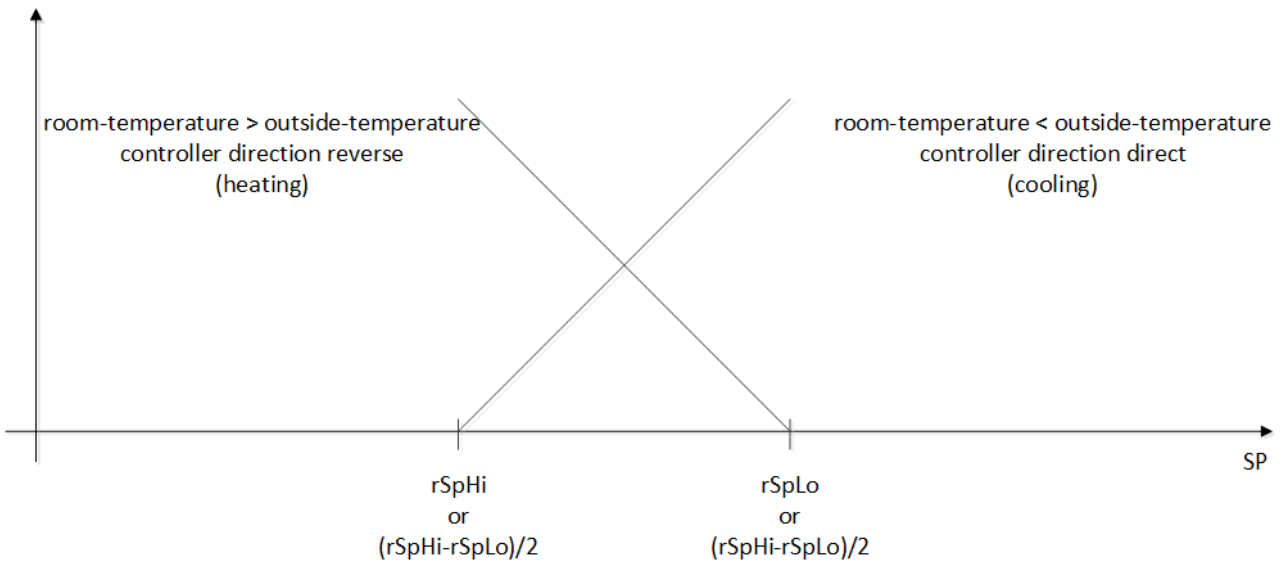
For heat recovery, the setpoint *rSpRcv* is optionally calculated from the mean value of the upper and lower setpoint, *rSpHi* and *rSpLo*, or depending on the control direction of the heat recovery system. The method is defined through the input variable *bSlcnSpRcv*:

b SlcnSpRcv	rSpRcv
TRUE	Mean value of <i>rSpLo</i> and <i>rSpHi</i>
FALSE	Depends on direction of action, defined through input <i>bActRcv</i>

If the setpoint is defined depending on the direction of action, the following applies:

bActRcv	Control direction	rSpRcv
TRUE	direct (cooling)	<i>rSpHi</i>
FALSE	indirect (heating)	<i>rSpLo</i>

Heat recovery



VAR_INPUT

```

bEn      : BOOL;
rSpHi    : REAL;
rSpLo    : REAL;
rSpMax   : REAL;
rSpMin   : REAL;
bActnRcv : BOOL;
bSlcnSpRcv : BOOL;

```

bEn: function block enable. If *bEn* = FALSE, all output parameters are 0.0.

rSpHi: Upper setpoint input value to be checked.

rSpLo: Lower setpoint input value to be checked.

rSpMax: Maximum setpoint.

rSpMin: Minimum setpoint.

bActnRcv: Direction of action of the downstream heat recovery.

bSlcnSpRcv: Setpoint selection of the downstream heat recovery system.

VAR_OUTPUT

```

rPrSpHi  : REAL;
rPrSpLo  : REAL;
rSpRcv   : REAL;
bErr     : BOOL;
sErrDescr : T_MAXSTRING;

```

rPrSpHi: Output value for the upper setpoint.

rPrSpLo: Output value for the lower setpoint.

rSpRcv: Output value for the resulting heat recovery setpoint.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description

01: Warning: The setpoints are not in a logical order: Either (*rSpMin* >= *rSpMax*) OR (*rSpHi* < *rSpMin*) OR (*rSpLo* > *rSpMax*)

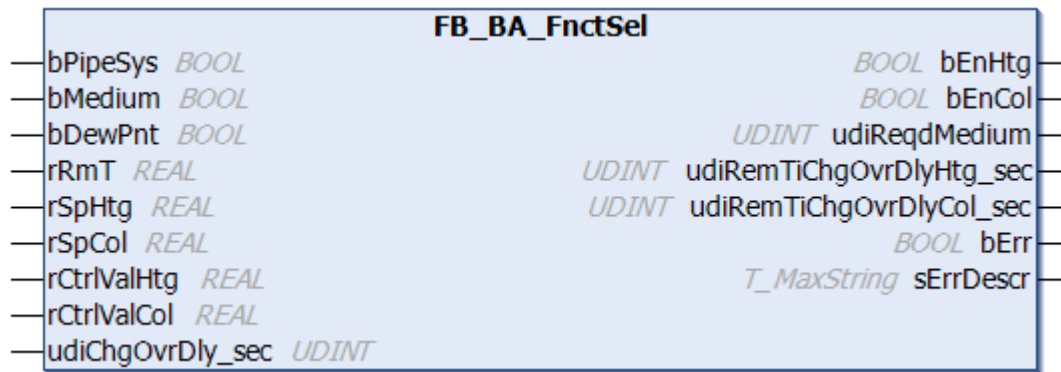
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2 Room automation**6.1.2.3.2.1.2.1 Heating, cooling****Function blocks**

Name	Description
FB_BA_FnctSel [▶ 447]	Function selection (heating and/or cooling) in two- or four-pipe network.
FB_BA_RmTAdj [▶ 450]	Adjustment of the room temperature setpoint.
FB_BA_SpRmT [▶ 453]	Adjustment of the room temperature setpoint

6.1.2.3.2.1.2.1.1 FB_BA_FnctSel



The function block is used for enabling heating or cooling mode in a room.

The distribution network type plays a significant role:

In a two-pipe system, all rooms served by the plant can either be heated or cooled at the same time.

In a four-pipe system, the room conditioning can be demand-based, i.e. some rooms can be heated, while other rooms can be cooled by the same plant.

The function block used for each room, as already mentioned, selects its controllers, depending on which type of piping system is available:

Two-pipe network

The two-pipe system is selected if the function block has a FALSE entry at input *bPipeSys*. Since all rooms served by the plant can only either be heated or cooled, the choice is specified centrally for all rooms via the input *bMedium*. If *bMedium* is FALSE, the room heating controller is selected. If the input is TRUE the cooling controller is selected. The controller enable states *bEnHtg* and *bEnCol* are always issued with a delay of *udiChgOvrDly_sec* [s]. In other words, heating cannot be enabled until the cooling enable state *bEnCol* for *udiChgOvrDly_sec* is FALSE, and vice versa. In addition to the elapsing of this changeover time, the system checks that the output from controller to be switched off is 0.0. This is based on feedback at the inputs *rCtrlValHtg* and *rCtrlValCol*. In this way, a drastic change from heating to cooling and vice versa is avoided.

Four-pipe network

The four-pipe system is selected if the function block has a TRUE entry at input *bPipeSys*. In this case, the choice of controller can be different for the individual rooms as required, based on the room temperature *rRmT* and the setpoints *rSpHtg* for heating and *rSpCol* for cooling. If the room temperature exceeds the cooling setpoint, the cooling controller is activated (*bEnCol*), if it falls below the heating setpoint, the heating controller is activated (*bEnHtg*). If the temperature is between the two setpoints, both controllers are switched off (energy-neutral zone). Here too, the output of the controller enable states *bEnHtg* and *bEnCol* is delayed by *udiChgOvrDly_sec* [s] (see two-pipe network). In addition to the elapsing of this changeover time, the system checks that the output from controller to be switched off is 0.0. This is based on feedback at the inputs *rCtrlValHtg* and *rCtrlValCol*. In this way, a drastic change from heating to cooling and vice versa is avoided, if the changeover time is inadequate.

Dew-point monitor (*bDewPnt*)

In both systems (two- and four-pipe) the dew-point monitor has the task of deactivating cooling immediately, if required.

Program sequence

The function block can have 3 possible states:

1. Waiting for heating or cooling enable
2. Heating enable
3. Cooling enable

In the first step, the function block waits for compliance with the conditions required for heating or cooling:

Heating	Cooling
Cooling controller output = 0 (<i>rCtrlValCol</i>)	Heating controller output = 0 (<i>rCtrlValHtg</i>)
Room temperature (<i>rRmT</i>) < heating setpoint (<i>rSpHtg</i>)	Room temperature (<i>rRmT</i>) > cooling setpoint (<i>rSpCol</i>)
Cooling controller enable (<i>bEnCol</i>) is FALSE over at least the changeover time <i>udiChgOvrDly_sec</i> [s]	Heating controller enable (<i>bEnHtg</i>) is FALSE over at least the changeover time <i>udiChgOvrDly_sec</i> [s]
Four-pipe system is selected (<i>bPipeSys</i> =TRUE) OR two-pipe system is selected and heating medium is available (<i>bPipeSys</i> =FALSE AND <i>bMedium</i> =FALSE)	Four-pipe system is selected (<i>bPipeSys</i> =TRUE) OR two-pipe system is selected and cooling medium is available (<i>bPipeSys</i> =FALSE AND <i>bMedium</i> =TRUE)
	The dew-point monitor does not respond (<i>bDewPnt</i> =TRUE)

If a chain of conditions is met, the function block switches to the respective state (heating or cooling) and remains in this state until the corresponding controller issues 0 at the function block input (*rCtrlValHtg*/*rCtrlValCol*). This ensures that only one controller is active at any one time, even if a high heating controller output, for example, would call for a brief cooling intervention (overshoot). Heating or cooling continues until there is no longer a demand.

There are 3 exceptions, for which heating or cooling is immediately interrupted:

1. A two-pipe system (*bPipeSys*=FALSE) is in heating mode (*bEnHtg*), but a switch to cooling medium occurred *bMedium*=TRUE
2. A two-pipe system (*bPipeSys*=FALSE) is in cooling mode (*bEnCol*), but a switch to heating medium occurred *bMedium*=FALSE
3. The dew-point monitor was triggered (*bDewPnt*=TRUE) in cooling mode (two- or four-pipe system)

In these cases the heating or cooling enable states are canceled, and the plant switches to standby.

Demand message (*udiReqdMedium*)

To notify the plant of the current demand for heating or cooling, a demand ID is issued at the function block output, i.e. for each room, depending on the actual and set temperature. These can be collected and evaluated centrally. The evaluation always takes place, irrespective of the network type (two- or four-pipe).

<i>udiReqdMedium</i>	Medium	Room temperature
1	No medium is requested	$rRmT > rSpHtg$ AND $rRmT < rSpCol$
2	Heating medium is requested	$rRmT < rSpHtg$
3	Cooling medium is requested	$rRmT > rSpCol$

Error handling

The heating setpoint must not be greater than or equal to the cooling setpoint, since this would result in temperature range with simultaneous heating and cooling demand. However, since the function block only issues one enable state at a time (i.e. heating or cooling), the case is harmless from a plant engineering perspective. In this case only a warning message is issued (*bErr*=TRUE, *sErrDescr*=warning message); the function block does not interrupt its cycle.

VAR_INPUT

```

bPipeSys      : BOOL;
bMedium       : BOOL;
bDewPnt       : BOOL;
rRmT          : REAL;
rSpHtg        : REAL;
rSpCol        : REAL;
rCtrlValHtg   : REAL;
rCtrlValCol   : REAL;
udiChgOvrDel_sec : UDINT;

```

bPipeSys: In two-pipe system *bPipeSys* is FALSE, in four-pipe systems it is TRUE.

bMedium: Current supply of the whole two-pipe network with cooling or heating medium. If heating medium is active, *bMedium* is FALSE.

bDewPnt: Dew-point monitor: If *bDewPnt* = FALSE, the cooling controller is locked.

rTRm: Room temperature.

rSpHtg: Heating setpoint.

rSpCol: Cooling setpoint.

rCtrlValHtg: Current output value of the heating controller. Used internally as switching criterion from heating to cooling: *rCtrlValHtg* must be 0.

rCtrlValCol: Current output value of the cooling controller. Used internally as switching criterion from cooling to heating: *rCtrlValCol* must be 0.

udiChgOvrDel_sec: Switchover delay [s] from heating to cooling or vice versa. Internally limited to a minimum value of 0.

VAR_OUTPUT

```

bEnHtg           : BOOL;
bEnCol           : BOOL;
udiReqdMedium    : UDINT;
udiRemTiChgOvrDlyHtg_sec : UDINT;
udiRemTiChgOvrDlyCol_sec : UDINT;
bErr             : BOOL;
sErrDescr        : T_MAXSTRING;

```

bEnHtg: Heating controller enable.

bEnCol: Cooling controller enable.

udiReqdMedium:

udiReqdMedium	Medium	Room temperature
1	No medium is requested	$rRmT > rSpHtg$ AND $rRmT < rSpCol$
2	Heating medium is requested	$rRmT < rSpHtg$
3	Cooling medium is requested	$rRmT > rSpCol$

udiRemTiChgOvrDlyHtg_sec: Countdown [s] for switchover delay from cooling to heating.

udiRemTiChgOvrDlyCol_sec: Countdown [s] for switchover delay from heating to cooling.

bErr: In case of a fault, e.g. if warning stages are active, this output is set to TRUE.

sErrDescr: Contains the error description.

Error description
01: Warning: The heating setpoint is higher than or equal to the cooling setpoint

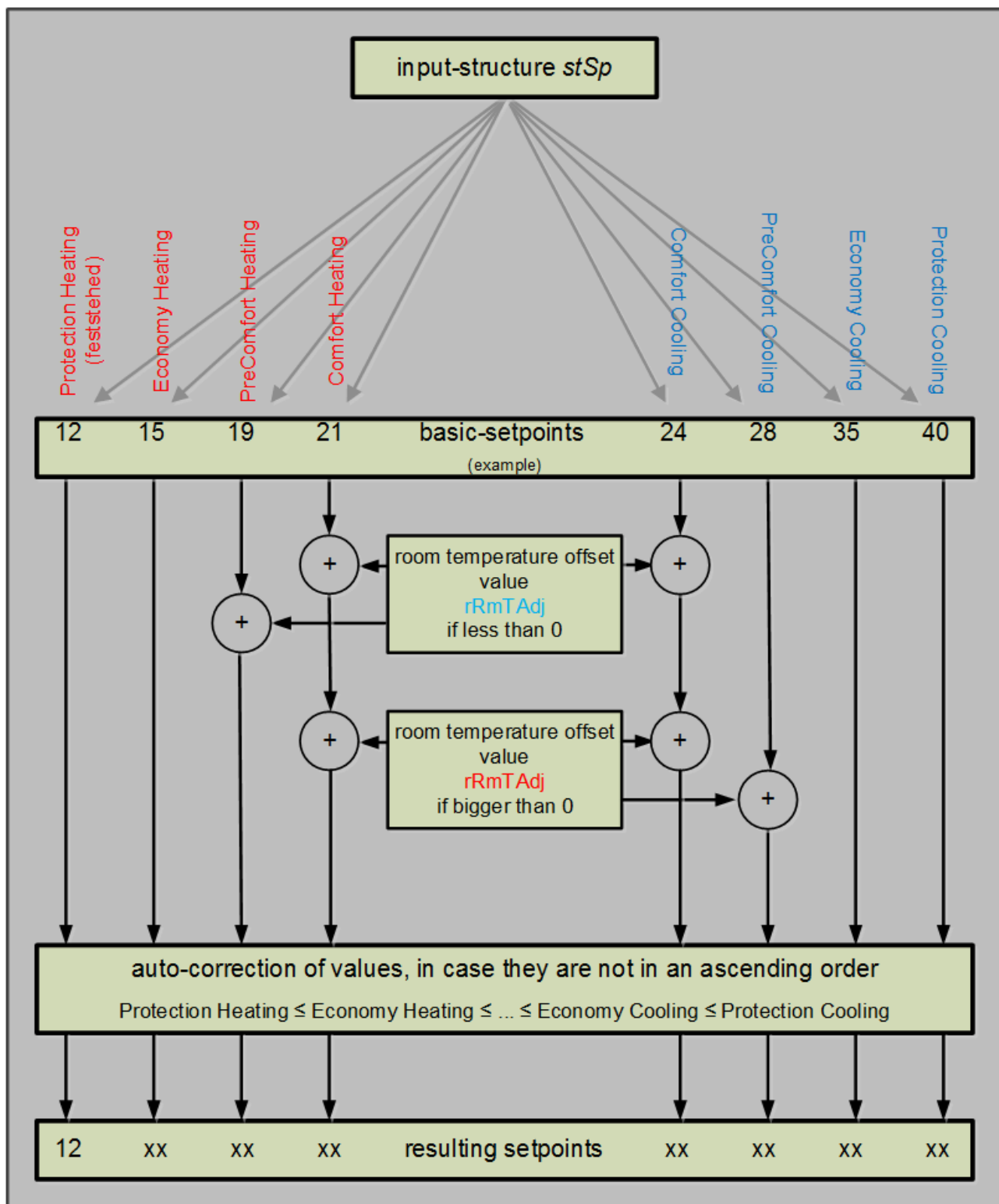
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.1.2 FB_BA_RmTAdj



The function block *FB_BA_RmTAdj* is used for user adjustment of the room temperature setpoint. It shifts the setpoints at the input of a function block depending on an offset *rRmTAdj*, as shown in the following diagram. At the *rRmTAdj* input, the value of a resistance potentiometer or a bus-capable field device can be used for the setpoint correction.



If the set value *rRmTAdj* is greater than zero, room heating is requested: The Comfort Heating value is increased by *rRmTAdj*. At the same time, the values for Comfort Cooling and PreComfort Cooling are increased. If the value *rRmTAdj* is less than zero, a lower room temperature is requested. Analog to the heating case, the values for Comfort Cooling, Comfort Heating and PreComfort Heating are now reduced by the value *rRmTAdj*.

Auto-correction

The temperature adjustment is intended for small corrections of the values. Although it is possible to enter any input values, a heating system will only work in a meaningful manner if the setpoints have ascending values in the following order:

- Protection Heating
- Economy Heating
- Precomfort Heating
- Comfort Heating
- Comfort Cooling
- Precomfort Cooling
- Economy Cooling
- Protection Cooling

Auto-correction works according to the following principle: Starting with the value Economy Heating, the system checks whether this value is smaller than the lower value of Protection Heating. If this is the case, the value for Economy Heating is adjusted to match the value for Protection Heating. The system then checks whether the value for Precomfort Heating is less than Economy Heating and so on, until the value for Protection Cooling is compared with the value for Economy Cooling. If one or several values were corrected, this is indicated with a TRUE signal at output *bValCorr*.

VAR_INPUT

```
rRmTAdj : REAL;
stSp    : ST_BA_SpRmT;
```

rRmTAdj: Room temperature offset value.

stSp: Input structure for the setpoints (see [ST_BA_SpRmT \[► 601\]](#)).

VAR_OUTPUT

```
bValCorr : BOOL;
rPrPrtcHtg : REAL;
rPrEcoHtg : REAL;
rPrPreCmfHtg : REAL;
rPrCmfHtg : REAL;
rPrPrtcCol : REAL;
rPrEcoCol : REAL;
rPrPreCmfCol : REAL;
rPrCmfCol : REAL;
stPrSp : ST_BA_SpRmT;
```

bValCorr: Autocorrection for the values was performed, see above.

rPrPrtcHtg: Resulting Protection Heating setpoint.

rPrEcoHtg: Resulting Economy Heating setpoint.

rPrPreCmfHtg: Resulting PreComfort Heating setpoint.

rPrCmfHtg: Resulting Comfort Heating setpoint.

rPrCmfCol: Resulting Comfort Cooling setpoint.

rPrPreCmfCol: Resulting PreComfort Cooling setpoint.

rPrEcoCol: Resulting Economy Cooling setpoint.

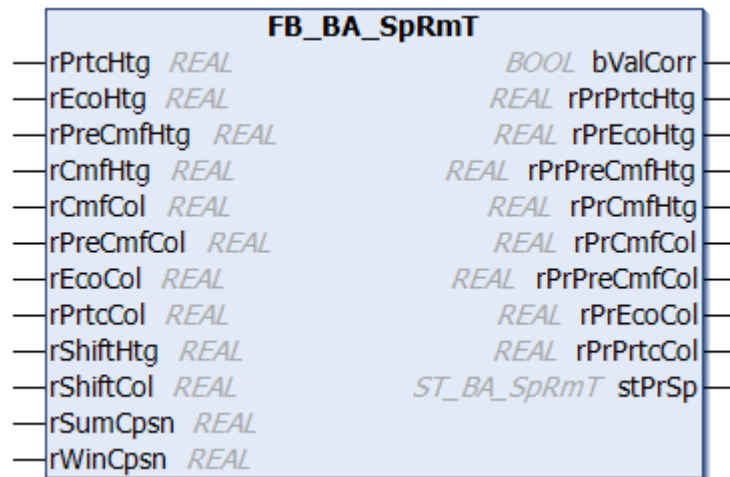
rPrPrtcCol: Resulting Protection Cooling setpoint.

stPrSp: Consolidated output of the resulting values in a structure (see [ST_BA_SpRmT \[► 601\]](#)).

Requirements

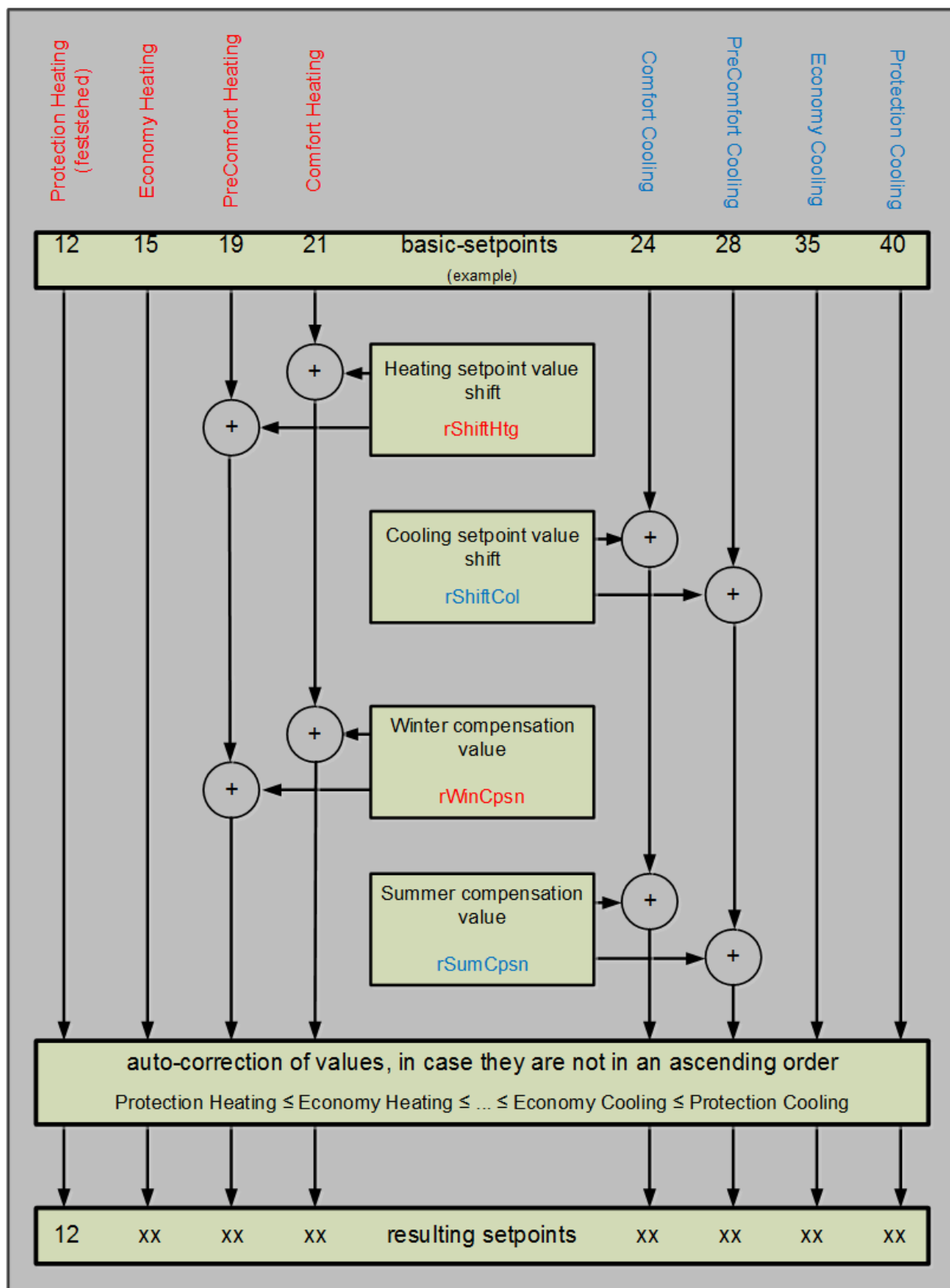
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.1.3 FB_BA_SpRmT



The function block *FB_BA_SpRmT* assigns setpoints for cooling and heating operation to each of the energy levels Protection, Economy, PreComfort and Comfort.

The following graphics illustrates the behavior of the function block; the entered values should be regarded as examples:



The parameter *rShiftHtg* is applied to the Comfort and Precomfort values for the heating mode as central setpoint shift. In addition, winter compensation *rWinCpsn* is applied. Similarly, the following applies for the cooling mode: The parameter *rShiftCol* is applied to the Comfort and Precomfort values. In addition, the summer compensation value *rSumCpsn* is applied.

Auto-correction

The setpoint shift is intended for small corrections of the values. Although it is possible to enter any input values, a heating system will only work in a meaningful manner if the setpoints have ascending values in the following order:

- Protection Heating
- Economy Heating
- Precomfort Heating
- Comfort Heating
- Comfort Cooling
- Precomfort Cooling
- Economy Cooling
- Protection Cooling

Auto-correction works according to the following principle: Starting with the value Economy Heating, the system checks whether this value is smaller than the lower value of Protection Heating. If this is the case, the value for Economy Heating is adjusted to match the value for Protection Heating. The system then checks whether the value for Precomfort Heating is less than Economy Heating and so on, until the value for Protection Cooling is compared with the value for Economy Cooling. If one or several values were corrected, this is indicated with a TRUE signal at output *bValCorr*.

VAR_INPUT

```
rSumCpsn    : REAL;
rWrWinCpsn  : REAL;
```

rSumCpsn: Summer compensation value

rWinCpsn: Winter compensation value

VAR_OUTPUT

```
bValCorr      : BOOL;
rPrPrtcHtg    : REAL;
rPrEcoHtg     : REAL;
rPrPreCmfHtg  : REAL;
rPrCmfHtg     : REAL;
rPrCmfCol     : REAL;
rPrPreCmfCol  : REAL;
rPrEcoCol     : REAL;
rPrPrtcCol    : REAL;
stPrSp        : ST_BA_SpRmT;
```

bValCorr: Autocorrection: At least one of the resulting setpoints was adjusted such that the values continue to monotonically increase.

rPrPrtcHtg: Resulting Protection Heating setpoint.

rPrEcoHtg: Resulting Economy Heating setpoint.

rPrPreCmfHtg: Resulting PreComfort Heating setpoint.

rPrCmfHtg: Resulting Comfort Heating setpoint.

rPrCmfCol: Resulting Comfort Cooling setpoint.

rPrPreCmfCol: Resulting PreComfort Cooling setpoint.

rPrEcoCol: Resulting Economy Cooling setpoint.

rPrPrtcCol: Resulting Protection Cooling setpoint.

stPrSp: Consolidated output of the resulting values in a structure (see [ST_BA_SpRmT](#) [► 601]).

VAR_INPUT_CONSTANT_PERSISTENT (Parameter)

```

rShiftCol  : REAL := 0;
rShiftHtg  : REAL := 0;
rPrtcCol   : REAL := 35;
rEcoCol    : REAL := 28;
rPreCmfCol : REAL := 25;
rCmfCol    : REAL := 23;
rCmfHtg    : REAL := 21;
rPreCmfHtg : REAL := 18;
rEcoHtg    : REAL := 14;
rPrtcHtg   : REAL := 6;

```

rShiftCol: Cooling setpoint value shift.

rShiftHtg: Heating setpoint value shift.

rPrtcCol: Basic Protection Cooling setpoint.

rEcoCol: Basic Economy Cooling setpoint.

rPreCmfCol: Basic PreComfort Cooling setpoint.

rCmfCol: Basic Comfort Cooling setpoint.

rCmfHtg: Basic Comfort Heating setpoint.

rPreCmfHtg: Basic PreComfort Heating setpoint.

rEcoHtg: Basic Economy Heating setpoint.

rPrtcHtg: Basic Protection Heating setpoint.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.2 Lighting**Function blocks**

Name	Description
FB_BA_Toggle [► 456]	Switching of lamps.

6.1.2.3.2.1.2.2.1 FB_BA_Toggle

The function block is used to switch an actuator on or off.

The input *bEn* is used for enabling the function block.

A positive edge at the input *bOn* results in setting of output *bQ*. The output is reset by a rising edge at the *bOff* input. If a rising edge is presented to *bToggle*, the output is negated; i.e., if On it goes Off, and if Off it goes On.

VAR_INPUT

```

bEn      : BOOL;
bOn      : BOOL;
bOff     : BOOL;
bToggle  : BOOL;

```

- bEn:** Function block enable.
- bOn:** Switches the output on.
- bOff:** Switches the output off.
- bToggle:** Negates the current output state.

VAR_OUTPUT

```
bQ      : BOOL;
```

bQ: Control output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

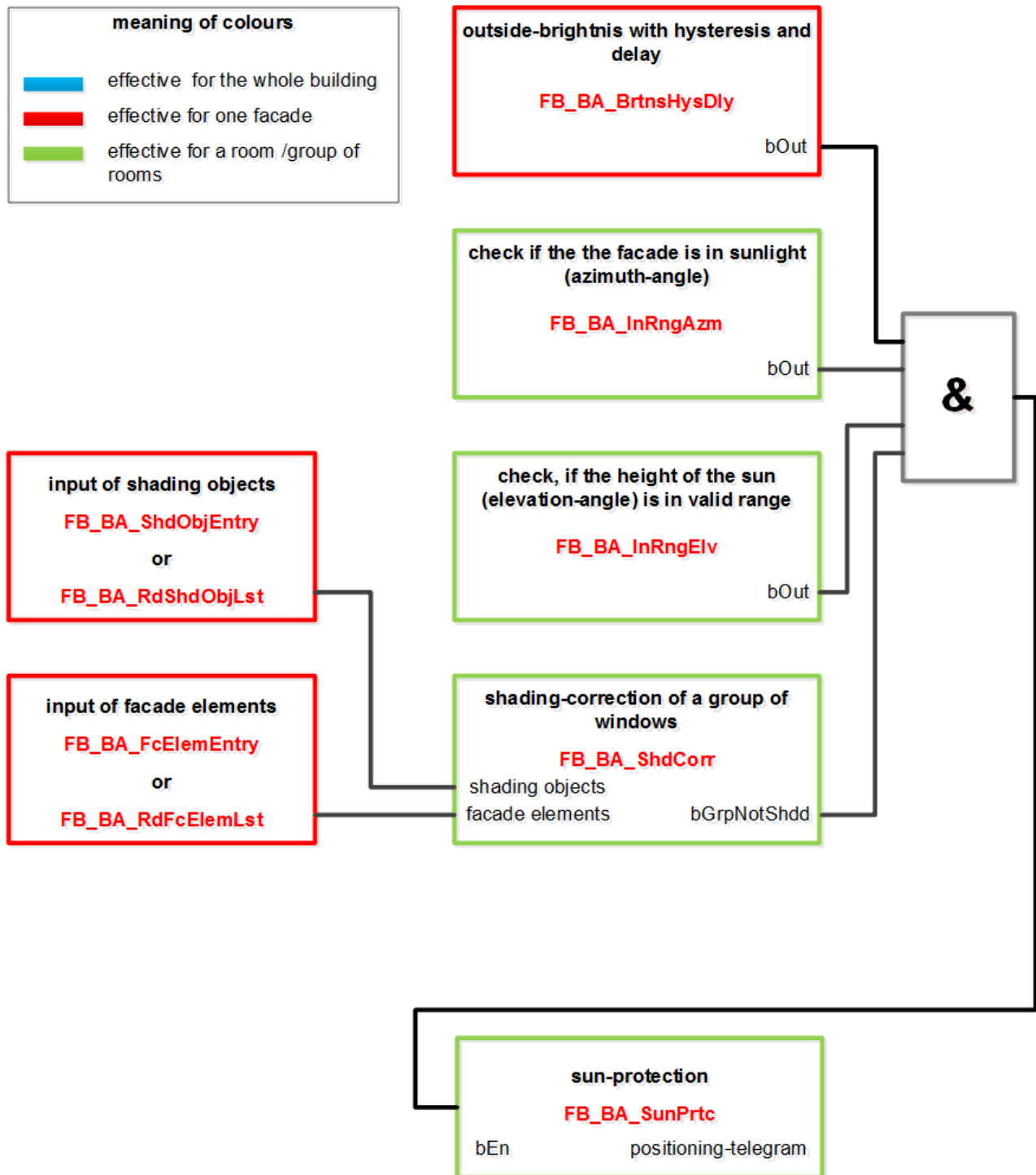
6.1.2.3.2.1.2.3 Shading

- [Overview of shading correction \[► 459\]](#)
- [Shading correction: Basic principles and definitions \[► 459\]](#)
- [Overview of automatic sun protection \[► 467\]](#)
- [Sun protection: Basic principles and definitions \[► 469\]](#)
- [List of shading elements \[► 474\]](#)
- [List of facade elements \[► 474\]](#)

Function blocks

Name	Description
FB_BA_BldPosEntry [► 474]	Sun protection function: Input of blind positions.
FB_BA_CalcSunPos [► 476]	Calculation of sun position
FB_BA_FcdElemEntry [► 478]	Shading correction: Input of facade elements per function block.
FB_BA_InRngAzlm [► 483]	Verification of valid sun position and sun direction range (azimuth angle)
FB_BA_InRngElv [► 485]	Verification of valid sun position and sun elevation range (elevation angle)
FB_BA_RdFcdElemLst [► 488]	Shading correction: Input of facade elements via data list (csv).
FB_BA_RdShdObjLst [► 492]	Shading correction: Input of shading objects via data list (csv).
FB_BA_RolBldActr [► 496]	Roller shutter actuator
FB_BA_ShdCorr [► 498]	Shading correction function block
FB_BA_ShdObjEntry [► 501]	Shading correction: Input of shading objects per function block.
FB_BA_SunBldActr [► 504]	Blind actuator
FB_BA_SunBldEvt [► 509]	Output of a specified blind position and angle in percent
FB_BA_SunBldIcePrtc [► 510]	Anti-icing
FB_BA_SunBldPosDly [► 511]	Switch-on delay for blinds/groups of blinds
FB_BA_SunBldPrioSwi4 [► 512]	Priority control, 4 inputs
FB_BA_SunBldPrioSwi8 [► 513]	Priority control, 8 inputs
FB_BA_SunBldScn [► 514]	Manual operation with scene selection and programming
FB_BA_SunBldSwi [► 517]	Manual operation
FB_BA_SunBldTwilgtAuto [► 519]	Automatic twilight function
FB_BA_SunBldWndPrtc [► 520]	Protection against wind damage
FB_BA_SunPrtc [► 522]	Sun protection function, see Overview of automatic sun protection (shading correction)

6.1.2.3.2.1.2.3.1 Overview of shading correction



6.1.2.3.2.1.2.3.2 Shading correction: Basic principles and definitions

The shading correction can be used in conjunction with the automatic sun function or louvre adjustment. The function checks whether a window or a window group that is assigned to a room, for example, is temporarily placed in the shade by surrounding buildings or parts of its own building. Sun shading for windows that stand in the shadow of surrounding buildings or trees is not necessary and may even be disturbing under certain circumstances. On the basis of data entered regarding the facade and its surroundings, the shading correction determines which parts of the front are in the shade. Hence, it is then possible to decide whether the sun protection should be active for individual windows or window groups.

Apart from the current position of the sun, the shading of the individual windows depends on three things:

- the orientation of the facade

- the position of the windows
- the positioning of the shading objects

The following illustrations are intended to describe these interrelationships and to present the parameters to be entered.

Orientation of the facade

Observation from above

For the pure observation of the shadow thrown on the facade, a two-dimensional coordinate system is ultimately required, therefore the X and Y axis were placed on the facade. The zero point is thereby at the bottom left on the base, as if one were regarding the facade from the front. For the calculation of the shading objects the Z component is then also added. Its axis points from away the facade and has the same zero point as the X and Y axis.

In the northern hemisphere, the horizontal sun position (azimuth angle) is determined from the north direction by definition. The facade orientation is likewise related to north, wherein the following applies to the line of sight from a window in the facade:

Line of sight	Facade orientation
North	$\beta=0^\circ$
East	$\beta=90^\circ$
South	$\beta=180^\circ$
West	$\beta=270^\circ$

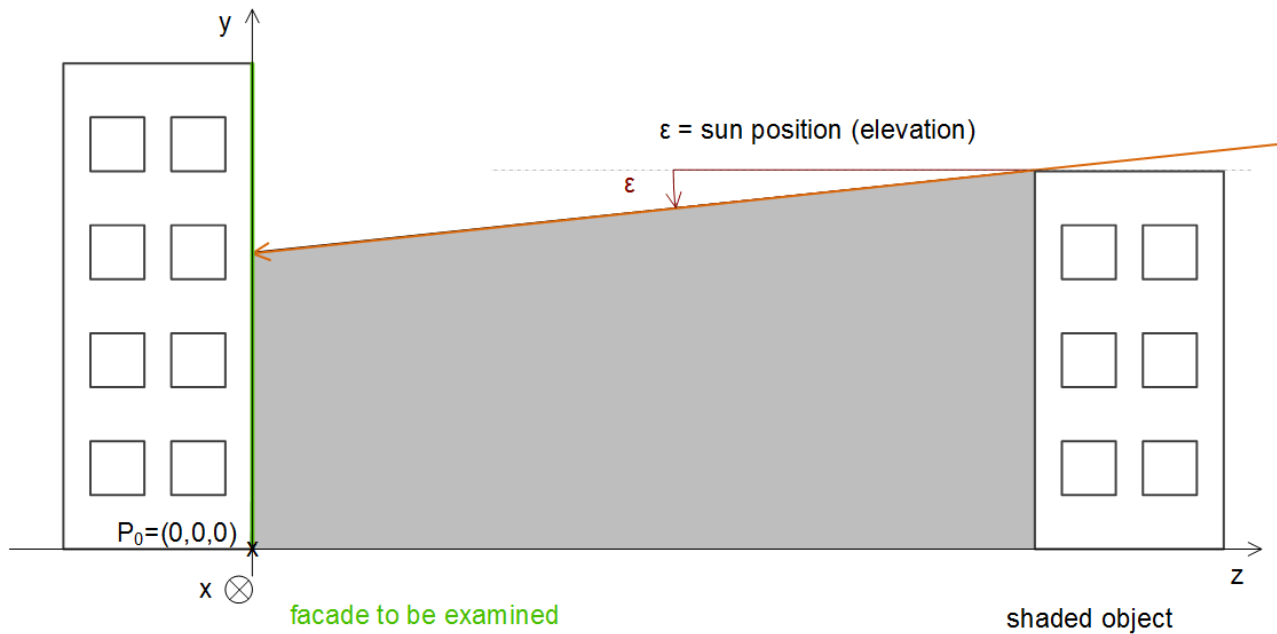
In the southern hemisphere the sun path is the other way round: Although it also rises in the east, at midday it is in the north. The facade orientation is adjusted to this path:

Line of sight	Facade orientation
South	$\beta=0^\circ$
East	$\beta=90^\circ$
North	$\beta=180^\circ$
West	$\beta=270^\circ$

For convenience, the other explanations refer to the northern hemisphere. The calculations for the southern hemisphere are analogous. When the function block `FB_BA_ShdCorr` [► 498] (shading correction) is parameterized they are activated through a boolean input, *bSouth*

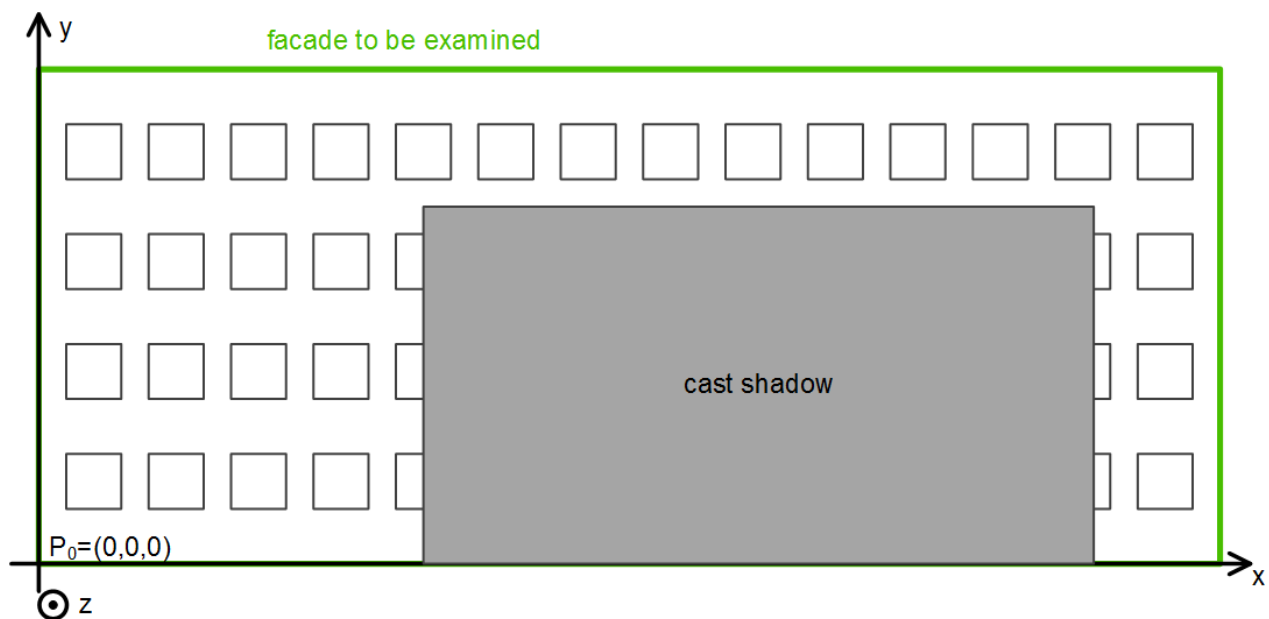
The following two illustrations are intended to further clarify the position of the point of origin P_0 as well as the orientation of the coordinate system:

Observation from the side



The angle of elevation (height of the sun) can be represented using this illustration: by definition this is 0° at sunrise (horizontal incidence of light) and can reach maximally 90° , but this applies only to places within the Tropic of Cancer and the Tropic of Capricorn.

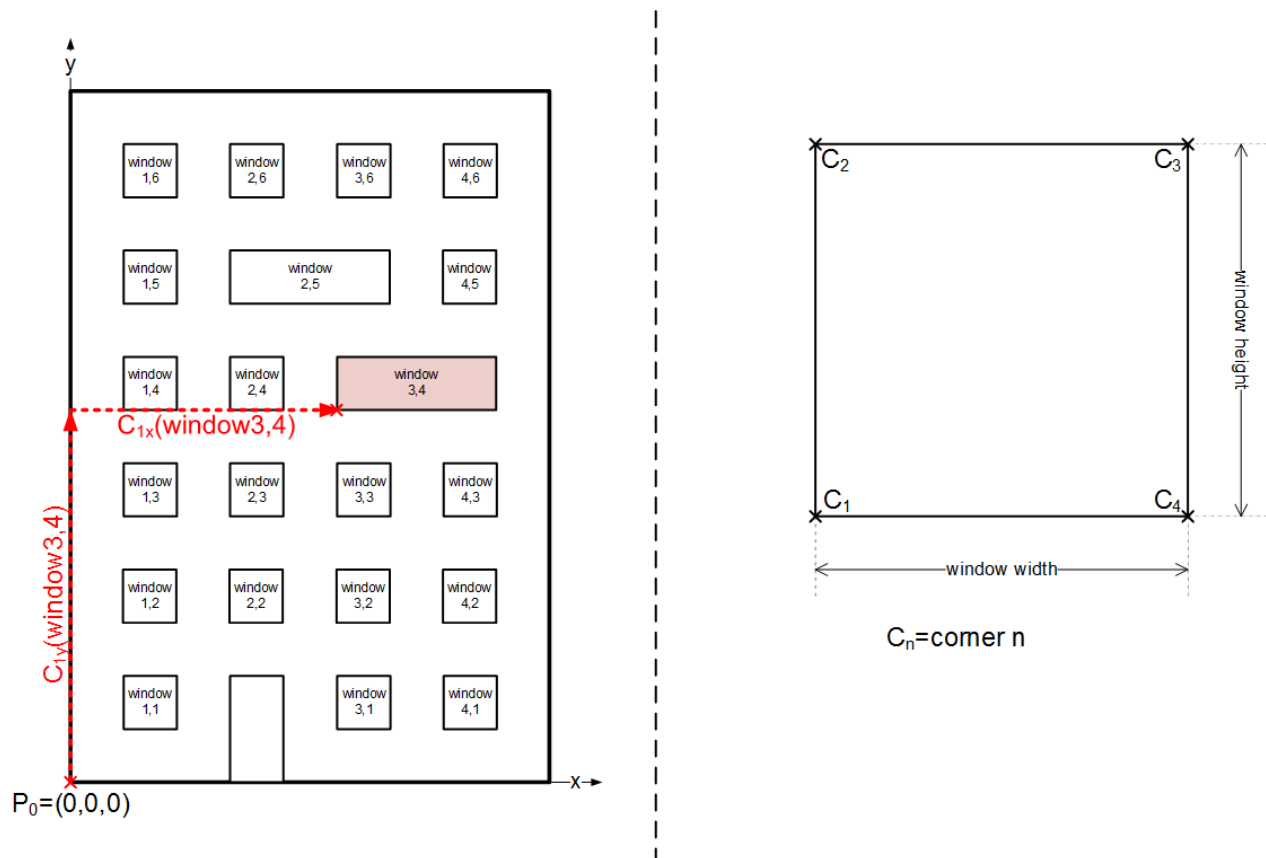
Observation from the front



Here, the position of the point of origin, P_0 , at the bottom left base point of the facade is once more very clear. Beyond that the X-Y orientation is illustrated, which is important later for the entry of the window elements.

Position of the windows

The position of the windows is defined by the specification of their bottom left corner in relation to the facade coordinate system. Since a window lies flat on the facade, the entry is restricted to the X and Y coordinates.



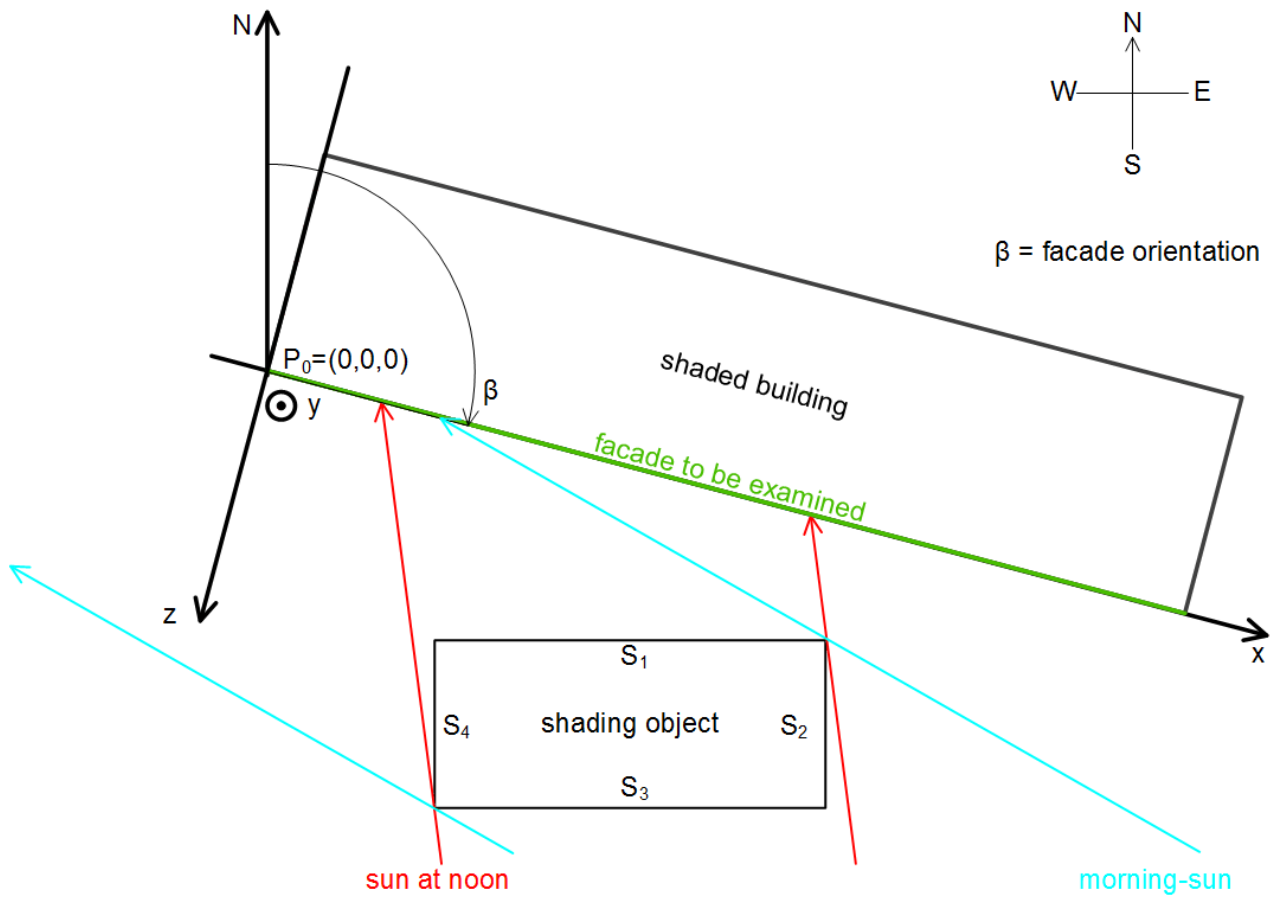
In addition, the window width and the window height have to be specified.

The position of each window corner on the facade is determined internally from the values entered. A window is considered to be in the shade if all corners lie in the shade.

Positioning of the shading objects

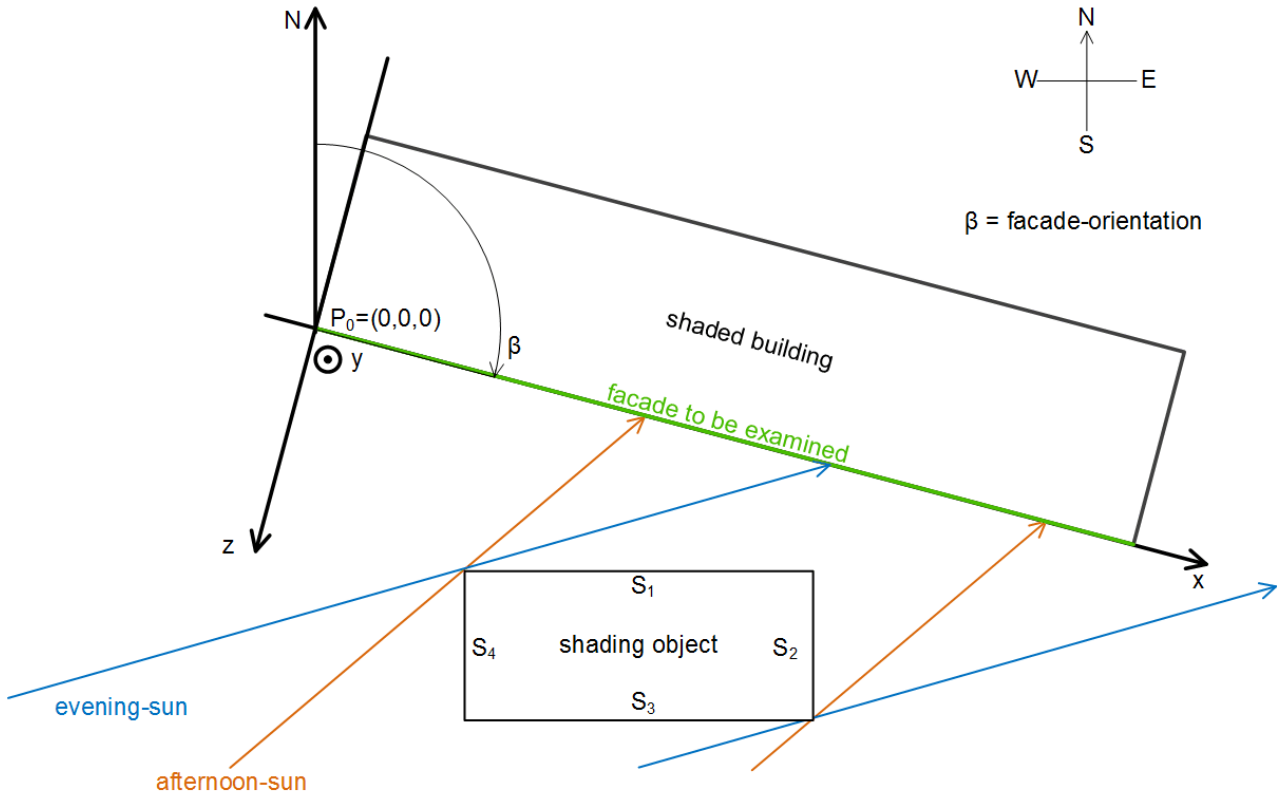
When describing the shading objects, distinction is made between angular objects (building, column) and objects that are approximately spherical (e.g. trees). Angular objects can be subdivided into rectangular shadow-casting facades depending on their shadow projection; you should consider which surfaces cast the main shadows over the day:

Morning/noon

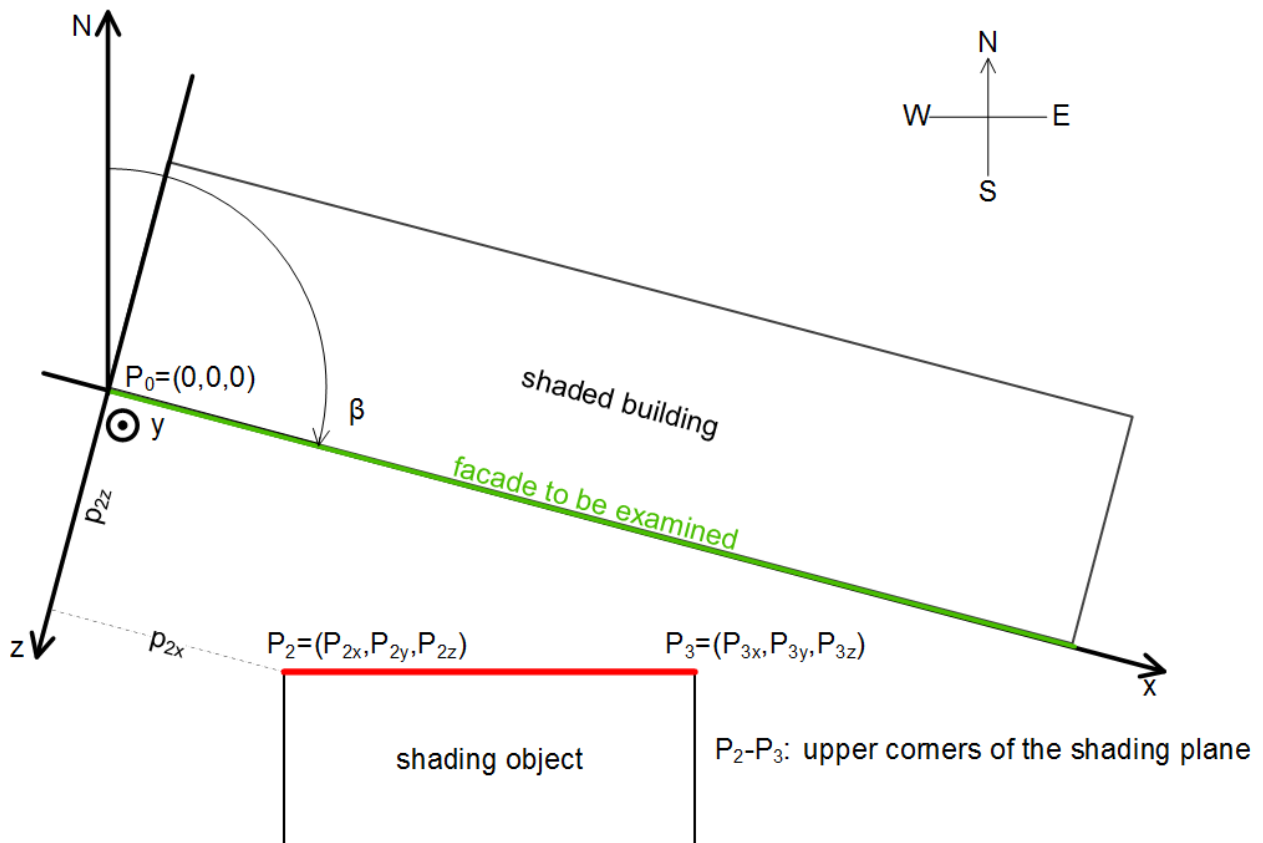


In the morning and around noon, the shadow is mainly cast by the sides S_1 and S_4 . S_2 and S_3 do not have to be considered, unless they are higher.

Afternoon/evening



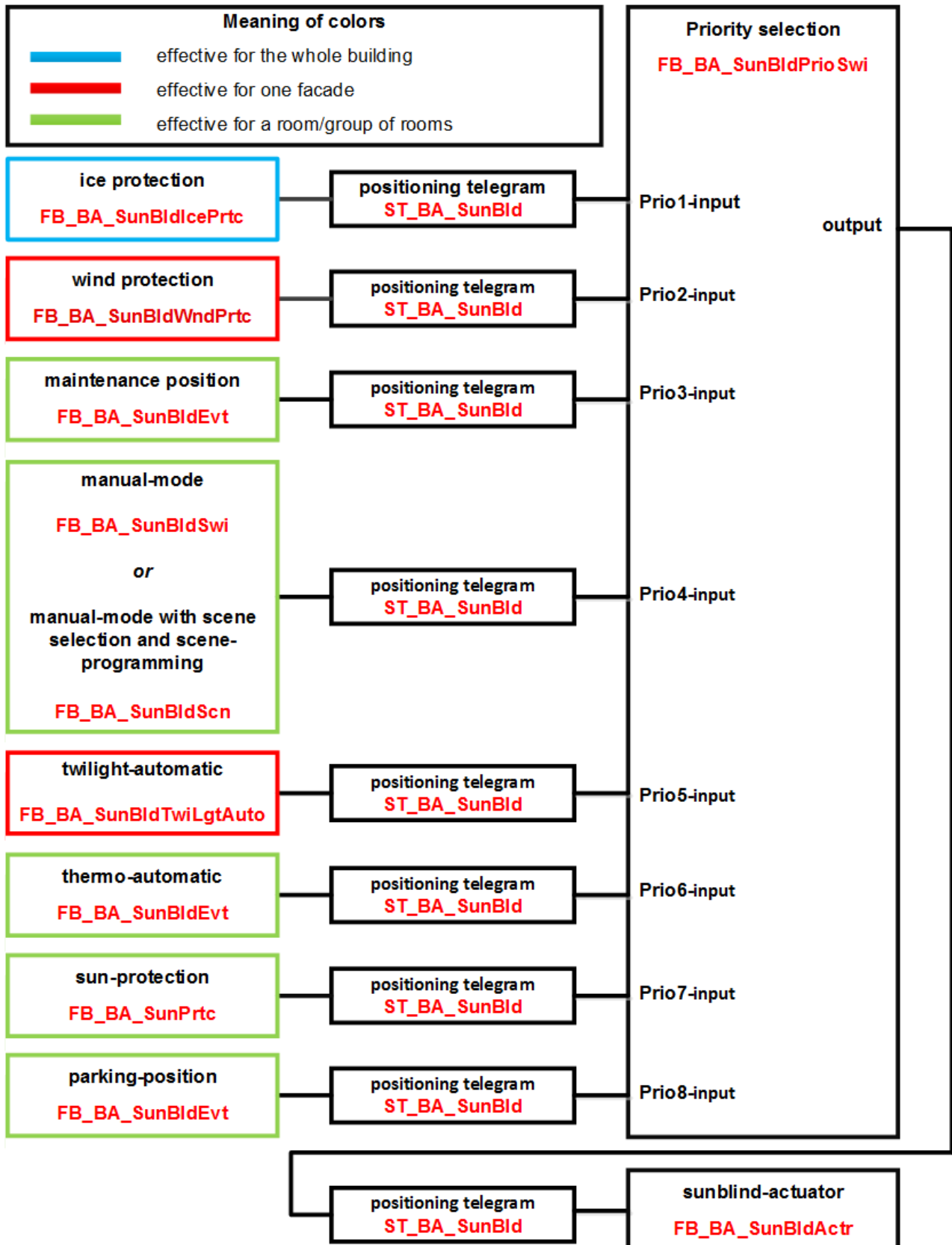
In the afternoon and evening, the total shade can be determined solely through S_1 and S_2 . In this case it is therefore sufficient to specify S_1, S_2 and S_4 as shadow casters. The entry is made on the basis of the four corners or their coordinates in relation to the zero point of the facade:



In this sketch only the upper points, P_2 and P_3 , are illustrated due to the plan view. The lower point P_1 lies underneath P_2 and P_4 lies underneath P_3 .

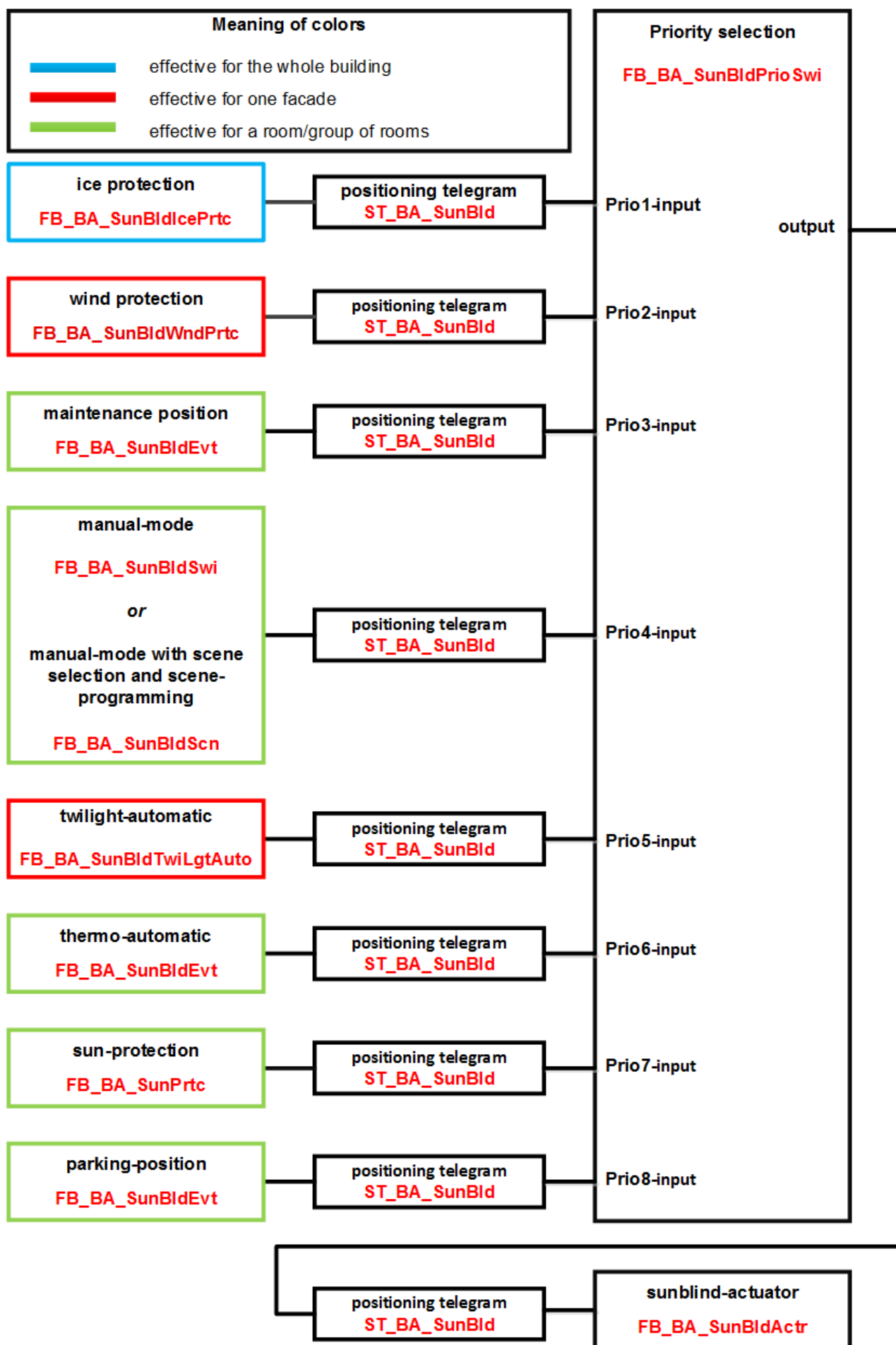
The input of shadow-casting ball elements is done by entering the center of the ball and its radius:

Ball elements



A "classification" of the ball element as in the case of the angular building is of course unnecessary, since the shadow cast by a ball changes only its direction, but not its size.

6.1.2.3.2.1.2.3.3 Overview of automatic sun protection



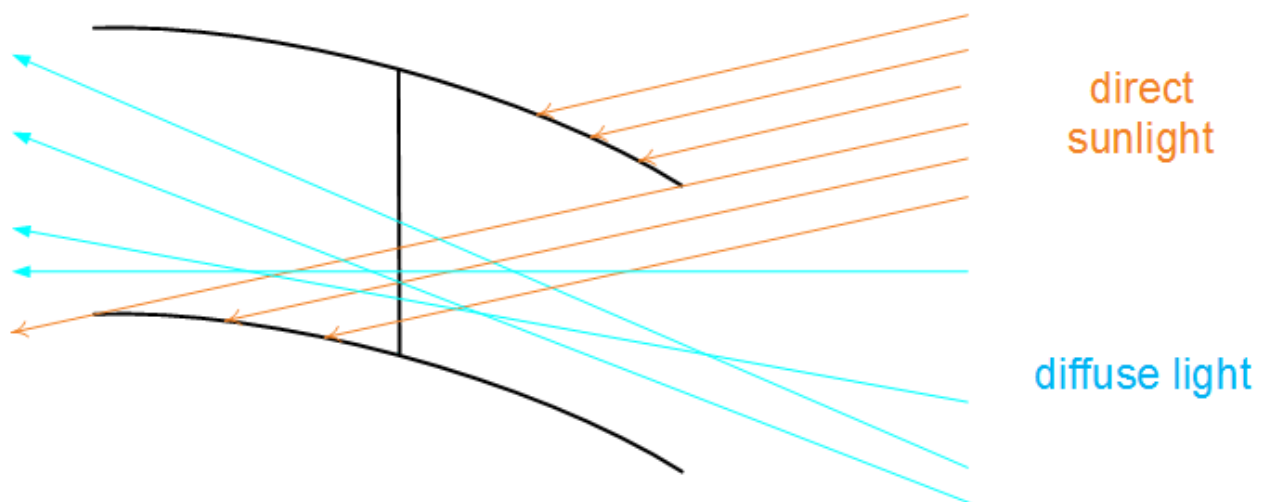
6.1.2.3.2.1.2.3.4 Sun protection: Basic principles and definitions

The direct incidence of daylight is regarded as disturbing by persons in rooms. On the other hand, however, people perceive natural light to be more pleasant in comparison with artificial light. Two options for glare protection are to be presented here:

- Louvre adjustment
- Height adjustment

Louvre adjustment

A louvered blind that can be adjusted offers the option of intelligent sun protection here. The position of the louvres is cyclically adapted to the current position of the sun, so that no direct daylight enters through the blinds, but as much diffuse daylight can be utilized as possible.



The illustration shows that diffuse light can still enter from underneath, whereas no further direct daylight, or theoretically only a single ray, can enter. The following parameters are necessary for the calculation of the louvre angle:

- the current height of the sun (angle of elevation)
- the sun position, i.e. the azimuth angle
- the facade orientation
- the louvre width
- the louvre spacing

effective elevation angle

If the blind is viewed in section as above, the angle of incidence does not depend solely on the solar altitude (elevation), but also on the direction of the sun:

- If the facade orientation and the sun position (azimuth) are the same, i.e. the sunlight falls directly onto the facade, the effective light incidence angle is the same as the current elevation angle.
- However, if the sunlight falls at an angle onto the facade as seen from the sun direction, the effective angle is larger for the same angle of elevation.

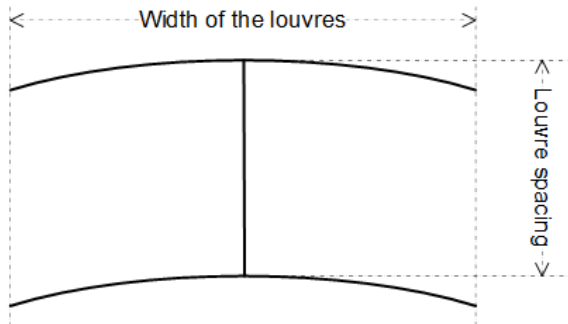
This relationship can easily be illustrated with a set square positioned upright on the table: Viewed directly from the side you can see a triangle with two 45° angles and one 90° angle. If the triangle is rotated, the side on the table appears to become shorter and the two original 45° angles change. The triangle appears to be getting steeper.

We therefore refer to the "effective elevation angle", which describes the proportion of light that falls directly onto the blind.

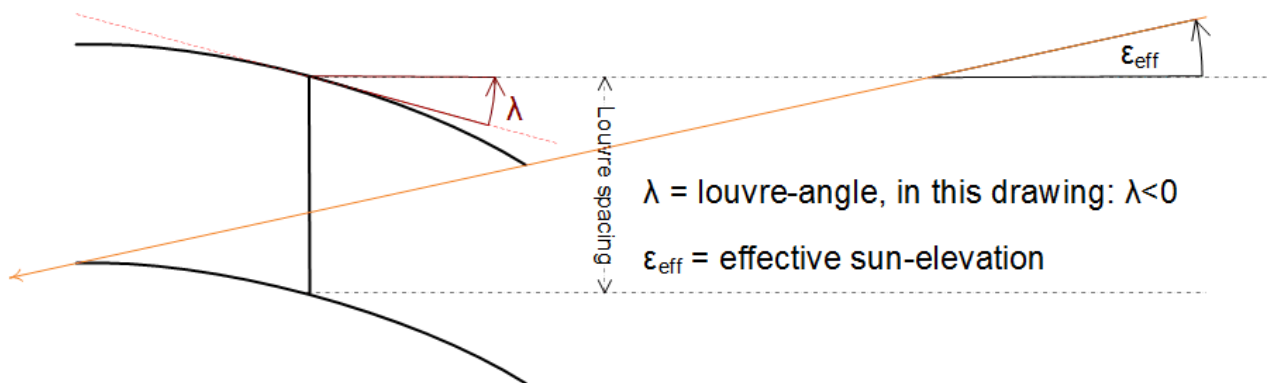
The following three images illustrate the relationship between the effective elevation angle and the blind dimensions, and how the resulting louvre angle λ changes during the day:

louvre-angle

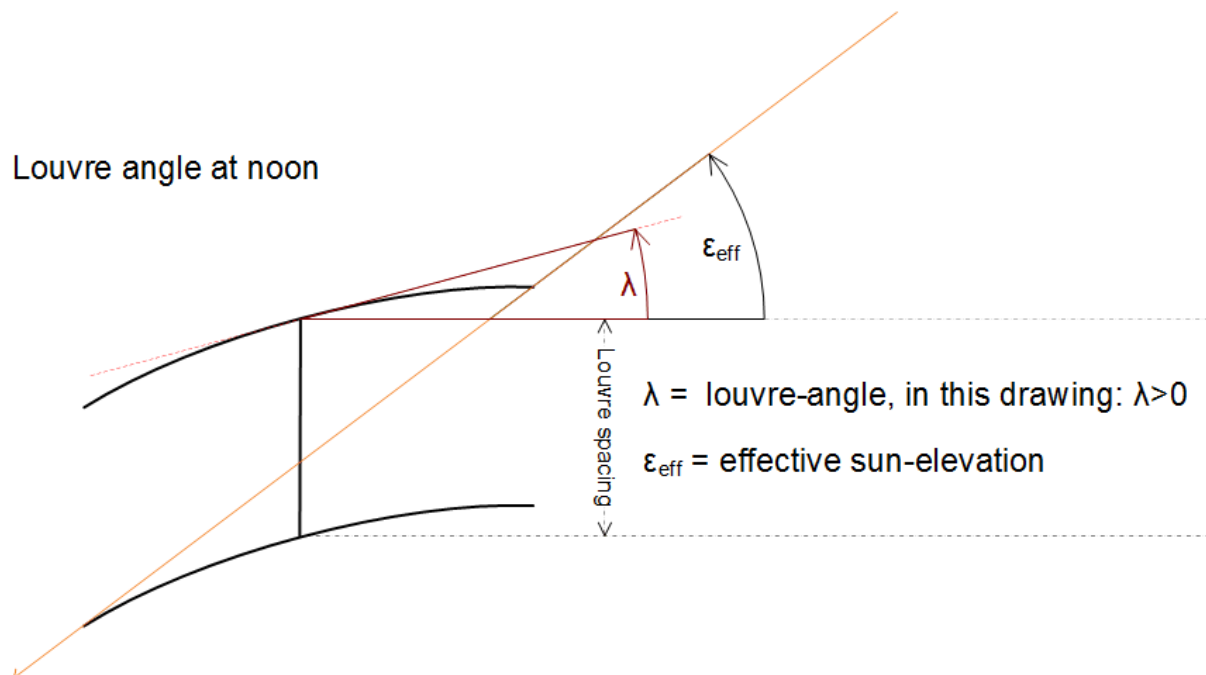
louvre at an angle of $\lambda=0$



Louvre-angle in the morning and in the evening

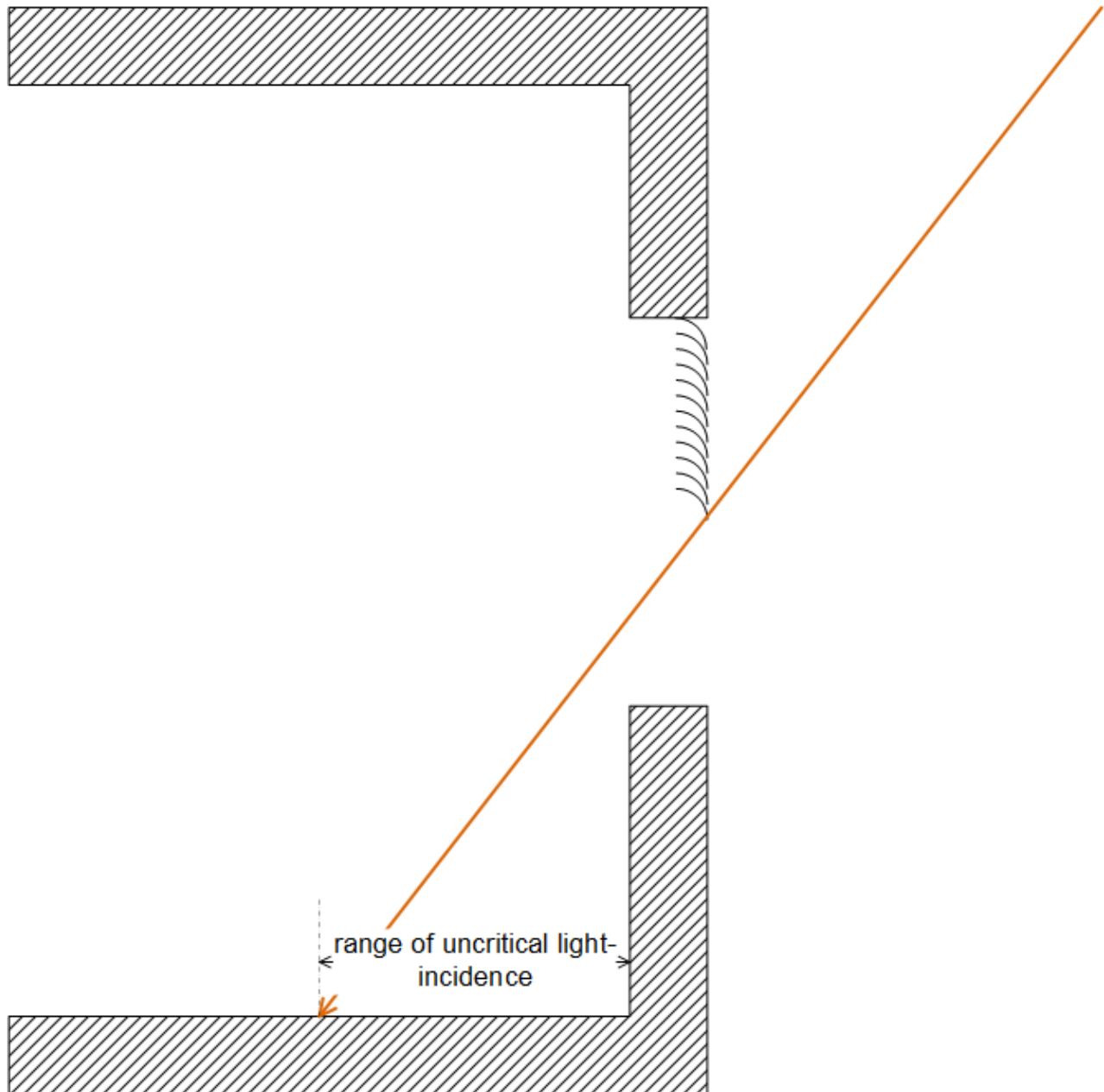


Louvre angle at noon



Height adjustment

With a high position of the sun at midday, the direct rays of sunlight do not penetrate into the full depth of the room. If direct rays of sunlight in the area of the window sill are regarded as uncritical, the height of the sun protection can be adapted automatically in such a way that the rays of sunlight only ever penetrate into the room up to an uncritical depth.

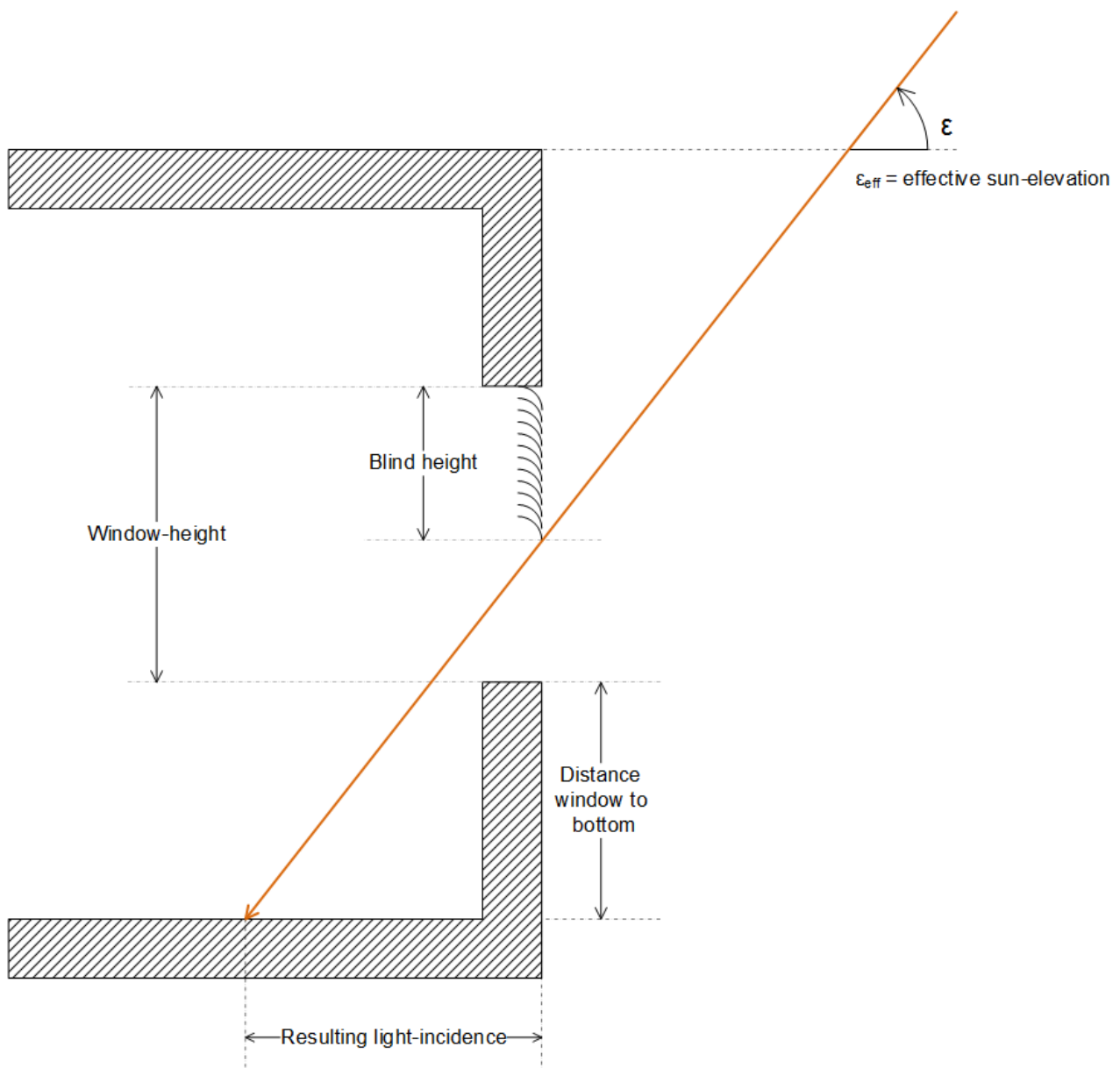


In order to be able to calculate at any time the appropriate blind height that guarantees that the incidence of sunlight does not exceed a certain value, the following values are necessary.

Required for the calculation of the respective blind height:

- Height of the sun (elevation)
- Window height
- Distance between the window and the floor

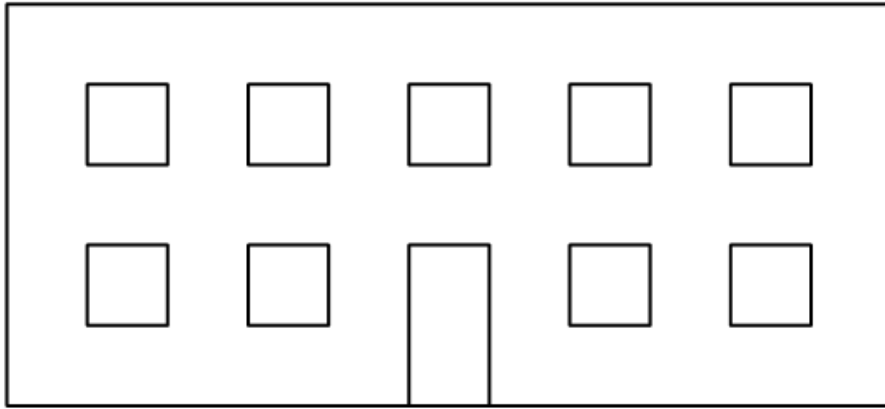
The following illustration shows where these parameters are to be classified:



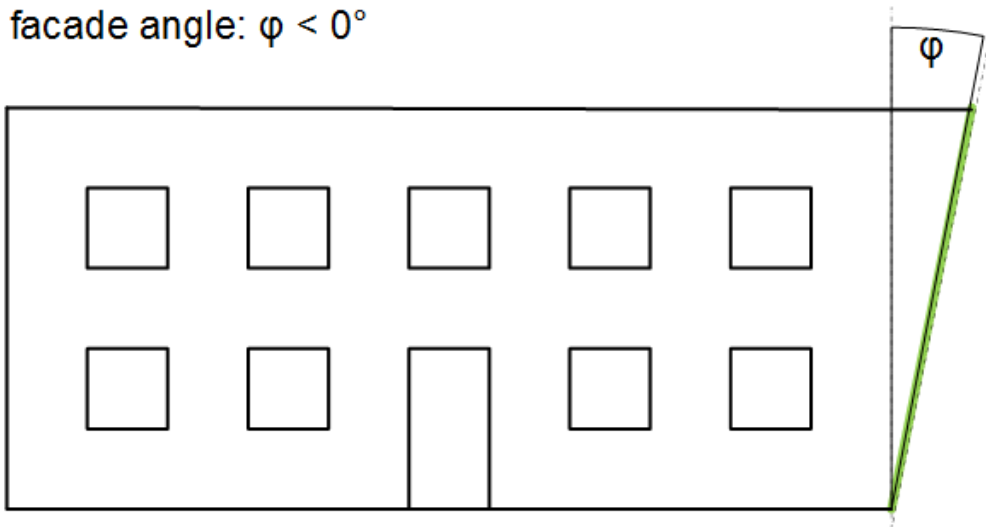
Influence of the facade inclination

In both of the methods of sun protection described, it was assumed that the facade and thus the windows are perpendicular to the ground. In the case of an inclined facade, however, the incidence of light changes such that this influence will also be taken into account. The facade inclination is defined as follows:

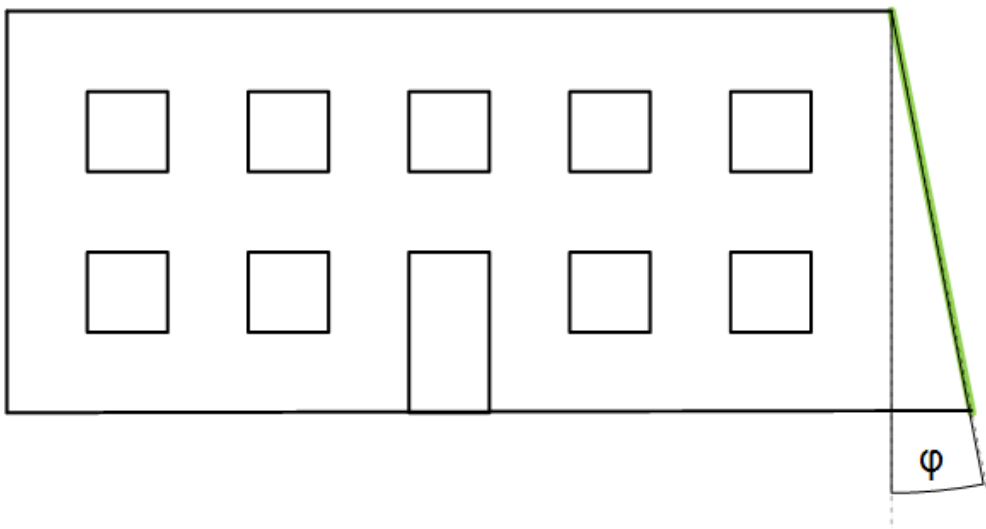
facade angle: $\varphi = 0^\circ$



facade angle: $\varphi < 0^\circ$



facade angle: $\varphi > 0^\circ$



6.1.2.3.2.1.2.3.5 List of shading elements

The data of all shading objects (building components, trees, etc.) per facade are stored in a field of structure elements of type `ST_BA_ShObj` [► 601] within the program.

The shading correction `FB_BA_ShCorr` [► 498] reads the information from this list. The management function block `FB_BA_ShObjEntry` [► 501] reads and writes it as input/output variable.

It is therefore advisable to declare this list globally:

```
VAR_GLOBAL
    arrShdObj : ARRAY[1..gBA_cMaxShdObj] OF ST_BA_ShObj;
END_VAR
```

The variable `gBA_cMaxShdObj` represents the upper limit of the available elements and is defined as a global constant within the program library:

```
VAR_GLOBAL CONSTANT
    gBA_cMaxShdObj : INT := 20;
END_VAR
```

6.1.2.3.2.1.2.3.6 List of facade elements

The data of all windows (facade elements) per facade are saved within the program in a field of structure elements of the type `ST_BA_FcdElem` [► 600].

The management function block `FB_BA_FcdElemEntry` [► 478] and the shading correction `FB_BA_ShCorr` [► 498] read and write to this list (the latter sets the shading information); they access this field as input/output variables.

It is therefore advisable to declare this list globally:

```
VAR_GLOBAL
    arrFcdElem : ARRAY[1..uiMaxRowFcd, 1..uiMaxColumnFcd] OF ST_BA_FcdElem;
END_VAR
```

The variables `uiMaxColumnFcd` and `uiMaxRowFcd` define the upper limit of the available elements and are declared as global constants within the program library:

```
VAR_GLOBAL CONSTANT
    uiMaxRowFcd : UINT := 10;
    uiMaxColumnFcd : UINT := 20;
END_VAR
```

6.1.2.3.2.1.2.3.7 FB_BA_BldPosEntry



This function block is used for entering interpolation points for the function block `FB_BA_SunPrtc` [► 522], if this function block is operated in height positioning mode with the aid of a table, see `E_BA_PosMod` [► 598].

In addition to the operating modes "Fixed shutter height" and "Maximum incidence of light", the function block `FB_BA_SunPrtc` [► 522] also offers the possibility to control the shutter height in relation to the position of the sun by means of table entries. By entering several interpolation points, the shutter height relative to the respective sun position is calculated by linear interpolation. However, since incorrectly entered values can lead to malfunctions in `FB_BA_SunPrtc` [► 522], this function block is to be preceded by the function block

FB_BA_BldPosEntry. Four interpolation points can be parameterized on this function block, whereby a missing entry is evaluated as a zero entry.

The function block does not sort the values entered independently, but instead ensures that the positions of the sun entered in the respective interpolation points are entered in ascending order. Unintentional erroneous entries are noticed faster as a result.

The values chosen for $rSunElv1$.. $rSunElv4$ must be unique; for example, the following situation must be avoided:

[$rSunElv1 = 10$; $rPos1 = 50$] and at the same time [$rSunElv2 = 10$; $rPos2 = 30$].

This would mean that there would be two different target values for one and the same value, which does not allow a unique functional correlation to be established.

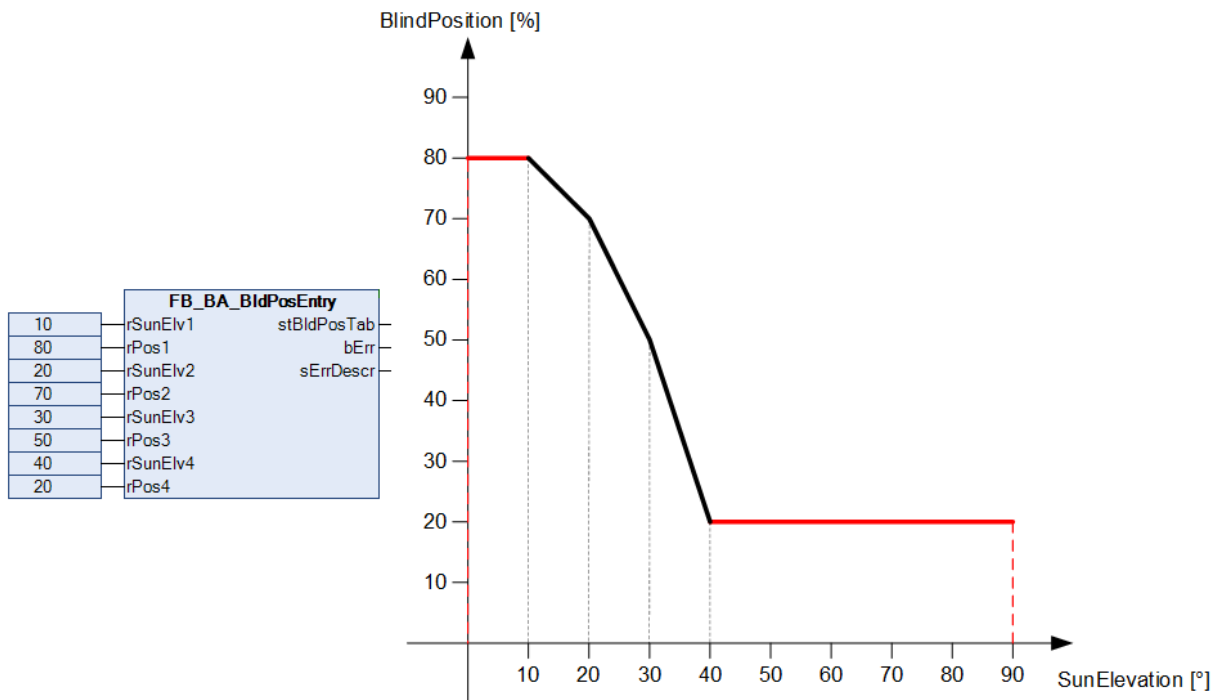
On top of that the entries for the position of the sun and shutter height must lie within the valid range.

Mathematically this means that the following conditions must be satisfied:

- $rSunElv1 < rSunElv2 < rSunElv3 < rSunElv4$ - (values ascending and not equal)
- $0 \leq rSunElv \leq 90$ ([°] - scope source values)
- $0 \leq rPos \leq 100$ (in percent - scope target values)

The function block checks the entered values for these conditions and issues an error message if they are not met. In addition, the value $bValid$ of ST_BA_BldPosTab [► 599] is set to FALSE.

Furthermore the function block independently ensures that the boundary areas are filled out: Internally, a further interpolation point is set at $rSunElv = 0$ with $rBldPos1$ and another one above $rSunElv4$ at $rSunElv = 90$ with $rBldPos4$. This ensures that a meaningful target value is available for all valid input values $0 \leq rSunElv \leq 90$, **without** the user having to enter $rSunElv = 0$ and $rSunElv = 90$:



This increases the actual number of interpolation points transferred to the function block FB_BA_SunPrtc [► 522] to 6; see ST_BA_BldPosTab [► 599].

The interpolation of the values takes place in the glare protection function block.

VAR_INPUT

```
rSunElv1 : REAL;
rPos1    : REAL;
rSunElv2 : REAL;
rPos2    : REAL;
rSunElv3 : REAL;
rPos3    : REAL;
rSunElv4 : REAL;
rPos4    : REAL;
```

rSunElv1: Sun position at the first interpolation point (0°..90°).

rPos1: Blind position (degree of closure) at the first interpolation point (0%..100%).

rSunElv2: Sun position at the second interpolation point (0°..90°).

rPos2: Blind position (degree of closure) at the second interpolation point (0%..100%).

rSunElv3: Sun position at the third interpolation point (0°..90°).

rPos3: Blind position (degree of closure) at the third interpolation point (0%..100%).

rSunElv4: Sun position at the fourth interpolation point (0°..90°).

rPos4: Blind position (degree of closure) at the fourth interpolation point (0%..100%).

VAR_OUTPUT

```
stBldPosTab : ST_BA_BldPosTab;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
```

stBldPosTab: Transfer structure of the interpolation points, see [ST_BA_BldPosTab](#) [► 599].

bErr: This output is switched to TRUE if the parameters entered are erroneous.

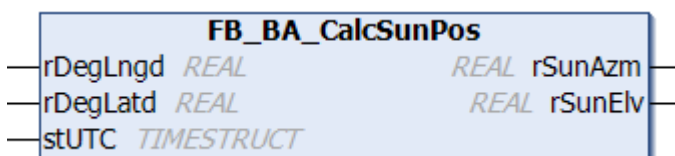
sErrDescr: Contains the error description.

Error description
01: Error: The x-values (elevation values) in the table are either not listed in ascending order, or they are duplicated.
02: Error: An elevation value that was entered is outside the valid range of 0°..90°.
03: Error: An position value that was entered is outside the valid range of 0°...100%.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

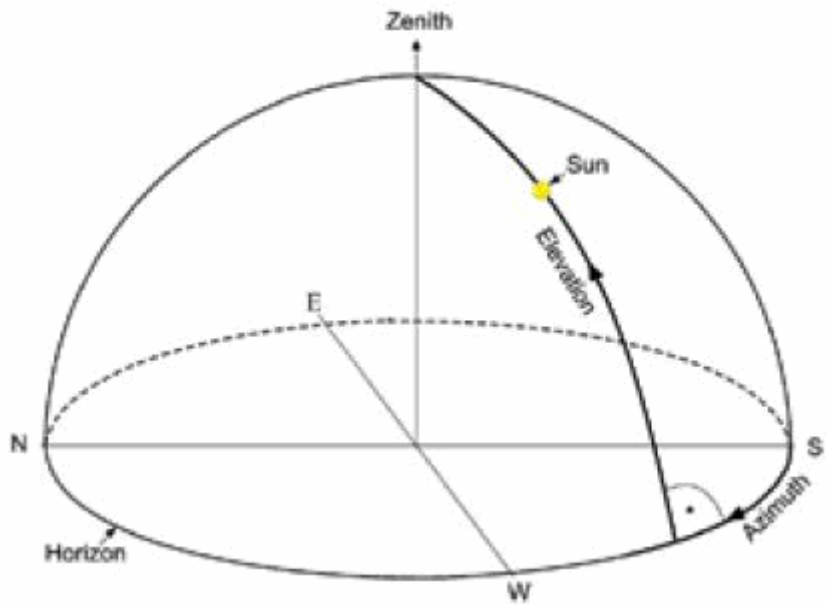
6.1.2.3.2.1.2.3.8 FB_BA_CalcSunPos



Calculation of sun position based on the date, time, longitude and latitude.

The position of the sun for a given point in time can be calculated according to common methods with a defined accuracy. For applications with moderate requirements, the present function block is sufficient. As the basis for this, the SUNAE algorithm was used, which represents a favorable compromise between accuracy and computing effort.

The position of the sun at a fixed observation point is normally determined by specifying two angles. One angle indicates the height above the horizon; 0° means that the sun is in the horizontal plane of the location; a value of 90° means that the sun is perpendicular to the observer. The other angle indicates the direction at which the sun is positioned. The SUNAE algorithm is used to distinguish whether the observer is standing on the northern hemisphere (longitude > 0 degrees) or on the southern hemisphere (longitude < 0 degrees) of the earth. If the observation point is in the northern hemisphere, then a value of 0° is assigned for the northern sun direction and it then runs in the clockwise direction around the compass, i.e. 90° is east, 180° is south, 270° is west etc. If the point of observation is in the southern hemisphere, then 0° corresponds to the southern direction and it then runs in the counter clockwise direction, i.e. 90° is east, 180° is north, 270° is west etc.



The time has to be specified as coordinated world time (UTC, Universal Time Coordinated, previously referred to as GMT, Greenwich Mean Time).

The latitude is the northerly or southerly distance of a location on the Earth's surface from the equator, in degrees [°]. The latitude can assume a value from 0° (at the equator) to ±90° (at the poles). A positive sign thereby indicates a northern direction and a negative sign a southern direction. The longitude is an angle that can assume values up to ±180° starting from the prime meridian 0° (an artificially determined North-South line). A positive sign indicates a longitude in an eastern direction and a negative sign in a western direction. Examples:

Location	Longitude	Latitude
Sydney, Australia	151.2°	-33.9°
New York, USA	-74.0	40.7°
London, England	-0.1°	51.5°
Moscow, Russia	37.6°	55.7°
Peking, China	116.3°	39.9°
Dubai, United Arab Emirates	55.3°	25.4°
Rio de Janeiro, Brazil	-43.2°	-22.9°
Hawaii, USA	-155.8°	20.2°
Verl, Germany	8.5°	51.9°

If the function block `FB_BA_CalcSunPos` returns a negative value for the solar altitude `rSunElv`, the sun is invisible. This can be used to determine sunrise and sunset.

VAR_INPUT

```
rDegLngd : REAL;  
rDegLatd : REAL;  
stUTC : TIMESTRUCT;
```

rDegLngd: Longitude [°].

rDegLatd: Latitude [°].

stUTC: Input of the current time as coordinated world time (see TIMESTRUCT). The function block `FB_BA_GetTime` [► 531] can be used to read this time from a target system.

VAR_OUTPUT

```
rSunAzm : REAL;  
rSunElv : REAL;
```

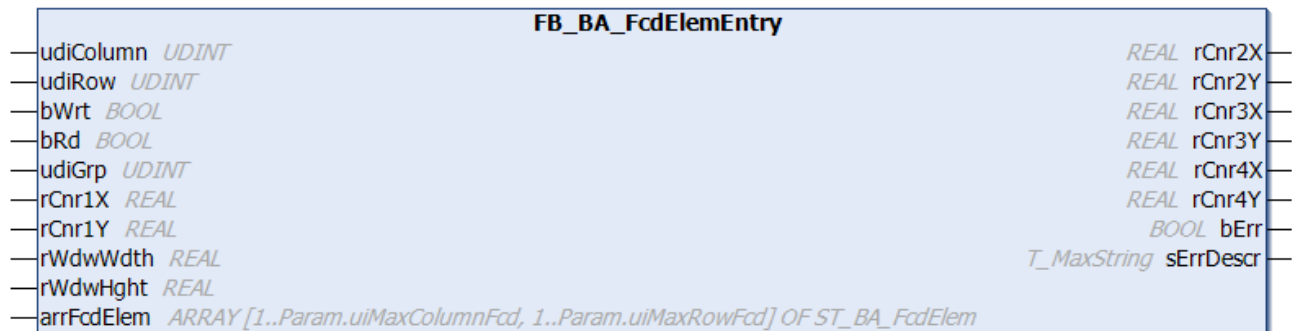
rSunAzm: Direction of the sun (northern hemisphere: 0° north ... 90° east ... 180° south ... 270° west ... / southern hemisphere: 0° south ... 90° east ... 180° north ... 270° west ...).

rSunElv: Height of the sun (0° horizontal - 90° vertical).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.9 FB_BA_FcdElemEntry



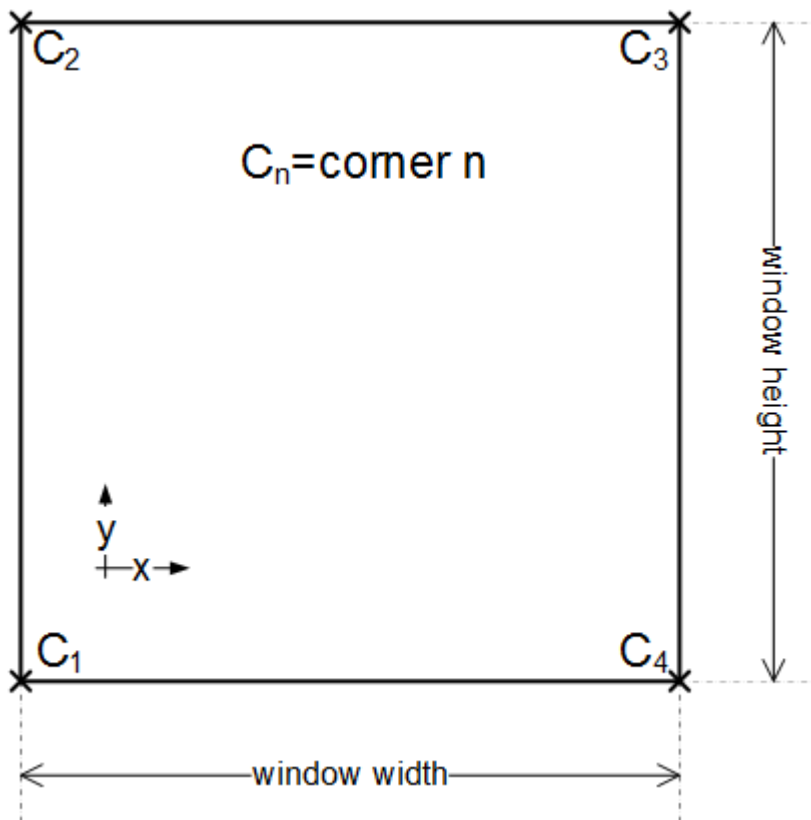
This function block serves the administration of all facade elements (windows) in a facade, which are saved globally in a [list of facade elements](#) [► 474]. It is intended to facilitate inputting element information - not least with regard to using the TC3 PLC HMI. A schematic illustration of the objects with description of the coordinates is given in [Shading correction: principles and definitions](#) [► 459].

The facade elements are declared in the global variables as a two-dimensional field above the window columns and rows:

```
VAR_GLOBAL
    arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Each element *arrFcdElem[x,y]* contains the information for an individual facade element (ST_BA_FcdElem [► 600]). The information includes the group affiliation, the dimensions (width, height) and the coordinates of the corners. The function block thereby accesses this field directly via the IN-OUT variable *arrFcdElem*.

Note: The fact that the coordinates of corners C2 to C4 are output values arises from the fact that they are formed from the input parameters and are to be available for use in a visualization:



All entries in [m]!

```

rCnr2X = rCnr1X
rCnr2Y = rCnr1Y + rWdwHght (window hight)
rCnr3X = rCnr1X + rWdwWdth (window width)
rCnr3Y = rCnr2Y
rCnr4X = rCnr1X + rWdwWdth (window width)
rCnr4Y = rCnr1Y

```

The function block is used in three steps:

- Read
- Change
- Write

Read

The entries *udiColumn* and *udiRow* are used to select the corresponding element from the list, *arrFcdElem[udiColumn, udiRow]*. A rising edge on *bRd* reads the following data from the list element:

- *usiGrp* group membership,
- *rCnr1X* x-coordinate of corner point 1 [m]
- *rCnr1Y* y-coordinate of corner point 1 [m]
- *rWdwWdth* window width [m]
- *rWdwHght* window height [m]

These are then assigned to the corresponding input variables of the function block, which uses them to calculate the coordinates of corners C_2 - C_4 as output variables in accordance with the correlation described above. It is important here that the input values are not overwritten in the reading step. Hence, all values can initially be displayed in a visualization.

Change

In a next program step the listed input values can then be changed. The values entered are constantly checked for plausibility. The output *bErr* indicates whether the values are valid (*bErr*=FALSE). If the values are invalid, a corresponding error message is issued at output *sErrDescr*. See also "Error (*bErr*=TRUE)" below.

Write

The parameterized data are written to the list element with the index *nId* upon a rising edge on *bWrt*, regardless of whether they represent valid values or not. The element structure ST_BA_FcdElem [► 600] therefore also contains a plausibility bit *bVld*, which forwards precisely this information to the function block FB_BA_Shdcorr [► 498] to prevent miscalculations.

This approach is to be regarded only as a proposal. It is naturally also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on *bWrt*.

Error (*bErr*=TRUE)

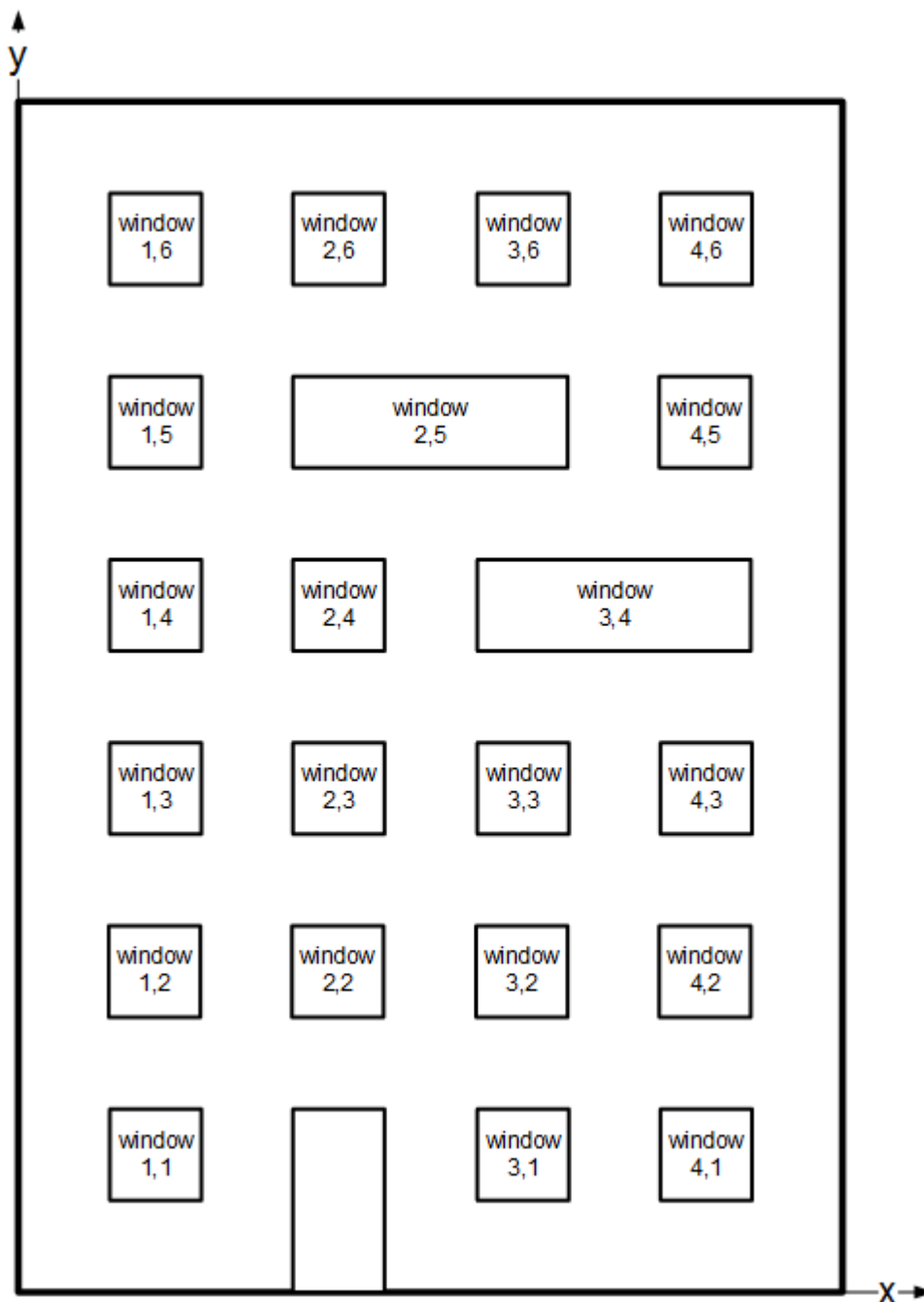
The function block FB_BA_Shdcorr [► 498], which judges whether all windows in a group are shaded, will only perform its task if all windows in the examined group have valid entries.

This means:

- *usiGrp* must be greater than 0
- *rCnr1X* must greater than or equal to 0.0
- *rCnr1Y* must greater than or equal to 0.0
- *rWdwWdth* must be greater than 0
- *rWdwHght* must be greater than 0

If one of these criteria is not met, it is interpreted as incorrect input, and the error output *bErr* is set at the function block output of FB_BA_FcdElemEntry. Within the window element ST_BA_FcdElem [► 600], the plausibility bit *bVld* is set to FALSE.

If on the other hand **all** entries of a facade element are zero, it is regarded as a valid, deliberately omitted facade element:



In the case of a facade of 6x4 windows, the elements window (2.1), window (3.5) and window (4.4) would be empty elements here.

VAR_INPUT

```

udiColumn : UDINT;
udiRow    : UDINT;
bWrt      : BOOL;
bRd       : BOOL;
udiGrp    : UDINT;
rCnrlX    : REAL;
rCnrlY    : REAL;
rWdwWdth  : REAL;
rWdwHght  : REAL;

```

udiColumn: Column index of the selected component on the facade. This refers to the selection of a field element of the array stored in the IN-OUT variable *arrFcdElem*.

udiRow: ditto. row index. *udiRow* and *udiColumn* must not be zero! This is due to the field definition, which always starts with 1; see above.

bRd: A positive edge at this input causes the information of the selected element, *arrFcdElem[udiColumn,udiRow]*, to be read into the function block and assigned to the input variables *diGrp* to *rWdwHght*. The resulting output variables are *rCnr2X* to *rCnr4Y*. If data are already present on the inputs *diGrp* to *rWdwHght* at time of reading, then the data previously read are immediately overwritten with these data.

bWrt: A positive edge writes the entered and calculated values into the selected field element *arrFcdElem[udiColumn, udiRow]*.

udiGrp: Group membership. Internally limited to a minimum value of 0.

rCnr1X: X-coordinate of corner point 1 [m].

rCnr1Y: Y-coordinate of corner point 1 [m].

rWdwWdth: Window width [m].

rWdwHght: Window height [m].

VAR_OUTPUT

```
rCnr2X : REAL;
rCnr2Y : REAL;
rCnr3X : REAL;
rCnr3Y : REAL;
rCnr4X : REAL;
rCnr4Y : REAL;
bErr   : BOOL;
sErrDesc : T_MAXSTRING;
```

rCnr2X: Calculated X-coordinate of corner point 2 of the window [m]. See "[Note \[► 478\]](#)" above.

rCnr2Y: Calculated Y-coordinate of corner point 2 of the window [m]. See "[Note \[► 478\]](#)" above.

rCnr3X: Calculated X-coordinate of corner point 3 of the window [m]. See "[Note \[► 478\]](#)" above.

rCnr3Y: Calculated Y-coordinate of corner point 3 of the window [m]. See "[Note \[► 478\]](#)" above.

rCnr4X: Calculated X-coordinate of corner point 4 of the window [m]. See "[Note \[► 478\]](#)" above.

rCnr4Y: Calculated Y-coordinate of corner point 4 of the window [m]. See "[Note \[► 478\]](#)" above.

bErr: Result verification for the entered values.

sErrDesc: Contains the error description.

Error description
01: Error: Index error! udiColumn and/or udiRow are outside the permitted limits, 1.. uiMaxColumnFcd and 1.. uiMaxColumnFcd, respectively. See list of facade elements.
02: Error: The group index is 0, but at the same time another entry of the facade element is not zero. Only if all entries of a facade element are zero is it considered to be a valid, deliberately omitted facade component, otherwise it is interpreted as an incorrect entry. NOTE: Group entries less than zero are internally limited to zero.
03: Error: The X-component of the first corner point (Corner1) is less than zero.
04: Error: The Y-component of the first corner point (Corner1) is less than zero.
05: Error: The window width is less than or equal to zero.
06: Error: The window height is less than or equal to zero.

VAR_IN_OUT

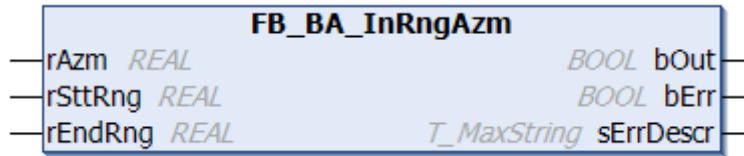
```
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: List of facade elements (see [List of facade elements \[► 474\]](#)).

Requirements

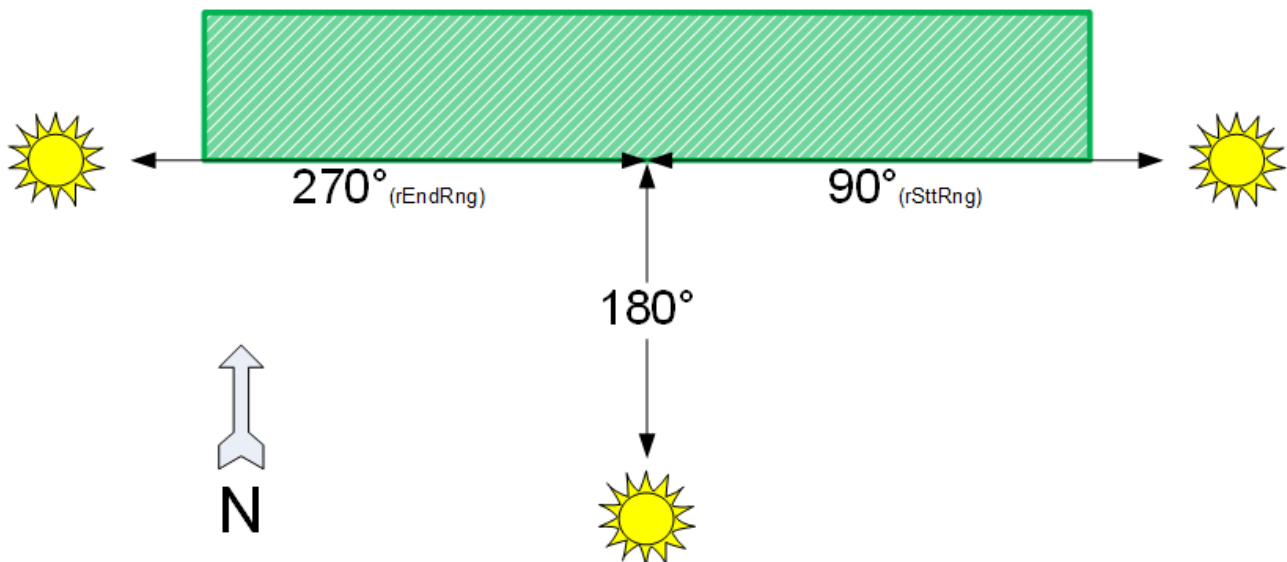
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.10 FB_BA_InRngAzm

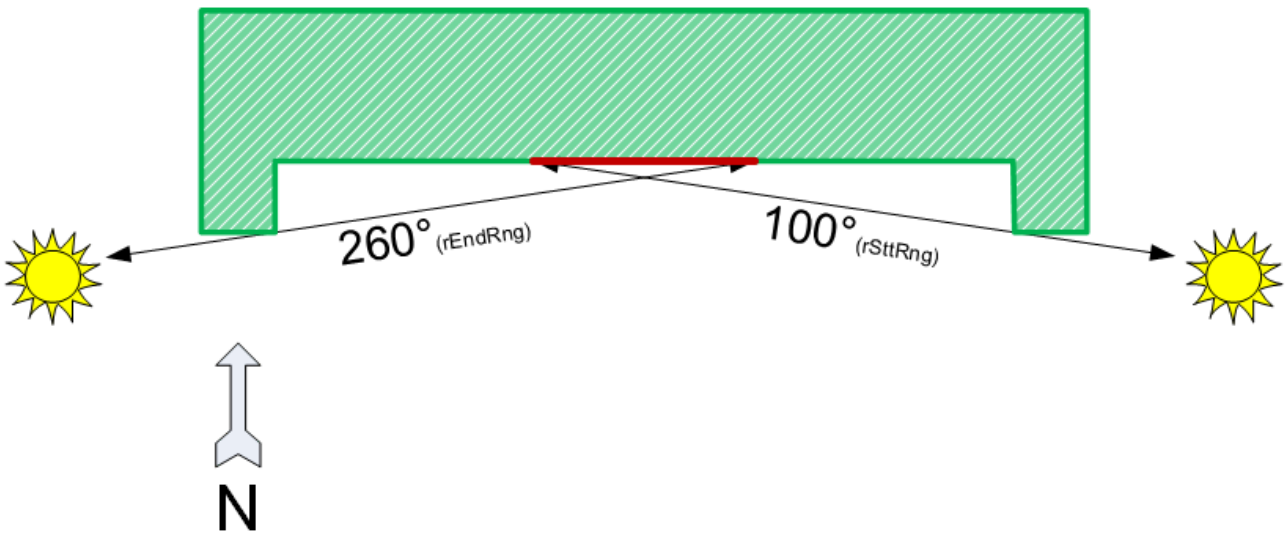


This function block checks whether the current azimuth angle (horizontal position of the sun) lies within the limits entered. As can be seen in the [overview \[► 467\]](#), the function block provides an additional evaluation as to whether the sun shading of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

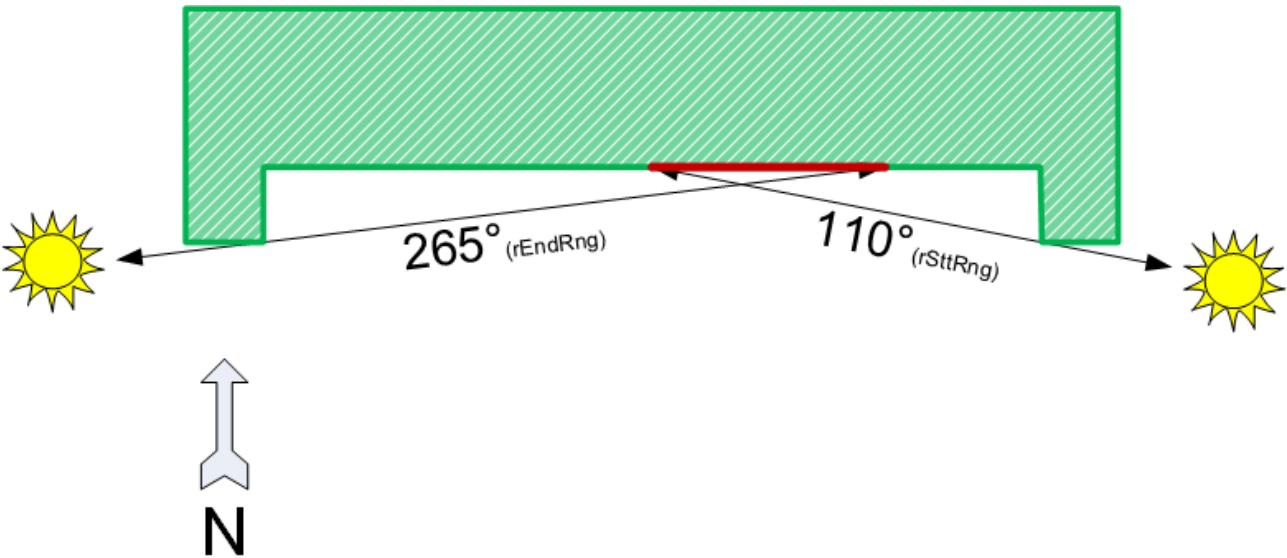
A smooth facade is always irradiated by the sun at an azimuth angle of *Facade orientation*-90° to *Facade orientation*+90°.



If the facade has lateral projections, however, this range is limited. This limitation can be checked with the help of this function block. However, the position of the window group on the facade also plays a role. If it lies centrally, this gives rise to the following situation (the values are only examples):



The values change for a group at the edge:



The start of the range $rSttRng$ may be greater than the end $rEndRng$, in which case values beyond 0° are considered:

Sample

rAzm	10.0°
rSttRng	280.0°
rEndRng	20.0°
bOut	TRUE

However, the range regarded may not be greater than 180° or equal to 0° – this would be unrealistic. Such entries result in an error on the output $bErr$ – the test output $bOut$ is then additionally set to FALSE.

VAR_INPUT

```
rAzm    : REAL;  
rSttRng : REAL;  
rEndRng : REAL;
```

rAzm: Current azimuth angle.

rSttRng: Start of range [°].

rEndRng: End of range [°].

VAR_OUTPUT

```
bOut      : BOOL;  
bErr      : BOOL;  
sErrDescr : T_MAXSTRING;
```

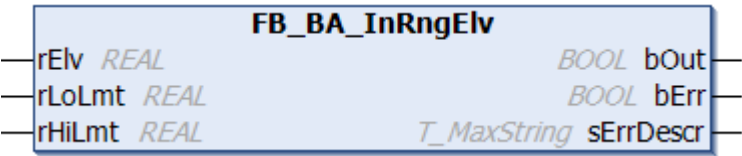
- bOut:** The facade element is in the sun if the output is TRUE.
- bErr:** This output is switched to TRUE if the parameters entered are erroneous.
- sErrDescr:** Contains the error description.

Error description
01: Error: <i>rSttRng</i> or <i>rEndRng</i> less than 0° or greater than 360°.
02: Error: The difference between <i>rSttRng</i> and <i>rEndRng</i> is greater than 180°. This range is too large for analyzing the insolation on a facade.

Requirements

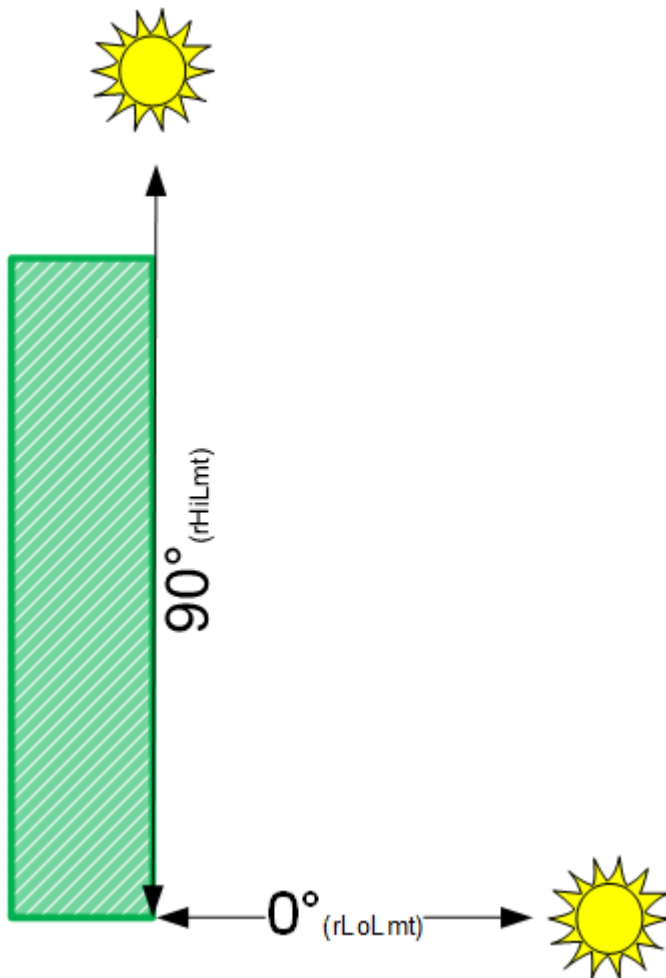
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.11 FB_BA_InRngElv

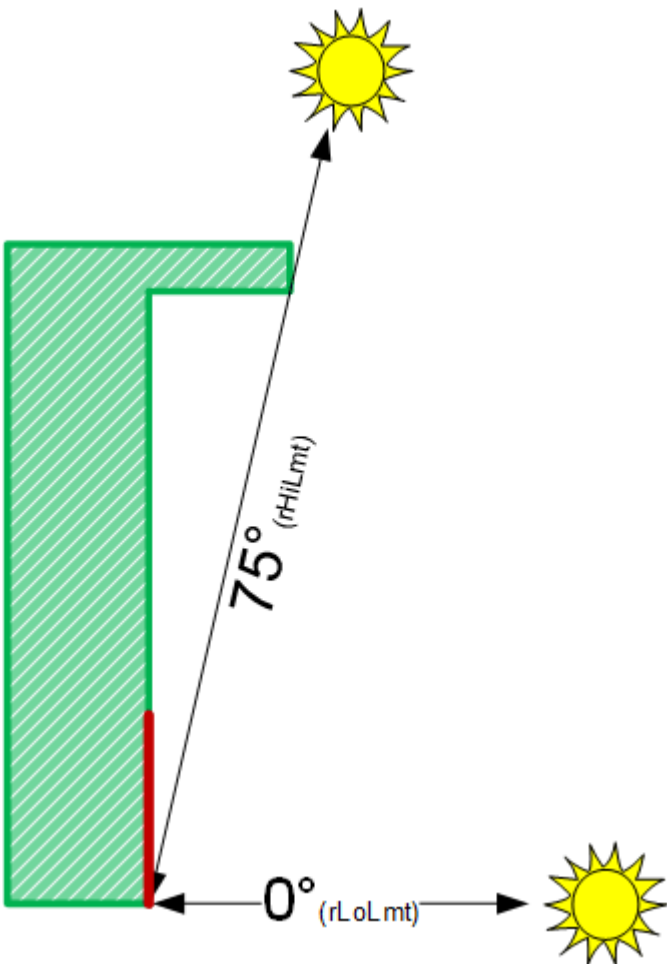


This function block checks whether the current angle of elevation (vertical position of the sun) lies within the limits entered. As can be seen in the [overview \[► 467\]](#), the function block provides an additionally evaluation as to whether the sun shading of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

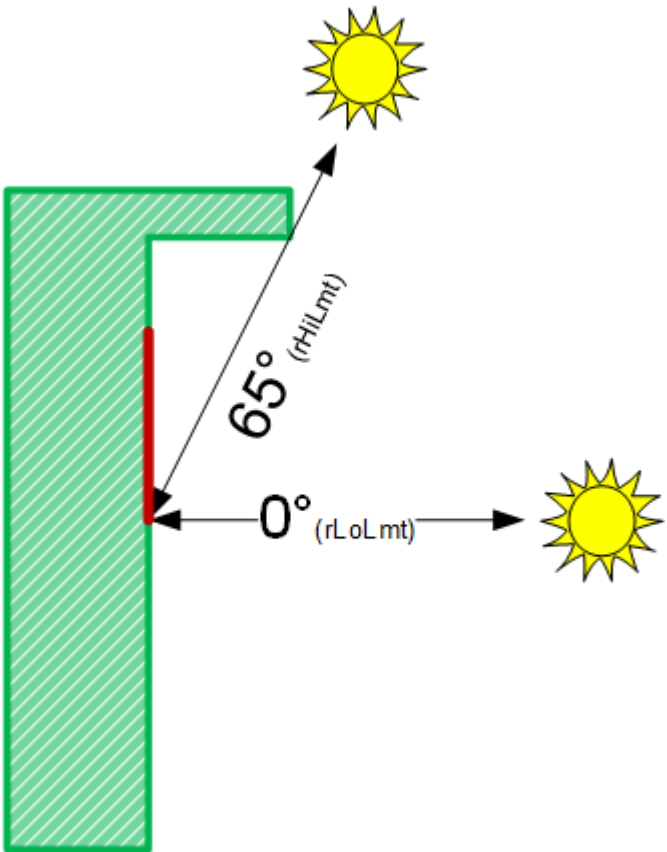
A normal vertical facade is irradiated by the sun at an angle of elevation of 0° to maximally 90°.



If the facade has projections, however, this range is limited. This limitation can be checked with the help of this function block. However, the position of the window group on the facade also plays a role. If it lies in the lower range, this gives rise to the following situation (the values are only examples):



The values change for a group below the projection:



The lower observation limit, *rLoLmt*, may thereby not be greater than or equal to the upper limit, *rHiLmt*. Such entries result in an error on the output *bErr* – the test output *bOut* is then additionally set to FALSE.

VAR_INPUT

```
rElv    : REAL;
rLoLmt  : REAL;
rHiLmt  : REAL;
```

rElv: Current elevation angle [°].

rLoLmt: Lower limit value [°].

rHiLmt: Upper limit value [°].

VAR_OUTPUT

```
bOut     : BOOL;
bErr     : BOOL;
sErrDescr : T_MAXSTRING;
```

bOut: The facade element is in the sun

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description
01: Error: <i>rHiLmt</i> less than or equal to <i>rLoLmt</i> .
02: Error: <i>rLoLmt</i> is less than 0° or <i>rHiLmt</i> is greater than 90°.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.12 FB_BA_RdFcdElemLst

FB_BA_RdFcdElemLst	
bStt <i>BOOL</i>	<i>BOOL</i> bBusy
sDataFile <i>STRING</i>	<i>UDINT</i> udiAmtSetsRd
sLogFile <i>STRING</i>	<i>BOOL</i> bErr
sNetId <i>T_AmsNetId</i>	<i>T_MaxString</i> sErrDescr
arrFcdElem <i>ARRAY [1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem</i>	<i>BOOL</i> bErrDataSet

With the help of this function block, data for facade elements (windows) can be imported from a pre-defined Excel table in csv format into the list of facade elements [► 474]. In addition the imported data are checked for plausibility and errors are written to a log file.

The following example shows the Excel table with the entries of the window elements.

All text fields are freely writable. The fields marked in green are important; each line in these fields identifies a data set.

The following rules are to be observed:

- A data set must always start with a '@'.
- The indices *IndexColumn* and *IndexRow* must lie within the defined limits, see [List of facade elements](#) [► 474]. These indices directly describe the facade element in the list *arrFcdElem* to which the data from the set are saved.
- Window width and window height must be greater than zero
- The corner coordinates P1x and P1y must be greater than or equal to zero.
- Each window element must be assigned to a group 1..255.
- For system-related reasons the total size of the table may not exceed 65534 bytes.
- This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)".

It is not necessary to describe all window elements that would be possible by definition or declaration. Before the new list is read in, the function block deletes the entire old list in the program. All elements that are not described by entries in the Excel table then have pure zero entries and are thus marked as non-existent and also non-evaluable, since the function block for shading correction, FB BA ShdCorr [► 498], does not accept elements with the group entry '0'.

EN_FacadeElements.xls										
	A	B	C	D	E	F	G	H	I	J
1	Number	Description		IndexColumn	IndexRow	Window-Width	Window-Height	P1x	P1y	Group
2				(Axis)	(Floor)	[m]	[m]	[m]	[m]	
3		Text								
4	1	Description	@	1	1	1,2	1,3	1,5	1	2
5	2	Description	@	0	1	1,2	1,3	2,7	1	2
6	3	Description	@	3	1	1,2	1,3	4,4	1	2
7	4	Description	@	4	1	1,2	1,3	6,1	1	2
8	5	Description	@	5	1	1,2	1,3	7,8	1	2
9	6	Description	@	6	1	1,2	1,3	9,5	1	2
10	7	Description	@	7	1	1,2	1,3	11,2	1	2
11	8	Description	@	8	1	1,2	1,3	12,9	1	2
12	9	Description	@	9	1	1,2	1,3	14,6	1	2
13	10	Description	@	10	1	1,2	1,3	16,3	1	2
14	11	Description	@	1	1	1,2	1,3	1,5	4	3
15	12	Description	@	0	1	1,2	1,3	2,7	4	3
16	13	Description	@	3	1	1,2	1,3	4,4	4	3
17	14	Description	@	4	1	1,2	1,3	6,1	4	3
18	15	Description	@	5	1	1,2	1,3	7,8	4	3
19	16	Description	@	6	1	1,2	1,3	9,5	4	3
20	17	Description	@	7	1	1,2	1,3	11,2	4	3
21	18	Description	@	8	1	1,2	1,3	12,9	4	3
22	19	Description	@	9	1	1,2	1,3	14,6	4	3
23	20	Description	@	10	1	1,2	1,3	16,3	4	3
24	21	Description	@	1	1	1,2	1,3	1,5	7	4
25	22	Description	@	0	1	1,2	1,3	2,7	7	4
26	23	Description	@	3	1	1,2	1,3	4,4	7	4
27	24	Description	@	4	1	1,2	1,3	6,1	7	4
28	25	Description	@	5	1	1,2	1,3	7,8	7	4
29	26	Description	@	6	1	1,2	1,3	9,5	7	4
30	27	Description	@	7	1	1,2	1,3	11,2	7	4
31	28	Description	@	8	1	1,2	1,3	12,9	7	4
32	29	Description	@	9	1	1,2	1,3	14,6	7	4
33	30	Description	@	10	1	1,2	1,3	16,3	7	4
34	31	Description	@	1	1	1,2	1,3	1,5	10	5
35	32	Description	@	0	1	1,2	1,3	2,7	10	5
36	33	Description	@	3	1	1,2	1,3	4,4	10	5
37	34	Description	@	4	1	1,2	1,3	6,1	10	5
38	35	Description	@	5	1	1,2	1,3	7,8	10	5
39	36	Description	@	6	1	1,2	1,3	9,5	10	5
40	37	Description	@	7	1	1,2	1,3	11,2	10	5
41	38	Description	@	8	1	1,2	1,3	12,9	10	5
42	39	Description	@	9	1	1,2	1,3	14,6	10	5
43	40	Description	@	10	1	1,2	1,3	16,3	10	5
44										

Log file

Each time the reading function block is restarted, the log file is rewritten and the old contents are deleted. If there is no log file, it will be automatically created first. The log file then contains either an OK message or a list of all errors that have occurred. Errors connected with the opening, writing or closing of the log file itself

cannot be written at the same time. Therefore, always note the output *sErrDescr* of the reading function block that indicates the last error code. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.

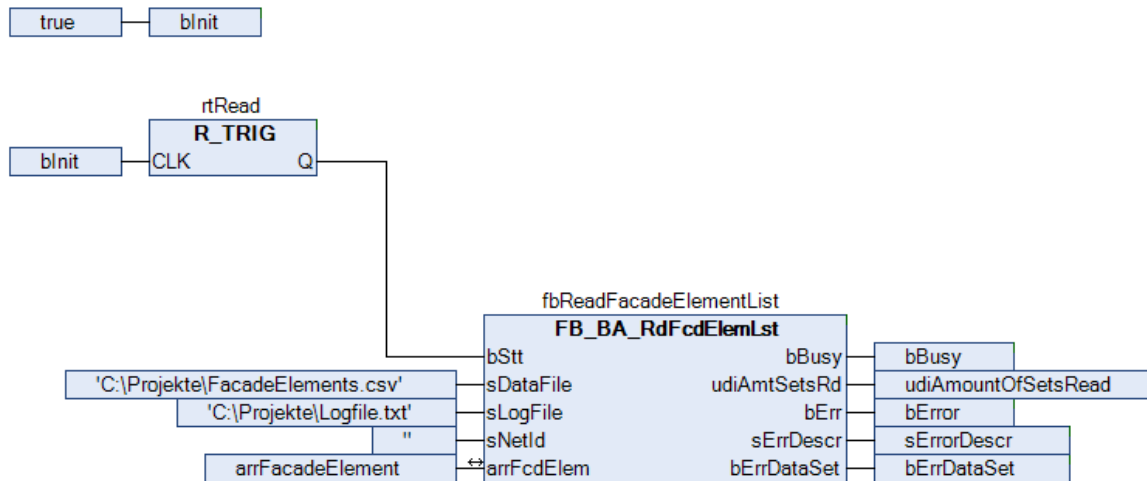
Program sample

```

PROGRAM ReadFacadeElements
VAR
    bInit          :   BOOL;
    rtRead         :   R_TRIG;
    fbReadFacadeElementList : FB_BA_RdFcdElemLst;
    arrFacadeElement : ARRAY [1..uiMaxColumnFcd, 1..uiMaxRowFcd] OF ST_BA_FcdElem;

    bBusy          :   BOOL;
    udiAmountOfSetsRead : UDINT;
    bError         :   BOOL;
    sErrorDescr    :   T_MaxString;
    bErrDataSet    :   BOOL;
END_VAR

```



In this sample the variable *bInit* is initially set to TRUE when the PLC starts. Hence, the input *bStt* of the function block *fbReadFacadeElementList* receives a once-only rising edge that triggers the reading process. The file "FacadeElements.csv" is read, which is located in the folder "C:\Projects\". The log file "Logfile.txt" is then saved in the same folder. If this log file does not yet exist it will be created, otherwise the existing contents are overwritten. Reading and writing take place on the same computer on which the PLC is located. This is defined by the input *sNetID* = "" (=local). All data are written to the list *arrFcdElem* declared in the program. During reading and writing the output *bBusy* is set to TRUE. The last file handling error that occurred is displayed at *sErrDescr*; *bErr* is TRUE. If an error is detected in the data set, this is displayed at *bErrDataSet* and described in more detail in the log file. The number of found and read data rows is displayed at *udiAmtSetsRd* for verification purposes.

The errors marked were "built into" the following Excel list. This gives rise to the log file shown:

	A	B	C	D	E	F	G	H	I	J	K
1	Number	Description		IndexColumn (Axis)	IndexRow (Floor)	Window wid (m)	Window hei (m)	P1x (m)	P1y (m)	Group	
2											
3		Text									
4	1	Description @		1	1	1,2	1,3	1,5	1,4	2	
5	2	Description @		0	1	1,2	1,3	2,7	1	2	
6	3	Description @		3	1	1,2	1,3	4,4	1	2	
7	4	Description @		4	-1	1,2	1,3	6,1	1	2	
8	5	Description @		5	1	1,2	1,3	7,8	1	2	
9	6	Description @		6	1	0	1,3	9,5	1	2	
10	7	Description @		7	1	1,2	1,3	11,2	1	2	
11	8	Description @		8	1	1,2	1,3	12,9	1	2	
12	9	Description @		9	1	1,2	1,3	14,6	1	2	
13	10	Description @		10	1	1,2	1,3	16,3	1	5	
14	11	Description @		1	2	1,2	1,3	1	-1	5	
15	12	Description @		2	2	1,2	1,3	2,7	3	5	
16	13	Description @		3	2	1,2	1,3	4,4	4	5	
17	14	Description @		4	2	1,2	1,3	4,4	4	5	
18	15	Description @		5	2	1,2	0	7,8	4	5	
19	16	Description @		6	2	1,2	1,3	9,5	4	5	
20	17	Description @		7	2	1,2	1,3	11,2	4	5	
21	18	Description @		8	2	1,2	1,3	12,9	4	5	
22	19	Description @		9	2	1,2	1,3	14,6	4	3	
23	20	Description @		10	2	1,2	1,3	16,3	4	3	
24											
25	31	Description @		1	3	1,2	1,3	1	7	3	
26	32	Description @		2	3	1,2	1,3	-1	6	3	
27	33	Description @		3	3	1,2	1,3	4,4	7	3	
28	34	Description @		4	3	1,2	1,3	6,1	7	0	
29	35	Description @		5	3	1,2	1,3	7,8	7	3	
30	36	Description @		6	3	1,2	1,3	9,5	7	3	
31	37	Description @		7	3	1,2	1,3	11,2	7	3	
32	38	Description @		8	3	1,2	1,3	12,9	7	7	
33	39	Description @		9	3	1,2	1,3	14,6	7	7	
34	40	Description @		10	3	1,2	1,3	16,3	7	7	

LogFacade.txt - Editor

Datei Bearbeiten Format Ansicht ?

Index-Error in Data-Set #2
 Index-Error in Data-Set #4
 Validation-Error in Data-Set #6, Errordescription:05: Error: The window width is less than or equal to zero.
 Validation-Error in Data-Set #11, Errordescription:04: Error: The Y-component of the first corner (Corner1) is less than zero.
 Validation-Error in Data-Set #15, Errordescription:06: Error: The window height is less than or equal to zero.
 Validation-Error in Data-Set #22, Errordescription:03: Error: The X-component of the first corner (Corner1) is less than zero.
 Validation-Error in Data-Set #24, Errordescription:02: Error: The group index is 0, but at the same time another entry of the facade element is not zero. See manual for this FB.

The first error is in data set 2 and is an index error, since "0" is not permitted.

The next error in data set 6 was found after validation of the data with the internally used function block **FB BA_ShdObjEntry** [► 501] and allocated an error description. The third and the fourth errors likewise occurred after the internal validation.



Important here is that the data set numbers (in this case 22 and 24) do not go by the numbers entered in the list, but by the actual sequential numbers: only 30 data sets were read in here.

VAR_INPUT

```
bStt      : BOOL;
sDataFile : STRING;
sLogFile  : STRING;
sNetId    : T_AmsNetId;
```

bStt: A TRUE edge on this input starts the reading process.

sDataFile: Contains the path and file name for the data file to be opened. This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)". If the file is opened with a simple text editor, the values must be separated by semicolons. Example of an entry: *sDataFile:= 'C:\Projekte\FacadeElements.csv'*

sLogFile: ditto. Log file for the accumulating errors. This file is overwritten each time the function block is activated, so that only current errors are contained.

sNetId: A string can be entered here with the AMS Net ID of the TwinCAT computer on which the files are to be written/read. An empty string can be specified for the local computer (see T_AmsNetId).



The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

VAR_OUTPUT

```
bBusy      : BOOL;
udiAmtSetsRd : UDINT;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
bErrDataSet : BOOL;
```

bBusy: This output is TRUE as long as elements are being read from the file.

udiAmtSetsRd: Number of data sets read.

bErr: This output is switched to TRUE, if a file write or read error has occurred.

sErrDescr: Contains the error description.

Error description
01: File handling error: Opening the log file - the ADS error number is stated.
02: File handling error: Open the data file - the ADS error number is stated.
03: File handling error: Reading the data file - the ADS error number is stated.
04: Error: During reading of the data file it was determined that the file is too large (number of bytes larger than udiMaxDataFileSize)
05: File handling error: Writing to the log file - the ADS error number is stated.
06: File handling error: Closing the data file - the ADS error number is stated.
07: File handling error: Writing to the log file (OK message if no errors were detected) - the ADS error number is stated.
08: File handling error: Closing the log file - the ADS error number is stated.

bErrDataSet: This output is set to TRUE, if the read data sets are faulty. Further details are entered in the log file.

VAR_IN_OUT

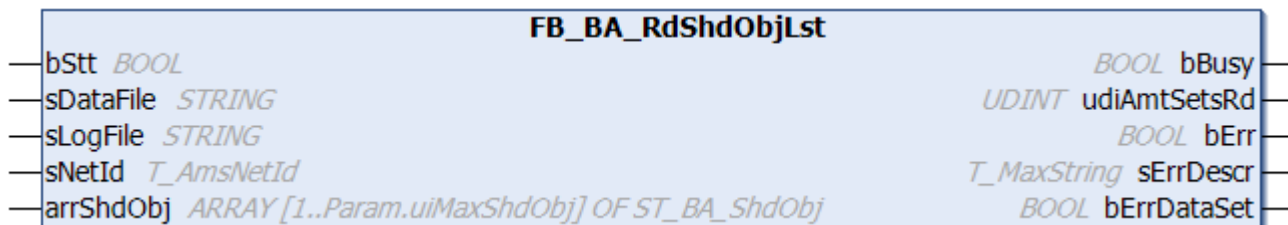
```
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: [List of facade elements](#) [[► 474](#)].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.13 FB_BA_RdShdObjLst



With the help of this function block, data for shading objects can be imported from a pre-defined Excel table in csv format into the [list of shading objects](#) [[► 474](#)]. In addition the imported data are checked for plausibility and errors are written to a log file.

The following example shows the Excel table with the entries of the window elements.

All text fields are freely writable. The fields marked in green are important; each line in these fields identifies a data set. The columns G to J have a different meaning depending on whether the type rectangle or ball is concerned. The columns K to M are to be left empty in the case of balls. With regard to the rectangle coordinates, only the relevant data are entered and the remainder are internally calculated, see FB BA_ShdObjEntry [► 501].

The following rules are to be observed:

- A data set must always start with a '@'.
- The monthly entries must not be 0 or greater than 12; all other combinations are possible.

Examples:

Start=1, End=1: shading in January.

Start=1, End=5: shading from the beginning of January to the end of May.

Start=11, End=5: shading from the beginning of November to the end of May (the following year).

- Window width and window height must be greater than zero
- The z-coordinates P1z and P3z or Mz must be greater than zero.
- The radius must be greater than zero.
- For system-related reasons the total size of the table may not exceed 65534 bytes.
- This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)".

Is not necessary to describe all shading objects that are possible per facade. Only those contained in the list ultimately take effect.

DE_ShadingObjects.csv													
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Number	Description		Type	Begin	End	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Tetragon	(Month)	(Month)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Globe									
4		Text											
5	1	Description	@	0	1	2	-94,75	0	36,06	11	-70,71	11	68,59
6	2	Description	@	0	1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Description	@	0	1	2	62,23	0	0	14,47	62,23	14,47	8
8	4	Description	@	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Description	@	0	1	2	46	0	13	14,47	46	14,47	38,89
10	6	Description	@	0	1	2	0	0	14	9	35	9	14
11	7	Description	@	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Description	@	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Description	@	0	1	2	27,8	0	14	9,8	35	9,8	14
14													
15	10	Description	@	1	1	2	27	15	40	6			
16	11	Description	@	1	1	2	38	15	36	6			
17	12	Description	@	1	1	2	-14	4	4	1,5			
18	13	Description	@	1	1	2	-6,5	6	6	3,2			
19	14	Description	@	1	1	2	-7	9	6	1,2			
20	15	Description	@	1	1	2	-1	6	8	3,2			
21	16	Description	@	1	1	2	-1	9	8	1,2			

Log file

Each time the reading function block is restarted, the log file is rewritten and the old contents are deleted. If there is no log file, it will be automatically created first. The log file then contains either an OK message or a list of all errors that have occurred. Errors connected with the opening, writing or closing of the log file itself cannot be written at the same time. Therefore, always note the output *sErrDescr* of the reading function block that indicates the last error code. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.

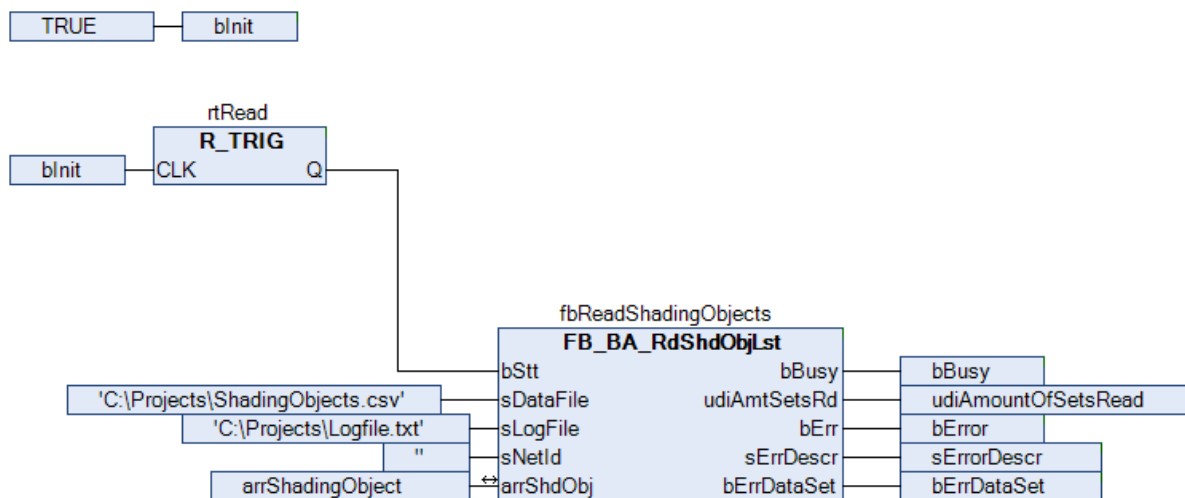
Program sample

```

PROGRAM ReadShadingObjects
VAR
    bInit          :   BOOL;
    rtRead         :   R_TRIG;
    fbReadShadingObjects : FB_BA_RdShdObjLst;
    arrShadingObject : ARRAY [1..uiMaxShdObj] OF ST_BA_ShObj;

    bBusy          :   BOOL;
    udiAmountOfSetsRead : UDINT;
    bError         :   BOOL;
    sErrorDescr    :   T_MaxString;
    bErrDataSet    :   BOOL;
END_VAR

```



In this sample the variable *bInit* is initially set to TRUE when the PLC starts. Hence, the input *bStt* on the function block *fbReadShadingObjects* receives a once-only rising edge that triggers the reading process. The file "ShadingObjects.csv" is read, which is located in the folder "C:\Projects\". The log file "Logfile.txt" is then saved in the same folder. If this log file does not yet exist it will be created, otherwise the existing contents are overwritten. Reading and writing take place on the same computer on which the PLC is located. This is defined by the input *sNetID* = "" (=local). All data are written to the list *arrShdObj* declared in the program. During reading and writing the output *bBusy* is set to TRUE. The last file handling error that occurred is displayed at *sErrDescr*; *bErr* is TRUE. If an error is detected in the data set, this is displayed at *bErrDataSet* and described in more detail in the log file. The number of found and read data rows is displayed at *udiAmtSetsRd* for verification purposes.

The errors marked were built into the following Excel list. This gives rise to the log file shown:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Number	Description		Type	Begin	End	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Tetragon	(Month)	(Month)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Globe									
4		Text											
5	1	Description	@	2	1	2	-94,75	-4	36,06	11	-70,71	11	68,59
6	2	Description	@	0	-1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Description	@	0	1	13	62,23	0	0	14,47	62,23	14,47	8
8	4	Description	@	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Description	@	0	1	2	46	0	-13	14,47	46	14,47	38,89
10	6	Description	@	0	1	2	0	0	14	9	35	9	-14
11	7	Description	@	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Description	@	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Description	@	0	1	2	27,8	0	14	9,8	35	9,8	14
14													
15	11	Description	@	1	1	2	27	15	-40	6			
16	12	Description	@	1	1	2	38	15	36	6			
17	13	Description	@	1	1	2	-14	4	4	1,5			
18	14	Description	@	1	1	2	-6,5	6	6	3,2			
19	15	Description	@	1	1	2	-7	9	6	1,2			
20	16	Description	@	1	1	2	-1	6	8	3,2			
21	17	Description	@	1	1	2	-1	9	8	1,2			

The first error is in data set 3 and is a type error, since "2" is not defined.

The next error in data set 6 was found after validation of the data with the internally used function block `FB_BA_ShdObjEntry` [► 501] and allocated an error description. The third error likewise occurred after the internal validation.



Important here it that the data set number (in this case 11) does not go by the number entered in the list, but by the actual sequential number: only 16 data sets were read in here.

VAR_INPUT

```
bStt      : BOOL;
sDataFile : STRING;
sLogFile  : STRING;
sNetId    : T_AmsNetId;;
```

bStt: A TRUE edge on this input starts the reading process.

sDataFile: Contains the path and file name for the data file to be opened. This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)". If the file is opened with a simple text editor, the values must be separated by semicolons. Example of an entry: `sDataFile:= 'C:\Projects\ShadingObjects.csv'`

sLogFile: ditto. Log file for the accumulating errors. This file is overwritten each time the function block is activated, so that only current errors are contained.

sNetId: A string can be entered here with the AMS Net ID of the TwinCAT computer on which the files are to be written/read. An empty string can be specified for the local computer (see `T_AmsNetId`).



The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

VAR_OUTPUT

```
bBusy      : BOOL;
udiAmtSetsRd : UDINT;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
bErrDataSet : BOOL;
```

bBusy: This output is TRUE as long as elements are being read from the file.

udiAmtSetsRd: Number of data sets read.

bErr: This output is switched to TRUE, if a file write or read error has occurred.

sErrDescr: Contains the error description.

Error description
01: File handling error: Opening the log file - the ADS error number is stated.
02: File handling error: Open the data file - the ADS error number is stated.
03: File handling error: Reading the data file - the ADS error number is stated.
04: Error: During reading of the data file it was determined that the file is too large (number of bytes larger than udiMaxDataFileSize)
05: File handling error: Writing to the log file - the ADS error number is stated.
06: File handling error: Closing the data file - the ADS error number is stated.
07: File handling error: Writing to the log file (OK message if no errors were detected) - the ADS error number is stated.
08: File handling error: Closing the log file - the ADS error number is stated.

bErrDataSet: This output is set to TRUE, if the read data sets are faulty. Further details are entered in the log file.

VAR_IN_OUT

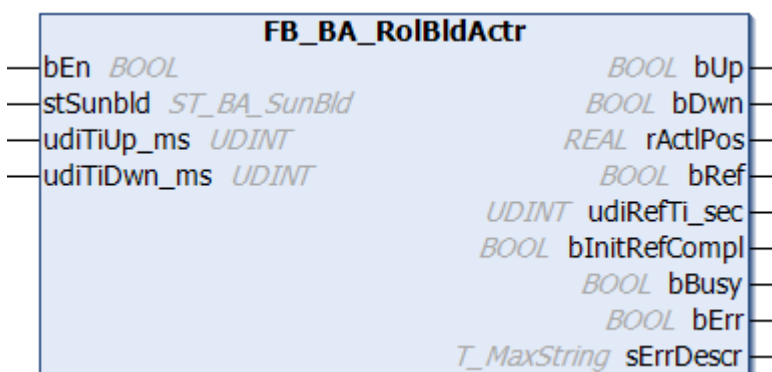
```
arrShdObj: ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShObj;
```

arrShdObj: List of shading objects [► 474].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.14 FB_BA_RolBldActr



This function block is used to position a roller shutter over two outputs: up and down. The positioning telegram `stSunBld` [► 602] can be used to move the roller shutter to any position. In addition, the positioning telegram `stSunBld` [► 602] offers manual commands, which can be used to move the roller shutter to particular positions. These manual commands are controlled by the function block `FB_BA_SunBldSwi` [► 517].

Structure of the blind positioning telegram `stSunBld` [► 602].

```
TYPE ST_BA_SunBld:
STRUCT
```

```

    rPos      : REAL;
    rAngl     : REAL;
    bManUp    : BOOL;
    bManDwn   : BOOL;
    bManMod   : BOOL;
```

```

    bActv      : BOOL;
END_STRUCT
END_TYPE

```

The current height position is not read in by an additional encoder; it is determined internally by the runtime of the roller shutter.

The two runtime parameters *udiTiUp* (roller shutter travel-up time [ms]) and *udiTiDwn* (roller shutter travel-down time [ms]) take account of the different movement characteristics.

As a rule, the function block controls the roller shutter based on the information from the positioning telegram *stSunBld* [► 602]. If automatic mode is active (*bManMod*=FALSE), the roller shutter always moves to the current position; changes are reflected immediately. In manual mode (*bManMod*=TRUE), the roller shutter is controlled by the commands *bManUp* and *bManDwn*.

Referencing

Safe referencing refers to a situation when the roller shutter is upwards-controlled for longer than its complete travel-up time. The position is then always "0". Since roller shutter positioning without encoder is always error-prone, it is important to use automatic referencing whenever possible: Whenever "0" is specified as the target position, the roller shutter initially moves upwards normally, based on continuous position calculation. Once the calculated position value 0% is reached, the output *bUp* continues to be held for the complete travel-up time + 5s.

For reasons of flexibility there are now two possibilities to interrupt the referencing procedure: Until the calculated 0% position is reached, a change in position continues to be assumed and executed. Once this 0% position is reached, the roller shutter can still be moved with the manual "travel-down" command. These two sensible restrictions make it necessary for the user to ensure that the roller shutter is referenced safely whenever possible.

After a system restart, the function block executes a reference run. Completion of the initial referencing is indicated through a TRUE signal at output *bInitRefCmpl*. The initial referencing can also be terminated through a manual "travel-down" command.

VAR_INPUT

```

bEn      : BOOL;
stSunBld : ST_BA_Sunblind;
udiTiUp_ms : UDINT;
udiTiDwn_ms : UDINT;

```

bEn: Enable input for the function block. As long as this input is TRUE, the actuator function block accepts and executes commands as described above. A FALSE signal on this input resets the control outputs *bUp* and *bDwn* and the function block remains in a state of rest.

stSunBld: Positioning telegram, see *ST_BA_SunBld* [► 602].

udiTiUp_ms: Complete time for driving up [ms].

udiTiDwn_ms: Complete time for driving down in ms.

VAR_OUTPUT

```

bUp      : BOOL;
bDwn     : BOOL;
rActlPos : REAL;
bRef     : BOOL;
udiRefTi_sec : UDINT;
bInitRefCmpl : BOOL;
bBusy    : BOOL;
bErr     : BOOL;
sErrDescr : STRING;

```

bUp: Roller shutter control output up.

bDwn: Roller shutter control output down.

rActlPos: Current position in percent.

bRef: The roller shutter is in referencing mode, i.e. the output *bUp* is set for the complete travel-up time + 5s. Only a manual "down" command can move the roller shutter in the opposite direction and terminate this mode.

udiRefTi_sec: Referencing countdown display [s].

blnitRefCompl: Initial referencing process complete.

bBusy: A positioning or a referencing procedure is in progress.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

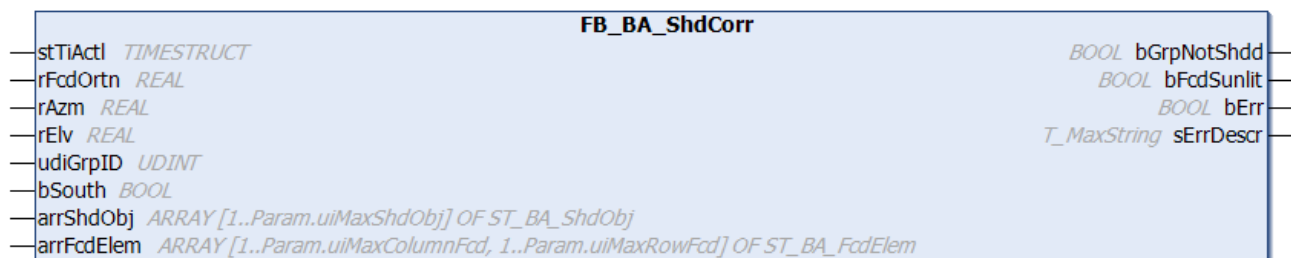
sErrDescr: Contains the error description.

Error description
01: Error: The total travel-up or travel-down time (udiTiUp_ms/udiTiDwn_ms) is zero.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.15 FB_BA_ShdCorr



The function block is used to assess the shading of a group of windows on a facade.

The function block FB_BA_ShdCorr calculates whether a window group lies in the shadow of surrounding objects. The result, which is output at the output `bGrpNotShdd`, can be used to judge whether sun shading makes sense for this window group.

The function block thereby accesses two lists, which are to be defined:

- The parameters that describe the shading elements that are relevant to the facade on which the window group is located. This [list of shading objects](#) [► 474] is used as input variable `arrShdObj` for the function block, since the information is read only.
- The data of the elements (window) of the facade in which the group to be regarded is located. This [list of facade elements](#) [► 474] is accessed via the IN/OUT variable `arrFcdElem`, since not only the window coordinates are read, but the function block FB_BA_ShdCorr also stores the shading information for each window corner in this list. In this way, the information can also be used in other parts of the application program.

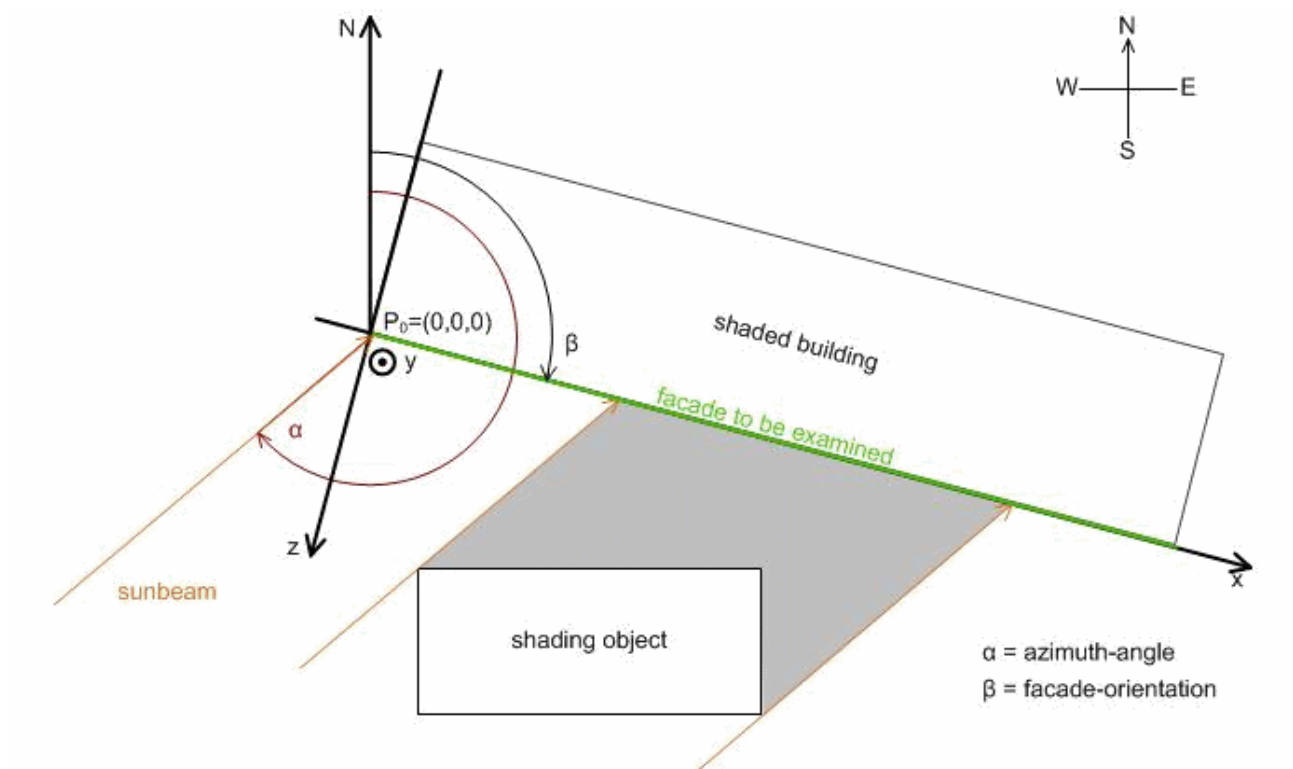
On the basis of the facade orientation (`rFcdOrtn`), the direction of the sun (`rAzm`) and the height of the sun (`rElv`), a calculation can be performed for each corner of a window to check whether this lies in a shaded area. A window group is considered to be completely shaded if all corners are shaded.

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):

Line of sight	Facade orientation
North	$\beta=0^\circ$
East	$\beta=90^\circ$
South	$\beta=180^\circ$
West	$\beta=270^\circ$

The function block performs its calculations only if the sun is actually shining on the facade. Considering the drawing presented in the introduction, this is the case if:

Facade orientation < azimuth angle < facade orientation + 180°

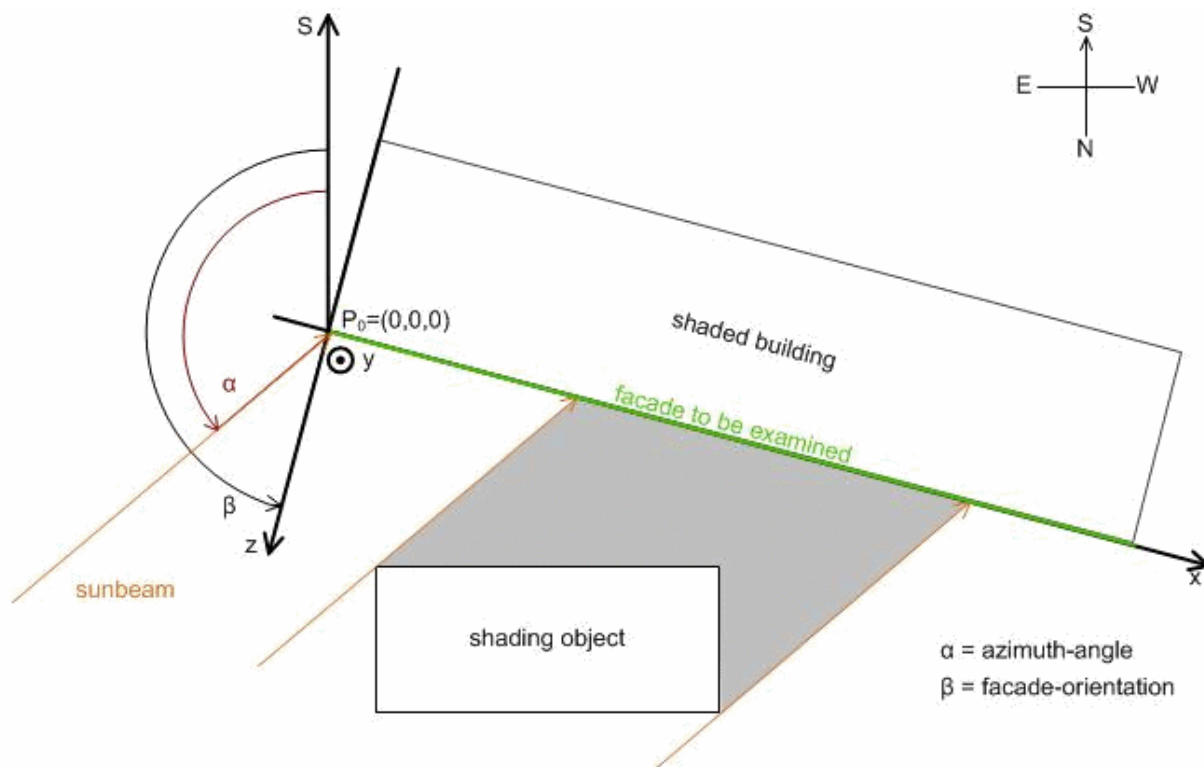


In addition, a calculation is also not required, if the sun has not yet risen, i.e. the solar elevation is below 0° . In both cases the output *bFcdSunlit* is set to FALSE.

The situation is different for the southern hemisphere. The following applies to the facade orientation (looking out the window):

Line of sight	Facade orientation
South	$\beta = 0^\circ$
East	$\beta = 90^\circ$
North	$\beta = 180^\circ$
West	$\beta = 270^\circ$

The internal calculation or the relationship between facade and sunbeam also changes:



To distinguish between the situation in the northern and southern hemisphere, set the input parameter *bSouth* to FALSE (northern hemisphere) or TRUE (southern hemisphere)

VAR_INPUT

```
stTiActl   : TIMESTRUCT;
rFcdOrtn   : REAL;
rAzm       : REAL;
rElv       : REAL;
diGrpID    : DINT;
bSouth     : BOOL;
arrShdObj  : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShObj;
```

stTiActl: Input of the current time - local time in this case, since this time takes into account the shaded months. If the UTC time (or GMT) is used, the month may change in the middle of the day, depending on the location on the earth (see TIMESTRUCT).

rFcdOrtn: Facade orientation, see illustration above.

rAzm: Direction of the sun at the time of observation [°].

rElv: Solar altitude at the time of observation [°].

diGrpId: Window group regarded. The group 0 is reserved here for unused window elements, see [FB_BA_FcdElemEntry](#) [► 478]. A 0-entry would lead to an error output (*bErr*=TRUE). The function block is then not executed any further and *bGrpNotShdd* is set to FALSE.

bSouth: FALSE: Calculations refer to conditions in the northern hemisphere - TRUE: In the southern hemisphere

arrShdObj: [List of shading objects](#) [► 474].

VAR_OUTPUT

```
bGrpNotShdd : BOOL;
bFcdSunlit  : BOOL;
bErr        : BOOL;
sErrDescr   : T_MAXSTRING
```

bGrpNotShdd: Is TRUE as long as the window group is not calculated as shaded.

bFcdSunlit: This output is set to TRUE if the sun is shining on the facade. See description above.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description
01: Error: The index of the window group <code>usiGrpId</code> under consideration is 0.
02: Error: An element of the facade list is invalid. This is specified in the error description <code>sErrDescr</code> as <code>arrFcdElem[nColumn,nRow]</code> .

VAR_IN_OUT

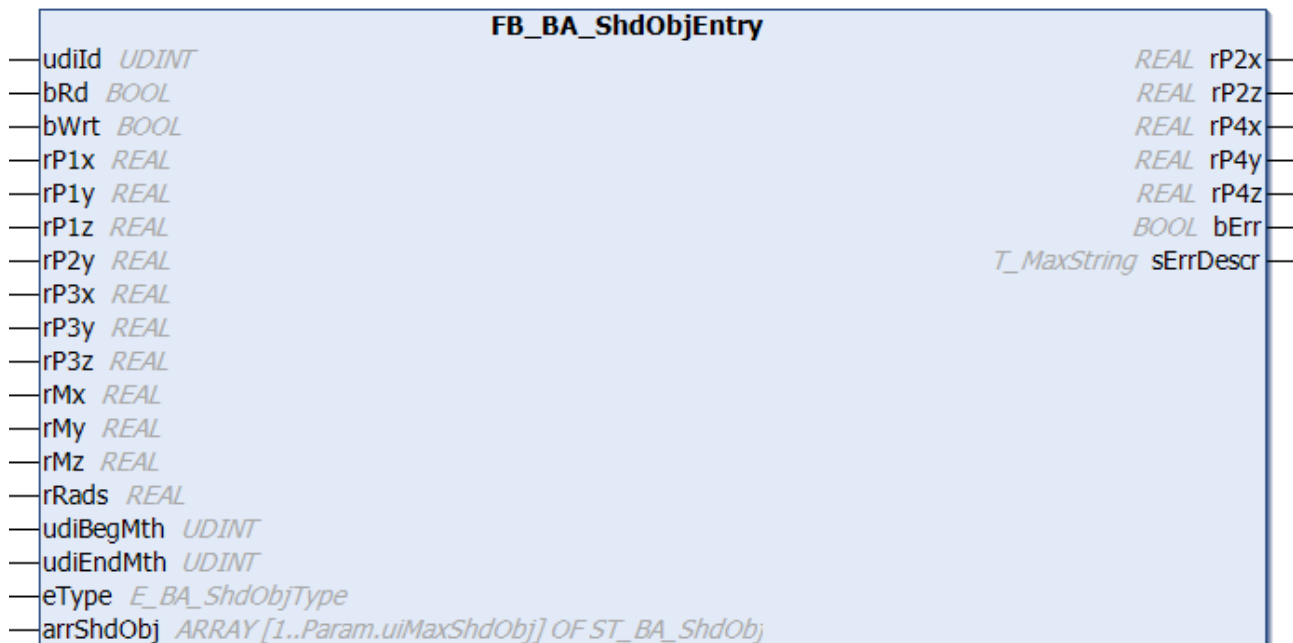
```
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: List of facade elements [[► 474](#)].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.16 FB_BA_ShObjEntry



This function block serves for the administration of all shading elements in a facade, which are globally saved in a [list of shading elements](#) [[► 474](#)]. It is intended to facilitate the input of the element information - also with regard to the use of the visualization. A schematic representation of the objects with description of the coordinates is shown in [Shading correction: Principles and definitions](#) [[► 459](#)].

The shading elements are declared in the global variables:

```
VAR_GLOBAL
    arrShdObj : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShObj;
END_VAR
```

Each individual element `arrShdObj[1]` to `arrShdObj[uiMaxShdObj]` carries the information for one shading element (ST_BA_ShObj [[► 601](#)]). This information consists of the selected type of shading (rectangle or sphere) and the respectively associated coordinates. For a rectangle, these are the corner points (`rP1x`, `rP1y`, `rP1z`), (`rP2x`, `rP2y`, `rP2z`), (`rP3x`, `rP3y`, `rP3z`) and (`rP4x`, `rP4y`, `rP4z`) for a sphere this are the center point (`rMx`, `rMy`, `rMz`) and the radius `rRads`. In addition, the phase of the shading can be defined via the inputs `udiBegMth` and `udiEndMth`, which is important in the case of objects such as trees that bear no foliage in winter.

The function block thereby directly accesses the field of this information via the IN-OUT variable `arrShdObj`.

Note: The fact that the rectangle coordinates $rP2x$, $rP2z$, $rP4x$, $rP4y$, and $rP4z$ are output values results from the fact that they are formed from the input parameters:

$rP2x = rP1x$; $rP2z = rP1z$; $rP4x = rP3x$; $rP4y = rP1y$; $rP4z = rP3z$;

That limits the input of a rectangle to the extent that the lateral edges stand vertically on the floor ($rP2x = rP1x$ and $rP4x = rP3x$), that the rectangle has no inclination ($rP2z = rP1z$ and $rP4z = rP3z$) and can only have a different height "upwards", i.e. in the positive y-direction ($rP4y = rP1y$).

The function block is used in three steps:

- Read
- Change
- Write

Read

Selection of the element from the list *arrShdObj[iId]* is based on the entry at *udiId*. A rising edge on *bRd* reads the data. These values are assigned to the input and output variables of the function block. These are the input values $rP1x$, $rP1y$, $rP1z$, $rP2y$, $rP3x$, $rP3y$, $rP3z$, rMx , rMy , rMz , $rRads$ and the object enumerator *eType* and the output values $rP2x$, $rP2z$, $rP4x$, $rP4y$ and $rP4z$. It is important here that the input values are not overwritten in the reading step. Hence, all values can initially be displayed in a visualization.

Change

In a next program step the listed input values can then be changed. If a rectangle is preselected at input *eType* [► 598] via the value "*eObjectTypeTetragon*", the output values $rP2x$, $rP2z$, $rP4x$, $rP4y$ and $rP4z$ result from the rectangle coordinates that were entered (see above).

The values entered are constantly checked for plausibility. The output *bErr* indicates whether the values are valid (*bErr*=FALSE). If the value is invalid, a corresponding error message is issued at output *sErrDescr*. If a rectangle is defined, only the inputs $rP1x$, $rP1y$, $rP1z$, $rP2y$, $rP3x$, $rP3y$ and $rP3z$ have to be described; the inputs rMx , rMy , rMz and $rRads$ do not have to be linked. For a sphere definition, only rMx , rMy , rMz and $rRads$ have to be described; the rectangle coordinates can remain unlinked

Write

The parameterized data are written to the list element with the index *udiId* upon a rising edge on *bWrt*, regardless of whether they represent valid values or not. The element structure *ST_BA_ShdObj* [► 601] therefore contains a plausibility bit *bVld*, which forwards precisely this information to the function block *FB_BA_ShdCorr* [► 498] to prevent miscalculations.

This approach is to be regarded only as a proposal. It is naturally also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on *bWrt*.

VAR_INPUT

```

udiId      : UDINT;
bRd        : BOOL;
bWrt       : BOOL;
rP1x       : REAL;
rP1y       : REAL;
rP1z       : REAL;
rP2y       : REAL;
rP3x       : REAL;
rP3y       : REAL;
rP3z       : REAL;
rMx        : REAL;
rMy        : REAL;
rMz        : REAL;
rRads      : REAL;
udiBegMth  : UDINT;
udiEndMth  : UDINT;
eType      : E_BA_ShdObjType;

```

udiId: Index of the selected element. This refers to the selection of a field element of the array saved in the IN-OUT variable *arrShdObj*. The variable *udiId* **must not be zero!** This is due to the field definition, which starts with 1. However, an incorrect input is recognized and displayed as such at *bErr/sErrDescr*.

bRd: The information of the selected element, *arrShdObj[udiId]*, is read into the function block with a positive edge at this input and assigned to the input variables *rP1x* to *eType* and the output variables *rP2x* to *rP4z*. If data are already present on the inputs *rP1x* to *eType* at this time, then the data previously read are immediately overwritten with these data.

bWrt: A positive edge writes the values applied to the inputs *rP1x* to *eType* and the values determined and assigned to the outputs *rP2x* to *rP4z* to the selected field element *arrShdObj[udiId]*.

rP1x: X-coordinate of point 1 of the shading element (rectangle) [m].

rP1y: Y-coordinate of point 1 of the shading element (rectangle) [m].

rP1z: Z-coordinate of point 1 of the shading element (rectangle) [m].

rP2y: Y-coordinate of point 2 of the shading element (rectangle) [m].

rP3x: X-coordinate of point 3 of the shading element (rectangle) [m].

rP3y: Y-coordinate of point 3 of the shading element (rectangle) [m].

rP3z: Z-coordinate of point 3 of the shading element (rectangle) [m].

rMx: X-coordinate of the center of the shading element (ball) [m].

rMy: Y-coordinate of the center of the shading element (ball) [m].

rMz: Z-coordinate of the center of the shading element (ball) [m].

rRads: Radius of the shading element (ball) [m].

udiBegMth: Beginning of the shading period (month).

udiEndMth: End of the shading period (month).

eType: Selected type of element: Rectangle or sphere (see [E BA_ShdObjType](#) [► 598]).

Remark about the shading period:

The entries for the months may not be 0 or greater than 12, otherwise all combinations are possible.

Examples:

Start=1, End=1: shading in January.

Start=1, End=5: shading from the beginning of January to the end of May.

Start=11, End=5: shading from the beginning of November to the end of May (the following year).

VAR_OUTPUT

```
rP2x      : REAL;
rP2z      : REAL;
rP4x      : REAL;
rP4y      : REAL;
rP4z      : REAL;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
```

rP2x: Calculated X-coordinate of point 2 of the shading element (rectangle) [m]. See "[Note \[► 501\]](#)" above.

rP2z: Calculated Z-coordinate of point 2 of the shading element (rectangle) [m]. See "[Note \[► 501\]](#)" above.

rP4x: Calculated X-coordinate of point 4 of the shading element (rectangle) [m]. See "[Note \[► 501\]](#)" above.

rP4y: Calculated Y-coordinate of point 4 of the shading element (rectangle) [m]. See "[Note \[► 501\]](#)" above.

rP4z: Calculated Z-coordinate of point 4 of the shading element (rectangle) [m]. See "[Note \[► 501\]](#)" above.

bErr: Result of the plausibility check for the values entered. For a rectangle, the internal angle is 360° and the points are in a plane *in front of* the facade under consideration. In the case of a ball the center must likewise lie in front of the facade and the radius must be greater than zero.

sErrDescr: Contains the error description.

Error description
01: Error: The input <i>udiId</i> is outside the permissible limits 1.. <i>uiMaxShdObj</i> .
02 Error: The sum of the angles of the rectangle is not 360°. This means that the corners are not in the order P1, P2, P3 and P4 but rather P1, P3, P2 and P4. This results in a crossed-over rectangle.
03: Error: The corners of the square are not on the same level.
04: Error: The z-component of P1 is less than zero. This corner would thus lie behind the facade.
05: Error: The z-component of P3 is less than zero. This corner would thus lie behind the facade.
06: Error: P1 is equal to P2. The object entered is thus not a rectangle.
07: Error: P1 is equal to P3. The object entered is thus not a rectangle.
08: Error: P1 is equal to P4. The object entered is thus not a rectangle.
09: Error: P2 is equal to P3. The object entered is thus not a rectangle.
10: Error: P2 is equal to P4. The object entered is thus not a rectangle.
11: Error: P3 is equal to P4. The object entered is thus not a rectangle.
12: Error: The radius entered is zero.
13: Error: The z-component of the ball center is less than zero. This point would thus lie behind the facade.
14: Error: Error object type <i>eType</i> - neither rectangle nor ball.
15: Error: Month input error.

VAR_IN_OUT

```
arrShdObj : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;
```

arrShdObj: List of shading objects [► 474].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.17 FB_BA_SunBldActr

FB_BA_SunBldActr	
— bEn <i>BOOL</i>	<i>BOOL</i> bUp —
— stSunbld <i>ST_BA_SunBld</i>	<i>BOOL</i> bDwn —
— udiTiUp_ms <i>UDINT</i>	<i>REAL</i> rActlPos —
— udiTiDwn_ms <i>UDINT</i>	<i>REAL</i> rActlAngl —
— udiTurnTiUp_ms <i>UDINT</i>	<i>BOOL</i> bRef —
— udiTurnTiDwn_ms <i>UDINT</i>	<i>UDINT</i> udiRefTi_sec —
— udiBckLshTiUp_ms <i>UDINT</i>	<i>BOOL</i> bInitRefCompl —
— udiBckLshTiDwn_ms <i>UDINT</i>	<i>BOOL</i> bBusy —
— rAnglLmtUp <i>REAL</i>	<i>BOOL</i> bErr —
— rAnglLmtDwn <i>REAL</i>	<i>T_MaxString</i> sErrDescr —

This function block is used for positioning of a louvered blind via two outputs: drive up and drive down. The blind can be driven to any desired (height) position and louvre angle via the positioning telegram stSunBld [► 602]. On top of that, the positioning telegram stSunBld [► 602] also contains manual commands with which the blind can be moved individually to certain positions. These manual commands are controlled by the function block FB_BA_SunBldSwi [► 517].

Structure of the blind positioning telegram stSunBld [► 602].

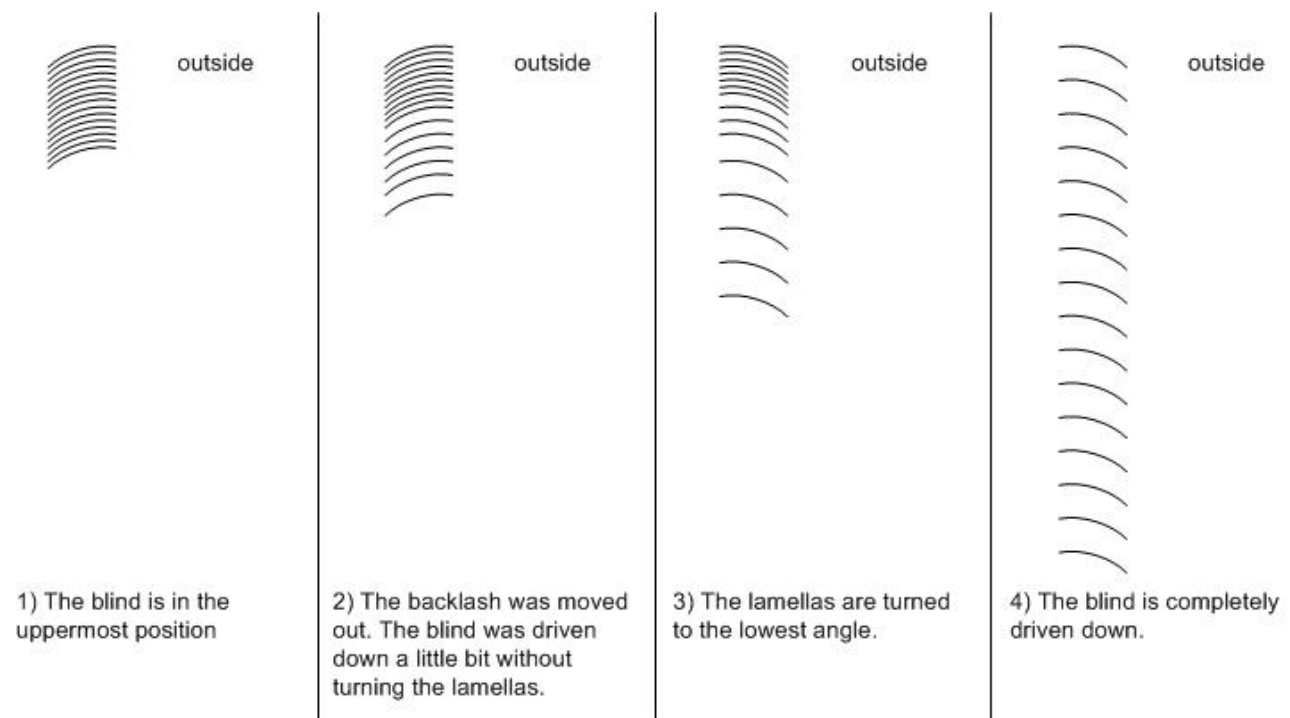
```

TYPE ST_BA_SunBld:
STRUCT
    rPos      : REAL;
    rAngl     : REAL;
    bManUp    : BOOL;
    bManDwn   : BOOL;
    bManMod   : BOOL;
    bActv     : BOOL;
END_STRUCT
END_TYPE

```

The current height position and the louvre angle are not read in by an additional encoder, but determined internally by the travel time of the blind. The calculation is based on the following travel profile (regarded from the highest and lowest position of the blind):

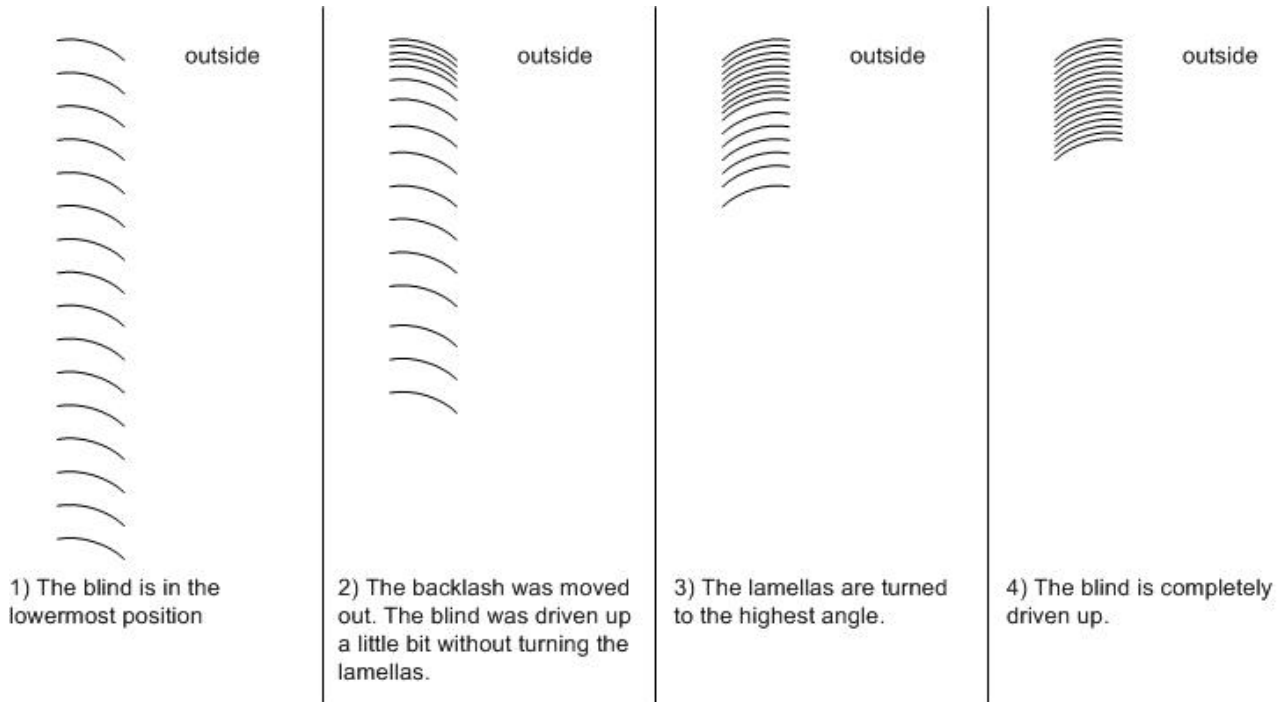
Downward travel profile:



More detailed explanations of the terms "backlash" and "turning" are given here in the downward movement:

The blind normally describes its downward movement with the louvre low point directed outwards, as in fig. 3).

If the blind is in an initial position with the low point directed inwards (i.e. after the conclusion of an upward movement), then a certain time elapses after a new downward movement begins before the louvres start to turn from the "inward low point" to the "outward low point". During this time the louvre angle does not change; the blind only drives downward (fig. 1 and fig. 2). This time is an important parameter for the movement calculation and is entered in the function block under *udiBckLshTiDwn_ms* [ms]. Since it is not known at any point after a blind movement of any length whether backlash has already taken effect, the backlash of the downward movement or its travel time can be measured most reliably if the blind was first raised fully. A further important parameter is the time interval of the subsequent turning of the louvres from the "Outward low point" to the "Inward low point". This time should be entered as *udiTurnTiDwn_ms* [ms] at the function block.

Upward travel profile:**More detailed explanations of the terms "backlash" and "turning" are given here in the upward movement:**

The circumstances are similar to the downward movement described above: The blind normally describes its upward movement with the louver low point directed inwards, as in fig. 3).

If the blind is in an initial position with the low point directed outwards (i.e. after the conclusion of a downward movement), then a certain time elapses after a new upward movement begins before the louvers start to turn from the "Outward low point" to the "Inward low point". During this time the louver angle does not change; the blind only drives upward (fig. 1 and fig. 2). Also this time is an important parameter for the movement calculation and is entered in the function block under *udiBckLshTiUp_ms* [ms]. Since it is not known at any point after a blind movement of any length whether backlash has already taken effect, the backlash of the upward movement or its travel time can be measured most reliably if the blind was first driven fully downward. A further important parameter is the time interval of the subsequent turning of the louvers from the "Outward low point" to the "Inward low point". This time should be entered as *udiTurnTiUp_ms* [ms] at the function block.

Parameterization

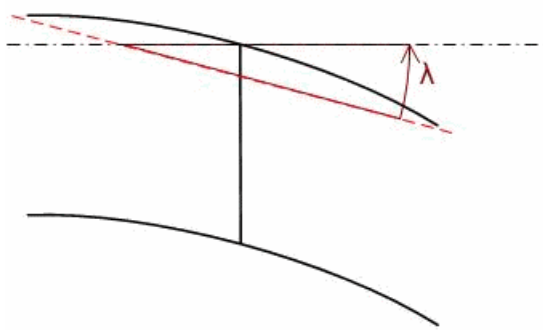
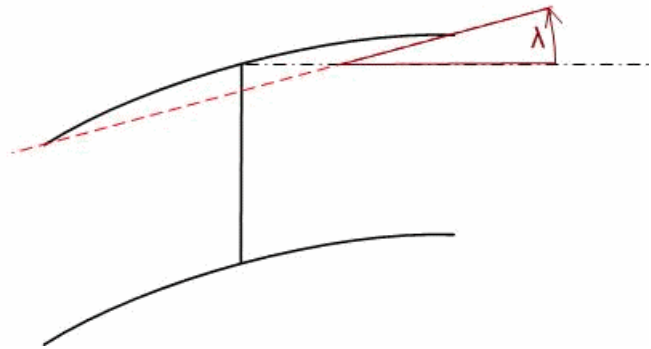
For the calculation of the (height) position and the louver angle, the following times now have to be determined for both the upward and downward movement:

- the travel time of the backlash (*udiBckLshTiUp_ms* / *udiBckLshTiDwn_ms* [ms])
- the turning duration (*udiTurnTiUp_ms* / *udiTurnTiDwn_ms* [ms])
- the total travel time (*udiTiUp_ms* / *udiTiDwn_ms* [ms])

Furthermore the following are required for the calculation:

- the highest louver angle after turning upwards (*rAnglLmtUp* [°])
- the lowest louver angle after turning downwards (*rAnglLmtDwn* [°])

The louver angle λ is defined by a notional straight line through the end points of the louver to the horizontal.

louvre angle $\lambda < 0$ louvre angle $\lambda > 0$ 

Functioning

As a rule, the function block controls the blind based on the information from the positioning telegram `stSunBld` [► 602]. If automatic mode is active (`bManMod=FALSE`), then the current position and louvre angle are always driven to, wherein changes are immediately accounted for. The height positioning takes priority: First the entered height and afterwards the louvre angle are driven to. For reasons of the simplicity the position error due to the angle movement is disregarded. In manual mode (`bManMod=TRUE`), the blind is controlled by the commands `bManUp` and `bManDwn`.

An automatic movement command is triggered whenever a change from manual to automatic mode occurs.

Referencing

Secure referencing is ensured if the blind is driven upward for longer than its complete drive-up time. The position is then in any case "0" and the louvre angle is at its maximum. Since blind positioning without an encoder is naturally always susceptible to error, it is important to automatically reference as often as possible: each time the "0" position is to be driven to (the angle is unimportant), the blind initially drives upward quite normally with continuous position calculation. Once the calculated position value 0% is reached, the output `bUp` continues to be held for the complete travel-up time + 5 s.

For reasons of flexibility, there are two ways to interrupt the referencing process: Until the calculated 0% position is reached, a change in position continues to be assumed and executed. Once this 0% position is reached, the blind can still be moved with the manual "travel-down" command. These two sensible limitations make it necessary for the user to ensure that the blind is securely referenced as often as possible.

After a system restart, the function block executes a reference run. Completion of the initial referencing is indicated through a TRUE signal at output `bInitRefCmpl`. The initial referencing can also be terminated through a manual "travel-down" command.

Target accuracy

Since the function block determines the blind position solely via run times, the cycle time of the PLC task plays a crucial role for positioning accuracy. If the switching time for a louvre angle range of -70° to 10° is 1 second, for example, the accuracy at a cycle time of 50 ms is $\pm 4^\circ$.

VAR_INPUT

<code>bEn</code>	: BOOL;
<code>stSunbld</code>	: ST_BA_SunBld;
<code>udiTiUp_ms</code>	: UDINT;
<code>udiTiDwn_ms</code>	: UDINT;
<code>udiTurnTiUp_ms</code>	: UDINT;
<code>udiTurnTiDwn_ms</code>	: UDINT;
<code>udiBckLshTiUp_ms</code>	: UDINT;
<code>udiBckLshTiDwn_ms</code>	: UDINT;
<code>rAnglLmtUp</code>	: REAL;
<code>rAnglLmtDwn</code>	: REAL;

bEn: Enable input for the function block. As long as this input is TRUE, the actuator function block accepts and executes commands as described above. A FALSE signal on this input resets the control outputs `bUp` and `bDwn` and the function block remains in a state of rest.

stSunbld: Positioning telegram, (see `ST_BA_SunBld` [► 602]).

udiTiUp_ms: Complete time for driving up [ms].

udiTiDwn_ms: Complete time for driving down [ms].

udiTurnTiUp_ms: Time for turning the louvres in the upward direction [ms].

udiTurnTiDwn_ms: Time for turning the louvres in the downward direction [ms].

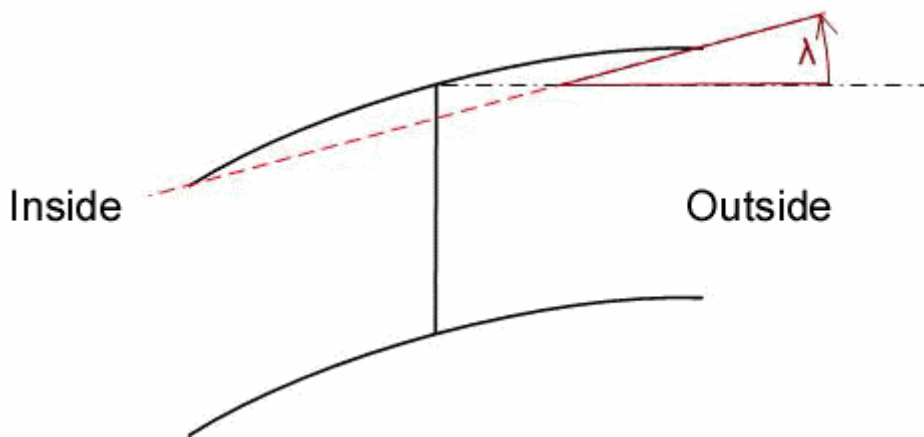
udiBckLshTiUp_ms: Time to traverse the backlash in the upward direction [ms]. This input is internally limited to a minimum value of 0.

udiBckLshTiDwn_ms: Time to traverse the backlash in the downward direction [ms]. This input is internally limited to a minimum value of 0.

rAnglLmtUp: Highest position of the louvres [°].

This position is reached once the blind has moved to the top position.

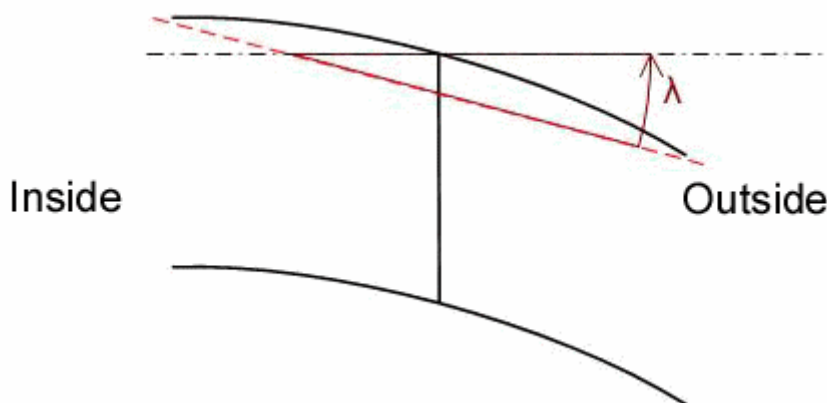
The louvre angle λ , as defined above, is then typically greater than zero.



rAnglLmtDwn: Lowest position of the louvres [°].

This position is reached once the blind has moved to the bottom position.

The louvre angle λ , as defined above, is then typically less than zero.



VAR_OUTPUT

```
bUp      : BOOL;
bDwn     : BOOL;
rActlPos : REAL;
rActlAngl : REAL;
bRef     : BOOL;
udiRefTi_sec : UDINT;
bInitRefCompl : BOOL;
```

```

bBusy      : BOOL;
bErr       : BOOL;
sErrDesc   : T_MAXSTRING;

```

bUp: Control output for blind up.

bDwn: Control output for blind down.

rActlPos: Current position in percent.

rActlAngl: Current louvre angle [°].

bRef: The blind is referencing, i.e. the output *bUp* is set for the complete travel-up time + 5s. Only a manual "down" command can move the blind in the opposite direction and terminate this mode.

udiRefTi_sec: Referencing countdown display [s].

blnitRefCompl: Initial referencing process complete.

bBusy: A positioning or a referencing procedure is in progress.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

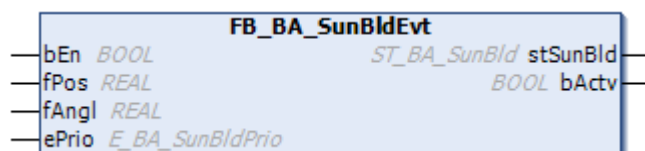
sErrDesc: Contains the error description.

Error description
01: Error: Up/Down timer = 0.
02: Error: Turning timer = 0.
03: Error: Louvre angle limits: The upper limit is less than or equal to the lower limit ($rAnglLmtUp \leq rAnglLmtDwn$).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.18 FB_BA_SunBldEvt



This function block serves to preset the position and angle for any desired event. It can be used, for example, in order to drive to a parking position or to drive the blind upward for maintenance.

The function is activated via the input *bEn*. If this is the case, the active flag in the positioning telegram (*bActv* in *stSunBld*) at output *stSunBld* [► 602] is set, and the values entered for the In/Out variables *rPos* for the blind height [%] and *rAngl* the louvre angle [°] are passed on in this telegram. If the function is no longer active due to the resetting of *bEn*, then the active flag in the positioning telegram *stSunBld* [► 602] is reset and the positions for height and angle are set to "0". The priority function block (e.g. *FB_BA_SunBldPrioSwi4* [► 512]) enables a function with lower priority to take over the control by resetting.

VAR_INPUT

```

bEn      : BOOL;
rPos     : REAL;
rAngl    : REAL;

```

bEn: A TRUE signal on this input activates the function block and transfers the entered setpoint values together with the active flag in the positioning telegram *ST_BA_SunBld* [► 602]. A FALSE signal resets the active flag again and sets position and angle to zero.

rPos: Height position of the blind [%] in case of activation.

rAngl: Louvre angle of the blind [°] in case of activation.

VAR_OUTPUT

```
stSunBld : ST_BA_SunBld;
bActv    : BOOL;
```

bActv: Corresponds to the boolean value *bActv* in the blind telegram [ST_BA_SunBld \[► 602\]](#) and is solely used to indicate whether the function block sends an active telegram.

stSunBld: Output structure of the blind positions, see [ST_BA_SunBld \[► 602\]](#)

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.19 FB_BA_SunBldIcePrtc



The function block *FB_BA_SunBldIcePrtc* deals with direction-independent anti-freezing.

The weather protection has the highest priority in the blind controller (see [overview \[► 467\]](#)) and is intended to ensure that the blind is not damaged by ice or wind.

Impending icing up is detected by the fact that, during precipitation detection at *bRainSns*, the measured outside temperature *rOtsT* is below the frost limit *rFrstT*. This event is saved internally and remains active until it is ensured that the ice has melted again. In addition, the outside temperature must have exceeded the frost limit value for the entered deicing time *udiDeiceTi_sec* [s]. For safety reasons the icing event is persistently saved, i.e. also beyond a PLC failure. Thus, if the controller fails during the icing up or deicing period, the blind is considered to be newly iced up when then the controller restarts and the deicing timer starts from the beginning again.

If there is a risk of icing, the blind is moved to the protective position specified by *rPosProt* (height position in percent) and *rAnglProt* (louvre angle [°]).

VAR_INPUT

```
bEn      : BOOL;
rOtsT    : REAL;
bRainSns : BOOL;
rFrstT   : REAL;
udiDeiceTi_sec : UDINT;
rPosProt : REAL;
rAnglProt : REAL;
```

bEn: The function block has no function if this input is FALSE. In the positioning telegram [ST_BA_SunBld \[► 602\]](#) 0 is output for the position and the angle, and *bActv* is FALSE. This means that another function takes over control of the blind via the priority controller.

rOtsT: Outside temperature [°C].

bRainSns: Input for a rain sensor.

rFrstT: Icing up temperature limit value [°] Celsius. This value may not be greater than 0. Otherwise an error is output.

udiDeiceTi_sec: Time until the deicing of the blind after icing up [s]. After that the icing up alarm is reset.

rPosProt: Height position of the blind [%] in the case of protection.

rAnglProt: Louvre angle of the blind [°] in the case of protection.

VAR_OUTPUT

```
stSunBld      : ST_BA_SunBld;
bActv        : BOOL;
bIceAlm      : BOOL;
udiRemTiIceAlm_sec : UDINT;
```

stSunBld: Output structure of the blind positions, see [ST_BA_SunBld \[► 602\]](#).

bActv: Corresponds to the boolean value *bActv* in the blind telegram [ST_BA_SunBld \[► 602\]](#) and is solely used to indicate whether the function block sends an active telegram.

bIceAlm: Indicates the icing up alarm.

udiRemTiIceAlm_sec: In the case of impending icing up (*bIceAlm*=TRUE), this second counter is set to the deicing time. As soon as the temperature lies above the frost point entered (*rFrstT*), the remaining number of seconds until the 'all-clear' signal is given (*bIceAlm*=FALSE) are indicated here. This output is 0 as long as no countdown of the time is taking place.

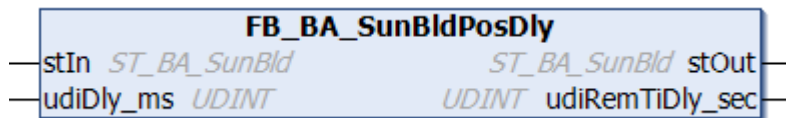


If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.20 FB_BA_SunBldPosDly



This function block delays changes in position based on automatic commands.

If an event, e.g. weather protection, results in too many blind drives being started at the same time, fuses may be triggered by motor starting current peaks. It is therefore advisable to start the blind drives slightly staggered, in order to avoid excessive total current values.

This function block relays automatic commands from the input telegram [stIn \[► 602\]](#) to the output telegram [stOut \[► 602\]](#) with a delay. A distinction is made between three cases

1. the blind position *rPos* has changed in automatic mode (*bManMode*=FALSE in telegram *stIn*)
2. the louvre angle *rAngl* has changed in automatic mode (*bManMode*=FALSE in telegram *stIn*)
3. manual mode has just been exited, i.e. automatic mode has just become active (falling edge *bManMode* in telegram *stIn*)

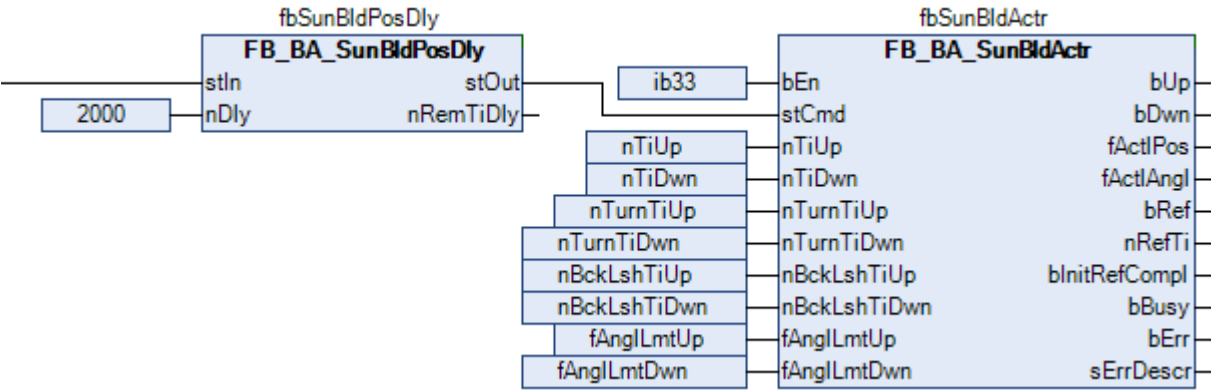
The output telegram *stOut* is always a direct copy of the input telegram *stIn*. In these three cases, however, the output telegram *stOut* is set for the time *udiDly_ms* [ms].

This ensures that the blind controlled via the function block [FB_BA_SunBldActr \[► 504\]](#) is kept at its position during the delay period. Each further change based on the criteria mentioned above within the delay time restarts the timer.

However, a change to manual in the input telegram (*bManMode* = TRUE) cancels the delay timer immediately. The (manual) telegram is passed on without delay. In this way, **only** automatic telegrams are delayed.

Application

Preferably directly before the blind actuator function block:



VAR_INPUT

```
stIn      : ST_BA_Sunblind;
udiDly_ms : UDINT;
```

- stIn:** Input positioning telegram, see [ST_BA_SunBld](#) [► 602].
- udiDly_ms:** Delay time of the active bit in the positioning telegram [ms].

VAR_OUTPUT

```
stOut      : ST_BA_Sunblind;
udiRemTiDly_sec : UDINT;
```

- stOut:** Output positioning telegram, see [ST_BA_SunBld](#) [► 602].
- udiRemTiDly_sec:** Display output for elapsed delay time [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.21 FB_BA_SunBldPrioSwi4



The function block is used for priority control for up to 4 positioning telegrams (*stSunBld_Prio1* ... *stSunBld_Prio4*) of type [ST_BA_SunBld](#) [► 602].

Structure of the blind positioning telegram [ST_BA_Sunbld](#) [► 602].

```
TYPE ST_BA_SunBld:
STRUCT
    rPos      : REAL;
    rAngl     : REAL;
    bManUp    : BOOL;
    bManDwn   : BOOL;
    bManMod   : BOOL;
```

```

    bActv      : BOOL;
END_STRUCT
END_TYPE

```

Up to 4 positioning telegrams from different control function blocks can be applied to this function block. The telegram on *stSunBld_Prio1* has the highest priority and that on *stSunBld_Prio4* the lowest. The active telegram with the highest priority is output at the output *stSunBld*. "Active" means that the variable *bActv* is set within the structure of the positioning telegram.

This function block is to be programmed in such a way that one of the applied telegrams is always active. If no telegram is active, an empty telegram is output, i.e. *rPos*=0, *rAngl*=0, *bManUp*=FALSE, *bManDwn*=FALSE, *bManMod*=FALSE, *bActv*=FALSE. Since the blind function block [FB_BA_SunBldActr \[► 504\]](#) or the roller blind function block [FB_BA_RolBldActr \[► 496\]](#) does not take account of the flag *bActv*, this telegram would be interpreted as movement command to position "0", i.e. fully open. The absence of an active telegram therefore does not represent a safety risk for the blind.

VAR_INPUT

```

stSunBld_Prio1 : ST_BA_SunBld;
stSunBld_Prio2 : ST_BA_SunBld;
stSunBld_Prio3 : ST_BA_SunBld;
stSunBld_Prio4 : ST_BA_SunBld;

```

stSunBld_Prio1..stSunBld_Prio4: Positioning telegrams available for selection. *stSunBld_Prio1* has the highest priority and *stSunBld_Prio4* the lowest.

VAR_OUTPUT

```

stSunBld      : ST_BA_SunBld;
udiActvPrio   : UDINT;

```

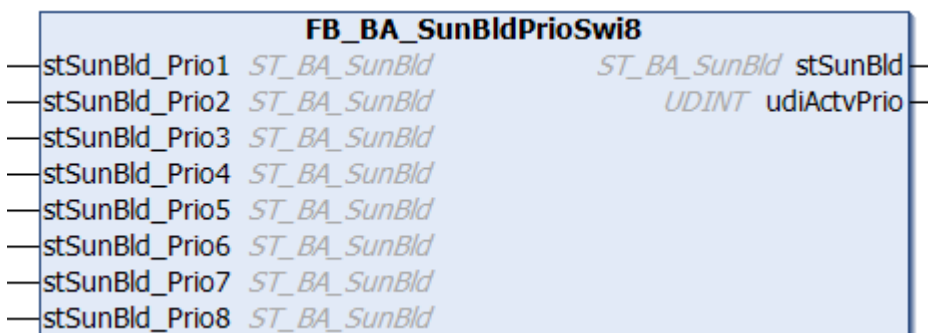
stSunBld: Resulting positioning telegram.

udiActvPrio: Active positioning telegram. If none is active, "0" is output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.22 FB_BA_SunBldPrioSwi8



The function block is used for priority control for up to 8 positioning telegrams (*stSunBld_Prio1* ... *stSunBld_Prio8*) of type [ST_BA_SunBld \[► 602\]](#).

Structure of the blind positioning telegram [ST_BA_Sunbld \[► 602\]](#).

```

TYPE ST_BA_SunBld:
STRUCT
    rPos      : REAL;
    rAngl     : REAL;
    bManUp    : BOOL;
    bManDwn   : BOOL;
    bManMod   : BOOL;
    bActv     : BOOL;
END_STRUCT
END_TYPE

```

Up to 8 positioning telegrams from different control function blocks can be applied to this function block. The telegram on *stSunBld_Prio1* has the highest priority and that on *stSunBld_Prio8* the lowest. The active telegram with the highest priority is output at the output *stSunBld*. "Active" means that the variable *bActv* is set within the structure of the positioning telegram.

This function block is to be programmed in such a way that one of the applied telegrams is always active. If no telegram is active, an empty telegram is output, i.e. *rPos*=0, *rAngl*=0, *bManUp*=FALSE, *bManDwn*=FALSE, *bManMod*=FALSE, *bActv*=FALSE. Since the blind function block *FB_BA_SunBldActr* [► 504] or the roller blind function block *FB_BA_RolBldActr* [► 496] does not take account of the flag *bActv*, this telegram would be interpreted as movement command to position "0", i.e. fully open. The absence of an active telegram therefore does not represent a safety risk for the blind.

VAR_INPUT

```
stSunBld_Prio1 : ST_BA_SunBld;
stSunBld_Prio2 : ST_BA_SunBld;
stSunBld_Prio3 : ST_BA_SunBld;
stSunBld_Prio4 : ST_BA_SunBld;
stSunBld_Prio5 : ST_BA_SunBld;
stSunBld_Prio6 : ST_BA_SunBld;
stSunBld_Prio7 : ST_BA_SunBld;
stSunBld_Prio8 : ST_BA_SunBld;
```

stSunBld_Prio1..stSunBld_Prio8: Positioning telegrams available for selection. *stSunBld_Prio1* has the highest priority and *stSunBld_Prio8* the lowest.

VAR_OUTPUT

```
stSunBld : ST_BA_SunBld;
udiActvPrio : UDINT;
```

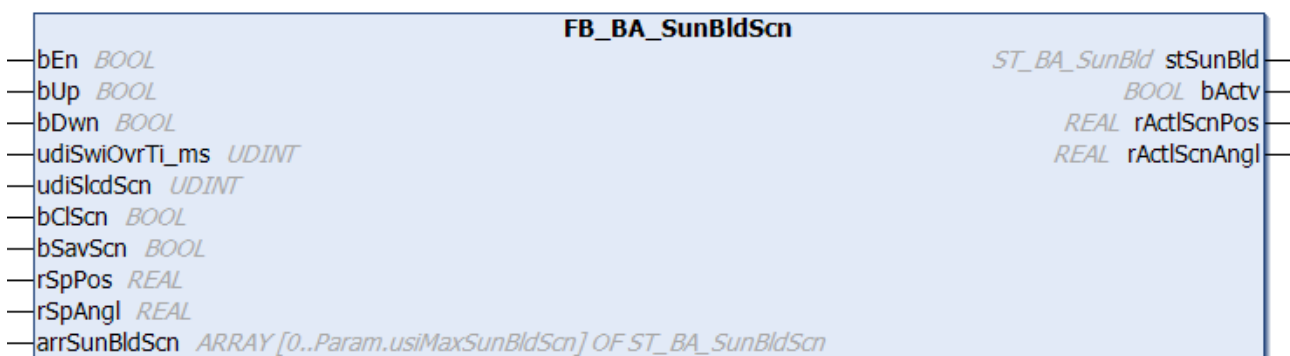
stSunBld: Resulting positioning telegram.

udiActvPrio: Active positioning telegram. If none is active, "0" is output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.23 FB_BA_SunBldScn



This function block represents an extension of the manual controller *FB_BA_SunBldSwi* [► 517] by a scene memory and a call function. The blind control *FB_BA_SunBldActr* [► 504] or the roller blind control *FB_BA_RolBldActr* [► 496] can be active in manual mode and also directly target previously stored positions (scenes). Up to 21 scenes can be saved.

Structure of the blind positioning telegram *ST_BA_Sunbld* [► 602].

```
TYPE ST_BA_SunBld:
STRUCT
    rPos      : REAL;
    rAngl     : REAL;
    bManUp    : BOOL;
    bManDwn   : BOOL;
```



```

    bManMod      : BOOL;
    bActv        : BOOL;
END_STRUCT
END_TYPE

```

Operation

In manual mode, the function block controls the blind function block [FB_BA_SunBldActr](#) [► 504] or the roller shutter function block [FB_BA_RolBldActr](#) [► 496] via the command inputs *bUp* and *bDwn*; *bUp* has priority. The commands are passed on to the respective commands *bManUp* and *bManDwn* of the positioning telegram. If a command input is activated that is longer than the entered time *udiSwiOvrTi_ms* [ms], then the corresponding control command latches. Activating a command input again releases this latch.

A rising edge on *bSavScn* saves the current position and louvre angle in the scene selected in *udiSlcdScn*. This procedure is possible at any time, even during active positioning. The selected scene is called with *bClScn*, i.e. the saved position and angle values are driven to.

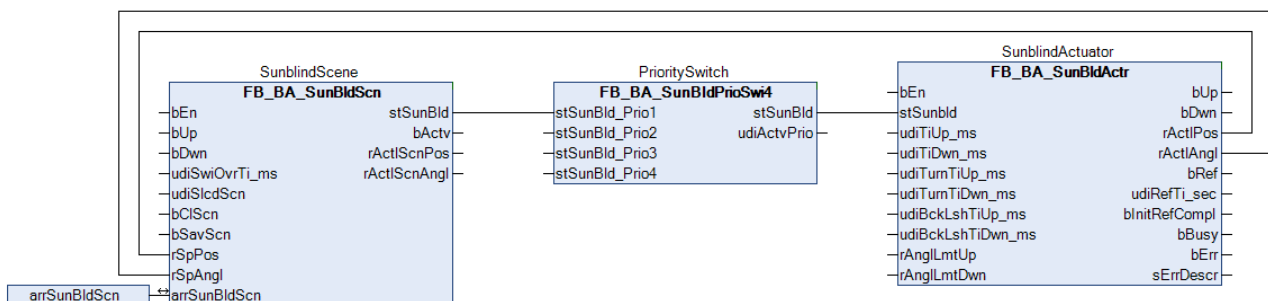
If the function block is activated by input *bEn*=TRUE, bit *bActv* is set immediately in the positioning telegram. The function block uses this to notify a priority switch ([FB_BA_SunBldPrioSwi4](#) [► 512] or [FB_BA_SunBldPrioSwi8](#) [► 513]) of its priority over lower priorities. If the command "Call Scene" is not active (*bClScn* =TRUE), the bit *bManMod* is also set in the positioning telegram to notify the connected actuator function blocks that they should respond to manual commands.

If the function block is deactivated by *bEn*=FALSE, both bits, *bActv* and *bManMod*, are set to FALSE again.

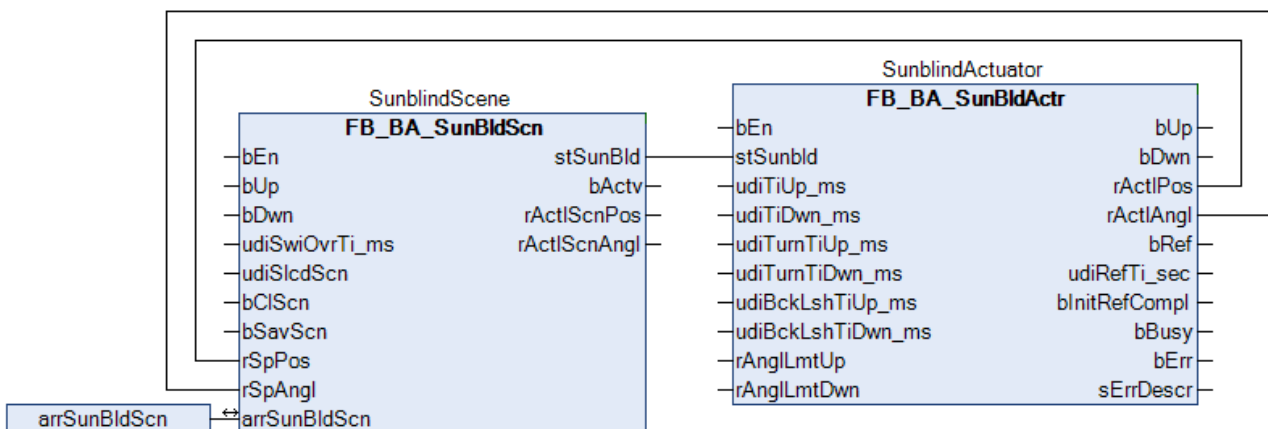
Linking to the blind function block

Like the "normal" manual mode function block [FB_BA_SunBldSwi](#) [► 517], the scene selection function block can be connected either via an upstream priority control [FB_BA_SunBldPrioSwi4](#) [► 512] or [FB_BA_SunBldPrioSwi8](#) [► 513], or directly via the blind function block. The connection is established via the positioning telegram [ST_BA_Sunbld](#) [► 602]. Furthermore the scene function block requires the current positions from the blind function block for the reference blind:

Use of a priority controller:



Direct connection:



VAR_INPUT

```

bEn          : BOOL;
bUp          : BOOL;
bDwn        : BOOL;
udiSwiOvrTi_ms : UDINT;
udiSlcdScn   : UDINT;
bClScn       : BOOL;
bSavScn      : BOOL;
rSpPos       : REAL;
rSpAngl      : REAL;

```

bEn: The function block has no function if this input is FALSE. In the positioning telegram ST BA Sunbld [► 602], 0 is output for the position and the angle - *bManMod* and *bActv* are FALSE. For a connection with priority controller this means that another functionality takes over control of the blind. Conversely, a direct connection allows the blind to drive directly to the 0 position, i.e. fully up, since the actuator function block does not evaluate the bit *bActv* itself.

bUp: Command input for blind up.

bDwn: Command input for blind down.

udiSwiOvrTi_ms: Time [ms] until the corresponding manual command in the positioning telegram ST BA Sunbld [► 602] switches to latching mode, if the command input is activated permanently. Internally limited to a minimum value of 0.

udiSlcdScn: Selected scene which should either be saved (*bSavScn*) or called (*bClScn*). Internally limited to a minimum value of 0 to *cMaxSunBldScn*.

bClScn: Call selected scene.

bSavScn: Save selected scene.

rSpPos: Set position [%] that is to be saved in the selected scene. This must be linked to the actual position of the actuator function block FB BA SunBldActr [► 504] or FB BA RolBldActr [► 496] of the reference blind/roller shutter, in order to be able to save a position that was previously approached manually. Internally limited to values between 0 and 100.

rSpAngl: ditto. Louvre angle [°].

VAR_OUTPUT

```

stSunBld     : ST_BA_SunBld;
bActv        : BOOL;
rActlScnPos  : REAL;
rActlScnAngl : REAL;

```

stSunBld: Positioning telegram, see ST BA SunBld [► 602].

bActv: Corresponds to the boolean value *bActv* in the blind telegram ST BA SunBld [► 602] and is solely used to indicate whether the function block sends an active telegram.

rActlScnPos: Indicates the saved relative blind height position [%] for the currently selected scene.

rActlScnAngl: ditto. Louvre angle [°].



If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

VAR_IN_OUT

```

arrSunBldScn : ARRAY[0..Param.usiMaxSunBldScn] OF ST_BA_SunBldScn;

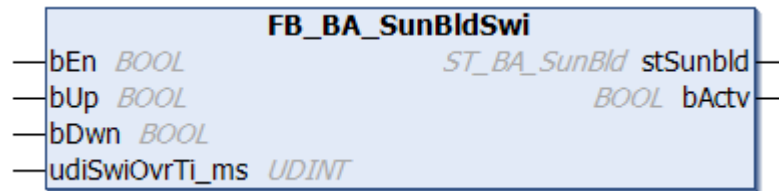
```

arrSunBldScn: Table with the scene entries of the type ST BA SunBldScn [► 603].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.24 FB_BA_SunBldSwi



This function block can be used to control the blind [FB_BA_SunBldActr](#) [► 504] or roller shutter [FB_BA_RolBldActr](#) [► 496] in manual mode. The connection takes place via the positioning telegram [ST_BA_Sunbld](#) [► 602] either directly or with an additional priority controller.

Structure of the blind positioning telegram [ST_BA_Sunbld](#) [► 602].

```

TYPE ST_BA_SunBld:
STRUCT
    rPos      : REAL;
    rAngl     : REAL;
    bManUp    : BOOL;
    bManDwn   : BOOL;
    bManMod   : BOOL;
    bActv     : BOOL;
END_STRUCT
END_TYPE

```

Operation

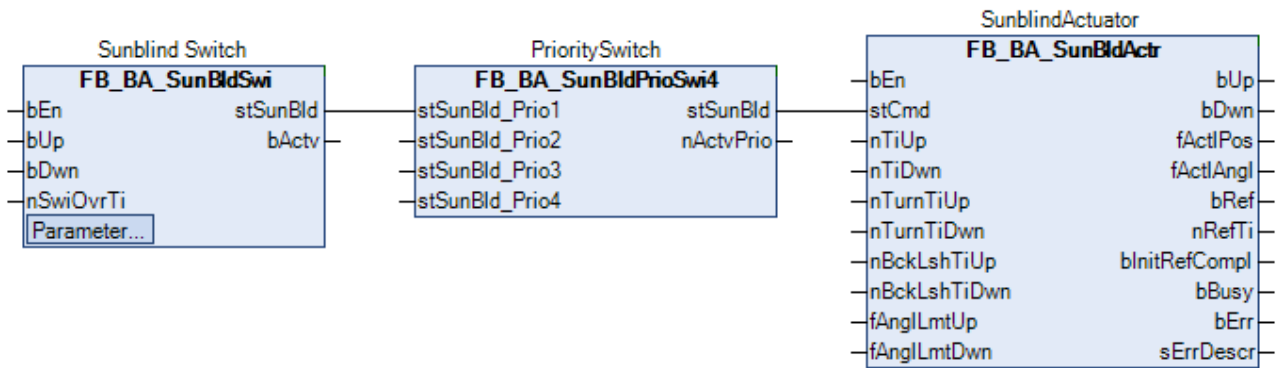
In manual mode, the function block controls the blind function block [FB_BA_SunBldActr](#) [► 504] or the roller shutter function block [FB_BA_RolBldActr](#) [► 496] via the command inputs *bUp* and *bDwn*; *bUp* has priority. The commands are passed on to the respective commands *bManUp* and *bManDwn* of the positioning telegram. If a command input is activated that is longer than the entered time *udiSwiOvrTi_ms* [ms], then the corresponding control command latches. Activating a command input again releases this latch. If the function block is activated by input *bEn*=TRUE, bit *bActv* is set immediately in the positioning telegram. The function block uses this to notify a priority switch ([FB_BA_SunBldPrioSwi4](#) [► 512] or [FB_BA_SunBldPrioSwi8](#) [► 513]) of its priority over lower priorities. At the same time, the bit *bManMod* is set in the positioning telegram to notify the connected actuator function blocks that they should respond to manual commands.

If the function block is deactivated by *bEn*=FALSE, both bits, *bActv* and *bManMod*, are set to FALSE again.

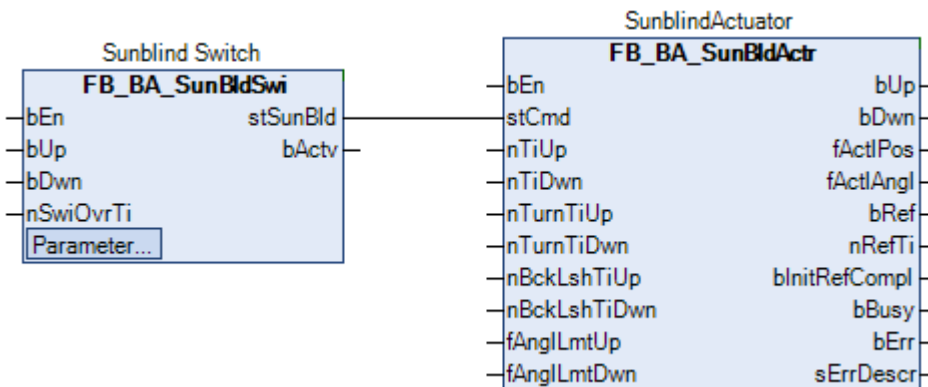
Linking to the blind function block

The manual mode function block can be connected either via an upstream priority control [FB_BA_SunBldPrioSwi4](#) [► 512] or [FB_BA_SunBldPrioSwi8](#) [► 513], or directly at the blind function block. The connection is established via the positioning telegram [ST_BA_Sunbld](#) [► 602].

Use of a priority controller:



Direct connection:



VAR_INPUT

```

bEn      : BOOL;
bUp      : BOOL;
bDwn     : BOOL;
udiSwiOvrTi_ms : UDINT;

```

bEn: The function block has no function if this input is FALSE. In the positioning telegram `ST_BA_Sunbld` [► 602], 0 is output for the position and the angle - `bManMod` and `bActv` are FALSE. For a connection with priority controller this means that another functionality takes over control of the blind. Conversely, a direct connection allows the blind to drive directly to the 0 position, i.e. fully up, since the actuator function block does not evaluate the bit `bActv` itself.

bUp: Command input for blind up.

bDwn: Command input for blind down.

udiSwiOvrTi_ms: Time [ms] until the corresponding manual command in the positioning telegram `ST_BA_Sunbld` [► 602] switches to latching mode, if the command input is activated permanently. Internally limited to a minimum value of 0.

VAR_OUTPUT

```

stSunBld : ST_BA_SunBld;
bActv    : BOOL;

```

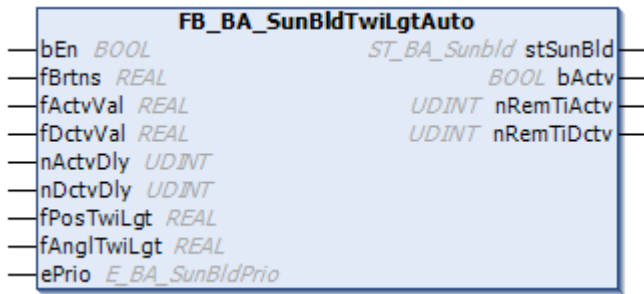
stSunBld: Positioning telegram, see `ST_BA_SunBld` [► 602].

bActv: Corresponds to the boolean value `bActv` in the blind telegram `ST_BA_SunBld` [► 602] and is solely used to indicate whether the function block sends an active telegram.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.25 FB_BA_SunBldTwilGtAuto



This function block controls the blind when the outdoor brightness has fallen below a limit value.

The automatic twilight function operates with both a value hysteresis and a temporal hysteresis: If the outdoor brightness value *rBrtns* [lux] falls below the value *rActvVal* [lux] for the time *udiActvDly_sec* [s], the function block is active and will provide the blind positions *rPosTwilGt* (height [%]) and *rAnglTwilGt* (louvre angle [°]) specified for the input variables at the output in the positioning telegram *ST_BA_Sunbld* [► 602]. If the outdoor brightness exceeds the value *rActvVal* [lux] for the time *udiDctvDly_sec* [s], automatic mode is no longer active. The active flag in the positioning telegram *ST_BA_Sunbld* [► 602] is reset and the positions for height and angle are set to "0". A function with a lower priority can then take over control.

VAR_INPUT

```
bEn          : BOOL;
rBrtns       : REAL;
rActvVal     : REAL;
rDctvVal     : REAL;
udiActvDly_sec : UDINT;
udiDctvDly_sec : UDINT;
rPosTwilGt   : REAL;
rAnglTwilGt  : REAL;
```

bEn: The function block has no function if this input is FALSE. In the positioning telegram *ST_BA_Sunbld* [► 602] 0 is output for the position and the angle, and *bActv* is FALSE. This means that another function takes over control of the blind via the priority controller.

rBrtns: Outdoor brightness [lx].

rActvVal: Activation limit value [lx]. The value *rActvVal* is internally limited to values from 0 to *rDctvVal*.

rDctvVal: Deactivation limit value [lx]. Internally limited to a minimum value of 0.

udiActvDly_sec: Activation delay [s]. Internally limited to a minimum value of 0.

udiDctvDly_sec: Deactivation delay [s]. Internally limited to a minimum value of 0.

rPosTwilGt: Vertical position of the blind [%] if the automatic twilight function is active. Internally limited to values between 0 and 100.

rAnglTwilGt: Louvre angle of the blind [°] if the automatic twilight function is active.

VAR_OUTPUT

```
stSunBld      : ST_BA_SunBld;
bActv         : BOOL;
udiRemTiActv_sec : UDINT;
udiRemTiDctv_sec : UDINT;
```

stSunBld: Output structure of the blind positions, see *ST_BA_SunBld* [► 602].

bActv: Corresponds to the boolean value *bActv* in the blind telegram *ST_BA_SunBld* [► 602] and is solely used to indicate whether the function block sends an active telegram.

udiRemTiActv_sec: Shows the time remaining [s] after falling below the switching value *rActvVal* until automatic mode is activated. This output is 0 as long as no countdown of the time is taking place.

udiRemTiDctv_sec: Shows the time remaining [s] after exceeding of the switching value *rDctvVal* until automatic mode is disabled. This output is 0 as long as no countdown of the time is taking place.



If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.26 FB_BA_SunBldWndPrtc



The function block *FB_BA_SunBldWndPrtc* deals with the direction-dependent wind protection.

The weather protection has the highest priority in the blind controller (see [overview \[► 467\]](#)) and is intended to ensure that the blind is not damaged by ice or wind.

If the measured wind speed is above the value *rWndSpdStrmOn* for the time *udiDlyStrmOn_sec* [s], it is assumed that high winds are imminent. The storm is regarded as having subsided, so that the blind can be moved safely, once the wind speed falls below the value *rWndSpdStrmOff* for the time *udiDlyStrmOff_sec* [s]. For safety reasons the storm event is also persistently saved. Thus, if the controller fails during a storm, the sequence timer is started again from the beginning when the controller is restarted.

If there is a risk of high wind, the blind is moved to the protection position specified by *rPosProt* (height position in percent) and *rAnglProt* (louvre angle [°]).

VAR_INPUT

```

bEn          : BOOL;
rWndSpd      : REAL;
rWndSpdStrmOn : REAL;
rWndSpdStrmOff : REAL;
udiDlyStrmOn_sec : UDINT;
udiDlyStrmOff_sec : UDINT;
rPosProt     : REAL;
rAnglProt    : REAL;

```

bEn: The function block has no function if this input is FALSE. In the positioning telegram *ST_BA_Sunbld* [► 602] 0 is output for the position and the angle, and *bActv* is FALSE. This means that another function takes over control of the blind via the priority controller.

rWndSpd: Wind speed. The unit of entry is arbitrary, but it is important that no value is smaller than 0 and that the values become larger with increasing speed.

rWndSpdStrmOn: Wind speed limit value for the activation of the storm alarm. This value may be not smaller than 0 and must lie above the value for the deactivation. Otherwise an error is output. The unit of entry must be the same as that of the input *rWndSpd*. A value greater than this limit value triggers the alarm after the specified time *udiDlyStrmOn_sec*.

rWndSpdStrmOff: Wind speed limit value for the deactivation of the storm alarm. This value may be not smaller than 0 and must lie below the value for the activation. Otherwise an error is output. The unit of entry must be the same as that of the input *rWndSpd*. A value smaller than or equal to this limit value resets the alarm after the specified time *udiDlyStrmOff_sec*.

udiDlyStrmOn_sec: Time delay until the storm alarm is triggered [s].

udiDlyStrmOff_sec: Time delay until the storm alarm is reset [s].

rPosProt: Height position of the blind [%] in the case of protection.

rAnglProt: Louvre angle of the blind [°] in the case of protection.

VAR_OUTPUT

```
stSunBld      : ST_BA_SunBld;
bActv         : BOOL;
bStrmAlm      : BOOL;
udiRemTiStrmDetc_sec : UDINT;
udiRemTiStrmAlm_sec : UDINT;
```

stSunBld: Output structure of the blind positions, see [ST_BA_SunBld](#) [► 602].

bActv: Corresponds to the boolean value *bActv* in the blind telegram [ST_BA_SunBld](#) [► 602] and is solely used to indicate whether the function block sends an active telegram.

bStrmAlm: Indicates the storm alarm.

udiRemTiStrmDetc_sec: In the non-critical case, this second counter continuously shows the alarm delay time *udiDlyStrmOn_sec*. If the measured wind speed *rWndSpd* is above the activation limit value *rWndSpdStrmOn*, the seconds to the alarm are counted down. This output is 0 as long as no countdown of the time is taking place.

udiRemTiStrmAlm_sec: As soon as the storm alarm is initiated, this second counter initially constantly indicates the deactivation time delay of the storm alarm *udiDlyStrmOff_sec*. If the measured wind speed *rWndSpd* falls below the deactivation limit value *rWndSpdStrmOff*, the seconds to the all-clear signal (*bStrmAlm*=FALSE) are counted down. This output is 0 as long as no countdown of the time is taking place.

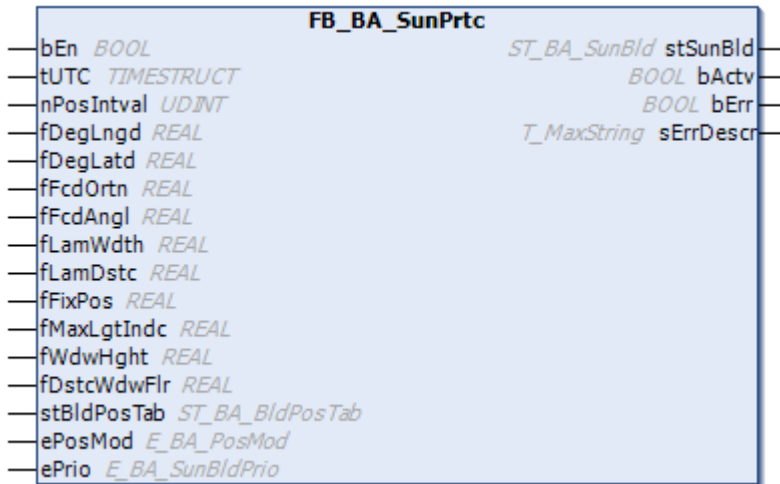


If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.27 FB_BA_SunPrtc



The function block is used for glare protection with the aid of a slatted blind.

Glare protection is realized through variation of the louvre angle and positioning of the blind height.

The louvre angle is set as a function of the sun position such that direct glare is prevented, while letting as much natural light through as possible.

Three different operating modes are available for varying the blind height.

1. When sun protection is active, the blind moves to a fixed height. The height value is specified with the variable `rFixPos`.
2. The blind position is varied as a function of the sun position. The position is specified in the table (ST_BA_BldPosTab [► 599]). See also description of FB_BA_BldPosEntry [► 474].
3. The high of the blind is calculated based on the window geometry such that the sun's rays reach a specified depth in the room. The incidence depth of the sun's rays is defined with the variable `rMaxLgtIndc`.

In order to avoid excessive repositioning of the louvre angle, the variable `udiPosIntval_min` can be used to specify a time interval, within which the louvre angle is not adjusted. In order to avoid glare, the angle is always changed sufficiently for the respective time interval.

The following conditions must be met for positioning the blind and setting the louvre angle.

- 1. The input `bEn` must be TRUE.
- 2. The sun must have risen. (elevation > 0)
- 3. The function block is parameterized correctly (`bErr=FALSE`)

VAR_INPUT

```

bEn          : BOOL;
stUTC        : Timestruct;
udiPosIntval_min : UDINT;
rDegLngd     : REAL;
rDegLatd     : REAL;
rFcdOrtn     : REAL;
rFcdAngl     : REAL;
rLamWdth     : REAL;
rLamDstc     : REAL;
rFixPos      : REAL;
rMaxLgtIndc  : REAL;
rWdwHght     : REAL;
rDstcWdwFr   : REAL;
stBldPosTab  : ST_BA_BldPosTab;
ePosMod      : E_BA_PosMod;

```


bEn: If this input is set to FALSE the positioning is inactive, i.e. the active bit (*bActv*) is reset in the positioning structure *stSunBld* of the type *ST_BA_Sunbld* [► 602] and the function block itself remains in a standstill mode. If on the other hand the function block is activated, then the active bit is TRUE and the function block outputs its control values (*rPos*, *rAngl*) in the positioning structure at the appropriate times.

stUTC: Input of current time as coordinated world time (UTC - Universal Time Coordinated, previously referred to as GMT, Greenwich Mean Time) (see TIMESTRUCT). The function block *FB_BA_GetTime* [► 531] can be used to read this time from a target system.



A jump of more than 300 seconds leads to immediate repositioning, if the blind is in the sun and glare protection is active, based on the above criteria. This functionality was added to ensure a reproducible program execution.

udiPosIntval_min: Positioning interval in minutes - time between two blind position outputs. Valid range: 1 min...720 min.

rDegLngd: Longitude [°]. Valid range: - 180°...180°.

rDegLatd: Latitude [°]. Valid range: - 90°...90°.

rFcdOrtn: Facade orientation [°]:

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):

Line of sight	Facade orientation
North	$\beta=0^\circ$
East	$\beta=90^\circ$
South	$\beta=180^\circ$
West	$\beta=270^\circ$

The following applies for the southern hemisphere:

Line of sight	Facade orientation
South	$\beta=0^\circ$
East	$\beta=90^\circ$
North	$\beta=180^\circ$
West	$\beta=270^\circ$

rFcdAngl: Facade inclination [°]. See *facade inclination* [► 472].

rLamWdth: Width of the louvres in mm, see *Louvre adjustment* [► 469].

rLamDstc: Louvre spacing in mm, see *Louvre adjustment* [► 469].

rFixPos: Fixed (constant) shutter height [0..100%]. Applies if *ePosMod* = *ePosModFix* (see enumerator *E_BA_PosMod* [► 598]).

rMaxLgtIndc: Maximum desired light incidence in mm measured from the outside of the wall (see *Height adjustment* [► 471]). The parameters *rWdwHght* and *rDstcWdwFlr* are used to calculate how high the blinds must be, depending on the position of the sun, such that the incidence of light does not exceed the value *rMaxLgtIndc*. Applies if *ePosMod* = *ePosModeMaxIncidence* (see enumerator *E_BA_PosMod* [► 598]).

rWdwHght: Window height in mm for the calculation of the shutter height if the mode "maximum desired incidence of light" is selected.

rDstcWdwFlr: Distance between the floor and the window sill in mm for the calculation of the shutter height if the mode "maximum desired incidence of light" is selected.

stBldPosTab: Table of 6 interpolation points, 4 of which are parameterizable, from which a blind position is then given in relation to the position of the sun by linear interpolation. Applies if *ePosMod* = *ePosModFix* (see enumerator *E_BA_PosMod* [► 598]). For a more detailed description please refer to *FB_BA_BldPosEntry* [► 474].

ePosMod: Selection of the positioning mode, see enumerator `E_BA_PosMod` [► 598].

VAR_OUTPUT

```
stSunBld      : ST_BA_SunBld;
bActv         : BOOL;
bErr          : BOOL;
sErrorDescr   : T_MAXSTRING;
```

stSunBld: Output structure of the blind positions, see `ST_BA_SunBld` [► 602]

bActv: The function block is in active state, i.e. no error is pending, the function block is enabled, and the sun position is in the specified facade area (the facade is sunlit).

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description
01: Error: The duration of the positioning interval is less than or equal to zero, or it exceeds 720 min.
02: Error: The longitude entered is not within the valid range from -180°..180°.
03: Error: The latitude entered is not within the valid range from -90°..90°.
04: Error: The value entered for the facade inclination <i>rFcdAngl</i> is outside the valid range of -90°..90°.
05: Error: The value for the louvre spacing (<i>rLamDstc</i>) is greater than or equal to the value for the louvre width (<i>rLamWdth</i>). This does not represent a "valid" blind, since the louvres cannot close fully. Mathematically, this would lead to errors.
06: Error: The value entered for the louvre width <i>rLamWdth</i> is zero.
07: Error: The value entered for the louvre spacing <i>rLamDstc</i> is zero.
08: Error: The value entered for the fixed blind height (<i>rFixPos</i>) is greater than 100 or less than 0. At the same time, "fixed blind height" positioning is selected - <i>ePosMod</i> = <i>ePosModFix</i> .
09: Error: The bit "values valid" (<i>bVld</i>) in the positioning table <i>stBldPosTab</i> is not set - invalid values: see <code>FB_BA_BldPosEntry</code> . At the same time, "Table" positioning is selected - <i>ePosMod</i> = <i>ePosModTab</i> .
10: Error: The value entered for the maximum required light incidence <i>rMaxLgtIndc</i> is less than or equal to zero. At the same time, "maximum light incidence" is selected - <i>ePosMod</i> = <i>ePosModMaxIndc</i> .
11: Error: The value entered for the window height <i>rWdwHght</i> is less than or equal to zero. At the same time, "maximum light incidence" is selected - <i>ePosMod</i> = <i>ePosModMaxIndc</i> .
12: Error: The distance between lower window edge and floor <i>rDstcWdwFlr</i> that was entered is less than zero. At the same time, "maximum light incidence" is selected - <i>ePosMod</i> = <i>ePosModMaxIndc</i> .
13: Error: An invalid positioning mode is entered at input <i>ePosMod</i> .



If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

Requirements

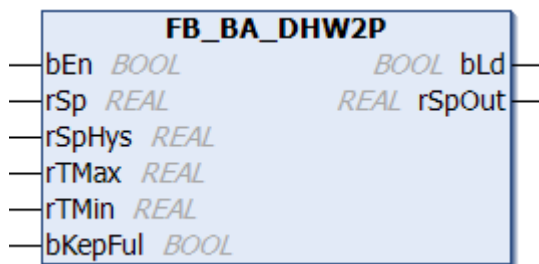
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.3 Provision of hot water

Function blocks

Name	Description
<code>FB_BA_DHW2P</code> [► 525]	Charge control for a hot water tank via an on-off controller.
<code>FB_BA_LglPrev</code> [► 527]	Function block for disinfecting service water and destroying legionella.

6.1.2.3.2.1.3.1 FB_BA_DHW2P



This function block controls the heating of a hot water tank via an on-off controller. Tank heating is activated at input *bEn*. If tank heating is active the output *bLd* is TRUE. The variable *rSp* is used to transfer the setpoint for the hot water temperature to the function block. At input *rTMin* a minimum selection of all temperature sensors for the hot water tank is connected, at input *rTMax* a maximum selection of all temperature sensors.

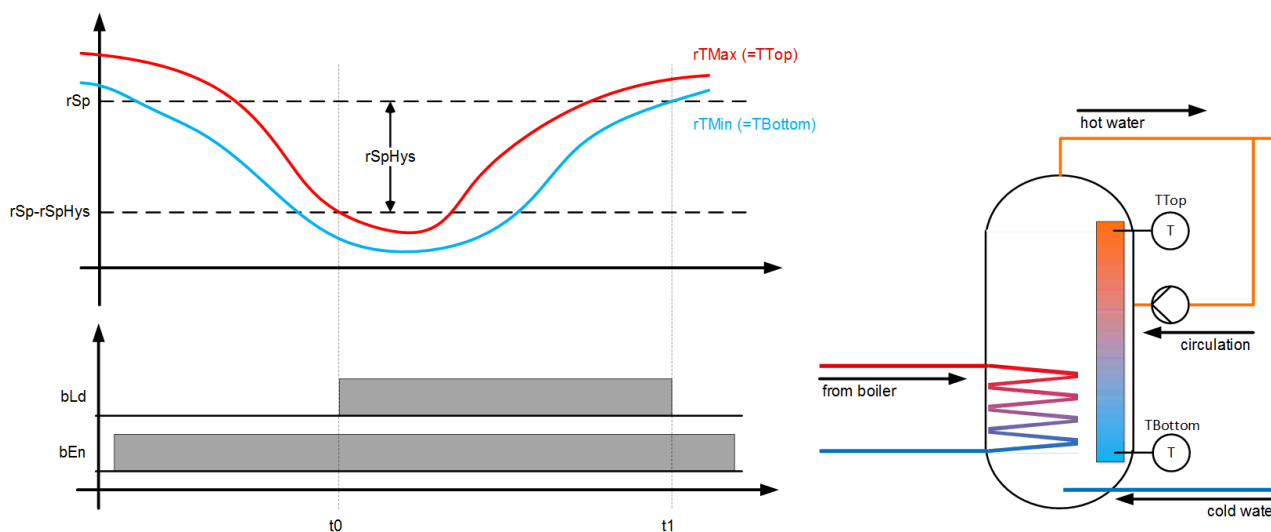
Due to the thermal stratification in the hot water tank, the sensor at the top is generally the one showing the highest temperature, the one at the bottom the lowest.

The tank can be charged in two ways via the variables *bKepFul*:

bKepFul = FALSE

Charging is requested if *rTMax* falls below the value of *rSp-rSpHys*. The charge request is disabled if *rTMin* is above the setpoint of *rSp*.

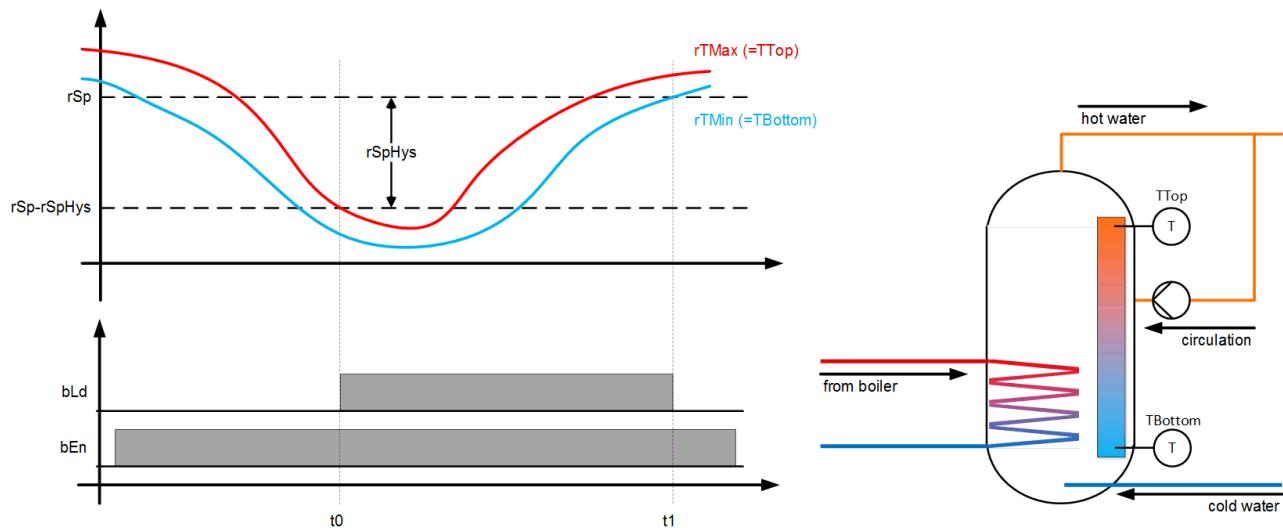
Due to the fact that the sensor at the top generally measures the highest temperature, the heating is not switched on until the hot water tank has been discharged.



bKepFul = TRUE

Charging is requested if *rTMin* falls below the value of *rSp-rSpHys*. The charge request is disabled once *rTMin* is above the setpoint again.

Selecting the minimum of all tank temperatures ensures that the coldest point of the tank is used for control purposes. Recharging takes place when the tank is no longer full.



VAR_INPUT

```

bEn      : BOOL;
rSp      : REAL;
rSpHys   : REAL;
rTMax    : REAL;
rTMin    : REAL;
bKepFul  : BOOL;

```

bEn: Enable boiler charging.

rSp: Service water temperature setpoint [°C].

rSpHys: Hysteresis, recommended 1°K to 5°K.

rTMax: Maximum selection of all tank temperatures [°C].

rTMin: Minimum selection of all tank temperatures [°C].

bKepFul: Control temperature selection:

FALSE = rT_{Max} is used to request bLd , rT_{Min} to switch off

TRUE = rT_{Min} alone controls switching on/off of bLd

VAR_OUTPUT

```

bLd      : BOOL;
rSpOut   : REAL;

```

bLd: Enable charging mode.

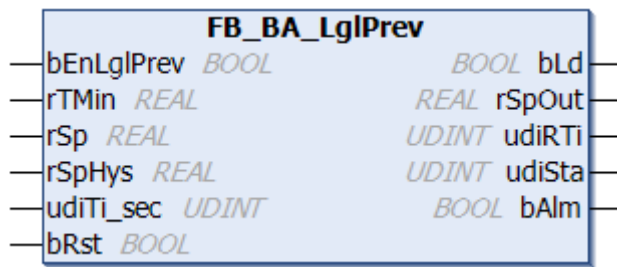
rSpOut: Setpoint transfer to charging circuit:

- $rSpOut = rSp$ (input) if the function block is enabled
- $rSpOut = 0$ if the function block is not enabled

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.3.2 FB_BA_LglPrev

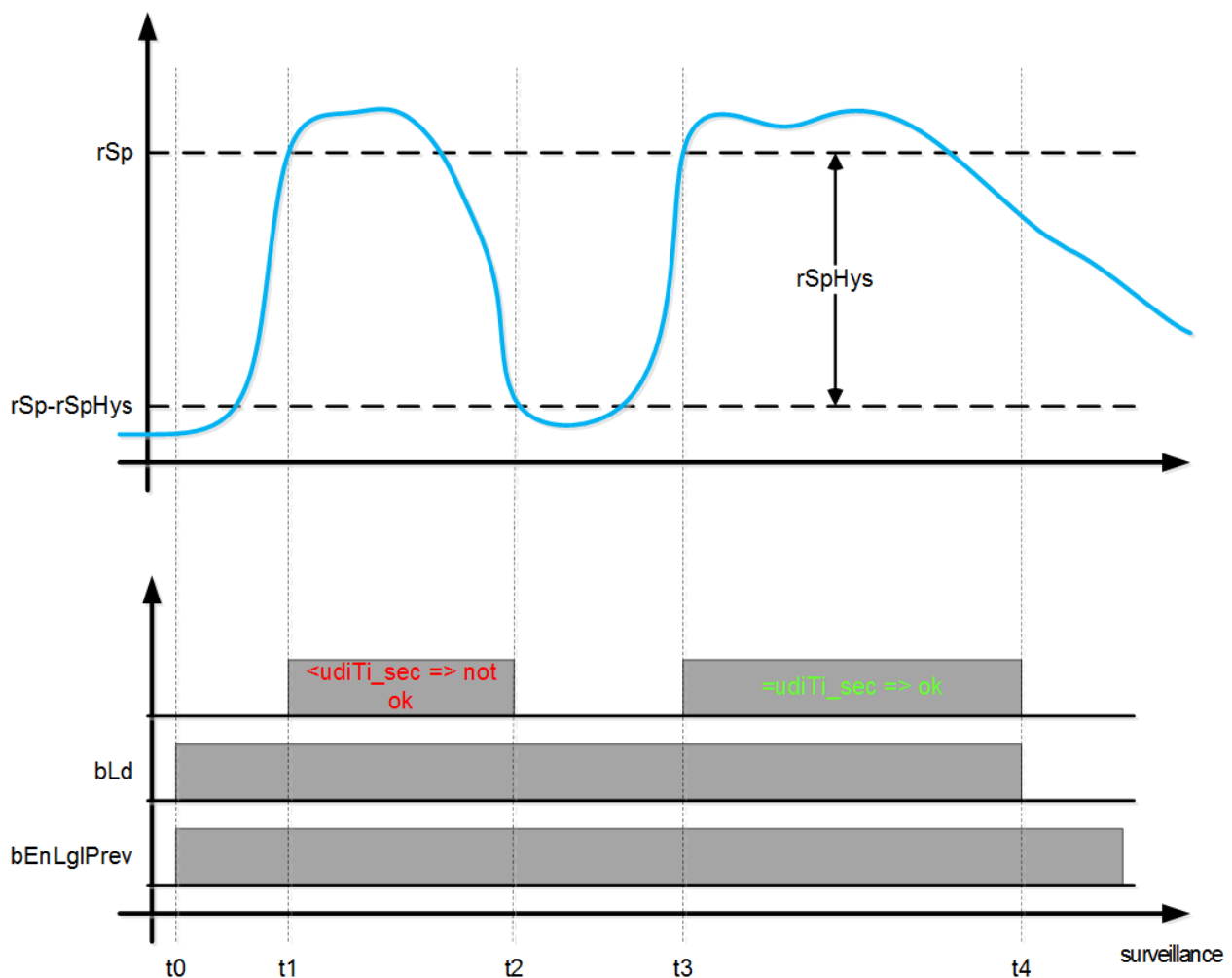


This function block is used for disinfection of the service water and for killing off Legionella. Disinfection mode is activated at input *bEnLglPrev* via a timer program. It is advisable to run the disinfection at least once per week (during the night). The temperature should be at least 70 °C. The activation interval at *bEnLglPrev* must be adequately long. The output *bLd* activates tank heating.

For hot water tanks with several temperature sensors, a minimum selection feature for all sensors must be connected at *rTMin*.

If *rTMin* exceeds the value of *rSp*, a monitoring timer is started with a time of *udiTi_sec* [s]. If the minimum tank temperature *rTMin* remains above *rSp - rSpHys* while the timer is active, the tank was heated adequately. If circulation is active, the output *bLd* must be linked to enabling of the circulation pump, to ensure that the water pipe within the service water system is included in the disinfection. If the temperature has fallen below *rSp - rSpHys* during the disinfection process, the process must be restarted and run until the time *udiTi_sec* has fully elapsed. If the disinfection was successful, the output *bLd* is reset.

If the disinfection process was incomplete during the function block activation (*bEnLglPrev*), this is indicated with the output *bAlm*. The output must be reset with *bRst*.



Explanation of the diagram:

t0 Start of the legionella program and switching of output *bLd*. Heating of the hot water tank.

t1 The tank has reached the temperature *rSp*. The timer for the heating time is started.

t2 The minimum tank temperature has fallen below *rSp - rSpHys*. The timer for the heating time is reset.

t3 The temperature exceeds *rSp* again, and the heating timer is started again.

t4 The Minimum tank temperature was above the limit *rSp - rSpHys* over the period *udiTi_sec*; the disinfection was successful. *bLd* is reset, and the hot water tank switches back to normal operation.

VAR_INPUT

```
bEnLglPrev : BOOL;
rTMin      : REAL;
rSp        : REAL;
rSpHys     : REAL;
udiTi_sec  : UDINT;
bRst       : BOOL;
```

bEnLglPrev: Enabling of disinfection operation via a timer program.

rTMin: Minimum tank temperature [°C]. Minimum selection of temperature sensors at the top and bottom.

rSp: Setpoint for disinfection [°C].

rSpHys: Temperature difference [K] lower limit; always calculated absolute.

udiTi_sec: Monitoring period [s].

bRst: Resetting of the legionella alarm;

VAR_OUTPUT

```
bLd       : BOOL;
rSpOut    : REAL;
udiRTi    : UDINT;
udiSta    : UDINT;
```

bLd: Anti-legionella mode active.

rSpOut: Setpoint transfer to charging circuit:

- *rSp* (input) if the function block is enabled
- 0 if the function block is not enabled

udiRTi: Disinfection mode timer countdown.

udiSta: Disinfection program status:

1. The disinfection operation was successful.
2. The disinfection was completed successfully. After the disinfection, and to reactivate legionella prevention, *bEnLglPrev* must be FALSE.
3. The disinfection operation is active.
4. Disinfection was not successful. Alarm is pending.
5. Disinfection was not successful, the alarm was acknowledged.
6. Controller restart, or legionella mode has not yet been requested.

bAlm: The temperature setpoint was not reached consistently over via the interval *udiTi_sec*, so that adequate disinfection is not guaranteed.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.4 System

Function blocks

Name	Description
FB_BA_CnvtTiSt [► 529]	Conversion year, month, day, hour, minute and second in time structure
FB_BA_ExtTiSt [► 530]	Conversion time structure in year, month, day, hour, minute and second
FB_BA_GetTime [► 531]	Internal clock with time information - can be synchronized with system time
FB_BA_SetTime [► 532]	Setting the system time
FB_BA_WrtPersistDat [► 534]	Writes persistent data

6.1.2.3.2.1.4.1 FB_BA_CnvtTiSt



The function block *FB_BA_CnvtTiSt* can be used to consolidate the different components of a time structure.

Note:



The function block does not check for incorrect entries, such as an hour entry of 99. It makes sense to check this in the connected function blocks, which have to check the time structure in any case. The limit values are shown as part of the variable explanations.

VAR_INPUT

```
wYear      : WORD;
wMonth     : WORD;
wDay       : WORD;
wHour      : WORD;
wMinute    : WORD;
wSecond    : WORD;
wMilliseconds : WORD;
```

wYear: The year (1970..2106).

wMonth: The month (1..12).

wDay: The day of the month (1..31).

wHour: The hour (0..23).

wMinute: The minutes (0..59).

wSecond s: The seconds (0..59).

wMillisecond: The milliseconds (0..999).

VAR_OUTPUT

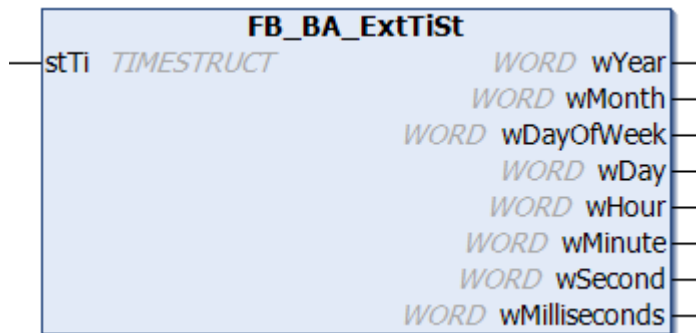
```
stTi      : TIMESTRUCT;
```

stTi: Output time structure (see TIMESTRUCT)

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.4.2 FB_BA_ExtTiSt



The function block *FB_BA_ExtTiSt* resolves a time structure into the different components, so that it can be used for time conditions, for example.

VAR_INPUT

```
stTi      : Timestruct;
```

stTi: Input time structure (see *Timestruct*)

VAR_OUTPUT

```
wYear      : WORD;
wMonth     : WORD;
wDayOfWeek : WORD;
wDay       : WORD;
wHour      : WORD;
wMinute    : WORD;
wSecond    : WORD;
wMilliseconds : WORD;
```

wYear: The year (1970..2106).

wMonth: The month (1..12).

wDayOfWeek: The day of the week (0(Sun)..0(Sat)).

wDay: The day of the month (1..31).

wHour: The hour (0..23).

wMinute: The minutes (0..59).

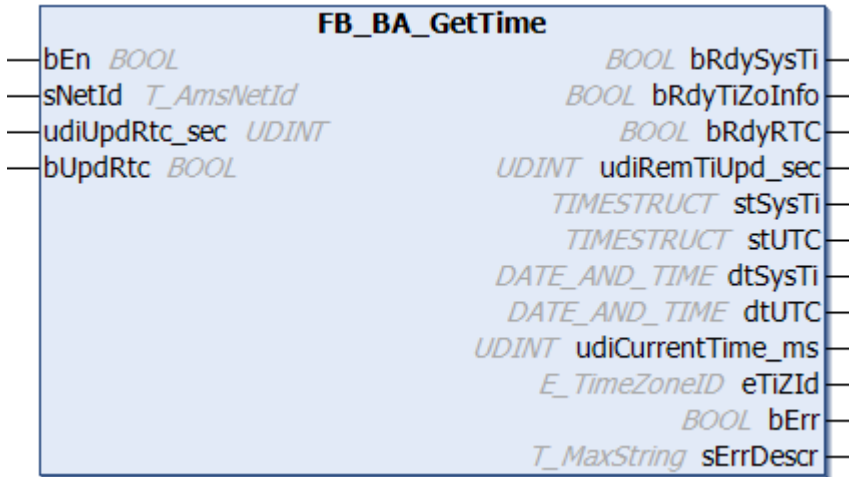
wSecond: The seconds (0..59).

wMilliseconds: The milliseconds (0..999).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.4.3 FB_BA_GetTime



With this function block an internal clock (Real Time Clock RTC) can be implemented in the TwinCAT PLC. When the function block is enabled via *bEn*, the RTC clock is initialized with the current NT system time. One system cycle of the CPU is used to calculate the current RTC time. The function block must be called once per PLC cycle in order for the current time to be calculated. Within the function block, an instance of the function blocks NT_GetTime, FB_GetTimeZoneInformation and RTC_EX2 is called. The time is output at the outputs *stSysTi* for the read system time and *stUtc* for the Coordinated Universal Time (UTC). This is determined internally from the system time and the time zone. If the system time and/or the time zone was entered incorrectly, the UTC time will also be wrong.

The system time is read cyclically via the timer to be set *udiUpdRTC_sec* [s]; it is used to synchronize the internal RTC clock. The time information (time zone, time shift relative to UTC, summer/winter time) is read in the same cycle. The output *udiRemTiUpd_sec* indicates the seconds remaining to the next read cycle. The time structures that are output, *stSysTi* and *stUtc*, can be resolved with the aid of the function block FB_BA_ExtTiSt [► 530] into the components day, month, hour, minute etc.

i Notes regarding read/wait cycle

During the read cycle, the outputs *bRdySysTi* and *bRdyTiZoInfo* change to FALSE, and the enumerator *eTiZId* shows 0 = *eTimeZoneID_Unknown*. If the read operation was successful, the outputs switch back to TRUE or show the respective information for summer or winter time, if available. If the read operation was unsuccessful - internally the system waits for a response for 5 seconds - the outputs remain at FALSE or 0, and another wait cycle is started before the next read cycle. Although the internal RTC clock is not synchronized in the event of an error and may still show the right time, the time information may be wrong, and therefore also the UTC time. Errors during the read cycle will, any case, show up in *bErr* and *sErrDescr*. The countdown output *udiRemTiUpd_sec* is not restarted until the wait cycle starts.

VAR_INPUT

```
bEn          : BOOL;
sNetId       : T_AmsNetId;;
udiUpdRtc_sec : UDINT;
bUpdRtc      : BOOL;
```

bEn: Enables the function block. If *bEn* = TRUE, then the RTC clock is initialized with the NT system time.

sNetId: This parameter can be used to specify the AmsNetId (see T_AmsNetId) of the TwinCAT computer whose NT system time is to be read as timebase. If it is to be run on the local computer, an empty string can be entered.

udiUpdRtc_sec: Time specification [s] with which the RTC clock is regularly synchronized with the NT system time. Internally this value is limited to a minimum of 5 seconds, in order to ensure correct processing of the internal function blocks.

bUpdRtc: In parallel with the time *udiUpdRtc_sec*, the RTC clock can be synchronized via a positive edge at this input.

VAR_OUTPUT

```

bRdySysTi      : BOOL;
bRdyTiZoInfo   : BOOL;
bRdyRTC        : BOOL;
udiRemTiUpd_sec : UDINT;
stSysTi        : TIMESTRUCT;
stUTC          : TIMESTRUCT;
dtSysTi        : DT;
dtUTC          : DT;
udiCurrentTime_ms : UDINT
eTiZId         : E_TimeZoneID;
bErr           : BOOL;
sErrDescr      : T_MAXSTRING;

```

bRdySysTi: The system time was read successfully from the target system.

bRdyTiZoInfo: The additional time information (time zone, time shift relative to UTC and summer/winter time) was read successfully.

bRdyRTC: This output is set if the function block has been initialized at least once. If this output is set, then the values for date, time and milliseconds at the outputs are valid.

udiRemTiUpd_sec: Countdown to next synchronization/update of the time information.

stSysTi: System time of the read target system (see TIMESTRUCT). The time structure can be resolved with the aid of the function block [FB_BA_ExtTiSt \[► 530\]](#) into its components: day, month, hour, minute etc.

Note: If the function block is not enabled (*bEn*=FALSE), the output *stSysTi* and its subelements (day month, etc.) show 0.

stUTC: Coordinated world time (see TIMESTRUCT). This is determined internally from the system time and the time information read from the target system. The time structure can be resolved with the aid of the function block [FB_BA_ExtTiSt \[► 530\]](#) into its components: day, month, hour, minute etc. **Note:** If the function block is not enabled (*bEn*=FALSE), the output *stUTC* and its subelements (day month, etc.) show 0.

dtSysTi / dtUTC: As *stSysTi* / *stUTC*, but in DATE-AND-TIME format: year-month-day-hours-minutes-seconds. Note: If the function block is not enabled (*bEn*=FALSE), the outputs *dtSysTi* and *dtUTC* show DT#1970-01-01-00:00, since this is the lower limit, which corresponds to the zeros in the structure representation of *stSysTi* / *stUTC*.

udiCurrentTime_ms: Current time of day in ms.

eTiZId: Enumerator for summer/winter time information (see E_TimeZoneID).

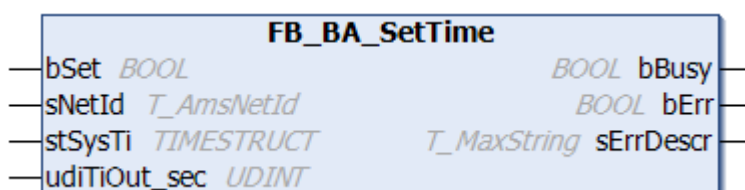
bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description
01: Warning: ADS error when reading the time (<i>FB_NT_GetTime</i>). The ADS error number is stated.
02: Warning: ADS error when reading the time zone information (<i>FB_GetTimeZoneInformation</i>). The ADS error number is stated.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

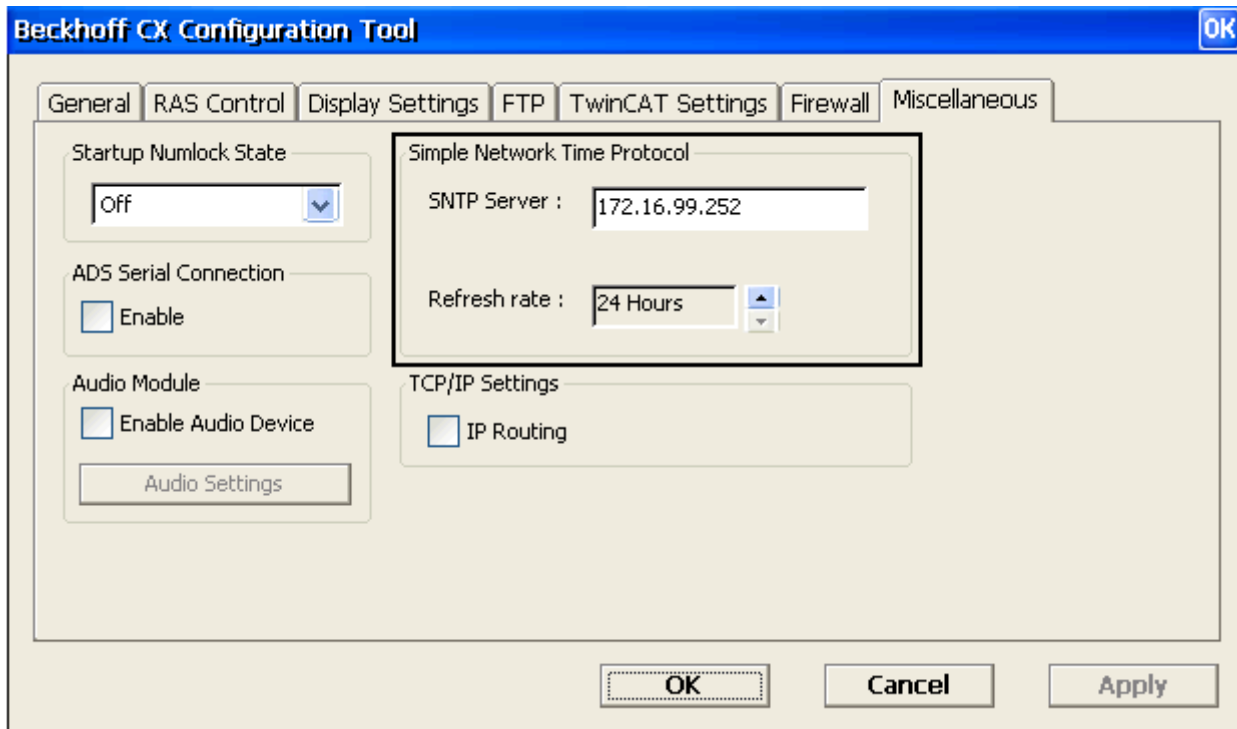
6.1.2.3.2.1.4.4 FB_BA_SetTime

The function block *FB_BA_SetTime* can be used to set the local NT system time and the date for a TwinCAT system (the local NT system time is shown in the taskbar). The system time is specified via the structure *stSysTi*.

Internally, an instance of the function block *NT_SetLocalTime* from the *TcUtilities* library is called in the function block.



The local NT system time can also be synchronized with a reference time with the aid of the SNTP protocol. For further information please refer to the Beckhoff Information System under: Beckhoff Information System > Embedded-PC > Operating systems > CE > SNTP: Simple Network Time Protocol



VAR_INPUT

```
bSet      : BOOL;
sNetId    : T_AmsNetId;
stSysTi   : TIMESTRUCT;
udiTiOut_sec : UDINT;
```

bSet: Activation of the function block with a rising edge.

sNetId: This parameter can be used to specify the AmsNetID of the TwinCAT computer, whose local NT system time is to be set. An empty string *sNetId* := ""; can also be specified for the local computer (see *T_AmsNetId*).

stSysTi: Structure with the new local NT system time (see *TIMESTRUCT*). If the time is not available as structure, it is advisable to use the function block *FB_BA_CnvTtSt* [► 529], which brings the subvariables of date and time in a structure together.

udiTiOut_sec: Indicates the timeout time [s], which must not be exceeded during execution.

VAR_OUTPUT

```
bBusy     : BOOL;
bError    : BOOL;
sErrorDescr : T_MAXSTRING;
```

bBusy: If the function block is activated via a rising edge at *bSet*, this output is set and remains set until feedback occurs.

bErr: This output is set to TRUE, if either the system time to be transferred is incorrect or an ADS error occurs during the transfer.

sErrDescr: Contains the error description.

Error description
01: Error: Error: range exceeded year
02: Error: Error: range exceeded month
03: Error: Error: range exceeded day of the month
04: Error: Error: range exceeded hour
05: Error: Error: range exceeded minute
06: Error: Error: range exceeded second
07: Error: Error: range exceeded millisecond
08: Warning: An ADS error occurred while setting the time (FB NT_SetLocalTime). The ADS error number is stated.

Time specification limits

The time structure *stUtcTi* that was created is internally checked for limits (see `TIMESTRUCT`)

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.4.5 FB_BA_WrtPersistDat



When activated, the function block *FB_BA_WrtPersistDat* first saves the persistent data in the Port_xxx.bootdata file. It is not necessary to explicitly specify the port or runtime system at which the PLC is located; this is determined internally. Once the data has been written, the content of the file Port_xxx.bootdata is copied to the backup file Port_xxx.bootdata-old. Thus both files are always synchronized. In case the original file with the persistent data is not readable, the backup copy, which is then read, contains the same data.



In any case, the checkmark "Clear Invalid Persistent Data" must be removed (see [Description of persistent data handling under TwinCAT 3](#) [► 535])

The function block can be started in two ways:

Via a positive edge at input *bStt*, if the function block is not in the set start-up phase.

Initially once the start-up phase is completed after a reset or TwinCAT restart. The duration is set at *udiInitSttDly_sec* in seconds. If "0" is entered there, the duration of the start-up phase is 0 and an initial execution of the function block is skipped.

No commands are accepted at *bStt* during the start-up phase.

If errors occur while reading, writing, opening or closing the files, this is indicated by a corresponding error message at *bErr/sErrDescr*. After an internally fixed waiting time of two seconds, the function block automatically attempts to execute the command (read, write, open or close) again.

It is therefore advisable to keep an eye on the error outputs or to evaluate them.

It is also important to note whether the backup file for the persistent data was loaded during the TwinCAT restart or after a reset. This indicates that the original file cannot be read and that the memory card of the controller is defective. It can be queried for each runtime system with the Boolean assignment of *TwinCAT_SystemInfoVarList._AppInfo.OldBootData* (see *PlcAppSystemInfo*).

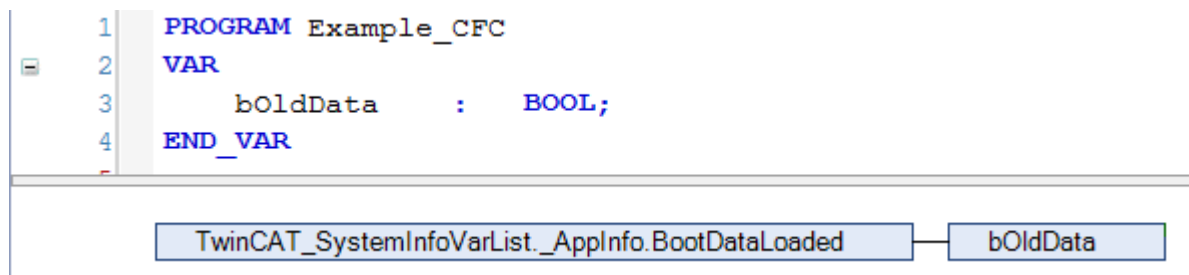
Sample in ST:

```

1  PROGRAM Example_ST
2  VAR
3      bOldData      :   BOOL;
4  END_VAR
5
6  bOldData:=TwinCAT_SystemInfoVarList._AppInfo.OldBootData;

```

Sample in CFC:



NOTICE

File handle conflict

Make sure that only this function block and only one instance of it accesses the persistent data. If several function blocks open a file and do not close it again, unforeseen file handle conflicts can occur which cannot be intercepted. The persistent data will then no longer be updated in the xxx.bootdata file.

Description of persistent data handling under TwinCAT 3

TwinCAT saves the persistent data for each runtime system in a file during each orderly shutdown, i.e. when switching from Run to Config or Stop mode.

The file name consists of the ADS port name of the runtime system with the file extension *.bootdata*, e.g.: *Port_851.bootdata* and is stored in the TwinCAT directory under *TwinCAT\3.1\Boot\PLC*.

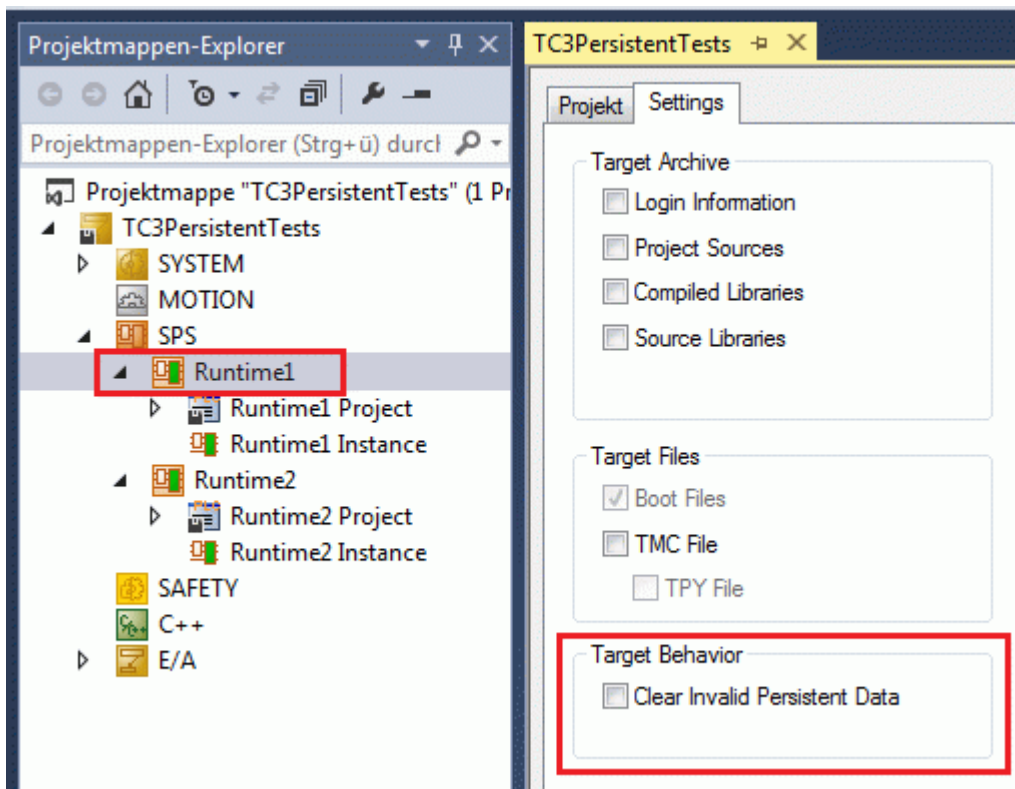
When the system is restarted, i.e. when switching to run mode, this file is read and then saved as *Port_xxx.bootdata-old*.

If the file *Port_xxx.bootdata-old* already exists, it is overwritten.

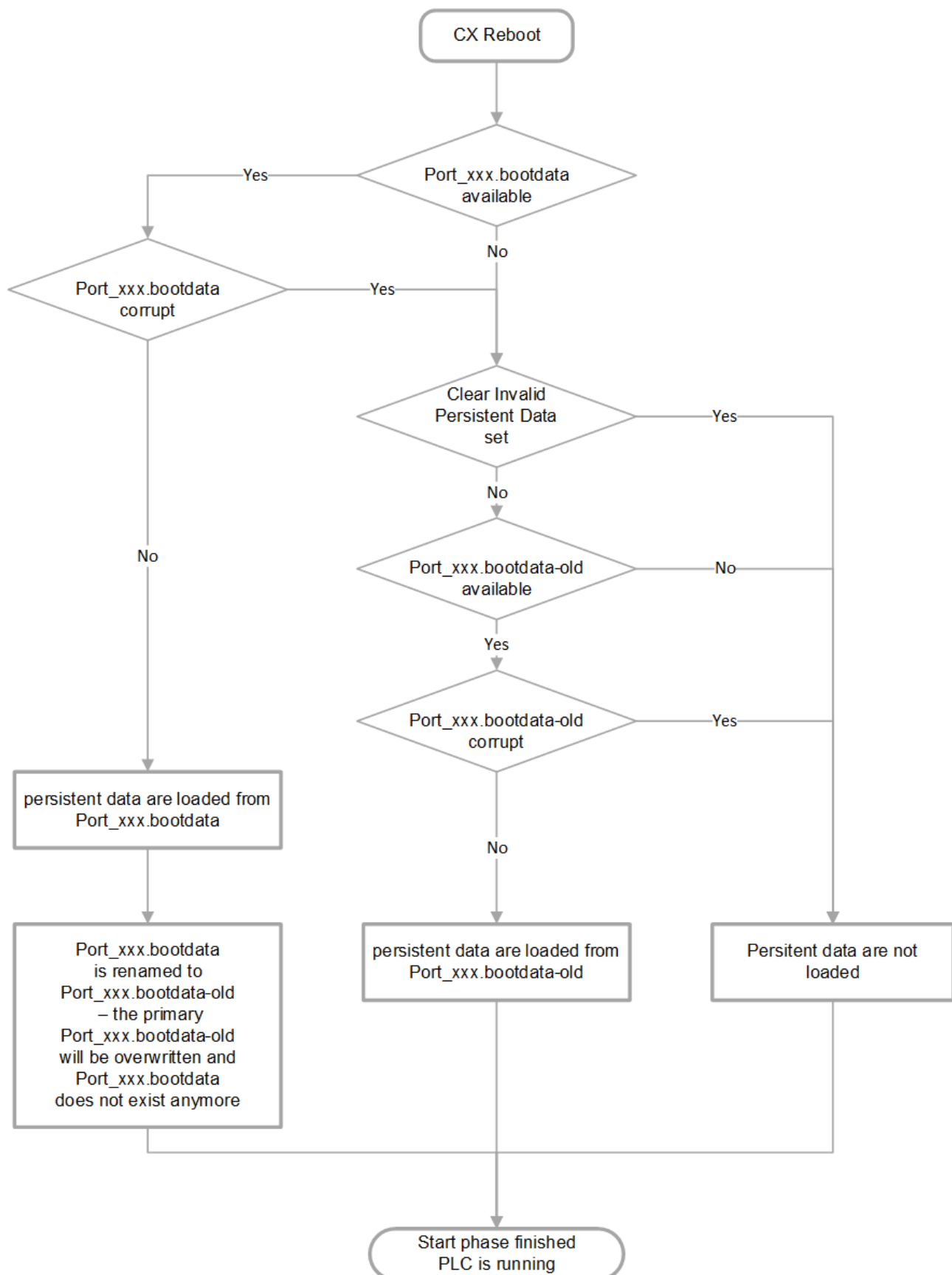
The original file *Port_xxx.bootdata* then no longer exists. It is created again automatically when switching to Stop mode or by the function block *FB_WritePersistentData* from the *TC2_Uilities* library.

This behavior applies to each runtime system; each system has its own files with persistent data.

If the file is defective when the TwinCAT system is restarted, the system automatically accesses the backup file *Port_xxx.bootdata-old*. However, this behavior only applies if the **Clear Invalid Persistent Data** checkmark is unchecked in the runtime settings. If it is checked and the original file is defective, no data will be read.



The Port_xxx.bootdata-old backup file is also used when the controller is de-energized. In this case, too, the current persistent data is not stored in Port_xxx.bootdata. When the system is restarted, only the old data is available, unless a more up-to-date file was created by the FB_WritePersistentData function block before the system was switched off.

**VAR_INPUT**

```

bStt      : BOOL;
udiInitSttDly_sec : UDINT;

```

bStt: A rising edge at this input starts the function block if it is not in the start-up phase.

udiInitSttDly_sec: Start-up phase after a reset or TwinCAT restart. The duration is set in seconds. Once the start-up phase has elapsed, the function block is automatically started once. No commands are accepted at *bStt* during the start-up phase. If "0" is set at *udiInitSttDly_sec*, the start-up phase is skipped. This input is preconfigured with 10 s.

VAR_OUTPUT

```
bBusy          : BOOL;
udiRemTiInitSttDly_sec : UDINT;
bErr           : BOOL;
sErrDescr      : T_MaxString;
```

bBusy: The function block is being executed.

udiRemTiInitSttDly_sec: Countdown of the set startup phase.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description
01: Error: The number of the ADS port issued by the PLC is "0"
02: Warning: Error when writing the persistent data via the internal function block <i>FB_WritePersistentData</i> . Additionally its error number.
03: Warning: Error when opening the backup file (xxx.bootdata-old) via the internal function block <i>FB_FileOpen</i> . Additionally its error number.
04: Warning: Error when reading the original file (xxx.bootdata) via the internal function block <i>FB_FileRead</i> . Additionally its error number
05: Warning: Error when writing to the backup file (xxx.bootdata-old) via the internal function block <i>FB_FileWrite</i> . Additionally its error number.
06: Warning: Error when closing the original file (xxx.bootdata) via the internal function block <i>FB_FileClose</i> . Additionally its error number.
07: Warning: Error when closing the backup file (xxx.bootdata-old) via the internal function block <i>FB_FileClose</i> . Additionally its error number.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

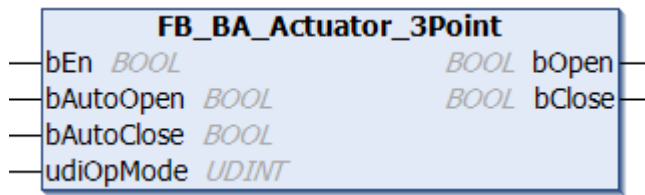
6.1.2.3.2.1.5 Universal

6.1.2.3.2.1.5.1 Actuators

Function blocks

Name	Description
FB_BA_Actuator_3Point [► 539]	Control of three-point dampers or valves
FB_BA_Anlg3Pnt [► 540]	Analog value for three-point converters
FB_BA_AntBlkg [► 541]	Blocking protection for pump or actuators
FB_BA_Motor1St [► 542]	Control of single-speed drives
FB_BA_Motor2St [► 543]	Control of two-speed drives
FB_BA_PWM [► 544]	Pulse width modulation function block

6.1.2.3.2.1.5.1.1 FB_BA_Actuator_3Point



The function block is used to control a 3-point actuator, e.g. a 3-point flap or a 3-point valve.

The command for opening the actuator is connected to output *bOpen*.

The command for closing the actuator is connected to output *bClose*.

In automatic mode (*udiOpMode*=0) the control commands of *bCmdOpen* and *bCmdClose* are forwarded directly to the outputs *bOpen* and *bClose*.

The *udiOpMode* input is used to determine the operating mode of the 3-point actuator:

- 0 = Automatic
- 1 = Stop (*bOpen* = *bClose* = FALSE)
- 2 = Close
- 3 = Open

VAR_INPUT

```

bEn      : BOOL;
bAutoOpen : BOOL;
bAutoClose : BOOL;
udiOpMode : UDINT
  
```

bEn: General enable of the function block.

bAutoOpen: Command to open the actuator.

bAutoClose: Command to close the actuator.

udiOpMode: Select operating mode (0 = Automatic, 1 = Stop (*bOpen* = *bClose* = FALSE), 2 = Close, 3 = Open)

VAR_OUTPUT

```

bOpen  : BOOL;
bClose : BOOL;
  
```

bOpen: Open control output.

bClose: Close control output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.1.2 FB_BA_Anlg3Pnt



The function block is intended for control of three-point actuators for valves or dampers.

A continuous control signal for positioning an actuator is converted into binary commands for opening and closing.

If the deviation between the set position value *rIn* and the calculated actual position value *rPos* of the actuator exceeds the set threshold value *rHys*/2, the function block starts to correct the position by switching the outputs *bOpn* or *bCls*, depending on the magnitude of the control deviation:

	bOpn	bCls
$rIn - rPos > rHys/2$	TRUE	FALSE
$rIn - rPos < -rHys/2$	FALSE	TRUE

If the function block reaches an end position *rOut*=0 or *rOut*=100 through a corresponding input value *rIn*, the corresponding switching output remains permanently set in order to safely reach this end position at the valve or damper:

	bOpn	bCls
<i>rOut</i> = 0	FALSE	permanently TRUE
<i>rOut</i> = 100	permanently TRUE	FALSE

Any deactivation of the continuous signal must be implemented by the user through external programming.

The input *rIn* is automatically limited to the range 0..100% internally.

This also applies to the entries *rHys* and *rRefVal*. The travel times *udiTiCls_ms* and *udiTiOpn_ms* both have a lower limit value of 10 (milliseconds).

A rising edge at *bRef* triggers a referencing command (the calculated actual position is set to *rRefVal*).

If the drive has limit switches, they can be sampled directly via the digital input and used for referencing at *bRef*.

VAR_INPUT

```

rIn      : REAL;
rHys     : REAL;
udiTiCls_ms : UDINT;
udiTiOpn_ms : UDINT;
bRef     : BOOL;
rRefVal  : REAL;
bCloseInit : BOOL;

```

rIn: Setpoint for the actuator position [0 - 100%]. Internally limited to values between 0 and 100.

rHys: Hysteresis for the actuator position [0 - 100%]. Internally limited to values between 0 and 100.

udiTiCls_ms: Run time of the actuator from open to closed [ms]. Internally limited to values between 0 and 100.

udiTiOpn_ms: Run time of the actuator from closed to open [ms]. Internally limited to values between 0 and 100.

bRef: Edge references the internal position memory of the drive to value of *rRefVal* [0 - 100%].

rRefVal: Value for referencing the actuator with *bRef* [0 - 100%]. Internally limited to values between 0 and 100.

bCloseInit: If this input is TRUE, output *bCIs* is TRUE for the time *udiTiOpn_ms*

VAR_OUTPUT

```
bCIs      : BOOL;
bOpn      : BOOL;
rPos      : REAL;
```

bCIs: Output for closing the actuator.

bOpn: Output for opening the actuator.

rPos: Current calculated actuator position [0 - 100%].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.1.3 FB_BA_AntBlkg



This function block prevents blocking of pumps or actuators after prolonged idle periods by issuing a switch-on pulse.

The maximum idle period before such a pulse is issued is determined by the value of *udiTiOffMin_sec*. For logging the idle time, the input *bFdb* must be linked to the operating feedback from the unit. The length of the pulse is parameterized with *udiTiImplLngt_sec*. The input *bExe* should be used if the blocking protection pulses are to be issued cyclically based on a switching schedule, rather than depending on the idle times. A rising edge at *bExe* immediately triggers output of a pulse to *bQ*. Generally, a pulse output only occurs if the function block at *bEn* is enabled.

VAR_INPUT

```
bEn        : BOOL;
bFdb       : BOOL;
bExe       : BOOL;
udiTiOffMin_sec : UDINT;
udiTiImplLngt_sec : UDINT;
```

bEn: Enable function block.

bFdb: Input for connecting the feedback signal of a motor or valve.

bExe: Rising edge forces a pulse output.

udiTiOffMin_sec: Minimum switch-off time [s]: a pulse is issued once the time *udiTiOffMin_sec* has elapsed without movement of the unit.

udiTiImplLngt_sec: Length of the blocking protection pulse [s] at *bQ*.

VAR_OUTPUT

```
bQ         : BOOL;
udiRTiOffMin_sec : UDINT;
udiRTiImplLngt_sec : UDINT;
```

bQ: Pulse output.

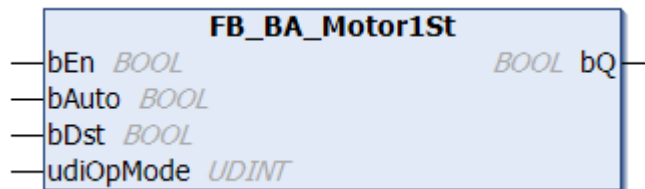
udiRTiOffMin_sec: Remaining time [s] before the next pulse is issued in the absence of movement.

udiTilmplLngt_sec: Remaining residual time [s] of the pulse at *bQ*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.1.4 FB_BA_Motor1St



Function block for controlling a simple single-stage motor.

The input *bEn* is used for general enabling of the motor.

The input *udiOpMode* is used to set the operating mode of the motor:

- 0 = Automatic
- 1 = Manual off
- 2 = Manual on

In automatic mode (*udiOpMode* = 0) the motor can be operated via the input *bAuto* (*bAuto* = *bQ* = TRUE).

The collection of all possible malfunctions of a motor is connected to *bDst*.

VAR_INPUT

```
bEn      : BOOL;
bAuto    : BOOL;
bDst     : BOOL;
udiOpMode : UDINT;
```

bEn: Enable motor.

bAuto: Request of the actuator in automatic mode (*udiOpMode* = 0).

bDst: Input for collecting the possible motor malfunctions.

udiOpMode: Select the operating mode (0 = Automatic, 1 = Manual off, 2 = Manual on).

VAR_OUTPUT

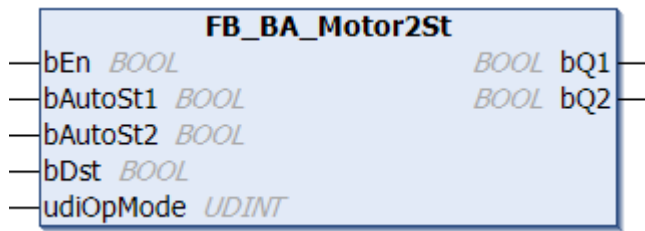
```
bQ : BOOL;
```

bQ: Control output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.1.5 FB_BA_Motor2St



Function block for controlling a simple two-stage motor.

The input *bEn* is used for general enabling of the motor.

The input *udiOpMode* is used to set the operating mode of the motor:

- 0 = Automatic
- 1 = Manual off
- 2 = Manual stage 1
- 3 = Manual stage 2

In automatic mode (*udiOpMode*= 0) the desired stage can be set via the inputs *bAutoSt1* (stage 1) and *bAutoSt2* (stage 2).

The collection of all possible malfunctions of a motor is connected to *bDst*.

VAR_INPUT

```
bEn      : BOOL;
bAutoSt1 : BOOL;
bAutoSt2 : BOOL;
bDst     : BOOL;
udiOpMode : UDINT;
```

bEn: Enable motor.

bAutoSt1: Request of the actuator at stage 1 in automatic mode (*udiOpMode*= 0).

bAutoSt2: Request of the actuator at stage 2 in automatic mode (*udiOpMode*= 0).

bDst: Input for collecting the possible motor malfunctions.

udiOpMode: Select the operating mode (0 = Automatic, 1 = Manual off, 2 = Manual stage 1, 3 = Manual stage 2).

VAR_OUTPUT

```
bQ1 : BOOL;
bQ2 : BOOL;
```

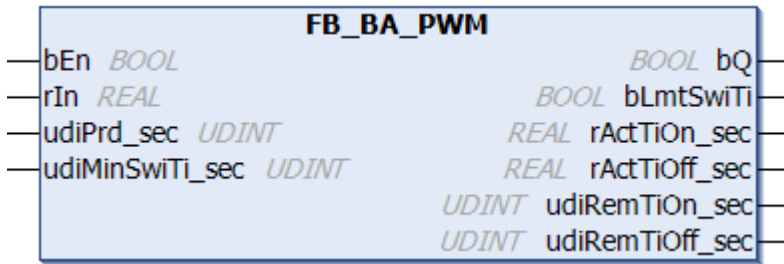
bQ1: Control output stage 1.

bQ2: Control output stage 2.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.1.6 FB_BA_PWM



The function block calculates switch-on and switch-off times *rActTiOn_sec* and *rActTiOff_sec* [s] from an analog input signal *rIn* (0..100%, **internally limited**) and the period *udiPrd_sec* [s].

The following relationships apply:

- 100% at the input of a switch-on time *rActTiOn_sec* of the total period *udiPrd_sec* and a switch-off time *rActTiOff_sec* of 0 s
- 0% at the input of a switch-on time *rActTiOn_sec* of 0 s and a switch-off time *rActTiOff_sec* of the total period *udiPrd_sec*.

In addition, *udiMinSwiTi_sec* [s] can be used to set a lower limit for the switching time, in order to prevent damage to drives caused by too short actuating pulses. This behavior is only valid for $0 < rIn < 100$!

If *rIn*=0 or 100, the output *bQ* remains deleted or set. After the period time has elapsed, the current input signal is evaluated again. If it is still set to 0 or 100, there is no change of state of *bQ*.

Switching characteristics

1. A FALSE signal at input *bEn* disables the function block and sets *bQ* to FALSE. Only the switch-on and switch-off times are continuously calculated and displayed at the outputs *rActTiOn_sec*/*rActTiOff_sec* [s].
2. A rising edge at input *bEn* enables the function block: It will initially jump to a decision step. Depending on the previous state of the switching output *bQ*, the switching step is now accessed. However, if the input *rIn* is set to 0, an immediate jump occurs to the Off step (*bQ*=FALSE), or to the On step if *rIn*=100 (*bQ*=TRUE), irrespective of the previous state of *bQ*. The minimum switching time is deactivated for these two cases.
3. A countdown timer with the current calculated starting value runs in the respective active step (ON or OFF), which is based on the pulse/pause ratio. The on- or off-step is completed with the calculated time, irrespective of whether the pulse/pause ratio changes in the meantime. The respective countdown is displayed at the outputs *udiRemTiOn_sec*/*udiRemTiOff_sec* in full seconds.
4. Completion of the on- or off-step is followed by a jump back to the decision step (point 2).

VAR_INPUT

```
bEn      : BOOL;
rIn      : REAL;
udiPrd_sec : UDINT;
udiMinSwiTi_sec : UDINT;
```

bEn: Activation of pulse width modulation.

rIn: Input signal, internally limited to 0..100%.

udiPrd_sec: Period time[s]. Internally limited to a minimum value of 0.

udiMinSwiTi_sec: Minimum switch-on time [s], to avoid too short pulses. Internally limited to values between 0 and *udiPrd_sec*.

VAR_OUTPUT

```
bQ      : BOOL;
bLmtSwiTi : BOOL;
rActTiOn_sec : REAL;
rActTiOff_sec : REAL;
udiRemTiOn_sec : UDINT;
udiRemTiOff_sec : UDINT;
```

bQ: PWM output.

bLmtSwiTi: Information output to indicate that the input signal is so low that the minimum switch-on time is used as limit.

rActTiOn_sec: Information output: Calculated switch-on time.

rActTiOff_sec: Information output: Calculated switch-off time.

udiRemTiOn_sec: Switch-on timer countdown.

udiRemTiOff_sec: Switch-off timer countdown.

Requirements

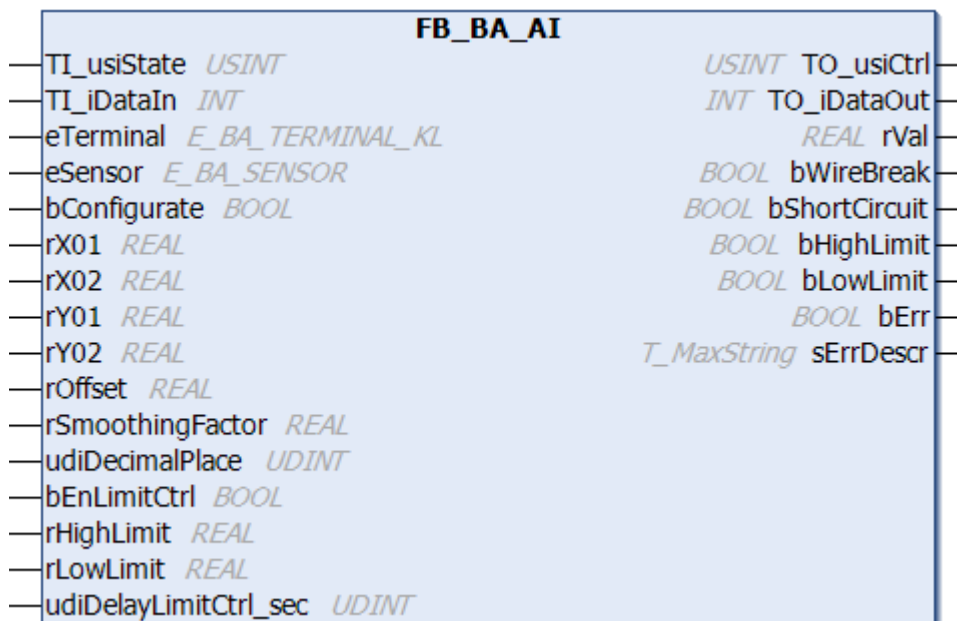
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.2 Analog inputs/outputs

Function blocks

Name	Description
FB_BA_AI [► 545]	Acquisition of analog input signals
FB_BA_AO [► 548]	Control of analog actuators with integrated scaling function
FB_BA_KL32xxConfig	Parameterization of the connected sensor type on an input channel from the PLC

6.1.2.3.2.1.5.2.1 FB_BA_AI



The function block is used for measured data processing and terminal configuration of all standard K-bus analog input terminals.

Terminal configuration

The first step when using this function block is to select the corresponding terminal type *eTerminal*. Correct functioning of the function block is only guaranteed if the terminal selected with the *eTerminal* variable matches the terminal actually inserted and linked.

With the terminals for resistance temperature measurement of type KL3208_0010 and KL320x_0000, the

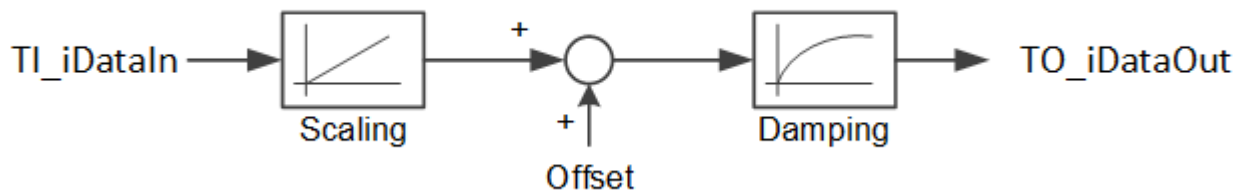
temperature sensor used at the terminal input is additionally selected with the enumeration `eSensor`. A rising edge at `bConfigure` writes the terminal settings to the terminal using the variables `TO_usiCtrl` and `TO_iDataOut`.

Measured value scaling

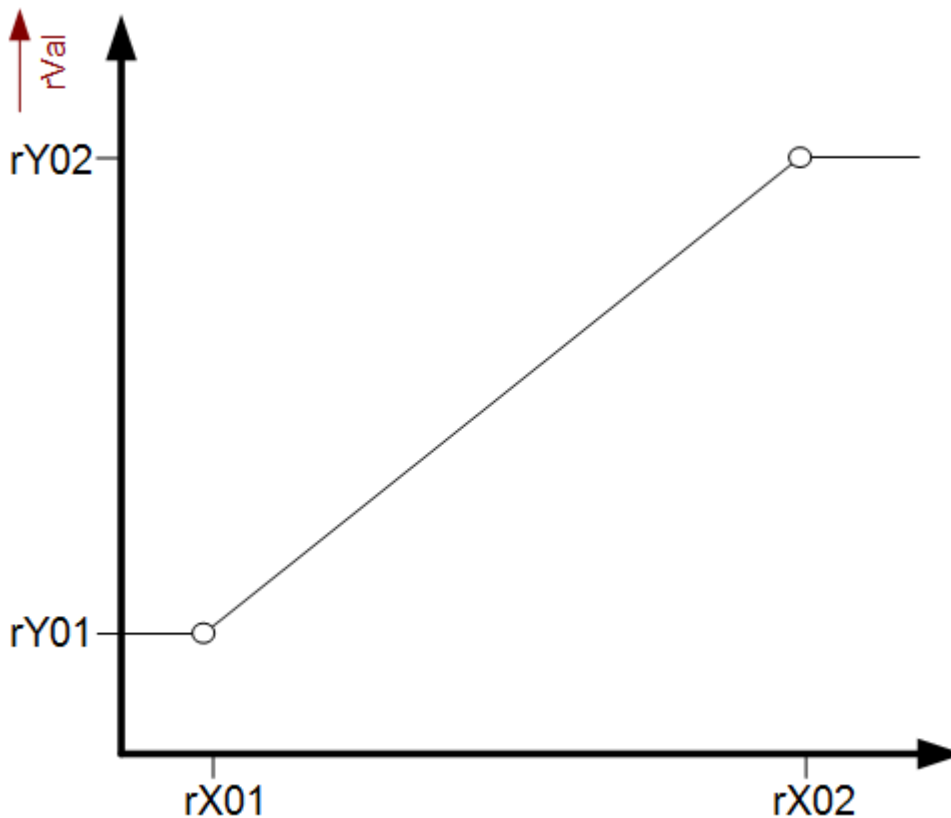
At the analog input terminals of types:

KL300x, KL306x, KL3132_0000, KL3162_0000, KL3172_0000, KL3172_0500, KL3172_1000, KL3182_0000, KL3404, KL3464, KL3408, KL3468, the raw value `TI_iDataIn` of the terminal is scaled by the internal function block `FB_BA_Chrc02` [► 583]. (See diagram). The parameterization of the scaling function is carried out using parameters `X(01/02)` and `Y(01/02)`. A signal offset that may be required for measured value correction can be parameterized with the variable `rOffset`. `rSmoothingFactor` attenuates the scaled measurement signal, including the offset.

Signal flow:

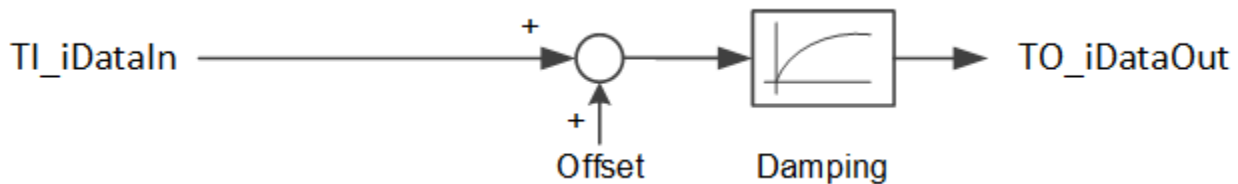


Scaling:



With the temperature measurement terminals of types KL3208_0010 and KL320x_0000, the measuring signal is not scaled within the function block, but directly in the terminal. Attenuation of excessively fluctuating measuring signals takes place with the factor `rSmoothingFactor`. If an error is detected at the terminal, such as `bShortCircuit` or `bWireBreak`, the last valid value is output until the error is eliminated at the output of the function block.

Signal flow:



Measuring range monitoring

The variables *rHighLimit* and *rLowLimit* are used to monitor the measured value *rVal* at the output of the function block for compliance with a valid range. If the measured value *rVal* is above *rHighLimit* or below *rLowLimit*, this is displayed at the outputs *bHighLimit* or *bLowLimit*. Measuring range monitoring is deactivated if the input variable *bEnLimitCtrl* is FALSE or if an error (e.g. short circuit) is detected by means of the terminal status *TI_usiState*.

The response of the measuring range monitoring can be delayed by the variable *udiDelayLimitCtrl_sec*.

VAR_INPUT

```

TI_usiState      : USINT;
TI_iDataIn       : INT;
eTerminal        : E_BA_TERMINAL_KL;
eSensor          : E_BA_SENSOR;
bConfigure       : BOOL;
rX01             : REAL;
rX02             : REAL;
rY01             : REAL;
rY02             : REAL;
rOffset          : REAL;
rSmoothingFactor : REAL;
udiDecimalPlace  : UDINT(1..6);
bEnLimitCtrl     : BOOL;
rHighLimit       : REAL;
rLowLimit        : REAL;
udiDelayLimitCtrl_sec : UDINT;
  
```

TI_usiState: Linking with the corresponding status byte of the Bus Terminal in the I/O area of the program.

TI_iDataIn: Linking with the corresponding raw data (Data In) of the Bus Terminal in the I/O area of the program (0..32767).

eTerminal: Selection of the respective Bus Terminal (see E_BA_Terminal_KL).

eSensor: Selection of the sensor type (see E_BA_Sensor).

bConfigure: A rising edge starts the configuration of the Bus Terminal.

rX01: x-value for the interpolation point P1.

rX02: x-value for the interpolation point P2.

rY01: y-value for the interpolation point P1.

rY02: y-value for the interpolation point P2.

rOffset: Offset.

rSmoothingFactor: attenuation factor. Internally limited to between 1 and 10000.

udiDecimalPlace: Specifies the decimal places for the value *rVal*. Preset to 1.

bEnLimitCtrl: Enable limit value monitoring.

rHighLimit: Upper limit value.

rLowLimit: Lower limit value.

udiDelayLimitCtrl_sec: Time delay until limit value monitoring is activated.

VAR_OUTPUT

```

TO_usiCtrl    : USINT;
TO_iDataOut   : INT;
rVal          : REAL;
bWireBreak    : BOOL;
bShortCircuit : BOOL;
bHighLimit    : BOOL;
bLowLimit     : BOOL;
bErr          : BOOL;
sErrDescr     : T_MAXSTRING;

```

TO_usiCtrl: Linking with the corresponding control byte of the Bus Terminal in the I/O area of the program.

TO_iDataOut: Linking with the corresponding raw data (Data Out) of the Bus Terminal in the I/O area of the program.

rVal: Scaled output value.

bWireBreak: Broken wire at the sensor.

bShortCircuit: Short-circuit at the sensor.

bHighLimit: Upper limit value exceeded.

bLowLimit: Value below lower limit.

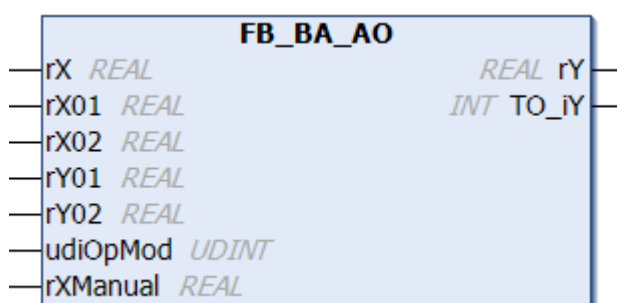
bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

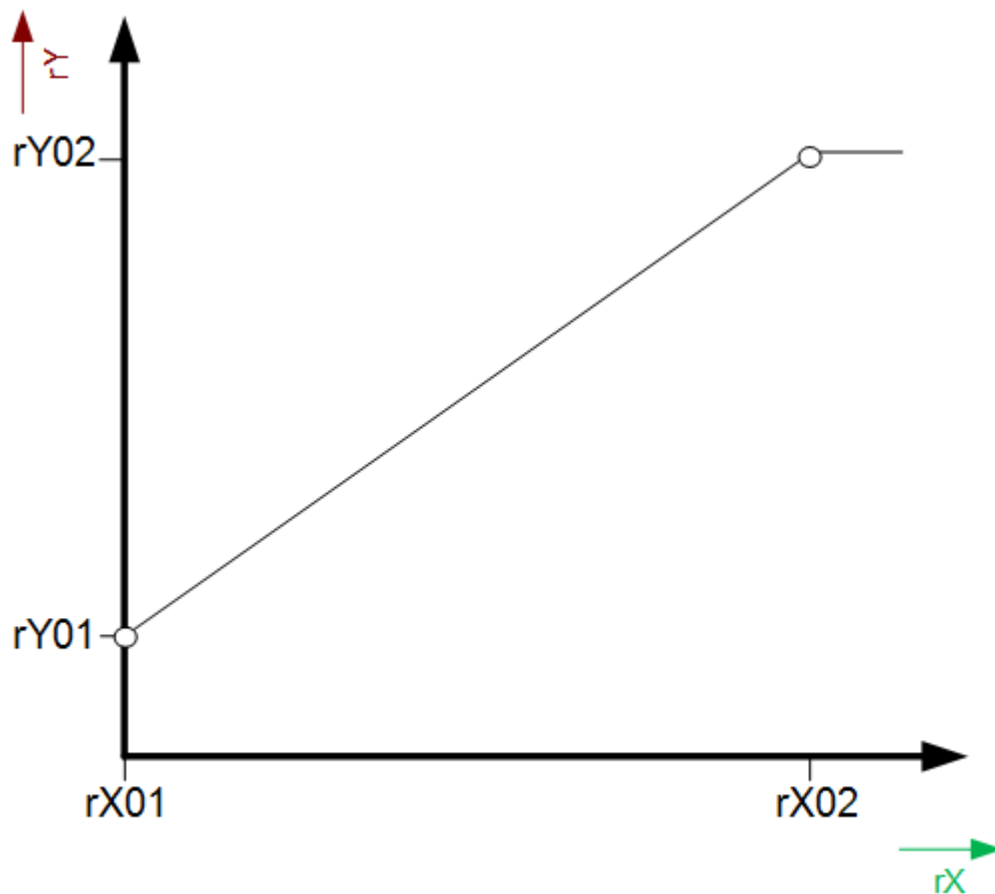
Error description
01: Error: Incorrect scaling parameter <i>rX01/rX02/rY01/rY02</i>
02: Error: Check the terminal configuration <i>KL32xx eTerminal/eSensor/TL_usiState/TL_iDataIn/TO_usiCtrl/IO_iDataOut</i>

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.2.2 FB_BA_AO

The function block *FB_BA_AO* is used for output and scaling of an analog output. The input value at *rX* is converted into an output value from 0 to 32767 or 0-10 Volt or 4-20 mA by means of a linear equation.



The operating mode is set via the input *udiOpMod*:

- 0 = Automatic
- 1 = Manual

In automatic mode (*udiOpMod*= 0) the input value *rX* is passed on.

In manual mode (*udiOpMod*= 1) the input value *rXManual* is passed on.

VAR_INPUT

```
rX      : REAL;
rX01    : REAL;
rX02    : REAL;
rY01    : REAL;
rY02    : REAL;
udiOpMod : UDINT(0..1);
rXManual : REAL;
```

rX: Input value of the process in automatic mode.

rX01: x-value for the interpolation point P1.

rX02: x-value for the interpolation point P2.

rY01: y-value for the interpolation point P1.

rY02: y-value for the interpolation point P2.

udiOpMod: Selection of the operating mode (0 = Automatic, 1 = Manual).

rXManual: Input value for manual operation.

VAR_OUTPUT

```
rY      : REAL;
TO_iY   : INT;
```

rY: Output signal as floating point number.

TO_iY: Output signal as integer value for linking to the output value of the bus terminal in the I/O section of the program.

Requirements

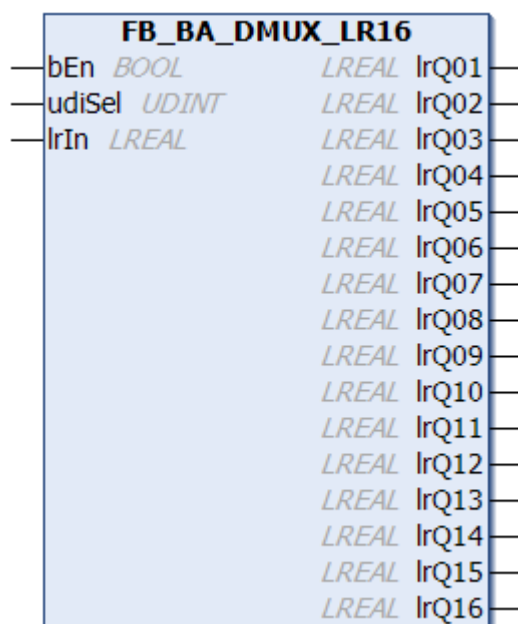
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3 General control functions

Function blocks

Name	Description
FB_BA_DMUX_XX [► 550]	Demultiplexer function blocks
FB_BA_MMUX_XX [► 553]	The function blocks activate an input value on the output, depending on a selector and the corresponding input selector condition
FB_BA_MUX_XX [► 556]	Multiplexer function blocks
FB_BA_PrioSwi_XX [► 558]	Priority switch
FB_BA_Blink [► 560]	Simple oscillator function block
FB_BA_FIFO04 [► 561]	Sequential control of up to four units
FB_BA_FIFO08 [► 563]	Sequential control of up to eight units
FB_BA_StepCtrl08 [► 565]	Step sequence function block, 8 steps
FB_BA_StepCtrl12 [► 568]	Step sequence function block, 12 steps
FB_BA_FIFO04_XX [► 571]	
FB_BA_FIFO08_XX [► 572]	

6.1.2.3.2.1.5.3.1 FB_BA_DMUX_XX



Demultiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different output parameters (4, 8, 12 and 16), but they all have the same functionality.

The function block FB_BA_DMUX_LR16 is described as an example.

In active state (*bEn*=TRUE), the function block outputs the value at input *IrIn* at the output (*IrQ01..IrQ16*) whose number is entered at input *udiSel*. All other outputs are set to 0 (for boolean demultiplexers to FALSE).

Example:

Inputs	Outputs
bEn = TRUE	IrQ01 = 0.0
udiSel = 5	IrQ02 = 0.0
IrIn = 32.5	IrQ03 = 0.0
	IrQ04 = 0.0
	IrQ05 = 32.5
	IrQ06 = 0.0
	IrQ07 = 0.0
	IrQ08 = 0.0
	IrQ09 = 0.0
	IrQ10 = 0.0
	IrQ11 = 0.0
	IrQ12 = 0.0
	IrQ13 = 0.0
	IrQ14 = 0.0
	IrQ15 = 0.0
	IrQ16 = 0.0

If the value entered at *udiSel* is greater than the number of outputs, the value of *IrIn* is output at the "highest" output:

Inputs	Outputs
bEn = TRUE	IrQ01 = 0.0
udiSel = 25	IrQ02 = 0.0
IrIn = 32.5	IrQ03 = 0.0
	IrQ04 = 0.0
	IrQ05 = 0.0
	IrQ06 = 0.0
	IrQ07 = 0.0
	IrQ08 = 0.0
	IrQ09 = 0.0
	IrQ10 = 0.0
	IrQ11 = 0.0
	IrQ12 = 0.0
	IrQ13 = 0.0
	IrQ14 = 0.0
	IrQ15 = 0.0
	IrQ16 = 32.5

If *bEn* = FALSE, 0.0 is output at all outputs, or FALSE for boolean demultiplexers.

VAR_INPUT

```
bEn      : BOOL;
udiSel   : UDINT;
rIn      : LREAL;
```

bEn: Activation of the block function.

udiSel: Number of the output (*lrQ00...lrQ16*), which is to take on the value of input *lrIn*.

lrIn: Value to be output.

VAR_OUTPUT

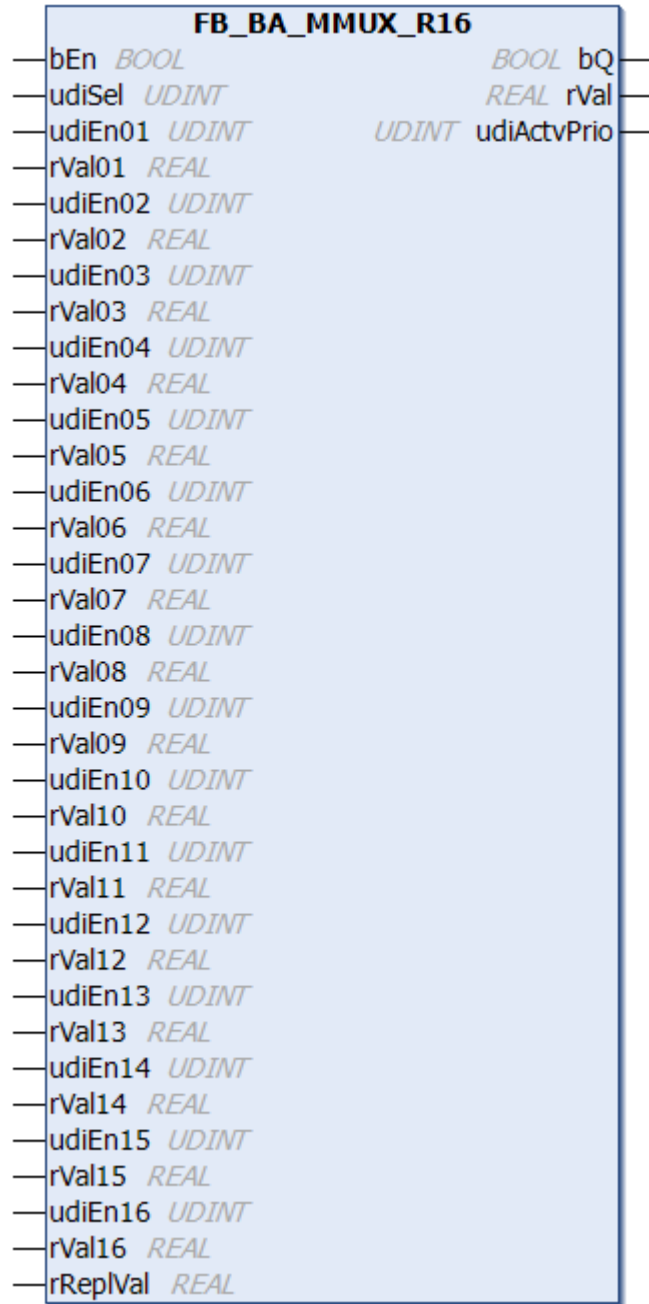
```
lrQ00 : LREAL;  
lrQ01 : LREAL;  
lrQ02 : LREAL;  
lrQ03 : LREAL;  
lrQ04 : LREAL;  
lrQ05 : LREAL;  
lrQ06 : LREAL;  
lrQ07 : LREAL;  
lrQ08 : LREAL;  
lrQ09 : LREAL;  
lrQ10 : LREAL;  
lrQ11 : LREAL;  
lrQ12 : LREAL;  
lrQ13 : LREAL;  
lrQ14 : LREAL;  
lrQ15 : LREAL;  
lrQ16 : LREAL;
```

lrQ00...lrQ16: Value outputs.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.2 FB_BA_MMUX_XX



The function block activates an input value on the output, depending on a selector and the corresponding input selector condition.

Multiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different input parameters (4, 8, 12, 16 and 24), but they all have the same functionality.

The function block FB_BA_MMUX_R16 is described as an example.

In active state (*bEn=TRUE*), the function block activates one of the input values *rValxx* at output *rVal*, depending on a selector *udiSel* and the corresponding input selector condition *udiEnxx*.

If several input selector conditions *udiEn01...udiEn16* are identical and the selector *udiSel* matches a condition, the input value *rVal01...rVal16* of the lowest active selector condition is activated at output *rVal*. *udiEn01* is the lowest selector condition, *udiEn16* the highest.

The output variable *bQ* indicates that the selector *udiSel* matches the input selector condition *udiEnxx*.

The output variable *udiActvPrio* indicates the active selector condition.

If no selector condition is active, *rRepVal* is output at *rVal*. *bQ* is then FALSE and *udiActvPrio* shows 255.

Example:

Inputs		Output	
Variable	Value	Variable	Value
bEn	TRUE	bQ	TRUE
udiSel	5	rVal	1.123
udiEn01	4	udiActvPrio	7
rVal01	123		
udiEn02			
rVal02			
udiEn03	3		
rVal03	321		
udiEn04			
rVal04			
udiEn05	8		
rVal05	345		
udiEn06			
rVal06			
udiEn07	5		
rVal07	1.123		
udiEn08			
rVal08			
udiEn09	5		
rVal09	5.4321		
udiEn10			
rVal10			
udiEn11			
rVal11			
udiEn12			
rVal12			
udiEn13			
rVal13			
udiEn14			
rVal14			
udiEn15			
rVal15			
udiEn16			
rVal16			
rRepVal			

If no priority is active, the value of the global constant Const.udiNoActvPrio [► 605] is output at *udiActvPrio*.

VAR_INPUT

```

bEn      : BOOL;
udiSel    : UDINT;
udiEn01   : UDINT := Const.udiNoActvPrio;
rVal01    : REAL;
udiEn02   : UDINT := Const.udiNoActvPrio;
rVal02    : REAL;
udiEn03   : UDINT := Const.udiNoActvPrio;
rVal03    : REAL;
udiEn04   : UDINT := Const.udiNoActvPrio;

```

```

rVal04      : REAL;
udiEn05     : UDINT := Const.udiNoActvPrio;
rVal05      : REAL;
udiEn06     : UDINT := Const.udiNoActvPrio;
rVal06      : REAL;
udiEn07     : UDINT := Const.udiNoActvPrio;
rVal07      : REAL;
udiEn08     : UDINT := Const.udiNoActvPrio;
rVal08      : REAL;
udiEn09     : UDINT := Const.udiNoActvPrio;
rVal09      : REAL;
udiEn10     : UDINT := Const.udiNoActvPrio;
rVal10      : REAL;
udiEn11     : UDINT := Const.udiNoActvPrio;
rVal11      : REAL;
udiEn12     : UDINT := Const.udiNoActvPrio;
rVal12      : REAL;
udiEn13     : UDINT := Const.udiNoActvPrio;
rVal13      : REAL;
udiEn14     : UDINT := Const.udiNoActvPrio;
rVal14      : REAL;
udiEn15     : UDINT := Const.udiNoActvPrio;
rVal15      : REAL;
udiEn16     : UDINT := Const.udiNoActvPrio;
rVal16      : REAL;
rReplVal    : REAL;

```

bEn: Activation of the block function.

udiSel: Selector. Internally limited to values between 0 and 4294967294.

udiEn01..udiEn16: Input selector condition.
The input variables are pre-initialized to the value 255.

rVal01...rVal16: Input values to select from.

rReplVal: Substitute value, if no input selector condition is active

VAR_OUTPUT

```

bQ          : BOOL;
rVal        : REAL;
udiActvPrio : UDINT;

```

bQ: TRUE, if the selector *udiSel* matches an input selector condition *udiEnxx*.

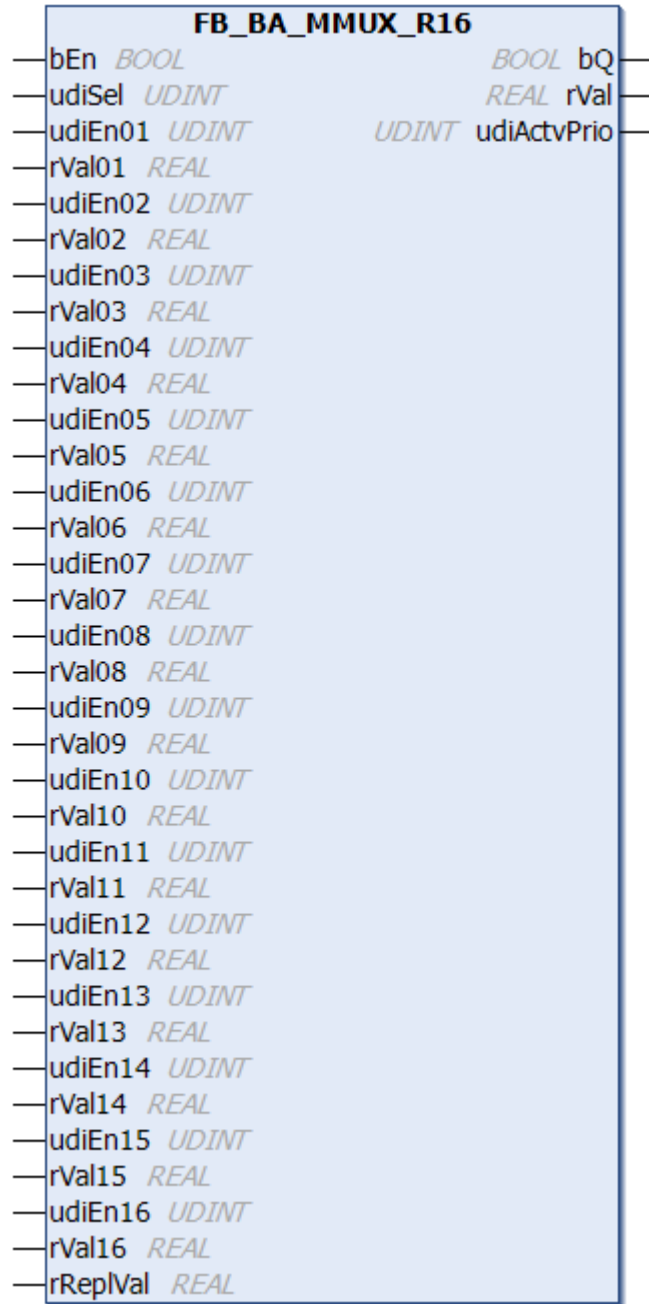
rVal: Value of the selected input selector condition.

udiActvPrio: Indicates which input selector condition is active. If no priority is active, the value of the global constant Const.udiNoActvPrio [► 605] is output at *udiActvPrio*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.3 FB_BA_MUX_XX



Multiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different input parameters (4, 8, 12 and 16), but they all have the same functionality.

The function block FB_BA_MUX_LR16 is described as an example.

In active state ($bEn=TRUE$), the function block outputs the input value ($IrIn01..IrIn16$) at output IrQ , whose number is entered at input $udiSel$.

Example:

Inputs	Output
bEn = TRUE	IrQ = 16.5
udiSel = 5	
lrIn01 = 15.9	
lrIn02 = 32.5	
lrIn03 = 17.4	
lrIn04 = 5.84	
lrIn05 = 9.56	
lrIn06 = 16.5	
lrIn07 = 32.781	
lrIn08 = 25.4	
lrIn09 = 44.5	
lrIn10 = 66.1	
lrIn11 = 45.5	
lrIn12 = 83.3	
lrIn13 = 54.56	
lrIn14 = 33.8	
lrIn15 = 98.5	
lrIn16 = 71.3	

If the value entered at *udiSel* is greater than the number of inputs, the "highest-ranking" input is output at *IrQ*:

Inputs	Output
bEn = TRUE	IrQ = 2.3
udiSel = 25	
lrIn01 = 15.9	
lrIn02 = 32.5	
lrIn03 = 17.4	
lrIn04 = 5.84	
lrIn05 = 9.56	
lrIn06 = 16.5	
lrIn07 = 32.781	
lrIn08 = 25.4	
lrIn09 = 44.5	
lrIn10 = 66.1	
lrIn11 = 45.5	
lrIn12 = 83.3	
lrIn13 = 54.56	
lrIn14 = 33.8	
lrIn15 = 98.5	
lrIn16 = 71.3	

If *bEn*=FALSE, 0.0 is output at *IrQ*, or FALSE for boolean multiplexers.

VAR_INPUT

```

bEn      : BOOL;
udiSel   : UDINT;
lrIn00   : LREAL;
lrIn01   : LREAL;
lrIn02   : LREAL;
lrIn03   : LREAL;
lrIn04   : LREAL;
lrIn05   : LREAL;
lrIn06   : LREAL;
lrIn07   : LREAL;

```

```
lrIn08 : LREAL;  
lrIn09 : LREAL;  
lrIn10 : LREAL;  
lrIn11 : LREAL;  
lrIn12 : LREAL;  
lrIn13 : LREAL;  
lrIn14 : LREAL;  
lrIn15 : LREAL;  
lrIn16 : LREAL;
```

bEn: Activation of the block function.

udiSel: Number of the input, whose value is to be output at lrQ .

lr00...lr16: Input values to select from.

VAR_OUTPUT

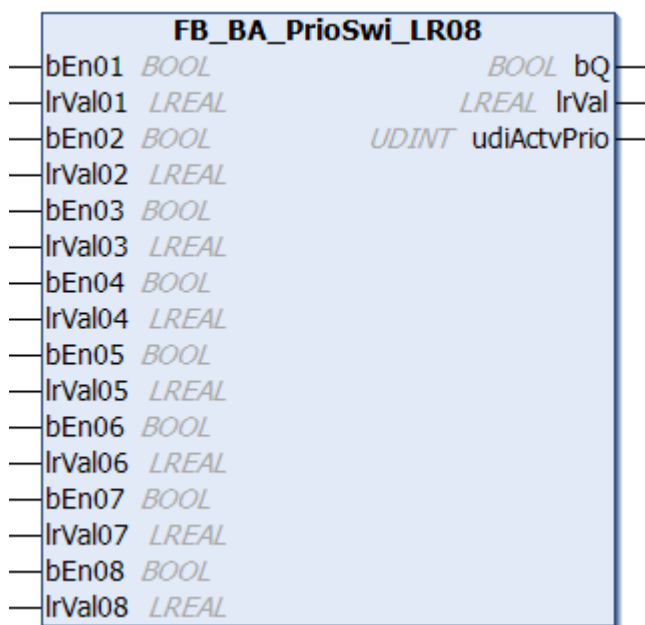
```
lrQ      : LREAL;
```

IrQ: Value of the selected input.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3 BA from v1.1.6.0

6.1.2.3.2.1.5.3.4 FB_BA_PrioSwi_XX



The priority switches exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different output sizes (4, 8, 12 and 16 or 24), but they all have the same functionality.

The function block FB BA PrioSwi LR08 is described as an example.

Priority switches are available for selecting different values. At output *IrVal* the value with the highest priority is applied whose input *bEnxx* is TRUE.

Example:

Inputs			Outputs		
bEn01	FALSE		bQ	TRUE	
lrVal01		32.5	lrVal		5.84
bEn02	FALSE		udiActvPrio		3
lrVal02		17.4			
bEn03	TRUE				
lrVal03		5.84			
bEn04	TRUE				
lrVal04		9.56			
bEn05	FALSE				
lrVal05		16.5			
bEn06	TRUE				
lrVal06		32.781			
bEn07	FALSE				
lrVal07		25.4			
bEn08	TRUE				
lrVal08		44.5			

If none of the priorities is enabled, the output *bQ* switches to FALSE. 0 is output at *lrVal* and *udiActvPrio*. For a boolean priority switch, FALSE is then output at *bVal*.

Inputs			Outputs		
bEn01	FALSE		bQ	FALSE	
lrVal01		32.5	lrVal		0.0
bEn02	FALSE		udiActvPrio		0
lrVal02		17.4			
bEn03	FALSE				
lrVal03		5.84			
bEn04	FALSE				
lrVal04		9.56			
bEn05	FALSE				
lrVal05		16.5			
bEn06	FALSE				
lrVal06		32.781			
bEn07	FALSE				
lrVal07		25.4			
bEn08	FALSE				
lrVal08		44.5			

If no priority is active, the value of the global constant *ConstudiNoActvPrio* is output at *udiActvPrio*.

VAR_INPUT

```

bEn01   : BOOL;
lrVal01 : LREAL;
bEn02   : BOOL;
lrVal02 : LREAL;
bEn03   : BOOL;
lrVal03 : LREAL;
bEn04   : BOOL;
lrVal04 : LREAL;
bEn05   : BOOL;
lrVal05 : LREAL;
bEn06   : BOOL;
lrVal06 : LREAL;
bEn07   : BOOL;

```

```
lrVal07 : LREAL;
bEn08   : BOOL;
lrVal08 : LREAL;
```

bEn01...bEn08: Enabling the priority value.

lrVal01...lrVal08: Priority value.

VAR_OUTPUT

```
bQ      : BOOL;
lrVal   : LREAL;
udiActvPrio : UDINT;
```

bQ: Output to indicate whether a priority is enabled.

lrVal: Output of the value of the current (highest) priority that is enabled.

udiActvPrio: Current (highest) priority that is enabled.

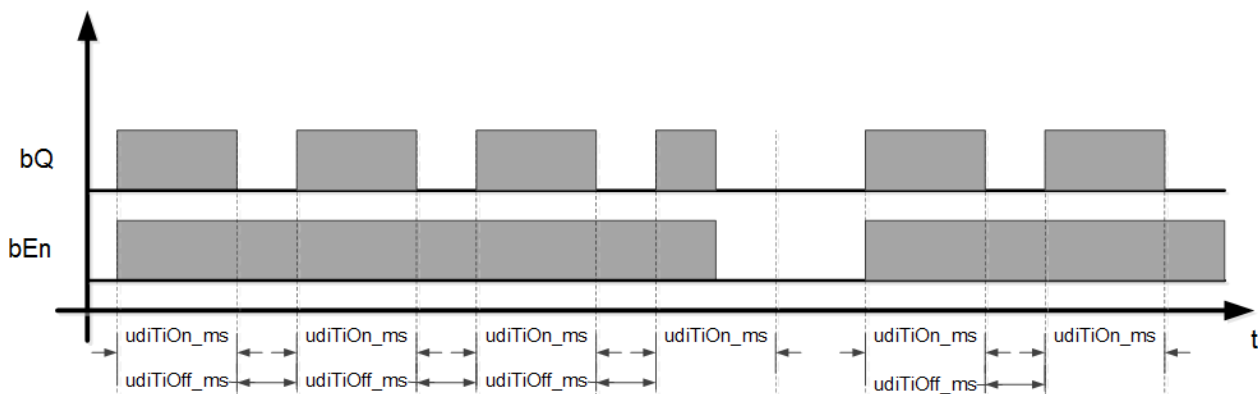
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.5 FB_BA_Blink



This function block is an oscillator with adjustable pulse and pause time, *udiTiOn_ms* and *udiTiOff_ms* [ms]. It is enabled with a TRUE signal at *bEn* and starts with the pulse phase.



udiTiNextSwi_sec is a countdown [s] to the next change of *bQ*.

VAR_INPUT

```
bEn      : BOOL;
udiTiOn_ms : UDINT;
udiTiOff_ms : UDINT;
```

bEn: Function block enable.

udiTiOn_ms: pulse time [ms].

udiTiOff_ms: pause time [ms].

VAR_OUTPUT

```
bQ      : BOOL;
udiTiNextSwi_sec : UDINT;
```

bQ: Oscillator output.

udiTiNextSwi_sec: Countdown to next change of *bQ* [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.6 FB_BA_FIFO04



The function block *FB_BA_FIFO04* enables sequential control of up to four units, with automatic switching of the switch-on sequence based on operating hours.

The function block is available in two versions: for a sequence of four or [eight](#) [► 563] units.

Units with fewer operating hours take precedence in the sequence over units with more operating hours.

A rising edge at *bChg* forces a sequence change. The units with the fewest operating hours are set to the top of the FIFO and thus given priority for switching on.

In the sequence only units are entered that are enabled at inputs *bEn01*..*bEn04*. *udiNum* indicates the number of requested units.

The operating hours of the units are entered at inputs *udiActvTi01_h* to *udiActvTi04_h*. If all these inputs are set to a constant value of zero, the sequence change is controlled cyclically, depending on *bChg*.

The first unit is removed from the FIFO, the other units are advanced, and the first unit is appended at the end of the FIFO again. As a result is an alternating sequence of units.

If more units are requested at input *udiNum* than are available at inputs *bEn01* to *bEn04*, this is indicated with TRUE at *bErr*.

Error handling

If more units are requested at input *udiNum* than are available at inputs *bEn01* to *bEn04*, this is indicated with TRUE at *bErr*.

VAR_INPUT

```

bEn          : BOOL;
udiNum       : UDINT;
bChg         : BOOL;
bEn01        : BOOL;
bEn02        : BOOL;
bEn03        : BOOL;
bEn04        : BOOL;
udiActvTi01_h : UDINT;
udiActvTi02_h : UDINT;
udiActvTi03_h : UDINT;
udiActvTi04_h : UDINT;

```

bEn: Enables the function block.

udiNum: Number of units.

bChg: Force sequence change.

bEn01...bEn04: Enable unit 1...enable unit 4.

udiActvTi01_h...udiActvTi04_h: Operating hours unit 1...operating hours unit 4.

VAR_OUTPUT

```
bQ01      : BOOL;
bQ02      : BOOL;
bQ03      : BOOL;
bQ04      : BOOL;
udiNextOn  : UDINT;
udiNextOff : UDINT;
arrFIFO    : ARRAY [1..4] OF UDINT;
udiNumOfEn : UDINT;
bErr       : BOOL;
sErrDesc   : STRING;
```

bQ01...bQ04: Switches unit 1..4.

udiNextOn: Number of the unit that is switched on next.

udiNextOff: Number of the unit that is switched on next.

arrFIFO: FIFO buffer as a field.

udiNumOfEn: Number of devices, depending on the individual enable states.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDesc: Contains the error description.

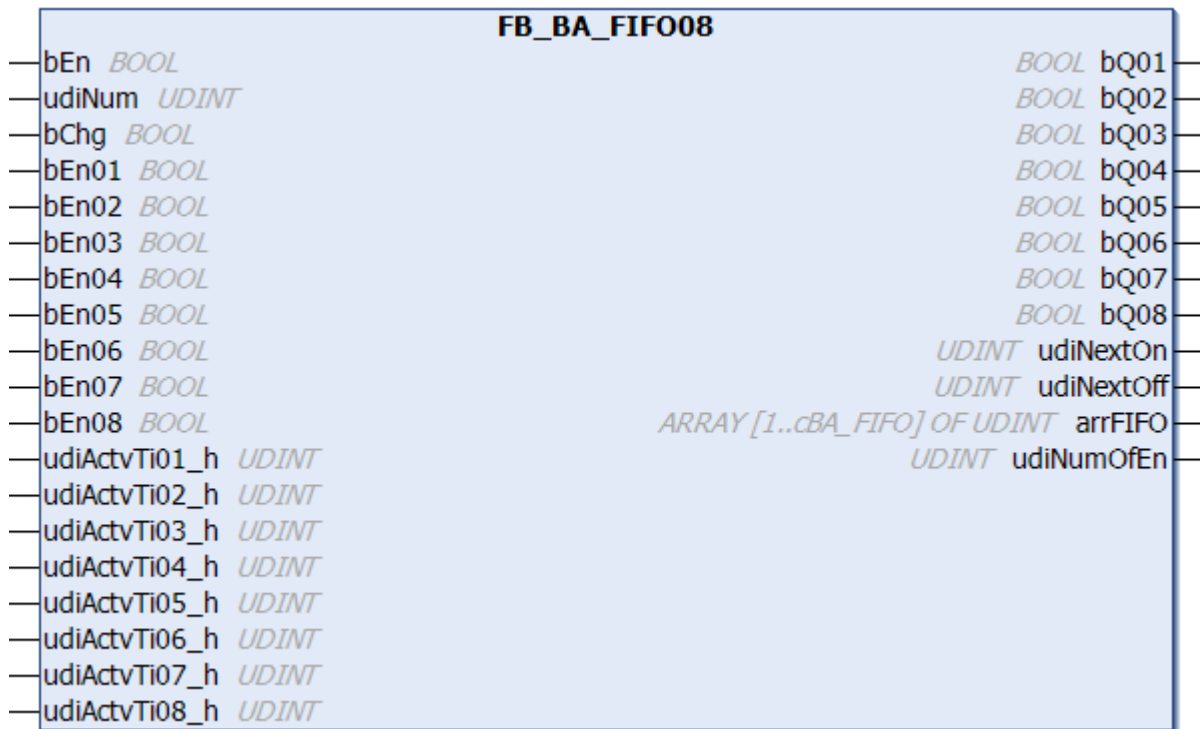
Error description

01: Warning: More than 4 devices are entered at input udiNum. The number is limited to the number enabled at inputs bEn01..bEn04.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.7 FB_BA_FIFO08



The function block *FB_BA_FIFO08* enables sequential control of up to eight units, with automatic switching of the switch-on sequence based on operating hours.

The function block is available in two versions: for a sequence of four [► 561] or eight units.

Units with fewer operating hours take precedence in the sequence over units with more operating hours.

A rising edge at *bChg* forces a sequence change. The units with the fewest operating hours are set to the top of the FIFO and thus given priority for switching on.

In the sequence only units are entered that are enabled at inputs *bEn01*..*bEn08*. *udiNum* indicates the number of requested units.

The operating hours of the units are entered at inputs *udiActvTi01_h* to *udiActvTi08_h*. If all these inputs are set to a constant value of zero, the sequence change is controlled cyclically, depending on *bChg*.

The first unit is removed from the FIFO, the other units are advanced, and the first unit is appended at the end of the FIFO again. As a result is an alternating sequence of units.

If more units are requested at input *udiNum* than are available at inputs *bEn01* to *bEn08*, this is indicated with TRUE at *bErr*.

Error handling

If more units are requested at input *udiNum* than are available at inputs *bEn01* to *bEn08*, this is indicated with TRUE at *bErr*.

VAR_INPUT

```

bEn          : BOOL;
udiNum       : UDINT;
bChg        : BOOL;
bEn01       : BOOL;
bEn02       : BOOL;
bEn03       : BOOL;
bEn04       : BOOL;
bEn05       : BOOL;
bEn06       : BOOL;
bEn07       : BOOL;
bEn08       : BOOL;
udiActvTi01_h : UDINT;
udiActvTi02_h : UDINT;
udiActvTi03_h : UDINT;
udiActvTi04_h : UDINT;
udiActvTi05_h : UDINT;

```



```

udiActvTi06_h : UDINT;
udiActvTi07_h : UDINT;
udiActvTi08_h : UDINT;

```

bEn: Enables the function block.

udiNum: Number of units.

bChg: Force sequence change.

bEn01...bEn08: Enable unit 1...enable unit 8.

udiActvTi01_h...udiActvTi08_h: Operating hours unit 1...operating hours unit 8.

VAR_OUTPUT

```

bQ01      : BOOL;
bQ02      : BOOL;
bQ03      : BOOL;
bQ04      : BOOL;
bQ05      : BOOL;
bQ06      : BOOL;
bQ07      : BOOL;
bQ08      : BOOL;
udiNextOn  : UDINT;
udiNextOff : UDINT;
arrFIFO    : ARRAY [1..8] OF UDINT;
udiNumOfEn : UDINT;
bErr       : BOOL;
sErrDescr  : STRING;

```

bQ01...bQ08: Switches unit 1..8.

udiNextOn: Number of the unit that is switched on next.

udiNextOff: Number of the unit that is switched on next.

arrFIFO: FIFO buffer as a field.

udiNumOfEn: Number of devices, depending on the individual enable states.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

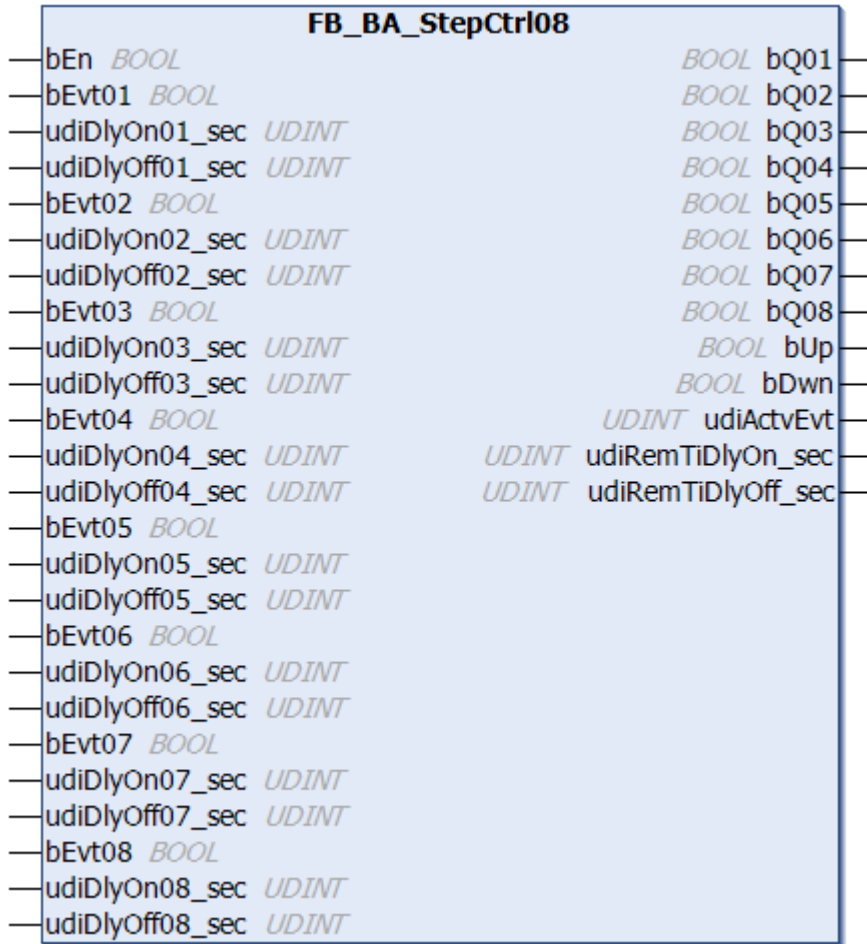
Error description

01: Warning: More than 8 devices are entered at input udiNum. The number is limited to the number enabled at inputs bEn01..bEn08.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.8 FB_BA_StepCtrl08

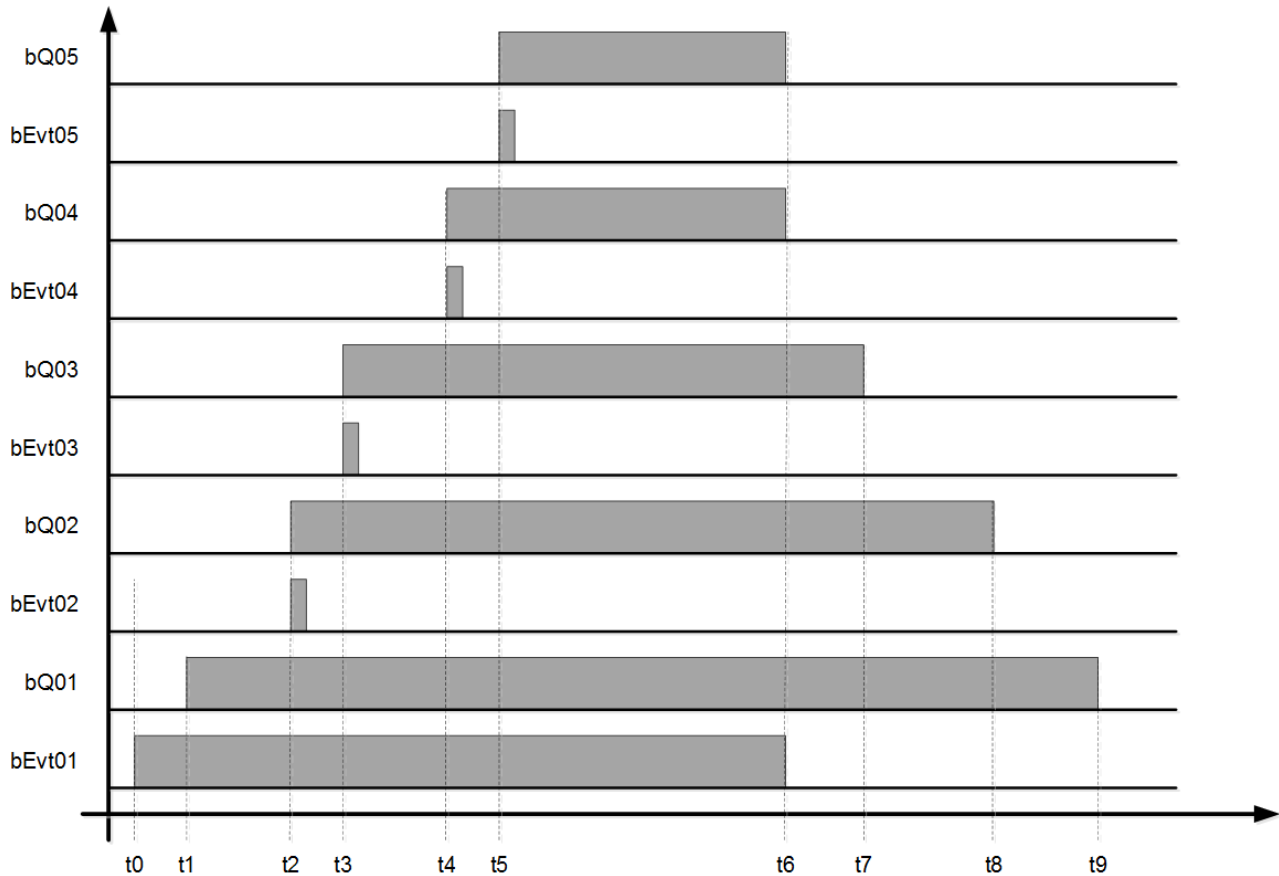


The function block is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs of *bQ01* to *bQ08* are set to FALSE. The control sequence starts at input *bEvt01*. Once the timer *udiDlyOn01_sec* (see Parameters) has elapsed, the corresponding output *bQ01* is set. Further stages are activated after a rising edge at the inputs *bEvt02* to *bEvt08*, in each case delayed via the timers *udiDlyOn02_sec* to *udiDlyOn08_sec*. If *bEvt01* becomes FALSE once the control chain is up and running, the control sequence switches back in reverse order. Switching off of the outputs is delayed by the timers *udiDlyOff01_sec* to *udiDlyOff08_sec*; see Parameters.

The outputs *bUp* and *bDwn* indicate whether the control chain is in ascending or descending state. The variable *udiActvEvt* indicates the current step of the control chain. "0" means the step sequence is not active.

The output *udiRemTiDlyOn_sec* indicates the time remaining to the next step during up-switching of the control chain. The output *udiRemTiDlyOff_sec* indicates the time remaining to the next lower step during down-switching of the control chain.

Example



- t0 step sequence switch-on
- t1 switch on step 1 $udiDlyOn01\_sec = t1 - t0$
- t2 event enable step 2, switch on step 2, $udiDlyOn02\_sec = 0$
- t3 event enable step 3, switch on step 3, $udiDlyOn03\_sec = 0$
- t4 event enable step 4, switch on step 4, $udiDlyOn04\_sec = 0$
- t5 event enable step 5, switch on step 5, $udiDlyOn05\_sec = 0$
- t6 disable the step sequence, disable step 5, disable step 4; $udiDlyOff05\_sec = 0$, $udiDlyOff04\_sec = 0$
- t7 disable step 3, $udiDlyOff03\_sec = t7 - t6$
- t8 disable step 2, $udiDlyOff02\_sec = t8 - t7$
- t9 disable step 1, $udiDlyOff01\_sec = t9 - t8$

VAR_INPUT

```

bEn          : BOOL;
bEvt01       : BOOL;
udiDlyOn01_sec : UDINT;
udiDlyOff01_sec : UDINT;
bEvt02       : BOOL;
udiDlyOn02_sec : UDINT;
udiDlyOff02_sec : UDINT;
bEvt03       : BOOL;
udiDlyOn03_sec : UDINT;
udiDlyOff03_sec : UDINT;
bEvt04       : BOOL;
udiDlyOn04_sec : UDINT;
udiDlyOff04_sec : UDINT;
bEvt05       : BOOL;
udiDlyOn05_sec : UDINT;
udiDlyOff05_sec : UDINT;
bEvt06       : BOOL;
udiDlyOn06_sec : UDINT;
udiDlyOff06_sec : UDINT;
bEvt07       : BOOL;
udiDlyOn07_sec : UDINT;

```

```

udiDlyOff07_sec : UDINT;
bEvt08          : BOOL;
udiDlyOn08_sec  : UDINT;
udiDlyOff08_sec : UDINT;

```

bEn: Enable function block.

bEvt01..08: Switch-on command for steps 1 to 8.

udiDlyOn01..08_sec: Switch-on delay for output *bQ01 .. 08* [s].

udiDlyOff01..08_sec: Switch-off delay for output *bQ01 .. 08* [s].

VAR_OUTPUT

```

bQ01          : BOOL;
bQ02          : BOOL;
bQ03          : BOOL;
bQ04          : BOOL;
bQ05          : BOOL;
bQ06          : BOOL;
bQ07          : BOOL;
bQ08          : BOOL;
bUp           : BOOL;
bDwn          : BOOL;
udiActvEvt    : UDINT;
udiRemTiDlyOn_sec : UDINT;
udiRemTiDlyOff_sec : UDINT;

```

bQ01..08: Step 1 to 8 On.

bUp: Control chain is in ascending state.

bDwn: Control chain is in descending state.

udiActvEvt: Active step, display 0..8; "0" represents an active step sequence.

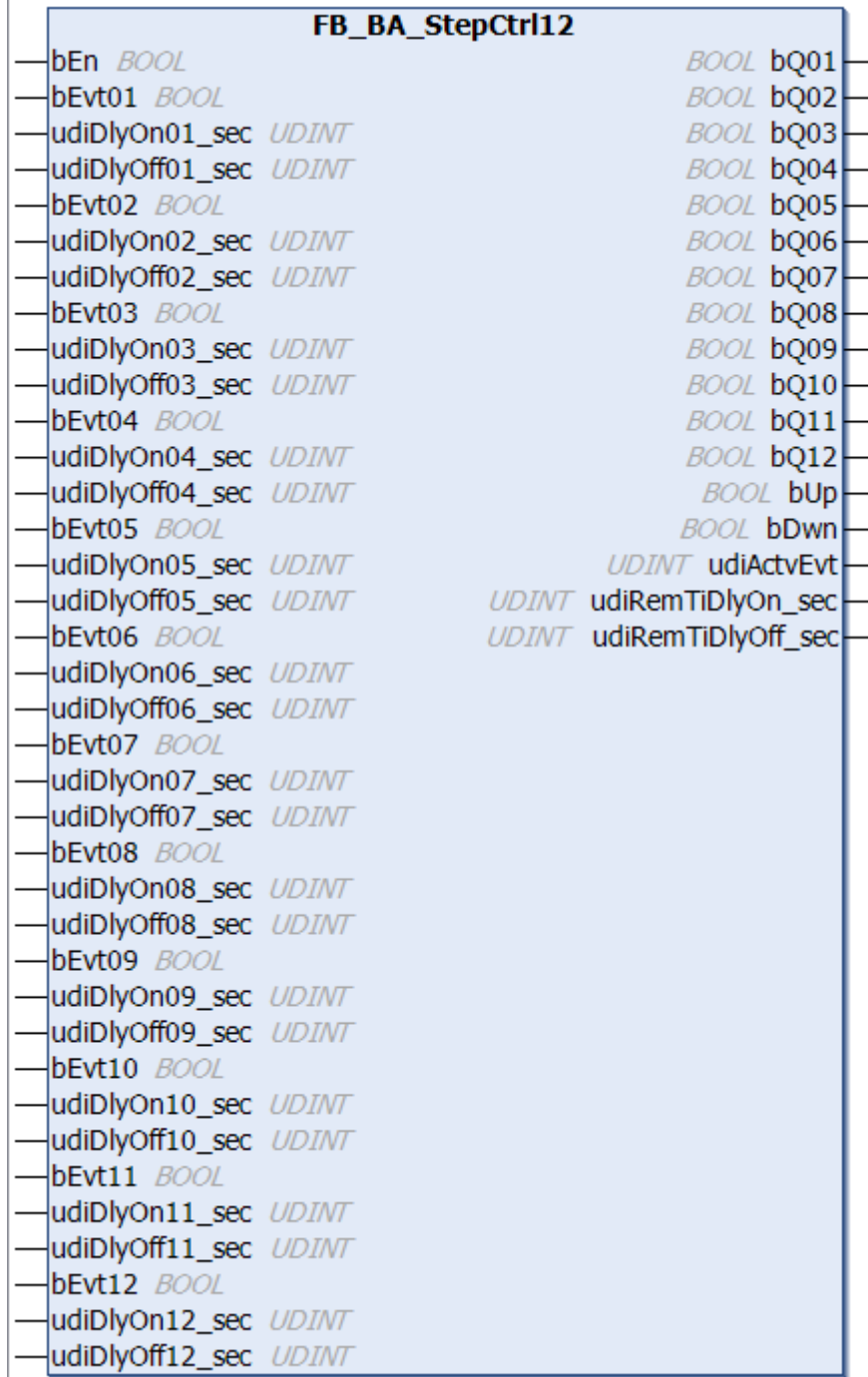
udiRTiDlyOn: Time remaining to up-switching to the next step [s].

udiRTiDlyOff: Time remaining to down-switching to the previous step [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.9 FB_BA_StepCtrl12

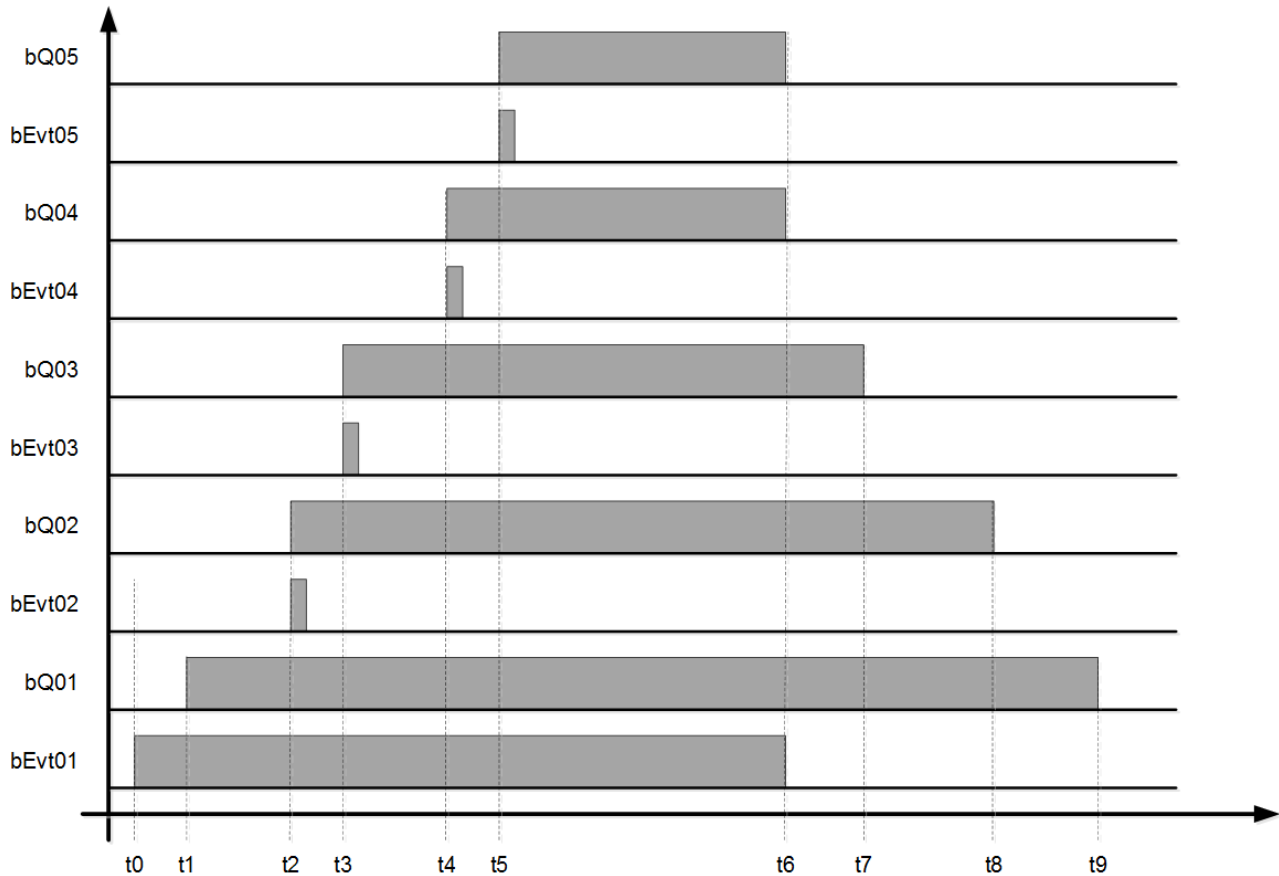


The function block is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs of *bQ01* to *bQ12* are set to FALSE. The control sequence starts at input *bEvt01*. Once the timer *udiDlyOn01_sec* (see Parameters) has elapsed, the corresponding output *bQ01* is set. Further stages are activated after a rising edge at the inputs *bEvt02* to *bEvt12*, in each case delayed via the timers *udiDlyOn02_sec* to *udiDlyOn12_sec*. If *bEvt01* becomes FALSE once the control chain is up and running, the control sequence switches back in reverse order. Switching off of the outputs is delayed by the timers *udiDlyOff01_sec* to *udiDlyOff12_sec*; see Parameters.

The outputs *bUp* and *bDwn* indicate whether the control chain is in ascending or descending state. The variable *udiActvEvt* indicates the current step of the control chain. "0" means the step sequence is not active.

The output *udiRemTiDlyOn_sec* indicates the time remaining to the next step during up-switching of the control chain. The output *udiRemTiDlyOff_sec* indicates the time remaining to the next lower step during down-switching of the control chain.

Example



- t0 step sequence switch-on
- t1 switch on step 1 $udiDlyOn01\_sec = t1 - t0$
- t2 event enable step 2, switch on step 2, $udiDlyOn02\_sec = 0$
- t3 event enable step 3, switch on step 3, $udiDlyOn03\_sec = 0$
- t4 event enable step 4, switch on step 4, $udiDlyOn04\_sec = 0$
- t5 event enable step 5, switch on step 5, $udiDlyOn05\_sec = 0$
- t6 disable the step sequence, disable step 5, disable step 4; $udiDlyOff05\_sec = 0$, $udiDlyOff04\_sec = 0$
- t7 disable step 3, $udiDlyOff03\_sec = t7 - t6$
- t8 disable step 2, $udiDlyOff02\_sec = t8 - t7$
- t9 disable step 1, $udiDlyOff01\_sec = t9 - t8$

VAR_INPUT

```

bEn           : BOOL;
bEvt01        : BOOL;
udiDlyOn01_sec : UDINT;
udiDlyOff01_sec : UDINT;
bEvt02        : BOOL;
udiDlyOn02_sec : UDINT;
udiDlyOff02_sec : UDINT;
bEvt03        : BOOL;
udiDlyOn03_sec : UDINT;
udiDlyOff03_sec : UDINT;
bEvt04        : BOOL;
udiDlyOn04_sec : UDINT;
udiDlyOff04_sec : UDINT;
bEvt05        : BOOL;
udiDlyOn05_sec : UDINT;
udiDlyOff05_sec : UDINT;
bEvt06        : BOOL;
udiDlyOn06_sec : UDINT;
udiDlyOff06_sec : UDINT;
bEvt07        : BOOL;
udiDlyOn07_sec : UDINT;

```

```

udiDlyOff07_sec : UDINT;
bEvt08          : BOOL;
udiDlyOn08_sec  : UDINT;
udiDlyOff08_sec : UDINT;
bEvt09          : BOOL;
udiDlyOn09_sec  : UDINT;
udiDlyOff09_sec : UDINT;
bEvt10          : BOOL;
udiDlyOn10_sec  : UDINT;
udiDlyOff10_sec : UDINT;
bEvt11          : BOOL;
udiDlyOn11_sec  : UDINT;
udiDlyOff11_sec : UDINT;
bEvt12          : BOOL;
udiDlyOn12_sec  : UDINT;
udiDlyOff12_sec : UDINT;

```

bEn: Enable function block.

bEvt01..12: Switch-on command for steps 1 to 12.

udiDlyOn01..12_sec: Switch-on delay for output *bQ01* .. 12 [s].

udiDlyOff01..12_sec: Switch-off delay for output *bQ01* .. 12 [s].

VAR_OUTPUT

```

bQ01          : BOOL;
bQ02          : BOOL;
bQ03          : BOOL;
bQ04          : BOOL;
bQ05          : BOOL;
bQ06          : BOOL;
bQ07          : BOOL;
bQ08          : BOOL;
bQ09          : BOOL;
bQ10          : BOOL;
bQ11          : BOOL;
bQ12          : BOOL;
bUp           : BOOL;
bDwn          : BOOL;
udiActvEvt    : UDINT;
udiRemTiDlyOn_sec : UDINT;
udiRemTiDlyOff_sec : UDINT;

```

bQ01..12: Step 1 to 12 On.

bUp: Control chain is in ascending state.

bDwn: Control chain is in descending state.

udiActvEvt: Active step, display 0..12; "0" represents an active step sequence.

udiRTiDlyOn: Time remaining to up-switching to the next step [s].

udiRTiDlyOff: Time remaining to down-switching to the previous step [s].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.10 FB_BA_FIFO04_XX



This function block is used to evaluate the FiFo memory from [FB BA FIFO04](#) [► 561]. The inputs are linked according to the FIFO table to the corresponding outputs of the function block *FB_BA_FIFO04_BOOL* or *FB_BA_FIFO04_REAL*.

Sample:

In the sample the array contains: 4,3,1,2,0,0,0,0. The following result is output in *FB_BA_FIFO04_REAL*:

rIn01 at output *rVal04*

rIn02 at output *rVal03*

rIn03 at output *rVal01*

rIn04 at output *rVal02*

VAR_INPUT

```
arrFIFO      : Array [1..4] OF UDINT;  
rIn01 - rIn04 : REAL;
```

aFIFO: Contains the assignment table with a maximum of eight values. The first value indicates where the first input was copied to, the second value where the second input was copied to and so on. No assignment takes place with "0".

rIn01 – rIn04: Setpoints to be linked.

VAR_OUTPUT

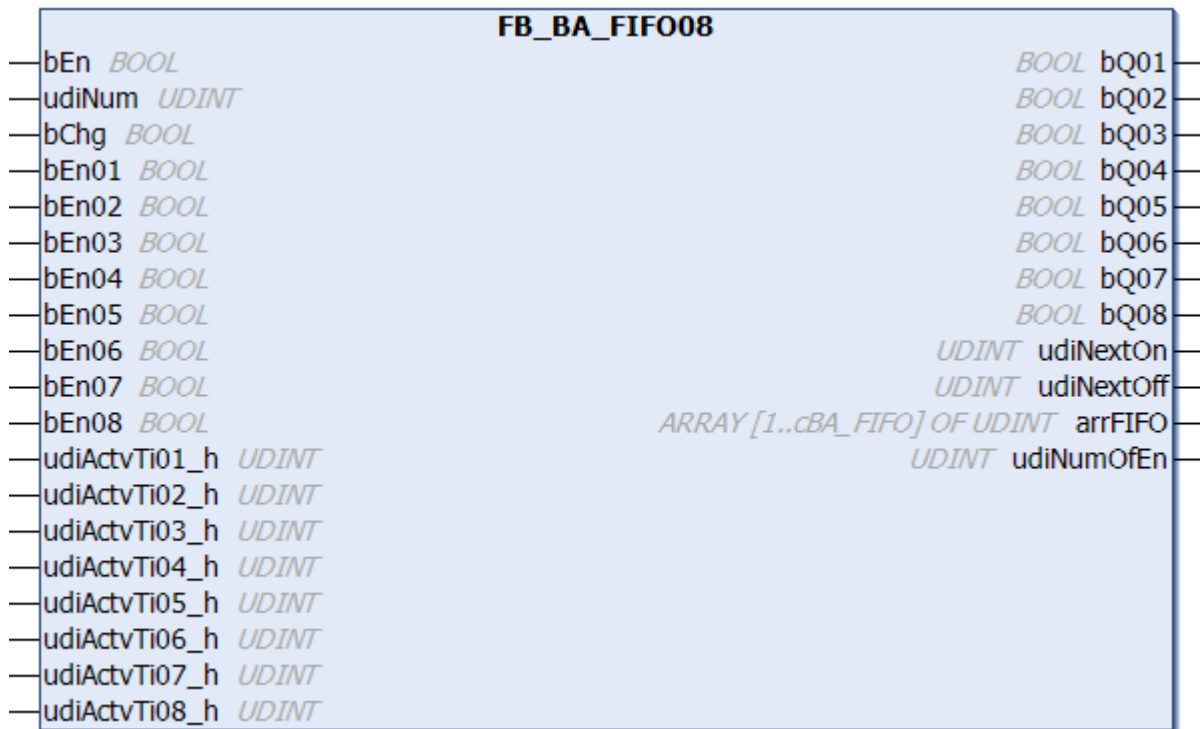
```
rVal01 - rVal04 : REAL;
```

rVal01 – rVal04: Actuator setpoint, input value linked according to FIFO table.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3.11 FB_BA_FIFO08_XX



This function block is used to evaluate the FiFo memory from FB_BA_FIFO08 [► 563]. The inputs are linked according to the FIFO table to the corresponding outputs of the function block *FB_BA_FIFO08_BOOL* or *FB_BA_FIFO08_REAL*.

Sample:

In the sample the array contains: 4,3,1,2,0,0,0,0. The result output in *FB_BA_FIFO08_REAL* is

rIn01 at output *rVal04*

rIn02 at output *rVal03*

rIn03 at output *rVal01*

rIn04 at output *rVal02*

.

VAR_INPUT

```
arrFIFO      : Array [1..8] OF UDINT;
rIn01 - rIn08 : REAL;
```

aFIFO: Contains the assignment table with a maximum of eight values. The first value indicates where the first input was copied to, the second value where the second input was copied to and so on. No assignment takes place with "0".

rIn01 – rIn08: Setpoints to be linked.

VAR_OUTPUT

```
rVal01 - rVal08 : REAL;
```

rVal01 – rVal08: Actuator setpoint, input value linked according to FIFO table.

Requirements

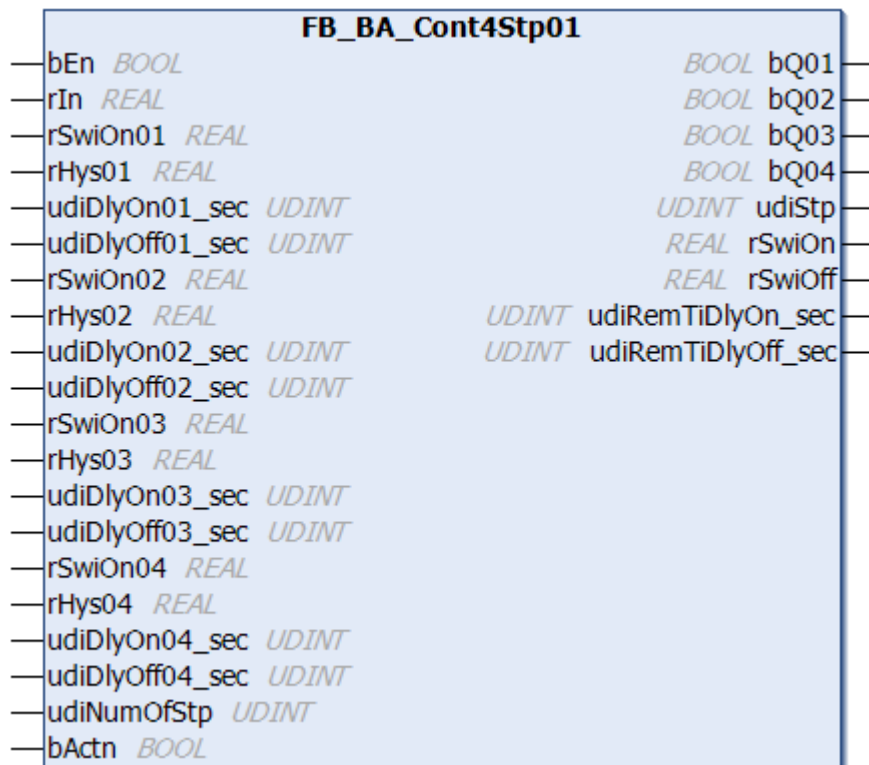
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.4 Hysteresis, 2-point control

Function blocks

Name	Description
FB_BA_Cont4Stp01 [► 573]	Step switch with four stages
FB_BA_Swi2P [► 578]	Two-point switch
FB_BA_SwiHys2P [► 580]	Two-point switch with one switching point

6.1.2.3.2.1.5.4.1 FB_BA_Cont4Stp01

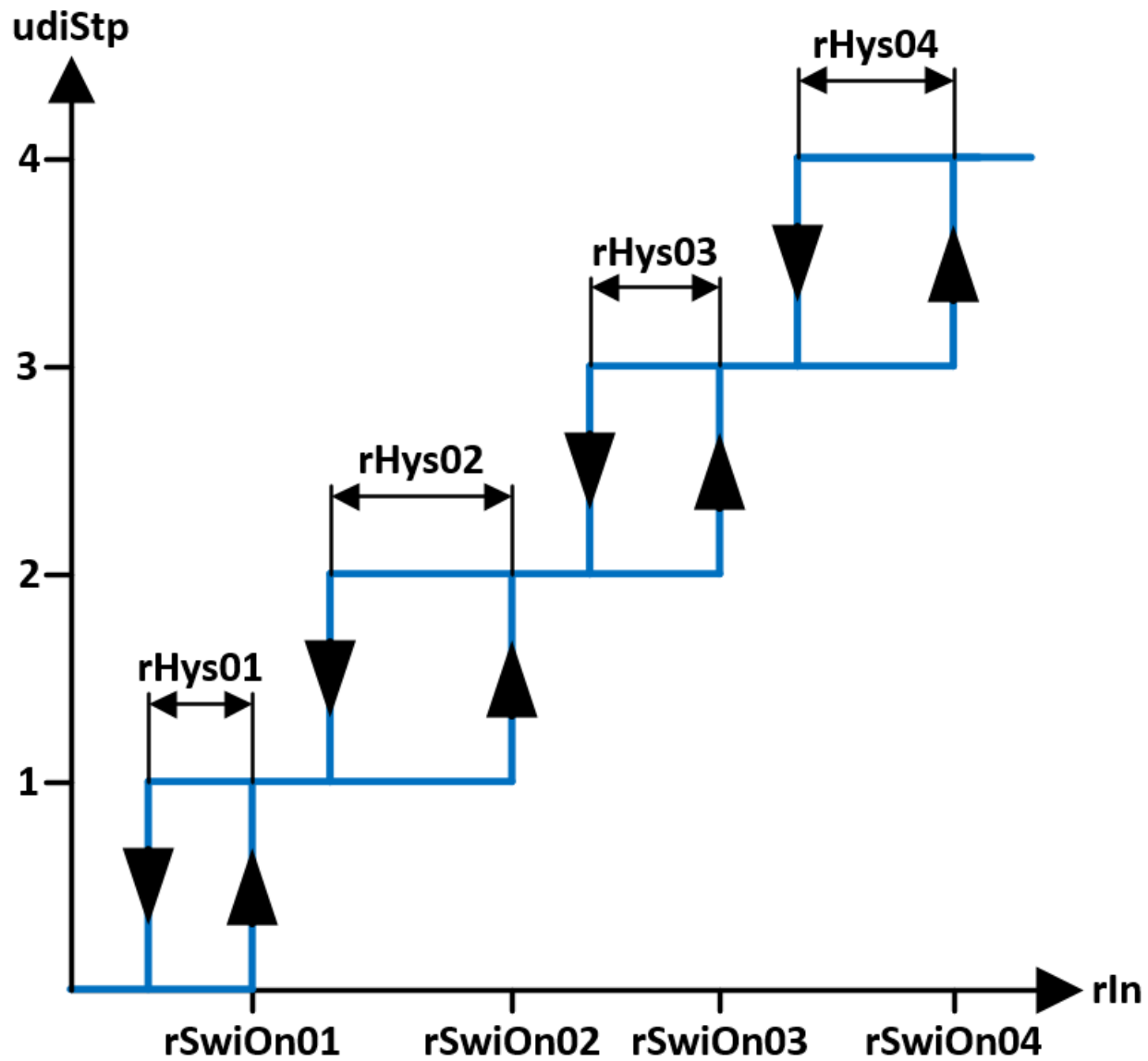


The function block determines the resulting switching stages of a multi-level unit, depending on the input signal.

Four switch-on thresholds and four hysteresis values can be parameterized.

Diagram 01

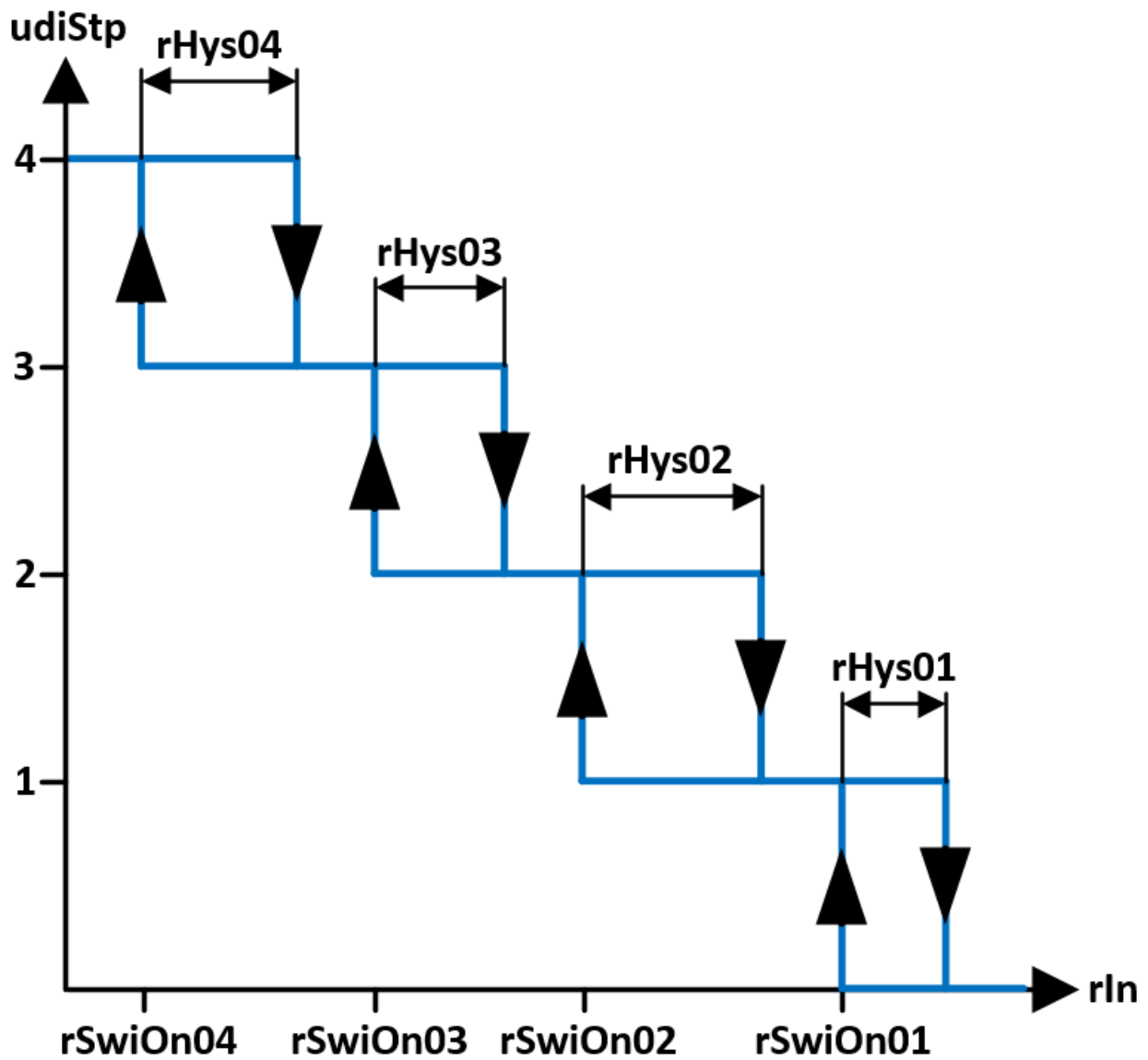
Control direction of parameter `bActn` = FALSE = Reverse = Heating



udiStp	udiNumOfStp	rSwiOn	rSwiOff	udiRemTiDlyOn_sec	udiRemTiDlyOff_sec	bQ01	bQ02	bQ03	bQ04
0	0	rSwiOn01	rSwiOn01 - rHys01	udiDlyOn01_sec	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	rSwiOn02	rSwiOn01 - rHys01	udiDlyOn02_sec	udiDlyOff01_sec	TRUE	FALSE	FALSE	FALSE
2	>= 2	rSwiOn03	rSwiOn02 - rHys02	udiDlyOn03_sec	udiDlyOff02_sec	TRUE	TRUE	FALSE	FALSE
3	>= 3	rSwiOn04	rSwiOn03 - rHys03	udiDlyOn04_sec	udiDlyOff03_sec	TRUE	TRUE	TRUE	FALSE
4	>= 4	rSwiOn04	rSwiOn04 - rHys04	0	udiDlyOff04_sec	TRUE	TRUE	TRUE	TRUE

Diagram 02

Control direction parameter *bActn* = TRUE = Direct = Cooling



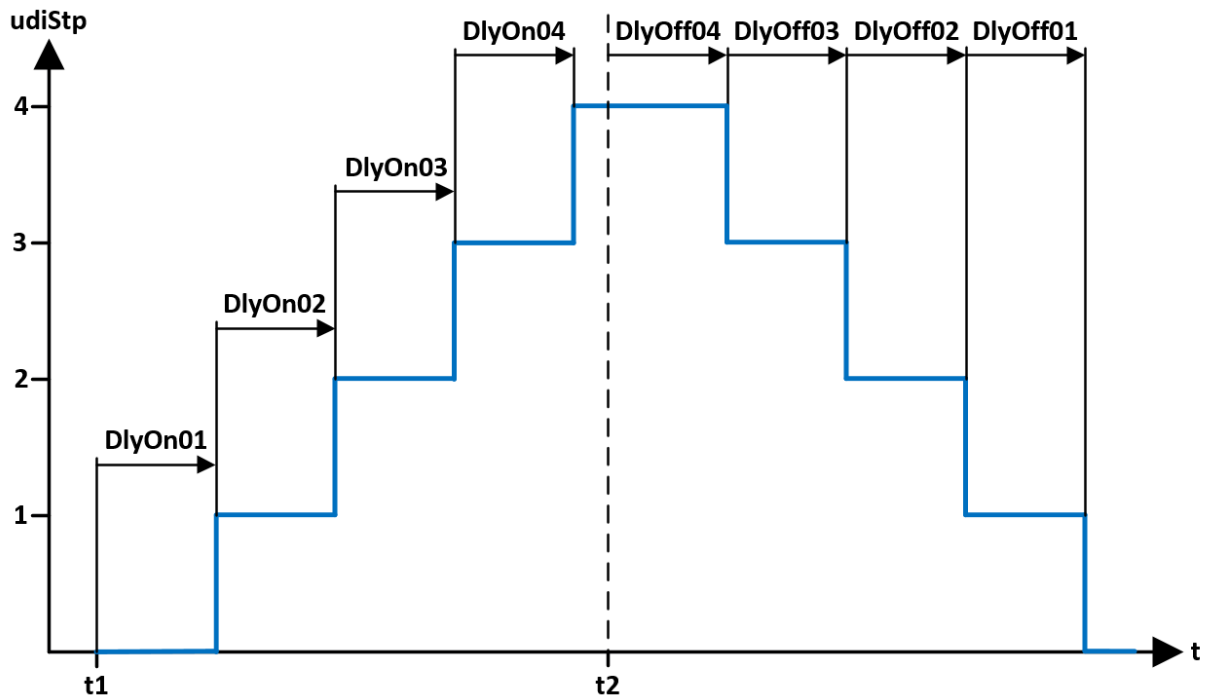
udiStp	udiNumOfStp	rSwiOn	rSwiOff	udiRemTiDlyOn_sec	udiRemTiDlyOff_sec	bQ01	bQ02	bQ03	bQ04
0	0	rSwiOn01	rSwiOn01 + rHys01	udiDlyOn01_sec	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	rSwiOn02	rSwiOn01 + rHys01	udiDlyOn02_sec	udiDlyOff01_sec	TRUE	FALSE	FALSE	FALSE
2	>= 2	rSwiOn03	rSwiOn02 + rHys02	udiDlyOn03_sec	udiDlyOff02_sec	TRUE	TRUE	FALSE	FALSE
3	>= 3	rSwiOn04	rSwiOn03 + rHys03	udiDlyOn04_sec	udiDlyOff03_sec	TRUE	TRUE	TRUE	FALSE
4	4	rSwiOn04	rSwiOn04 + rHys04	0	udiDlyOff04_sec	TRUE	TRUE	TRUE	TRUE

Diagram 03

Timing of the switch-on and switch-off delays

At time t_1 , **rIn** jumps from **rSwiOn01** to **rSwiOn04**

At time t_2 , rIn jumps from $rSwiOn04$ to $rSwiOn01$ - $rHys01$



VAR_INPUT

```

bEn          : BOOL;
rIn          : REAL;
rSwiOn01     : REAL;
rHys01       : REAL;
udiDlyOn01_sec : UDINT;
udiDlyOff01_sec : UDINT;
rSwiOn02     : REAL;
rHys02       : REAL;
udiDlyOn02_sec : UDINT;
udiDlyOff02_sec : UDINT;
rSwiOn03     : REAL;
rHys03       : REAL;
udiDlyOn03_sec : UDINT;
udiDlyOff03_sec : UDINT;
rSwiOn04     : REAL;
rHys04       : REAL;
udiDlyOn04_sec : UDINT;
udiDlyOff04_sec : UDINT;
udiNumOfStp  : UDINT;
bActn        : BOOL;

```

bEn: General enable of the function block. If bEn is FALSE, all outputs are set to 0.

rIn: Input value, from which the switching state is derived.

rSwiOn01: Switch-on point stage 01

rHys01: Absolute value hysteresis stage 01

udiDlyOn01_sec: Switch-on delay stage 01

udiDlyOff01_sec: Switch-off delay stage 01

rSwiOn02: Switch-on point stage 02

rHys02: Absolute value hysteresis stage 02

udiDlyOn02_sec: Switch-on delay stage 02

udiDlyOff02_sec: Switch-off delay stage 02

rSwiOn03: Switch-on point stage 03

rHys03: Absolute value hysteresis stage 03

udiDlyOn03_sec: Switch-on delay stage 03

udiDlyOff03_sec: Switch-off delay stage 03

rSwiOn04: Switch-on point stage 04

rHys04: Absolute value hysteresis stage 04

udiDlyOn04_sec: Switch-on delay stage 04

udiDlyOff04_sec: Switch-off delay stage 04

udiNumOfStp: Number of stages that are required.
The input is limited to a range from 0 to 4

bActn: Input variable used to determine the control direction of the step switch.
TRUE = direct = cooling; FALSE = reverse = heating

VAR_OUTPUT

```
bQ01      : BOOL;
bQ02      : BOOL;
bQ03      : BOOL;
bQ04      : BOOL;
udiStp     : UDINT;
rSwiOn     : REAL;
rSwiOff    : REAL;
udiRemTiDlyOn_sec : UDINT;
udiRemTiDlyOff_sec : UDINT;
```

bQ01: Display of status step 01
TRUE = ON; FALSE = OFF
udiStp >= 1

bQ02: Display of status step 02
TRUE = ON; FALSE = OFF
udiStp >= 2

bQ03: Display of status step 03
TRUE = ON; FALSE = OFF
udiStp >= 3

bQ04: Display of status step 04
TRUE = ON; FALSE = OFF
udiStp >= 4

udiStp: Shows the current step of the step switch

rSwiOn: Shows the next switch-on point

rSwiOff: Shows the next switch-off point

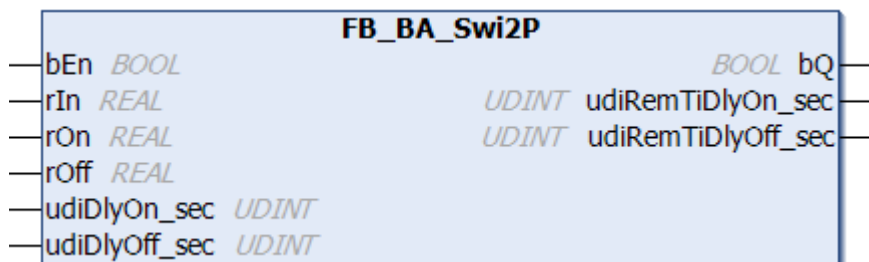
udiRemTiDlyOn_sec: If the switch-on point for switching to the next level is met, the progress of the switch-on delay time is displayed here.

udiRemTiDlyOff_sec: If the switch-off point for switching down to the next level is met, the progress of the switch-off delay time is displayed here.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

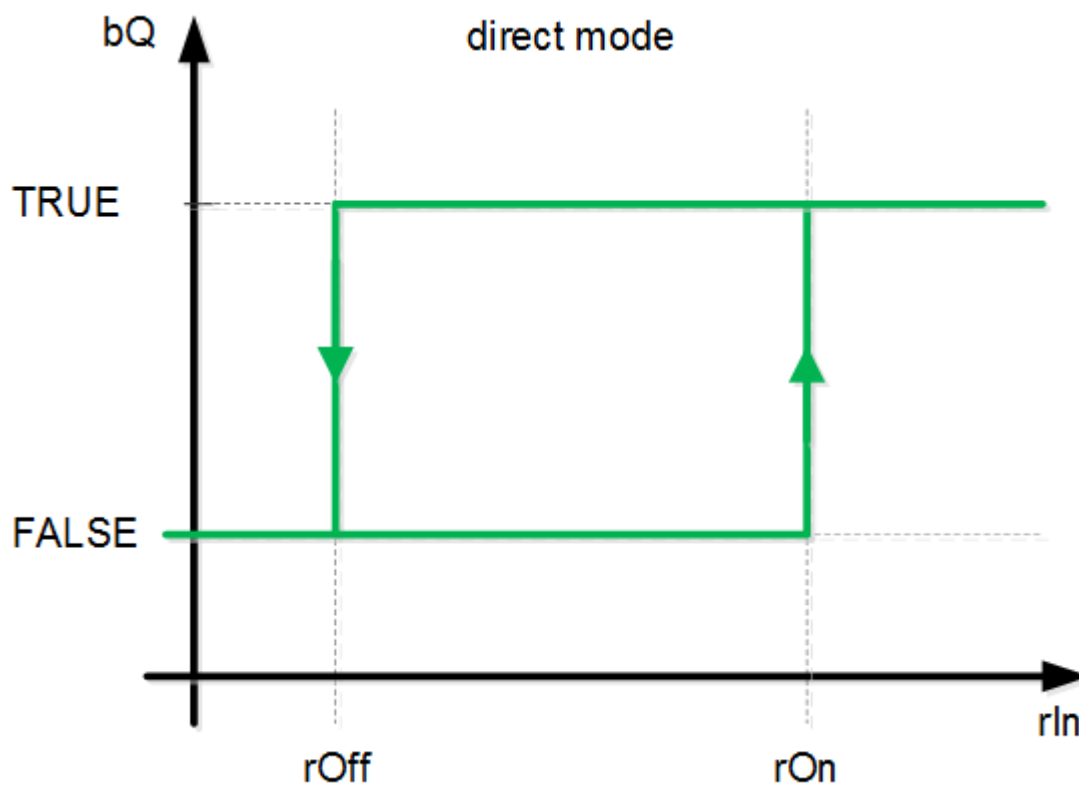
6.1.2.3.2.1.5.4.2 FB_BA_Swi2P



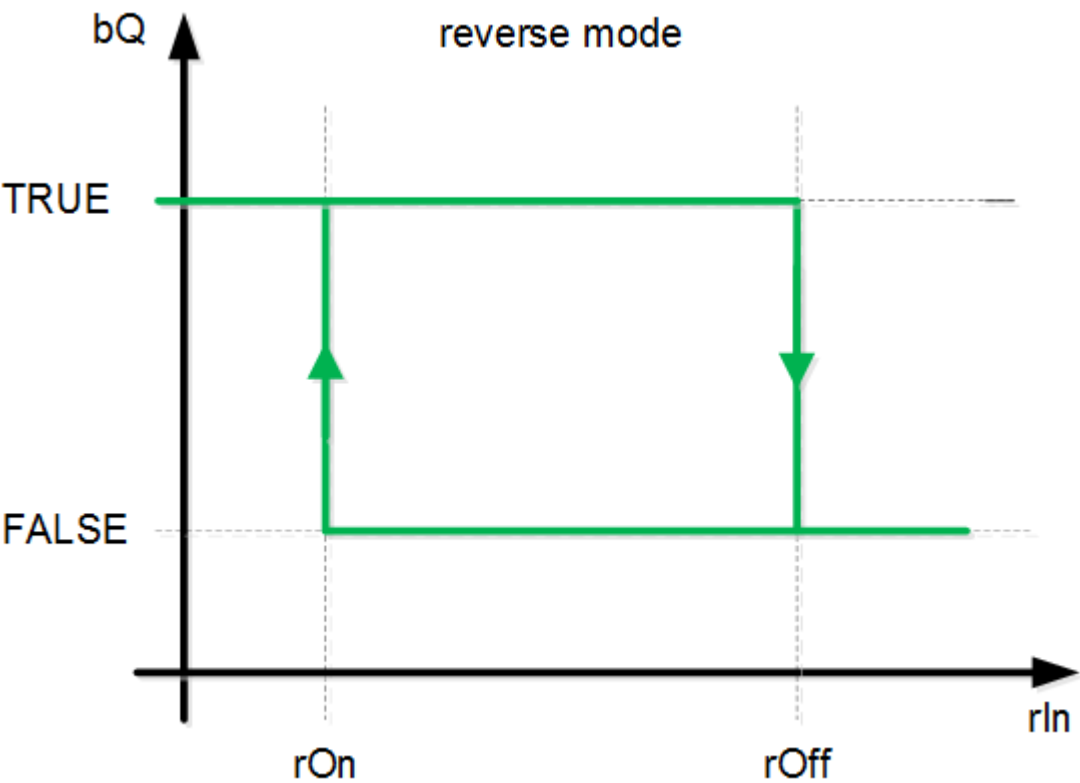
The function block *FB_BA_Swi2P* is a two-point switch with one switch-on point and one switch-off point.

A general function block enable can be implemented at input *bEn*. The output *bQ* is FALSE as long as *bEn* is FALSE. The control direction of the function block depends on the relative position of the switch-on/switch-off points.

If the switch-on point is greater than the switch-off point, the control direction is direct/synchronous (cooling mode).



If the switch-off point is greater than the switch-on point, the control direction is indirect/reversed (heating mode).



VAR_INPUT

```
bEn      : BOOL;  
rIn      : REAL;  
rOn      : REAL;  
rOff     : REAL;  
udiDlyOn_sec : UDINT;  
udiDlyOff_sec : UDINT;
```

bEn: General enable of the function block.

rIn: Input value.

rOn: Switch-on point.

rOff: Switch-off point.

udiDlyOn_sec: Switch-on delay.

udiDlyOff_sec: Switch-off delay.

VAR_OUTPUT

```
bQ      : BOOL;  
udiRemTiDlyOn_sec : UDINT;  
udiRemTiDlyOff_sec : UDINT;
```

bQ: Control output.

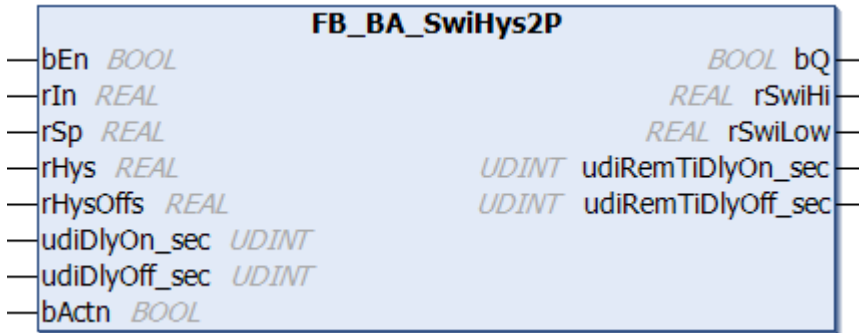
udiRemTiDlyOn_sec: Remaining time of the switch-on delay.

udiRemTiDlyOff_sec: Remaining time of the switch-off delay.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.4.3 FB_BA_SwiHys2P



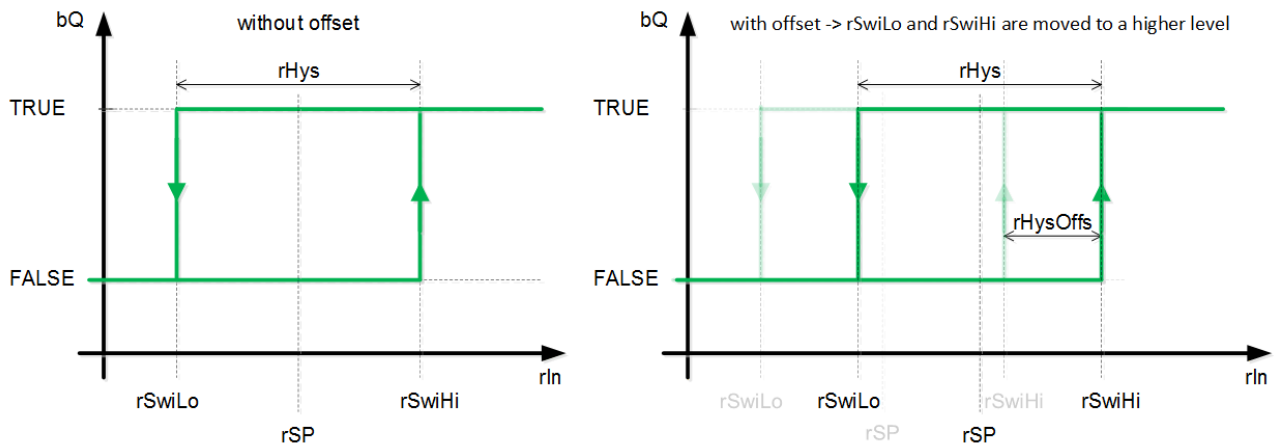
The function block *FB_BA_SwiHys2P* is a two-point switch with adjustable hysteresis and hysteresis offset.

A general function block enable can be implemented at input *bEn*. If the function block is locked, the output *bQ* is FALSE. The setpoint for the two-point switch is connected at input *rSp*. The control direction of the function block depends on the input variable *bActn*.

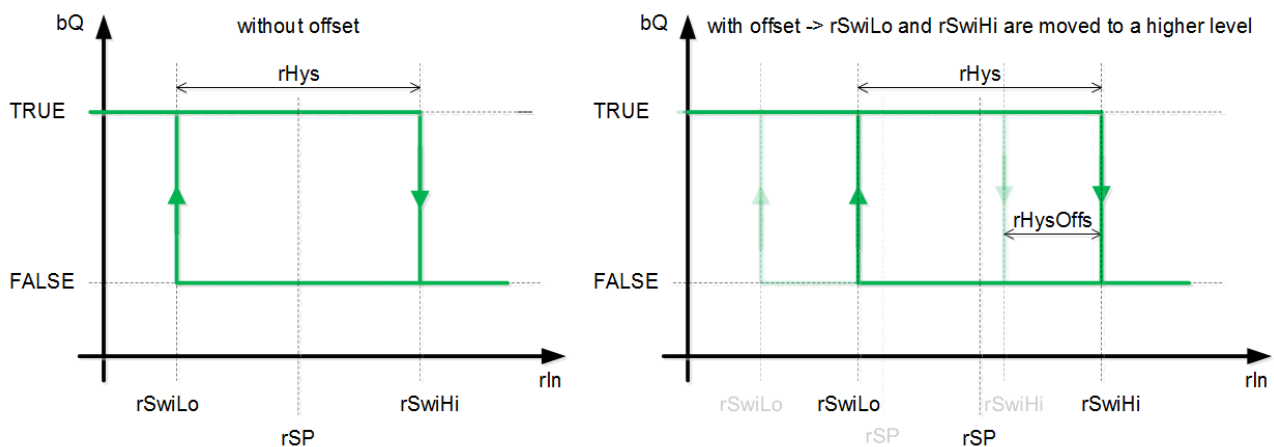
The active switching points result from the setpoint, the hysteresis and the hysteresis offset. They are output at *rSwiHi* and *rSwiLo*.

- The upper switching point results from $rSp + rHys/2 + rHysOffs$.
- The lower switching point results from $rSp + rHys/2 + rHysOffs$.

If *bActn* TRUE, the result is direct/synchronous control direction (cooling mode).



If *bActn* is FALSE, the result is indirect/reversed control direction (heating mode).



VAR_INPUT

```

bEn      : BOOL;
rIn      : REAL;
rSp      : REAL;
rHys     : REAL;
rHysOffs : REAL;
udiDlyOn_sec : UDINT;
udiDlyOff_sec : UDINT;
bActn    : BOOL;

```

bEn: General enable of the function block.

rIn: Input value.

rSp: Setpoint input.

rHys: Hysteresis.

rHysOffs: Hysteresis offset.

udiDlyOn_sec: Switch-on delay

udiDlyOff_sec: Release delay

bActn: Control direction.

VAR_OUTPUT

```

bQ      : BOOL;
rSwiHi  : REAL;
rSwiLo  : REAL;
udiRemTiDlyOn_sec : UDINT;
udiRemTiDlyOff_sec : UDINT;

```

bQ: Output.

rSwiHi: Upper switching point.

rSwiLo: Lower switching point.

udiRemTiDlyOn_sec: Time remaining before switching on.

udiRemTiDlyOff_sec: Time remaining before switching off.

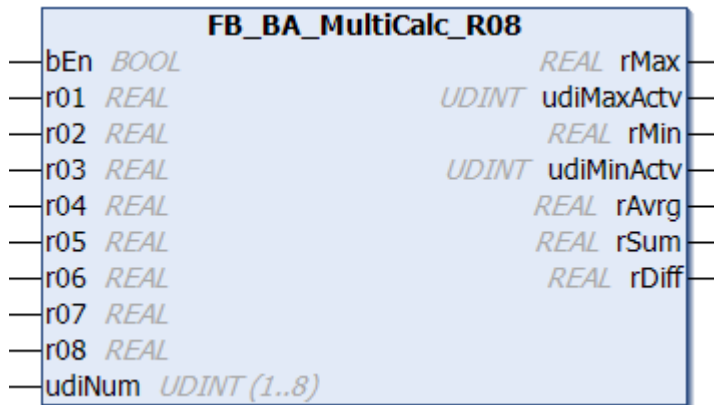
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5 Mathematical functions**Function blocks**

Name	Description
FB BA MultiCalc XX [► 582]	Multi-calculation function blocks
FB BA Chrct02 [► 583]	Linear interpolation for 2 interpolation points
FB BA Chrct04 [► 584]	Linear interpolation for 4 interpolation points
FB BA Chrct07 [► 586]	Linear interpolation for 7 interpolation points
FB BA Chrct32 [► 587]	Linear interpolation for 32 interpolation points
FB BA TiAvg [► 589]	Arithmetic mean value over time

6.1.2.3.2.1.5.5.1 FB_BA_MultiCalc_XX



The multi-calculation function blocks exist for the variable types LREAL and REAL, although they all have the same functionality.

The function block FB_BA_R08 is described as an example.

In enabled state (*bEn*=TRUE), the function block determines the following from the 8 input values *r01*...*r08*:

- the maximum value of all inputs *rMax*
- the input at which this maximum value occurs *udiMaxActv*
- the minimum value of all inputs *rMin*
- the input at which this minimum value occurs *udiMinActv*
- the mean value of all inputs *rAvg*
- the sum of all inputs *rSum*
- the difference between the maximum and minimum value *rDiff*

If not all inputs are used for the calculation, the number can be limited via an entry at *udiNum*: *udiNum*=6, for example, can be used to limit the calculations to inputs *r01* to *r06*.

Any entry greater than 8 is automatically limited to 8, any entry less than 1 is automatically set to 1.

Sample:

Inputs	Output
<i>bEn</i> = TRUE	<i>rMax</i> = 32
<i>r01</i> = 32	<i>udiMaxActv</i> = 1
<i>r02</i> = 17	<i>rMin</i> = 5
<i>r03</i> = 5	<i>udiMinActv</i> = 3
<i>r04</i> = 9	<i>rAvg</i> = 18.5
<i>r05</i> = 16	<i>rSum</i> = 111
<i>r06</i> = 32	<i>rDiff</i> = 27
<i>r07</i> = 25	
<i>r08</i> = 44	
<i>udiNum</i> = 6	

If *bEn*=FALSE, 0 is output at all outputs.

VAR_INPUT

```

bEn      : BOOL;
r01      : REAL;
r02      : REAL;
r03      : REAL;
r04      : REAL;
r05      : REAL;
r06      : REAL;
r07      : REAL;
r08      : REAL;
udiNum   : UDINT;

```

bEn: Activation of the block function.

r01...r08: Input values to be used for the calculation.

udiNum: Number of input values to be used for the calculation.

VAR_OUTPUT

```

rMax      : REAL;
udiMaxActv : UDINT;
rMin      : REAL;
udiMinActv : UDINT;
rAvrg     : REAL;
rSum      : REAL;
rDiff     : REAL;

```

rMax: Maximum value of all inputs.

udiMaxActv: Input at which the maximum value is present.

rMin: Minimum value of all inputs.

udiMinActv: Input at which the minimum value is present.

rAvrg: Mean value of all inputs.

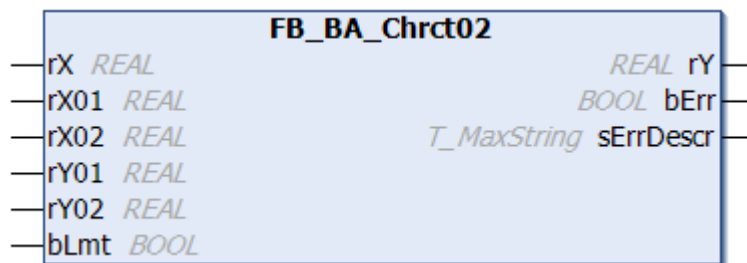
rSum: Sum of all inputs.

rDiff: Difference between maximum and minimum value.

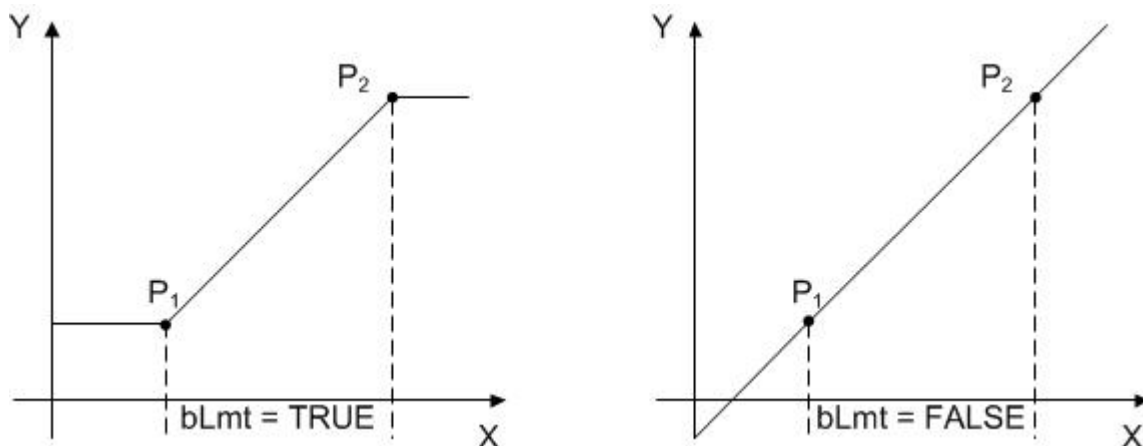
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5.2 FB_BA_Chrc02



The function block FB_BA_Chrc02 represents a linear interpolation with 2 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points $[rX1/rY1]$ and $[rX2/rY2]$. If the input variable *bLmt* is TRUE, *rY* is limited by *rY01* and *rY02*. If *bLmt* is FALSE, *rY* is not limited.



Error handling

The input values for $rX[n+1]$ must always be at least 0.0000001 greater than the values for $rX[n]$. In the event of an error the variable *sErrDescr* indicates that at one point of the characteristic curve the values are not monotonically increasing.

VAR_INPUT

```
rX      : REAL;
rX01    : REAL;
rX02    : REAL;
rY01    : REAL;
rY02    : REAL;
bLmt    : BOOL;
```

rX: Input value of the characteristic curve.

rX01: X-value for interpolation point P1.

rX02: X-value for interpolation point P2.

rY01: Y-value for interpolation point P1.

rY02: Y-value for interpolation point P2.

bLmt: Limit for the output value *rY*.

VAR_OUTPUT

```
rY      : REAL;
bErr    : BOOL;
sErrDescr : T_MAXSTRING;
```

rY: Calculated output value of the characteristic curve.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description
01: Error: rX01 must not be equal to rX02.

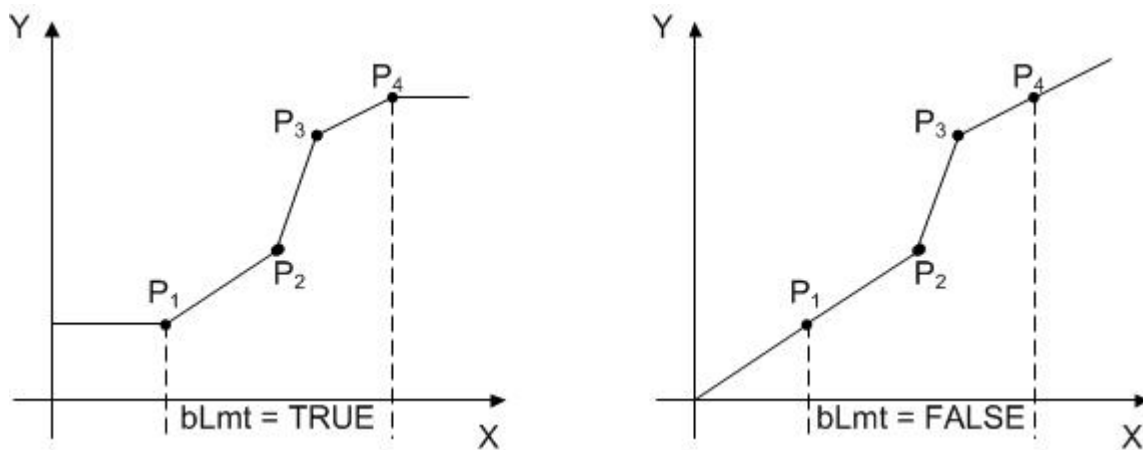
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5.3 FB_BA_Chrc04



The function block *FB_BA_Chrc04* represents a linear interpolation with 4 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points $[rX1/rY1]$ to $[rX4/rY4]$. If the input variable *bLmt* is TRUE, *rY* is limited by *rY01* and *rY04*. If *bLmt* is FALSE, *rY* is not limited.



Error handling

The input values for $rX[n+1]$ must always be at least 0.0000001 greater than the values for $rX[n]$. In the event of an error the variable *sErrDescr* indicates that at one point of the characteristic curve the values are not monotonically increasing.

VAR_INPUT

```
rX      : REAL;
rX01    : REAL;
rX02    : REAL;
rX03    : REAL;
rX04    : REAL;
rY01    : REAL;
rY02    : REAL;
rY03    : REAL;
rY04    : REAL;
bLmt    : BOOL;
```

rX: Input value of the characteristic curve.

rX01: X-value for interpolation point P1.

rX02: X-value for interpolation point P2.

rX03: X-value for interpolation point P3.

rX04: X-value for interpolation point P4.

rY01: Y-value for interpolation point P1.

rY02: Y-value for interpolation point P2.

rY03: Y-value for interpolation point P3.

rY04: Y-value for interpolation point P4.

bLmt: Limit for the output value *rY*.

VAR_OUTPUT

```
rY      : REAL;
bErr     : BOOL;
sErrDescr : T_MAXSTRING;
```

rY: Calculated output value of the characteristic curve.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

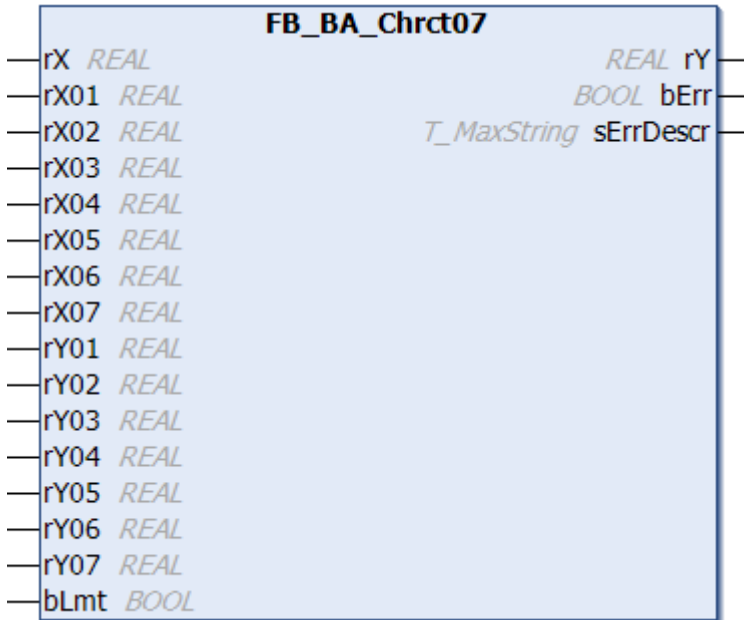
Error description

01: Error: at the specified element. The sequence must always be $rX01 > rX02 > rXn$ or $rX01 < rX02 < rXn$.

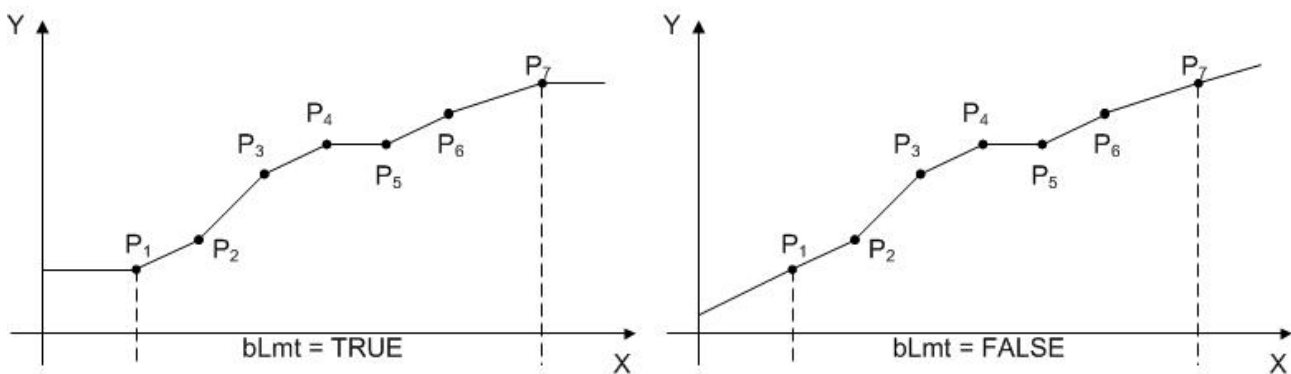
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5.4 FB_BA_Chrct07



The function block **FB_BA_Chrct07** represents a linear interpolation with 7 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points `[rX1/rY1]` to `[rX7/rY7]`. If the input variable `bLmt` is TRUE, `rY` is limited by `rY01` and `rY07`. If `bLmt` is FALSE, `rY` is not limited.



Error handling

The input values for `rX[n+1]` must always be at least 0.0000001 greater than the values for `rX[n]`. In the event of an error the variable `sErrDescr` indicates that at one point of the characteristic curve the values are not monotonically increasing.

VAR_INPUT

```

rX      : REAL;
rX01    : REAL;
rX02    : REAL;
rX03    : REAL;
rX04    : REAL;
rX05    : REAL;
rX06    : REAL;
rX07    : REAL;
rY01    : REAL;
rY02    : REAL;
rY03    : REAL;

```

```

rY04 : REAL;
rY05 : REAL;
rY06 : REAL;
rY07 : REAL;
bLmt : BOOL;

```

rX: Input value of the characteristic curve.

rX01: X-value for interpolation point P1.

rX02: X-value for interpolation point P2.

rX03: X-value for interpolation point P3.

rX04: X-value for interpolation point P4.

rX05: X-value for interpolation point P5.

rX06: X-value for interpolation point P6.

rX07: X-value for interpolation point P7.

rY01: Y-value for interpolation point P1.

rY02: Y-value for interpolation point P2.

rY03: Y-value for interpolation point P3.

rY04: Y-value for interpolation point P4.

rY05: Y-value for interpolation point P5.

rY06: Y-value for interpolation point P6.

rY07: Y-value for interpolation point P7.

bLmt: Limit for the output value *rY*.

VAR_OUTPUT

```

rY      : REAL;
bErr    : BOOL;
sErrDescr : T_MAXSTRING;

```

rY: Calculated output value of the characteristic curve.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

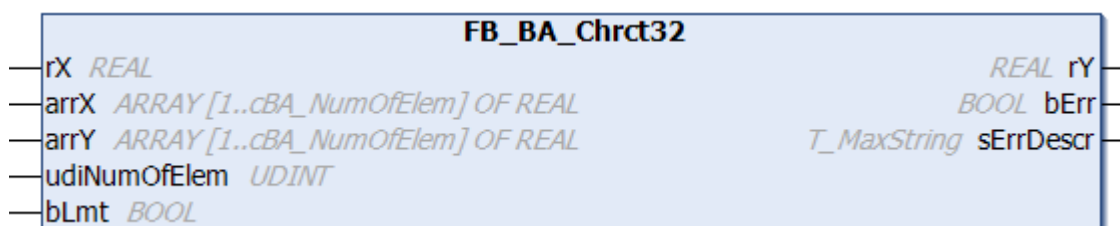
Error description

01: Error: at the specified element. The sequence must always be $rX01 > rX02 > rXn$ or $rX01 < rX02 < rXn$.

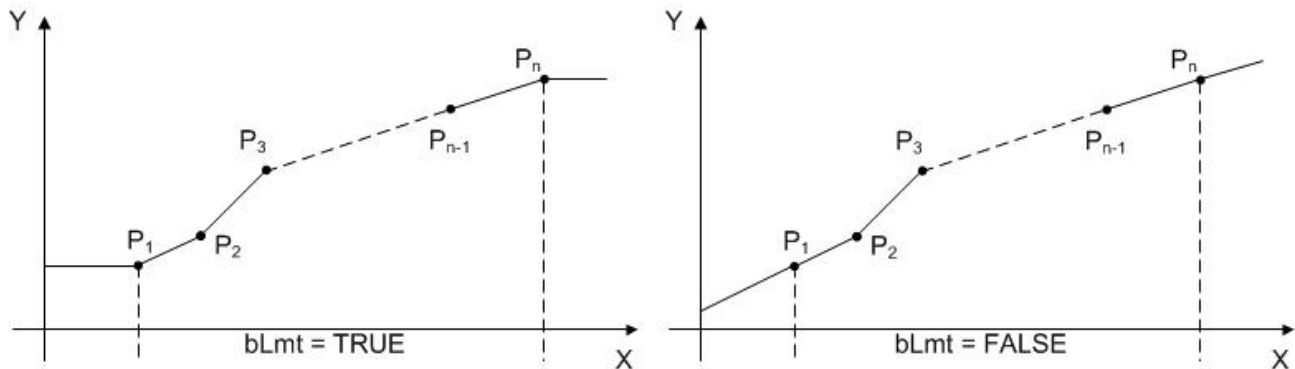
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5.5 FB_BA_Chrc32



The function block FB_BA_Chrc32 represents a linear interpolation with up to 32 interpolation points and can be used to generate a characteristic curve. In contrast to the "smaller" interpolation function blocks FB_BA_Chrc02 [► 583], FB_BA_Chrc04 [► 584] and FB_BA_Chrc07 [► 586], and in the interest of clarity, the interpolation points are determined via field variables $[arrX[1]/arrY[1]$ to $[arrX[n]/arrY[n]]$. If the input variable *bLmt* is TRUE, *rY* is limited by *arrY[1]* and *arrY[n]*. If *bLmt* is FALSE, *rY* is not limited.



Error handling

The input values for $rX[n+1]$ must always be at least 0.0000001 greater than the values for $rX[n]$. In the event of an error the variable *sErrDescr* indicates that at one point of the characteristic curve the values are not monotonically increasing.

The parameter for the number of interpolation points, *diNumOfElem*, must be in the range 2..32.

VAR_INPUT

```
rX      : REAL;
arrX    : ARRAY [1..cBA_NumOfElem] OF REAL;
arrY    : ARRAY [1..cBA_NumOfElem] OF REAL;
diNumOfElem : DINT (2..32);
bLmt    : BOOL;
```

rX: Input value of the characteristic curve

arrX: Field with the X-values for the interpolation points.

arrY: Field with the Y-values for the interpolation points.

diNumOfElem: Number of interpolation points. Internally limited to values between 2 and 32.

bLmt: Limit for the output value *rY*.

VAR_OUTPUT

```
rY      : REAL;
bErr    : BOOL;
sErrDescr : T_MAXSTRING;
```

rY: Calculated output value of the characteristic curve.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

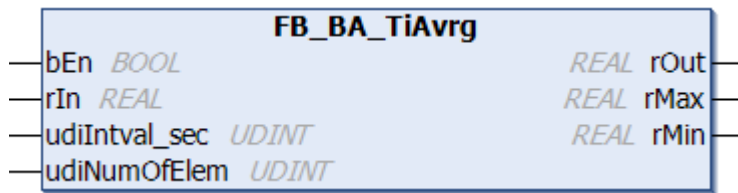
Error description

01: Error: at the specified element. The sequence must always be $rX01 > rX02 > rXn$ or $rX01 < rX02 < rXn$.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5.6 FB_BA_TiAavg



The function block *FB_BA_TiAavg* calculates the arithmetic mean value of an analog value that was logged over a certain period. Discrete values are written into a FIFO buffer. *udiIntval_sec* specifies the time interval [s] over which the values are logged and written into the FIFO. Values are written if the input *bEn* is TRUE. The variable *udiNumOfElem* is used to determine the size of the FIFO buffer. It is limited to 1..512. The function block can be used for calculating an hourly mean outside temperature over a day, for example. In this case *udiNumOfElem* would be 24 and *udiIntval_sec* would be 3600 seconds. *bEn* is the general enable of the function block. If *bEn* = FALSE, the FIFO buffer within the function block is deleted completely, and no data are recorded.

Example:

udiNumOfElem = 5

	First cycle		Second cycle		Third cycle		Fourth cycle	
	rIn	rOut	rIn	rOut	rIn	rOut	rIn	rOut
t0	2	$2/1 = 2$	6	$(4+6+7+7+6)/5 = 6$	1	$(7+6+5+4+1)/5 = 4.6$	3	rIn = 3
t1	4	$(2+4)/2 = 3$	5	$(6+7+7+6+5)/5 = 6.25$	2	$(6+5+4+1+2)/5 = 3.6$	1.5	rIn = 1.5
t2	6	$(2+4+6)/3 = 4$	4	$(7+7+6+5+4)/5 = 5.8$	4	$(5+4+1+2+4)/5 = 3.2$		
t3	7	$(2+4+6+7)/4 = 4.75$	2	$(7+7+6+5+4)/5 = 5.8$	5	$(4+1+2+4+5)/5 = 3.2$		
t4	7	$(2+4+6+7+7)/5 = 5.2$	1	$(7+7+6+5+4)/5 = 5.8$	4	$(1+2+4+5+4)/5 = 3.2$		

VAR_INPUT

```

bEn      : BOOL;
rIn      : REAL;
udiIntval_sec : UDINT;
udiNumOfElem : UDINT;

```

bEn: Enables the function block.

rIn: Input value for averaging.

udiIntVal_SEC: Time interval [s] for writing new values into the FIFO. Internally limited to a value between 1 and 2147483.

udiNumOfElem: Size of the FIFO buffer. A change resets the previous averaging. Internally limited to a value between 1 and 512.

VAR_OUTPUT

```

rOut     : REAL;
rMax     : REAL;
rMin     : REAL;

```

rOut: Calculated mean value.

rMax: Largest value in the FIFO buffer.

rMin: Smallest value in the FIFO buffer.

Requirements

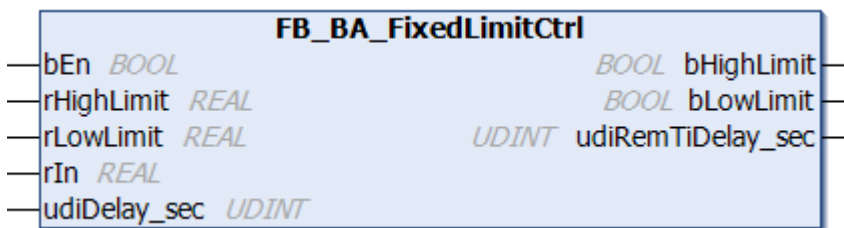
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.6 Monitoring functions

Function blocks

Name	Description
FB_BA_FdbCtrlBinary [► 591]	Feedback monitoring of an actuator by means of digital feedback.
FB_BA_FixedLimitCtrl [► 590]	Limit value monitoring of a fixed value.
FB_BA_SlidingLimitCtrl [► 592]	Sliding limit value monitoring.

6.1.2.3.2.1.5.6.1 FB_BA_FixedLimitCtrl



Function block for monitoring a fixed limit value.

The input *bEn* is used for enabling the function block.

A tolerance range is defined around the value *rIn* to be monitored.

The tolerance range results from an upper limit value *rInHighLimit* and a lower limit value *rInLowLimit*.

If the value *rIn* exceeds the upper limit value of the tolerance range, then the output *bHighLimit* becomes TRUE. A response delay of the output *bHighLimit* must be parameterized with the timer *udiDelay_sec*.

If the value *rIn* falls below the lower limit of the tolerance range, output *bLowLimit* becomes TRUE. A response delay of the output *bLowLimit* must be parameterized with the timer *udiDelay_sec*.

VAR_INPUT

```

bEn      : BOOL;
rHighLimit : REAL := 32;
rLowLimit : REAL := 16;
rIn      : REAL;
udiDelay_sec : UDINT;

```

bEn: Function block enable.

rHighLimit: Default upper limit value, preset to 32.

rLowLimit: Default lower limit value, preset to 16.

rIn: Input value to be monitored.

udiDelay_sec: Output response delay [s]. Internally limited to values between 0 and [Const.udiTiSec \[\[► 605\]\(#\)\]](#).

VAR_OUTPUT

```

bHighLimit : BOOL;
bLowLimit  : BOOL;
udiRemTiDelay_sec : UDINT;

```

bHighLimit: Upper limit value reached.

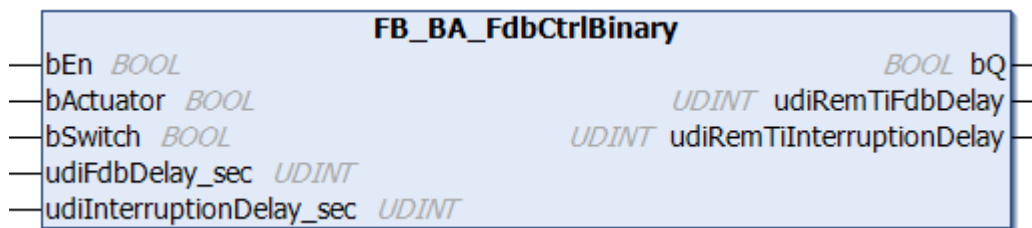
bLowLimit: Lower limit value reached.

udiRemTiDelay_sec: Time remaining after a limit value has been exceeded until either the output *bHighLimit* or *bLowLimit* responds.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.6.2 FB_BA_FdbCtrlBinary



The function block is used for feedback monitoring of an actuator by means of digital feedback. Application examples of the function block are, for example, an operation feedback monitoring, a process feedback monitoring or the run monitoring of a drive by means of limit switches.

The input *bEn* is used for enabling the function block. If *bEn* is FALSE, the message output *bQ* will always be FALSE.

The switching actuator output of the unit to be monitored is connected to the input *bActuator*. The *bSwitch* input is used to connect the feedback signal (e.g. differential pressure switch, flow monitor or limit switch).

By means of the timer *udiFdbDelay_sec* [s] a response delay of the feedback control after the start of the unit is set.

The second timer *udiInterruptionDelay_sec* [s] serves for a response delay of the feedback control after reaching the final state.

VAR_INPUT

```
bEn           : BOOL;
bActuator     : BOOL;
bSwitch       : BOOL;
udiFdbDelay_sec : UDINT;
udiInterruptionDelay_sec : UDINT;
```

bEn: Function block enable.

bActuator: Feedback of the switching output.

bSwitch: Feedback signal from the process.

udiFdbDelay_sec: Response delay [s] of the monitoring function when the actuator is started. Internally limited to values between 0 and [Const.udiTiSec](#) [► 605].

udiInterruptionDelay_sec: Response delay [s] of the monitoring function when the actuator has already been started successfully (e.g. pressure fluctuations when monitoring the running of a fan). Internally limited to values between 0 and [Const.udiTiSec](#) [► 605].

VAR_OUTPUT

```
bQ           : BOOL;
udiRemTiFdbDelay : UDINT;
udiRemTiInterruptionDelay : UDINT;
```

bQ: Output an error message if the feedback signal is not present within the parameterized time of *udiFdbDelay_sec*, or the feedback signal has been interrupted longer than after *udiInterruptionDelay_sec*.

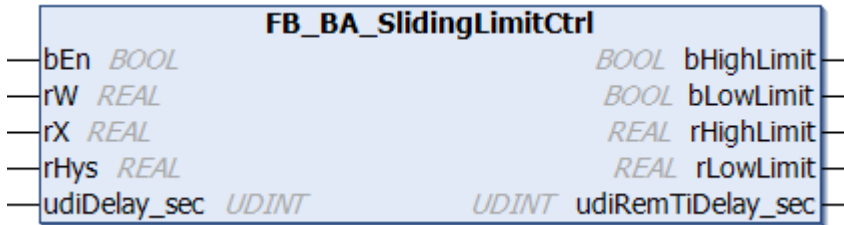
udiRemTiFdbDelay: Remaining time [s] until output *bErrOpn* is set.

udiRemTiInterruptDelay: Remaining time [s] until output *bErrSwi* is set.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.6.3 FB_BA_SlidingLimitCtrl

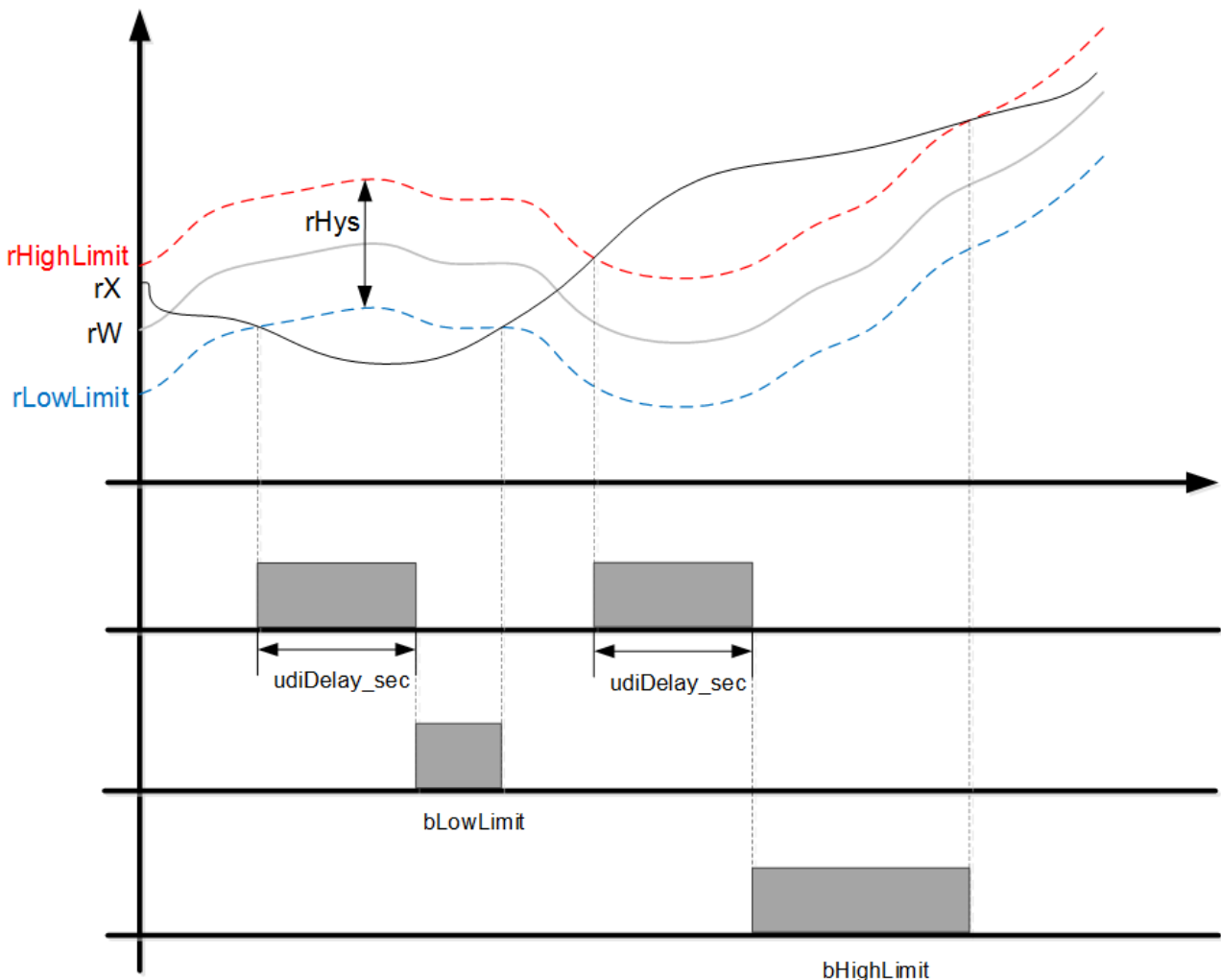


Function block for monitoring a floating setpoint.

The input *bEn* is used for enabling the function block.

To check the function of a control system, the actual value is compared with the setpoint of the controlled system.

If the deviation of setpoint and actual value is within the tolerance range *rHys*, then the control is OK. If the actual value deviates from the setpoint by an amount outside this tolerance range over a longer period of time, the timer **udiDelay_sec** is started. After the timer has expired, if the control deviation is permanent, either the output *bLowLimit* or *bHighLimit* TRUE of the function block outputs a message.



VAR_INPUT

```

bEn      : BOOL;
rW       : REAL;
rX       : REAL;
rHys     : REAL;
udiDelay_sec : UDINT;

```

bEn: Function block enable.

rW: Setpoint.

rX: Actual value.

rHys: Hysteresis.

udiDelay_sec: Output response delay [s]. Internally limited to values between 0 and Const.udiTiSec [► 605].

VAR_OUTPUT

```

bHighLimit : BOOL;
bLowLimit  : BOOL;
rHighLimit : REAL;
rLowLimit  : REAL;
udiRemTiDelay_sec : UDINT;

```

bHighLimit: Upper limit value reached.

bLowLimit: Lower limit value reached.

rHighLimit: Output of the upper limit value.

rLowLimit: Output of the lower limit value.

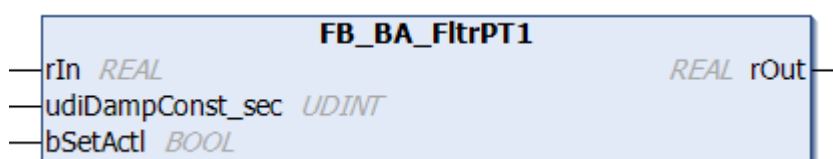
udiRemTiDelay_sec: Time remaining after a limit value has been exceeded until either the output *bHighLimit* or *bLowLimit* responds.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.7 Ramps, filters, controllers**Function blocks**

Name	Description
<u>FB_BA_FltrPT1</u> [► 593]	First order filter
<u>FB_BA_RampLmt</u> [► 594]	Ramp limitation
FB_BA_SeqCtrl	Sequence controller (function block in Tc3_BA_Common)
FB_BA_SeqLink	Sequence linker (function block in Tc3_BA_Common)
FB_BA_PIDCtrl	PID controller (function block in Tc3_BA_Common)

6.1.2.3.2.1.5.7.1 FB_BA_FltrPT1

First order filter.



When the function block is first called (system start), the output *rOut* is automatically set (once) to the input *rIn*.

VAR_INPUT

```
rIn      : REAL;
udiDampConst_sec : UDINT;
bSetAct1 : BOOL;
```

rIn: Input signal

udiDampConst_sec: Filter time constant [s]. Internally limited to values between 0 and 86400.

bSetAct1: A rising edge at this input sets the output value *rOut* to the input value *rIn*.

VAR_OUTPUT

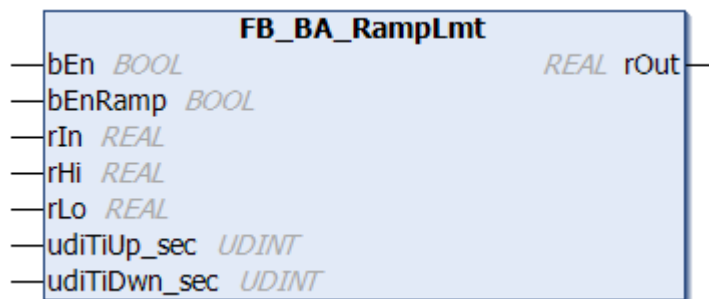
```
rOut : REAL;
```

rOut: Filtered output signal.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

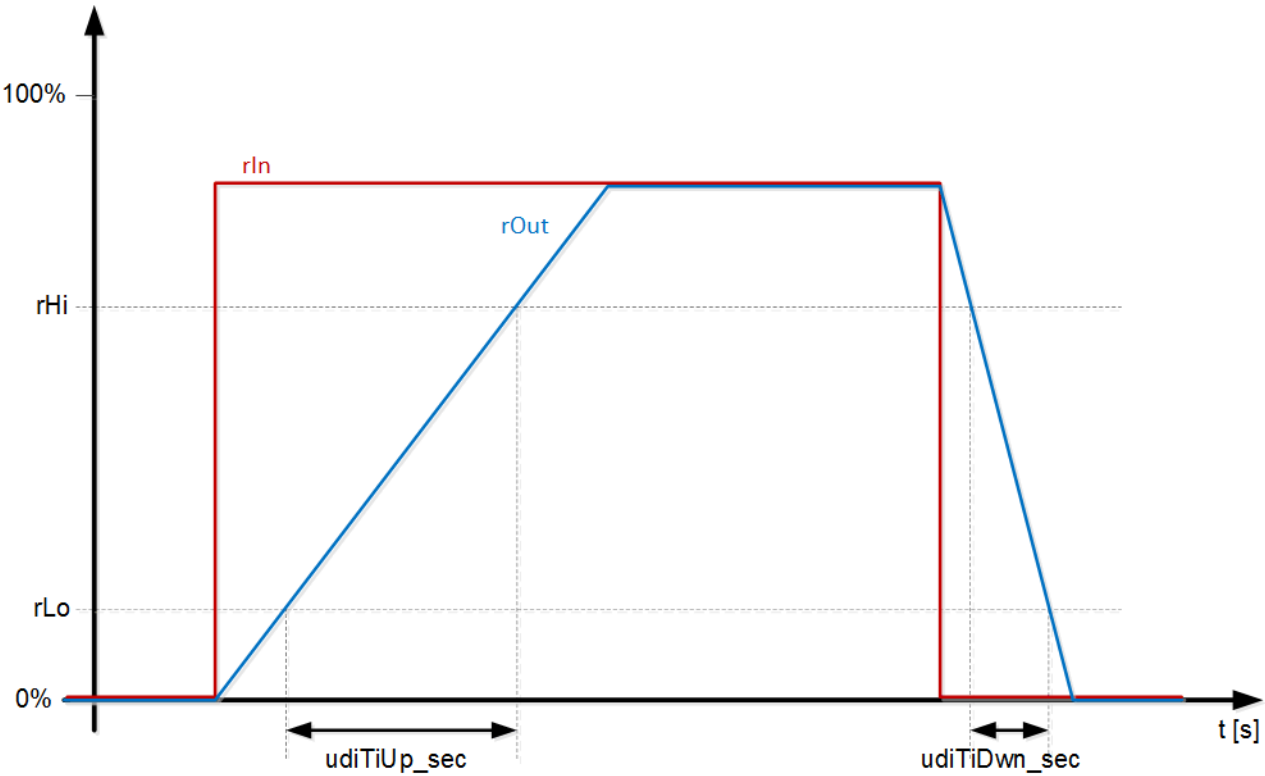
6.1.2.3.2.1.5.7.2 FB_BA_RampLmt



The function block limits the increase or decrease speed of an input signal.

An increase of *rIn* results in the output *rOut* to be limited to the slope of $(rHi-rLo)/udiTiUp$.

A decrease of *rIn* results in the output *rOut* to be limited to the slope of $(rHi-rLo)/udiTiUp$.



VAR_INPUT

```
bEn      : BOOL;  
bEnRamp  : BOOL;  
rIn      : REAL;  
rHi      : REAL;  
rLo      : REAL;  
udiTiUp_sec : UDINT;  
udiTiDwn_sec : UDINT;
```

- bEn:** Enable function block if FALSE, in which case $rOut = 0.0$.
- bEnRamp:** Enable ramp limitation if FALSE, in which case $rOut = rIn$.
- rIn:** Input value of the ramp function
- rHi:** Upper interpolation point for calculating the ramps.
- rLo:** Lower interpolation point for calculating the ramps. rHi must be greater than rLo , otherwise an error is output!
- udiTiUp_sec:** Rise time [s].
- udiTiDwn_sec:** Fall time [s]

VAR_OUTPUT

```
rOut      : REAL;
```

rOut: Output signal, slope-limited through the ramps

Requirements

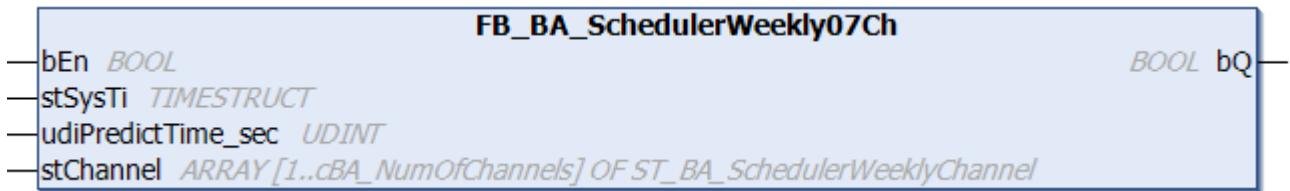
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.8 Calendar

Function blocks

Name	Description
FB_BA_SchedulerWeeklyXXCh [► 596]	Weekly scheduler
FB_BA_CalenderXXCh [► 597]	Yearly scheduler

6.1.2.3.2.1.5.8.1 FB_BA_SchedulerWeeklyXXCh



Weekly timer with 1, 7 or 28 timer channels.

The function block *FB_BA_SchedulerWeekly07Ch* is described as an example.

The function block is used to enter a total of up to 7 switch-on periods.

Each switch-on period can be assigned an switch-on time [hh:mm:ss] and a switch-off time [hh:mm:ss].

The variables *bMonday* to *bSunday* can be used to select on which days of the week the switch-on period should be active.

A switch-on period is only active if the variable *bEn* of the channel is set to TRUE.

For irregular but recurring events, the variable *bResetAfterOn* can be set to TRUE. This will automatically reset the enable of channel *bEn* to FALSE after the event has finished.

To facilitate data entry, a rising edge at *bAllActive* sets *bEn* and all days of the week (*bMonday* to *bSunday*) to TRUE.

The function block is only active if a TRUE signal is present at *bEn*.

For demand-dependent switch-on optimization, switching on of the output *bQ* can be brought forward by the time of the variable *udiPredictTime_sec*.



The switch-on and switch-off points of a channel must be in the same year. The switch-off point must not be earlier than the switch-on point. Otherwise the switch-off point is automatically corrected and set to the same value as the switch-on point. If the switch-on point is equal to the switch-off point, the channel remains off.

VAR_INPUT

```
bEn          : BOOL;
stSysTi      : Timestruct;
udiPredictTime_sec : UDINT;
```

bEn: General function block enable.

stSysTi: Structure with the local NT system time (see *Timestruct*).

udiPredictTime_sec: Precalculated switch-on time. Internally limited to values between 0 and 43200.

VAR_OUTPUT

```
bQ : BOOL;
```

bQ: Switching output

VAR_IN_OUT

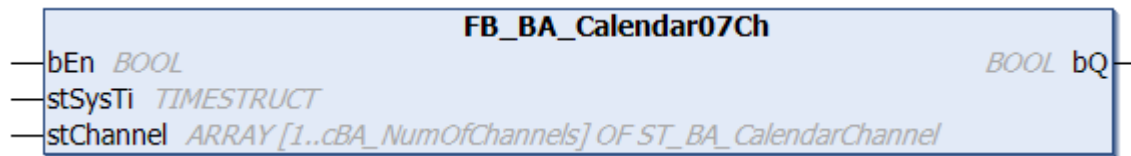
```
arrChannel: ARRAY [1..cBA_NumOfChannels] OF ST_BA_SchedulerWeeklyChannel;
```

arrChannel: Weekly scheduler; with the single-channel function block, the name of the variable is *stChannel* (see *ST_BA_SchedulerWeeklyChannel* [► 604]). Internally limited to the respective number of possible channels via the variable *cBA_NumOfChannels*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.8.2 FB_BA_CalenderXXCh



Yearly scheduler with 1, 7 or 28 channels.

The function block *FB_BA_Calender07Ch* is described as an example.

This function block is used to enter periods such as school holidays or company holidays.

The function block is enabled by the input variable *bEnable*.

The input *stSsyTi* is linked to the current system time.

If the time switching condition is fulfilled, output *bQ* is set.

Within the calendar, a time period is described by a switch-on date [day, month, hour, minute] and a switch-off date [day, month, hour, minute].

A switch-on period is only active if the variable *bEn* of the channel is set to TRUE.

For irregular but recurring periods, the variable *bResetAfterOn* can be set to TRUE. The enable parameter *bEn* is then automatically reset to FALSE after the time has elapsed.



The switch-on and switch-off points of a channel must be in the same year. The switch-off point must not be earlier than the switch-on point. Otherwise the switch-off point is automatically corrected and set to the same value as the switch-on point.

If the switch-on point is equal to the switch-off point, the channel remains off.

VAR_INPUT

```
bEn          : BOOL;
stSysTi      : Timestruct;
```

bEn: General function block enable.

stSysTi: Structure with the local NT system time (see *Timestruct*).

VAR_IN_OUT

```
arrChannel: ARRAY [1..7] OF ST_BA_CalenderChannel;
```

arrChannel: Yearly scheduler; with the single-channel function block, the name of the variable is *stChannel* (see *ST_BA_CalenderChannel* [► 603]). Internally limited to the respective number of possible channels via the variable *cBA_NumOfChannels*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2 DUTs

6.1.2.3.2.2.1 Enums

Enumerations

Name	Description
E_BA_PosMod [► 598]	Enumerator for the definition of the positioning mode.
E_BA_ShObjType [► 598]	Enumerator for selecting the shading object type.
E_BA_Sensor	Enumerator for selecting a sensor type for measuring analog values.
E_BA_Terminal_KL	Enumerator for selecting the respective Bus Terminal.

6.1.2.3.2.2.1.1 E_BA_PosMod

Enumerator for the definition of the positioning mode.

```

TYPE E_BA_PosMod :
(
    PosModFix:= 0,
    PosModTab,
    PosModMaxIndc
);
END_TYPE

```

PosModFix: The shutter height is a fixed value, which is set at function block [FB_BA_SunPrtc](#) [[► 522](#)] via the value *IrFixPos* [%].

PosModTab: The height positioning takes place with the help of a table of 6 interpolation points, 4 of which are parameterizable. A blind position in relation to the position of the sun is then calculated from these points by linear interpolation. For a more detailed description please refer to [FB_BA_BldPosEntry](#) [[► 474](#)].

PosModMaxIndc: The positioning takes place with specification of the maximum desired incidence of light.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.1.2 E_BA_ShObjType

Enumerator for selecting the shading object type.

```

TYPE E_BA_ShObjType :
(
    ObjTypeTetragon := 0,
    ObjTypeGlobe
);
END_TYPE

```

ObjTypeTetragon: Object type is a rectangle.

ObjTypeGlobe: Object type is a ball.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.2 Structures

Structures

Name	Description
ST_BA_BldPosTab [▶ 599]	Structure of the interpolation point entries for the height adjustment of the blind.
ST_BA_Cnr [▶ 599]	Information about window corners.
ST_BA_Fcd [▶ 600]	Facade-specific data for activating the automatic functions.
ST_BA_FcdElem [▶ 600]	List entry for a facade element (window).
ST_BA_ShObj [▶ 601]	List entry for a shading object
ST_BA_SpRmT [▶ 601]	Room temperature setpoints.
ST_BA_Sunbld [▶ 602]	Structure of the blind positioning telegram.
ST_BA_SunBldScn [▶ 603]	Table entry for a blind scene.
ST_BA_CalenderChannel [▶ 603]	Input of calendar entries.
ST_BA_SchedulerWeeklyChannel [▶ 604]	Input of time switch entries.

6.1.2.3.2.2.2.1 ST_BA_BldPosTab

Structure of the interpolation point entries for the height adjustment of the blind.

```

TYPE ST_BA_BldPosTab:
STRUCT
    rSunElv    : ARRAY[0..5] OF REAL;
    rPos       : ARRAY[0..5] OF REAL;
    bVld       : BOOL;
END_STRUCT
END_TYPE

```

rSunElv / rPos: The 6 interpolation points that are transferred, wherein the array elements 0 and 5 represent the automatically generated edge elements mentioned above.

bVld: Validity flag for the function block [FB_BA_SunPrtc](#) [[▶ 522](#)]. It is set to TRUE by the function block [FB_BA_BldPosEntry](#) [[▶ 474](#)] if the data entered correspond to the validity criteria described.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.2.2 ST_BA_Cnr

Information about window corners.

```

TYPE ST_BA_Cnr :
STRUCT
    rX         : REAL;
    rY         : REAL;
    bShdd      : BOOL;
END_STRUCT
END_TYPE

```

rX: X-coordinate of the window (on the facade).

rY: Y-coordinate of the window (on the facade).

bShdd: Information as to whether this corner is in the shade: *bShdd*=TRUE: Corner is in the shade.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.3 ST_BA_Fcd

Facade-specific data at room level for activating the automatic functions.

```

TYPE ST_BA_Fcd:
STRUCT
  rSunPrtcAngl      : REAL;
  rSunPrtcPos       : REAL;
  rFcdThAutoPos     : REAL;
  rFcdThAutoAngl    : REAL;
  bFcdThAutoEn      : BOOL;
  bThAutoEn         : BOOL;
  bTwiLgtAutoEn     : BOOL;
  bSunPrtcEn        : BOOL;
END_STRUCT
END_TYPE

```

rSunPrtcAngl: Sun protection: Current calculated position [%] for the blinds.

rSunPrtcPos: Sun protection: Current calculated louvre angle [°] for the blinds.

rFcdThAutoPos: Thermo-automatic function for whole facade: Currently valid position [%] for the blinds (heating or cooling position).

rFcdThAutoAngl: Thermo-automatic function for whole facade: Currently valid louvre angle [°] for the blinds (heating or cooling position).

bFcdThAutoEn: Thermo-automatic function for whole facade enabled.

bThAutoEn: Thermo-automatic function enabled.

bTwiLgtAutoEn: Automatic twilight function enabled.

bSunPrtcEn: Automatic sun protection enabled.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.4 ST_BA_FcdElem

List entry for a facade element (window).

```

TYPE ST_BA_FcdElem:
STRUCT
  rWdwWdth      : REAL;
  rWdwHght      : REAL;
  stCnr         : ARRAY [1..4] OF ST_BA_Cnr;
  diGrp         : DINT;
  bVld          : BOOL;
END_STRUCT
END_TYPE

```

rWdwWdth: Width of the window.

rWdwHght: Height of the window.

stCnr: Coordinates of the window corners and information as to whether this corner point is in the shade; see [ST_BA_Cnr](#) [► 599].

bVld: Plausibility of the data entered: *bVld*=TRUE: Data are plausible.

diGrp: Group membership of the element.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.5 ST_BA_ShObj

List entry for a shading object.

```

TYPE ST_BA_ShObj :
STRUCT
  rP1x      : REAL;
  rP1y      : REAL;
  rP1z      : REAL;
  rP2x      : REAL;
  rP2y      : REAL;
  rP2z      : REAL;
  rP3x      : REAL;
  rP3y      : REAL;
  rP3z      : REAL;
  rP4x      : REAL;
  rP4y      : REAL;
  rP4z      : REAL;
  rMx       : REAL;
  rMy       : REAL;
  rMz       : REAL;
  rRads     : REAL;
  diBegMth  : USINT;
  diEndMth  : USINT;
  eType     : E_BA_ShObjType;
  bVld      : BOOL;
END_STRUCT
END_TYPE

```

rP1x .. rP4z: Corner coordinates. Of importance only if the element is a square.

rMx .. rMz: Center coordinates. Of importance only if the element is a ball.

rRads: Radius of the ball. Of importance only if the element is a ball.

diBegMth: Beginning of the shading period (month).

diEndMth: End of the shading period (month).

eType: Object type, see [E_BA_ShObjType](#) [► 598].

bVld: Plausibility of the data: *bVld*=TRUE: Data are plausible.

Remark about the shading period:

The entries for the months may not be 0 or greater than 12, otherwise all combinations are possible.

Examples:

Start=1, End=1: shading in January.

Start=1, End=5: shading from the beginning of January to the end of May.

Start=11, End=5: shading from the beginning of November to the end of May (the following year).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.6 ST_BA_SpRmT

Room temperature setpoints.

```

TYPE ST_BA_SpRmT :
STRUCT
    rPrtcHtg      : REAL := 12.0;
    rEcoHtg       : REAL := 15.0;
    rPreCmfHtg    : REAL := 19.0;
    rCmfHtg       : REAL := 21.0;
    rPrtcCol      : REAL := 40.0;
    rEcoCol       : REAL := 35.0;
    rPreCmfCol    : REAL := 28.0;
    rCmfCol       : REAL := 24.0;
END_STRUCT
END_TYPE

```

The values in the structure are defined with the preset values.

The variables have the following meaning:

rPrtcHtg: Protection Heating.

rEcoHtg: Economy Heating.

rPreCmfHtg: Pre-Comfort Heating.

rCmfHtg: Comfort Heating.

rPrtcCol: Protection Cooling.

rEcoCol: Economy Cooling.

rPreCmfCol: Pre-Comfort Cooling.

rCmfCol: Comfort Cooling.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.2.7 ST_BA_SunBld

Structure of the blind positioning telegram.

```

TYPE ST_BA_SunBld:
STRUCT
    rPos          : REAL;
    rAngl         : REAL;
    bManUp        : BOOL;
    bManDwn       : BOOL;
    bManMod       : BOOL;
    bActv         : BOOL;
END_STRUCT
END_TYPE

```

rPos: Transferred shutter height [%].

rAngl: Transferred louvre position [°].

bManUp: Manual command: blind up.

bManDwn: Manual command: blind down.

bManMod: TRUE: Manual mode is active. FALSE: Automatic mode is active.

bActv: The sender of the telegram is active. This bit is only evaluated by the priority control [FB_BA_SunBldPrioSwi4](#) [► 512] or [FB_BA_SunBldPrioSwi8](#) [► 513]. The sun protection actuators [FB_BA_SunBldActr](#) [► 504] and [FB_BA_RolBldActr](#) [► 496] ignore it.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.8 ST_BA_SunBldScn

Table entry for a blind scene.

```

TYPE ST_BA_SunBldScn:
STRUCT
  rPos      : REAL;
  rAngl     : REAL;
END_STRUCT
END_TYPE

```

rPos: Shutter height [%].

rAngl: Louvre position [°].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.9 ST_BA_CalenderChannel

Structure for entering calendar entries.

```

TYPE ST_BA_CalenderChannel:
STRUCT
  udiOn_Day      : UDINT(1..31);
  udiOn_Month    : UDINT(1..12);
  udiOn_hh       : UDINT(0..23);
  udiOn_mm       : UDINT(0..59);
  udiOff_Day     : UDINT(1..31);
  udiOff_Month   : UDINT(1..12);
  udiOff_hh      : UDINT(0..23);
  udiOff_mm      : UDINT(0..59);
  bEn            : BOOL;
  bResetAfterOn  : BOOL;
  bQ             : BOOL;
END_STRUCT
END_TYPE

```

udiOn_Day: Switch-on point for day.

udiOn_Month: Switch-on point for month.

udiOn_hh: Switch-on point for hour.

udiOn_mm: Switch-on point for minute.

udiOff_Day: Switch-off point for day.

udiOff_Month: Switch-off point for month.

udiOff_hh: Switch-off point for hour.

udiOff_mm: Switch-off point for minute.

bEn: TRUE -> Enable channel, FALSE -> bQ = FALSE

bResetAfterOn: One-time and non-recurring switching-on.

bQ: Channel status output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.10 ST_BA_SchedulerWeeklyChannel

Structure for entering time switch entries.

```

TYPE ST_BA_SchedulerWeeklyChannel:
STRUCT
    udiOn_hh      : UDINT(0..23);
    udiOn_mm      : UDINT(0..59);
    udiOn_ss      : UDINT(0..59);
    udiOff_hh     : UDINT(0..23);
    udiOff_mm     : UDINT(0..59);
    udiOff_ss     : UDINT(0..59);
    bAllActive    : BOOL;
    bEn           : BOOL;
    bMonday       : BOOL;
    bTuesday      : BOOL;
    bWednesday    : BOOL;
    bThursday     : BOOL;
    bFriday       : BOOL;
    bSaturday     : BOOL;
    bSunday       : BOOL;
    bResetAfterOn : BOOL;
    bQ            : BOOL;
END_STRUCT
END_TYPE

```

udiOn_hh: Switch-on point for hour.

udiOn_mm: Switch-on point for minute.

udiOn_ss: Switch-on point for second.

udiOff_hh: Switch-off point for hour.

udiOff_mm: Switch-off point for minute.

udiOff_ss: Switch-off point for second.

bAllActive: Activation of the timer condition for all weekdays.

bEn: TRUE -> Enable channel, FALSE -> bQ = FALSE.

bMonday: Switch-on point for Monday.

bTuesday: Switch-on point for Tuesday.

bWednesday: Switch-on point for Wednesday.

bThursday: Switch-on point for Thursday.

bFriday: Switch on point for Friday.

bSaturday: Switch-on point for Saturday.

bSunday: Switch-on point for Sunday.

bResetAfterOn: One-time and non-recurring switching-on.

bQ: Channel status output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.3 GVLs

6.1.2.3.2.3.1 Constants

Global constants

```
VAR_GLOBAL CONSTANT
  rClsZero      : REAL := 0.00001;

  udiNoActvPrio : UDINT := 4294967295;

  udiTiSec      : UDINT := 4294967295;

  wSUNDAY       : WORD := 0;
  wMONDAY       : WORD := 1;
  wTUESDAY      : WORD := 2;
  wWEDNESDAY    : WORD := 3;
  wTHURSDAY     : WORD := 4;
  wFRIDAY       : WORD := 5;
  wSATURDAY     : WORD := 6;

  TimeValue24h_ms : UDINT := 86400000;
END_VAR
```

rClsZero: Reference value to avoid division by zero.

udiNoActvPrio: The value of the constants indicates that no priority is active.

udiTiSec: Constant for specifying a time in seconds.

wSUNDAY: Constant value for Sunday.

wMONDAY: Constant value for Monday.

wTUESDAY: Constant value for Tuesday.

wWEDNESDAY: Constant value for Wednesday.

wTHURSDAY: Constant value for Thursday.

wFRIDAY: Constant value for Friday.

wSATURDAY: Constant value for Saturday.

TimeValue24h_ms: Time value for 24 hours in milliseconds.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.3.2 Parameter

Global parameters

```
VAR_GLOBAL CONSTANT
  usiMaxSunBldScn : USINT := 20;
  uiMaxRowFcd     : UINT  := 10;
  uiMaxColumnFcd  : UINT  := 20;
  uiMaxShdObj     : UINT  := 20;
  udiMaxDataFileSize : UDINT := 100000;
END_VAR
```

usiMaxSunBldScn: Maximum number of scenes that are processed by the function block [FB_BA_SunBldScn](#) [► 514].

uiMaxRowFcd: Maximum number of floors for which the shading correction applies (horizontal arrangement of windows).

uiMaxColumnFcd: Maximum number of axes for which the shading correction applies (vertical arrangement of windows).

uiMaxShdObj: Maximum number of shading objects that cast shadows on the facade.

udiMaxDataFileSize: Maximum file size for the Excel list (in bytes), which is read by the function blocks [FB_BA_RdFcdElemLst \[► 488\]](#) and [FB_BA_RdShdObjLst \[► 492\]](#).

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.3 PLC project templates

6.1.3.1 Standard PLC BA template

The *standard PLC BA template* is a PLC template for a standard TF8040 project.

With the *standard PLC BA template* all necessary libraries and project settings are loaded for an easy start with TF8040. The template is very suitable for making a start with a TF8040 project.

Structure

The *standard project template* contains all necessary declarations, FB calls and libraries for the first commissioning of a TF8040 controller.

Template for a *Building Automation PLC project* with the following basic content:

Settings

- **Task**
 - PLC cycle time: 45 ms

References

- **Tc3_BA2**
- **Tc3_BA2_Common**
- **Tc3_BACnetRev14**

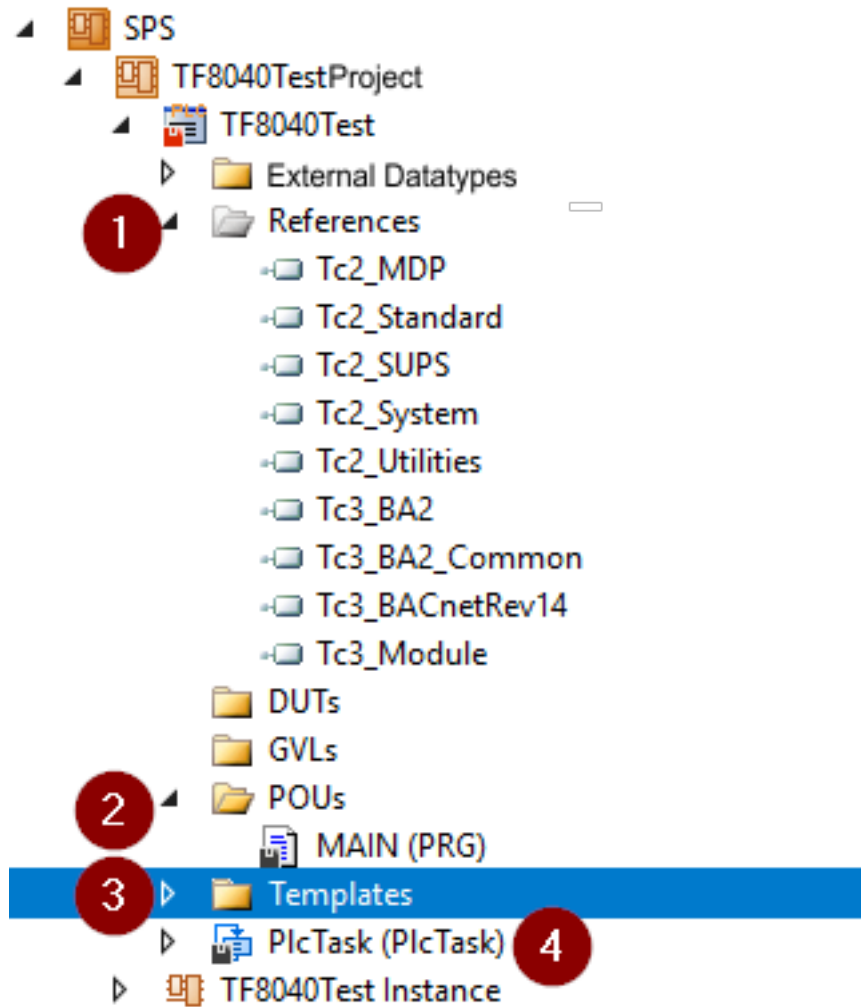
Programs

- **MAIN**
 - **FB_BA_DPAD**
Contains preconfigured levels for small projects.
 - **Project structure**
Contains declarations and FB calls for:
 - Control cabinet
 - Device

Templates

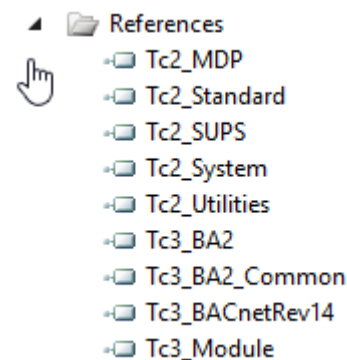
Contains all the templates required for compiling the project template without errors.

Structure in the solution tree



- **References**

All the libraries required for a TF8040 project are loaded here.



- **MAIN POU**

The standard project structure is called in the MAIN POU.



This template provides a project structure as an application example with eight levels based on an example from VDI 3814.

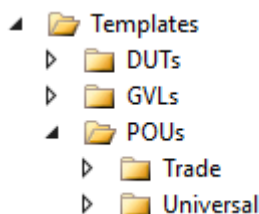
```

1  PROGRAM MAIN
2  VAR
3      DPAD      : FB_BA_CarelessDPAD := (
4          aIdentifier := [
5              (* Level 1 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.eLocation, sSeparator_ObjectName := '-'),
6              (* Level 2 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.eBuilding, sSeparator_ObjectName := '-'),
7              (* Level 3 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.eControlCabinet, sSeparator_ObjectName := '-'),
8              (* Level 4 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.eFloor, sSeparator_ObjectName := '-'),
9              (* Level 5 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.eTrade, sSeparator_ObjectName := '-'),
10             (* Level 6 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.ePlant, sSeparator_ObjectName := '-'),
11             (* Level 7 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.eAggregate, sSeparator_ObjectName := '-'),
12             (* Level 8 *) (eMode:=E_BA_DPADMode.eInclude, eNodeType:=E_BA_NodeType.eFunction, sSeparator_ObjectName := '-'),
13         ]
14     );
15
16     // Level 1 Standort bzw. Ort oder Stadt
17     // Level 1 Site respectively location or town
18     SiteA      : FB_BA_View := (
19         sObjectName := 'A',
20         sDescription := 'A',
21         eDPADMode := E_BA_DPADMode.eInclude
22     );
23
24     // Level 1 Standort bzw. Ort oder Stadt Site respectively location or town
25     SiteA();
26     // Level 2 Gebäude Building
27     B();
28     // Level 3 ISP -> Informationsschwerpunkt information focal point
29     IFP();
30     // Level 4 Stockwerk --> O01 = Obergeschoss 01, U01 = Untergeschoss 01, E00 = Erdgeschoss floor -> F01 = floor 01, B01 = bas
31     F01();
32     // Level 5 Gewerk Wärmeversorgungsanlagen Heat supply systems
33     H();
34     // Level 5 Gewerk Allgemein Trade General
35     G();
36     // Level 5 Gewerk Lufttechnische Anlagen Trade ventilation system
37     AC();
38     // Level 5 Gewerk Raumautomation Trade room automation
39
40     // Level 5 Gewerk Außengeräte Trade Outdoor devices
41     OD();

```

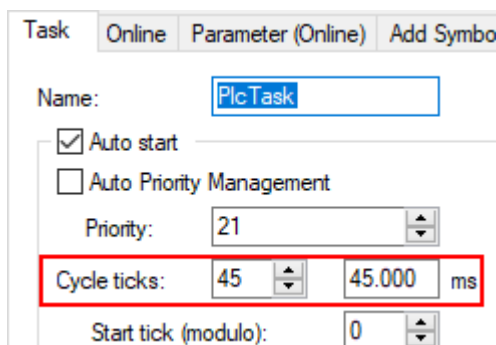
• Templates

All templates required to get started are located in this folder. They enable error-free compilation of the PLC project template. Further templates for the implementation of project-specific applications can be found in the [Template Repository](#) [► 1066].



• PLC task

The cycle time of the PLC task is set to 45 ms and should not be set smaller for performance reasons.



Project-specific programming

• Control

Once the project can be compiled without errors, the preparation is complete.

• Import templates

Project-specific programming begins with the integration of templates from the [Template Repository](#) [► 1066].

Now start with the project-specific programming.

6.1.4 Templates

6.1.4.1 DUTs

6.1.4.1.1 Enumerations

6.1.4.1.1.1 Building

6.1.4.1.1.1.1 *E_BA_BuildingMode*

The enumeration describes the current state of the building depending on the daily schedule.

```
TYPE E_BA_BuildingMode :
(
    eDefault      := 1,
    eNightWatch   := 2,
    eCleaning     := 3
) UDINT;
END_TYPE
```

Name	Description
eDefault	Standard (day) operation.
eNightWatch	Night watchman tour
eCleaning	Building cleaning

6.1.4.1.1.2 Plant

6.1.4.1.1.2.1 AC

6.1.4.1.1.2.1.1 *E_AC_PlantStep01*

Plant steps in the start program of an air conditioning system

```
TYPE E_AC_PlantStep01 :
(
    eOff           := 1,
    ePreRinse      := 2,
    eDamper        := 3,
    eFanSupplyAir  := 4,
    eFanExtractAir := 5,
    eEnablingTemperatureControl := 6,
    eEnablingTemperatureSequenceLink := 7,
    eEnablingHumidityControl := 8,
    eEnablingHumiditySequenceLink := 9,
    eEnablingLimitMonitoring := 10,
    eOn            := 11
) UDINT;
END_TYPE
```

Name	Description
eOff	Off
ePreRinse	Pre-flushing
eDamper	Damper
eFanSupplyAir	Supply air fan
eFanExtractAir	Exhaust air fan
eEnablingTemperatureControl	Enable temperature control.
eEnablingTemperatureSequenceLink	Enable temperature sequence link.
eEnablingHumidityControl	Enable humidity control
eEnablingHumiditySequenceLink	Enable humidity sequence link.
eEnablingLimitMonitoring	Enable limit value monitoring
eOn	On

6.1.4.1.1.2.1.2 E_AC_SeqNumber_H

Sequence number controller where the order of the sequence numbers are taken into account.

```

TYPE E_AC_SeqNumber_H :
(
  eHumidifier      := 1,
  eDehumidifier    := 2,
  eOff             := 3
) UDINT;
END_TYPE

```

Name	Description
eHumidifier	Sequence number humidifier.
eDehumidifier	Sequence number dehumidifier.
eOff	No sequence controller active.

6.1.4.1.1.2.1.3 E_AC_SeqNumber_T

Sequence number controller where the order of the sequence numbers are taken into account.

```

TYPE E_AC_SeqNumber_T :
(
  eReHeater        := 1,
  ePreHeater       := 2,
  eMixedAir        := 3,
  eEnergyRecovery  := 4,
  eCooler          := 5,
  eOff             := 6
) UDINT;
END_TYPE

```

Name	Description
eReHeater	Sequence number reheater.
ePreHeater	Sequence number preheater.
eMixedAir	Sequence number mixed air.
eEnergyRecovery	Sequence number energy recovery.
eCooler	Sequence number cooler.
eOff	No sequence controller active.

6.1.4.1.1.2.1.4 E_BA_AC_OpMod01

```

TYPE E_BA_AC_OpMod01 :
(
  Invalid          := 0,

```

```

eOff          := 1,
eOn           := 2,
eEmergency    := 3,
eFrost        := 4,
eSmokeExtractionProgram := 5,
eSmokeExtractionSupplyAir := 6,
eSmokeExtractionExhaustAir := 7,
eFire         := 8,
eNightCooling := 9,
eCoolDownProtection := 10,
eOverHeatingProtection := 11,
eAlarm        := 12,
eForcedVentilation := 13,
eCentralSwitchOff := 14
) UDINT;
END_TYPE

```

Name	Description
eOff	Off
eOn	On
eEmergency	Emergency
eFrost	Frost
eSmokeExtractionProgram	Smoke extraction program
eSmokeExtractionSupplyAir	Smoke extraction supply air
eSmokeExtractionExhaustAir	Smoke extraction exhaust air
eFire	Fire alarm
eNightCooling	Night cooling
eCoolDownProtection	Support operation, cooling protection
eOverHeatingProtection	Overheating protection
eAlarm	Fault
eForcedVentilation	Forced ventilation
eCentralSwitchOff	Central shutdown

6.1.4.1.1.2.1.5 E_BA_AC_OpMod02

```

TYPE E_BA_AC_OpMod01 :
(
  Invalid          := 0,
  eOff             := 1,
  eStep1           := 2,
  eStep2           := 3,
  eStep3           := 4,
  eEmergency        := 5,
  eFrost           := 6,
  eSmokeExtractionProgram := 7,
  eSmokeExtractionSupplyAir := 8,
  eSmokeExtractionExhaustAir := 9,
  eFire            := 10,
  eNightCooling    := 11,
  eCoolDownProtection := 12,
  eOverHeatingProtection := 13,
  eAlarm           := 14,
  eForcedVentilation := 15,
  eCentralSwitchOff := 16
) UDINT;
END_TYPE

```


Name	Description
eOff	Off
eStep1	Zone 1
eStep2	Zone 2
eStep3	Zone 3
eEmergency	Emergency
eFrost	Frost
eSmokeExtractionProgramm	Smoke extraction program
eSmokeExtractionSupplyAir	Smoke extraction supply air
eSmokeExtractionExhaustAir	Smoke extraction exhaust air
eFire	Fire alarm
eNightCooling	Night cooling
eCoolDownProtection	Support operation, cooling protection
eOverHeatingProtection	Overheating protection
eAlarm	Fault
eForcedVentilation	Forced ventilation
eCentralSwitchOff	Central shutdown

6.1.4.1.1.2.1.6 E_BA_AC_SelSpErc

```

TYPE E_BA_AC_SelSpErc :
(
  eAction          := 1, //E_BA_Action.eDirect = SpCol, E_BA_Action.eReverse = SpHtg,
  eAverageHtgCol   := 2
) UDINT;
END_TYPE

```

6.1.4.1.1.3 Process

6.1.4.1.1.3.1 E_BA_Process

```

TYPE E_BA_Process :
(
  Invalid           := 0,
  eOff              := 1,
  ePreRinse         := 2,
  eDriveDamper      := 3,
  eFanSupplyAir     := 4,
  eFanExtractAir    := 5,
  eEnablingTemperatureControl := 6,
  eEnablingHumidityControl   := 7,
  eEnablingLimitMonitoring   := 8
) UDINT;
END_TYPE

```

6.1.4.1.1.4 System

6.1.4.1.1.4.1 E_BA_UPSType

Selection of the appropriate emergency power supply.

```

TYPE E_BA_UPSType :
(
  eNone            := 0,

  eCX50x0,
  eCX51x0,
  eCX81xx,
  eCX9020_U900,
  eCB3011
);
END_TYPE

```

6.1.4.1.1.5 Universal

6.1.4.1.1.5.1 *E_BA_3Point*

```
TYPE E_BA_3Point :
(
  eOff      := 1,
  eOpen     := 2,
  eClose    := 3
) UDINT;
END_TYPE
```

6.1.4.1.1.5.2 *E_BA_3PointA*

```
TYPE E_BA_3PointA :
(
  eAutomatic := 1,
  eOff       := 2,
  eOpen      := 3,
  eClose     := 4
) UDINT;
END_TYPE
```

6.1.4.1.1.5.3 *E_BA_3Step*

```
TYPE E_BA_3Step :
(
  eOff      := 1,
  eStep1    := 2,
  eStep2    := 3,
  eStep3    := 4
) UDINT;
END_TYPE
```

6.1.4.1.1.5.4 *E_BA_3StepA*

```
TYPE E_BA_3StepA :
(
  eAutomatic := 1,
  eOff       := 2,
  eStep1     := 3,
  eStep2     := 4,
  eStep3     := 5
) UDINT;
END_TYPE
```

6.1.4.1.1.5.5 *E_BA_Conversion_kFactor*

The enumeration shows the unit of the conversation factor.

```
{attribute 'qualified_only'}
TYPE E_BA_Conversion_kFactor :
(
  e_Pa      := 0,
  e_hPa     := 1,
  e_mbar    := 2,
  e_mm_HG   := 3,
  e_in_HG   := 4,
  e_mm_WS   := 5,
  e_psi     := 6,
  e_inches_H2O := 7
) UDINT;
END_TYPE
```

6.1.4.1.1.5.6 *E_BA_EnergyLvl*

```
TYPE E_BA_EnergyLvl :
(
  eProtection := 1,
  eEconomy    := 2,
  eComfort    := 3
) UDINT;
END_TYPE
```

6.1.4.1.1.5.7 *E_BA_EnergyLvlA*

```
TYPE E_BA_EnergyLvlA :
(
  eAutomatic      := 1,
  eProtection     := 2,
  eEconomy        := 3,
  eComfort        := 4
) UDINT;
END_TYPE
```

6.1.4.1.1.5.8 *E_BA_EnergyLvlEx*

```
TYPE E_BA_EnergyLvlEx :
(
  eProtection     := 1,
  eEconomy        := 2,
  ePreComfort     := 3,
  eComfort        := 4
) UDINT;
END_TYPE
```

6.1.4.1.1.5.9 *E_BA_EnergyLvlExA*

```
TYPE E_BA_EnergyLvlExA :
(
  eAutomatic      := 1,
  eProtection     := 2,
  eEconomy        := 3,
  ePreComfort     := 4,
  eComfort        := 5
) UDINT;
END_TYPE
```

6.1.4.1.1.5.10 *E_BA_OnOff*

The enumeration shows the operation modes On and Off.

```
TYPE E_BA_OnOff :
(
  eOff            := 1,
  eOn             := 2
) UDINT;
END_TYPE
```

Name	Description
eOff	Off
eOn	On

6.1.4.1.1.5.11 *E_BA_OnOffA*

```
TYPE E_BA_OnOffA :
(
  eAutomatic      := 1,
  eOff            := 2,
  eOn             := 3
) UDINT;
END_TYPE
```

6.1.4.1.1.5.12 *E_BA_MdltCmd*

The enumeration shows the modulation commands Off, On, Minimum and Maximum.

```
{attribute'qualified_only'}
TYPE E_BA_MdltCmd:
(
  eOff            := 1,
  eOn             := 2,
  eMin           := 3,
  eMax           := 4
) UDINT;
END_TYPE
```

6.1.4.1.2 Types

6.1.4.1.2.1 Facade

6.1.4.1.2.1.1 *ST_BA_Facade*

This structure serves as a transmit telegram, for the transmission of facade-specific data to other controllers.

```

TYPE ST_BA_Facade :
STRUCT
    stSunBld          : ST_BA_SunBld;
    fSunPrtc_Position : REAL;
    fSunPrtc_Angle    : REAL;
    bThAuto_Release   : BOOL;
    bTwilGtAuto_Release : BOOL;
    bSunPrtc_State     : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
stSunBld	ST_BA_SunBld ▶ 217	Resulting blind telegram of the facade, related to the highest priority. This may include the following criteria: • Burglary, fire or ice • Storm shelter • Maintenance • Thermal automatic facade • Facade twilight automatic • Facade parking position
fSunPrtc_Position	REAL	Current sun protection position [%].
fSunPrtc_Angle	REAL	Current slat angle [°].
bThAuto_Release	BOOL	Enabling the thermal automatic for the zone / group control.
bTwilGtAuto_Release	BOOL	Enabling the twilight automatic for the zone / group control.
bSunPrtc_State	BOOL	The sun protection is activated when it exceeds a certain switch-on value for a certain switch-on time. Conversely, it is switched off if it falls below a certain switch-off value over a certain switch-off time.

6.1.4.1.2.2 Weather station

6.1.4.1.2.2.1 *ST_BA_WeatherStation*

This structure serves as a standard transmit telegram for the data of a weather station.

```

TYPE ST_BA_BuildingAlarms :
STRUCT
    bDisturb          : BOOL;
    fLatitude          : REAL;
    fLongitude         : REAL;
    fSunAzimuth        : REAL;
    fSunElevation      : REAL;
    bRain              : BOOL;
    stDateTime         : TIMESTRUCT;
    fOutsideTemperature : REAL;
    fDewPointTemperature : REAL;
    fPressureAbs       : REAL;
    fPressureRel       : REAL;
    fHumidityAbs       : REAL;
    fHumidityRel       : REAL;
    fBrightnessNorth   : REAL;

```

```

    fBrightnessEast      : REAL;
    fBrightnessSouth     : REAL;
    fBrightnessWest      : REAL;
    fDawn                : REAL;
    fGlobalRadiation     : REAL;
    fWindDirection       : REAL;
    fWindSpeed           : REAL;
END_STRUCT
END_TYPE

```

Name	Type	Description
bDisturb	BOOL	The weather station reports a malfunction.
fLatitude	REAL	Latitude of the installation site [°]
fLongitude	REAL	Longitude of the installation site [°]
fSunAzimuth	REAL	Current position of the sun [°]
fSunElevation	REAL	Current sun elevation [°]
bRain	BOOL	Rain sensor
stDateTime	<u>TIMESTRUCT</u>	Date / Time
fOutsideTemperature	REAL	Temperature [°C]
fDewPointTemperature	REAL	Dew point temperature [°C]
fPressureAbs	REAL	Absolute air pressure [hPa]
fPressureRel	REAL	Relative air pressure [hPa]
fHumidityAbs	REAL	Absolute humidity [g/m³]
fHumidityRel	REAL	Relative humidity [%rF]
fBrightnessNorth	REAL	Light sensor north [kLux]
fBrightnessEast	REAL	Light sensor east [kLux]
fBrightnessSouth	REAL	Light sensor south [kLux]
fBrightnessWest	REAL	Light sensor west [kLux]
fDawn	REAL	Dawn [Lux]
fGlobalRadiation	REAL	Global radiation [W/m²]
fWindDirection	REAL	Wind direction [°]
fWindSpeed	REAL	Wind speed [m/s]

6.1.4.1.2.2.2 ST_BA_WSC11Data

This structure is used to transfer the data of a Thies WSC11 weather station.

```

TYPE ST_BA_WSC11Data :
STRUCT
    stStatus          : ST_BA_WSC11Status;
    fCaseTemperature  : REAL;
    fLatitude         : REAL;
    fLongitude        : REAL;
    fSunAzimuth       : REAL;
    fSunElevation     : REAL;
    bRain             : BOOL;
    stDateTime        : TIMESTRUCT;
    sTimeFormat       : STRING(7);
    fOutsideTemperature : REAL;
    fDewPointTemperature : REAL;
    fPressureAbs      : REAL;
    fPressureRel      : REAL;
    fHumidityAbs      : REAL;
    fHumidityRel      : REAL;
    fBrightnessNorth  : REAL;
    fBrightnessEast   : REAL;
    fBrightnessSouth  : REAL;
    fBrightnessWest   : REAL;
    fDawn             : REAL;
    fGlobalRadiation  : REAL;
    fWindDirection    : REAL;

```

```

    fWindSpeed          : REAL;
END_STRUCT
END_TYPE

```

Name	Type	Description
stStatus	ST_BA_WSC11Status [► 617]	Status variable of the WSC11
fCaseTemperature	REAL	Housing internal temperature of the WSC11 [°C]
fLatitude	REAL	Latitude of the installation site [°]
fLongitude	REAL	Longitude of the installation site [°]
fSunAzimuth	REAL	Current position of the sun [°]
fSunElevation	REAL	Current sun elevation [°]
bRain	BOOL	Rain sensor
stDateTime	TIMESTRUCT	Date / Time
sTimeFormat	STRING(7)	Specification of the time format
fOutsideTemperature	REAL	Temperature [°C]
fDewPointTemperature	REAL	Dew point temperature [°C]
fPressureAbs	REAL	Absolute air pressure [hPa]
fPressureRel	REAL	Relative air pressure [hPa]
fHumidityAbs	REAL	Absolute humidity [g/m³]
fHumidityRel	REAL	Relative humidity [%rF]
fBrightnessNorth	REAL	Light sensor north [kLux]
fBrightnessEast	REAL	Light sensor east [kLux]
fBrightnessSouth	REAL	Light sensor south [kLux]
fBrightnessWest	REAL	Light sensor west [kLux]
fDawn	REAL	Dawn [Lux]
fGlobalRadiation	REAL	Global radiation [W/m²]
fWindDirection	REAL	Wind direction [°]
fWindSpeed	REAL	Wind speed [m/s]

6.1.4.1.2.2.3 ST_BA_WSC11Status

This structure is used to transfer the status of a Thies WSC11.

```

TYPE ST_BA_WSC11Status :
STRUCT
    bDewProtection          : BOOL;
    bSensorDryingPeriod     : BOOL;
    bInvalidRMC_Telegramm   : BOOL;
    bInvalidGPS_Time        : BOOL;
    bInvalidDataAD_Converter : BOOL;
    bInvalidDataPressureSensor : BOOL;
    bInvalidDataBrightnessSensorNorth : BOOL;
    bInvalidDataBrightnessSensorEast : BOOL;
    bInvalidDataBrightnessSensorSouth : BOOL;
    bInvalidDataBrightnessSensorWest : BOOL;
    bInvalidDataTwilightSensor : BOOL;
    bInvalidDataSolarRadiationSensor : BOOL;
    bInvalidDataOutsideTemperatureSensor : BOOL;
    bInvalidDataRainSensor : BOOL;
    bInvalidDataWindSpeedSensor : BOOL;
    bInvalidDataWindDirectionSensor : BOOL;
    bInvalidDataHumiditySensor : BOOL;
    bLatestRestartByWatchdogReset : BOOL;
    bInvalidEEPROM_Parameters : BOOL;
    bDefaultEEPROM_Parameters : BOOL;
    bLatestRestartWithNewFirmware : BOOL;
END_STRUCT
END_TYPE

```

The following descriptions refer to a TRUE at the respective parameter.

Name	Type	Description
bDewProtection	BOOL	Rain sensor: dew protection active.
bSensorDryingPeriod	BOOL	Rain sensor: drying phase of the sensor surface.
bInvalidRMC_Telegram	BOOL	GPS data: no valid RMC telegram received.
bInvalidGPS_Time	BOOL	RTC data from GPS receiver: time from GPS receiver are invalid.
bInvalidDataAD_Converter	BOOL	Values from the analog-to-digital converter are invalid.
bInvalidDataPressureSensor	BOOL	Air pressure: measured value from pressure sensor is invalid.
bInvalidDataBrightnessSensorNorth	BOOL	Measured value from light sensor north is invalid.
bInvalidDataBrightnessSensorEast	BOOL	Measured value from light sensor east is invalid.
bInvalidDataBrightnessSensorSouth	BOOL	Measured value from light sensor south is invalid.
bInvalidDataBrightnessSensorWest	BOOL	Measured value from light sensor west is invalid.
bInvalidDataTwilightSensor	BOOL	Measured value of twilight is invalid.
bInvalidDataSolarRadiationSensor	BOOL	Measured value from global radiation sensor is invalid.
bInvalidDataOutsideTemperatureSensor	BOOL	Measured value from air temperature sensor is invalid.
bInvalidDataRainSensor	BOOL	Measured value from rain sensor is invalid.
bInvalidDataWindSpeedSensor	BOOL	Measured value from wind speed sensor is invalid.
bInvalidDataWindDirectionSensor	BOOL	Measured value from wind direction sensor is invalid.
bInvalidDataHumiditySensor	BOOL	Measured values from humidity sensor are invalid (relative humidity, absolute humidity, dew point temperature).
bLatestRestartByWatchdogReset	BOOL	Last restart by watchdog reset.
bInvalidEEPROM_Parameters	BOOL	Internal EEPROM parameters are invalid.
bDefaultEEPROM_Parameters	BOOL	Internal EEPROM parameters contain the default values.
bLatestRestartWithNewFirmware	BOOL	Last restart was with new firmware.

6.1.4.1.2.3 Buildings

6.1.4.1.2.3.1 ST_BA_BuildingAlarms

This structure serves as a transmit telegram to transmit building-specific alarms to other controllers.

```

TYPE ST_BA_BuildingAlarms :
STRUCT
    bBurglary           : BOOL;
    bFireAlert          : BOOL;
    bCentralOff         : BOOL;
    bForcedVentilation  : BOOL;

```

```

    bSmokeExtraction      : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
bBurglary	BOOL	Burglar alarm
bFireAlert	BOOL	Fire alarm
bCentralOff	BOOL	Central off - command
bForcedVentilation	BOOL	Forced ventilation
bSmokeExtraction	BOOL	Smoke extraction

6.1.4.1.2.3.2 ST_BA_BuildingSunblind

This structure is used as a transmit telegram to transmit building-specific alarms, releases and information concerning blind control to other controllers.

```

TYPE ST_BA_BuildingSunblind :
STRUCT
    stSunBld                : ST_BA_SunBld;
    bGlobalThAuto_Release   : BOOL;
    bGlobalTwilgtAuto_Release : BOOL;
    bGlobalResetManMode     : BOOL;
    nSunPrtc_PositionInterval : UDINT;
END_STRUCT
END_TYPE

```

Name	Type	Description
stSunBld	ST_BA_SunBld [► 217]	Resulting positioning telegram from the building-wide alarms: fire, burglary or icing.
bGlobalThAuto_Release	BOOL	Global release criterion for the thermal automatic.
bGlobalTwilgtAuto_Release	BOOL	Global release criterion for the twilight automatic.
bGlobalResetManMode	BOOL	Global reset of the manual functions.
nSunPrtc_PositionInterval	UDINT	Positioning interval of the lamella setpoint tracing when using the automatic sun protection.

6.1.4.1.2.4 General

6.1.4.1.2.4.1 ST_BA_GeneralSettings

```

TYPE ST_BA_GeneralSettings :
STRUCT
    fFrostProtectionSetpoint : REAL;
    fTWth                    : REAL;
    fTWthLowLimit            : REAL;
    bTWthLowLimit            : BOOL;
    fTWthDamped              : REAL;
    bTWthDampedLowLimit      : BOOL;
END_STRUCT
END_TYPE

```


Name	Type	Description
fFrostProtectionSetpoint	REAL	Frost protection setpoint, e.g. for heating circuits in protection mode.
fTWth	REAL	Current value of the outside temperature.
fTWthLowLimit	REAL	Lower limit value of the outside temperature. Below this value, all frost protection functions in HVAC systems are activated.
bTWthLowLimit	BOOL	The variable is TRUE if the outside temperature is below the value of <i>fTWthLowCrit</i> .
fTWthDamped	REAL	Current value of the damped outside temperature.
bTWthDampedLowLimit	BOOL	The variable is TRUE if the damped outside temperature is below the value of <i>fTWthLowCrit</i> .

6.1.4.2 POU's

This chapter describes the function blocks contained in the template repository.

6.1.4.2.1 Technical system

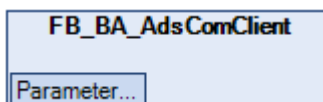
Templates from the individual technical systems.

6.1.4.2.1.1 Building automation

Templates for cross-system functions.

6.1.4.2.1.1.1 Communication

6.1.4.2.1.1.1.1 FB_BA_AdsComClient



The template reads global data, for example weather data and global parameters, from another controller in the GA network. It copies this data to the [GVL site \[► 818\]](#).



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

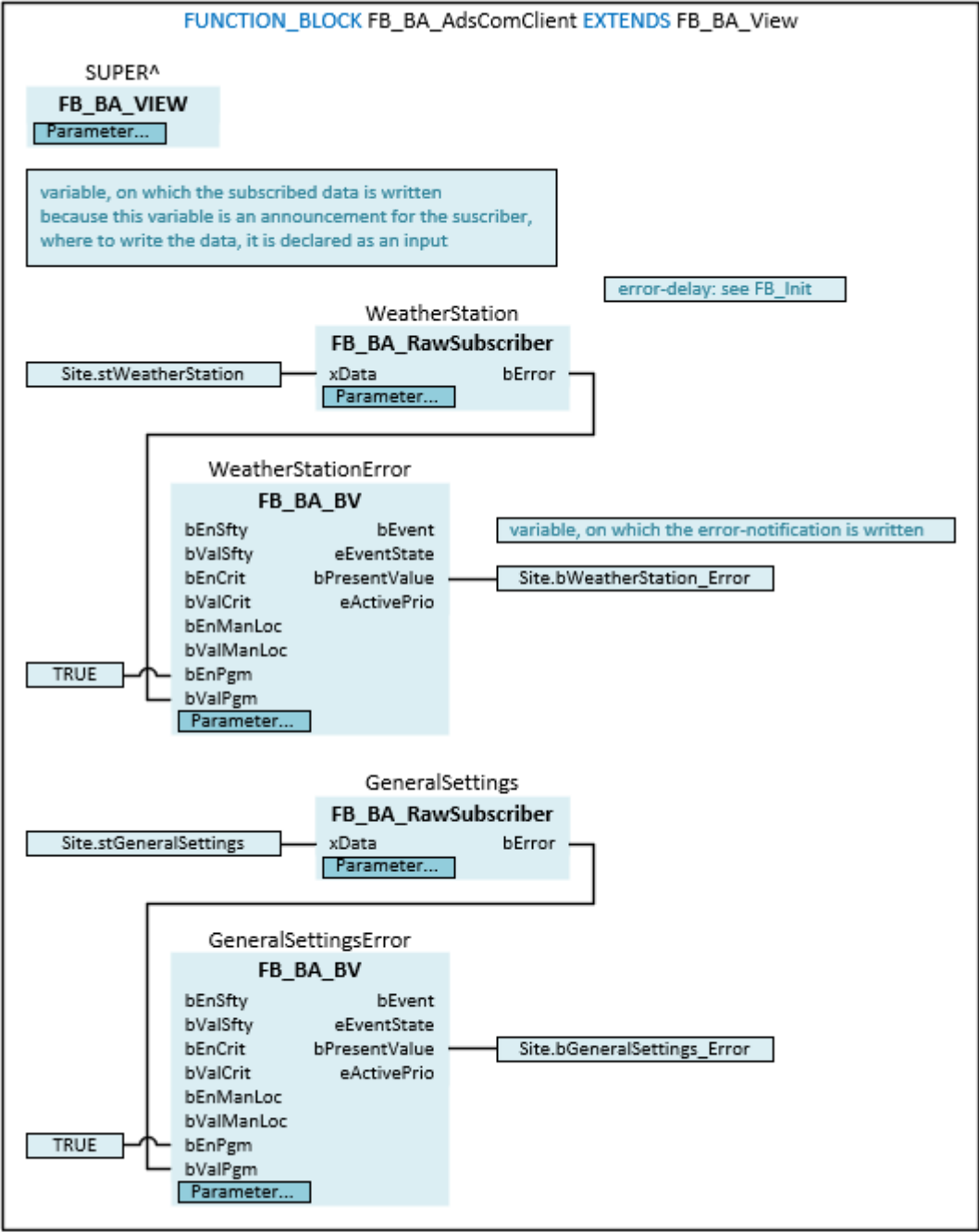
FB_BA_Base

FB_BA_BasePublisher

FB BA Object [► 195]

FB BA View [► 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_AdsComClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    WeatherStation      : FB_BA_RawSubscriber;
    WeatherStationError : FB_BA_BV;

    GeneralSettings     : FB_BA_RawSubscriber;
    GeneralSettingsError : FB_BA_BV;
END_VAR
```

🔧 Inputs CONSTANT

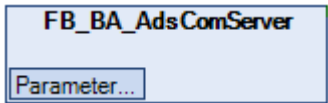
Name	Type	Description

WeatherStation	FB_BA_RawSubscriber	The subscriber <i>WeatherStation</i> accesses the TwinCAT network structure <i>WeatherStation</i> and stores the data into the created structure of the GVL <i>Site.stWeatherStation</i> .
WeatherStationError	FB_BA_BV [► 147]	Binary object indicating communication failure of the subscriber <i>WeatherStation</i> .
GeneralSettings	FB_BA_RawSubscriber	The subscriber <i>GeneralSettings</i> accesses the TwinCAT network structure <i>GeneralSettings</i> and stores the data in the created structure of the GVL <i>Site.stGeneralSettings</i> .
GeneralSettingsError	FB_BA_BV [► 147]	Binary object indicating communication failure of the subscriber <i>GeneralSettings</i> .

Requirements

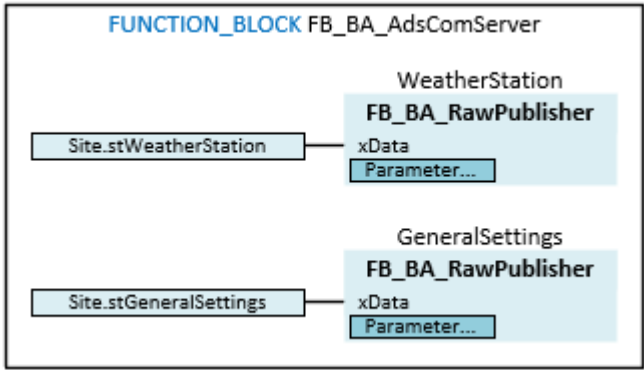
Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.1.2 FB_BA_AdsComServer



The template has the task to read data and parameters from the [GVL site \[► 818\]](#) and to provide them via FB_BA_RawSubscriber within the GA network.

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_AdsComServer
VAR_INPUT CONSTANT
    WeatherStation      : FB_BA_RawPublisher;
    GeneralSettings     : FB_BA_RawPublisher;
END_VAR
```

🚩 Inputs CONSTANT

Name	Type	Description
WeatherStation	FB_BA_RawSubscriber	The publisher <i>WeatherStation</i> publishes the TwinCAT network structure of the weather station data in the GVL <i>Site.stWeatherStation</i> .
GeneralSettings	FB_BA_RawSubscriber	The publisher <i>GeneralSettings</i> publishes the TwinCAT network structure of the weather parameters in the GVL <i>Site.stGeneralSettings</i> .

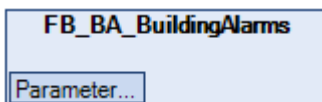
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2 GlobalEventsAndSettings

Global events and settings.

6.1.4.2.1.1.2.1 FB_BA_BuildingAlarms



This template summarizes building-specific safety criteria and makes them available in a globally declared structure *stBuildingAlarms* (see global variable list [Site \[► 818\]](#)).

The criteria in detail are:

1. Fire alarm (*FireAlert*): this alarm is read in directly via a binary input object.
2. Burglar alarm (*Bgly/Burglary*): this alarm is read in directly via a binary input object.
3. Central switch-off (*CentSwiOff*), actually no alarm. This input object makes it possible, for example, to reset manual flags building-wide or to switch off lights at this central point.
4. Forced ventilation (*ForcedVenilation*): in this template on reserve, to trigger forced ventilation.
5. Smoke extraction (*SmokeExtraction*): in this template on reserve, to trigger a smoke extraction operation.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingAlarms EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    FireAlert      : FB_BA_BI_IO;
    Bgly           : FB_BA_BI_IO;
    CentSwiOff     : FB_BA_BV_Op;
END_VAR
```

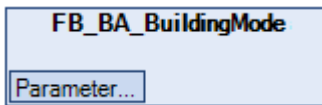
🔧 Inputs CONSTANT

Name	Type	Description
FireAlert	FB_BA_BI_IO [► 139]	Binary input object "Fire alert".
Bgly	FB_BA_BI_IO [► 139]	Binary input object "Burglary".
CentSwiOff	FB_BA_BV_Op [► 150]	Binary input object "Central switch off".

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.2 FB_BA_BuildingMode



Selection of **building mode** based on a schedule (*Sched*) and manual intervention (*OpModMan*).

The manual intervention has the following modes (*OpModMan*):

Value	Meaning
1	Automatic mode
2	manual selection Default
3	manual selection Nightwatch
4	manual selection Cleaning

If **Automatic mode** is selected, the function block *DeMuxManMod* sets the output *bQ01*. Since this output is not connected to the *PrioSwi*, the manual mode on the *PrioSwi* is disabled and the schedule *Sched* is active. This can assume the following values:

Value	Meaning
1	Default
2	Nightwatch
3	Cleaning

The currently selected building mode is then displayed via the function block *OpModPr* and made available via Publisher.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_View \[► 173\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingMode EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    OpModMan      : FB_BA_MV_Op;
```

```

OpModPr      : FB_BA_MV_Op;
Sched        : FB_BA_SchedM;
END_VAR
VAR
  DeMuxManMod : FB_BA_DMUX_B04;
  DeMuxSched  : FB_BA_DMUX_B04;
  PrioSwi     : FB_BA_PrioSwi_UDI08;
END_VAR

```

Inputs CONSTANT

Name	Type	Description
OpModMan	FB_BA_MV_Op [► 171]	The Multistate-Value object represents an operating modes switch with the operating modes Auto, Manual-Off and Manual-On.
OpModPr	FB_BA_MV_Op [► 171]	The Multistate-Value object indicates the state of the currently valid plant operation mode.
Sched	FB_BA_SchedM [► 159]	Schedule object (automatic) for the "BuildingEnergyLevel".

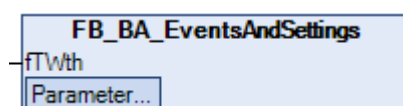
Variables

Name	Type	Description
DeMuxManMode	FB_BA_DMUX_B04 [► 550]	Conversion of the multistate value of the manual selection to a binary output.
DeMuxSched	FB_BA_DMUX_B04 [► 361]	Conversion of the multistate value of the schedule to a binary output.
PrioSwi	FB_BA_PrioSwi_UDI08 [► 369]	Prioritizing reconversion of the states to a resulting multistate or enumeration value for the building energy level.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.3 FB_BA_EventsAndSettings



The template represents the plant level.

It is used for the preparation and distribution of global data, which is required system- or building-wide for the control and regulation of the system and room automation.

The function block [FB_BA_BuildingAlarms \[► 623\]](#) is used to record global events that are required in a building or property at all automation stations of the system.

The function block [FB_BA_WeatherParameter](#) is used for the global provision of weather-specific parameters and setpoints.

The function block [FB_BA_BuildingMode \[► 624\]](#) is used for the global provision of building operating modes.

The global building data is stored in the GVL [site \[► 818\]](#).



The initialization of the template takes place within the method [FB_Init](#).

Inheritance hierarchy

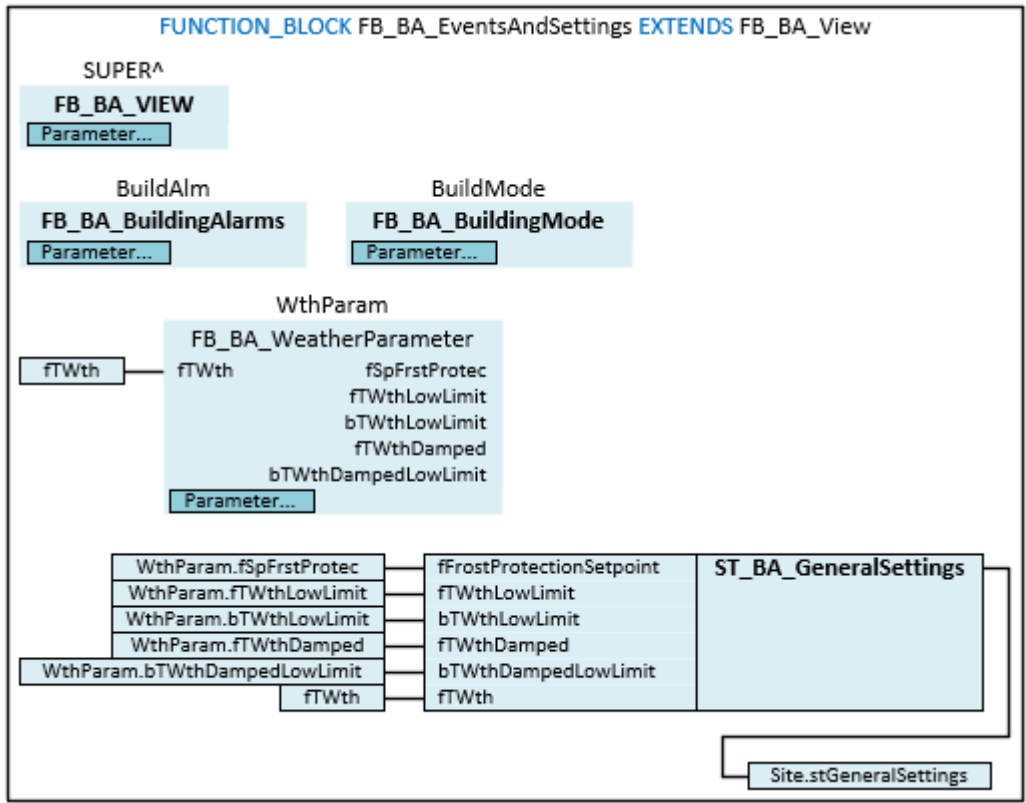
FB_BA_Base

 FB_BA_BasePublisher

 FB_BA_Object [► 195]

 FB_BA_View [► 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_BuildingGlobal EXTENDS FB_BA_View
VAR_INPUT
    fTWth      : REAL;
END_VAR
VAR_INPUT CONSTANT
    BuildAlm   : FB_BA_BuildingAlarms;
    BuildMode  : FB_BA_BuildingMode;
    WthParam   : FB_BA_WeatherParameter;
END_VAR
```

🔧 Inputs

Name	Type	Description
fTWth	REAL	Input value of the outside temperature.

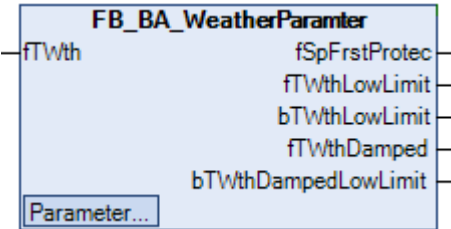
🔧 Inputs **CONSTANT**

Name	Type	Description
BuildAlm	FB_BA_BuildingAlarms [▶ 623]	Template for the building alarms.
BuildMode	FB_BA_BuildingMode [▶ 624]	Template for the building operation modes.
WthParam	FB_BA_WeatherParameter	Template for the evaluation of the outside temperature.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.4 FB_BA_WeatherParameter



The template calculates various values from the measured value of the outside temperature, which are required system-wide for the control and regulation of heating, ventilation and air conditioning systems.

The object *SpFrstProtec* is used to enter a frost protection setpoint. The frost protection setpoint is used system-wide in all HVAC templates with a water-side frost protection.

The function block *TWthLowLimitHys* calculates whether the outside temperature is below the critical value of *TWthLowLimitVal*. Depending on the value of the variable *bTWthLowLimit*, the frost protection function of the HVAC systems in the building is activated.

The weather-related enable of the heating systems is not dependent on the current, but on the damped outside temperature. The damping of the outside temperature is done with the function block *Average*. The object *TWthDamped* is used to display the value of the damped outside temperature. The function block *TWthDampedLowLimitHys* is used to check whether the damped outside temperature is below a value that will enable the heating systems in the building. The global weather-related enable is displayed with the object *FB_BA_BV_Op* [▶ 150] and written to the variable *bTWthDampedLowLimit* for further processing in other templates.

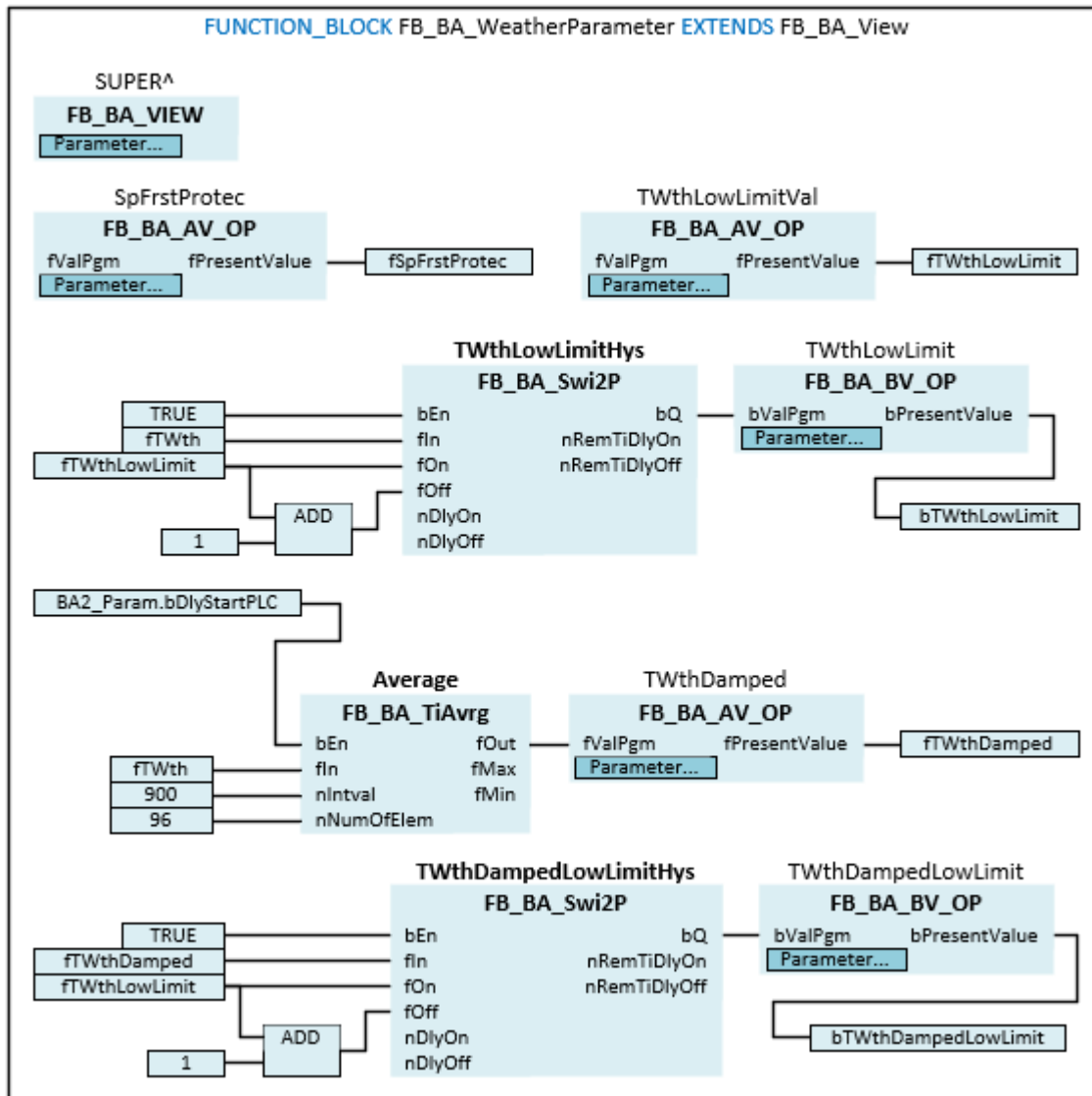


The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - FB_BA_Object [▶ 195]
 - FB_BA_View [▶ 173]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_Settings EXTENDS FB_BA_View
VAR_INPUT
    fTWth                : REAL;
END_VAR
VAR_OUTPUT
    fSpFrstProtec        : REAL;
    fTWthLowLimit        : REAL;
    bTWthLowLimit        : BOOL;
    fTWthDamped          : REAL;
    bTWthDampedLowLimit  : BOOL;
END_VAR
VAR_INPUT CONSTANT
    SpFrstProtec          : FB_BA_AV_Op;
    TWthLowLimitVal       : FB_BA_AV_Op;
    TWthLowLimit          : FB_BA_BV_Op;
    TWthDamped            : FB_BA_AV_Op;
    TWthDampedLowLimit    : FB_BA_BV_Op;
END_VAR
VAR
    TWthLowLimitHys       : FB_BA_Swi2P;
    Average               : FB_BA_TiAvrg;
    TWthDampedLowLimitHys : FB_BA_Swi2P;
END_VAR

```

Inputs

Name	Type	Description
fTWth	REAL	Input value of the outside temperature.

Outputs

Name	Type	Description
fSpFrstProtec	REAL	Frost protection setpoint, e.g. for heating circuits in protection mode.
fTWthLowLimit	REAL	Lower limit value of the outside temperature. Below this value, all frost protection functions in HVAC systems are activated.
bTWthLowLimit	BOOL	The output shows the operating message <i>Outside temperature lower limit</i> . The variable is TRUE if the outside temperature is below the value of <i>fTWthLowCrit</i> .
fTWthDamped	REAL	Current value of the damped outside temperature.
bTWthDampedLowLimit	BOOL	The output shows the operating message <i>Outside temperature damped lower limit</i> . The variable is TRUE if the damped outside temperature is below the value of <i>fTWthLowCrit</i> .

Inputs CONSTANT

Name	Type	Description
SpFrstProtec	FB_BA_AV_Op [► 136]	Analog value object for entering the frost protection setpoint, e.g. for heating circuits in protection mode.
TWthLowLimitVal	FB_BA_AV_Op [► 136]	Analog value object for entering the lower limit value of the outside temperature. Below this value, all frost protection functions in HVAC systems are activated.
TWthLowLimit	FB_BA_BV_Op [► 150]	Binary object displaying the <i>Outside temperature lower limit</i> operating message.
TWthDamped	FB_BA_AV_Op [► 136]	Analog value object for displaying the damped outside temperature. Below this value, all frost protection functions in HVAC systems are activated.
TWthDampedLowLimit	FB_BA_BV_Op [► 150]	Binary object displaying the <i>Outside temperature damped lower limit</i> operating message.

Variables

Name	Type	Description
TWthLowLimitHys	FB_BA_Swi2P [► 388]	Two-point switch which converts the outside temperature <i>fTWth</i> into a binary switching signal for <i>TWthLowLimit</i> .
Average	FB_BA_TiAvg [► 402]	The Average uses the outside temperature <i>fTWth</i> to determine the damped outside temperature <i>fTWthDamped</i> .
TWthDampedLowLimitHys	FB_BA_Swi2P [► 388]	Two-point switch which converts the damped outside temperature <i>fTWthDamped</i> into a binary switching signal for <i>TWthDampedLowLimit</i> .

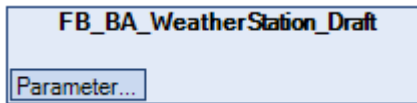
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.3 Weather station

Templates for the implementation of weather stations.

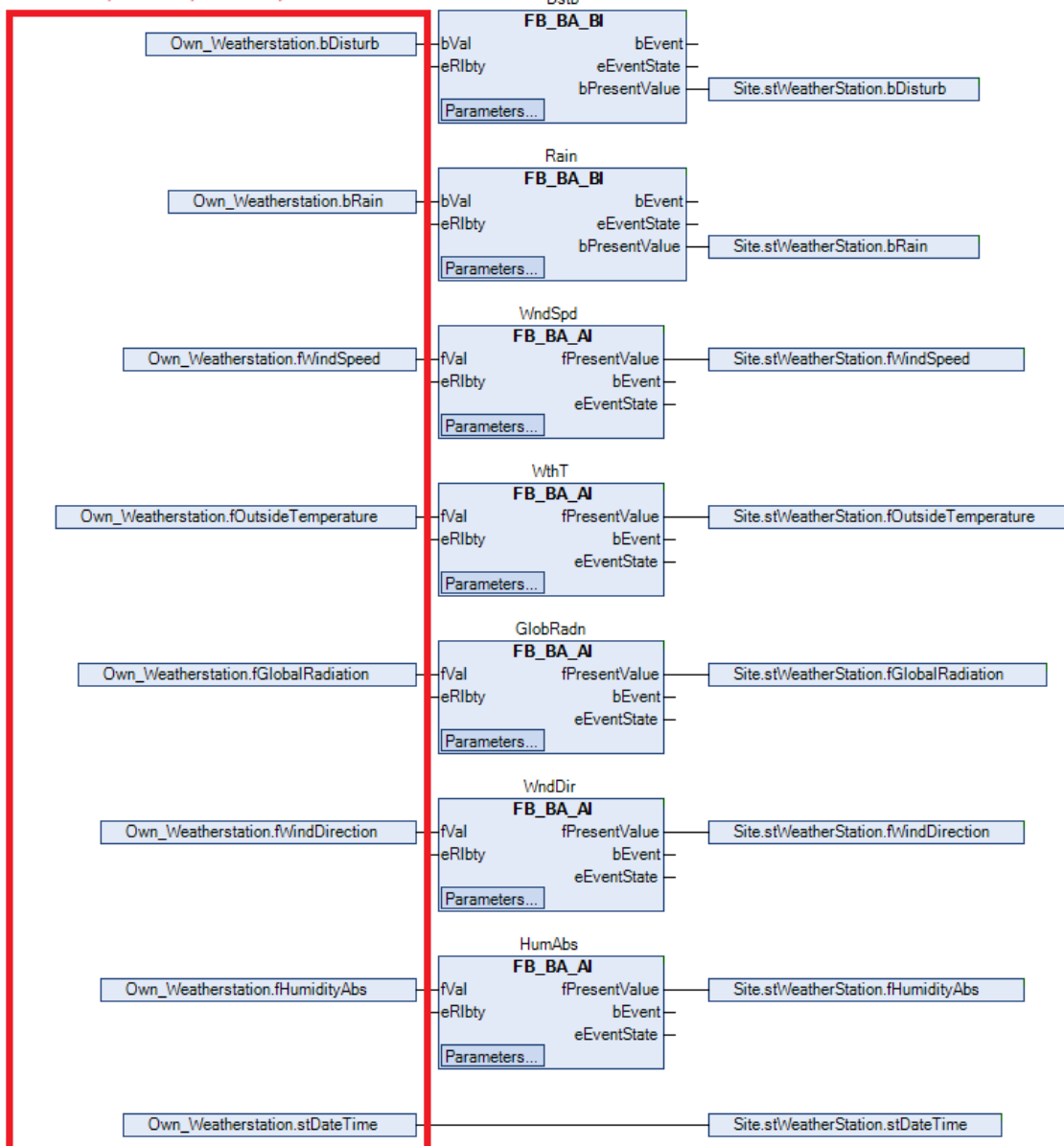
6.1.4.2.1.1.3.1 FB_BA_Weatherstation_Draft



This template represents a programming template for a weather station for which no prepared template is available. To the objects *Dstb ... SunElv* the values of the existing weather station are to be linked, if available, as well as the time. There is no object for the time of the type `TIMESTRUCT`, here the placeholder variable must be replaced. This is only for error-free translation of the base project.

The respective values *fPresentValue* are combined in a globally declared variable structure `stWeatherstation` (see [Site \[► 818\]](#)).

weather-data, available for the user
(in this example, not all possible objects are shown)





The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_View \[► 173\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_WeatherStation_Draft EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    Dstb          : FB_BA_BI;
    Rain          : FB_BA_BI;
    WthT          : FB_BA_AI;
    DewPtT        : FB_BA_AI;
    PrssAbs        : FB_BA_AI;
    PrssRel        : FB_BA_AI;
    HumAbs         : FB_BA_AI;
    HumRel         : FB_BA_AI;
    Brightness     : FB_BA_AI;
    Dawn           : FB_BA_AI;
    GlobRadn       : FB_BA_AI;
    WndDir         : FB_BA_AI;
    WndSpd         : FB_BA_AI;
    Latd           : FB_BA_AI;
    Lngt           : FB_BA_AI;
    SunAzm         : FB_BA_AI;
    SunElv         : FB_BA_AI;
END_VAR

VAR
    stDateTime_PLACEHOLDER : TIMESTRUCT;
END_VAR
```

🚩 Inputs CONSTANT

Name	Type	Description
Dstb	FB_BA_BI [► 138]	The weather station reports a malfunction.
Rain	FB_BA_BI [► 138]	Rain sensor.
WthT	FB_BA_AI [► 125]	Outside temperature [°C].
DewPtT	FB_BA_AI [► 125]	Dew point temperature [°C].
PrssAbs	FB_BA_AI [► 125]	Absolute air pressure [hPa].
PrssRel	FB_BA_AI [► 125]	Relative air pressure [hPa].
HumAbs	FB_BA_AI [► 125]	Absolute humidity [g/m³].
HumRel	FB_BA_AI [► 125]	Relative Absolute Humidity [g/m³].
Brightness_N	FB_BA_AI [► 125]	Directional light sensor north [Lux].
Brightness_S	FB_BA_AI [► 125]	Directional light sensor south [Lux].
Brightness_E	FB_BA_AI [► 125]	Directional light sensor east [Lux].
Brightness_W	FB_BA_AI [► 125]	Directional light sensor west [Lux].
Dawn	FB_BA_AI [► 125]	Dawn [Lux].
GlobRadn	FB_BA_AI [► 125]	Global radiation [W/m²].
WndDir	FB_BA_AI [► 125]	Wind direction [°].
WndSpd	FB_BA_AI [► 125]	Wind speed [m/s]
Latd	FB_BA_AI [► 125]	Latitude of the installation site [°].
Lngt	FB_BA_AI [► 125]	Longitude of the installation site [°].
SunAzm	FB_BA_AI [► 125]	Current position of the sun [°].
SunElv	FB_BA_AI [► 125]	Current sun elevation [°].

Variables

Name	Type	Description
stDateTime_PLACEHOLDER	TIMESTRUCT	Placeholder variable: instead of this variable, a suitable time structure can be linked, which provides the current time via the weather station.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.3.2 FB_BA_Weatherstation_Thies



This template prepares the data which are read from a Thies weather station via the function block [FB_BA_ThiesWSC11 \[► 816\]](#) and makes them available in a globally declared variable structure *stWeatherstation* (see global variable list [Site \[► 818\]](#)).

In addition, the hardware connection is established in this template via the variables *stRawDataIn* and *stRawDataOut*.

A detailed description of the integration of the weather station can be found [here \[► 799\]](#).



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Illustration

```
FUNCTION_BLOCK FB_BA_WeatherStation_Thies EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    SerialCommRst          : FB_BA_Bv_OP;
    WthStRst               : FB_BA_Bv_OP;
    SerialCommErr          : FB_BA_BV;
    Dstb                   : FB_BA_BV;
    Rain                   : FB_BA_BI;
    WthT                   : FB_BA_AI;
    DewPtT                 : FB_BA_AI;
    PrssAbs                : FB_BA_AI;
    PrssRel                : FB_BA_AI;
    HumAbs                 : FB_BA_AI;
    HumRel                 : FB_BA_AI;
    Brightness_N           : FB_BA_AI;
    Brightness_S           : FB_BA_AI;
    Brightness_E           : FB_BA_AI;
    Brightness_W           : FB_BA_AI;
    Dawn                   : FB_BA_AI;
    GlobRadn               : FB_BA_AI;
    WndDir                 : FB_BA_AI;
    WndSpd                 : FB_BA_AI;
    Latd                   : FB_BA_AI;
    Lngt                   : FB_BA_AI;
    SunAzm                 : FB_BA_AI;
    SunElv                 : FB_BA_AI;
END_VAR

VAR
    ThiesWSC11             : FB_BA_ThiesWSC11;
    bResetWeatherStation   : BOOL;
    bResetSerialCommunication : BOOL;
    DataConversion          : FB_BA_ThiesData;
    tonSerialCommErr        : TON;
    tonDstb                 : TON;

    stRawDataIn            AT %I* : KL6InData22B;
    stRawDataOut            AT %Q* : KL6OutData22B;
    fbSerialCtrl             : SerialLineControl;
    fbKL6Configuration      : KL6configuration;
    stTxBuff                : ComBuffer;
    stRxBuff                : ComBuffer;

    bSerialConfigError      : BOOL;
    nSerialConfigErrorID    : UDINT;
    bSerialCommError        : BOOL;
    nSerialCommErrorID      : UDINT;
END_VAR

VAR_INPUT CONSTANT PERSISTENT
{attribute 'parameterUnit':= 's'}
    nSerialCommErrDelay     : UDINT := 10;
{attribute 'parameterUnit':= 's'}
    nDstbDelay              : UDINT := 10;
END_VAR
```

 Inputs CONSTANT

Name	Type	Description
SerialCommRst	FB BA Bv OP [► 150]	Restart the serial communication if it is permanently faulty due to a configuration error or during operation.
WthStRst	FB BA Bv OP [► 150]	Restarting the weather station itself if it is permanently faulty. Here the PLC routine is restarted starting with the configuration and subsequent cyclic query. A hardware reset is not performed.
SerialCommErr	FB BA BV [► 147]	Message requiring acknowledgement: Serial communication or configuration permanently faulty.
Dstb	FB BA BV [► 147]	The weather station reports a malfunction.
Rain	FB BA BI [► 138]	Rain sensor.
WthT	FB BA AI [► 125]	Outside temperature [°C].
DewPtT	FB BA AI [► 125]	Dew point temperature [°C].
PrssAbs	FB BA AI [► 125]	Absolute air pressure [hPa].
PrssRel	FB BA AI [► 125]	Relative air pressure [hPa].
HumAbs	FB BA AI [► 125]	Absolute humidity [g/m³].
HumRel	FB BA AI [► 125]	Relative Absolute Humidity [g/m³].
Brightness_N	FB BA AI [► 125]	Directional light sensor north [Lux].
Brightness_S	FB BA AI [► 125]	Directional light sensor south [Lux].
Brightness_E	FB BA AI [► 125]	Directional light sensor east [Lux].
Brightness_W	FB BA AI [► 125]	Directional light sensor west [Lux].
Dawn	FB BA AI [► 125]	Dawn [Lux].
GlobRadn	FB BA AI [► 125]	Global radiation [W/m²].
WndDir	FB BA AI [► 125]	Wind direction [°].
WndSpd	FB BA AI [► 125]	Wind speed [m/s]
Latd	FB BA AI [► 125]	Latitude of the installation site [°].
Lngt	FB BA AI [► 125]	Longitude of the installation site [°].
SunAzm	FB BA AI [► 125]	Current position of the sun [°].
SunElv	FB BA AI [► 125]	Current sun elevation [°].

Variables

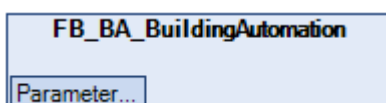
Name	Type	Description
ThiesWSC11	FB_BA_ThiesWSC11 [► 816]	Function block for unloading the data from the Thies WSC11
bResetWeatherStation	BOOL	Reset weather station.
bResetSerialCommunication	BOOL	Reset serial communication.
DataConversion	FB_BA_ThiesData [► 817]	Converts the Thies weather station data into project-specific data. Example brightness: conversion from kLux to Lux, because only this unit is supported by BACnet.
tonSerialCommErr	TON	Error delay communication error.
tonDstb	TON	Error delay weather station error
stRawDataIn	KL6InData22B	Input raw values from the serial terminal.
stRawDataOut	KL6InData22B	Output raw values to the serial terminal.
fbSerialCtrl	SerialLineControl	Communication block to the serial terminal. Runs in the fast task under the FastCycle method.
fbKL6Configuration	KL6configuration	Configuration block to the serial terminal. Runs in the fast task under the FastCycle method. This is used to set the serial parameters (baud rate, etc), but not the process image (see Thies weather station [► 799]).
stTxBuff	ComBuffer	Communication variable between the <i>fbSerialCtrl</i> of the fast task and the <i>ThiesWSC11</i> of the normal PLC task.
stRxBuff	ComBuffer	Communication variable between the <i>fbSerialCtrl</i> of the fast task and the <i>ThiesWSC11</i> of the normal PLC task.
bSerialConfigError	BOOL	Configuration error of the serial communication.
nSerialConfigErrorID	UDINT	Error number .
bSerialCommError	BOOL	Communication error of the serial communication.
nSerialCommErrorID	UDINT	Error number .

 Inputs CONSTANT PERSISTENT

Name	Type	Description
nSerialCommErrDelay	UDINT	Error delay communication error [s].
nDstbDelay	UDINT	Error delay weather station error [s].

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.4 *FB_BA_BuildingAutomation*

The template is a function block for calling all sub-templates that perform general tasks of the building automation trade.

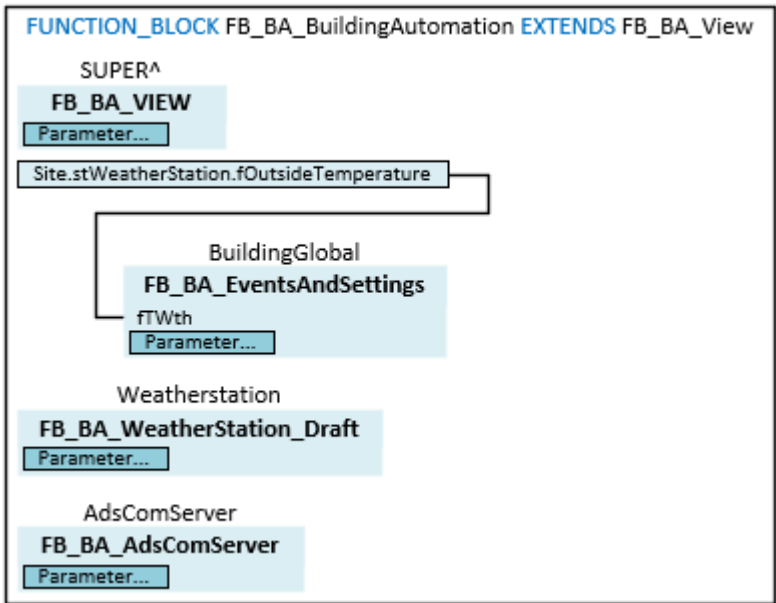


The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - FB BA Object [▶ 195]
 - FB BA View [▶ 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_BuildingAutomation EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    BuildingGlobal      : FB_BA_EventsAndSettings;
    Weatherstation      : FB_BA_WeatherStation_Draft;
    AdsComServer        : FB_BA_AdsComServer;
END_VAR
```

VAR_INPUT CONSTANT

Name	Type	Description
BuildingGlobal	FB BA EventsAndSettings [▶ 625]	The template includes weather station data, weather parameters, building modes, and building-specific safety criteria for a building automation project.
Weatherstation	FB BA WeatherStation_Draft [▶ 630]	The template is used to collect and display data from a weather station.
AdsComServer	FB BA AdsComServer [▶ 622]	The template accesses locally determined data and makes them available to a TwinCAT network. If the automation station to be programmed is not a data server within the GA network but a data client, the template FB BA AdsComClient [▶ 620] must be instantiated instead of the template FB BA AdsComServer [▶ 622] .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

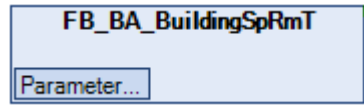
6.1.4.2.1.2 Room automation

Templates for creating room automation solutions.

6.1.4.2.1.2.1 HeatingCooling

Templates for heating and cooling.

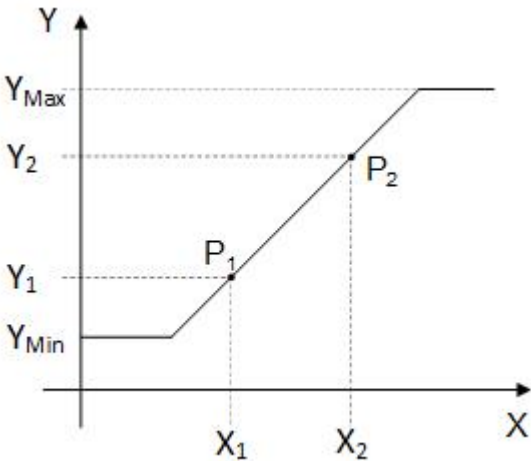
6.1.4.2.1.2.1.1 FB_BA_BuildingSpRmT



In this template the basic setpoints Protection Cooling ... Comfort Cooling and Protection Heating ... Comfort Heating are applied with correction values and made available as building-wide setpoints in a globally declared structure variable *stBuildingSpRmT* (see global variable list [Site \[► 818\]](#)).

The correction affects the Precomfort and Comfort values respectively, see [FB_BA_SpRmT \[► 243\]](#). These are:

- **central setpoint shift**
These two values, ShiftHtg and ShiftCol, are each added to the corresponding values for Precomfort and Comfort.
- **weather-compensated setpoint shift (summer or winter compensation)**
Using the outside temperature from the weather station data *stWeatherStation* (see global variable list [Site \[► 818\]](#)), correction values are calculated here once again, which are added to the values for Precomfort and Comfort.
The calculation is performed using the interpolation function blocks [FB_BA_Scale_02 \[► 728\]](#).



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)[FB_BA_View \[► 173\]](#)**Illustration**

```

FUNCTION BLOCK FB_BA_BuildingSpRmT EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    SumCpsn      : FB_BA_Scale_02;
    WinCpsn      : FB_BA_Scale_02;
    PrtcHtg      : FB_BA_AV_Op;
    EcoHtg       : FB_BA_AV_Op;
    PreCmfHtg    : FB_BA_AV_Op;
    CmfHtg       : FB_BA_AV_Op;
    PrtcCol      : FB_BA_AV_Op;
    EcoCol       : FB_BA_AV_Op;
    PreCmfCol    : FB_BA_AV_Op;
    CmfCol       : FB_BA_AV_Op;
    ShiftHtg     : FB_BA_AV_Op;
    ShiftCo      : FB_BA_AV_Op;
END_VAR

VAR
    SpRmT       : FB_BA_SpRmT;
END_VAR

```

 Inputs CONSTANT

Name	Type	Description
SumCpsn	FB_BA_Scale_02 [► 728]	
WinCpsn	FB_BA_Scale_02 [► 728]	
PrtcHtg	FB_BA_AV_Op [► 136]	
EcoHtg	FB_BA_AV_Op [► 136]	
PreCmfHtg	FB_BA_AV_Op [► 136]	
CmfHtg	FB_BA_AV_Op [► 136]	
PrtcCol	FB_BA_AV_Op [► 136]	
EcoCol	FB_BA_AV_Op [► 136]	
PreCmfCol	FB_BA_AV_Op [► 136]	
CmfCol	FB_BA_AV_Op [► 136]	
ShiftHtg	FB_BA_AV_Op [► 136]	
ShiftCo	FB_BA_AV_Op [► 136]	

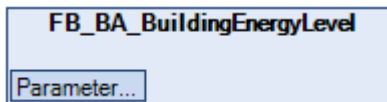
Variables

Name	Type	Description
SpRmT	FB_BA_SpRmT [► 243]	Function block for calculating the applied temperature setpoints.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.1.2 FB_BA_BuildingEnergyLevel



Selection of the **building energy level** based on a schedule (*Sched*) and a manual mode (*EnergLvlMan*).

The manual mode (*EnergLvlMan*) has the following values:

Value	Meaning
1	Automatic mode
2	manual selection Protection
3	manual selection Economy
4	manual selection Precomfort
5	manual selection Comfort

If **Automatic mode** is selected, the function block *DeMuxEnergLvl* sets the output *bQ01*. Since this output is not connected to the *PrioSwi*, the manual mode on the *PrioSwi* is disabled and the schedule *Sched* is active. This can assume the following values:

Value	Meaning
1	Protection
2	Economy
3	Precomfort
4	Comfort

The currently selected energy level is then displayed via the function block *EnergLvlPr* and made available via Publisher.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_View \[► 173\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingEnergyLevel EXTENDS FB_BA_View

VAR_INPUT CONSTANT
    EnergLvlMan      : FB_BA_MV_Op;
    EnergLvlPr       : FB_BA_MV_Op;
    Sched            : FB_BA_SchedM;
END_VAR

VAR
    DeMuxEnergLvl    : FB_BA_DMUX_B08;
    DeMuxSched        : FB_BA_DMUX_B04;
    PrioSwi           : FB_BA_PrioSwi_UDI08;
END_VAR
```

🚩 Inputs CONSTANT

Name	Type	Description
EnergLvlMan	FB_BA_MV_Op [▶ 171]	Input object Hand for the "BuildingEnergyLevel".
EnergLvlPr	FB_BA_MV_Op [▶ 171]	Resulting mode.
Sched	FB_BA_SchedM [▶ 159]	Schedule object (automatic) for the "BuildingEnergyLevel".

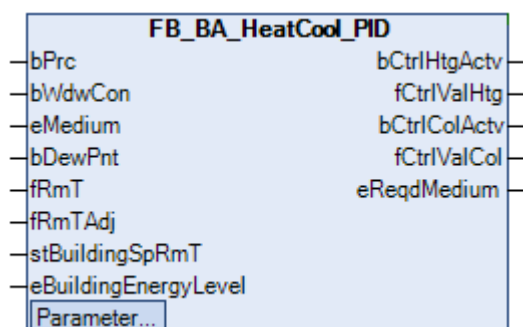
Variables

Name	Type	Description
DeMuxEnergLvl	FB_BA_DMUX_B08 [▶ 361]	Conversion of the multistate value of the manual mode to a binary output.
DeMuxSched	FB_BA_DMUX_B04 [▶ 361]	Conversion of the multistate value of the schedule to a binary output.
PrioSwi	FB_BA_PrioSwi_UDI08 [▶ 369]	Prioritizing reconversion of the states to a resulting multistate or enumeration value for the building energy level.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.1.3 FB_BA_HeatCool_PID



This template is used to control a heating-cooling zone.

The priority selection *EnergLvlSlcn* first determines the currently valid energy level.

The window contact always has priority and switches to the energy level *Protection* when the window is open. When there is a presence in the room, the *Comfort* level is always activated.

The function block *RmTAdj* [▶ 240] is used for a local shift of the room temperature setpoint for the energy levels *Pre-Comfort* and *Comfort*.

The instance *FncSel* of the function block *FB_BA_FncSel* activates the heating or cooling controller of the temperature zone.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA View \[► 173\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_HeatCool_PID EXTENDS FB_BA_View
VAR_INPUT
    bPrc                : BOOL;
    bWdwCon             : BOOL;
    eMedium             : E_BA_Medium;
    bDewPnt            : BOOL;
    fRmT               : REAL;
    fRmTAdj            : REAL;
    stBuildingSpRmT     : ST_BA_SpRmT;
    eBuildingEnergyLevel : E_BA_EnergyLvlEx;
END_VAR
VAR_OUTPUT
    bCtrlHtgActv       : BOOL;
    fCtrlValHtg        : REAL;
    bCtrlColActv       : BOOL;
    fCtrlValCol        : REAL;
    eReqdMedium        : E_BA_Medium;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    ePipeSys           : E_BA_PipeSys;
    nFnctSelChgOvrDly : UDINT;
END_VAR
VAR_INPUT CONSTANT
    CtrlHtg : FB_BA_Loop;
    CtrlCol : FB_BA_Loop;
END_VAR
VAR
    RmTAdj      : FB_BA_RmTAdj;
    EnergyLvlSlcn : FB_BA_PrioSwi_UDI04;
    SpSlcnHtg   : FB_BA_MUX_R04;
    SpSlcnCol   : FB_BA_MUX_R04;
    FnctSel     : FB_BA_FnctSel;
END_VAR
```

Inputs

Name	Type	Description
bPrc	BOOL	Presence detection.
bWdwCon	BOOL	Window contact (open = TRUE).
eMedium	E_BA_Medium [► 210]	Medium present (heating or cooling medium, only important for two-pipe operation).
bDewPnt	BOOL	Dew point sensor (alarm = TRUE).
fRmT	REAL	Room temperature [°C].
fRmTAdj	REAL	Room setpoint shift [K].
stBuildingSpRmT	ST_BA_SpRmT [► 215]	Structure of room setpoints (Protection Cooling..Comfort Cooling and Protection Heating ... Comfort Heating).
eBuildingEnergyLevel	E_BA_EnergyLvlEx [► 614]	Current building energy level.

Outputs

Name	Type	Description
bCtrlHtgActv	BOOL	Heating controller is active.
fCtrlValHtg	REAL	Control value heating valve.
bCtrlColActv	BOOL	Cooling controller is active.
fCtrlValCol	REAL	Control value cooling valve.
eReqdMedium	E_BA_Medium [► 210]	Requested medium (heating or cooling medium).

🔧 Inputs CONSTANT PERSISTENT

Name	Type	Description
ePipeSys	E_BA_PipeSys [► 210]	Selection of two or four-pipe system.
nFnctSelChgOvrDly	UDINT	Switchover delay [s] from heating to cooling or vice versa.

🔧 Inputs CONSTANT

Name	Type	Description
CtrlHtg	FB_BA_Loop [► 154]	Heating controller
CtrlCol	FB_BA_Loop [► 154]	Cooling controller

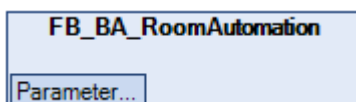
Variables

Name	Type	Description
RmTAdj	FB_BA_RmTAdj [► 240]	Function block that applies the sh to the corresponding setpoints and outputs them explicitly.
EnergLvlSlcn	FB_BA_PrioSwi_UDI04 [► 369]	Prioritizing selection and conversion of the possible energy levels into a numerical value.
SpSlcnHtg	FB_BA_MUX_R04 [► 367]	Selection of the heating setpoint.
SpSlcnCol	FB_BA_MUX_R04 [► 367]	Selection of the cooling setpoint.
FunctSel	FB_BA_FunctSel [► 237]	Function selection heating or cooling.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.2 FB_BA_RoomAutomation



Call program for the room automation specific templates

- FB_BA_RoomGeneral [► 643]
- Rooms, here as a sample FB_BA_RoomSample1 [► 644]

If the rooms are designed in the two-pipe system, the input *eMedium* must be assigned, which indicates whether heating or cooling is currently in progress.

In any case, the correct structure of the facade data for shading must be linked, which must be stored in the global variable list *Site* [► 818].

The template represents the technical system level.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA View \[► 173\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_RoomAutomation EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    RoomGeneral      : FB_BA_RoomGeneral;
    SampleRoom       : FB_BA_RoomSample1;
END_VAR
```

🔧 Inputs CONSTANT

Name	Type	Description
RoomGeneral	FB_BA_RoomGeneral [► 643]	Call program for subordinate room automation-specific templates in the plant level.
SampleRoom	FB_BA_RoomGeneral [► 643]	Calling a room template.

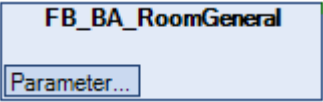
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.3 *General*

Call program for additional templates.

6.1.4.2.1.2.3.1 **FB_BA_RoomGeneral**



Call program for the room automation specific templates

- [FB_BA_BuildingEnergyLevel \[► 639\]](#)
- [FB_BA_BuildingSpRmT \[► 637\]](#)
- [FB_BA_BuildingSunprotection \[► 660\]](#)
- [FB_BA_Facade \[► 655\]](#)

The facade templates must be instantiated according to their number if they have blinds. In a typical square building, only the east, south and west sides are relevant.

The template represents the plant level.

The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

 FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

FB BA View [► 173]

Illustration

```

FUNCTION_BLOCK FB_BA_RoomGeneral EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    BuildEnergyLvl      : FB_BA_BuildingEnergyLevel;
    BuildSpRmT          : FB_BA_BuildingSpRmT;
    BuildSunPrtc        : FB_BA_BuildingSunprotection;
    FcdEast             : FB_BA_Facade;
    FcdSouth            : FB_BA_Facade;
    FcdWest             : FB_BA_Facade;
END_VAR

```

 Inputs CONSTANT

Name	Type	Description
BuildEnergyLvl	FB_BA_BuildingEnergyLevel [► 639]	Building energy level.
BuildSpRmT	FB_BA_BuildingSpRmT [► 637]	Building basic setpoints.
BuildSunPrtc	FB_BA_BuildingSunprotection [► 660]	Building-specific sun protection telegrams.
FcdEast	FB_BA_Facade [► 655]	Facade-specific blind data and telegrams Viewing direction east.
FcdSouth	FB_BA_Facade [► 655]	Facade-specific blind data and telegrams Viewing direction south.
FcdWest	FB_BA_Facade [► 655]	Facade-specific blind data and telegrams Viewing direction west

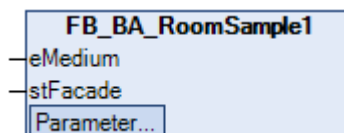
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.4 Room samples

Possible room templates.

6.1.4.2.1.2.4.1 FB_BA_RoomSample1



This template represents a sample room with heating, cooling and shading functions.

Since the requirements for rooms or zones are very different, corresponding changes and adjustments must be made to this template by the user.

- Heating/cooling function

The central function block of this area is the subordinate template [FB_BA_HeatCool_PID](#) [\[► 640\]](#), which controls a heating and a cooling valve (each a [FB_BA_ActuatorAnalog](#) [\[► 712\]](#)) on the basis of the room or zone sensors, the existing heating or cooling medium and the building data.

The presence detection and the window contacts are used to determine which energy level is to be reached: if the windows are open, the "Protection" level is selected as a priority. If they are closed and presence is detected, the Comfort level is decisive. Otherwise, the current generally applicable level of the building (global variable *eBuildingEnergyLevel*, see [Site](#) [\[► 818\]](#)) is assumed. In conjunction with a

room temperature sensor, in this sample the average of *RmTSen1* and *RmTSen2*, a small setpoint value shift *RmTAdj* and the room temperature setpoints (global Variable *stBuildingSpRmT*, see [Site \[► 818\]](#)), the corresponding controller is controlled. In a two-pipe system, the "correct" medium must be available throughout the building (*eMedium*). In cooling mode, the DewPointSensor must not have triggered.

- Shading

Using the template *FB_BA_SunblindZone*, 2 sun protection actuators (*FB_BA_SunBld* [\[► 651\]](#)) are controlled here as an example. For this purpose, the room temperature, as well as the presence for detection thermal automatic / sun protection is created. In addition to the room temperature levels (global variable *stBuildingSpRmT*, see [Site \[► 818\]](#)), which are required to assess the heating or cooling mode of the thermal automatic, the facade and building-specific telegrams and releases are retrieved via the global variable structure *stFacade* (see [Site \[► 818\]](#)).



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [\[► 195\]](#)

FB_BA_View [\[► 173\]](#)

Illustration

```
FUNCTION_BLOCK FB_BA_RoomSample1 EXTENDS FB_BA_View
VAR_INPUT
    eMedium          : E_BA_Medium;
    stFacade         : ST_BA_Facade;
END_VAR

VAR_INPUT CONSTANT
    Ctrl             : FB_BA_HeatCool_PID;
    VlvHtg           : FB_BA_ActuatorAnalog;
    VlvCol           : FB_BA_ActuatorAnalog;

    PrcDetc          : FB_BA_SensorBinary;
    Presence         : FB_BA_PresenceMonitoring;
    WdwCon1          : FB_BA_SensorBinary;
    WdwCon2          : FB_BA_SensorBinary;
    DewPointSensor   : FB_BA_SensorBinary;
    RmTSen1          : FB_BA_SensorAnalog;
    RmTSen2          : FB_BA_SensorAnalog;
    RmTAdj           : FB_BA_SensorAnalog;
    SunBldZone       : FB_BA_SunblindZone;
    SunBld1          : FB_BA_SunBld;
    SunBld2          : FB_BA_SunBld;
END_VAR

VAR
    RmTAvg           : FB_BA_EnAvrg02;
END_VAR

VAR
    bBlindUp         AT %I* : BOOL;
    bBlindDown       AT %I* : BOOL;
END_VAR
```



Inputs

Name	Type	Description
eMedium	E_BA_Medium [► 210]	Current medium, if there is a two-pipe system: heating or cooling
stFacade	ST_BA_Facade [► 615]	Facade-specific blind data and telegrams.

Inputs CONSTANT

Name	Type	Description
Ctrl	FB_BA_HeatCool_PID [► 640]	Heating-cooling control block.
VlvHtg	FB_BA_ActuatorAnalog [► 712]	Analog output object heating valve.
VlvCol	FB_BA_ActuatorAnalog [► 712]	Analog output object cooling valve.
PrcDetc	FB_BA_SensorBinary [► 790]	Binary input object occupancy sensor.
Presence	FB_BA_PresenceMonitoring [► 726]	Presence evaluation function block with switch-off delay and reset.
WdwCon1	FB_BA_SensorBinary [► 790]	Binary input object window contact.
WdwCon2	FB_BA_SensorBinary [► 790]	Binary input object window contact.
DewPointSensor	FB_BA_SensorBinary [► 790]	Binary input object dew point sensor.
RmTSen1	FB_BA_SensorAnalog [► 789]	Analog input object room temperature sensor.
RmTSen2	FB_BA_SensorAnalog [► 789]	Analog input object room temperature sensor.
RmTAdj	FB_BA_SensorAnalog [► 789]	Analog input object setpoint adjustment.
SunBldZone	FB_BA_SunblindZone [► 649]	Function block blind-specific zone functions.
SunBld1	FB_BA_SunBld [► 651]	Control block for a blind.
SunBld2	FB_BA_SunBld [► 651]	Control block for a blind.

VAR

Name	Type	Description
RmTAvg	FB_BA_EnAvrg02	Averaging over the room temperatures.
bBlindUp	BOOL	Control variable blinds up.
bBlindDown	BOOL	Control variable blinds down.

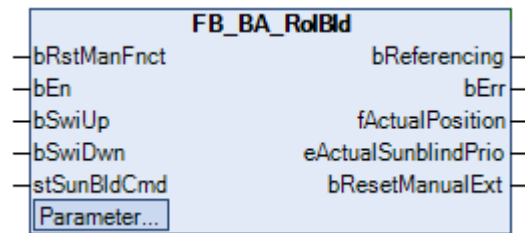
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.5 Sun protection

Templates for creating a custom sun protection.

6.1.4.2.1.2.5.1 FB_BA_RolBld



This template is used to control a roller blind actuator.

In the upper part of the representation of the template there is the local control option via a push button, which is enabled when one of the two inputs *bSwiUp* or *bSwiDwn* is activated.

This manual function is reset by the external control variables *bRstManFnc't* and *bResetManual_In*. Both variables come via the function block *FB_BA_Ext_SunblindBase*, which is also implemented in the further hierarchy via the inheriting function block *FB_BA_Ext_SunblindPosition*. Thereby *bRstManFnc't* appears as input, while *bResetManual_In* is only available as internal variable.

The external control (e.g. HMI) can generate telegrams via these function blocks, which are included in the inherited telegram variable *stSunblindCmdExt*.

The resulting diagram from the zone is routed via a drive delay *FB_BA_SunBldPosDly* [► 314].

This delays all automatic telegrams and is intended to ensure that, in the event of global control of blinds (e.g. fire alarm), not all blinds move at the same time and thus the breakaway starting current of the motors remains limited.

The three telegrams, that of the external positioner, the local manual controller via buttons and the delayed zone telegram, are passed on to the roller blind actuator via a telegram selector. The telegram selector has the task to put through the highest prioritized telegram. With telegrams of the same priority it is the last one sent (last writer wins).

At the end of the representation of the template, the current values and states are transferred to the external control, where they can be displayed.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_Ext_SunblindBase

FB_BA_Ext_SunblindPosition

Illustration

```
FUNCTION_BLOCK FB_BA_SunBld EXTENDS FB_BA_Ext_SunblindPosition
```

```
VAR_INPUT
    bRstManFnc't      : BOOL; // inherited
    bEn               : BOOL;
    bSwiUp            : BOOL;
    bSwiDwn           : BOOL;
    stSunBldCmd        : ST_BA_SunBld;
END_VAR
```

```
VAR_INPUT CONSTANT PERSISTENT
    nSwiOverTi        : UDINT;
    nPositioningDelay : UDINT;
```

```

nTiUp          : UDINT;
nTiDwn         : UDINT;
END_VAR

VAR
  SunBldSwi      : FB_BA_SunBldSwi := ( ePrio := E_BA_SunBldPrio.eManualActuator);
  EnSunBldSwi    : SR;

  ZonePosDly     : FB_BA_SunBldPosDly;
  PrioSwi        : FB_BA_SunBldTgmSel4;
  RolBldActr     : FB_BA_RolBldActr;

  bBusy          : BOOL;

  bCmdUp         AT %Q* : BOOL;
  bCmdDown       AT %Q* : BOOL;
END_VAR

```

Inputs

Name	Type	Description
bRstManFnct	BOOL	Input for resetting all internal manual functions, both those via the inputs <i>bSwiUp/bSwiDwn</i> , and external control (e.g. HMI).
bEn	BOOL	Enable the function block function.
bSwiUp	BOOL	Local push button "up".
bSwiDwn	BOOL	Local push button "down".
stSunBldCmd	<u>ST_BA_SunBld</u> [► 217]	Resulting telegram from the higher-level zone (room).

Inputs CONSTANT PERSISTENT

Name	Type	Description
nSwiOverTi	UDINT	Changeover time [ms] of the manual switch <i>SunBldSwi</i> for latching.
nPositioningDelay	UDINT	Transmission delay of the zone telegrams [ms] to distribute and limit breakaway starting currents in time.
nTiUp	UDINT	Complete time for driving up [ms].
nTiDwn	UDINT	Complete time for driving down [ms].

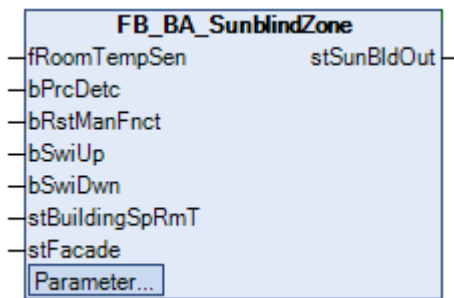
Variables

Name	Type	Description
SunBldSwi	<u>FB_BA_SunBldSwi</u> [► 321]	Function block for local positioning via push button inputs.
EnSunBldSwi	SR	Activation memory of the above mentioned function block.
ZonePosDly	<u>FB_BA_SunBldPosDly</u> [► 314]	Delay of telegrams from the zone (group) to avoid high breakaway starting currents due to simultaneity.
PrioSwi	<u>FB_BA_PrioSwi_UDI08</u> [► 369]	Prioritizing reconversion of the states to a resulting multistate or enumeration value for the building energy level.
RolBldActr	<u>FB_BA_RolBldActr</u> [► 298]	Function block for controlling a roller blind.
bBusy	BOOL	Internal active flag.
bCmdUp	BOOL	Output variable command "up".
bCmdDown	BOOL	Output variable command "down".

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.5.2 FB_BA_SunblindZone



In a blind zone, one or more blinds are combined for simultaneous control. The template *FB_BA_SunblindZone* bundles the predefined data of the facades and decides on the basis of a zone presence detection and local selection and deselection which functionality is active.

Together with the resulting telegram of high-priority functions from the facade, these functionalities are placed on a telegram selector at the end of the template. This then uses the priority to decide which telegram in the higher-level room template is passed on to the blinds.

Functions

- **Manual function**

The zone template contains a manual function that enables manual control of the blind via the key functions *bSwiUp/bSwiDwn*. The changeover time to latching, *nSwiOverTi* [ms], is pre-parameterized to 250 ms. The manual function is cleared via the input *bRstManFct*.

Info: the function block *FB_BA_SunBldSwi* [► 321] does not have automatic clearing of the latching. If a high-priority telegram, e.g. storm protection, wins in the zone, the blind moves according to its latching after this telegram is omitted.

- **Thermal automatic**

Thermal automatic is considered active if it is selected locally (parameter *bThAutoSlcn*), if no presence is detected (input *bPrcDetc*) and the facade template *FB_BA_Facade* [► 655] has determined the release for this building side. If it is active, the positioning is decided on the basis of the room temperature (input *fRoomTempSen*) and the building setpoints for heating and cooling: If the room temperature is above the building value for comfort cooling, the blinds move to a predefinable position. If, on the other hand, the temperature drops below the value for comfort heating, the blinds will open completely.

- **Automatic sun protection**

In contrast to the thermal automatic, the automatic sun protection is only active when presence is detected (input *bPrcDetc*). It must also be selected locally (parameter *bSunPrtcSlcn*) and be enabled by the facade template, among other things by the direction-dependent brightness. The blind position and angle are also determined in the facade.

- **Twilight automatic**

The twilight automatic is enabled in the facade on the basis of the brightness values. If the automatic for the zone is selected (input *bTwilGtAutoSlcn*) the blind moves to a predefined position in case of twilight (parameter *fTwilGtAutoPos/fTwilGtAutoAngl*).



The initialization of the template takes place within the method *FB_Init*.

Illustration

```
FUNCTION_BLOCK FB_BA_SunblindZone
VAR_INPUT
    fRoomTempSen      : REAL;
    bPrcDetc          : BOOL;
    bSwiUp            : BOOL;
    bSwiDwn           : BOOL;
    bRstManFct        : BOOL;
    stBuildingSpRmT    : ST_BA_SpRmT;
    stFacade           : ST_BA_Facade;
END_VAR
VAR_OUTPUT
    stSunBldOut        : ST_BA_SunBld;
END_VAR
```

```

VAR_INPUT CONSTANT PERSISTENT
    bThAutoSlcn      : BOOL;
    fThAutoColPos    : REAL;
    fThAutoColAngl   : REAL;
    bSunPrtcSlcn     : BOOL;
    bTwilLgtAutoSlcn : BOOL;
    fTwilLgtAutoPos   : REAL;
    fTwilLgtAutoAngl : REAL;
    nSwiOverTi       : UDINT;
END_VAR
VAR
    TwilLgtAuto      : FB_BA_SunBldEvt;
    SunPrtc          : FB_BA_SunBldEvt;
    ThAutoSwi        : FB_BA_Swi2P;
    ThAuto           : FB_BA_SunBldEvt;
    PrioSwi          : FB_BA_SunBldTgmSel8;
    ManSwi           : FB_BA_SunBldSwi;
    EnSunBldSwi      : SR;
END_VAR

```

Inputs

Name	Type	Description
fRoomtempSen	REAL	Room temperature sensor [°C].
bPrcDetc	BOOL	Presence detection.
bSwiUp	BOOL	Local push button "up".
bSwiDwn	BOOL	Local push button "down".
bRstManFnct	BOOL	Input for resetting all internal manual functions, both those via the inputs <i>bSwiUp/bSwiDwn</i> , and external control (e.g. HMI).
stBuildingSpRmT	ST_BA_SpRmT  215	Structure of room setpoints (Protection Cooling..Comfort Cooling and Protection Heating ... Comfort Heating).
stFacade	ST_BA_Facade  615	Facade-specific blind data and telegrams.

Outputs

Name	Type	Description
stSunBldOut	ST_BA_SunBld  217	resulting zone telegram.

Inputs CONSTANT PERSISTENT

Name	Type	Description
bThAutoSlcn	BOOL	Thermal automatic enable.
fThAutoColPos	REAL	Thermal automatic cooling position [%].
fThAutoColAngl	REAL	Thermal automatic cooling angle [°].
bSunPrtcSlcn	BOOL	Shading automatic enable.
bTwilLgtAutoSlcn	BOOL	Twilight automatic enable.
fTwilLgtAutoPos	REAL	Twilight automatic position [%].
fTwilLgtAutoAngl	REAL	Twilight automatic position [°].
nSwiOverTi	UDINT	Changeover time [ms] of the manual switch <i>SunBldSwi</i> for latching.

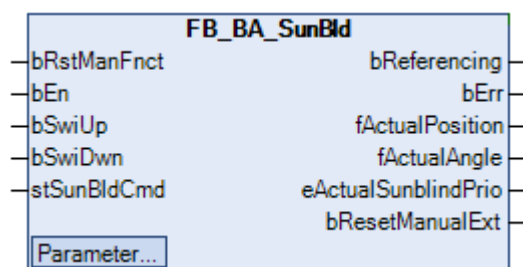
VAR

Name	Type	Description
TwiLgtAuto	FB_BA_SunBldEvt [► 311]	Telegram block for group (zone) twilight automatic.
SunPrtc	FB_BA_SunBldEvt [► 311]	Telegram block for group (zone) shading automatic.
ThAutoSwi	FB_BA_Swi2P [► 388]	Hysteresis block for heating/cooling changeover for group (zone) thermal automatic.
ThAuto	FB_BA_SunBldEvt [► 311]	Telegram block for group (zone) thermal automatic.
PrioSwi	FB_BA_PrioSwi_UDI08 [► 369]	Prioritizing reconversion of the states to a resulting multistate or enumeration value for the building energy level.
ManSwi	FB_BA_SunBldSwi [► 321]	Group (zone) push button block.
EnSunBldSwi	SR	Activation memory of the above mentioned function block.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.5.3 FB_BA_SunBld



This template is used to control a blind actuator.

In the upper part of the representation of the template there is the local control option via a push button, which is enabled when one of the two inputs *bSwiUp* or *bSwiDwn* is activated.

This manual function is reset by the external control variables *bRstManFnc* and *bResetManual_In*. Both variables come via the internal function block *FB_BA_Ext_SunblindBase*, which is also implemented in the further hierarchy via the internal inheriting function block *FB_BA_Ext_SunblindAngle*. Thereby *bRstManFnc* appears as input, while *bResetManual_In* is only available as internal variable.

The external control (e.g. HMI) can generate telegrams via these function blocks, which are included in the inherited telegram variable *stSunblindCmdExt*.

The resulting diagram from the zone is routed via a drive delay *FB_BA_SunBldPosDly* [► 314].

This delays all automatic telegrams and is intended to ensure that, in the event of global control of blinds (e.g. fire alarm), not all blinds move at the same time and thus the breakaway starting current of the motors remains limited.

The three telegrams, that of the external positioner, the local manual control via keys and the delayed zone telegram, are passed on to the blind control via a telegram selector. The telegram selector has the task to put through the highest prioritized telegram. With telegrams of the same priority it is the last one sent (last writer wins).

At the end of the representation of the template, the current values and states are transferred to the external control, where they can be displayed.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_Ext_SunblindBase

FB_BA_Ext_SunblindPosition

FB_BA_Ext_SunblindAngle

Illustration

FUNCTION_BLOCK FB_BA_SunBld EXTENDS FB_BA_Ext_SunblindAngle

```
VAR_INPUT
    bRstManFnct      : BOOL; // inherited
    bEn               : BOOL;
    bSwiUp            : BOOL;
    bSwiDwn           : BOOL;
    stSunBldCmd       : ST_BA_SunBld;
END_VAR
```

```
VAR_OUTPUT
    bReferencing      : BOOL;
    bErr              : BOOL;
    fActualPosition   : REAL;
    fActualAngle       : REAL;
    eActualSunblindPrio : E_BA_SunBldPrio;
    bResetManualExt   : BOOL;
END_VAR
```

```
VAR_INPUT CONSTANT PERSISTENT
    nSwiOverTi        : UDINT;
    nPositioningDelay : UDINT;
    nTiUp              : UDINT;
    nTiDwn             : UDINT;
    nTurnTiUp          : UDINT;
    nTurnTiDwn         : UDINT;
    nBckLshTiUp        : UDINT;
    nBckLshTiDwn       : UDINT;
    fAnglLmtUp         : REAL;
    fAnglLmtDwn        : REAL;
END_VAR
```

```
VAR
    SunBldSwi          : FB_BA_SunBldSwi := ( ePrio := E_BA_SunBldPrio.eManualActuator);
    EnSunBldSwi        : SR;

    ZonePosDly         : FB_BA_SunBldPosDly;
    PrioSwi             : FB_BA_SunBldTgmSel4;
    RolBldActr         : FB_BA_RolBldActr;

    bBusy              : BOOL;

    bCmdUp              AT %Q* : BOOL;
    bCmdDown            AT %Q* : BOOL;
END_VAR
```

 Inputs

Name	Type	Description
bRstManFnct	BOOL	Input for resetting all internal manual functions, both those via the inputs <i>bSwiUp</i> / <i>bSwiDwn</i> , and external control (e.g. HMI).
bEn	BOOL	Enable the function block function.
bSwiUp	BOOL	Local push button "up".
bSwiDwn	BOOL	Local push button "down".
stSunBldCmd	ST_BA_SunBld [► 217]	Resulting telegram from the higher-level zone (room).

 Outputs

Name	Type	Description
bReferencing	BOOL	Blind is being referenced.
bErr	BOOL	The internal function block <i>FB_BA_SunBldActr</i> is parameterized incorrectly.
fActualPosition	REAL	Current position (calculated).
fActualAngle	REAL	Current angle (calculated).
eActualSunblindPrio	E_BA_SunBldPrio [► 213]	Current priority with which the blind actuator is controlled.
bResetManualExt	BOOL	Manual function is reset externally (e.g. HMI).

 Inputs CONSTANT PERSISTENT

Name	Type	Description
nSwiOverTi	UDINT	Changeover time [ms] of the manual switch <i>SunBldSwi</i> for latching.
nPositioningDelay	UDINT	Transmission delay of the zone telegrams [ms] to distribute and limit breakaway starting currents in time.
nTiUp	UDINT	Complete time for driving up [ms].
nTiDwn	UDINT	Complete time for driving down [ms].
nTurnTiUp	UDINT	Time for turning the slats in the upward direction [ms].
nTurnTiDwn	UDINT	Time for turning the slats in the downward direction [ms].
nBckLshTiUp	UDINT	Time to traverse the backlash in the upward direction [ms].
nBckLshTiDwn	UDINT	Time to traverse the backlash in the downward direction [ms].
fAnglLmtUp	REAL	Highest position of the slats [°].
fAnglLmtDwn	REAL	Lowest position of the slats [°]. This position is reached once the blind has moved to the bottom position.

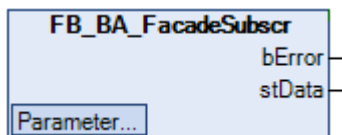
Variables

Name	Type	Description
SunBldSwi	FB_BA_SunBldSwi [► 321]	Function block for local positioning via push button inputs.
EnSunBldSwi	SR	Activation memory of the above mentioned function block.
ZonePosDly	FB_BA_SunBldPosDly [► 314]	Delay of telegrams from the zone (group) to avoid high breakaway starting currents due to simultaneity.
PrioSwi	FB_BA_PrioSwi_UDI08 [► 369]	Prioritizing reconversion of the states to a resulting multistate or enumeration value for the building energy level.
SunBldActr	FB_BA_SunBldActr [► 307]	Function block for controlling a blind.
bBusy	BOOL	Internal active flag.
bCmdUp	BOOL	Output variable command "up".
bCmdDown	BOOL	Output variable command "down".

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.5.4 FB_BA_FacadeSubscr



This template is used to receive the facade data of the type ST_BA_Facade [► 615] and is typically used in the floor controllers.

It represents an extension of FB_BA_Subscriber: In case of communication failure, the blinds telegram *stSunbld* is overwritten within the facade data with an active telegram position 0%.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

Illustration

```
FUNCTION_BLOCK FB_BA_FacadeSubscr EXTENDS FB_BA_Subscriber
VAR_OUTPUT
    stData          : ST_BA_Facade;
END_VAR
VAR
    fbHighPrio      : FB_BA_SunBldEvt := (ePrio := E_BA_SunBldPrio.eCommError);
END_VAR
```

🔴 Outputs

Name	Type	Description
stData	ST_BA_Facade [► 615]	Read facade telegram, overwritten in case of error.

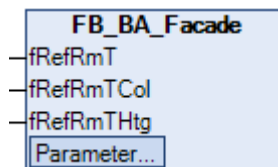
Variables

Name	Type	Description
_fbHighPrio	FB_BA_SunBldEvt [► 311]	Telegram block which activates a high-priority ($ePrio := E\_BA\_SunBldPrio.eCommError$) telegram with position 0% in the event of faulty communication and thus allows the blind to be raised completely.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.5.5 FB_BA_Facade



This template compiles the necessary telegrams for the shading of individual zones of a facade. In addition to the resulting alarm telegram from the building data (fire, burglary or icing), these are a communication error telegram and the facade-specific telegrams (storm, maintenance, facade thermal automatic, facade twilight automatic and parking position). In addition, the angle and position for a possible sun protection for this facade is calculated and transferred to the output *stFacade*.

Telegrams

- **Protection telegram communication error**

The global variable list *Site* [► 818] also contains the error states of the subscribers, which can only change to TRUE if the corresponding subscriber is actually used. In the event of a subscriber failure or a weather station malfunction, the blinds are raised as a precaution.

- **Protection telegram storm**

The data for storm protection - wind force and wind direction - come from the weather station. The field variables *aWndSpdLmtStormOn* and *aWndSpdLmtStormOff* can be used to define direction-dependent switch-on and switch-off values for this facade. In addition, the objects *DlyStrmOn* and *DlyStrmOff* define time hystereses. In the event of a storm, the blind raises completely.

- **Positioning telegram maintenance**

The objects *MntnPos* and *MntnAngl* can be used to specify a maintenance position and activate it with the enable *MntnEn*.

- **Positioning telegram facade thermal automatic**

A facade wide thermal automatic needs the temperature of a reference room as well as its heating and cooling setpoint. This data is specified via the input variables *fRefRmT*, *fRefRmTCol* and *fRefRmTHtg*. Four conditions must be met for the facade automation to become active:

- the building must be unoccupied, this is determined on the basis of the currently valid energy level.
- the global radiation-dependent release from the template *FB_BA_BuildingSunprotection* [► 660] must be given.
- the thermal automatic is selected via the object *FcdThAutoEn*.
- the facade must be illuminated by the sun.

If these conditions are met, the blind moves alternately to cooling position (objects *FcdThAutoColPos* and *FcdThAutoColAngl*) if the reference temperature exceeds the cooling value or to position 0 if the heating value is fallen below.

The thermal automatic can be selected and deselected via the object *FcdThAutoEn*.

- **Positioning telegram twilight automatic**

The template `FB_BA_BuildingSunprotection` [► 660] uses the brightness values and definable switch-on and switch-off thresholds to determine whether the twilight automatic is enabled or not. If it is enabled and selected via the object `FcdTwlGtAutoEn`, the blinds will be lowered completely in the event of twilight so that people still in the building are protected from view.

- **Positioning telegram parking position**

Using the objects `FcdParkPos` and `FcdParkAngl`, a general parking position can be defined and approached via the enable object `FcdParkEn`.

Sun protection calculation

Sun protection is calculated separately for each facade. For this purpose, the corresponding value for the facade is averaged on the basis of the facade orientation (north=0°, east=90°, south=180°, west=270°) and the 4 measured direction-dependent brightness values.

Info: In the southern hemisphere: South=0°, East=90°, North=180°, West=270°. The interpolation must be adjusted for this case.

Based on the averaged brightness value, a hysteresis block decides whether the automatic sun protection for the facade is active or not. This is done with the brightness thresholds `SunPrtcValOn/SunPrtcValOff` and the associated delay times `SunPrtcDlyOn/SunPrtcDlyOff`.

The automatic sun protection is preconfigured in such a way that, in the event of protection, it lowers the blinds completely and only varies the slat angle.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Illustration

```
FUNCTION_BLOCK FB_BA_Facade EXTENDS FB_BA_View
VAR_INPUT
    fRefRmt          : REAL;
    fRefRmtTCol      : REAL;
    fRefRmtTHtg      : REAL;
END_VAR

VAR_OUTPUT
    stFacade         : ST_BA_Facade;
END_VAR

VAR_INPUT CONSTANT PERSISTENT
    fFcdOrtn         : REAL;
    fFcdAngl         : REAL;
    fLamWdth         : REAL;
    fLamDstc         : REAL;
    fAzmSttRng       : REAL;
    fAzmEndRng       : REAL;
    fElvLoLmt        : REAL;
    fElvHiLmt        : REAL;
    aWndSpdLmtStormOn : ARRAY[1..32] OF REAL;
    aWndSpdLmtStormOff : ARRAY[1..32] OF REAL;
END_VAR

VAR_INPUT CONSTANT
    DlyStrmOn        : FB_BA_AV_Op;
    DlyStrmOff       : FB_BA_AV_Op;
    MntnEn           : FB_BA_BV_Op;
    MntnPos          : FB_BA_AV_Op;
```

```

MtnAngl      : FB_BA_AV_Op;
FcdThAutoEn  : FB_BA_BV_Op;
FcdThAutoColPos : FB_BA_AV_Op;
FcdThAutoColAngl : FB_BA_AV_Op;
FcdTwilGtAutoEn : FB_BA_BV_Op;
SunPrtcValOn  : FB_BA_AV_Op;
SunPrtcValOff : FB_BA_AV_Op;
SunPrtcDlyOn  : FB_BA_AV_Op;
SunPrtcDlyOff : FB_BA_AV_Op;
FcdParkEn    : FB_BA_BV_Op;
FcdParkPos   : FB_BA_AV_Op;
FcdParkAngl  : FB_BA_AV_Op;
WndPrtcState : FB_BA_BV_Op;
SunPrtcState : FB_BA_BV_Op;
END_VAR

VAR
  aStormDirection : ARRAY[1..32] OF REAL:= [0.0, 30.0, 60.0, 90.0, 120.0, 150.0, 180.0, 210.0, 240.0, 270.0, 300.0, 330.0, 360.0];

  StormOnInterpolation : FB_BA_Chrc32;
  fWndSpdStormOn       : REAL;

  StormOffInterpolation : FB_BA_Chrc32;
  fWndSpdStormOff       : REAL;

  SunBldWndPrtc : FB_BA_SunBldWndPrtc;

  CommError : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eCommError);
  SunBldMtnEvent : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eMaintenance);

  bFcdInSun : BOOL;
  InRngAzm : FB_BA_InRngAzm;
  InRngElv : FB_BA_InRngElv;

  FcdThAutoHtgCol : FB_BA_Swi2P;
  bFcdThAuto_Enable : BOOL;
  fFcdThAuto_Position : REAL;
  fFcdThAuto_Angle : REAL;
  FcdThAutoEvent : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eFacadeThermoAutomatic);

  bFcdTwilGtAuto_Enable : BOOL;
  FcdTwilGtAutoEvent : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eFacadeTwilightAutomatic);

  FcdParkPosEvent : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eParkPosition);

  SunBldTgmResult : FB_BA_SunBldTgmSel8;

  aBrtnsDirection : ARRAY[1..32] OF REAL := [0.0, 90.0, 180.0, 270.0, 360.0];
  aBrtnsAtDirection : ARRAY[1..32] OF REAL;
  BrtnsInterpolation : FB_BA_Chrc32;
  fBrtnsInterpolated : REAL;
  SunPrtcValOnOff : FB_BA_Swi2P;
  SunPrtc : FB_BA_SunPrtc;
END_VAR

```

Inputs

Name	Type	Description
fRefRmT	REAL	Room temperature of the reference room for the facade-wide thermal automatic.
fRefRmTCol	REAL	Room temperature setpoint cooling of the reference room for the facade-wide thermal automatic.
fRefRmTHtg	REAL	Room temperature setpoint heating of the reference room for the facade-wide thermal automatic.

Outputs

Name	Type	Description
stFacade	ST_BA_Facade [► 615]	Output structure of the collected facade data.

 Inputs **CONSTANT PERSISTENT**

Name	Type	Description
fFcdOrtn	REAL	Facade orientation northern hemisphere: north=0°, east=90°, south=180°, west=270°, in the southern hemisphere applies: south=0°, east=90°, north=180°, west=270°.
fFcdAngl	REAL	Inclination of the facade [°]. Inclined downwards, the angle is smaller, upwards it is greater than zero.
fLamWdth	REAL	Slat width [mm].
fLamDstc	REAL	Slat spacing [mm].
fAzmSttRng / fAzmEndRng	REAL	The facade is considered to be illuminated by the sun when the position of the sun is +/-90° of the facade orientation. With <i>fAzmSttRng</i> / <i>fAzmEndRng</i> the range can be restricted.
fElvLoLmt / fElvHiLmt	REAL	The facade is considered to be illuminated by the sun when the sun elevation is between 0 and 90°. With <i>fElvLoLmt</i> / <i>fElvHiLmt</i> the range can be restricted.
aWndSpdLmtStorm On	ARRAY[1..32] OF REAL	<i>aWndSpdLmtStormOn[01]...aWndSpdLmtStormOn[12]</i> : These values define the interpolation curve for the direction-dependent switch-on threshold of the storm detection. Here <i>aWndSpdLmtStormOn[01]</i> applies to the wind direction at 0° (north). The other field elements represent the interpolation points in 30° steps. The unit is m/s.
aWndSpdLmtStorm Off	ARRAY[1..32] OF REAL	<i>aWndSpdLmtStormOff[01]...aWndSpdLmtStormOff[12]</i> : These values define the interpolation curve for the direction-dependent switch-off threshold of the storm detection. Here <i>aWndSpdLmtStormOff[01]</i> applies to the wind direction at 0° (north). The other field elements represent the interpolation points in 30° steps. The unit is m/s.

🔧 Inputs **CONSTANT**

Name	Type	Description
DlyStrmOn	FB BA AV Op ▶ 136	Delay value for storm detection [s].
DlyStrmOff	FB BA AV Op ▶ 136	Delay value storm abated [s].
MntnEn	FB BA BV Op ▶ 150	Maintenance position On.
MntnPos	FB BA AV Op ▶ 136	Maintenance position Position [%].
MntnAngl	FB BA AV Op ▶ 136	Maintenance position Angle [°].
FcdThAutoEn	FB BA BV Op ▶ 150	Thermal automatic function enable.
FcdThAutoColPos	FB BA AV Op ▶ 136	Facade thermal automatic cooling position Position [%].
FcdThAutoColAngl	FB BA AV Op ▶ 136	Facade thermal automatic cooling position Angle [°].
FcdTwilGtAutoEn	FB BA BV Op ▶ 150	Facade twilight automatic enable.
SunPrtcValOn	FB BA AV Op ▶ 136	Brightness threshold sun protection on [lx].
SunPrtcValOff	FB BA AV Op ▶ 136	Brightness threshold sun protection off [lx].
SunPrtcDlyOn	FB BA AV Op ▶ 136	Delay sun protection on [s].
SunPrtcDlyOff	FB BA AV Op ▶ 136	Delay sun protection off [s].
FcdParkEn	FB BA BV Op ▶ 150	Facade parking position enable.
FcdParkPos	FB BA AV Op ▶ 136	Facade parking position Position.
FcdParkAngl	FB BA AV Op ▶ 136	Facade parking position Angle.
WndPrtcState	FB BA BV Op ▶ 150	Display object state storm protection.
SunPrtcState	FB BA BV Op ▶ 150	This object indicates whether the release of the sun protection on the part of the facade is given. Whether the blinds actually assume the sun protection position depends on criteria within the room zones (e.g. presence).

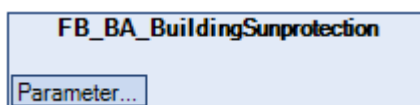
Variables

Name	Type	Description
aStormDirection	ARRAY[1..32] OF REAL	Pre-set field of cardinal points for weighting the storm limits.
StormOnInterpolation	FB_BA_Chrc32 [► 401]	Calculation block for the limit value storm detection.
fWndSpdStormOn	REAL	Calculated value limit value storm detection.
StormOffInterpolation	FB_BA_Chrc32 [► 401]	Calculation block for the limit value Storm abated.
fWndSpdStormOff	REAL	Calculated value limit value Storm abated.
SunBldWndPrtc	FB_BA_SunBldWndPrtc [► 327]	Triggering and telegram block for storm protection.
CommError	FB_BA_SunBldEvt [► 311]	Telegram block for the communication error.
SunBldMntnEvent	FB_BA_SunBldEvt [► 311]	Telegram block for the maintenance position.
bFcdInSun	BOOL	Facade in the sun.
InRngAzm	FB_BA_InRngAzm [► 283]	Sun direction is within the defined limits.
InRngElv	FB_BA_InRngElv [► 285]	Sun elevation is within the defined limits.
FcdThAutoHtgCol	FB_BA_Swi2P [► 388]	Hysteresis block for heating/cooling changeover for facade thermal automatic.
bFcdThAuto_Enable	BOOL	Enable thermal automatic.
fFcdThAuto_Position	REAL	Thermal automatic Position [%].
fFcdThAuto_Angle	REAL	Thermal automatic Angle [°].
FcdThAutoEvent	FB_BA_SunBldEvt [► 311]	Telegram block for facade thermal automatic.
bFcdTwiLgtAuto_Enable	BOOL	Enable twilight automatic.
FcdTwiLgtAutoEvent	FB_BA_SunBldEvt [► 311]	Telegram block for facade twilight automatic.
FcdParkPosEvent	FB_BA_SunBldEvt [► 311]	Telegram block for the parking position.
SunBldTgmResult	FB_BA_SunBldTgmSel8 [► 324]	Telegram selection block for the resulting blind telegram of the facade.
aBrtnsDirection	ARRAY [1..32] OF REAL	Preset field of cardinal points for weighting the brightness.
aBrtnsAtDirection	ARRAY [1..32] OF REAL	Input field for the brightness related to the above mentioned cardinal points.
BrtnsInterpolation	FB_BA_Chrc32 [► 401]	Calculation block for brightness.
fBrtnsInterpolated	REAL	Calculated value of the brightness.
SunPrtcValOnOff	FB_BA_Swi2P [► 388]	Hysteresis block for switching on/off the sun protection.
SunPrtc	FB_BA_SunPrtc [► 329]	Calculation block for sun protection.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.5.6 FB_BA_BuildingSunprotection



This template collects cross-building criteria for the blind functions, which are then further used in the facade instances of the [FB_BA_Facade \[► 655\]](#). These are in detail:

- **resulting telegram building alarms**

- **Protection telegram Fire**

The template `FB_BA_BuildingAlarms` [► 623] supplies the information "Fire alarm" via a globally declared structure variable `stBuildingAlarms` (see global variable list [Site](#) [► 818]). If this alarm is active, the blinds are raised completely via a `FB_BA_SunBldEvt` [► 311].

- **Protection telegram Burglar**

The template `FB_BA_BuildingAlarms` [► 623] supplies the information "burglar alarm" via a globally declared structure variable `stBuildingAlarms` (see global variable list [Site](#) [► 818]). If this alarm is active, the blinds are raised completely via a `FB_BA_SunBldEvt` [► 311]. This makes the burglar less hidden from view from the outside.

- **Protection telegram icing**

An imminent icing is detected by the fact that during a precipitation detection the measured outside temperature is below the frost limit value - this is specified here by the object *SpIce* and is preset to -2 °C. If the outside temperature exceeds the frost limit value for the time specified at the object *DeiceTi*, frost protection is canceled again. In the case of the icing alarm, a telegram is output at the function block `FB_BA_SunBldIcePrtc` [► 312], which causes the blinds to be raised completely.

- **Thermal automatic**

As a rule, a weather station does not measure the global heat radiation depending on the direction. Therefore, it makes sense to define a switch-on and a switch-off threshold per building. The objects *GlobalThAutoValOn* and *GlobalThAutoValOff* define these threshold values in W/m² and form the building-wide release of the thermal automatic via a hysteresis switch (object *GlobalThAutoRlse*).

- **Twilight automatic**

The twilight, which defines only a short period during the day, is also defined across buildings: the 4 direction-dependent brightness values of the weather station are averaged and assigned the threshold values *GlobalTwilGtAutoValOn* and *GlobalTwilGtAutoValOff*.

- **Sun protection**

The interval for repositioning the slats in the sun protection functions is specified here for the entire building.

- **Reset of the manual functions**

A building-wide criterion for the reset of the manual functions is defined here. It is based on the building schedule of energy levels, which give an inference of absence. Alternatively, a binary input object is available.

The above protection telegrams are combined on a priority switch `FB_BA_SunBldTgmSel4` [► 323] to a resulting telegram.

This is made available at the end of the template with the building-specific sun protection data in a globally declared variable structure *stBuildingSunBlind* (see global variable list [Site](#) [► 818]).



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

`FB_BA_Base`

`FB_BA_BasePublisher`

`FB_BA_Object` [► 195]

`FB_BA_View` [► 173]

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingSunprotection EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    SpIce          : FB_BA_AV_Op;
    SunPrtcPosIntval : FB_BA_AV_Op;
    DeiceTi        : FB_BA_AV_Op;
    GlobalThAutoValOn : FB_BA_AV_Op;
```

```

GlobalThAutoValOff      : FB_BA_AV_Op;
GlobalTwilGtAutoValOn   : FB_BA_AV_Op;
GlobalTwilGtAutoValOff  : FB_BA_AV_Op;
IceAlert                 : FB_BA_BV;
GlobalThAutoRlse        : FB_BA_BV_Op;
GlobalTwilGtAutoRlse    : FB_BA_BV_Op;
GlobalResetManMode      : FB_BA_BV_Op;
END_VAR

VAR
  FireAlert              : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eFire);
  Burglary               : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eBurglary);
  SunBldIcePrtc          : FB_BA_SunBldIcePrtc;
  BuildingAlarms         : FB_BA_SunBldTgmSel4;
  tonInitialWait         : TON;
  GlobalThAutoValOnOff   : FB_BA_Swi2P;
  GlobalTwilGtAutoValOnOff : FB_BA_Swi2P;
  rtManResetEnergLvl     : R_TRIG;
END_VAR

```

Inputs CONSTANT

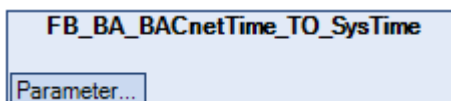
Name	Type	Description
Splce	FB_BA_AV_Op [► 136]	Analog input object: Icing temperature limit [°C].
SunPrtcPosIntval	FB_BA_AV_Op [► 136]	Analog input object: Readjustment interval of the slat angle [min].
DeiceTi	FB_BA_AV_Op [► 136]	Analog input object for deicing the blind after icing up [s]. After that the ice alarm is reset.
GlobalThAutoValOn	FB_BA_AV_Op [► 136]	Analog input object: Global radiation threshold for enabling the thermal automatic [W/m²].
GlobalThAutoValOff	FB_BA_AV_Op [► 136]	Analog input object: Global radiation threshold for switching off the thermal automatic [W/m²].
GlobalTwilGtAutoValOn	FB_BA_AV_Op [► 136]	Analog input object: Brightness threshold for enabling twilight automatic [lx].
GlobalTwilGtAutoValOff	FB_BA_AV_Op [► 136]	Analog input object: Brightness threshold for enabling twilight automatic [lx].
IceAlert	FB_BA_BV [► 147]	Binary display object: Icing alarm.
GlobalThAutoRlse	FB_BA_BV_Op [► 150]	Binary display object: building-wide enable of thermal automatic due to global radiation.
GlobalTwilGtAutoRlse	FB_BA_BV_Op [► 150]	Binary display object: building-wide enable of the thermal automatic due to the average brightness.
GlobalResetManMode	FB_BA_BV_Op [► 150]	Binary input object: Operating option for resetting manual functions.

Variables

Name	Type	Description
FireAlert	FB_BA_SunBldEvt [▶ 311]	Telegram block for the fire alarm: lets the blind raise completely.
Burglary	FB_BA_SunBldEvt [▶ 311]	Telegram block for burglary: lets the blind raise completely.
SunBldIcePrtc	FB_BA_SunBldIcePrtc [▶ 312]	Triggering and telegram block for the icing alarm.
BuildingAlarms	FB_BA_SunBldTgmSel4 [▶ 323]	Priority selection block.
tonInitialWait	TON	Start-up delay anti-icing.
GlobalThAutoValOnOff	FB_BA_Swi2P [▶ 388]	Hysteresis block for switching on/off the thermal automatic (global criterion).
GlobalTwiLgtAutoValOnOff	FB_BA_Swi2P [▶ 388]	Hysteresis block for switching on/off the twilight automatic (global criterion).
rtManResetEnergLvl	R_TRIG	Trigger reset of the manual function for the criterion energy level (continuous signal).

Requirements

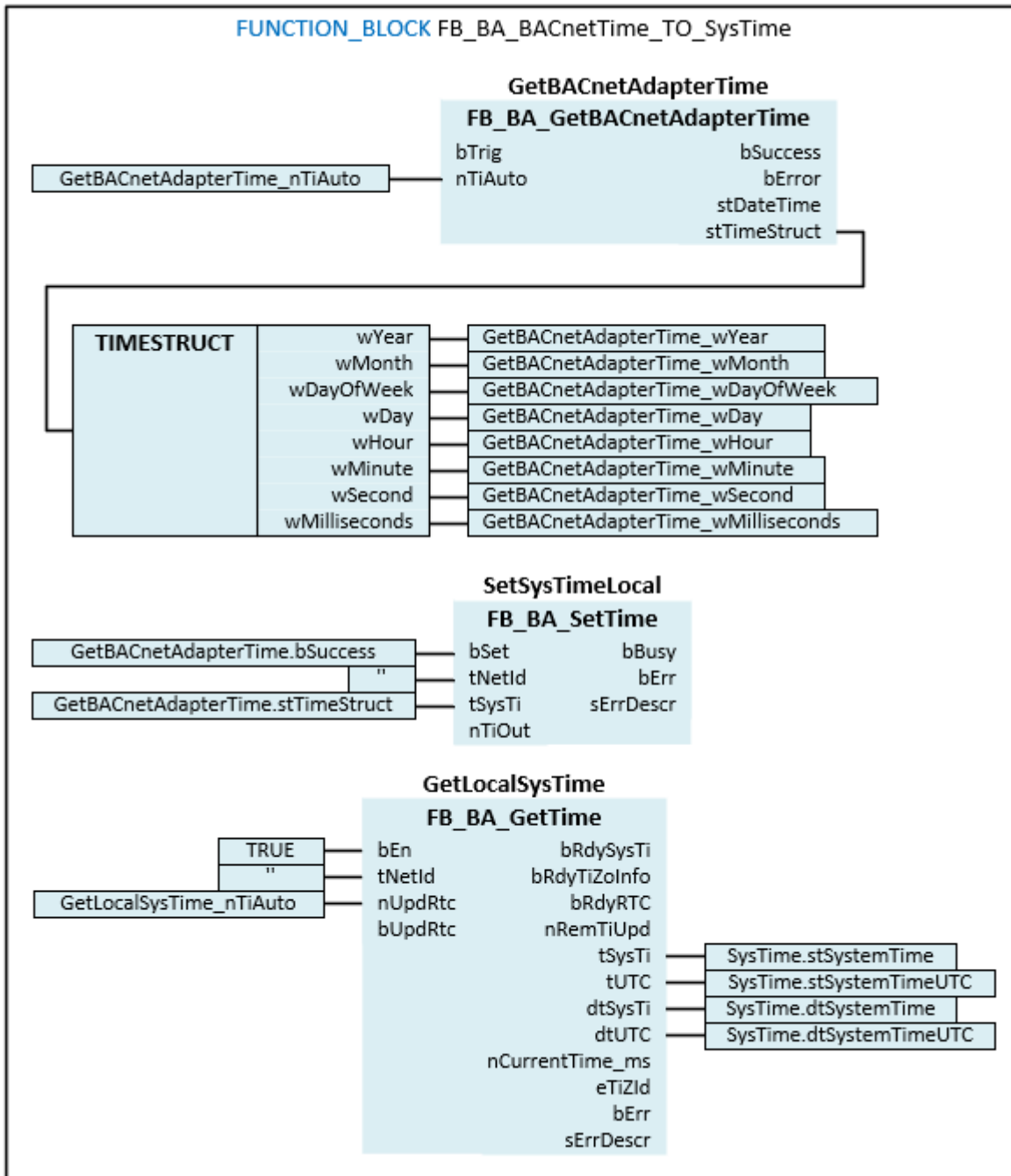
Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3 Automation control**6.1.4.2.1.3.1 Device****6.1.4.2.1.3.1.1 TimeSettings****6.1.4.2.1.3.1.1.1 FB_BA_BACnetTime_TO_SysTime**

The template reads the BACnet system time at regular intervals and writes it to the local NT system time of the TwinCAT system.

The NT system time is read at regular intervals and the time information is mapped in the PLC to local and global variables [[▶ 819](#)].

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_BACnetTime_TO_SysTime
VAR_INPUT CONSTANT
  {attribute 'parameterUnit':='s'}
  GetBACnetAdapterTime_nTiAuto      : UDINT := 600;
  {attribute 'parameterUnit':='s'}
  GetLocalSysTime_nTiAuto           : UDINT := 60;
END_VAR
VAR_INPUT CONSTANT
  SetSysTimeLocal                   : FB_BA_SetTime;
  GetLocalSysTime                   : FB_BA_GetTime;
  GetBACnetAdapterTime              : FB_BA_GetBACnetAdapterTime;
END_VAR
VAR
  GetBACnetAdapterTime_wYear        : WORD;
  GetBACnetAdapterTime_wMonth       : WORD;
  GetBACnetAdapterTime_wDayOfWeek   : WORD;
  GetBACnetAdapterTime_wDay         : WORD;
  GetBACnetAdapterTime_wHour        : WORD;
  GetBACnetAdapterTime_wMinute      : WORD;

```

```

GetBACnetAdapterTime_wSecond      : WORD;
GetBACnetAdapterTime_wMilliseconds : WORD;
END_VAR

```

Inputs CONSTANT

Name	Type	Description
GetBACnetAdapterTime_nTiAuto	UDINT	Based on this time information, the local BACnet system time of a TwinCAT system is regularly determined automatically.
GetLocalSysTime_nTiAuto	UDINT	Based on this time information, the local NT system time of a TwinCAT system is regularly determined.
SetSysTimeLocal	FB BA SetTime	The locally determined BACnet system time (GetBACnetAdapterTime) is written to the local NT system time of the TwinCAT system using the function block SetSysTimeLocal.
GetLocalSysTime	FB BA GetTime	The function block GetLocalSysTime determines the local NT system time of the TwinCAT system. The determined time is transferred to the global variable lists SysTime [▶ 819].
GetBACnetAdapterTime	FB BA GetBACnetAdapterTime [▶ 344]	The function block "GetBACnetAdapterTime" determines the local BACnet system time.

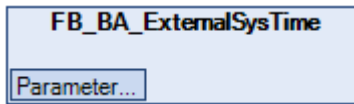
VAR

Name	Type	Description
GetBACnetAdapterTime_wYear	WORD	Year specification of the BACnet system time.
GetBACnetAdapterTime_wMonth	WORD	Month specification of the BACnet system time.
GetBACnetAdapterTime_wDayOfWeek	WORD	Indication of the day of the week of the BACnet system time.
GetBACnetAdapterTime_wDay	WORD	Indication of the day in the month of the BACnet system time.
GetBACnetAdapterTime_wHour	WORD	Hour specification of the BACnet system time.
GetBACnetAdapterTime_wMinute	WORD	Minute specification of the BACnet system time.
GetBACnetAdapterTime_wSecond	WORD	Seconds specification of the BACnet system time.
GetBACnetAdapterTime_wMilliseconds	WORD	Milliseconds specification of the BACnet system time.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.1.1.2 FB_BA_ExternalSysTime

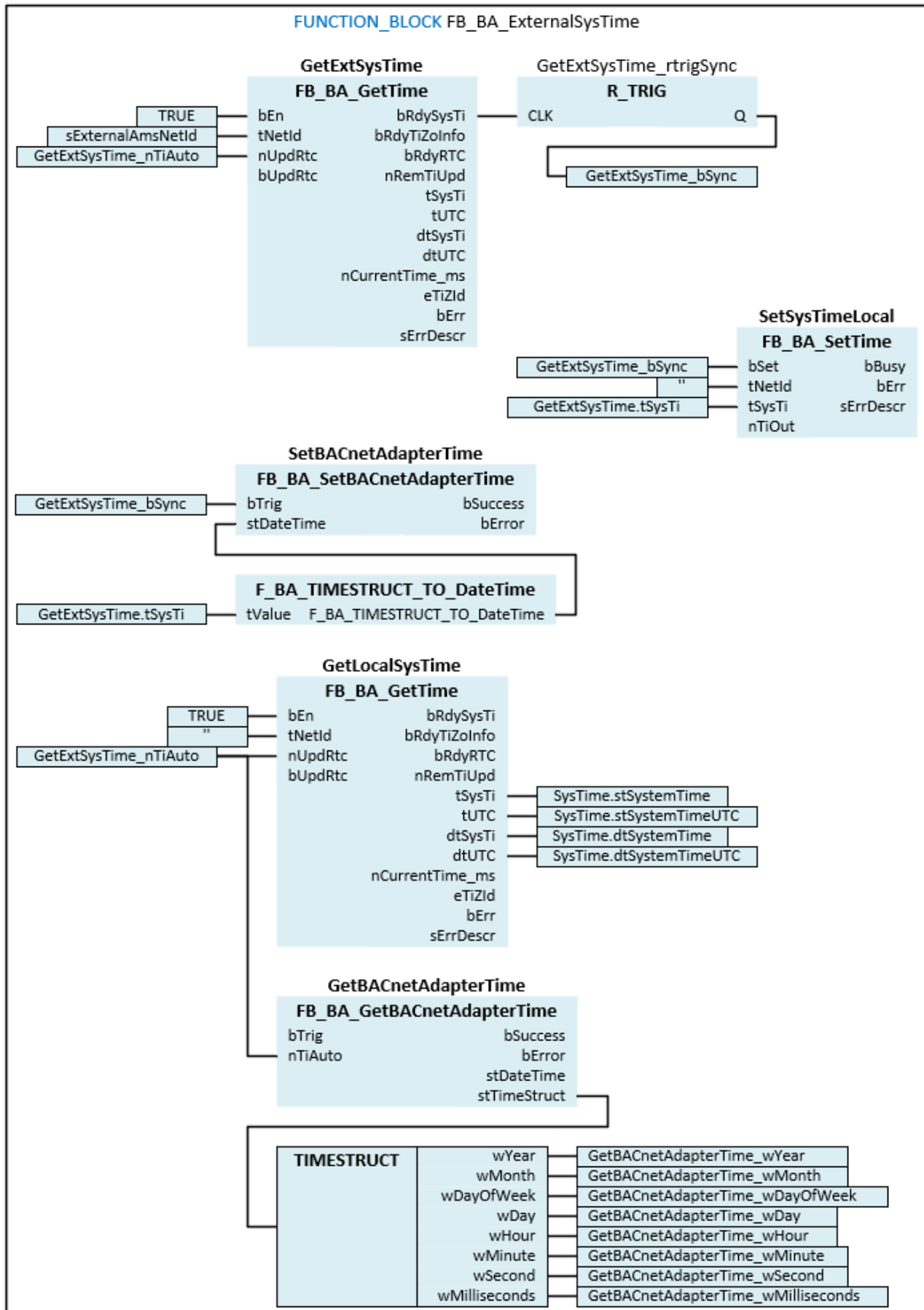


The template reads the NT system time of an external TwinCAT system at regular intervals.

If the readout was successful, then the external system time is written to the local NT system time of the TwinCAT system and the local BACnet system time of the TwinCAT system.

The two local system times are read out at regular intervals and mapped to local and global variables in the PLC.

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_ExternalSysTime
VAR_INPUT CONSTANT
    sExternalAmsNetId          : T_AmsNetId := '';
    {attribute 'parameterUnit':= 's'}
    GetExtSysTime_nTiAuto      : UDINT := 600;
    {attribute 'parameterUnit':= 's'}
    GetLocalSysTime_nTiAuto    : UDINT := 60;
END_VAR
VAR_INPUT CONSTANT
    GetExtSysTime              : FB_BA_GetTime;

    SetSysTimeLocal            : FB_BA_SetTime;
    SetBACnetAdapterTime       : FB_BA_SetBACnetAdapterTime;

    GetLocalSysTime            : FB_BA_GetTime;
    GetBACnetAdapterTime       : FB_BA_GetBACnetAdapterTime;
END_VAR
VAR
    GetExtSysTime_bSync        : BOOL;
    GetExtSysTime_rtrigSync    : R_TRIG;

    GetBACnetAdapterTime_wYear : WORD;
    GetBACnetAdapterTime_wMonth : WORD;
    GetBACnetAdapterTime_wDayOfWeek : WORD;
    GetBACnetAdapterTime_wDay : WORD;
    GetBACnetAdapterTime_wHour : WORD;
    GetBACnetAdapterTime_wMinute : WORD;
    GetBACnetAdapterTime_wSecond : WORD;
    GetBACnetAdapterTime_wMilliseconds : WORD;
END_VAR
```

 Inputs **CONSTANT**

Name	Type	Description

sExternalAmsNetId	T_AmsNetId	The AmsNetId of the TwinCAT computer whose NT system time is to be read is specified here.
GetBACnetAdapterTime_nTiAuto	UDINT	Based on this time information, the local BACnet system time of a TwinCAT system is regularly determined automatically.
GetLocalSysTime_nTiAuto	UDINT	Based on this time information, the local NT system time of a TwinCAT system is regularly determined.
GetExtSysTime	<u>FB BA GetTime</u>	The GetExtSysTime function block determines the external NT system time of a TwinCAT system.
SetSysTimeLocal	<u>FB BA SetTime</u>	The locally determined BACnet system time (GetBACnetAdapterTime) is written to the local NT system time of the TwinCAT system using the function block SetSysTimeLocal.
SetBACnetAdapterTime	<u>FB BA SetBACnetAdapterTime</u> [► 343]	The function block "SetBACnetAdapterTime" sets the local BACnet system time of the TwinCAT system.
GetLocalSysTime	<u>FB BA GetTime</u>	The function block GetLocalSysTime determines the local NT system time of the TwinCAT system. The determined time is transferred to the global variable lists <u>SysTime</u> [► 819].
GetBACnetAdapterTime	<u>FB BA GetBACnetAdapterTime</u> [► 344]	The function block "GetBACnetAdapterTime" determines the local BACnet system time.

VAR

Name	Type	Description
GetExtSysTime_bSync	BOOL	The variable triggers the writing of the external system time to the local NT system time of the TwinCAT system and the local BACnet system time of the TwinCAT system.
GetExtSysTime_rtrgSync	R_TRIG	The function block generates a rising edge after successfully reading the external system time <i>GetExtSysTime</i> and passes this on to <i>GetExtSysTime_bSync</i> .
GetBACnetAdapterTime_wYear	WORD	Year specification of the BACnet system time.
GetBACnetAdapterTime_wMonth	WORD	Month specification of the BACnet system time.
GetBACnetAdapterTime_wDayOfWeek	WORD	Indication of the day of the week of the BACnet system time.
GetBACnetAdapterTime_wDay	WORD	Indication of the day in the month of the BACnet system time.
GetBACnetAdapterTime_wHour	WORD	Hour specification of the BACnet system time.
GetBACnetAdapterTime_wMinute	WORD	Minute specification of the BACnet system time.
GetBACnetAdapterTime_wSecond	WORD	Seconds specification of the BACnet system time.
GetBACnetAdapterTime_wMilliseconds	WORD	Milliseconds specification of the BACnet system time.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.1.2 BACnetSettings

6.1.4.2.1.3.1.2.1 FB_BA_Parameter

FB_BA_Parameter

The function block contains the function block *FB_BA_Param* in the declaration part of the method *FB_init*. This is used to parameterize the global variable lists *XBA_BACnetParam* and *XBA_Param* [► 116] of the library *Tc3_XBA* [► 92]. In the implementation part of the method *FB_init* various parameterizations are listed for this purpose.



The initialization of the template takes place within the method *FB_Init*.



A call of the *FB_BA_Param* is not necessary due to the internal attributes *{attribute 'no_explicit_call' := 'No need to call this FB.'}* and *{attribute 'TcIgnorePersistent'}*, see *FB_BA_Param* [► 124]!

There is also a commented out example in the template, which contains the parameterization of the recipient list of the message/event class objects. This sample must be adapted to the project in question.

Illustration

```
FUNCTION_BLOCK FB_BA_Parameter
VAR
// Recipient      : FB_BACnet_RcpList;
END_VAR
```

Implementation part

```
// Recipient list to write (to all event classes):
// Recipient();
```

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.1.2.2 FB_BACnet_RcpList**FB_BACnet_RcpList**

The sample shows the parameterization of the recipient list of the message/event class objects. This sample must be adapted to the project in question.

Illustration

```
FUNCTION_BLOCK FB_BACnet_RcpList
VAR bWrite : BOOL ;

// Recipient list to write (to all event classes):
aRecipientList : T_BACnet_RecipientList := [(
    nProcessId      := 10000,
    stValidDays     := F_BACnet_ValidDays(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE),
    stFromTime      := F_BA_ToSTTime(T#0H),
    stToTime        := F_BA_ToSTTime(T#23H59M59S),
    bIssueConfirmed := FALSE,
    stEventTransitions := F_BACnet_EventTransitionBits(TRUE, TRUE, TRUE),
    stRecipient     := F_BACnet_DeviceRecipient(nDeviceInstance:=1445709)
), (
    nProcessId      := 30100,
    stValidDays     := F_BACnet_ValidDays(TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, FALSE),
    stFromTime      := F_BA_ToSTTime(T#0H),
    stToTime        := F_BA_ToSTTime(T#23H59M59S),
    bIssueConfirmed := TRUE,
    stEventTransitions := F_BACnet_EventTransitionBits(TRUE, TRUE, TRUE),
    stRecipient     := F_BACnet_EthernetRecipient(nIPAddress1:=192,168,10,200, nPort:=47808
, nNetworkNr:=444)
), (
    nProcessId      := 40100,
    stValidDays     := F_BACnet_ValidDays(TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, FALSE),
    stFromTime      := F_BA_ToSTTime(T#0H),
    stToTime        := F_BA_ToSTTime(T#23H59M59S),
    bIssueConfirmed := TRUE,
    stEventTransitions := F_BACnet_EventTransitionBits(TRUE, TRUE, TRUE),
    stRecipient     := F_BACnet_EthernetRecipient(nIPAddress1:=192,168,15,200, nPort:=47808
, nNetworkNr:=555)
)];
END_VAR
VAR
    bResult      : BOOL;
    _iObj        : I_BA_Object;
    _fbBACnetObj : POINTER TO FB_BACnet_BaseObject;
    _hRes        : HRESULT;
END_VAR
```

Implementation part

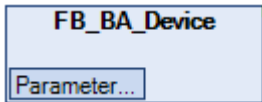
```
IF (XBA_Globals.Top.ProjectState = E_BA_ProjectState.eFirstOpCycle) THEN
    bWrite := TRUE;
END_IF
```

```
IF (bWrite) THEN
  bWrite := FALSE;
  // Iterate over all event class objects:
  _iObj := 0;
  WHILE (F_BA_IterateObjectIndex(_iObj, E_BA_ObjectType.eEventClass)) DO
    // Receive internal BACnet object from BA object:
    IF (NOT _iObj.GetBACnetObject(fbObject=>_fbBACnetObj)) THEN
      _iObj.LogMsg.Show(ADSLOG_MSGTYPE_ERROR, 'RL09', 'Failed to receive internal BACnet object!', T
RUE);
    ELSE
      // Write recipient list:
      _hRes := _fbBACnetObj^.WritePropertyRecipientList(aRecipientList);
      IF (FAILED(_hRes)) THEN
        ; // Do some error handling (An error message has already been logged here).
      END_IF
    END_IF
  END_WHILE
END_IF
```

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.1.3 FB_BA_Device



The template includes several basic functions.

The function block FB_BA_ProjectEx provides information about the project and offers information and diagnostic options. It triggers the persistent storage of data. A partial parameterization of the BACnet Device takes place via the function block FB_BA_ProjectEx.

The template FB_BA_TrendLogging contains various Trendlog objects.

The function block [FB_BA_Parameter \[► 670\]](#) is used to parameterize global parameter lists of the library [Tc3_XBA \[► 92\]](#).

Within the TimeSettings folder there are 2 possibilities to integrate the BACnet system time or the local NT system time of the TwinCAT system.

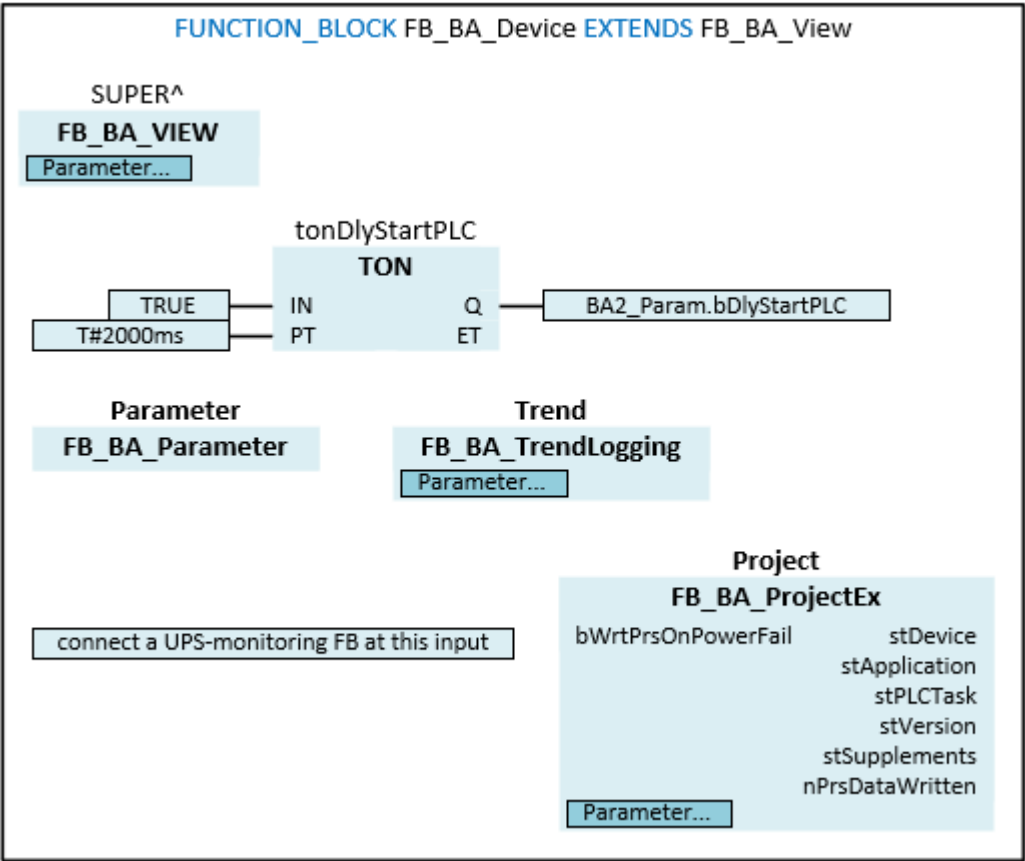


The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - [FB_BA_Object \[► 195\]](#)
 - [FB_BA_View \[► 173\]](#)

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Settings EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    Trend      : FB_BA_TrendLogging;
    Project    : FB_BA_ProjectEx;
    Parameter  : FB_BA_Parameter;
END_VAR
VAR
    tonDlyStartPLC : TON;
END_VAR
```

🔧 Inputs CONSTANT

Name	Type	Description
Trend	FB_BA_TrendLogging	Calling the Trendlog objects.
Project	FB_BA_ProjectEx	The function block provides information about the project and offers information and diagnostic options. It also stores the persistent data. It has the input <i>bWrtPrsOnPowerFail</i> , which stores the persistent data in case of an edge and is intended for linking a UPS function block. Furthermore, it is used to parameterize the object name and the object description of the BACnet Device object (server).
Parameter	FB_BA_Parameter [► 670]	Within the function block global parameter lists of the libraries <i>Tc3_XBA</i> [► 92], <i>Tc3_BACnetRev14</i> , <i>Tc3_BA2_Common</i> and <i>Tc3_BA2</i> [► 210] are parameterized.

Variables

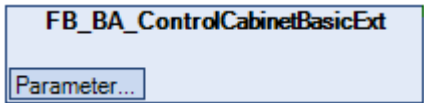
Name	Type	Description
tonDlyStartPLC	TON	The timing element sets the global variable bDlyStartPLC [▶ 820] delayed after a PLC restart.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.2 *Control cabinet*

6.1.4.2.1.3.2.1 **FB_BA_ControlCabinetBasicExt**



The template is used to control a collective error signal lamp and to acknowledge or reset alarms by means of a push button.

All alarms in the PLC are acknowledged, reset and output via the function block `FB_BA_EventObserver` used in the template.

The resulting collective alarm is used in the template `LampFault` [[▶ 716](#)] to control a collective error signal lamp. In case of a new, unacknowledged alarm or event, the signal lamp is flashing.

If only acknowledged alarms are present, the lamp is permanently on (new value message).

To acknowledge and reset an alarm, the acknowledge button must be pressed twice. If the acknowledgement and reset should be done with a keystroke, then in the GVL `BA_Param` the value of `eEvtMgmt_AckMode.eSingle` must be changed to `eEvtMgmt_AckMode.eEntire`.

The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

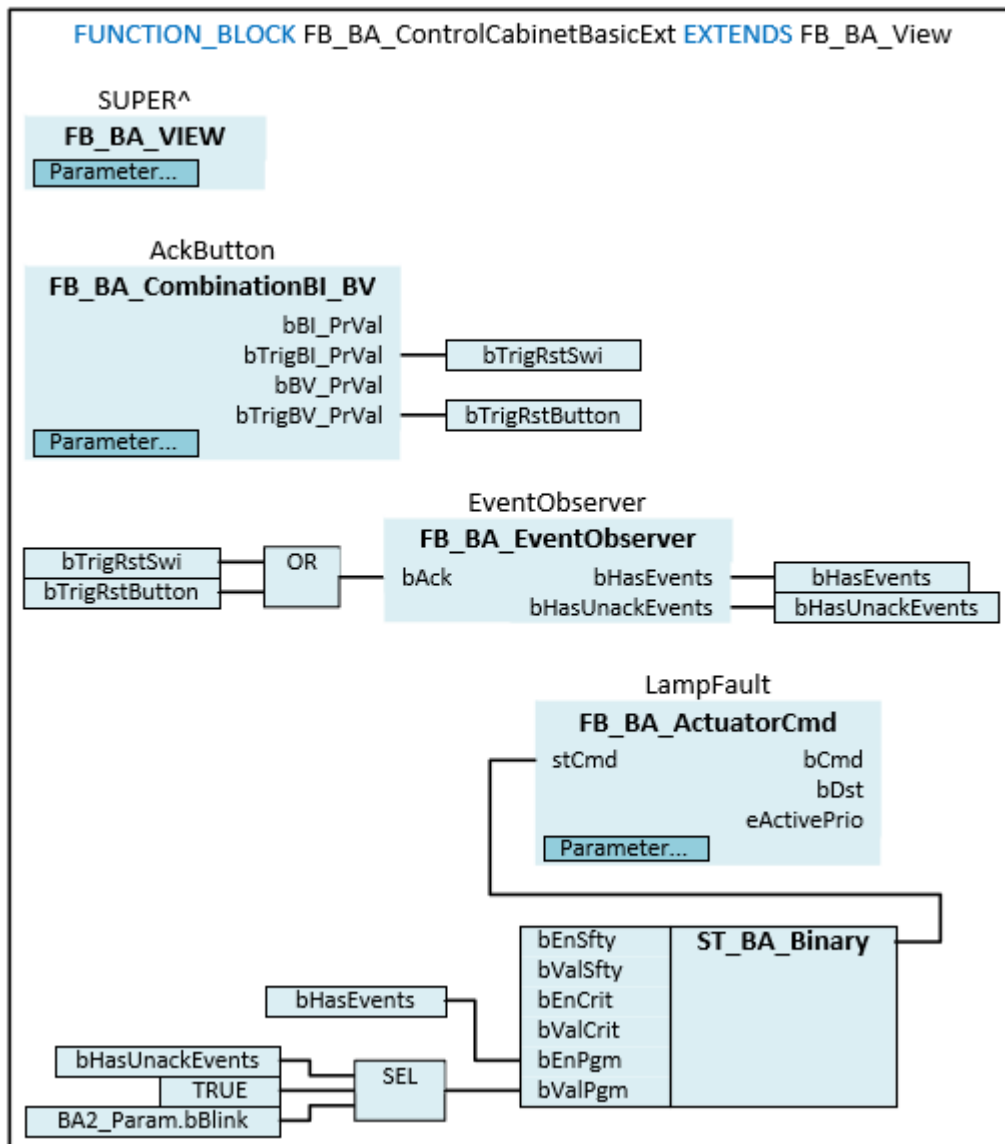
FB_BA_Base

 FB_BA_BasePublisher

 FB_BA_Object [[▶ 195](#)]

 FB_BA_View [[▶ 173](#)]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_ControlCabinetBasicExt EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  LampFault      : FB_BA_ActuatorCmd;
  AckButton      : FB_BA_CombinationBI_BV;
  EventObserver  : FB_BA_EventObserver;
END_VAR
VAR
  bHasEvent      : BOOL;
  bHasUnackEvent : BOOL;
  bTrigRstSwi    : BOOL;
  bTrigRstButton : BOOL;
END_VAR

```


🔧 VAR_INPUT CONSTANT

Name	Type	Description
LampFault	FB_BA_ActuatorCmd [▶ 716]	The template is used to control a collective error signal lamp.
AckButton	FB_BA_CombinationBI_BV	The template contains a binary input for connecting the acknowledgement button as well as a binary Value object for remote triggering of an acknowledgement from the management level.
EventObserver	FB_BA_EventObserver	The function block EventObserver realizes the evaluation of all alarms / events of the project structure (base framework) and their acknowledgement. The connection to the project structure is made by initializing the property <i>Parent</i> of the template FB_BA_EventObserver. In this template an assignment to the base object FB_BA_TopView of the project structure/ base framework is implemented at the property.

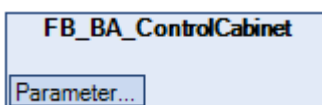
VAR

Name	Type	Description
bHasEvent	BOOL	This variable indicates that an alarm is present in the project.
bHasUnackEvent	BOOL	This variable indicates that there are unacknowledged alarms in the project.
bTrigRstSwi	BOOL	Display of the acknowledgement signal from the BV object <i>Input</i> (see FB_BA_CombinationBI_BV).
bTrigRstButton	BOOL	Display of the acknowledgement signal from the BV object <i>Value</i> (see FB_BA_CombinationBI_BV).

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.2.2 FB_BA_ControlCabinet



The template is used to collect and display control cabinet messages such as fuses, overcurrent protection devices, power recovery relays, etc. It mainly consists of 3 BI objects for displaying fuses.

The template serves as a template and should be adapted to the respective circumstances.

The base class FB_BA_ControlCabinetBasic [▶ 677] accesses the base framework. It evaluates the alarms of the automation station and a central acknowledgement can be triggered.



The initialization of the template takes place within the method FB_Init.

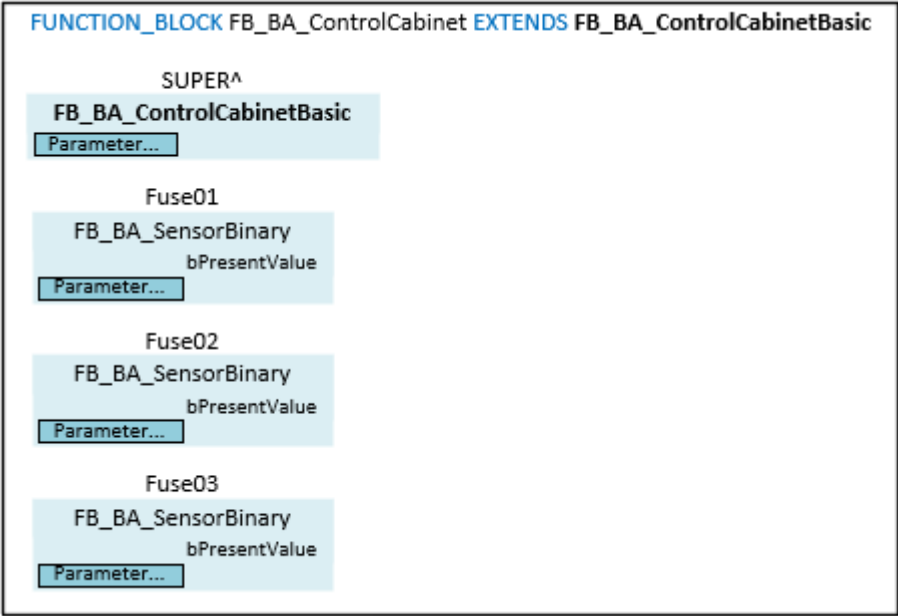
Inheritance hierarchy

```

FB_BA_Base
  FB_BA_BasePublisher
  
```

- [FB BA Object \[► 195\]](#)
- [FB BA View \[► 173\]](#)
- [FB BA ControlCabinetBasic \[► 677\]](#)

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_ControlCabinet EXTENDS FB_BA_ControlCabinetBasic
VAR_INPUT CONSTANT
    Fuse01    : FB_BA_SensorBinary;
    Fuse02    : FB_BA_SensorBinary;
    Fuse03    : FB_BA_SensorBinary;
END_VAR
```

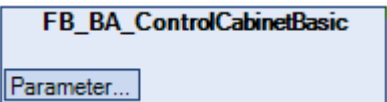
 VAR_INPUT CONSTANT

Name	Type	Description
Fuse01	FB_BA_SensorBinary [► 790]	The template contains a binary input for connecting a fuse message.
Fuse02	FB_BA_SensorBinary [► 790]	The template contains a binary input for connecting a fuse message.
Fuse03	FB_BA_SensorBinary [► 790]	The template contains a binary input for connecting a fuse message.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.2.3 FB_BA_ControlCabinetBasic



The template is used to control a collective error signal lamp, to acknowledge or reset alarms by means of a push button and to reset hardware in the control cabinet by outputting a momentary pulse.

All alarms in the PLC are acknowledged, reset and output via the function block `FB_BA_EventObserver` used in the template.

The resulting collective alarm is used in the template `LampFault` [► 716] to control a collective error signal lamp. In case of a new, unacknowledged alarm or event, the signal lamp is flashing.

If only acknowledged alarms are present, the lamp is permanently on (new value message).

A momentary pulse for resetting hardware, is output via the function block `RstHw` [► 716]. This is extended to two seconds by the timing element `tpResetHw`. In this way, relay switching within the control cabinet is reliably acknowledged by a sufficiently long pulse.

To acknowledge and reset an alarm, the acknowledge button must be pressed twice. If the acknowledgement and reset should be done with a keystroke, then in the GVL `BA_Param` the value of `eEvtMgmt_AckMode.eSingle` must be changed to `eEvtMgmt_AckMode.eEntire`.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

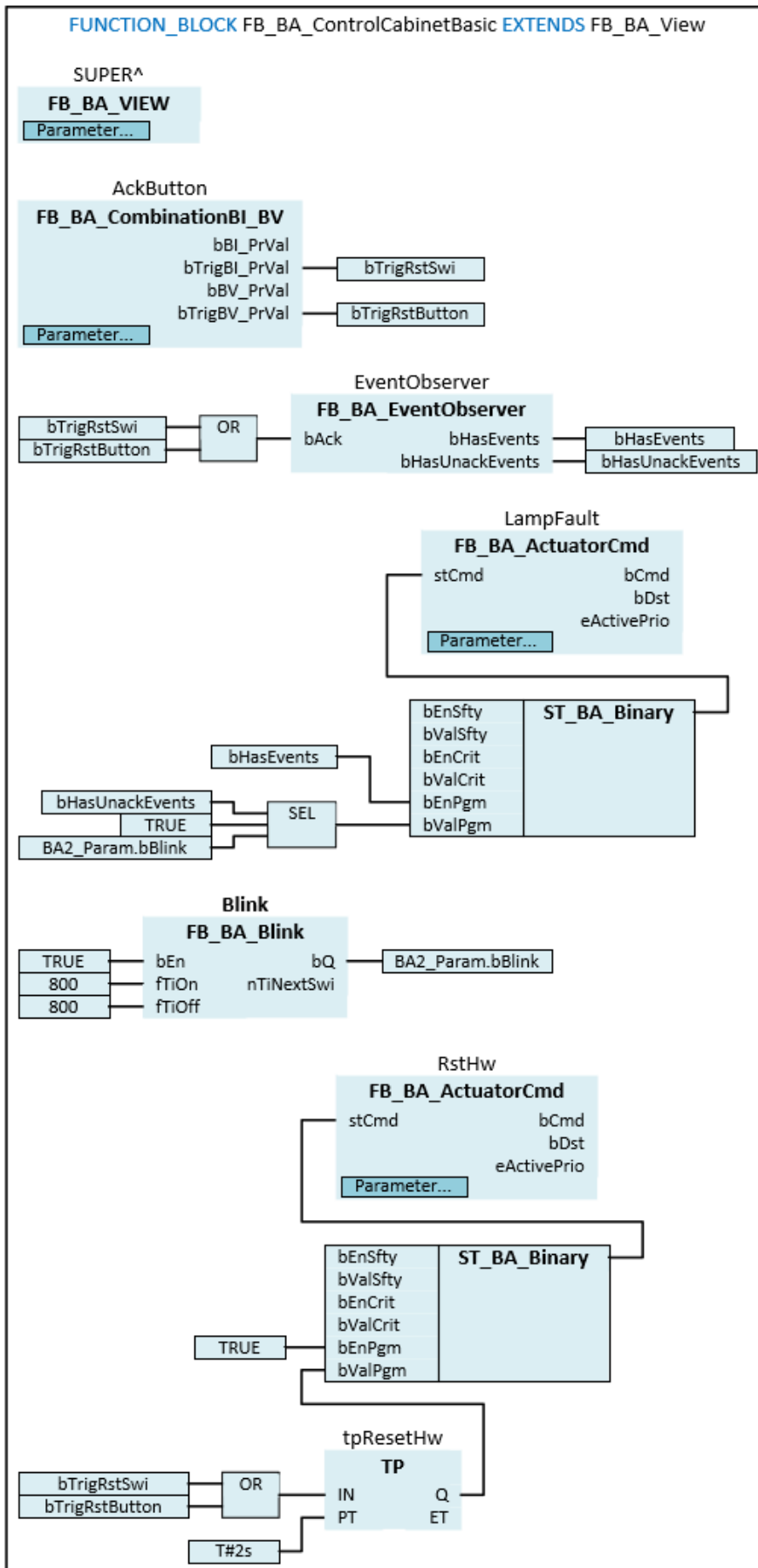
`FB_BA_Base`

`FB_BA_BasePublisher`

`FB_BA_Object` [► 195]

`FB_BA_View` [► 173]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_ControlCabinetBasic EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    LampFault      : FB_BA_ActuatorCmd;
    AckButton      : FB_BA_CombinationBI_BV;
    RstHw          : FB_BA_ActuatorCmd;
    EventObserver  : FB_BA_EventObserver;
END_VAR
VAR
    bHasEvent      : BOOL;
    bHasUnackEvent : BOOL;
    Blink          : FB_BA_Blink;
    tpResetHw      : TP;
    bTrigRstSwi    : BOOL;
    bTrigRstButton : BOOL;
END_VAR

```

VAR_INPUT CONSTANT

Name	Type	Description
LampFault	FB_BA_ActuatorCmd [► 716]	The template is used to control a collective error signal lamp.
AckButton	FB_BA_CombinationBI_BV	The template contains a binary input for connecting the acknowledgement button as well as a binary Value object for remote triggering of an acknowledgement from the management level.
RstHw	FB_BA_ActuatorCmd [► 716]	The template can be used to wipe in relay circuits (e.g. frost protection relay).
EventObserver	FB_BA_EventObserver	The function block EventObserver realizes the evaluation of all alarms / events of the project structure (base framework) and their acknowledgement. The connection to the project structure is made by initializing the property <i>Parent</i> of the template FB_BA_EventObserver. In this template an assignment to the base object FB_BA_TopView of the project structure/ base framework is implemented at the property.

VAR

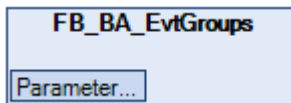
Name	Type	Description
bHasEvent	BOOL	This variable indicates that an alarm is present in the project.
bHasUnackEvent	BOOL	This variable indicates that there are unacknowledged alarms in the project.
Blink	FB_BA_Blink	The function block generates a blink pulse. This blink pulse is written to the global variable <i>BA2_Param.bBlink</i> .
tpResetHw	TP	The timing element extends the acknowledgement pulse for interfacing relay circuits (e.g. frost protection relay).
bTrigRstSwi	BOOL	Display of the acknowledgement signal from the BV object <i>Input</i> (see FB_BA_CombinationBI_BV).
bTrigRstButton	BOOL	Display of the acknowledgement signal from the BV object <i>Value</i> (see FB_BA_CombinationBI_BV).

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.3 Event classes

6.1.4.2.1.3.3.1 FB_BA_EvtGroups



The template provides nine event categories with the corresponding notification classes/event classes for intrinsic reporting. The intrinsic reporting of the BACnet objects within the templates is adapted to the use of these notification classes.

The notification classes as well as the alarm-capable objects get their notification class number from the global GVL [EC_ID](#) [[▶ 819](#)].



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

FB_BA_Base

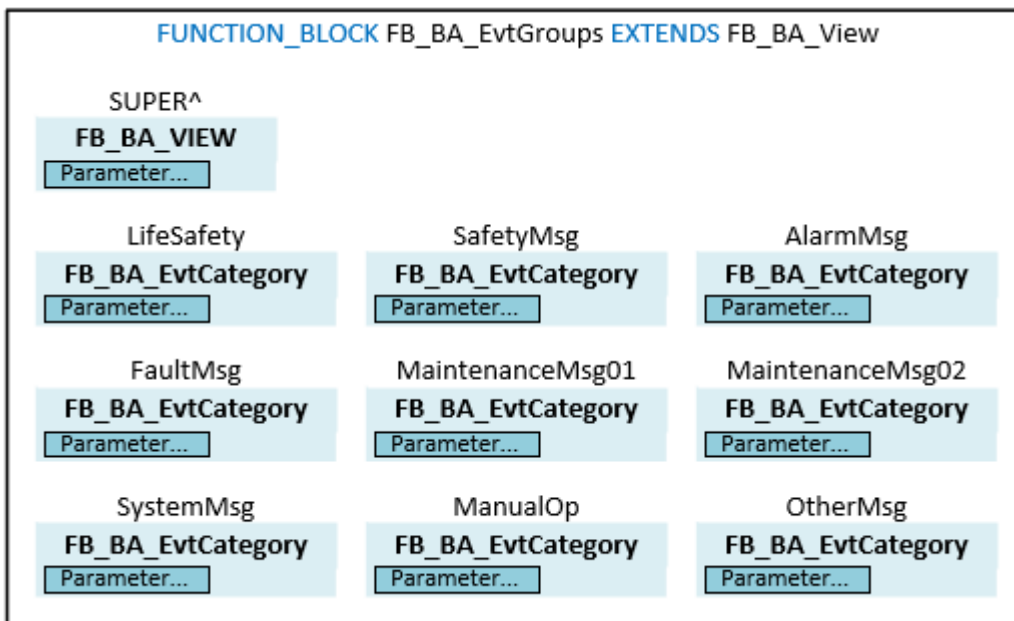
FB_BA_BasePublisher

FB_BA_Object [[▶ 195](#)]

FB_BA_View [[▶ 173](#)]

FB_BA_ControlCabinetBasic [[▶ 677](#)]

Block diagram



Intrinsic reporting

In case of intrinsic reporting, several predefined events can be reported to the recipients entered in the Notification Class Object / Event Class Object.

By means of intrinsic reporting, several conditions can be defined for a BACnet device, which are necessary to generate alarms or event messages and to send them to the specified recipients (e.g. operating devices). In case of intrinsic reporting, a change in the value of properties triggers an event message.

Intrinsic reporting is used, for example, to monitor binary I/O objects for status changes or analog I/O objects with the properties "High_Limit", "Low_Limit" and "Deadband" for limit violations.

Explanation of properties for intrinsic reporting

Event messages active (Event_Enable)

The property Event messages active (Event_Enable) is available in many event generating BACnet objects.

Three bits (flags = 0 or 1) can be used to individually set the disabling or enabling of message generation for the three events (TO-OFFNORMAL, TO-FAULT and TO-NORMAL).

Acknowledgement required (Ack_Required)

The Ack_Required property of the Notification Class Object determines whether events generated by TO-OFFNORMAL, TO-FAULT, and TO-NORMAL must be acknowledged.

Three bits (flags = 0 or 1) can be used in this PROPERTY to individually define which acknowledgements are required for the three events (TO-OFFNORMAL, TO-FAULT and TO-NORMAL).

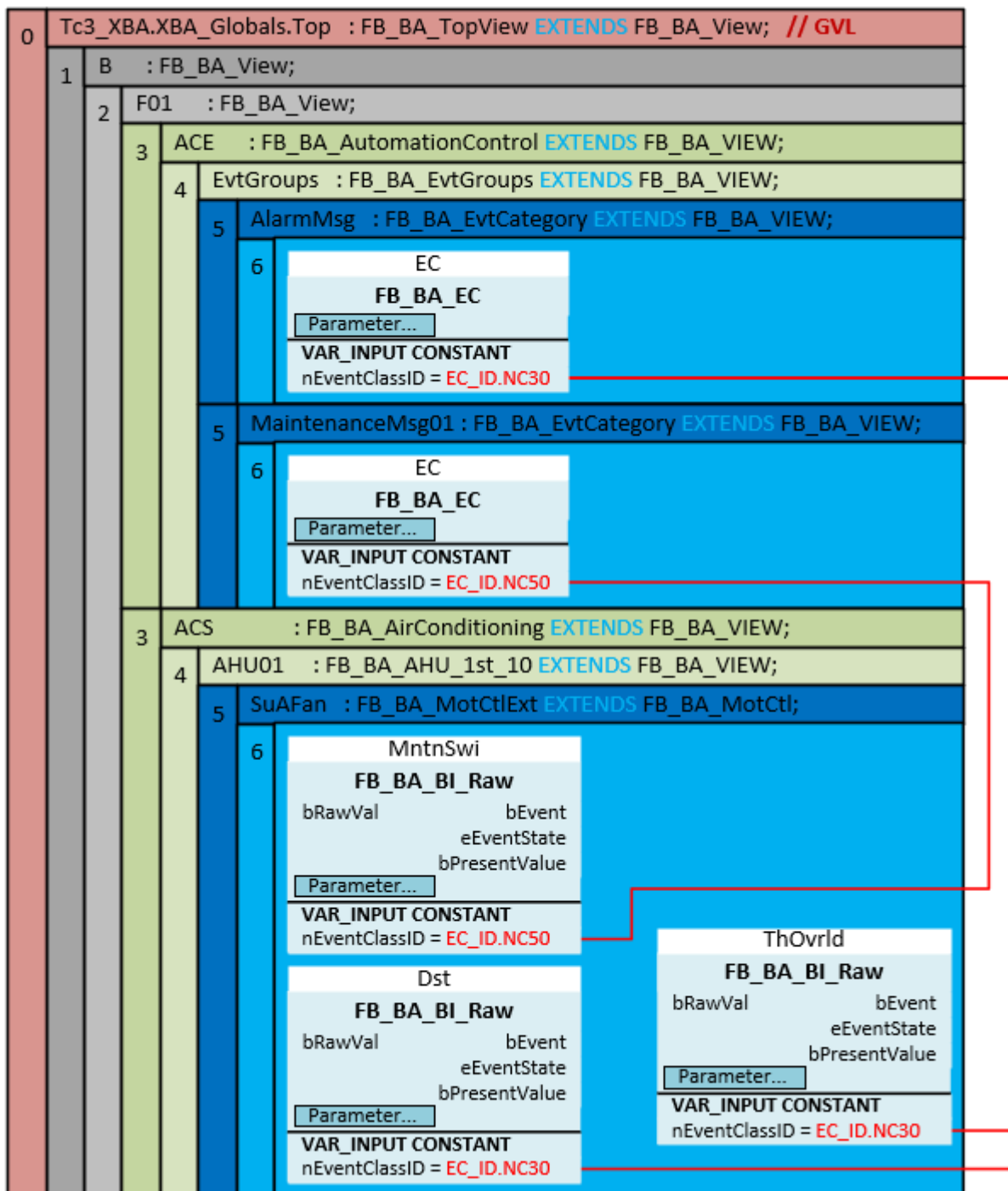
Acked_Transitions (Type BACnetEventTransitionBits)

The Acked_Transitions property must transfer three flags that separately indicate the acknowledgement status for TO_OFFNORMAL, TO_FAULT, and TO_NORMAL events. Each flag must have the value TRUE if no event of this type has yet occurred for the object.

The Ack_Required property of the referenced Notification Class determines whether acknowledgement is required.

Sample Assignment of alarm-capable objects and notification class objects

The diagram shows how the alarm-capable BI objects are assigned to the notification class objects FB_BA_EC based on the event class numbers EC_ID.NC30 and EC_ID.NC50.



Notification class matrix

This matrix is based on the AMEV2017.

Event category	Meaning	Priority	Notification class	NC object EC object	Example
Hazard notification (Life Safety)	Danger to life	00 - 29	EC_ID.N C10	LifeSafety.EC	Fire alarm, robbery
Hazard notification (Property Safety)	Safety message	30 - 59	EC_ID.N C20	SafetyMsg.EC	Burglary, unauthorized entry
Alarm message	Message indicates system failure or requires immediate intervention.	60 - 89	EC_ID.N C30	AlarmMsg.EC	Safety temperature limiter (STB), safety pressure limiter (SDB), excess temperature of water heating (WWB), safety valves, main pumps, V-belt monitors, frequency converters, refrigeration systems, voltage failure, etc.
Fault message	Message indicates abnormal operating state.	90 - 119	EC_ID.N C40	FaultMsg.EC	Temperature monitor (TW), pressure monitor (DW), temperature monitoring of heat exchanger (WT) and WWB, motor protection, elevator collective error message, mains pressures, etc.
Maintenance message	Indication of maintenance activity or similar.	120 - 149	EC_ID.N C50	MaintenanceMsg 01.EC	Operating hours, tank level, repair switch, etc.
Maintenance message	Indication of maintenance activity or similar.	120 - 149	EC_ID.N C51	MaintenanceMsg 02.EC	Filter end reached, filter dirty, etc.
System message	Fault message from the GA system.	150 - 219	EC_ID.N C60	SystemMsg.EC	Device malfunction, battery message, communication interruption, etc.
Manual message	Manual intervention	220	EC_ID.N C70	ManualOp.EC	Manual intervention
Subject to confirmation	Other messages	221 - 255	EC_ID.N C80	OtherMsg.EC	Operating state change, operation modes, trend memory full, etc.

Illustration

```

FUNCTION_BLOCK FB_BA_EvtGroups EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    LifeSafety      : FB_BA_EvtCategory;
    SafetyMsg       : FB_BA_EvtCategory;
    AlarmMsg        : FB_BA_EvtCategory;
    FaultMsg        : FB_BA_EvtCategory;
    MaintenanceMsg01 : FB_BA_EvtCategory;
    MaintenanceMsg02 : FB_BA_EvtCategory;
    SystemMsg       : FB_BA_EvtCategory;
    ManualOp        : FB_BA_EvtCategory;
    OtherMsg        : FB_BA_EvtCategory;
END_VAR

```

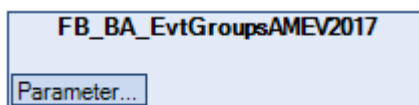
🚩 VAR_INPUT CONSTANT

Name	Type	Description
LifeSafety	FB_BA_EvtCategory [► 750]	The template calls the "Danger to life" event category.
SafetyMsg	FB_BA_EvtCategory [► 750]	The template calls the "Danger message, safety message" event category.
AlarmMsg	FB_BA_EvtCategory [► 750]	The template calls the "Alarm message" event category.
FaultMsg	FB_BA_EvtCategory [► 750]	The template calls the "Fault message" event category.
MaintenanceMsg01	FB_BA_EvtCategory [► 750]	The template calls the event category "Maintenance message".
MaintenanceMsg02	FB_BA_EvtCategory [► 750]	The template calls the event category "Maintenance message".
SystemMsg	FB_BA_EvtCategory [► 750]	The template calls the event category "System message".
ManualOp	FB_BA_EvtCategory [► 750]	The template calls the event category "Manual intervention".
OtherMsg	FB_BA_EvtCategory [► 750]	The template calls the event category "Other messages".

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.3.2 FB_BA_EvtGroupsAMEV2017



The template provides nine event categories with the corresponding notification classes/event classes for intrinsic reporting. The intrinsic reporting of the BACnet objects within the templates is adapted to the use of these notification classes.

The notification classes as well as the alarm-capable objects get their notification class number from the global GVL [EC_ID](#) [► 819].



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

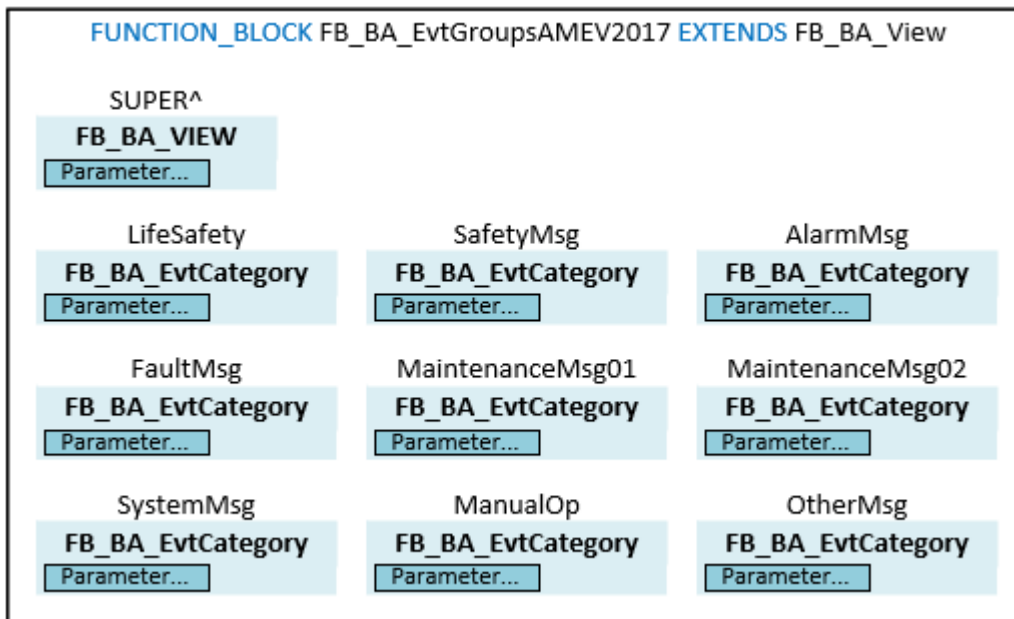
FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object](#) [► 195]

[FB_BA_View](#) [► 173]

[FB_BA_ControlCabinetBasic](#) [► 677]

Block diagram**Intrinsic reporting**

In case of intrinsic reporting, several predefined events can be reported to the recipients entered in the Notification Class Object / Event Class Object.

By means of intrinsic reporting, several conditions can be defined for a BACnet device, which are necessary to generate alarms or event messages and to send them to the specified recipients (e.g. operating devices). In case of intrinsic reporting, a change in the value of properties triggers an event message.

Intrinsic reporting is used, for example, to monitor binary I/O objects for status changes or analog I/O objects with the properties "High_Limit", "Low_Limit" and "Deadband" for limit violations.

Explanation of properties for intrinsic reporting**Event messages active (Event_Enable)**

The property Event messages active (Event_Enable) is available in many event generating BACnet objects.

Three bits (flags = 0 or 1) can be used to individually set the disabling or enabling of message generation for the three events (TO-OFFNORMAL, TO-FAULT and TO-NORMAL).

Acknowledgement required (Ack_Required)

The Ack_Required property of the Notification Class Object determines whether events generated by TO-OFFNORMAL, TO-FAULT, and TO-NORMAL must be acknowledged.

Three bits (flags = 0 or 1) can be used in this PROPERTY to individually define which acknowledgements are required for the three events (TO-OFFNORMAL, TO-FAULT and TO-NORMAL).

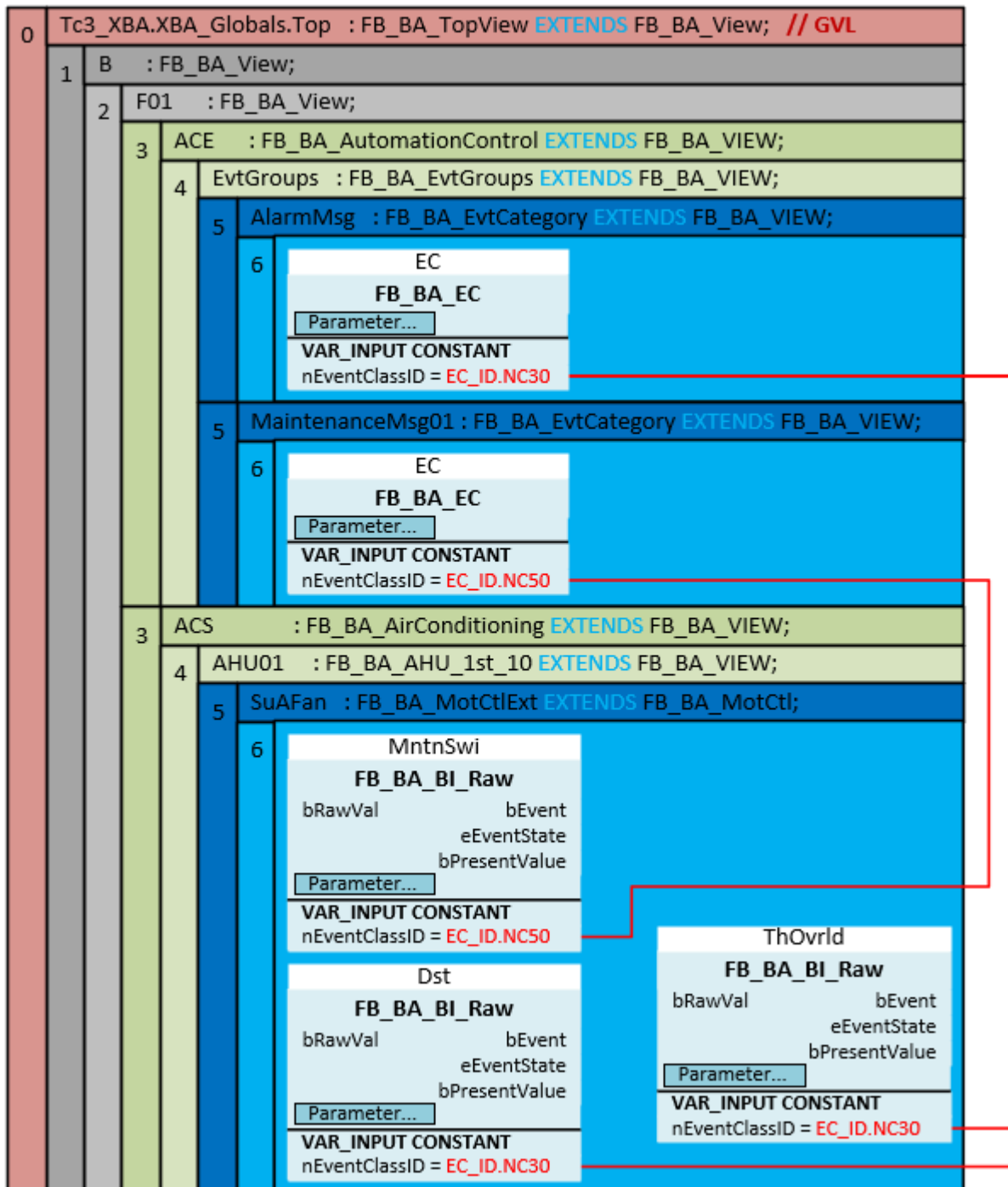
Acked_Transitions (Type BACnetEventTransitionBits)

The Acked_Transitions property must transfer three flags that separately indicate the acknowledgement status for TO_OFFNORMAL, TO_FAULT, and TO_NORMAL events. Each flag must have the value TRUE if no event of this type has yet occurred for the object.

The Ack_Required property of the referenced Notification Class determines whether acknowledgement is required.

Sample Assignment of alarm-capable objects and notification class objects

The diagram shows how the alarm-capable BI objects are assigned to the notification class objects FB_BA_EC based on the event class numbers EC_ID.NC30 and EC_ID.NC50.



Notification class matrix

This matrix is based on the AMEV2017.

Event category	Meaning	Priority	Notification class	NC object EC object	Example
Hazard notification (Life Safety)	Danger to life	00 - 29	EC_ID.N C10	LifeSafety.EC	Fire alarm, robbery
Hazard notification (Property Safety)	Safety message	30 - 59	EC_ID.N C20	SafetyMsg.EC	Burglary, unauthorized entry
Alarm message	Message indicates system failure or requires immediate intervention.	60 - 89	EC_ID.N C30	AlarmMsg.EC	Safety temperature limiter (STB), safety pressure limiter (SDB), excess temperature of water heating (WWB), safety valves, main pumps, V-belt monitors, frequency converters, refrigeration systems, voltage failure, etc.
Fault message	Message indicates abnormal operating state.	90 - 119	EC_ID.N C40	FaultMsg.EC	Temperature monitor (TW), pressure monitor (DW), temperature monitoring of heat exchanger (WT) and WWB, motor protection, elevator collective error message, mains pressures, etc.
Maintenance message	Indication of maintenance activity or similar.	120 - 149	EC_ID.N C50	MaintenanceMsg 01.EC	Operating hours, tank level, repair switch, etc.
Maintenance message	Indication of maintenance activity or similar.	120 - 149	EC_ID.N C51	MaintenanceMsg 02.EC	Filter end reached, filter dirty, etc.
System message	Fault message from the GA system.	150 - 219	EC_ID.N C60	SystemMsg.EC	Device malfunction, battery message, communication interruption, etc.
Manual message	Manual intervention	220	EC_ID.N C70	ManualOp.EC	Manual intervention
Subject to confirmation	Other messages	221 - 255	EC_ID.N C80	OtherMsg.EC	Operating state change, operation modes, trend memory full, etc.

Illustration

```

FUNCTION_BLOCK FB_BA_EvtGroups EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    LifeSafety      : FB_BA_EvtCategory;
    SafetyMsg       : FB_BA_EvtCategory;
    AlarmMsg        : FB_BA_EvtCategory;
    FaultMsg        : FB_BA_EvtCategory;
    MaintenanceMsg01 : FB_BA_EvtCategory;
    MaintenanceMsg02 : FB_BA_EvtCategory;
    SystemMsg       : FB_BA_EvtCategory;
    ManualOp        : FB_BA_EvtCategory;
    OtherMsg        : FB_BA_EvtCategory;
END_VAR

```

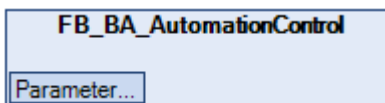
VAR_INPUT CONSTANT

Name	Type	Description
LifeSafety	FB_BA_EvtCategory [► 750]	The template calls the "Danger to life" event category.
SafetyMsg	FB_BA_EvtCategory [► 750]	The template calls the "Danger message, safety message" event category.
AlarmMsg	FB_BA_EvtCategory [► 750]	The template calls the "Alarm message" event category.
FaultMsg	FB_BA_EvtCategory [► 750]	The template calls the "Fault message" event category.
MaintenanceMsg01	FB_BA_EvtCategory [► 750]	The template calls the event category "Maintenance message".
MaintenanceMsg02	FB_BA_EvtCategory [► 750]	The template calls the event category "Maintenance message".
SystemMsg	FB_BA_EvtCategory [► 750]	The template calls the event category "System message".
ManualOp	FB_BA_EvtCategory [► 750]	The template calls the event category "Manual intervention".
OtherMsg	FB_BA_EvtCategory [► 750]	The template calls the event category "Other messages".

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.4 *FB_BA_AutomationControl*



The template contains basic functions and project information relevant for an automation station.

In the sub-template [FB_BA_Device](#) [► 672] information about the project and diagnostic options are offered. A parameterization of the BACnet device and general BACnet parameters takes place.

Below [FB_BA_Device](#) there are two possibilities to integrate the BACnet system time or the local NT system time of the TwinCAT system.

The sub-template [FB_BA_EvtGroupsAMEV2017](#) [► 685] provides event classes (notification classes) for the intrinsic reporting.

The sub-template [FB_BA_ControlCabinet](#) [► 676] is used to collect and display control cabinet messages. An evaluation of the alarms of the automation station takes place and a central acknowledgement can be triggered.



The initialization of the template takes place within the method [FB_Init](#).

Inheritance hierarchy

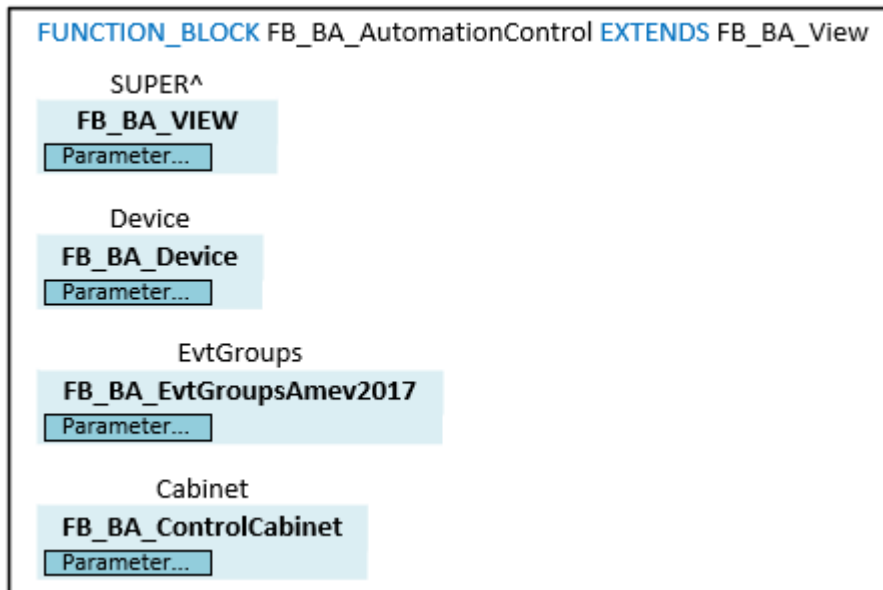
[FB_BA_Base](#)

[FB_BA_BasePublisher](#)

[FB_BA_Object](#) [► 195]

[FB_BA_View](#) [► 173]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_AutomationControl EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    Device          : FB_BA_Device;
    EvtGroups       : FB_BA_EvtGroupsAMEV2017;
    Cabinet         : FB_BA_ControlCabinet;
END_VAR

```

 VAR_INPUT CONSTANT

Name	Type	Description
Device	FB_BA_Device [► 672]	Information about the project and diagnostic options are offered in the template. A parameterization of the BACnet device and general BACnet parameters takes place. Below FB_BA_Device there are two possibilities to integrate the BACnet system time or the local NT system time of the TwinCAT system.
EvtGroups	FB_BA_EvtGroupsAMEV2017 [► 685]	The template provides event classes (notification classes) for intrinsic reporting.
Cabinet	FB_BA_ControlCabinet [► 676]	The template is used to collect and display control cabinet messages. An evaluation of the alarms of the automation station takes place and a central acknowledgement can be triggered.

Requirements

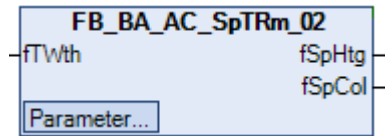
Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4 AirConditioning

6.1.4.2.1.4.1 Setpoint

6.1.4.2.1.4.1.1 Temperature

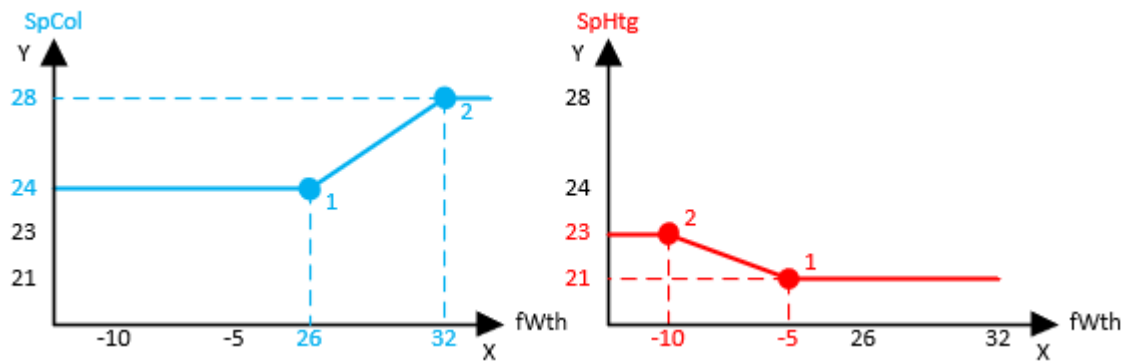
6.1.4.2.1.4.1.1.1 FB_BA_AC_SpTRm_02



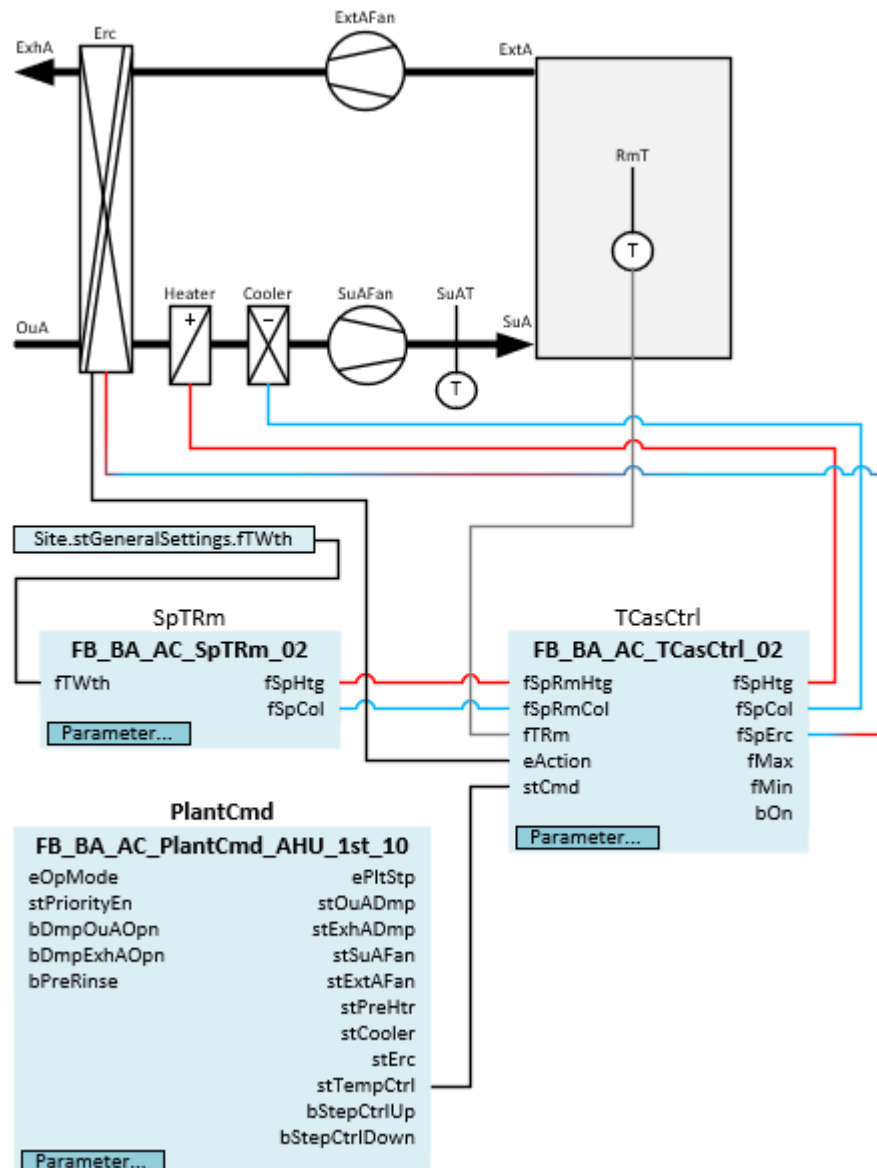
The template is a setpoint program for an air conditioning system.

It serves as a setpoint program for an exhaust air/supply air cascade with a separate room temperature setpoint for heating and cooling mode, including summer and winter compensation.

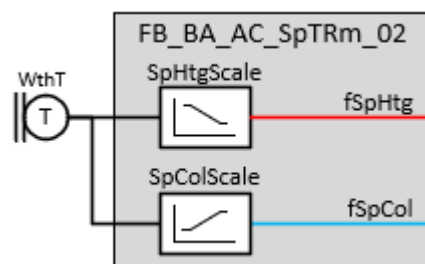
There are two basic room temperature setpoints ($SpHtgY1$, $SpColY1$), with an energy-neutral zone between the lower setpoint (heating mode) and the upper setpoint (cooling mode). An outside temperature-dependent offset of the basic room temperature setpoints is realized via two curve functions (summer/winter compensation).



Principle diagram01



Principle diagram02



The initialization of the template takes place within the method `FB_Init`.

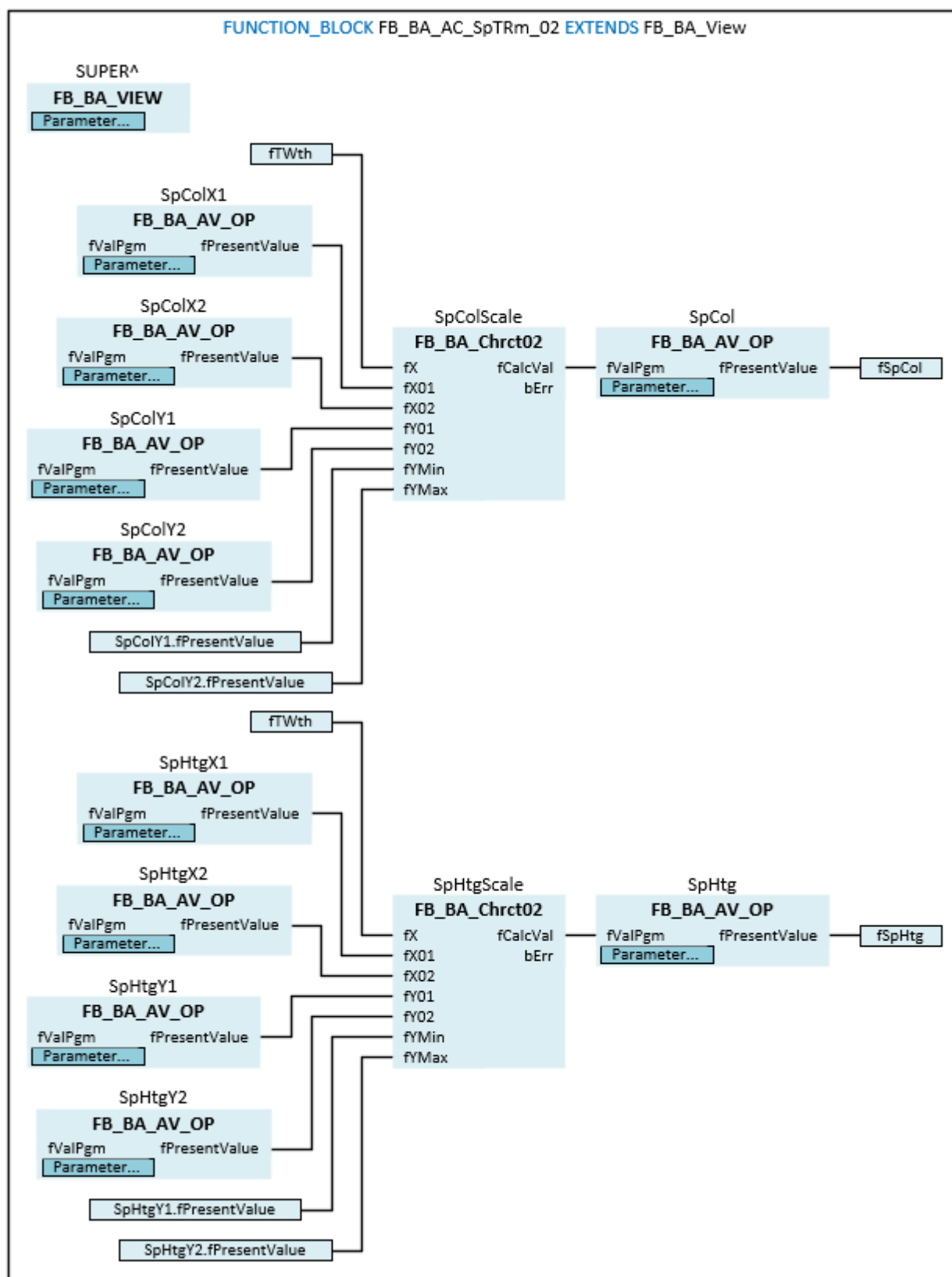
Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

Block diagram



```
FUNCTION_BLOCK FB_BA_AC_SpTRm_02 EXTENDS FB_BA_View
VAR_INPUT
    fTWth          : REAL;
END VAR
```

```

VAR_OUTPUT
    fSpHtg      : REAL;
    fSpCol      : REAL;
END_VAR
VAR_INPUT CONSTANT
    SpColX1     : FB_BA_AV_Op;
    SpColX2     : FB_BA_AV_Op;
    SpColY1     : FB_BA_AV_Op;
    SpColY2     : FB_BA_AV_Op;

    SpHtgX1     : FB_BA_AV_Op;
    SpHtgX2     : FB_BA_AV_Op;
    SpHtgY1     : FB_BA_AV_Op;
    SpHtgY2     : FB_BA_AV_Op;

    SpHtg       : FB_BA_AV_Op;
    SpCol       : FB_BA_AV_Op;
END_VAR
VAR
    SpColScale  : FB_BA_Chrc02;
    SpHtgScale  : FB_BA_Chrc02;
END_VAR

```

Inputs CONSTANT

Name	Type	Description
fTWth	REAL	Input value of the outside temperature.

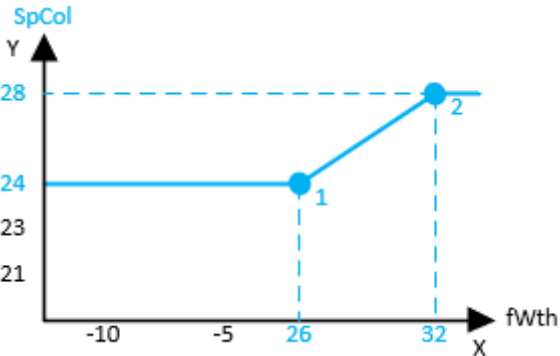
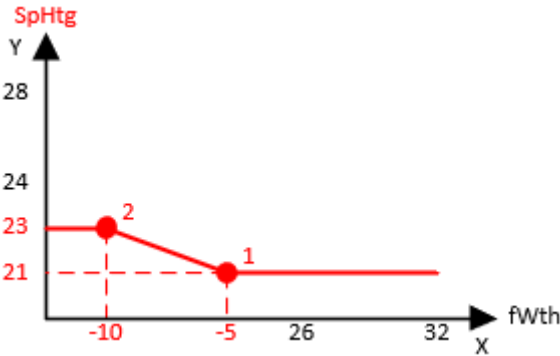
Outputs

Name	Type	Description
fSpHtg	REAL	Heating setpoint
fSpCol	REAL	Cooling setpoint

Inputs CONSTANT

Name	Type	Description
SpColX1	FB_BA_AV_Op  136	Input of the value for interpolation point X1 of the summer compensation for the cooling setpoint.
SpColX2	FB_BA_AV_Op  136	Input of the value for interpolation point X2 of the summer compensation for the cooling setpoint.
SpColY1	FB_BA_AV_Op  136	Input of the value for interpolation point Y1 of the summer compensation for the cooling setpoint. <i>SpColY1</i> is the base cooling setpoint.
SpColY2	FB_BA_AV_Op  136	Input of the value for interpolation point Y2 of the summer compensation for the cooling setpoint.
SpHtgX1	FB_BA_AV_Op  136	Input of the value for interpolation point X1 of the winter compensation for the heating setpoint.
SpHtgX2	FB_BA_AV_Op  136	Input of the value for interpolation point X2 of the winter compensation for the heating setpoint.
SpHtgY1	FB_BA_AV_Op  136	Input of the value for interpolation point Y1 of the winter compensation for the heating setpoint. <i>SpHtgY1</i> is the base heating setpoint.
SpHtgY2	FB_BA_AV_Op  136	Input of the value for interpolation point Y2 of the winter compensation for the heating setpoint.
SpHtg	FB_BA_AV_Op  136	Analog value object for displaying the calculated room temperature setpoint heating.
SpCol	FB_BA_AV_Op  136	Analog value object for displaying the calculated room temperature setpoint cooling.

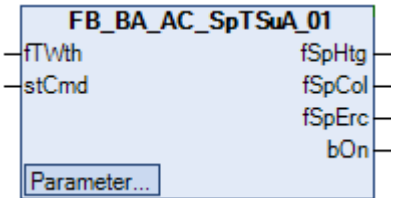
Variables

Name	Type	Description
SpColScale	FB_BA_Chrc02 [► 394]	The function block calculates the characteristic cooling setpoint curve (summer compensation) for the current room temperature as a function of the outside temperature. 
SpHtgScale	FB_BA_Chrc02 [► 394]	The function block calculates the characteristic heating setpoint curve (winter compensation) for the current room temperature as a function of the outside temperature. 

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.1.1.2 FB_BA_AC_SpTSuA_01

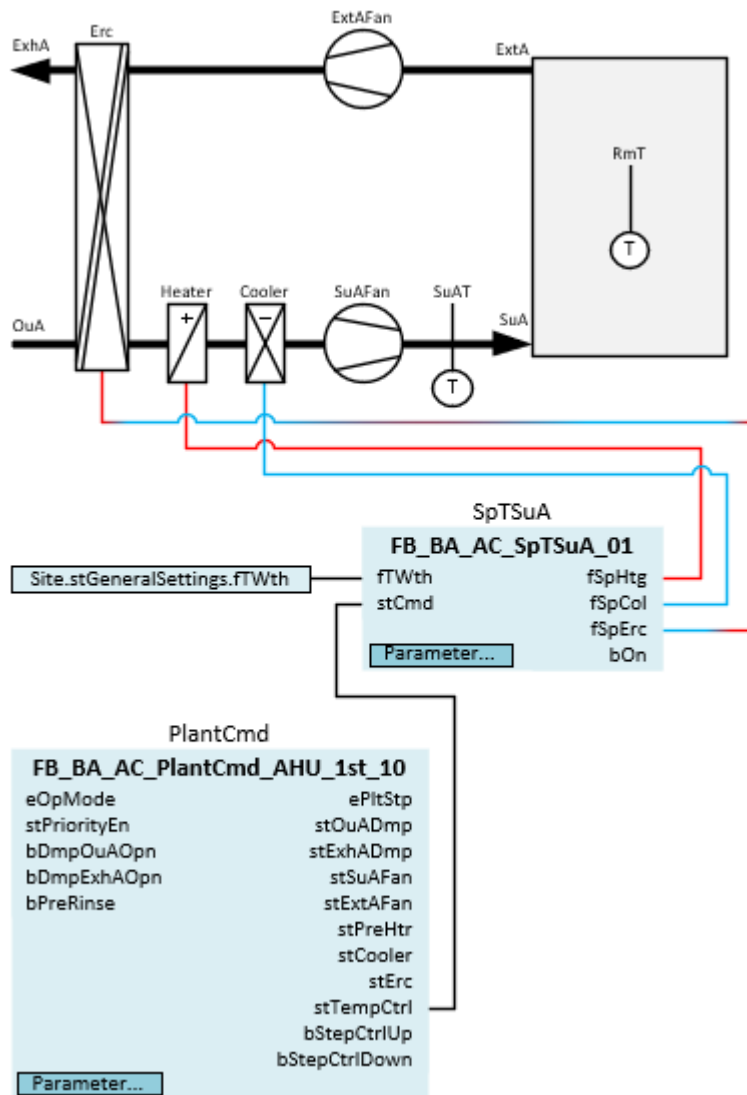


The template is used to determine the supply air temperature setpoint of an air conditioning system. The setpoint linked to the variables *fSpHtg*, *fSpCol* and *fSpErc* is transferred to the supply air controllers of the heater, cooler and energy recovery units.

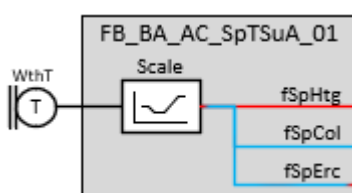
The supply air temperature setpoint is determined by the function block [FB_BA_Scale_04_Y2_3 \[► 733\]](#) depending on the outside temperature.

The supply air temperature sequence control is enabled with the variable *bOn* after evaluation of the control structure *stCmd*.

Principle diagram 01



Principle diagram 02



i

The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

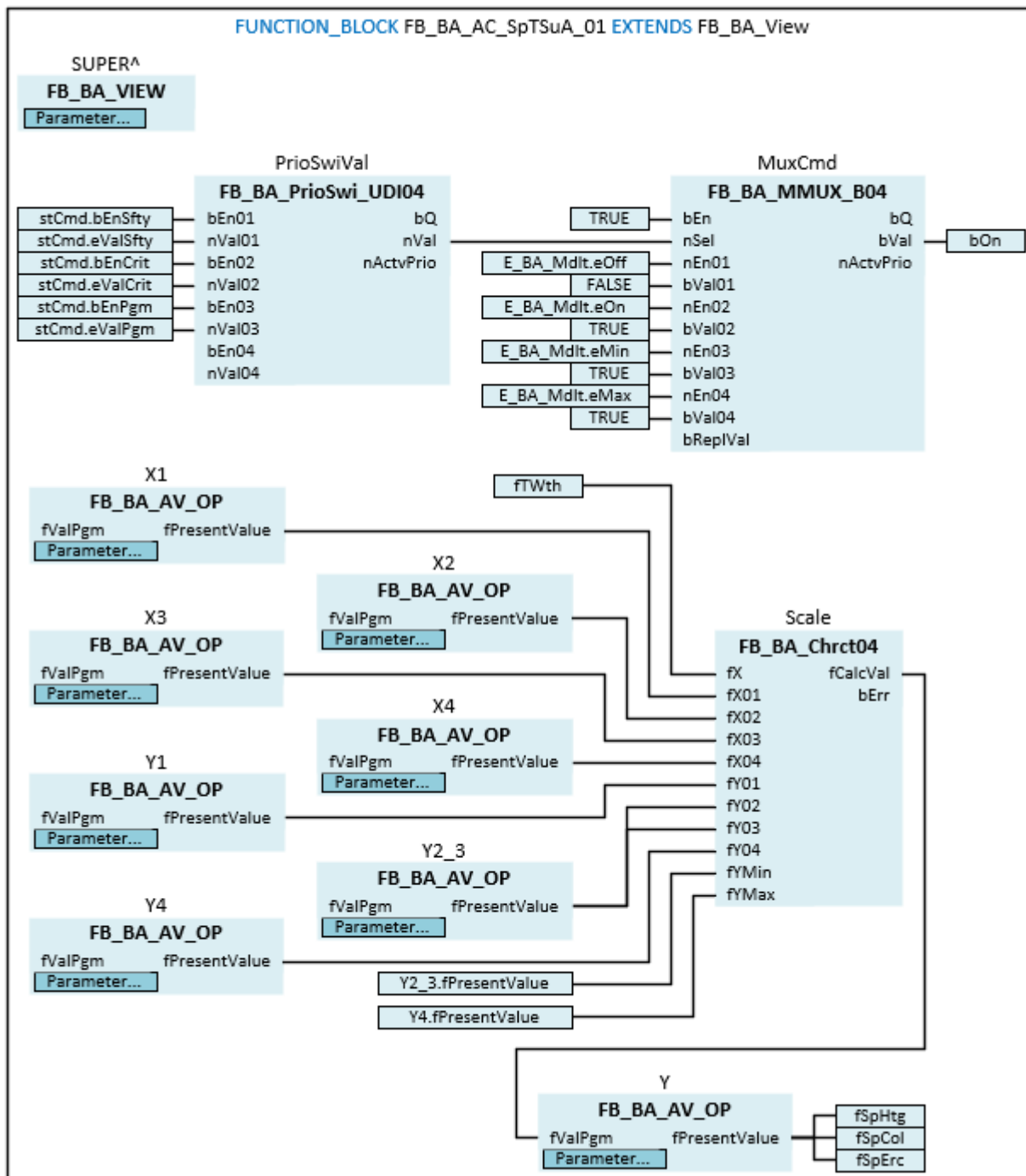
FB BA Base

FB_BA_BasePublisher

FB BA Object [► 195]

FB BA View [► 173]

Block diagram



Illustration

FUNCTION_BLOCK FB_BA_AC_SpTSuA_01 EXTENDS FB_BA_View

VAR_INPUT

fTWth : REAL;
stCmd : ST_BA_MdlT;

END_VAR

VAR_OUTPUT

fSpHtg : REAL;
fSpCol : REAL;
fSpErc : REAL;
bOn : BOOL;

END_VAR

VAR

PrioSwiVal : FB_BA_PrioSwi_UDI04;
MuxCmd : FB_BA_MMUX_B04;

END_VAR

🚩 Inputs CONSTANT

Name	Type	Description
fTWth	REAL	Input value of the outside temperature.
stCmd	ST_BA_MdlT [► 221]	The enables and modulation commands are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority determines the switching command at the binary output object <i>Cmd</i> and the control command at the analog output object <i>MdlT</i> .

🚩 Outputs

Name	Type	Description
fSpHtg	REAL	Heating setpoint
fSpCol	REAL	Cooling setpoint
fSpErc	REAL	Calculated setpoint of the supply air temperature. for energy recovery.
bOn	BOOL	The output shows the resulting result of the multiplexer <i>MuxCmd</i> . This is further processed within the ventilation system and is used to enable the supply air temperature sequence control.

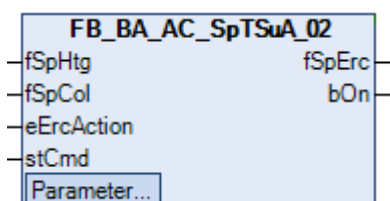
Variables

Name	Type	Description
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [► 369]	The priority switch <i>PrioSwiVal</i> [► 369] uses the command structure <i>stCmd</i> to determine the modulation command for the multiplexers <i>MuxCmd</i> and <i>MuxMdlT</i> .
MuxCmd	FB_BA_MMUX_B04 [► 364]	The multiplexer <i>MuxCmd</i> [► 364] determines the current switch value from the command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the binary output object <i>bOn</i> .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.1.1.3 FB_BA_AC_SpTSuA_02

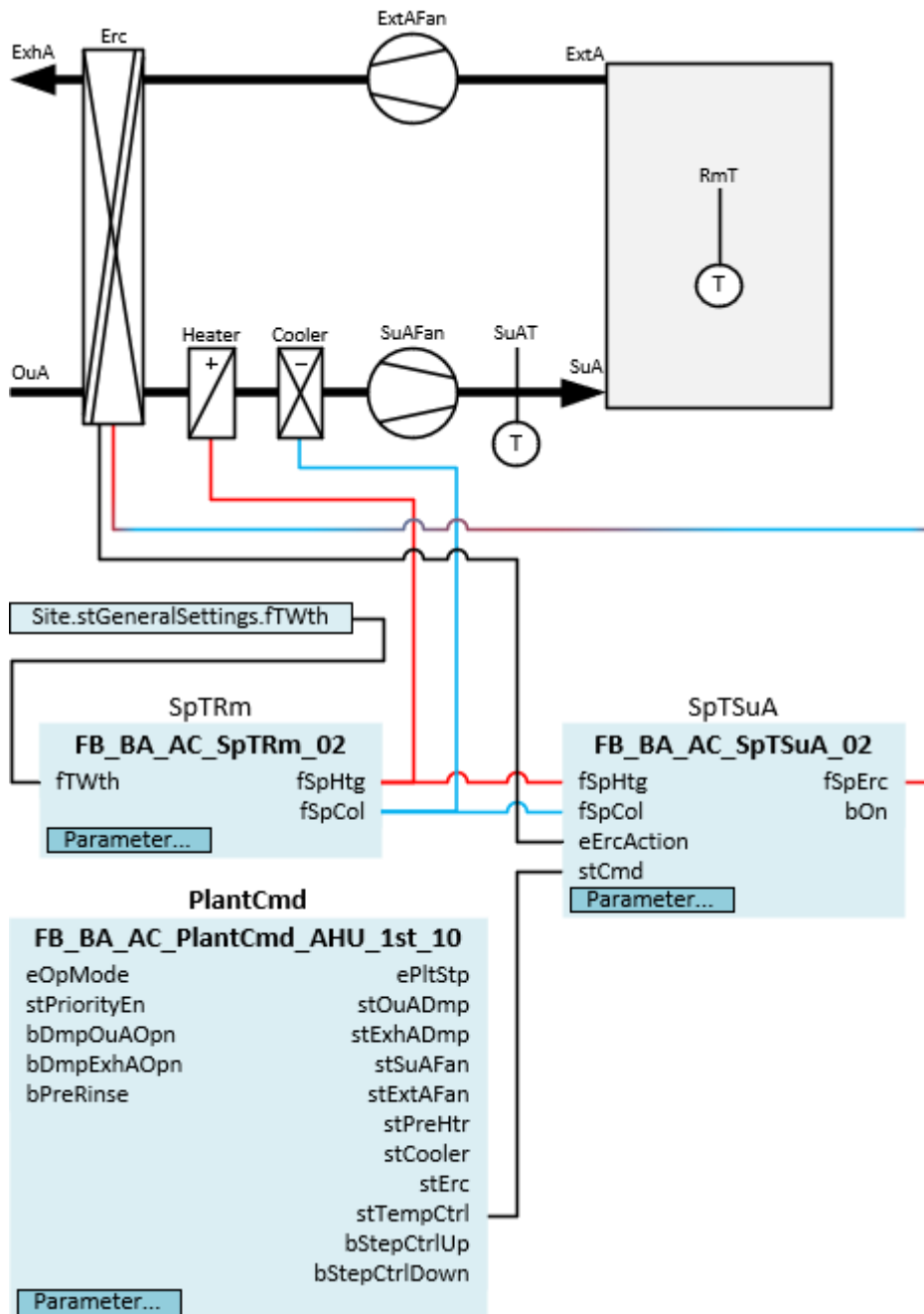


The template is used to determine the supply air temperature setpoint of an air conditioning system. The setpoint linked to the variables *fSpHtg*, *fSpCol* and *fSpErc* is transferred to the supply air controllers of the heater, cooler and energy recovery units.

The supply air temperature setpoint is determined by the function block *FB_BA_Scale_04_Y2_3* [► 733] depending on the outside temperature.

The supply air temperature sequence control is enabled with the variable *bOn* after evaluation of the control structure *stCmd*.

Principle diagram 01



Principle diagram 02



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

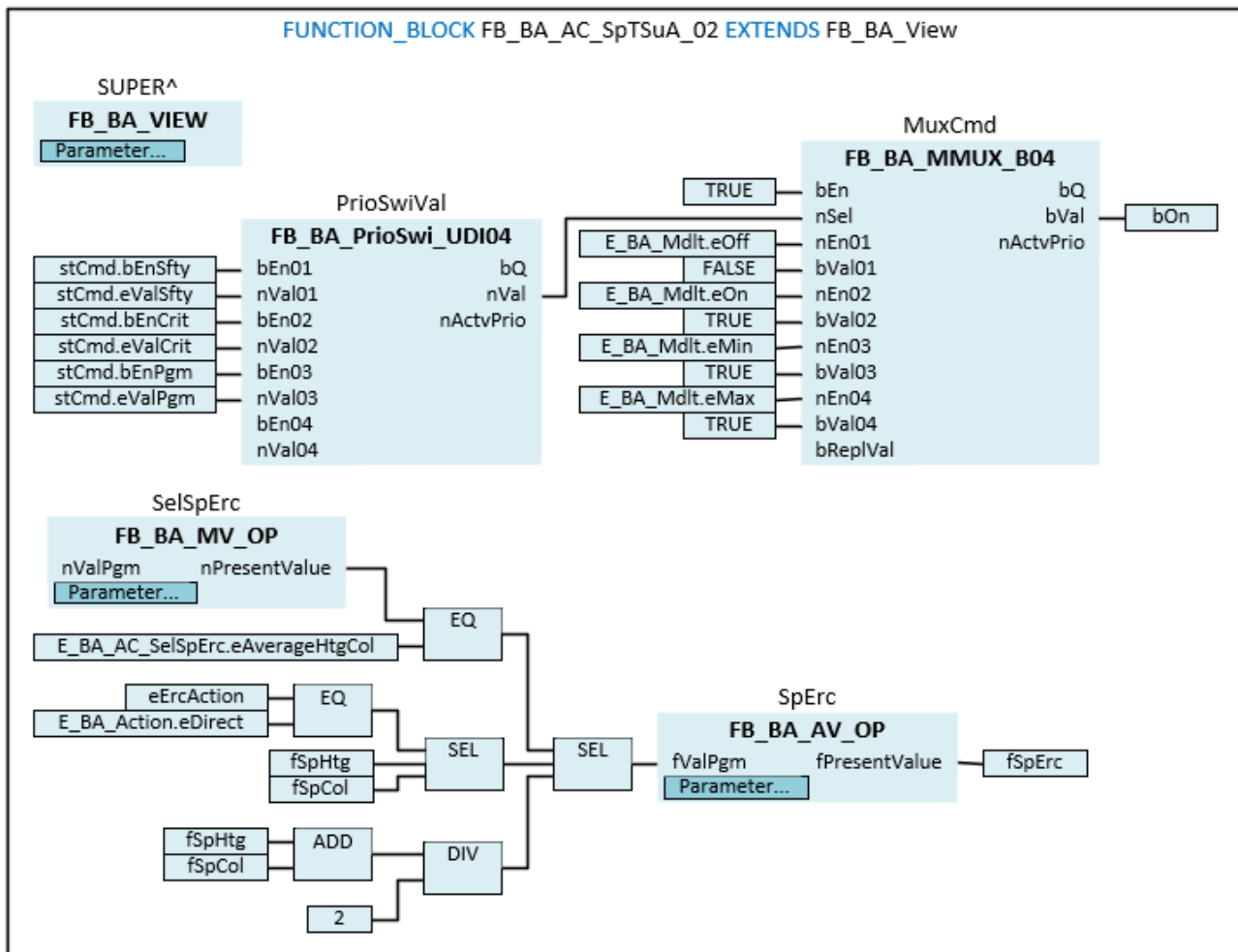
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_AC_SpTSuA_01 EXTENDS FB_BA_View
VAR_INPUT
    fTWth      : REAL;
    stCmd      : ST_BA_MdlT;
END_VAR
VAR_OUTPUT
    fSpHtg     : REAL;
    fSpCol     : REAL;
    fSpErc     : REAL;
    bOn        : BOOL;
END_VAR
VAR
    PrioSwiVal : FB_BA_PrioSwi_UDI04;
    MuxCmd      : FB_BA_MMUX_B04;
END_VAR

```

 Inputs CONSTANT

Name	Type	Description
fTWth	REAL	Input value of the outside temperature.
stCmd	ST_BA_MdlT [► 221]	The enables and modulation commands are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority determines the switching command at the binary output object <i>Cmd</i> and the control command at the analog output object <i>MdlT</i> .

🔌 Outputs

Name	Type	Description
fSpHtg	REAL	Heating setpoint
fSpCol	REAL	Cooling setpoint
fSpErc	REAL	Calculated setpoint of the supply air temperature. for energy recovery.
bOn	BOOL	The output shows the resulting result of the multiplexer <i>MuxCmd</i> . This is further processed within the ventilation system and is used to enable the supply air temperature sequence control.

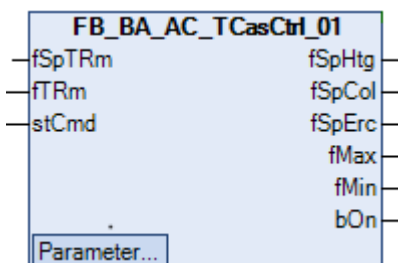
Variables

Name	Type	Description
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 369]	The priority switch <i>PrioSwiVal</i> [▶ 369] uses the command structure <i>stCmd</i> to determine the modulation command for the multiplexers <i>MuxCmd</i> and <i>MuxMdl</i> .
MuxCmd	FB_BA_MMUX_B04 [▶ 364]	The multiplexer <i>MuxCmd</i> [▶ 364] determines the current switch value from the command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the binary output object <i>bOn</i> .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.1.1.4 FB_BA_AC_TCasCtrl_01

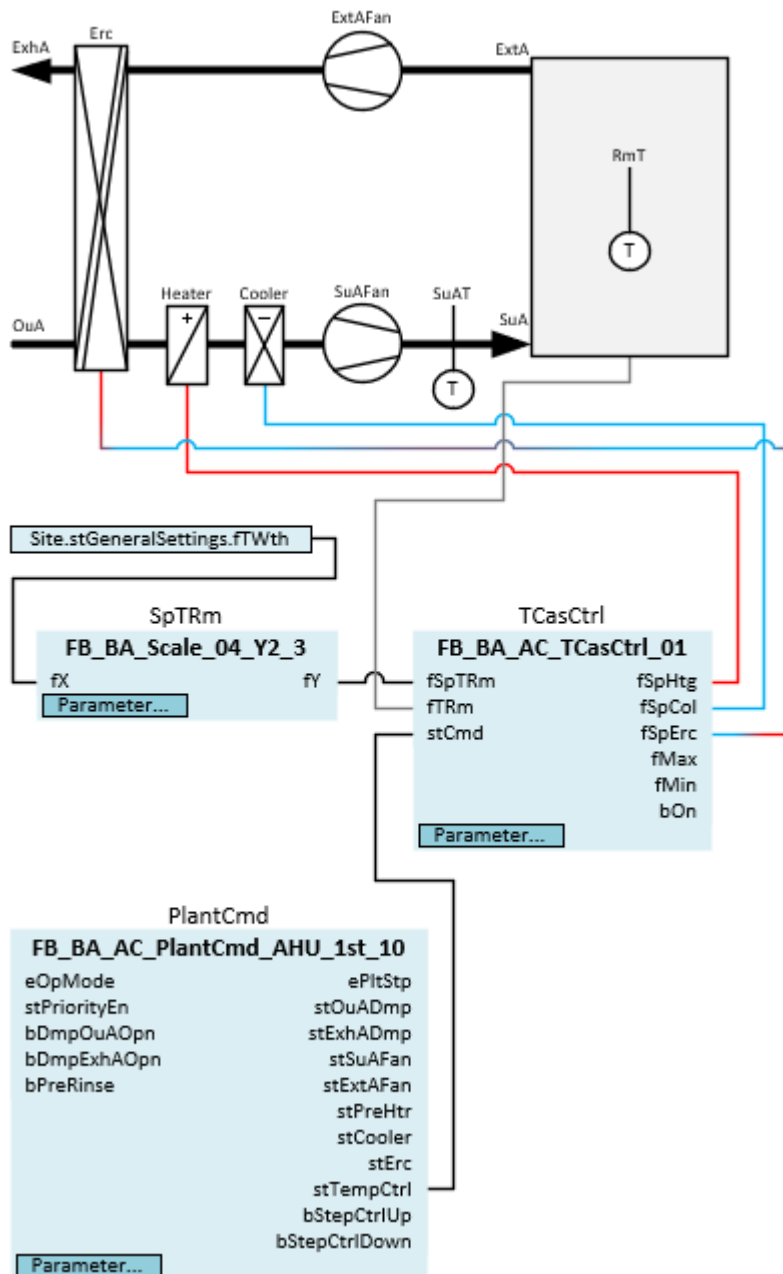


The template is used for room temperature control by means of room-supply air cascade.

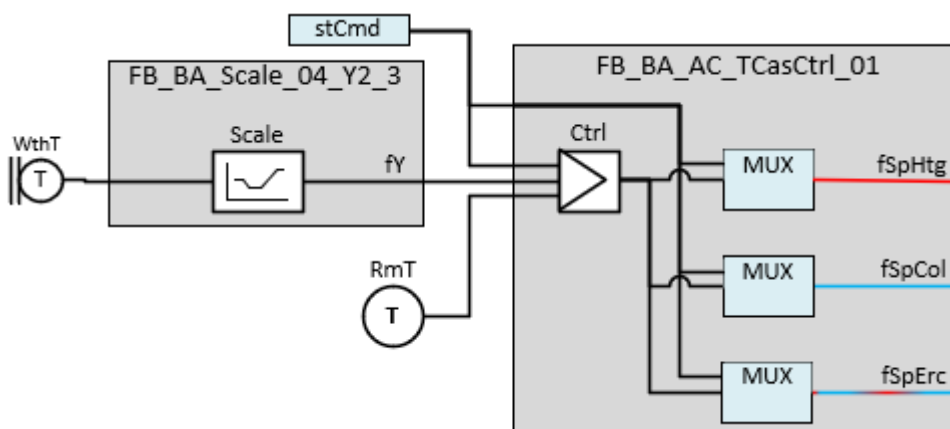
It consists of a master controller for calculating the supply air temperature setpoint for the heater, cooler and energy recovery units.

The supply air temperature sequence control is enabled with the variable *bOn* by evaluating the control structure *stCmd*.

Principle diagram01



Principle diagram02





The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

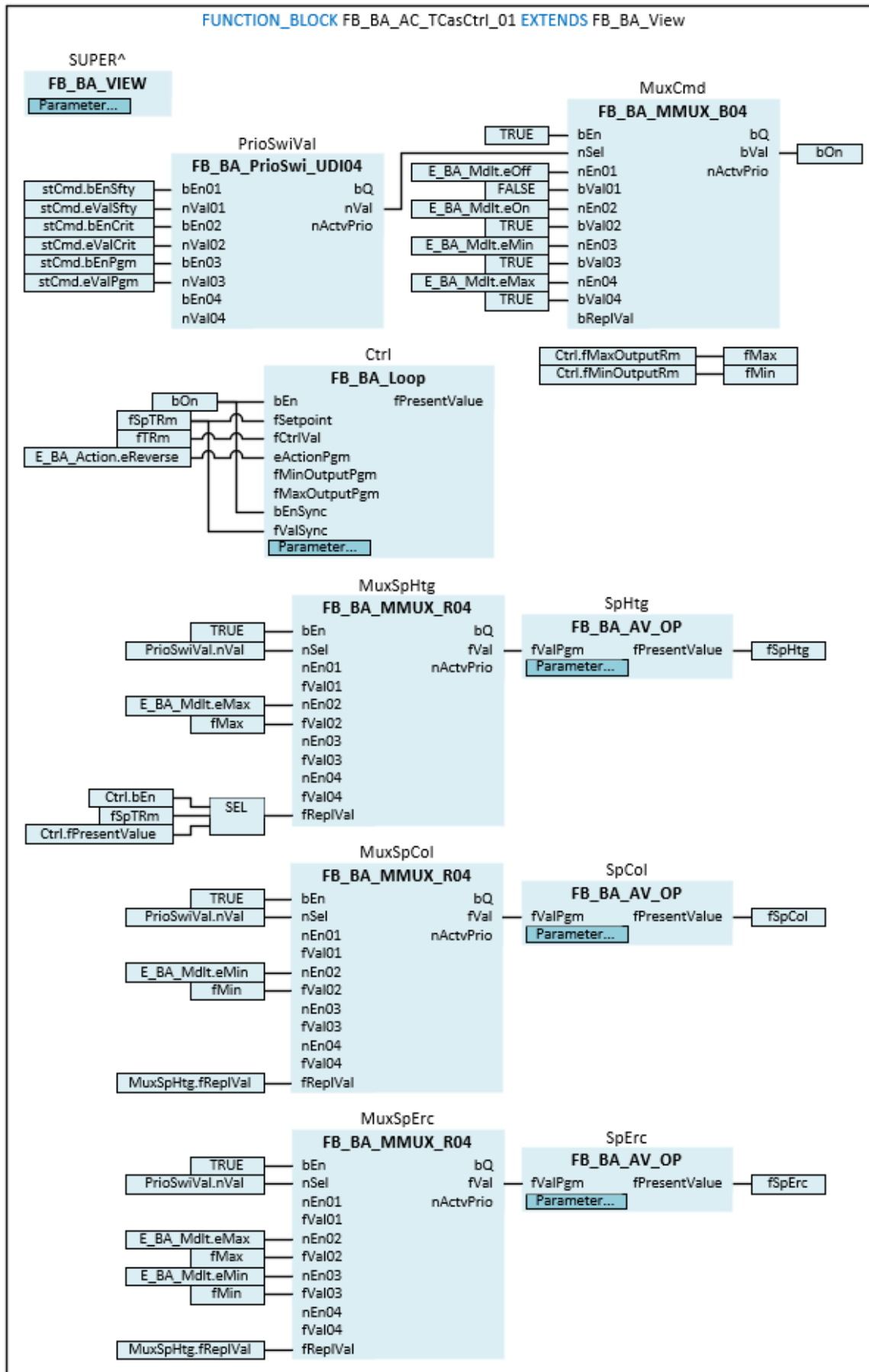
FB_BA_Base

 FB_BA_BasePublisher

 FB_BA_Object [[▶ 195](#)]

 FB_BA_View [[▶ 173](#)]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_AC_TCasCtrl_01 EXTENDS FB_BA_View
VAR_INPUT
    fSpTRm      : REAL;
    fTRm        : REAL;
    stCmd       : ST_BA_Mdlt;
END_VAR
VAR_OUTPUT
    fSpHtg      : REAL;
    fSpCol      : REAL;
    fSpErc      : REAL;
    fMax        : REAL;
    fMin        : REAL;
    bOn         : BOOL;
END_VAR
VAR_INPUT CONSTANT
    SpHtg       : FB_BA_AV_Op;
    SpCol       : FB_BA_AV_Op;
    SpErc       : FB_BA_AV_Op;
    Ctrl        : FB_BA_Loop;
END_VAR
VAR
    PrioSwiVal  : FB_BA_PrioSwi_UDI04;
    MuxCmd      : FB_BA_MMUX_B04;
    MuxSpHtg    : FB_BA_MMUX_R04;
    MuxSpCol    : FB_BA_MMUX_R04;
    MuxSpErc    : FB_BA_MMUX_R04;
END_VAR

```

Inputs CONSTANT

Name	Type	Description
fSpTRm	REAL	Input variable for the room temperature setpoint.
fTRm	REAL	Input variable to which the room temperature is applied. The room temperature is the control value of the PID master controller <i>Ctrl</i> . If no room temperature is available, the outlet air temperature of a ventilation system can be used as control value.
stCmd	ST_BA_Mdlt [► 221]	The enables and modulation commands are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority determines the switching command at the binary output object <i>Cmd</i> and the control command at the analog output object <i>Mdlt</i> .

Outputs

Name	Type	Description
fSpHtg	REAL	Heating setpoint
fSpCol	REAL	Cooling setpoint
fSpErc	REAL	Calculated setpoint of the supply air temperature. for energy recovery.
fMax	REAL	Upper value of the controller output limitation.
fMin	REAL	Lower value of the controller output limitation.
bOn	BOOL	The output shows the resulting result of the multiplexer <i>MuxCmd</i> . This is further processed within the ventilation system and is used to enable the supply air temperature sequence control.

🔧 Inputs CONSTANT

Name	Type	Description
SpHtg	FB BA AV Op [► 136]	Analog value object for displaying the calculated room temperature setpoint heating.
SpCol	FB BA AV Op [► 136]	Analog value object for displaying the calculated room temperature setpoint cooling.
SpErc	FB BA AV Op [► 136]	Analog value object for displaying the calculated supply air temperature setpoint Energy recovery.
Ctrl	FB BA Loop [► 154]	The LOOP object serves as a master controller for a room temperature control by means of a room-supply air cascade. It provides the supply air temperature setpoint.

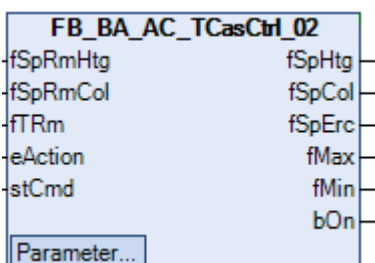
Variables

Name	Type	Description
PrioSwiVal	FB BA PrioSwi UDI04 [► 369]	The priority switch uses the command structure <i>stCmd</i> to determine the modulation command for the multiplexer <i>MuxCmd</i> .
MuxCmd	FB BA MMUX B04 [► 364]	The multiplexer determines the current switching command from the modulation command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the output <i>bOn</i> . The multiplexer defines the enabling conditions of the PID master controller <i>Ctrl</i> .
MuxSpHtg	FB BA MMUX R04 [► 364]	The multiplexer defines the setpoint heating <i>fSpHtg</i> depending on the priority switch <i>PrioSwiVal</i> .
MuxSpCol	FB BA MMUX R04 [► 364]	The multiplexer defines the cooling setpoint <i>fSpCol</i> depending on the priority switch <i>PrioSwiVal</i> .
MuxSpErc	FB BA MMUX R04 [► 364]	The multiplexer defines the setpoint energy recovery <i>fSpErc</i> depending on the priority switch <i>PrioSwiVal</i> .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.1.1.5 FB_BA_AC_TCasCtrl_02

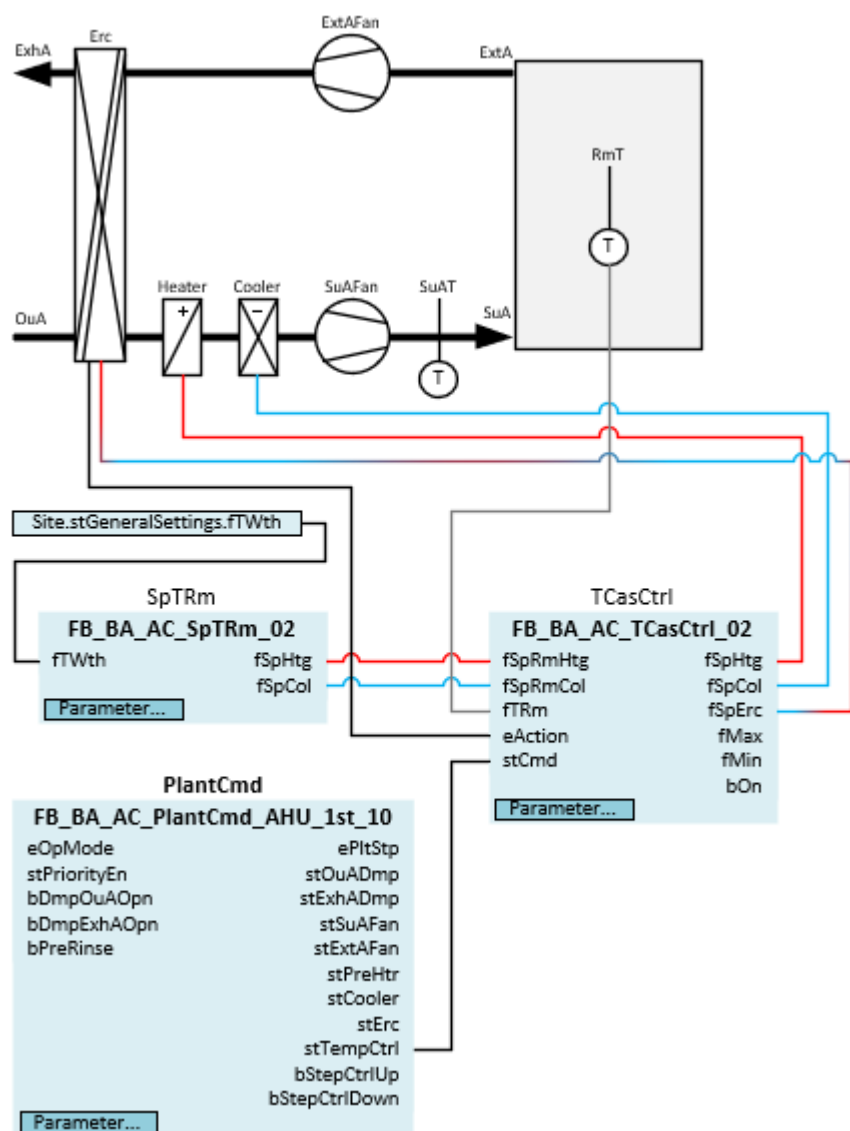


The template is used for room temperature control by means of room-supply air cascade.

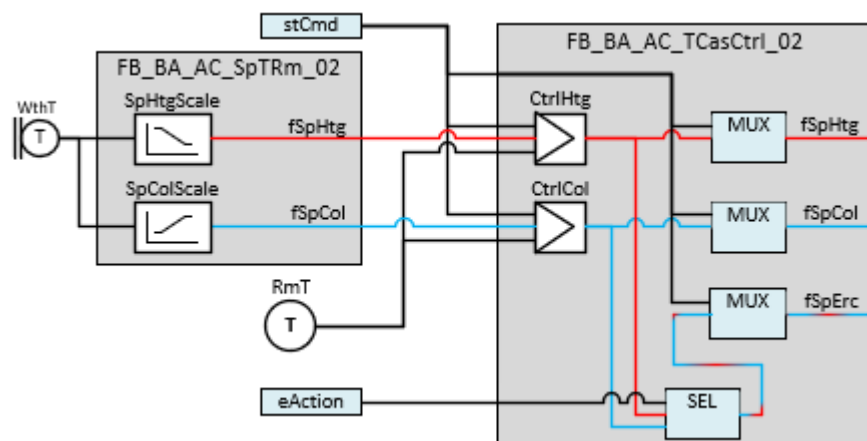
It consists of two master controllers for calculating the supply air temperature setpoints for the heater and cooler units. In addition, it determines the supply air temperature setpoint for energy recovery depending on its control direction.

The supply air temperature sequence control is enabled with the variable *bOn* by evaluating the control structure *stCmd*.

Principle diagram01



Principle diagram02



i

The initialization of the template takes place within the method `FB_Init`.

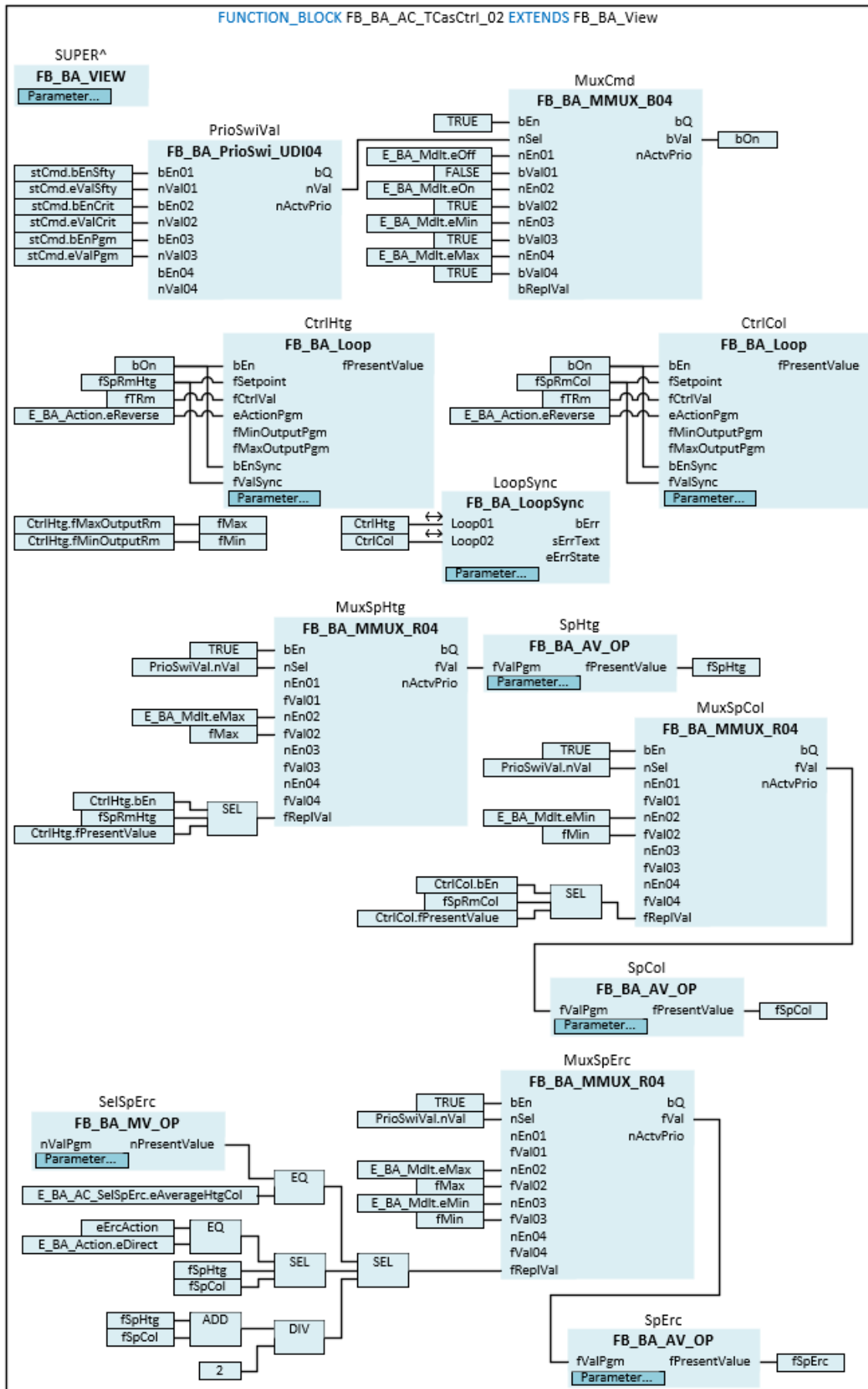
Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)[FB_BA_View \[► 173\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_AC_TCasCtrl_02 EXTENDS FB_BA_View
VAR_INPUT
    fSpRmHtg      : REAL;
    fSpRmCol      : REAL;
    fTRm          : REAL;
    eAction       : E_BA_Action := E_BA_Action.eDirect;
    stCmd         : ST_BA_Mdlt;
END_VAR
VAR_OUTPUT
    fSpHtg        : REAL;
    fSpCol        : REAL;
    fSpErc        : REAL;
    fMax          : REAL;
    fMin          : REAL;
    bOn           : BOOL;
END_VAR
VAR_INPUT CONSTANT
    SpHtg         : FB_BA_AV_Op;
    SpCol         : FB_BA_AV_Op;
    SpErc         : FB_BA_AV_Op;
    SelSpErc      : FB_BA_MV_Op;
    CtrlHtg       : FB_BA_Loop;
    CtrlCol       : FB_BA_Loop;
    LoopSync      : FB_BA_LoopSync;
END_VAR
VAR
    PrioSwiVal    : FB_BA_PrioSwi_UDI04;
    MuxCmd        : FB_BA_MMUX_B04;
    MuxSpHtg      : FB_BA_MMUX_R04;
    MuxSpCol      : FB_BA_MMUX_R04;
    MuxSpErc      : FB_BA_MMUX_R04;
END_VAR

```

Inputs CONSTANT

Name	Type	Description
fSpRmHtg	REAL	Input variable for the setpoint room temperature <u>Heating</u> [► 691].
fSpRmCol	REAL	Input variable for the setpoint room temperature <u>Cooling</u> . [► 691]
fTRm	REAL	Input variable to which the room temperature is applied. The room temperature is the control value of the PID master controller <i>Ctrl</i> . If no room temperature is available, the outlet air temperature of a ventilation system can be used as control value.
eAction	E_BA_Action	Input variable to which the control direction for the energy recovery is applied. The setpoint for the energy recovery is determined depending on the control direction.
stCmd	ST_BA_Mdlt [► 221]	The enables and modulation commands are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority determines the switching command at the binary output object <i>Cmd</i> and the control command at the analog output object <i>Mdlt</i> .

 Outputs

Name	Type	Description
fSpHtg	REAL	Heating setpoint
fSpCol	REAL	Cooling setpoint
fSpErc	REAL	Calculated setpoint of the supply air temperature. for energy recovery.
fMax	REAL	Upper value of the controller output limitation.
fMin	REAL	Lower value of the controller output limitation.
bOn	BOOL	The output shows the resulting result of the multiplexer <i>MuxCmd</i> . This is further processed within the ventilation system and is used to enable the supply air temperature sequence control.

 Inputs CONSTANT

Name	Type	Description
SpHtg	FB BA AV Op [► 136]	Analog value object for displaying the calculated room temperature setpoint heating.
SpCol	FB BA AV Op [► 136]	Analog value object for displaying the calculated room temperature setpoint cooling.
SpErc	FB BA AV Op [► 136]	Analog value object for displaying the calculated supply air temperature setpoint Energy recovery.
SelSpErc	FB BA MV Op [► 171]	The Multistate value object is used to select the strategy setpoint energy recovery. E BA AC SelSpErc.eAction [► 612] : = 1: Depends on the control direction, defined by input <i>eErcAction</i> . E BA AC SelSpErc.eAverageHtgCol [► 612] := 2: Average of input variables <i>fSpHtg</i> and <i>fSpCol</i> .
CtrlHtg	FB BA Loop [► 154]	The LOOP object serves as a master controller for a room temperature control by means of a room-supply air cascade. It provides the supply air temperature setpoint for heating.
CtrlCol	FB BA Loop [► 154]	The LOOP object serves as a master controller for a room temperature control by means of a room-supply air cascade. It provides the supply air temperature setpoint for cooling.
LoopSync	FB BA LoopSync [► 359]	The function block synchronizes the parameters for the two master controllers <i>CtrlHtg/CtrlCol</i> .

Variables

Name	Type	Description
PrioSwiVal	FB_BA_PrioSwi_UDI04 [► 369]	The priority switch uses the command structure <i>stCmd</i> to determine the modulation command for the multiplexer <i>MuxCmd</i> .
MuxCmd	FB_BA_MMUX_B04 [► 364]	The multiplexer determines the current switching command from the modulation command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the output <i>bOn</i> . The multiplexer defines the enabling conditions of the PID master controller <i>Ctrl</i> .
MuxSpHtg	FB_BA_MMUX_R04 [► 364]	The multiplexer defines the setpoint heating <i>fSpHtg</i> depending on the priority switch <i>PrioSwiVal</i> .
MuxSpCol	FB_BA_MMUX_R04 [► 364]	The multiplexer defines the cooling setpoint <i>fSpCol</i> depending on the priority switch <i>PrioSwiVal</i> .
MuxSpErc	FB_BA_MMUX_R04 [► 364]	The multiplexer defines the setpoint energy recovery <i>fSpErc</i> depending on the priority switch <i>PrioSwiVal</i> .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

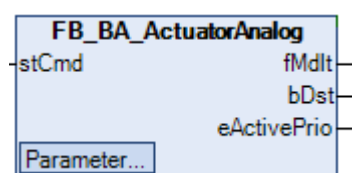
6.1.4.2.2 Universal

Templates for universal use in various building functions.

6.1.4.2.2.1 Aggregate

6.1.4.2.2.1.1 Analog

6.1.4.2.2.1.1.1 FB_BA_ActuatorAnalog



The template is used to control analog aggregates. It essentially consists of an AO object for controlling an aggregate and the function block *PlantLock*, which collects all safety-relevant faults.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

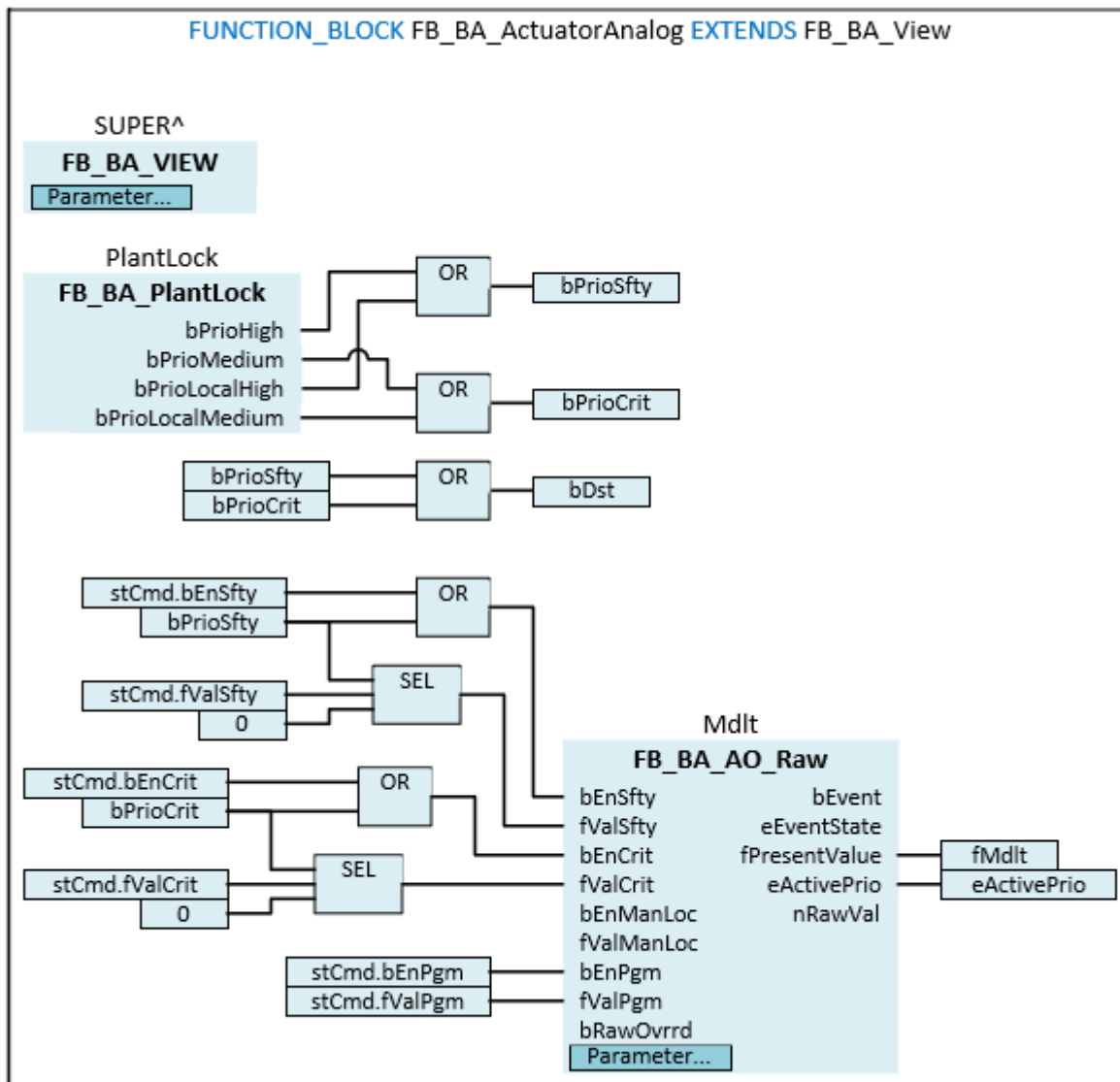
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB BA View [► 173]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
    stCmd          : ST_BA_Analog;
END_VAR
VAR_OUTPUT
    fMdlt          : REAL;
    bDst           : BOOL;
    eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
    Mdlt           : FB_BA_AO_Raw;
    PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
    bPrioSfty      : BOOL;
    bPrioCrit      : BOOL;
END_VAR

```

🔧 Inputs

Name	Type	Description
stCmd	ST_BA_Analog [► 220]	The enables and switching values are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority is output at the analog output object <i>Mdl</i> .

🔧 Outputs

Name	Type	Description
fMdl	REAL	Current value of the analog output object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [► 96]	Display of the active priority.

🔧 Inputs CONSTANT

Name	Type	Description
Mdl	FB_BA_AO_Raw [► 133]	Current value of the analog output object.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

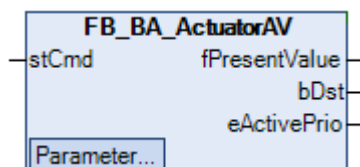
Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.1.1.2 FB_BA_ActuatorAV



The template is used to control analog aggregates. It essentially consists of an AV object for controlling an aggregate and the PlantLock function block, which collects all safety-relevant faults.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

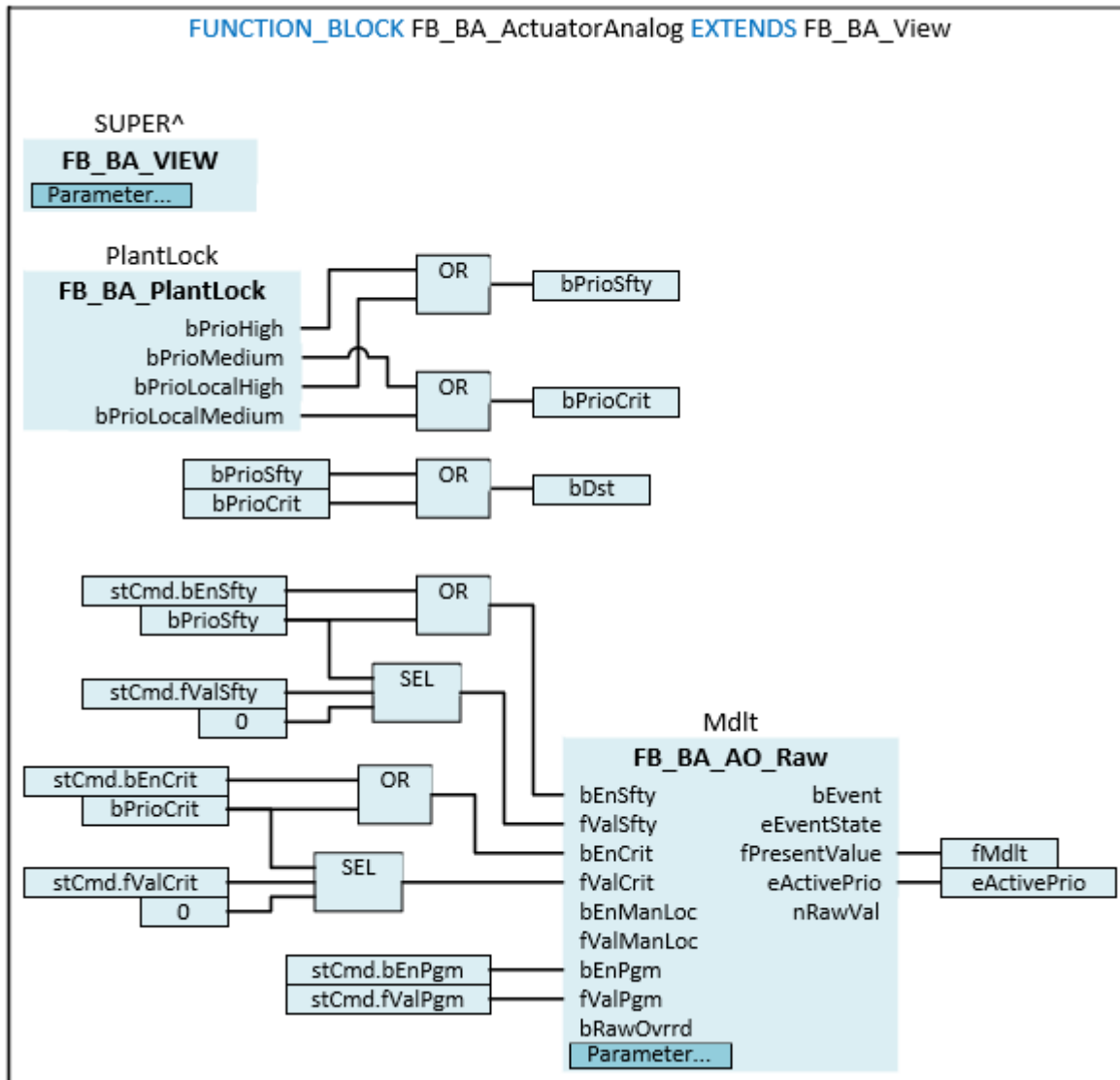
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_ActuatorAV EXTENDS FB_BA_View
VAR_INPUT
    stCmd          : ST_BA_Analog;
END_VAR
VAR_OUTPUT
    fPresentValue  : REAL;
    bDst           : BOOL;
    eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
    Val            : FB_BA_AV;
    PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
```



```

bPrioSfty      : BOOL;
bPrioCrit      : BOOL;
END_VAR

```

Inputs

Name	Type	Description
stCmd	ST_BA_Analog [► 220]	The enables and switching values are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority is output at the analog value object <i>Val</i> .

Outputs

Name	Type	Description
fPresentValue	REAL	Current value of the analog value object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [► 96]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Val	FB_BA_AV [► 134]	The analog value object determines the current control value.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Variables

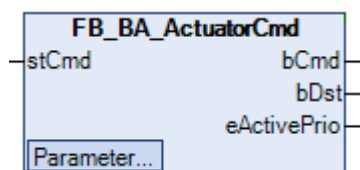
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.1.2 Binary

6.1.4.2.2.1.2.1 FB_BA_ActuatorCmd



The template is used to control binary aggregates. It essentially consists of a BO object for controlling an aggregate and the function block *PlantLock*, which collects all safety-relevant faults.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

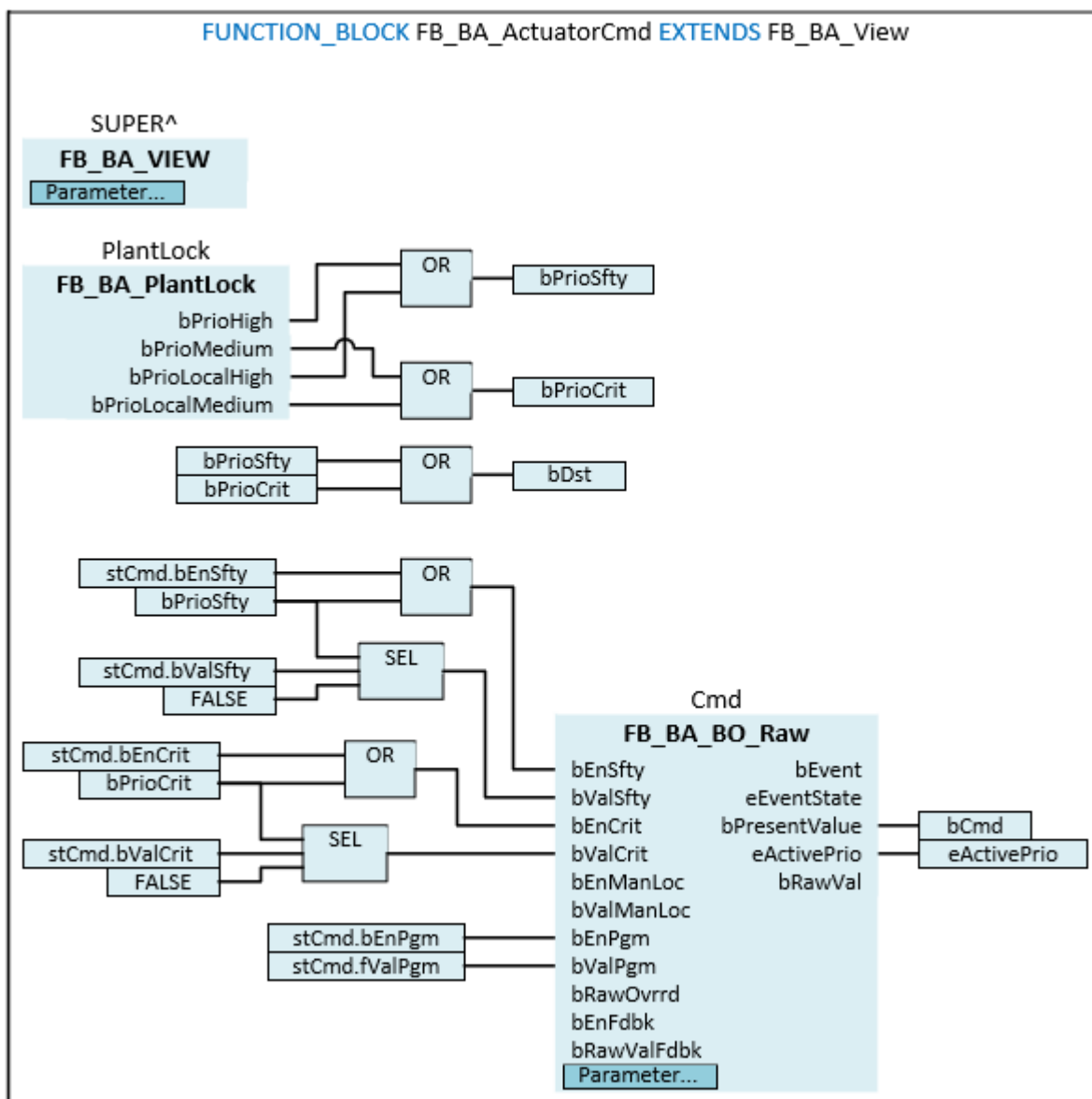
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
    stCmd          : ST_BA_Binary;
END_VAR
```

```

VAR_OUTPUT
  bCmd      : BOOL;
  bDst      : BOOL;
  eActivePrio : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
  Cmd      : FB_BA_BO_Raw;
  PlantLock : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty : BOOL;
  bPrioCrit : BOOL;
END_VAR

```

Inputs

Name	Type	Description
stCmd	ST_BA_Binary [► 221]	The enables and switching values are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority is output at the binary output object <i>Cmd</i> .

Outputs

Name	Type	Description
bCmd	BOOL	Current value of the binary output object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [► 96]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Cmd	FB_BA_BO_Raw [► 145]	The binary output object is used to output a switching command and transmit it to the I/O level.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Variables

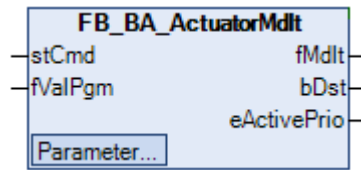
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.1.3 Modulation

6.1.4.2.2.1.3.1 FB_BA_ActuatorMdl



The template is used to control modulating aggregates. It essentially consists of an AO object for controlling an aggregate, the priority switch *PrioSwiVal* for determining the modulation command, the multiplexer *MuxMdl* for determining the control value and the function block *PlantLock*, which collects all safety-related faults.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

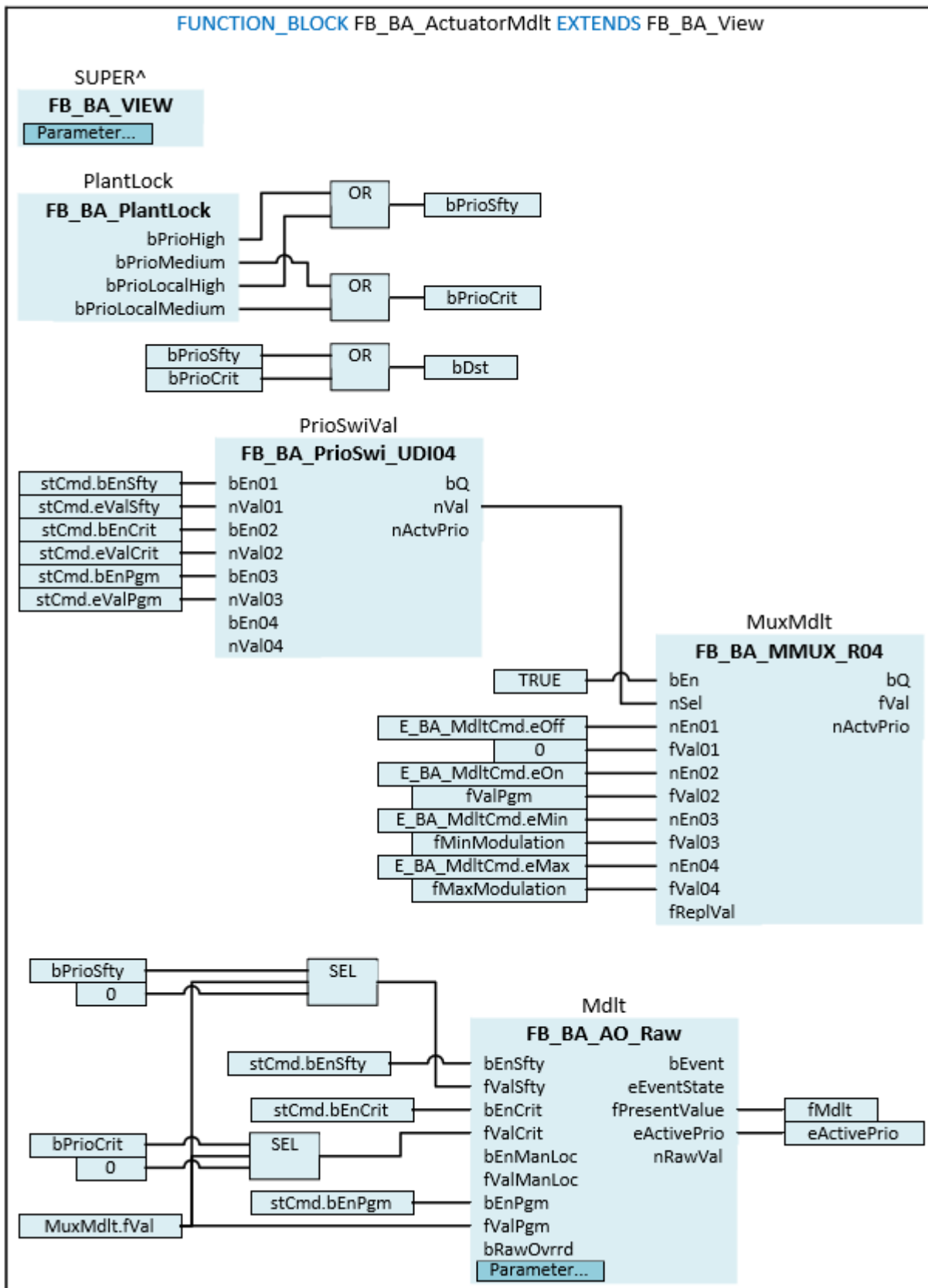
FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_View \[► 173\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_ActuatorMdlnt EXTENDS FB_BA_View
VAR_INPUT
    stCmd          : ST_BA_Mdlnt;
    fValPgm        : REAL;
END VAR

```

```

VAR_OUTPUT
  fMdlT          : REAL;
  bDst           : BOOL;
  eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fMinModulation : REAL := 20;
  {attribute 'parameterCategory' := 'Behaviour'}
  fMaxModulation : REAL := 100;
END_VAR
VAR_INPUT CONSTANT
  MdlT          : FB_BA_AO_Raw;
  PlantLock     : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty     : BOOL;
  bPrioCrit     : BOOL;

  PrioSwiVal    : FB_BA_PrioSwi_UDI04;
  MuxMdlT       : FB_BA_MMUX_R04;
END_VAR

```

Inputs

Name	Type	Description
stCmd	ST_BA_MdlT [► 221]	The enables and modulation commands are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority determines the switching command at the binary output object <i>Cmd</i> and the control command at the analog output object <i>MdlT</i> .
fValPgm	REAL	The control signal for modulation command <i>E_BA_MdlTCmd.eOn</i> is transmitted to the template via the input variable <i>fValPgm</i> .

Outputs

Name	Type	Description
fMdlT	REAL	Current value of the analog output object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [► 96]	Display of the active priority.

Inputs CONSTANT PERSISTENT

Name	Type	Description
fMinModulation	REAL	Constant minimum value when modulation command <i>E_BA_MdlTCmd.eMin</i> [► 614] is pending.
fMaxModulation	REAL	Constant maximum value when modulation command <i>E_BA_MdlTCmd.eMax</i> [► 614] is pending.

Inputs CONSTANT

Name	Type	Description
MdlT	FB_BA_AO_Raw [► 133]	Current value of the analog output object.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Variables

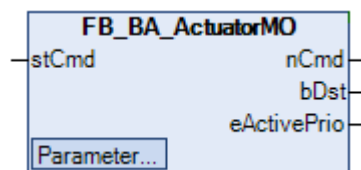
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.
bPrioSwiVal	FB BA PrioSwi UD104 [▶ 369]	The priority switch PrioSwiVal [▶ 369] uses the command structure <i>stCmd</i> to determine the modulation command for the multiplexers <i>MuxCmd</i> and <i>MuxMdl</i> .
MuxMdl	FB BA MMUX R04 [▶ 364]	The multiplexer MuxMdl [▶ 364] determines the current control value from the modulation values <i>fValPgm</i> , <i>fMinModulation</i> and <i>fMaxModulation</i> and the modulation command of the priority switch <i>PrioSwiVal</i> . The result is sent to the analog output object <i>Mdl</i> .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.1.4 Multistate

6.1.4.2.2.1.4.1 FB_BA_ActuatorMO



The template is used to control multi-stage aggregates. It essentially consists of an MO object for controlling an aggregate and the function block *PlantLock*, which collects all safety-relevant faults.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

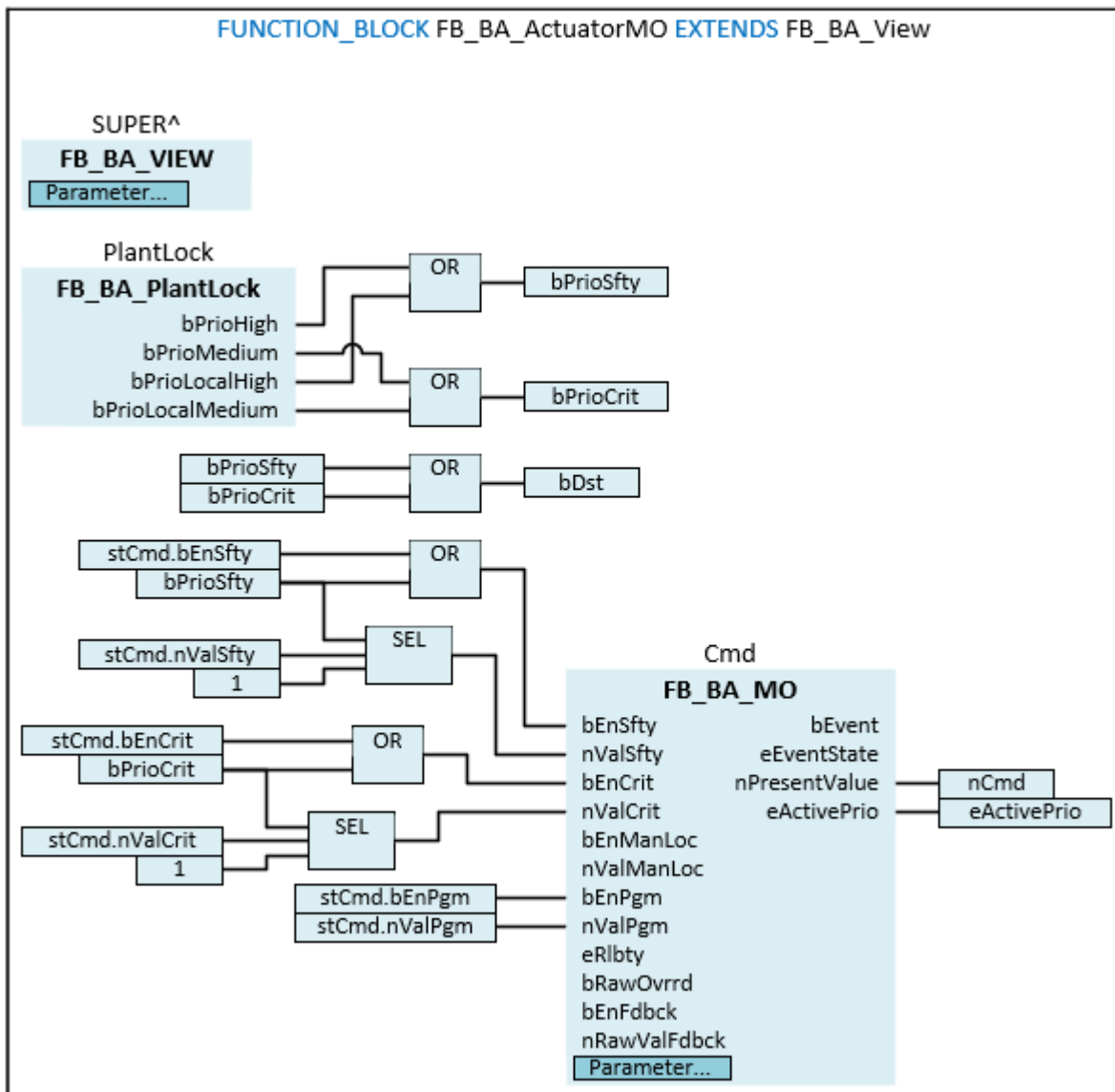
FB_BA_Base

FB_BA_BasePublisher

FB BA Object [▶ 195]

FB BA View [▶ 173]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_ActuatorMO EXTENDS FB_BA_View
VAR_INPUT
    stCmd          : ST_BA_Multistate;
END_VAR
VAR_OUTPUT
    nCmd           : UDINT;
    bDst           : BOOL;
    eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
    Cmd            : FB_BA_MO;
    PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
    bPrioSfty      : BOOL;
    bPrioCrit      : BOOL;
END_VAR

```


Inputs

Name	Type	Description
stCmd	ST_BA_Multistate [► 220]	The enables and switching values are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority is output at the multistate output object <i>Cmd</i> .

Outputs

Name	Type	Description
nCmd	UDINT	Current switch value of the multistate output object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [► 96]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Cmd	FB_BA_MO [► 165]	The multistate output object is used to output the current switch value.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

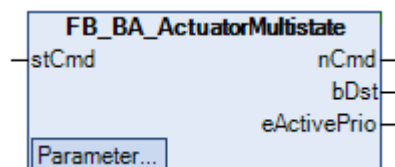
Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.1.4.2 FB_BA_ActuatorMultistate



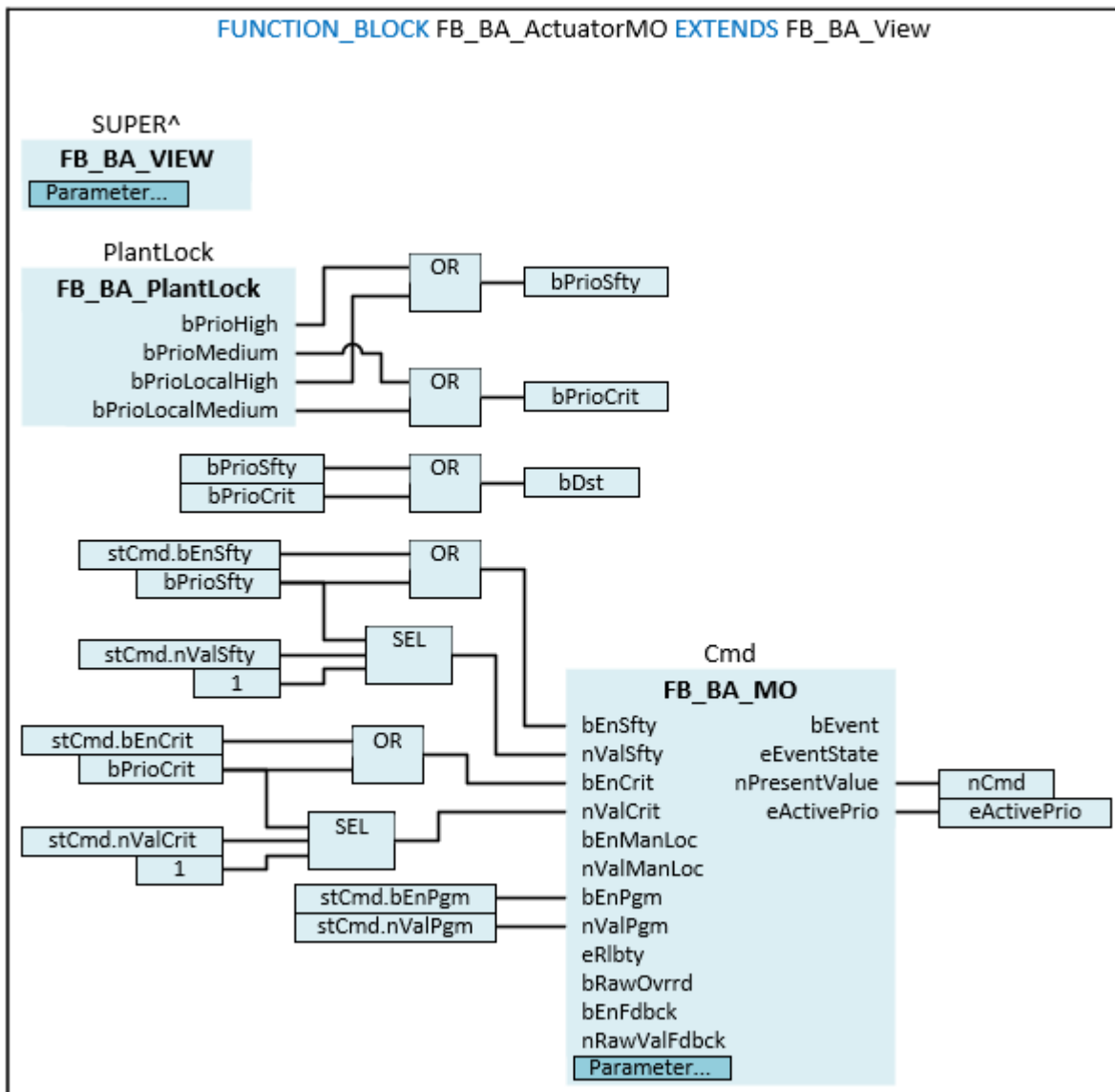
The template is used to control multi-stage aggregates. It essentially consists of an MO object for controlling an aggregate and the function block *PlantLock*, which collects all safety-relevant faults.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method *FB_Init*.

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_ActuatorMultistate EXTENDS FB_BA_View
VAR_INPUT
    stCmd          : ST_BA_Multistate;
END_VAR
VAR_OUTPUT
    nCmd           : UDINT;
    bDst           : BOOL;
    eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
    Cmd            : FB_BA_MO_Raw;
    PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
    bPrioSfty      : BOOL;
    bPrioCrit      : BOOL;
END_VAR

```

🔧 Inputs

Name	Type	Description
stCmd	ST_BA_Multistate [► 220]	The enables and switching values are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority is output at the multistate output object <i>Cmd</i> .

🔧 Outputs

Name	Type	Description
nCmd	UDINT	Current switch value of the multistate output object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [► 96]	Display of the active priority.

🔧 Inputs CONSTANT

Name	Type	Description
Cmd	FB_BA_MO_Raw [► 168]	The multistate output object is used to output the current switch value.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

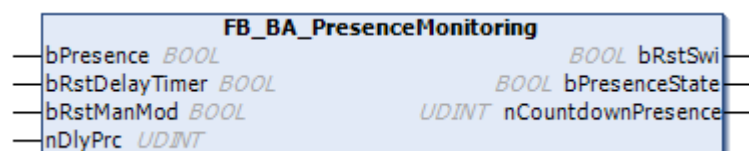
6.1.4.2.2.2 Control

Templates for presence monitoring and scaling.

6.1.4.2.2.2.1 Monitoring

Presence monitoring.

6.1.4.2.2.2.1.1 FB_BA_PresenceMonitoring



Universal presence monitoring template with reset inputs for delay timer and manual function.

Via the input *bPresence* a switch-off delayed occupancy signal is issued at the output *bPresenceState*.

The switch-off delay is defined by *nDlyPrc* [s]. After this time has elapsed, not only *bPresenceState* is set to FALSE again, but also a TRUE pulse at output *bRstSwi*. This pulse can be used to reset manual overrides, for example, on blind or light functions.

If a TRUE signal is given at input *bRstDelayTimer*, the delay timer is cleared and output *bPresenceState* goes to FALSE until presence is detected again by input *bPresence*.

A TRUE signal at input *bRstManMod* specifically triggers output *bRstSwi*.

Both reset functions are important for a central shutdown, where it is assumed that no one is left in place.

At the output *nCountdownPresence* the remaining time of the delay timer can be read in seconds for commissioning purposes.



The initialization of the template takes place within the method *FB_Init*.

Illustration

```
FUNCTION_BLOCK FB_BA_PresenceMonitoring
VAR_INPUT
    bPresence      : BOOL;
    bRstDelayTimer : BOOL;
    bRstManMod     : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    nDlyPrc      : UDINT;
END_VAR
VAR_OUTPUT
    bRstSwi      : BOOL;
    bPresenceState : BOOL;
    nCountdownPresence : UDINT;
END_VAR
VAR
    tofPrcDetc      : TOF;
    rtRstDelayTimer : R_TRIG;
    rtRstManMod     : R_TRIG;
    ftPrc           : F_TRIG;
END_VAR
```

Inputs

Name	Type	Description
bPresence	BOOL	The presence signal input is passed on to the output <i>bPresenceState</i> with a switch-off delay.
bRstDelayTimer	BOOL	Reset input for the switch-off delay. A TRUE signal at this input resets the internal timer.
bRstManMode	BOOL	Reset input for the manual override. A TRUE signal at this input generates a positive edge at the output <i>bRstSwi</i> .

Inputs CONSTANT PERSISTENT

Name	Type	Description
nDlyPrc	UDINT	Switch-off delay time [s]. Preset to 3600 in <i>FB_Init</i> .

🔌 Outputs

Name	Type	Description
bRstSwi	BOOL	Reset output for manual overrides.
bPresenceState	BOOL	Switch-off delayed presence state.
nCountDownPresence	UDINT	Remaining time of the delay timer in seconds.

Variables

Name	Type	Description
tofPrcDetc	TOF	Switch-off delay presence.
rtRstDelayTimer	R_TRIG	Trigger for the reset of the timer.
rtRstManMod	R_TRIG	Trigger for the reset of the manual override.
ftPrc	F_TRIG	Trigger for the reset of the manual override, but controlled by the omission of the presence.

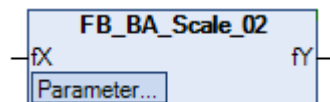
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.2 Scale

Templates for scaling analog values.

6.1.4.2.2.2.2.1 FB_BA_Scale_02



The template represents a linear interpolation with 2 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points $[X1/Y1]$ to $[X2/Y2]$. The calculated output value fY is limited by $YMin$ or $YMax$.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Illustration

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
    fX      : REAL;
END_VAR
VAR_OUTPUT
    fY      : REAL;
END_VAR
VAR_INPUT CONSTANT
```

```

X1      : FB_BA_AV_Op;
X2      : FB_BA_AV_Op;
Y1      : FB_BA_AV_Op;
Y2      : FB_BA_AV_Op;
YMin    : FB_BA_AV_Op;
YMax    : FB_BA_AV_Op;
Y       : FB_BA_AV_Op;
END_VAR
VAR
  Scale  : FB_BA_Chrc02;
END_VAR

```

Inputs

Name	Type	Description
fX	REAL	Input value of the characteristic curve.

Outputs

Name	Type	Description
fY	REAL	Calculated output value of the characteristic curve.

Inputs CONSTANT

Name	Type	Description
X1	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X2.
Y1	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y2.
YMin	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for the minimum limit of fY.
YMax	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for the maximum limit of fY.
Y	FB_BA_AV_Op [► 136]	Output of the calculated output value of the characteristic curve.

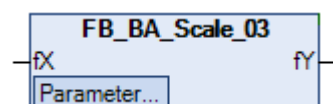
Variables

Name	Type	Description
Scale	FB_BA_Chrc02	The function block represents a linear interpolation with 2 interpolation points and can be used to generate a characteristic curve.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.2 FB_BA_Scale_03

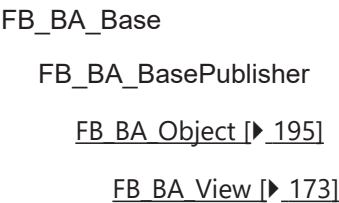


The template represents a linear interpolation with 3 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [X1/Y1] to [X3/Y3]. The calculated output value *fY* is limited by *YMin* or *YMax*.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy



Illustration

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
    fX      : REAL;
END_VAR
VAR_OUTPUT
    fY      : REAL;
END_VAR
VAR_INPUT CONSTANT
    X1      : FB_BA_AV_Op;
    X2      : FB_BA_AV_Op;
    X3      : FB_BA_AV_Op;
    Y1      : FB_BA_AV_Op;
    Y2      : FB_BA_AV_Op;
    Y3      : FB_BA_AV_Op;
    YMin    : FB_BA_AV_Op;
    YMax    : FB_BA_AV_Op;
    Y       : FB_BA_AV_Op;
END_VAR
VAR
    Scale   : FB_BA_Chrc03;
END_VAR
```

 Inputs

Name	Type	Description
fX	REAL	Input value of the characteristic curve.

 Outputs

Name	Type	Description
fY	REAL	Calculated output value of the characteristic curve.

🚩 Inputs CONSTANT

Name	Type	Description
X1	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X3.
Y1	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y2.
Y3	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y3.
YMin	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for the minimum limit of fY .
YMax	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for the maximum limit of fY .
Y	FB_BA_AV_Op [▶ 136]	Output of the calculated output value of the characteristic curve.

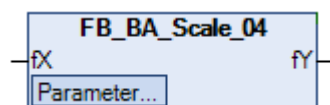
Variables

Name	Type	Description
Scale	FB_BA_Chrc03	The function block represents a linear interpolation with 3 interpolation points and can be used to generate a characteristic curve.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.3 FB_BA_Scale_04



The template represents a linear interpolation with 4 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points $[X1/Y1]$ to $[X4/Y4]$. The calculated output value fY is limited by $YMin$ or $YMax$.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object](#) [▶ 195]

[FB_BA_View](#) [▶ 173]

Illustration

```

FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
    fX      : REAL;
END_VAR
VAR_OUTPUT
    fY      : REAL;
END_VAR
VAR_INPUT CONSTANT
    X1      : FB_BA_AV_Op;
    X2      : FB_BA_AV_Op;
    X3      : FB_BA_AV_Op;
    X4      : FB_BA_AV_Op;
    Y1      : FB_BA_AV_Op;
    Y2      : FB_BA_AV_Op;
    Y3      : FB_BA_AV_Op;
    Y4      : FB_BA_AV_Op;
    YMin     : FB_BA_AV_Op;
    YMax     : FB_BA_AV_Op;
    Y        : FB_BA_AV_Op;
END_VAR
VAR
    Scale    : FB_BA_Chrc04;
END_VAR

```

 Inputs

Name	Type	Description
fX	REAL	Input value of the characteristic curve.

 Outputs

Name	Type	Description
fY	REAL	Calculated output value of the characteristic curve.

 Inputs CONSTANT

Name	Type	Description
X1	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X3.
X4	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X4.
Y1	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y2.
Y3	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y3.
Y4	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y4.
YMin	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for the minimum limit of fY.
YMax	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for the maximum limit of fY.
Y	FB_BA_AV_Op [► 136]	Output of the calculated output value of the characteristic curve.

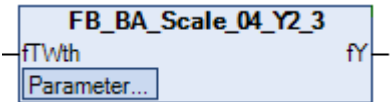
Variables

Name	Type	Description
Scale	FB_BA_Chrc04	The function block represents a linear interpolation with 4 interpolation points and can be used to generate a characteristic curve.

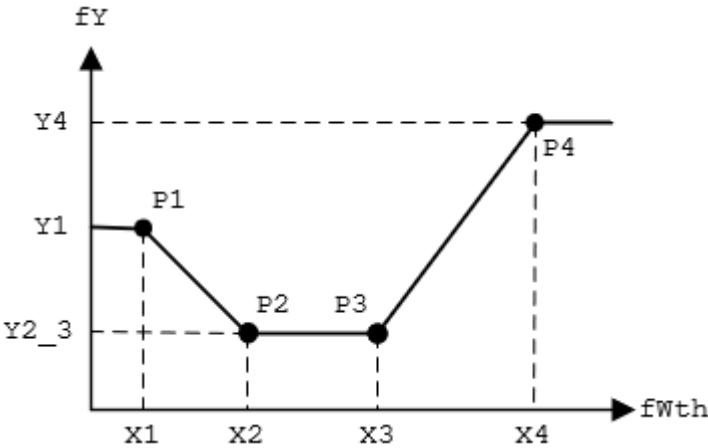
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.4 FB_BA_Scale_04_Y2_3



The template is a setpoint program for a supply air/exhaust air cascade with only one room temperature setpoint, including summer and winter compensation.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base
 FB_BA_BasePublisher
 FB_BA_Object [► 195]
 FB_BA_View [► 173]

Illustration

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
    fTWth      : REAL;
END_VAR
VAR_OUTPUT
    fY         : REAL;
END_VAR
VAR_INPUT CONSTANT
    x1         : FB_BA_AV_Op;
    x2         : FB_BA_AV_Op;
    x3         : FB_BA_AV_Op;
    x4         : FB_BA_AV_Op;
```

```

Y1      : FB_BA_AV_Op;
Y2_3    : FB_BA_AV_Op;
Y4      : FB_BA_AV_Op;
Y       : FB_BA_AV_Op;
END_VAR
VAR
  Scale  : FB_BA_Chrc04;
END_VAR

```

Inputs

Name	Type	Description
fTWth	REAL	Input value of the outside temperature.

Outputs

Name	Type	Description
fY	REAL	Calculated setpoint for the room temperature.

Inputs CONSTANT

Name	Type	Description
X1	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X3.
X4	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point X4.
Y1	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y1.
Y2_3	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for the interpolation points Y2/Y3.
Y4	FB_BA_AV_Op [► 136]	The AV object is used to specify the value for interpolation point Y4.
Y	FB_BA_AV_Op [► 136]	Output of the calculated, simple room temperature setpoint. It is output at output fY.

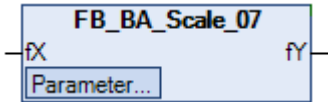
Variables

Name	Type	Description
Scale	FB_BA_Chrc04	The function block represents a linear interpolation with 4 interpolation points and can be used to generate a characteristic curve. The function block calculates the set value characteristic curve for the current room temperature, depending on the outside temperature.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.5 FB_BA_Scale_07

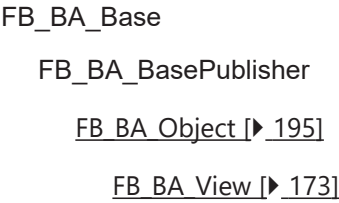


The template represents a linear interpolation with 7 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [X1/Y1] to [X7/Y7]. The calculated output value *fY* is limited by *YMin* or *YMax*.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy



Illustration

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
    fX      : REAL;
END_VAR
VAR_OUTPUT
    fY      : REAL;
END_VAR
VAR_INPUT CONSTANT
    X1      : FB_BA_AV_Op;
    X2      : FB_BA_AV_Op;
    X3      : FB_BA_AV_Op;
    X4      : FB_BA_AV_Op;
    X5      : FB_BA_AV_Op;
    X6      : FB_BA_AV_Op;
    X7      : FB_BA_AV_Op;
    Y1      : FB_BA_AV_Op;
    Y2      : FB_BA_AV_Op;
    Y3      : FB_BA_AV_Op;
    Y4      : FB_BA_AV_Op;
    Y5      : FB_BA_AV_Op;
    Y6      : FB_BA_AV_Op;
    Y7      : FB_BA_AV_Op;
    YMin    : FB_BA_AV_Op;
    YMax    : FB_BA_AV_Op;
    Y       : FB_BA_AV_Op;
END_VAR
VAR
    Scale   : FB_BA_Chrc07;
END_VAR
```

🚩 Inputs

Name	Type	Description
fX	REAL	Input value of the characteristic curve.

🔌 Outputs

Name	Type	Description
fY	REAL	Calculated output value of the characteristic curve.

🔌 Inputs CONSTANT

Name	Type	Description
X1	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X3.
X4	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X4.
X5	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X5.
X6	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X6.
X7	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point X7.
Y1	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y2.
Y3	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y3.
Y4	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y4.
Y5	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y5.
Y6	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y6.
Y7	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for interpolation point Y7.
YMin	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for the minimum limit of fY.
YMax	FB_BA_AV_Op [▶ 136]	The AV object is used to specify the value for the maximum limit of fY.
Y	FB_BA_AV_Op [▶ 136]	Output of the calculated output value of the characteristic curve.

Variables

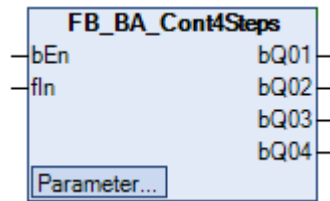
Name	Type	Description
Scale	FB_BA_Chrc07	The function block represents a linear interpolation with 7 interpolation points and can be used to generate a characteristic curve.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.3 ContinuousSteps

6.1.4.2.2.2.3.1 FB_BA_Cont4Steps



The template determines the resulting control steps of a 4-level aggregate, depending on the continuous input signal *fln*.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

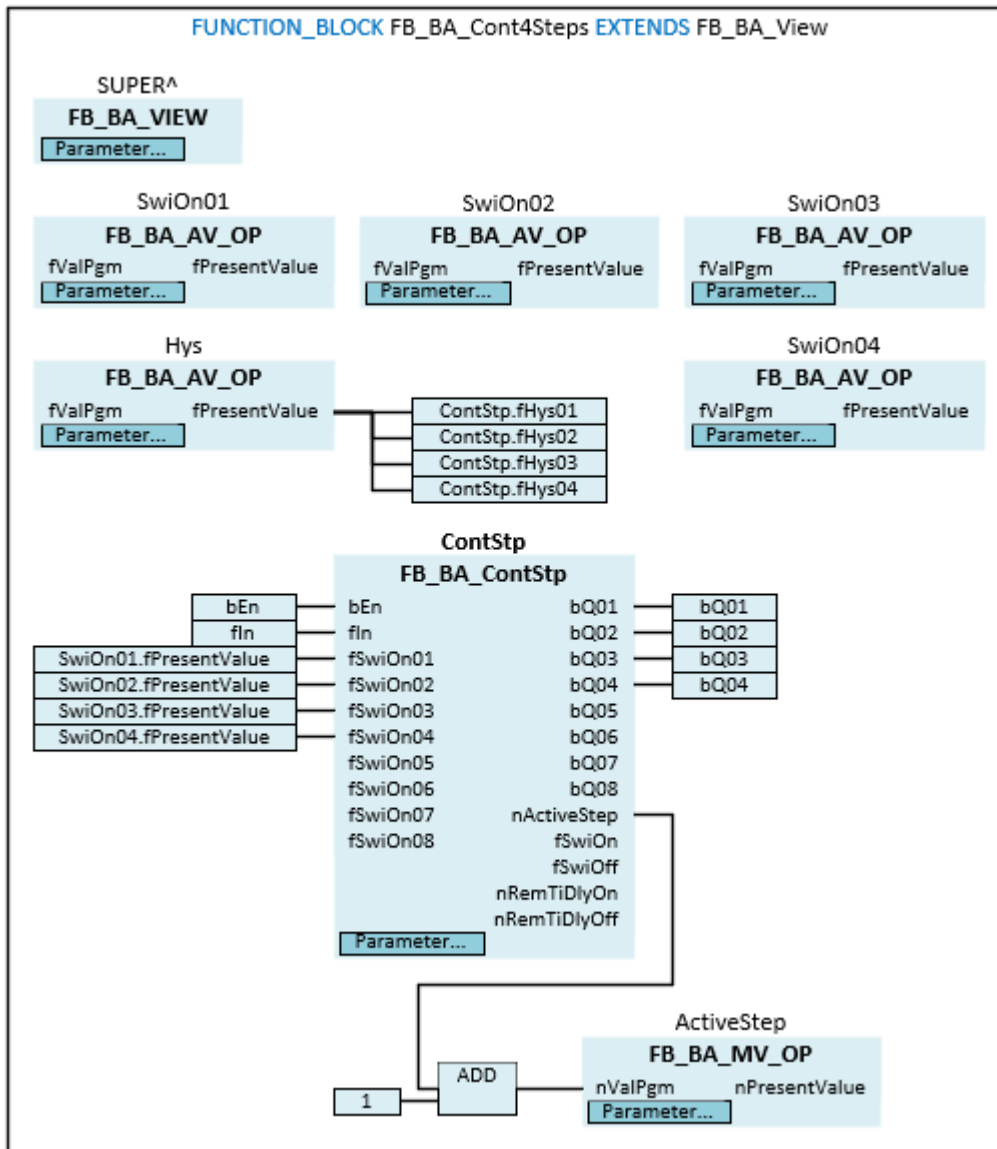
FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_View \[► 173\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_Cont4Steps EXTENDS FB_BA_View
VAR_INPUT
    bEn      : BOOL := TRUE;
    fIn      : REAL;
END_VAR
VAR_OUTPUT
    bQ01     : BOOL;
    bQ02     : BOOL;
    bQ03     : BOOL;
    bQ04     : BOOL;
END_VAR
VAR_INPUT CONSTANT
    SwitchOn01 : FB_BA_AV_Op;
    SwitchOn02 : FB_BA_AV_Op;
    SwitchOn03 : FB_BA_AV_Op;
    SwitchOn04 : FB_BA_AV_Op;
    Hys        : FB_BA_AV_Op;
    ActiveStep  : FB_BA_MV_Op;
END_VAR
VAR ContStp : FB_BA_ContStp;
END_VAR

```

🔧 Inputs

Name	Type	Description
bEn	BOOL	General enable of the template. If <i>bEn</i> is FALSE, all message outputs <i>bQ0x</i> are also FALSE.
fIn	REAL	Continuous input value from which the switching states are derived.

🔧 Outputs

Name	Type	Description
bQ01	BOOL	Shows status level 01
bQ02	BOOL	Shows status level 02
bQ03	BOOL	Shows status level 03
bQ04	BOOL	Shows status level 04

🔧 Inputs CONSTANT

Name	Type	Description
SwitchOn01	FB_BA_AV_Op [► 136]	AV object for entering the switch-on point step 01.
SwitchOn02	FB_BA_AV_Op [► 136]	AV object for entering the switch-on point step 02.
SwitchOn03	FB_BA_AV_Op [► 136]	AV object for entering the switch-on point step 03.
SwitchOn04	FB_BA_AV_Op [► 136]	AV object for entering the switch-on point step 04.
Hys	FB_BA_AV_Op [► 136]	AV object for entering the hysteresis for the switch-on points.
ActiveStep	FB_BA_AV_Op [► 136]	MV object for displaying how many steps are switched on.

Variables

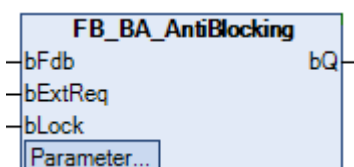
Name	Type	Description
ContStp	FB_BA_ContStp [► 356]	The function block determines the resulting control steps of a multi-stage aggregate depending on the continuous input signal <i>fIn</i> and is the core of this template

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.4 AntiBlocking

6.1.4.2.2.2.4.1 FB_BA_AntiBlocking



The template prevents blocking of pumps or actuators after prolonged idle periods by issuing a switch-on pulse.

A pulse output generally only occurs if the function block *FB_BA_AntBlkg* is enabled at *bEn*.

The maximum idle period before such a pulse is issued is determined by the value of the variable *nOffMin*. For logging the idle time, the input *bFdb* must be linked to the operating feedback from the aggregate. The length of the pulse is parameterized with the variable *nImplLngt*. For this function the operation mode *E_BA_AntBlkgMode.eOffTime* must be set (see *E_BA_AntBlkgMode*).

The input *bExtReq* should be used if the blocking protection pulse is to be issued cyclically based on a schedule, rather than depending on the idle times. A rising edge at *bExtReq* immediately triggers output of a pulse to *bQ*. For this function the operation mode *E_BA_AntBlkgMode.eExternalRequest* must be set (see *E_BA_AntBlkgMode*).



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

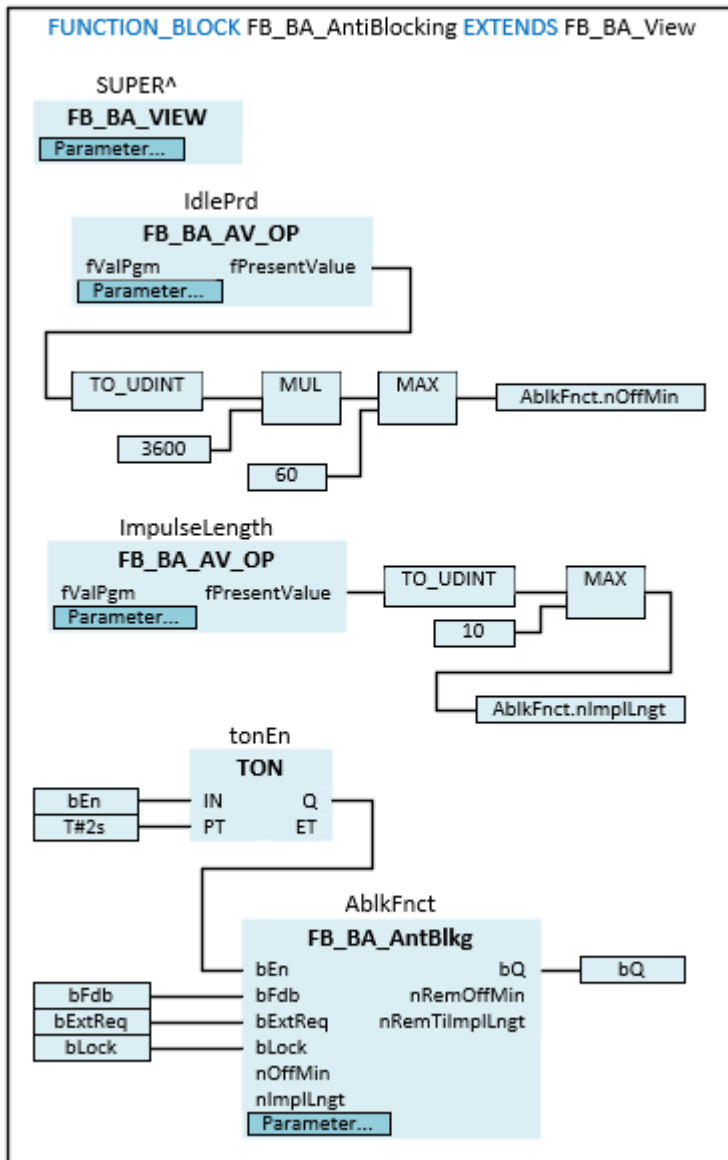
FB_BA_Base

 FB_BA_BasePublisher

 FB_BA_Object [[▶ 195](#)]

 FB_BA_View [[▶ 173](#)]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_AntBlkg EXTENDS FB_BA_View
VAR_INPUT
    bFdb          : BOOL;
    bExtReq       : BOOL;
    bLock         : BOOL;
END_VAR
VAR_OUTPUT
    bQ            : BOOL;
END_VAR
VAR_INPUT_CONSTANT
    IdlePrd       : FB_BA_AV_Op;
    ImpulseLength : FB_BA_AV_Op;
END_VAR
VAR_INPUT_CONSTANT_PERSISTENT
    {attribute 'parameterCategory':='Operation'}
    bEn           : BOOL := TRUE;
END_VAR
VAR
    AbkFnct       : FB_BA_AntBlkg;
    tonEn         : TON;
END_VAR
```

Inputs

Name	Type	Description
bFdb	BOOL	Input for connecting the feedback signal of a motor or valve. This input is only considered in the operation mode <i>E_BA_AntBlkgMode.eOffTime</i> .
bExtReq	BOOL	Active in the <i>E_BA_AntBlkgMode.eExternalRequest</i> operation mode. External request for a pulse, for example from a schedule. The blocking protection pulse is started with a rising edge.
bLock	BOOL	Active in the operation modes <i>E_BA_AntBlkgMode.eExternalRequest</i> or <i>E_BA_AntBlkgMode.eOffTime</i> . To prevent that e.g. the pump and the valve of a heater get a pulse at the same time, the output of the pulse is always suppressed until <i>bLock</i> is FALSE again. If <i>bLock</i> becomes TRUE during the output of a blocking protection pulse, then the blocking protection pulse is interrupted. After <i>bLock</i> is FALSE again, the blocking protection pulse is restarted.

Outputs

Name	Type	Description
bQ	BOOL	Output of the blocking protection pulse.

Inputs CONSTANT

Name	Type	Description
IdlePrd	FB BA AV Op [► 136]	AV object for input of the maximum pump standstill duration until a blocking protection pulse is issued.
ImpulseLength	FB BA AV Op [► 136]	A rising edge at this input switches the output value <i>fOut</i> to the input value <i>fIn</i> .

Inputs CONSTANT PERSISTENT

Name	Type	Description
bEn	BOOL	General enable of the template. If <i>bEn</i> is FALSE, the message output <i>bQ</i> is also FALSE.

Variables

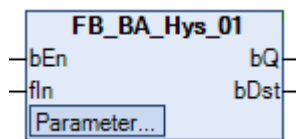
Name	Type	Description
bAbkFnct	FB BA AntBlkg [► 348]	The function block <i>AbkFnct</i> for outputting a blocking protection pulse is the core of this template.
tonEn	TON	Start-up delay of the function after the controller has started up.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.5 Hysteresis

6.1.4.2.2.5.1 FB_BA_Hys_01



The template represents a hysteresis function with fixed switching points.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

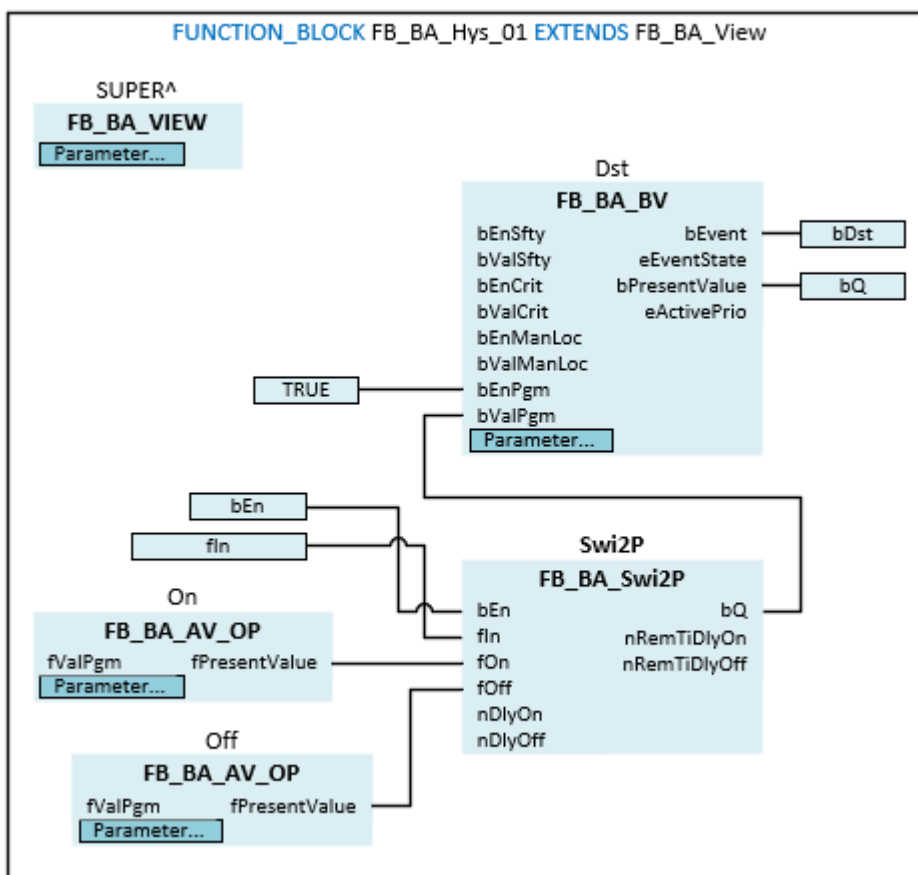
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Hys_01 EXTENDS FB_BA_View
VAR_INPUT
    bEn      : BOOL;
    fln      : REAL;
END_VAR
VAR_OUTPUT
```

```

    bQ      : BOOL;
    bDst    : BOOL;
END_VAR
VAR_INPUT CONSTANT
    On      : FB_BA_AV_Op;
    Off     : FB_BA_AV_Op;
    Q       : FB_BA_BV;
END_VAR
VAR
    Swi2P   : FB_BA_Swi2P;
END_VAR

```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the function block.
fIn	REAL	Actual value.

Outputs

Name	Type	Description
bQ	BOOL	Output of the current state of the hysteresis function.
bDst	BOOL	Display of a fault or the BV object is active. <i>bDst</i> is only active if the property <i>bEventDetectionEnable</i> of the BV object was set to TRUE. The monitoring of the binary feedback indicates a fault.

Inputs CONSTANT

Name	Type	Description
On	FB_BA_AV_Op [► 136]	AV object for input of the upper limit of the hysteresis function.
Off	FB_BA_AV_Op [► 136]	AV object for input of the lower limit of the hysteresis function.
Q	FB_BA_BV [► 147]	The binary object indicates the current state of the hysteresis function.

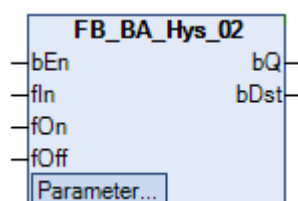
Variables

Name	Type	Description
Swi2P	FB_BA_Swi2P [► 388]	The function block Swi2P is the core of the hysteresis function.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.5.2 FB_BA_Hys_02



The template represents a sliding limit value monitoring with two switching points.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

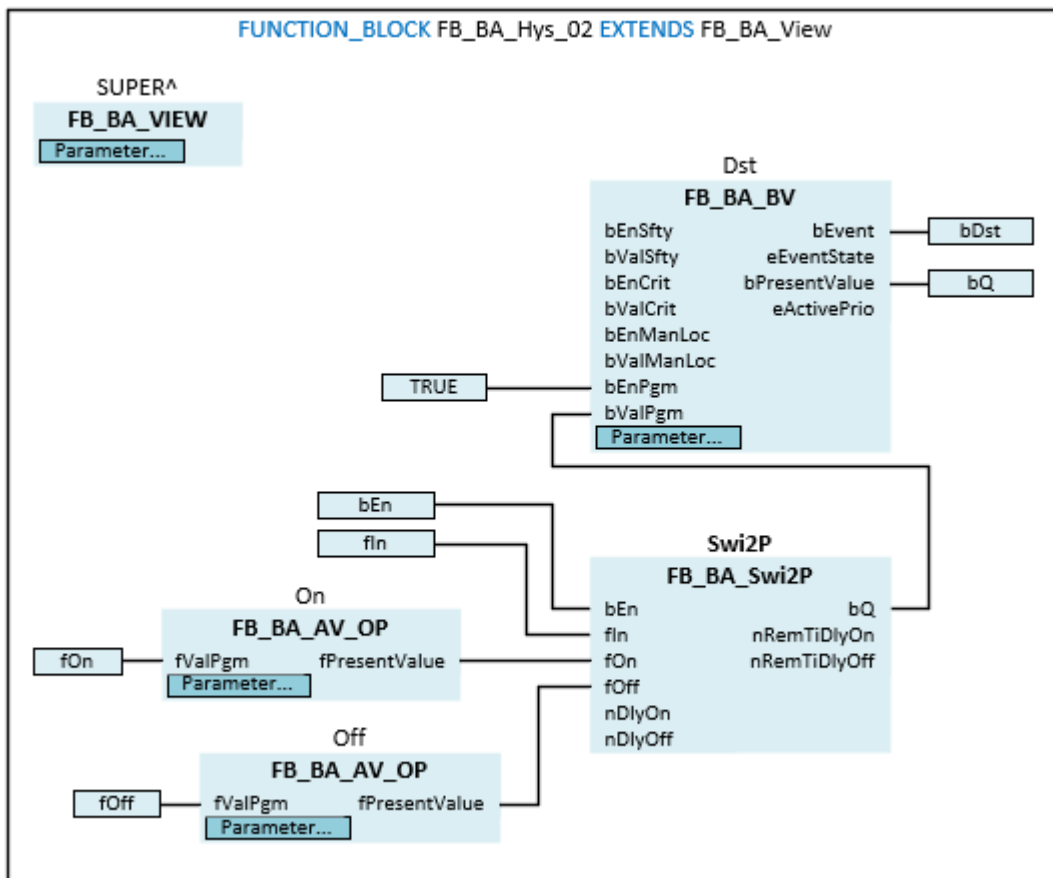
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Hys_01 EXTENDS FB_BA_View
VAR_INPUT
    bEn      : BOOL;
    fIn      : REAL;
    fOn      : REAL;
    fOff     : REAL;
END_VAR
VAR_OUTPUT
    bQ       : BOOL;
    bDst     : BOOL;
END_VAR
VAR_INPUT CONSTANT
    On       : FB_BA_AV_Op;
    Off      : FB_BA_AV_Op;
    Q        : FB_BA_BV;
END_VAR
```

```

VAR
    Swi2P      : FB_BA_Swi2P;
END_VAR

```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the function block.
fIn	REAL	Actual value.
fOn	REAL	Dynamic switch-on point.
fOff	REAL	Dynamic switch-off point.

Outputs

Name	Type	Description
bQ	BOOL	Output of the current state of the hysteresis function.
bDst	BOOL	Display of a fault or the BV object is active. <i>bDst</i> is only active if the property <i>bEventDetectionEnable</i> of the BV object was set to TRUE. The monitoring of the binary feedback indicates a fault.

Inputs CONSTANT

Name	Type	Description
On	FB_BA_AV_Op [► 136]	AV object for input of the upper limit of the hysteresis function.
Off	FB_BA_AV_Op [► 136]	AV object for input of the lower limit of the hysteresis function.
Q	FB_BA_BV [► 147]	The binary object indicates the current state of the hysteresis function.

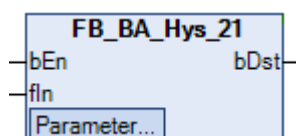
Variables

Name	Type	Description
Swi2P	FB_BA_Swi2P [► 388]	The function block Swi2P is the core of the hysteresis function.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.5.3 FB_BA_Hys_21



The template represents two hysteresis functions with fixed switching points.

It is used to monitor analog values, such as an air filter.

The hysteresis function *HysPreAlarm* triggers a pre-alarm. Maintenance can take place on an air filter, for example.

The hysteresis function *HysMainAlarm* triggers a main alarm and can be used as a fault that results in plant shutdown. The air filter must be serviced.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

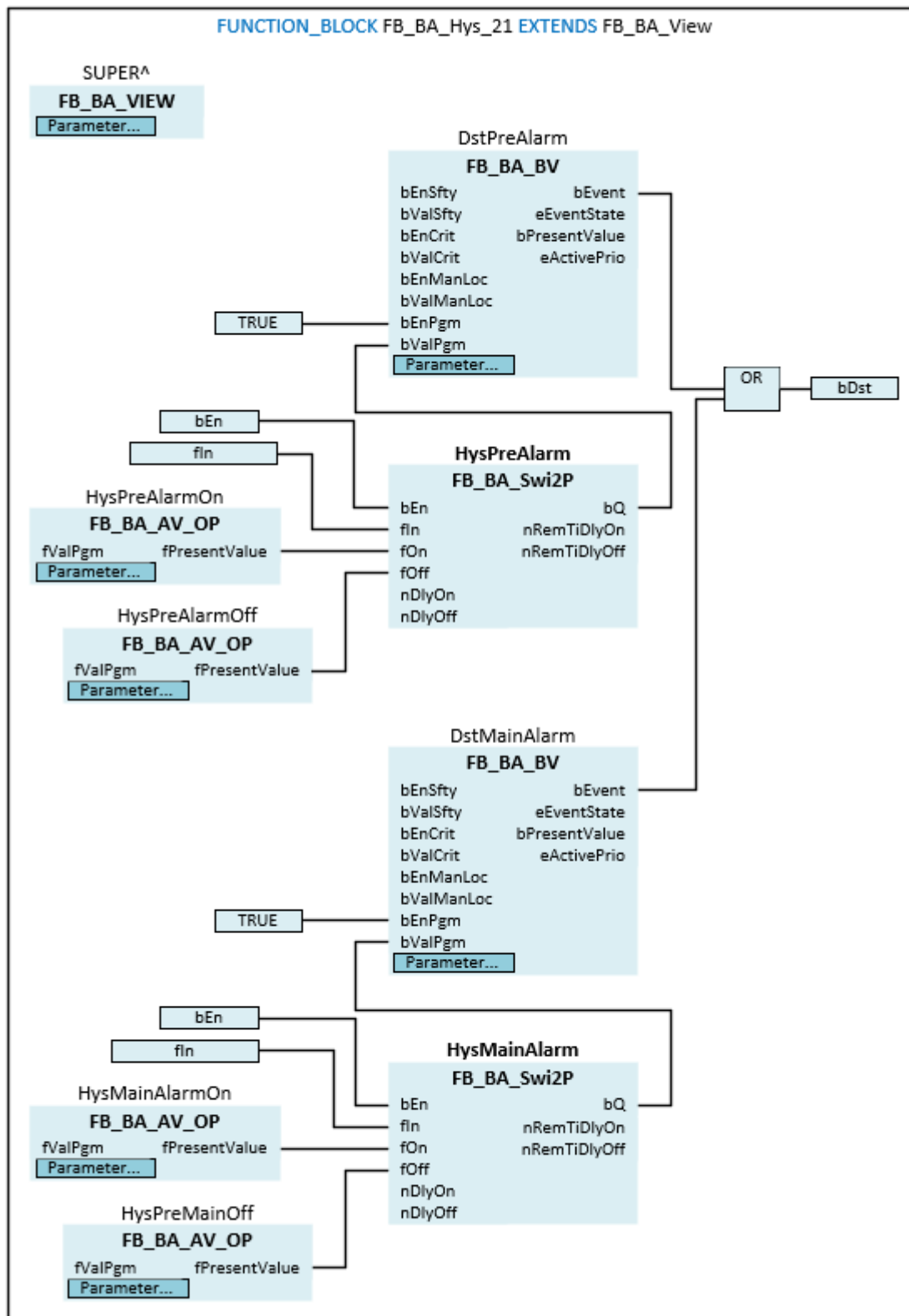
FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[► 195\]](#)

[FB_BA_View \[► 173\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_Hys_21 EXTENDS FB_BA_View
VAR_INPUT
    bEn          : BOOL;
    fln          : REAL;
END_VAR
VAR_OUTPUT
    bDst         : BOOL;
END_VAR
VAR_INPUT CONSTANT

```

```

HysPreAlarmOn      : FB_BA_AV_Op;
HysPreAlarmOff     : FB_BA_AV_Op;
HysMainAlarmOn     : FB_BA_AV_Op;
HysMainAlarmOff    : FB_BA_AV_Op;
DstPreAlarm        : FB_BA_BV;
DstMainAlarm       : FB_BA_BV;
END_VAR
VAR
  HysPreAlarm       : FB_BA_Swi2P;
  HysMainAlarm      : FB_BA_Swi2P;
END_VAR

```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the function block.
fIn	REAL	Actual value.

Outputs

Name	Type	Description
bDst	BOOL	Display of a fault or the BV object is active. <i>bDst</i> is only active if the property <i>bEventDetectionEnable</i> of the BV object was set to TRUE. The monitoring of the binary feedback indicates a fault.

Inputs CONSTANT

Name	Type	Description
HysPreAlarmOn	FB_BA_AV_Op [► 136]	AV object for entering the upper limit value of the hysteresis function <i>HysPreAlarm</i> .
HysPreAlarmOff	FB_BA_AV_Op [► 136]	AV object for entering the lower limit value of the hysteresis function <i>HysPreAlarm</i> .
HysMainAlarmOn	FB_BA_AV_Op [► 136]	AV object for entering the upper limit value of the hysteresis function <i>HysMainAlarm</i> .
HysMainAlarmOff	FB_BA_AV_Op [► 136]	AV object for entering the lower limit value of the hysteresis function <i>HysMainAlarm</i> .
DstPreAlarm	FB_BA_BV [► 147]	The binary object is used to indicate the pre-alarm of the hysteresis function <i>HysPreAlarm</i> . A message is triggered via Intrinsic Reporting.
DstMainAlarm	FB_BA_BV [► 147]	The binary object is used to indicate the main alarm of the hysteresis function <i>HysMainAlarm</i> . A message is triggered via Intrinsic Reporting. A fault that results in plant shutdown must be parameterized in the FB_init method via <i>eEnPlantLock</i> .

Variables

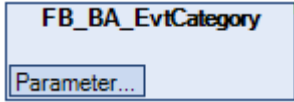
Name	Type	Description
HysPreAlarm	FB_BA_Swi2P [► 388]	The function block is the core of the hysteresis function pre-alarm.
HysMainAlarm	FB_BA_Swi2P [► 388]	The function block is the core of the hysteresis function main alarm.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.6 *EventClasses*

6.1.4.2.2.2.6.1 **FB_BA_EvtCategory**



The template contains a Notification Class Object.

Each BACnet object that is to generate messages by means of Intrinsic Reporting or Algorithmic Change Reporting must be assigned a Notification Class Object that contains the information for the distribution of the event messages.

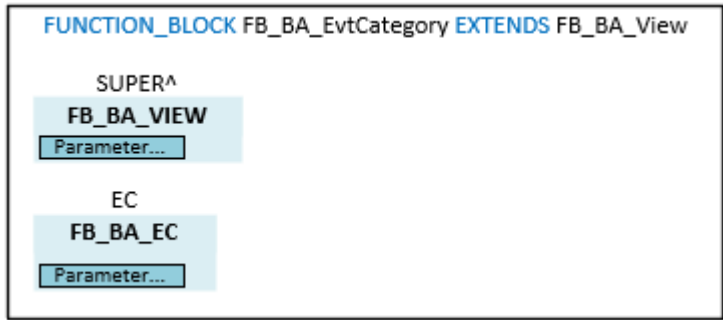
The Notification Class Object defines which priorities are assigned to the event messages, whether the events require acknowledgement and which recipients should receive the messages.

The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - FB_BA_Object [[▶ 195](#)]
 - FB_BA_View [[▶ 173](#)]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_EvtCategory EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    EC    : FB_BA_EC;
END_VAR
```

 Inputs **CONSTANT**

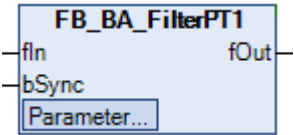
Name	Type	Description
EC	FB_BA_EC [▶ 152]	Notification Class Object.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.7 Filter

6.1.4.2.2.2.7.1 FB_BA_FilterPT1



First order filter.

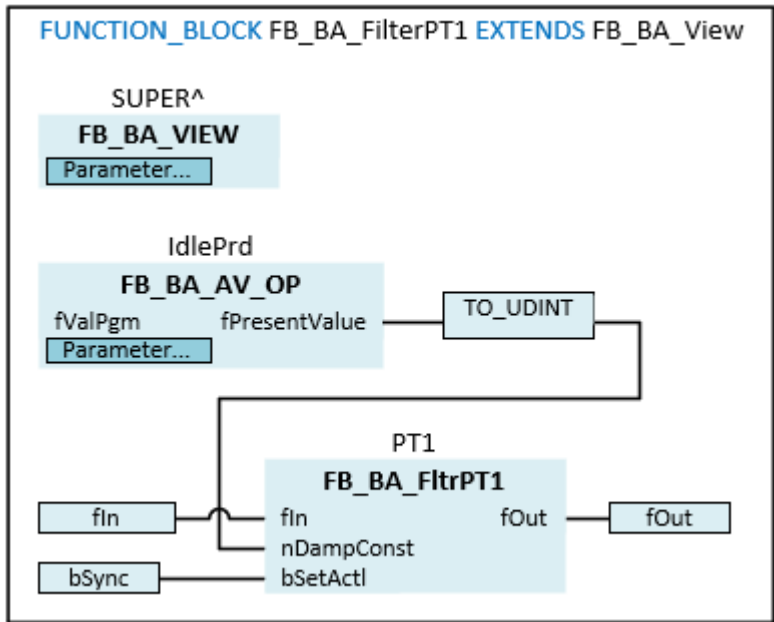


The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - FB_BA_Object [▶ 195]
 - FB_BA_View [▶ 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_FilterPT1 EXTENDS FB_BA_View
VAR_INPUT
    fln          : REAL;
    bSync        : BOOL;
END_VAR
VAR_OUTPUT
    fOut         : REAL;
END_VAR
```

```

VAR_INPUT CONSTANT
    DampingConstant      : FB_BA_AV_Op;
END_VAR
VAR
    PT1                  : FB_BA_FltrPT1;
END_VAR

```

Inputs

Name	Type	Description
fIn	REAL	Input signal.
bSync	BOOL	A rising edge at this input switches the output value <i>fOut</i> to the input value <i>fIn</i> .

Outputs

Name	Type	Description
fOut	REAL	Attenuated output signal.

Inputs CONSTANT

Name	Type	Description
DampingConstant	FB_BA_AV_Op [▸ 136]	AV object for entering filter time constant.

Variables

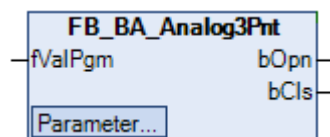
Name	Type	Description
PT1	FB_BA_FltrPT1	Notification Class Object.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.8 Analog3Point

6.1.4.2.2.2.8.1 FB_BA_Analog3Pnt



The template converts a continuous control signal for the positioning of a 3-point actuator into the binary switching commands Open/Close.



The initialization of the template takes place within the method FB_Init.

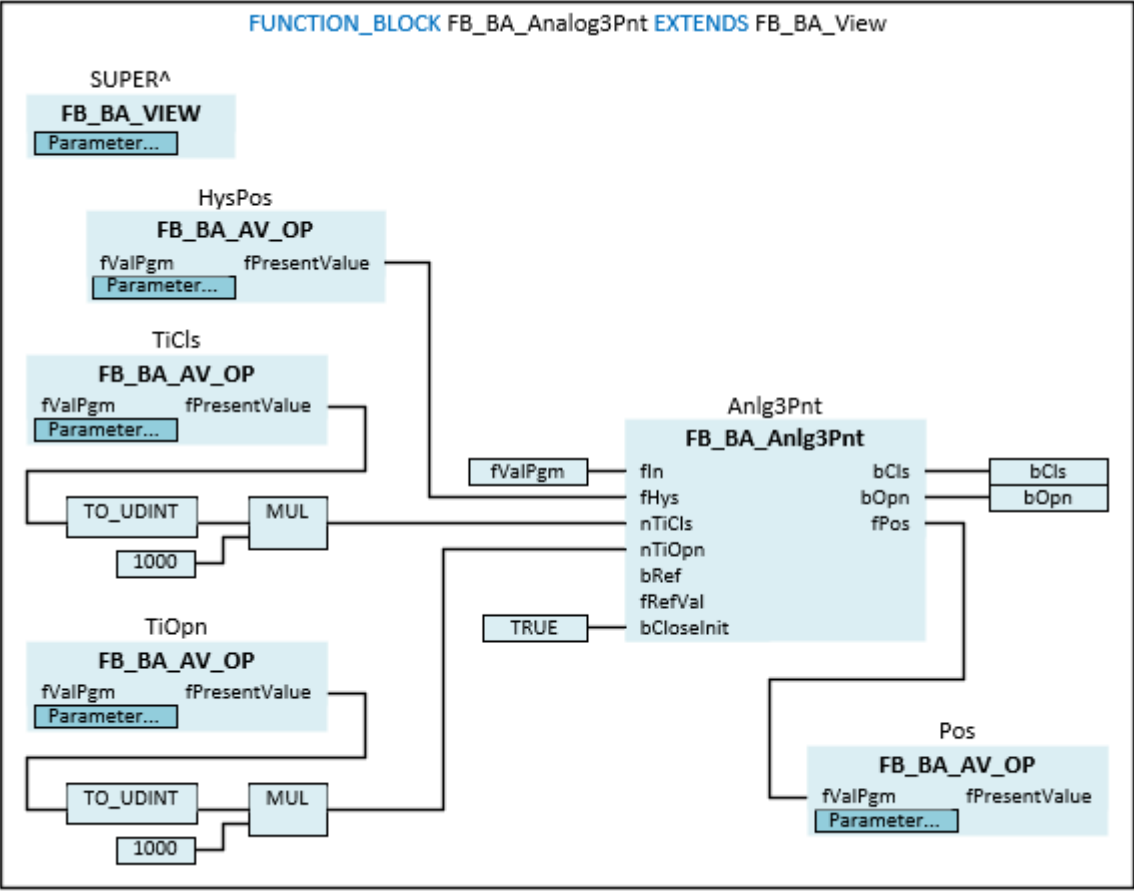
Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [[▸ 195](#)]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Analog3Pnt EXTENDS FB_BA_View
VAR_INPUT
    fValPgm      : REAL;
END_VAR
VAR_OUTPUT
    bOpn         : BOOL;
    bCls         : BOOL;
END_VAR
VAR_INPUT CONSTANT
    HysPos       : FB_BA_AV_Op;
    TiOpn        : FB_BA_AV_Op;
    TiCls        : FB_BA_AV_Op;
    Pos          : FB_BA_AV_Op;
END_VAR
VAR
    Anlg3Pnt     : FB_BA_Anlg3Pnt;
END_VAR
```

🔧 Inputs

Name	Type	Description
fValPgm	REAL	Control value for the position of the actuator.

🔌 Outputs

Name	Type	Description
bOpn	BOOL	Output for opening the actuator.
bCls	BOOL	Output for closing the actuator.

🚩 Inputs CONSTANT

Name	Type	Description
HysPos	FB_BA_AV_Op [▶ 136]	AV object for entering the hysteresis value to start the position change.
TiOpn	FB_BA_AV_Op	AV object for entering the opening time value.
TiCls	FB_BA_AV_Op	AV object for entering the closing time value.
Pos	FB_BA_AV_Op	Display of the calculated position.

VAR

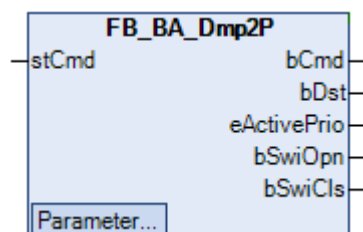
Name	Type	Description
Anlg3Pnt	FB_BA_Anlg3Pnt [▶ 346]	The function block is the core of the template and intended for controlling the three-point actuator. It converts an analog positioning signal into the binary open/close commands.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.3 Dampers

6.1.4.2.2.3.1 *FB_BA_Dmp2P*



The template is for the control of a two-point damper with integrated monitoring of both end positions.

It essentially consists of the base class [FB_BA_ActuatorCmd](#) [[▶ 716](#)] for controlling a binary aggregate and collecting all safety-related faults. The two templates *MonitOpn* and *MonitCls* are used to monitor the end positions.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

FB_BA_Base

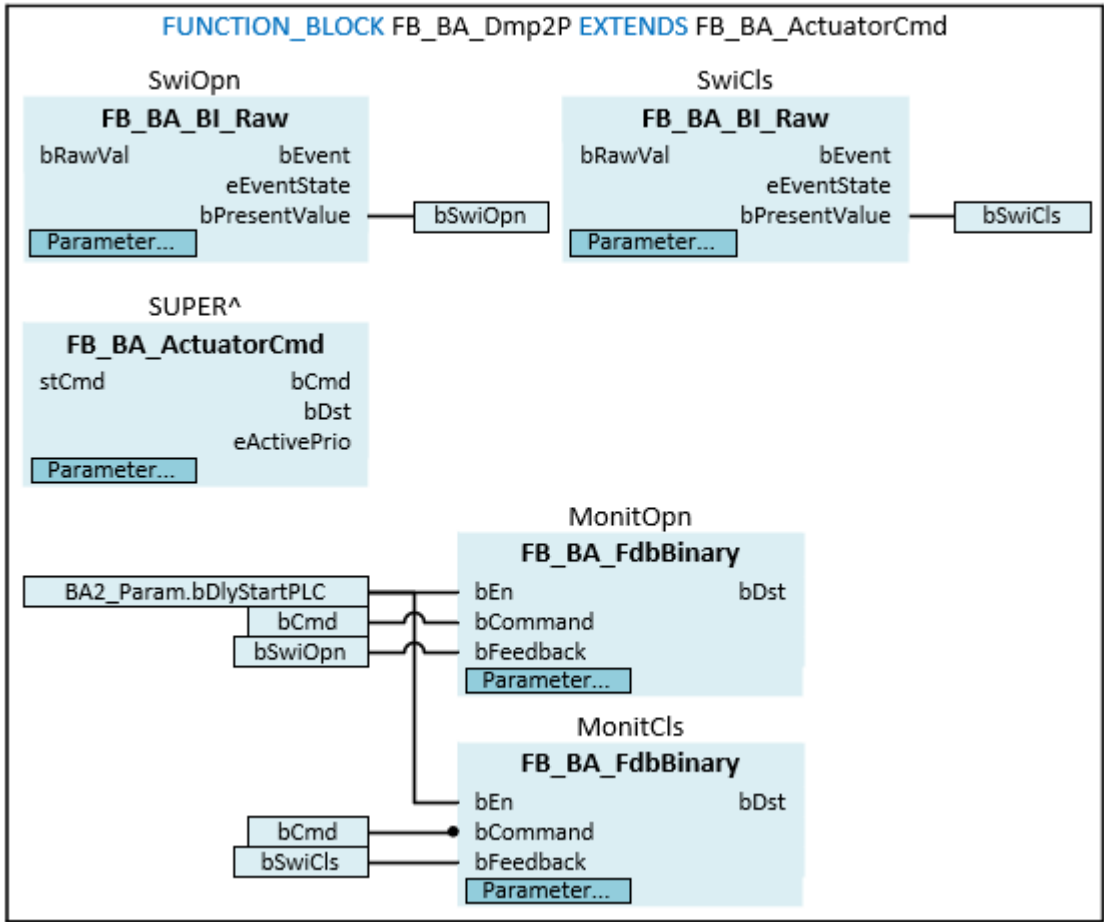
FB_BA_BasePublisher

FB_BA_Object [[▶ 195](#)]

FB_BA_View [[▶ 173](#)]

FB_BA_ActuatorCmd [[▶ 716](#)]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Dmp2P EXTENDS FB_BA_ActuatorCmd
VAR_OUTPUT
    bSwiOpn      : BOOL;
    bSwiCls      : BOOL;
END_VAR
VAR_INPUT CONSTANT
    SwiOpn       : FB_BA_BI_Raw;
    SwiCls       : FB_BA_BI_Raw;
    MonitOpen    : FB_BA_FdbBinary;
    MonitClose   : FB_BA_FdbBinary;
END_VAR
```

🔌 Outputs

Name	Type	Description
bSwiOpn	BOOL	End position "open" has been reached.
bSwiCls	BOOL	End position "closed" has been reached.

🔌 Inputs CONSTANT

Name	Type	Description
SwiOpn	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process the Open limit switch.
SwiCls	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process the Closed limit switch.
MonitOpen	FB_BA_FdbBinary	The template monitors the "Open" end position.
MonitClose	FB_BA_FdbBinary	The template monitors the "Closed" end position.

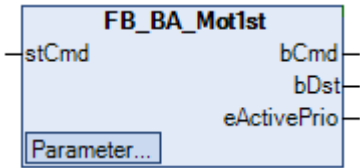
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.4 Motor

6.1.4.2.2.4.1 1st

6.1.4.2.2.4.1.1 FB_BA_Mot1st



The template is used for the control of a single-step motor with fault signal, e.g. a fan.

It essentially consists of the base class [FB_BA_ActuatorCmd](#) [► 716] for controlling a binary aggregate and collecting all safety-related faults. The BI object *Dst* indicates a motor fault.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

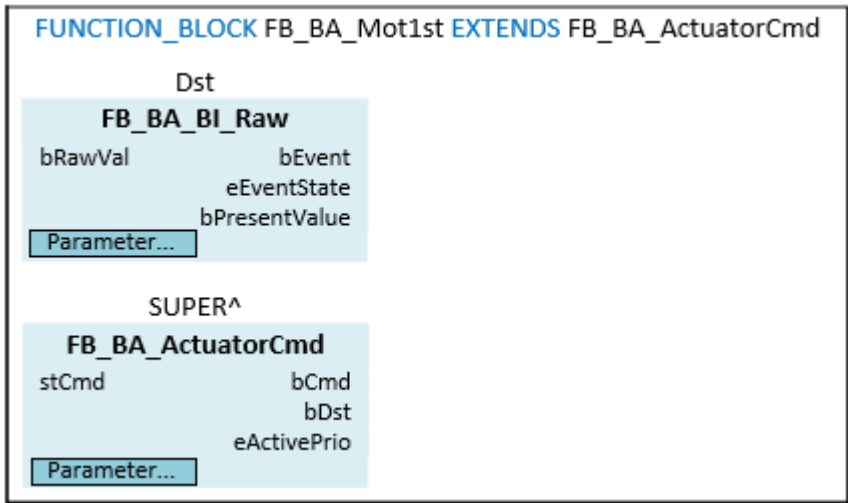
i

The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - [FB_BA_Object](#) [► 195]
 - [FB_BA_View](#) [► 173]
 - [FB_BA_ActuatorCmd](#) [► 716]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Mot1st EXTENDS FB_BA_ActuatorCmd
VAR_INPUT CONSTANT
    Dst      : FB_BA_BI_Raw;
END_VAR
VAR
END_VAR
```

🔧 Inputs **CONSTANT**

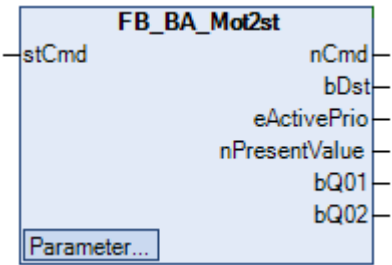
Name	Type	Description
Dst	FB_BA_BI_Raw [► 140]	Binary input object is used to process a fault.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.4.2 2st

6.1.4.2.2.4.2.1 FB_BA_Mot2st



The template is used for the control of a two-step motor with fault signal, e.g. a fan.

It mainly consists of the function block *StpCtrl* and the base class [FB_BA_ActuatorMO \[► 722\]](#) for the control of a multi-stage aggregate and the collection of all safety-relevant faults.

The BI objects *DstStp01* and *DstStp02* indicate faults in the respective steps.

The two templates *MoniFdbStp01* and *MoniFdbStp02* are used for feedback monitoring of steps 1 + 2, *FdbStp01* and *FdbStp02*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

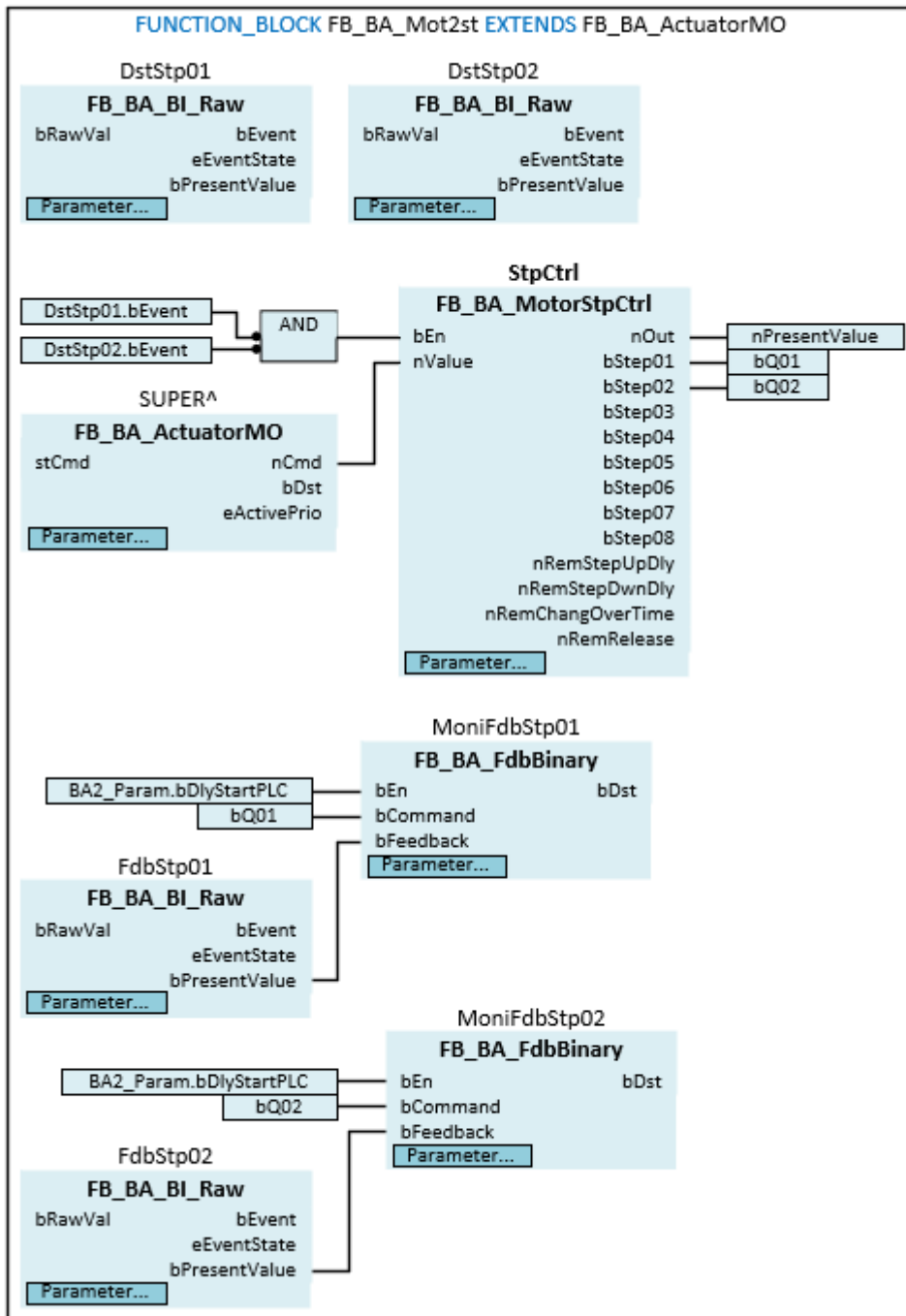


The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - [FB_BA_Object \[► 195\]](#)
 - [FB_BA_View \[► 173\]](#)
 - [FB_BA_ActuatorMO \[► 722\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_Mot2st EXTENDS FB_BA_ActuatorMO
VAR_OUTPUT
    nPresentValue    : UDINT;
    bQ01             : BOOL;
    bQ02             : BOOL;
END_VAR
VAR_INPUT CONSTANT
    DstStp01         : FB_BA_BI_Raw;
    DstStp02         : FB_BA_BI_Raw;
    FdbStp01         : FB_BA_BI_Raw;
    FdbStp02         : FB_BA_BI_Raw;
    MoniFdbStp01     : FB_BA_FdbBinary;
    MoniFdbStp02     : FB_BA_FdbBinary;
    StpCtrl           : FB_BA_MotorStpCtrl;
END_VAR

```

🔌 Outputs

Name	Type	Description
nPresentValue	UDINT	Current control step of the motor.
bQ01	BOOL	Variable for controlling step 1 of the motor. This variable must be linked to a bus terminal.
bQ02	BOOL	Variable for controlling step 2 of the motor. This variable must be linked to a bus terminal.

🔌 Inputs CONSTANT

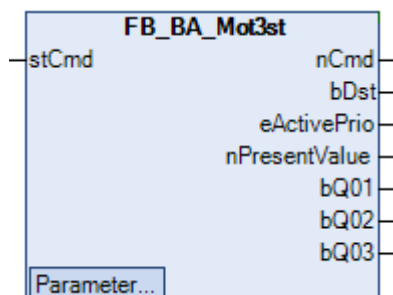
Name	Type	Description
DstStp01	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process the step 1 motor fault.
DstStp02	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process the step 2 motor fault.
FdbStp01	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process the feedback of step 1 of the motor.
FdbStp02	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process the feedback of step 2 of the motor.
MoniFdbStp01	FB_BA_FdbBinary	Template that monitors the feedback of step 1 of the motor.
MoniFdbStp02	FB_BA_FdbBinary	Template that monitors the feedback of step 2 of the motor.
StpCtrl	FB_BA_MotorStpCtrl [▶ 351]	The function block <i>StpCtrl</i> receives the numerical switch value from the base class <i>FB_BA_ActuatorMO</i> [▶ 722] and converts the switch value into individual control steps.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.4.3 3st

6.1.4.2.2.4.3.1 FB_BA_Mot3st



The template is used for the control of a three-step motor with fault signal, e.g. a fan.

It mainly consists of the function block *StpCtrl* and the base class *FB_BA_ActuatorMO* [▶ 722] for the control of a multi-stage aggregate and the collection of all safety-relevant faults.

The BI objects *DstStp01*, *DstStp02* and *DstStp03* indicate faults in the respective steps.

The two templates *MoniFdbStp01*, *MoniFdbStp02* and *MoniFdbStp03* are used for feedback monitoring of steps 1 + 2 + 3, *FdbStp01*, *FdbStp02* and *FdbStp03*.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

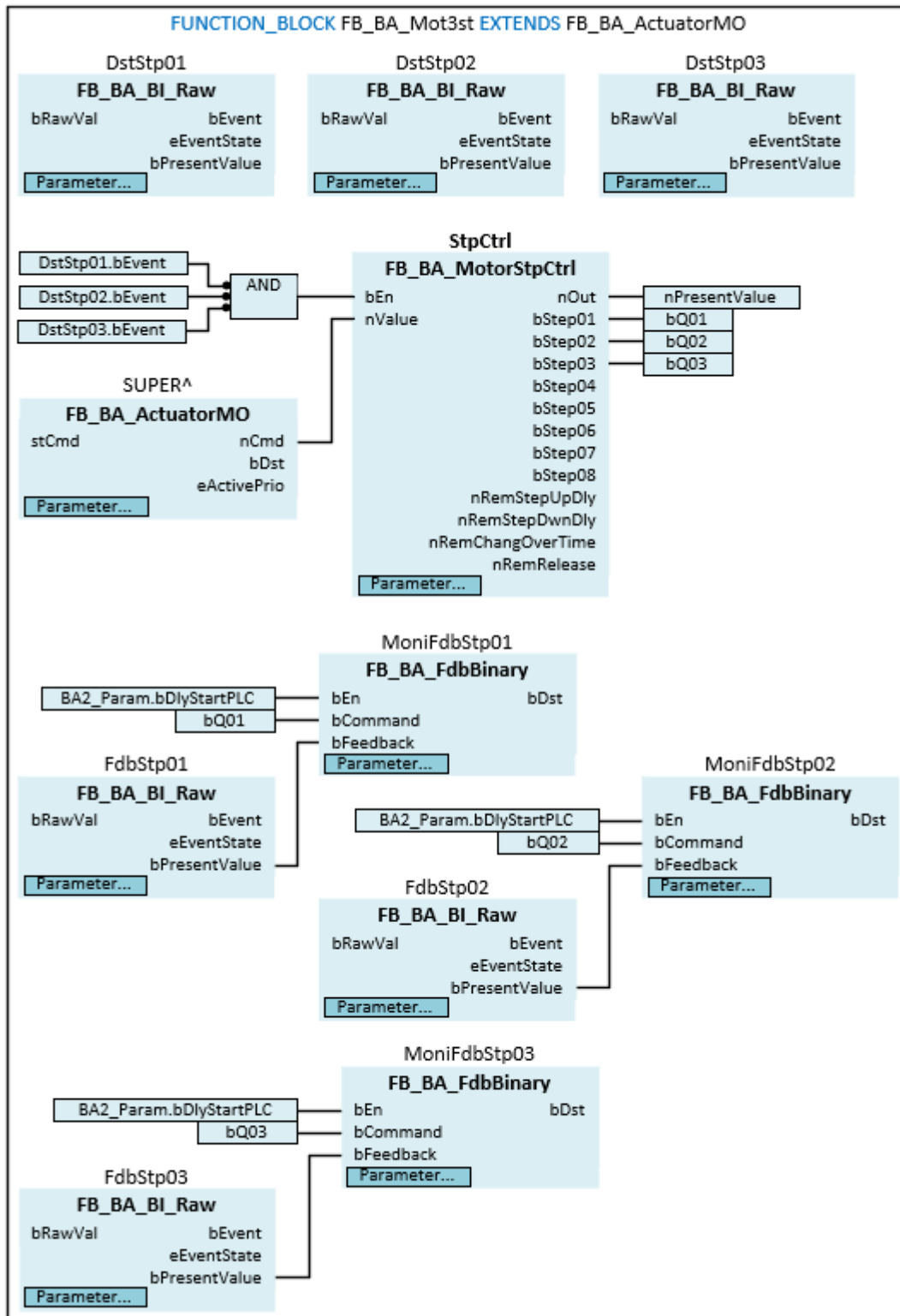
FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA View \[► 173\]](#)

[FB BA ActuatorMO \[► 722\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_Mot3st EXTENDS FB_BA_ActuatorMO
VAR_OUTPUT
    nPresentValue    : UDINT;
    bQ01             : BOOL;
    bQ02             : BOOL;
    bQ03             : BOOL;
END_VAR
VAR_INPUT CONSTANT
    DstStp01         : FB_BA_BI_Raw;
    DstStp02         : FB_BA_BI_Raw;
    DstStp03         : FB_BA_BI_Raw;

```

```

FdbStp01      : FB_BA_BI_Raw;
FdbStp02      : FB_BA_BI_Raw;
FdbStp03      : FB_BA_BI_Raw;
MoniFdbStp01  : FB_BA_FdbBinary;
MoniFdbStp02  : FB_BA_FdbBinary;
MoniFdbStp03  : FB_BA_FdbBinary;
StpCtrl       : FB_BA_MotorStpCtrl;
END_VAR

```

Outputs

Name	Type	Description
nPresentValue	UDINT	Current control step of the motor.
bQ01	BOOL	Variable for controlling step 1 of the motor. This variable must be linked to a bus terminal.
bQ02	BOOL	Variable for controlling step 2 of the motor. This variable must be linked to a bus terminal.

Inputs CONSTANT

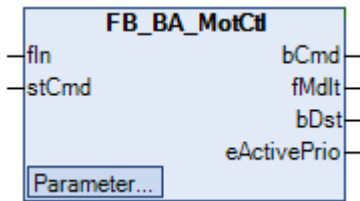
Name	Type	Description
DstStp01	FB_BA_BI_Raw [► 140]	Binary input object is used to process the step 1 motor fault.
DstStp02	FB_BA_BI_Raw [► 140]	Binary input object is used to process the step 2 motor fault.
DstStp03	FB_BA_BI_Raw [► 140]	Binary input object is used to process the step 3 motor fault.
FdbStp01	FB_BA_BI_Raw [► 140]	Binary input object is used to process the feedback of step 1 of the motor.
FdbStp02	FB_BA_BI_Raw [► 140]	Binary input object is used to process the feedback of step 2 of the motor.
FdbStp03	FB_BA_BI_Raw [► 140]	Binary input object is used to process the feedback of step 3 of the motor.
MoniFdbStp01	FB_BA_FdbBinary	Template that monitors the feedback of step 1 of the motor.
MoniFdbStp02	FB_BA_FdbBinary	Template that monitors the feedback of step 2 of the motor.
MoniFdbStp03	FB_BA_FdbBinary	Template that monitors the feedback of step 3 of the motor.
StpCtrl	FB_BA_MotorStpCtrl [► 351]	The function block <i>StpCtrl</i> receives the numerical switch value from the base class FB_BA_ActuatorMO [► 722] and converts the switch value into individual control steps.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.4.4 Control

6.1.4.2.2.4.4.1 FB_BA_MotCtl



The template is used to control and regulate a speed-controlled motor.

It mainly consists of a BO and AO object for controlling the motor (frequency converter), a BI object for displaying a fault and a PID controller for speed control. The function block *PlantLock* collects all safety-relevant faults. The enables and modulation commands are transmitted to the template via the command structure *stCmd*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

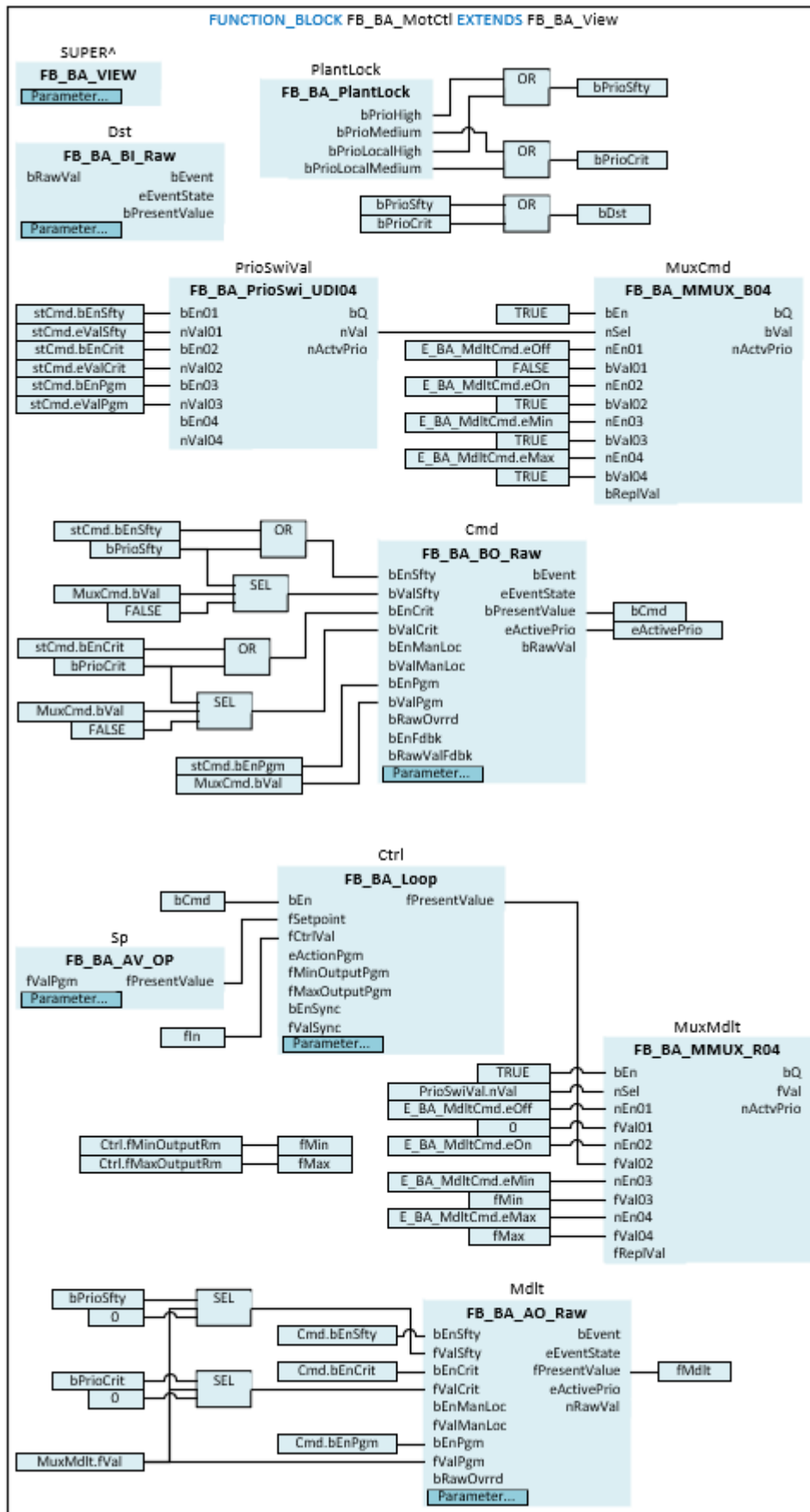
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_MotCtl EXTENDS FB_BA_View
VAR_INPUT
    fln          : REAL;
    stCmd        : ST_BA_Mdlt;
END_VAR
VAR_OUTPUT
    bCmd        : BOOL;
    fMdlt       : REAL;
    bDst        : BOOL;
    eActivePrio : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
    Dst      : FB_BA_BI_Raw;
    Cmd      : FB_BA_BO_Raw;
    Mdlt     : FB_BA_AO_Raw;
    Sp       : FB_BA_AV_Op;
    Ctrl     : FB_BA_Loop;
    PlantLock : FB_BA_PlantLock;
END_VAR
VAR
    bPrioSfty : BOOL;
    bPrioCrit : BOOL;
    PrioSwiVal : FB_BA_PrioSwi_UDI04;
    MuxCmd     : FB_BA_MMUX_B04;
    MuxMdlt    : FB_BA_MMUX_R04;
    fMax       : REAL;
    fMin       : REAL;
END_VAR

```

Inputs

Name	Type	Description
fln	REAL	The actual value for the controller <i>Ctrl</i> is connected to the input.
stCmd	ST_BA_Mdlt [▶ 221]	The enables and modulation commands are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority determines the switching command at the binary output object <i>Cmd</i> and the control command at the analog output object <i>Mdlt</i> .

Outputs

Name	Type	Description
bCmd	BOOL	Output of the switch value.
fMdlt	REAL	Current value of the analog output object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [▶ 96]	Display of the active priority.

🚩 Inputs CONSTANT

Name	Type	Description
Dst	FB_BA_BI_Raw [► 140]	Binary input object is used to process a fault.
Cmd	FB_BA_BO_Raw [► 145]	The binary output object is used to output a switching command and transmit it to the I/O level.
Mdlt	FB_BA_AO_Raw [► 133]	Current value of the analog output object.
Sp	FB_BA_AV_Op [► 136]	Entering the setpoint for the PID controller <i>Ctrl</i> .
Ctrl	FB_BA_Loop [► 154]	PID controller.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

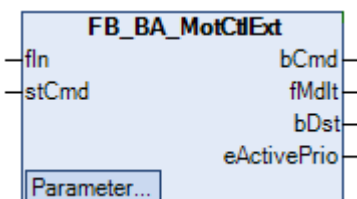
Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [► 369]	The priority switch <i>PrioSwiVal</i> [► 369] uses the command structure <i>stCmd</i> to determine the modulation command for the multiplexers <i>MuxCmd</i> and <i>MuxMdlt</i> .
MuxCmd	FB_BA_MMUX_B04 [► 364]	The multiplexer <i>MuxCmd</i> [► 364] determines the current switch value from the command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the binary output object <i>Cmd</i> .
MuxMdlt	FB_BA_MMUX_R04 [► 364]	The multiplexer <i>MuxMdlt</i> [► 364] determines the current control value from the modulation values <i>Ctrl.fPresentValue</i> , <i>fMin</i> , <i>fMax</i> and the modulation command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the analog output object <i>Mdlt</i> .
fMax	REAL	Maximum value of the controller, which is output when the modulation command <i>E_BA_MdltCmd.eMax</i> is pending (see <i>E_BA_Mdlt</i> [► 214]).
fMin	REAL	Minimum value of the controller, which is output when the modulation command <i>E_BA_MdltCmd.eMin</i> is pending (see <i>E_BA_Mdlt</i> [► 214]).

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.4.4.2 FB_BA_MotCtlExt



The template is used to control and regulate a speed-controlled motor. It consists essentially of the base class *FB_BA_MotCtl* [► 763].

The difference to the template [FB_BA_MotCtl](#) [[▶ 763](#)] are the additional BI objects *ThOvrl*d and *MntnSwi*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

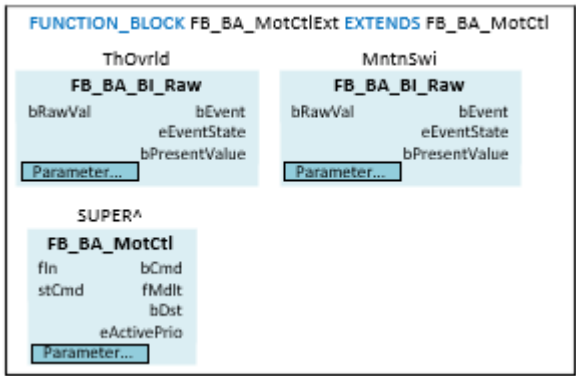


The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - [FB_BA_Object](#) [[▶ 195](#)]
 - [FB_BA_View](#) [[▶ 173](#)]
 - [FB_BA_MotCtl](#) [[▶ 763](#)]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_MotCtlExt EXTENDS FB_BA_MotCtl
VAR_INPUT CONSTANT
    ThOvrl    : FB_BA_BI_Raw;
    MntnSwi   : FB_BA_BI_Raw;
END_VAR
```

🔧 Inputs CONSTANT

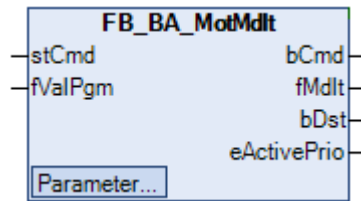
Name	Type	Description
ThOvrl	FB_BA_BI_Raw [▶ 140]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 140]	The binary input object is used to process a maintenance switch.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.4.5 Modulation

6.1.4.2.2.4.5.1 FB_BA_MotMdl



The template is used to control a speed-controlled motor.

It mainly consists of a BO and AO object for controlling the motor (frequency converter) and a BI object for displaying a fault. The function block *PlantLock* collects all safety-relevant faults. The enables and modulation commands are transmitted to the template via the command structure *stCmd*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

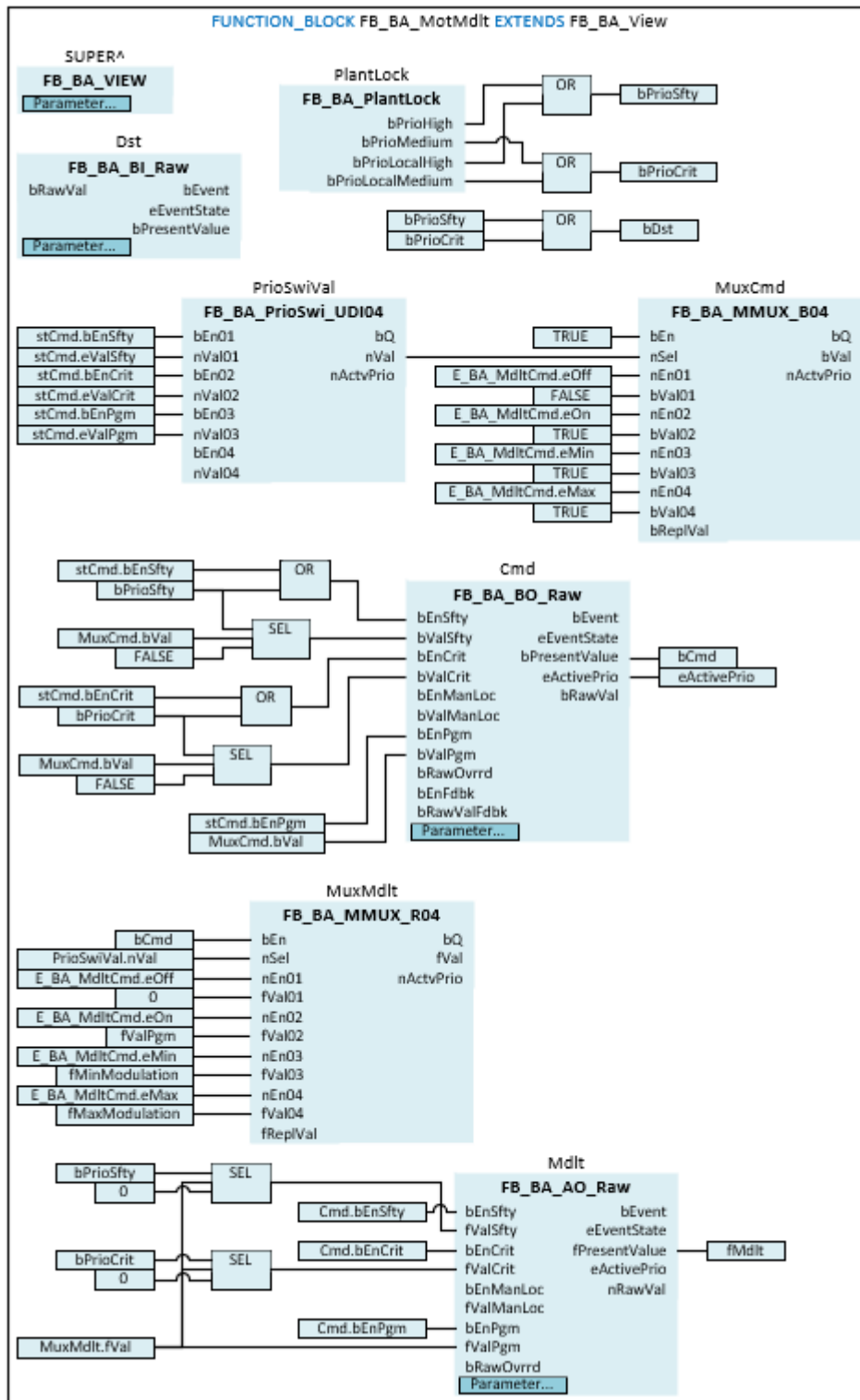
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [[▶ 195](#)]

FB_BA_View [[▶ 173](#)]

Block diagram



Illustration

FUNCTION_BLOCK FB_BA_MotCtl EXTENDS FB_BA_View

VAR_INPUT

stCmd : ST_BA_Mdlr;

fValPgm : REAL;

END_VAR

VAR_INPUT CONSTANT

Dst : FB_BA_BI_Raw;

Cmd : FB_BA_BO_Raw;

Mdlr : FB_BA_AO_Raw;

PlantLock : FB_BA_PlantLock;

END_VAR

VAR_INPUT CONSTANT PERSISTENT

```

{attribute 'parameterCategory' := 'Behaviour'}
fMinModulation    : REAL := 20;
{attribute 'parameterCategory' := 'Behaviour'}
fMaxModulation    : REAL := 100;
END_VAR
VAR_OUTPUT
  bCmd             : BOOL;
  fMdlT            : REAL;
  bDst             : BOOL;
  eActivePrio      : E_BA_Priority;
END_VAR
VAR
  bPrioSfty        : BOOL;
  bPrioCrit        : BOOL;
  PrioSwiVal       : FB_BA_PrioSwi_UDI04;
  MuxCmd           : FB_BA_MMUX_B04;
  MuxMdlT          : FB_BA_MMUX_R04;
END_VAR

```

Inputs

Name	Type	Description
stCmd	ST_BA_MdlT [► 221]	The enables and modulation commands are transmitted to the template via the command structure <i>stCmd</i> . The command with the highest priority determines the switching command at the binary output object <i>Cmd</i> and the control command at the analog output object <i>MdlT</i> .
fValPgm	REAL	Control value for the control of the motor.

Inputs CONSTANT

Name	Type	Description
Dst	FB_BA_BI_Raw [► 140]	Binary input object is used to process a fault.
Cmd	FB_BA_MO [► 165]	The multistate output object is used to output the current switch value.
MdlT	FB_BA_AO_Raw [► 133]	Current value of the analog output object.
PlantLock	FB_BA_PlantLock [► 121]	The function block <i>PlantLock</i> collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Inputs CONSTANT PERSISTENT

Name	Type	Description
fMinModulation	REAL	Constant minimum value which is output if the modulation command <i>E_BA_MdlTCmd.eMin</i> is pending (see E_BA_MdlT [► 214]).
fMaxModulation	REAL	Constant maximum value which is output if the modulation command <i>E_BA_MdlTCmd.eMax</i> is pending (see E_BA_MdlT [► 214]).

Outputs

Name	Type	Description
bCmd	BOOL	Output of the switch value.
fMdlT	REAL	Current value of the analog output object.
bDst	BOOL	The variable is an evaluation of the lock priorities "Safety" and "Critical" of the project structure and indicates a triggered event.
eActivePrio	E_BA_Priority [► 96]	Display of the active priority.

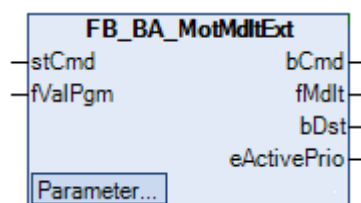
Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.
bPrioSwiVal	FB BA PrioSwi UDI04 [► 369]	The priority switch PrioSwiVal [► 369] uses the command structure <i>stCmd</i> to determine the modulation command for the multiplexers <i>MuxCmd</i> and <i>MuxMdl</i> .
MuxCmd	FB BA MMUX B04 [► 364]	The multiplexer MuxCmd [► 364] determines the current switch value from the command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the binary output object <i>Cmd</i> .
MuxMdl	FB BA MMUX R04 [► 364]	The multiplexer MuxMdl [► 364] determines the current control value from the modulation values <i>Ctrl.fPresentValue</i> , <i>fMin</i> , <i>fMax</i> and the modulation command of the priority switch <i>PrioSwiVal</i> . The resulting output is sent to the analog output object <i>Mdl</i> .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.4.5.2 FB_BA_MotMdlExt



The template is used to control a speed-controlled motor. It consists essentially of the base class [FB BA MotMdl \[► 768\]](#).

The difference to the template [FB BA MotMdl \[► 768\]](#) is the AV object *Sp* with the associated MAX selection. A minimum control value for the motor could thus be specified via the *Sp* object.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

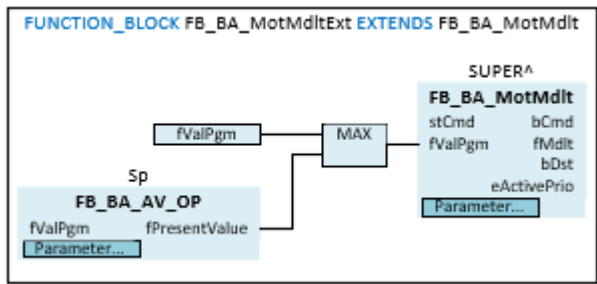
FB_BA_BasePublisher

FB BA Object [► 195]

FB BA View [► 173]

FB BA MotMdl [► 768]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_MotCtlExt EXTENDS FB_BA_MotMdl
VAR_INPUT CONSTANT
    Sp      : FB_BA_AV_Op;
END_VAR
```

🔧 Inputs CONSTANT

Name	Type	Description
Sp	FB_BA_AV_Op [► 136]	The AV object can be used to specify the minimum control value for the motor. A MAX selection then uses either this setpoint or the value from the input <i>fValPgm</i> of the base class <i>FB_BA_MotMdl</i> [► 768].

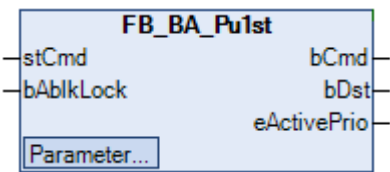
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.5 Pump

6.1.4.2.2.5.1 1st

6.1.4.2.2.5.1.1 FB_BA_Pu1st



The template is used to control a single-stage pump with binary inputs and outputs. It mainly consists of the base class *FB_BA_Mot1st* [► 756], the switch-off delay *DlyOff* and the anti-blocking protection function *Ablk* [► 739].

The pump is switched on externally by the priorities of the command structure *stCmd* of the base class *FB_BA_Mot1st* [► 756] or internally by the blocking protection function *Ablk* [► 739]. The external request via the priority program is switched off with a delay by the template *DlyOff*.

i The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

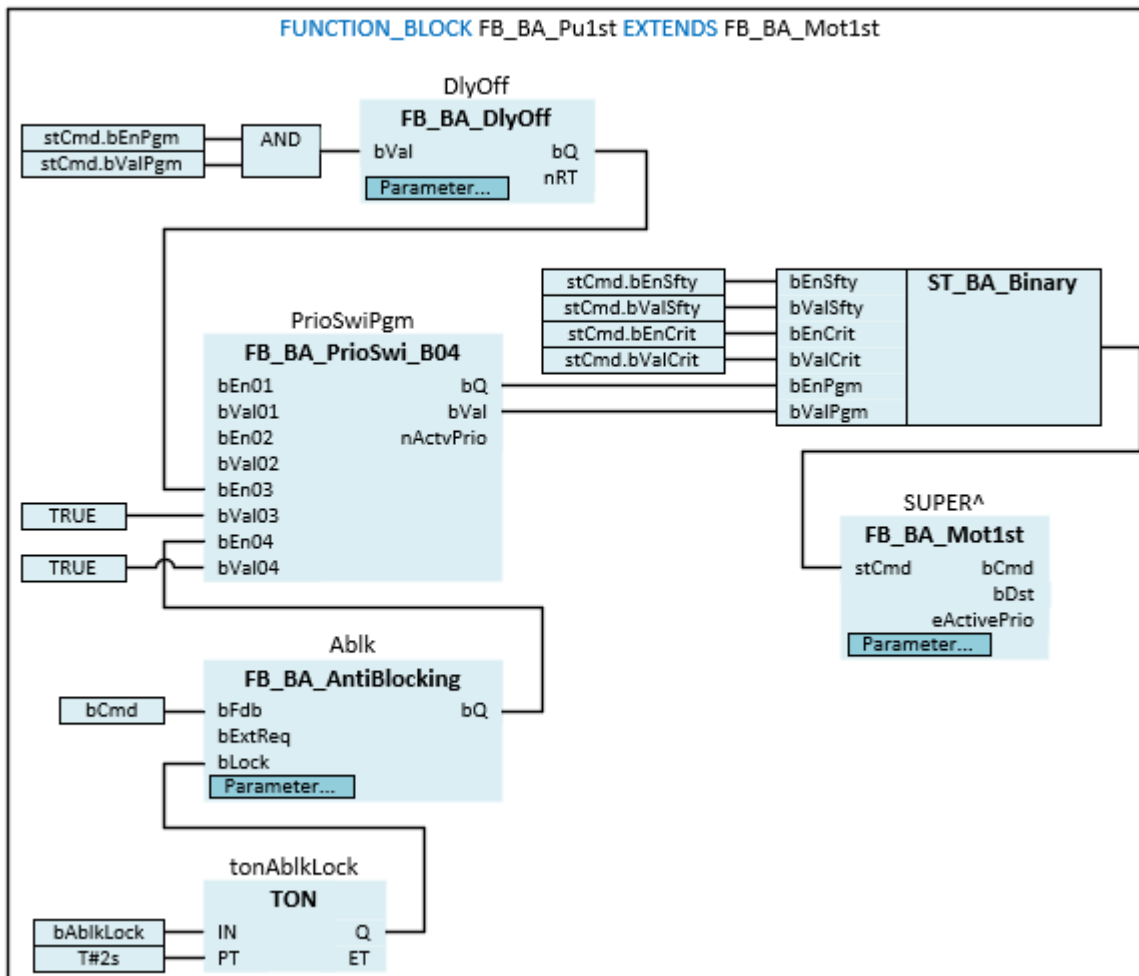
FB_BA_Object [► 195]

FB_BA_View [► 173]

FB_BA_ActuatorCmd [► 716]

FB_BA_Mot1st [► 756]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Pu1st EXTENDS FB_BA_Mot1st
VAR_INPUT
    bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
    DlyOff         : FB_BA_DlyOff;
    Ablk           : FB_BA_AntiBlocking;
END_VAR
VAR
    PrioSwiPgm     : FB_BA_PrioSwi_B04;
    tonAblkLock     : TON;
END_VAR
```

Inputs

Name	Type	Description
bAblkLock	BOOL	A TRUE at this input variable interrupts the blocking protection function Ablk [► 739]. It should be prevented that pumps and valves get a blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
DlyOff	FB_BA_DlyOff	The template serves as a delayed pump shutdown.
Ablk	FB_BA_AntiBlocking [► 739]	Anti-blocking protection.

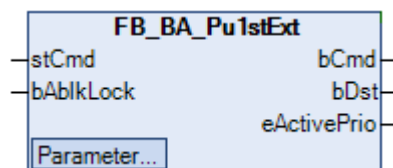
Variables

Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_B04	The priority switch <i>PrioSwiPgm</i> uses the switch-off delay DlyOff, the anti-blocking protection function Ablk [► 739] and the priority program of the command structure <i>stCmd</i> of the base class FB_BA_Mot1st [► 756] to determine the current switch value for the base class FB_BA_Mot1st [► 756].
tonAblkLock	TON	Switch-off delay of the blocking protection pulse.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.5.1.2 FB_BA_Pu1stExt



The template is used to control a single-stage pump with binary inputs and outputs. It consists essentially of the base class [FB_BA_Pu1st](#) [► 772].

The difference to the template [FB_BA_Pu1st](#) [► 772] are the additional BI objects *ThOvrl*d and *MntnSwi*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method [FB_Init](#).

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

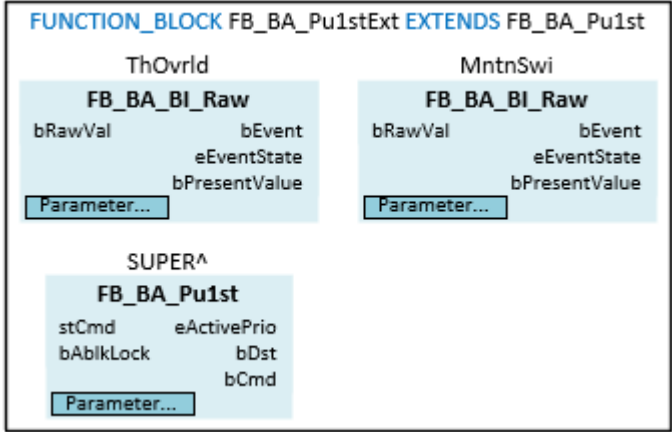
[FB_BA_Object \[► 195\]](#)

[FB_BA_View \[► 173\]](#)

[FB_BA_ActuatorCmd \[► 716\]](#)

[FB_BA_Mot1st \[► 756\]](#)

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Pu1stExt EXTENDS FB_BA_Pu1st
VAR_INPUT CONSTANT
    ThOvrl'd      : FB_BA_BI_Raw;
    MntnSwi       : FB_BA_BI_Raw;
END_VAR
```

🔧 Inputs CONSTANT

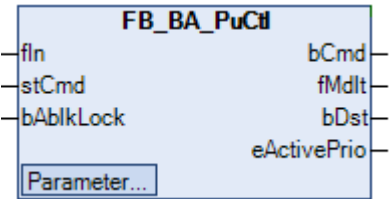
Name	Type	Description
ThOvrl'd	FB_BA_BI_Raw [► 140]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [► 140]	The binary input object is used to process a maintenance switch.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.5.2 Control

6.1.4.2.2.5.2.1 FB_BA_PuCtl



The template is used to control and regulate a speed-controlled pump with analog and binary inputs and outputs. It mainly consists of the base class [FB_BA_MotCtl](#) [► 763], the switch-off delay [DlyOff](#) and the anti-blocking protection function [Ablk](#) [► 739]. The template *FlowMonit* stands for the monitoring of the flow monitor *FlowSwi*.

The pump is switched on externally by the priorities of the command structure *stCmd* of the base class [FB_BA_MotCtl](#) [► 763] or internally by the blocking protection function [Ablk](#) [► 739]. The external request via the priority "Program" is switched off with a delay by the template [DlyOff](#).

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method [FB_Init](#).

Inheritance hierarchy

FB_BA_Base

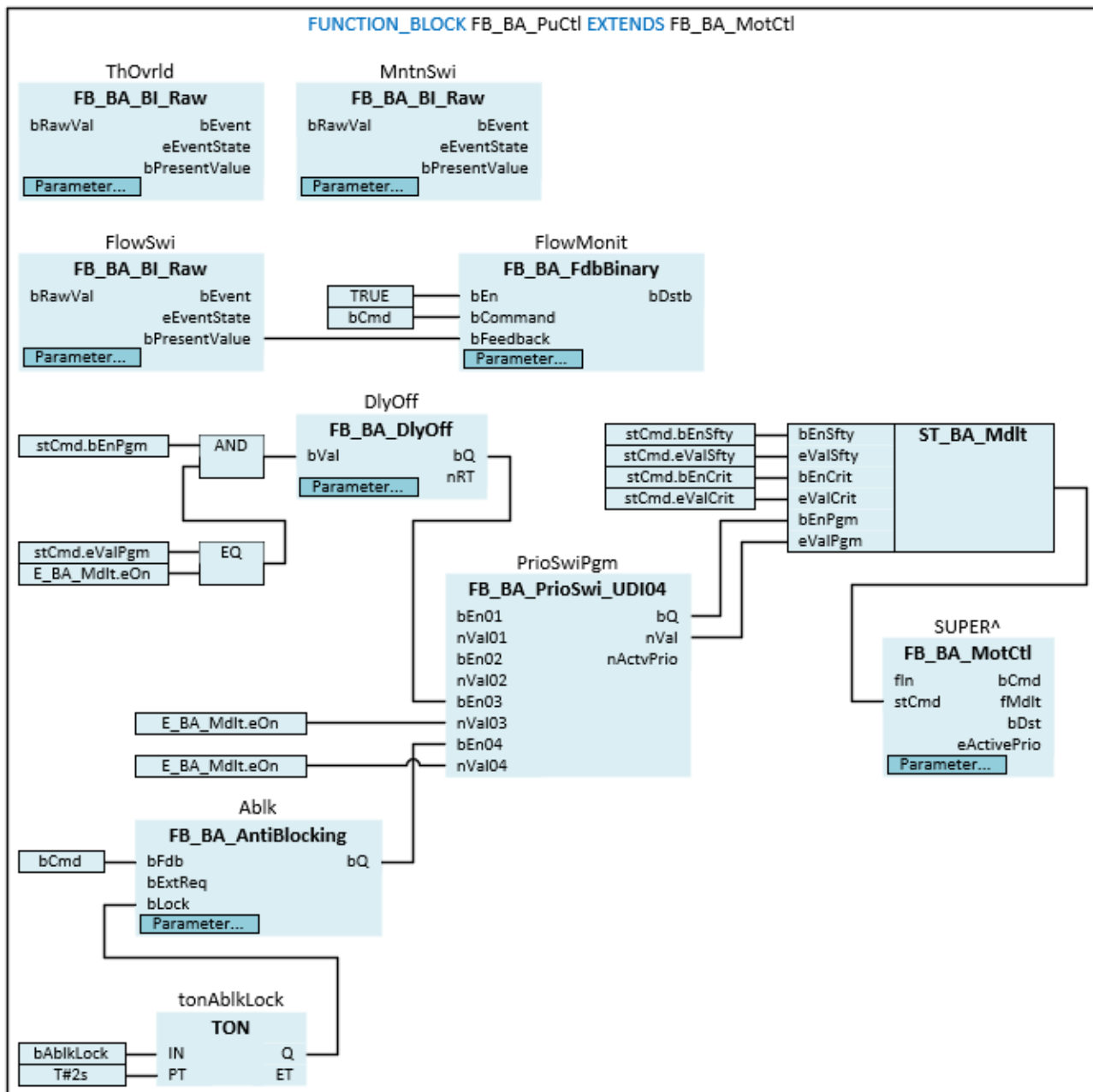
FB_BA_BasePublisher

[FB_BA_Object](#) [► 195]

[FB_BA_View](#) [► 173]

[FB_BA_MotCtl](#) [► 763]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_PuCtl EXTENDS FB_BA_MotCtl
VAR_INPUT
    bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
    ThOvrlid      : FB_BA_BI_Raw;
    MntnSwi       : FB_BA_BI_Raw;
    FlowSwi       : FB_BA_BI_Raw;
    FlowMonit     : FB_BA_FdbBinary;
    DlyOff        : FB_BA_DlyOff;
    Ablk          : FB_BA_AntiBlocking;
END_VAR
VAR
    PrioSwiPgm    : FB_BA_PrioSwi_UDI04;
    tonAblkLock    : TON;
END_VAR

```

Inputs

Name	Type	Description
bAblkLock	BOOL	A TRUE at this input variable interrupts the blocking protection function Ablk [► 739]. It should be prevented that pumps and valves get a blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
ThOvrd	FB_BA_BI_Raw [► 140]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [► 140]	The binary input object is used to process a maintenance switch.
FlowSwi	FB_BA_BI_Raw [► 140]	The binary input object is used to process a flow monitor.
FlowMonit	FB_BA_FdbBinary	Template for monitoring the flow monitor.
DlyOff	FB_BA_DlyOff	The template serves as a delayed pump shutdown.
Ablk	FB_BA_AntiBlocking [► 739]	Anti-blocking protection.

Variables

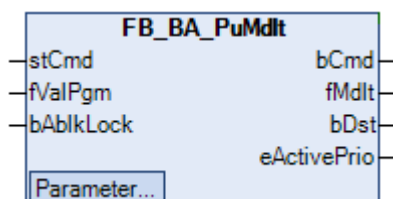
Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_UDI04	The priority switch <i>PrioSwiPgm</i> uses the switch-off delay <i>DlyOff</i> , the anti-lock protection function Ablk [► 739] and the priority program of the command structure <i>stCmd</i> of the base class FB_BA_MotCtl [► 763] to determine the current modulation value for the base class FB_BA_MotCtl [► 763].
tonAblkLock	TON	Switch-off delay of the blocking protection pulse.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.5.3 Modulation

6.1.4.2.2.5.3.1 FB_BA_PuMdlIt



The template is used to control a speed-controlled pump with analog and binary inputs and outputs. It mainly consists of the base class [FB_BA_MotMdlIt](#) [► 768], the switch-off delay *DlyOff* and the anti-blocking protection function [Ablk](#) [► 739]. The template *FlowMonit* stands for the monitoring of the flow monitor *FlowSwi*.

The pump is switched on externally via the priorities of the command structure *stCmd* of the base class [FB BA MotMdl \[► 768\]](#) or internally by the blocking protection function [Abk \[► 739\]](#). The external request via the priority program is switched off with a delay by the template *DlyOff*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

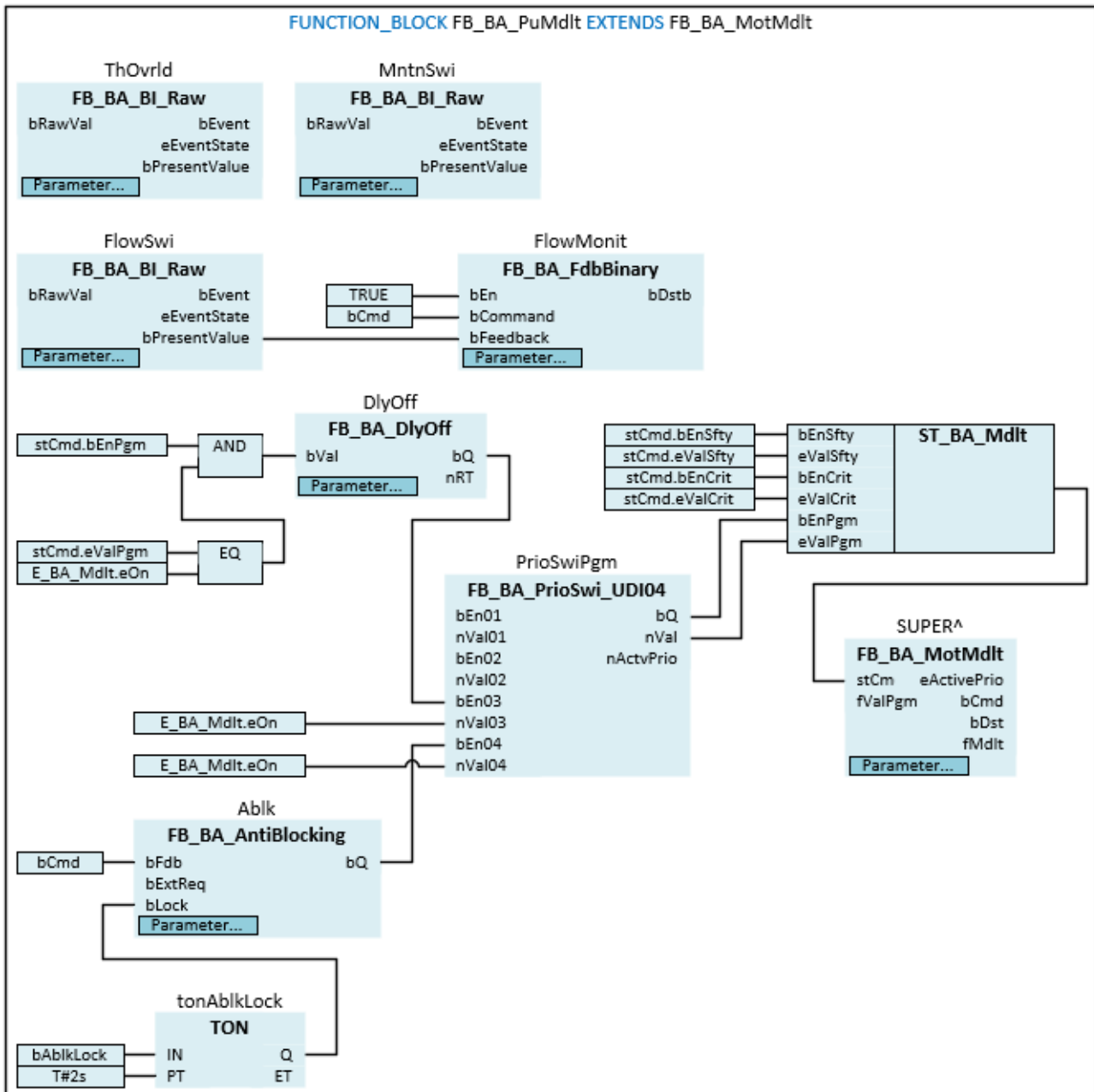
FB_BA_BasePublisher

[FB BA Object \[► 195\]](#)

[FB BA View \[► 173\]](#)

[FB BA MotMdl \[► 768\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_PuMdlr EXTENDS FB_BA_MotMdlr
VAR_INPUT
    bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
    ThOvrl        : FB_BA_BI_Raw;
    MntnSwi       : FB_BA_BI_Raw;
    FlowSwi       : FB_BA_BI_Raw;
    FlowMonit     : FB_BA_FdbBinary;
    DlyOff        : FB_BA_DlyOff;
    Abk           : FB_BA_AntiBlocking;
END_VAR
VAR
    PrioSwiPgm    : FB_BA_PrioSwi_B04;
    tonAblkLock    : TON;
END_VAR

```

Inputs

Name	Type	Description
bAbkLock	BOOL	A TRUE at this input variable interrupts the blocking protection function Abk [► 739]. It should be prevented that pumps and valves get a blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
ThOvrd	FB_BA_BI_Raw [► 140]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [► 140]	The binary input object is used to process a maintenance switch.
FlowSwi	FB_BA_BI_Raw [► 140]	The binary input object is used to process a flow monitor.
FlowMonit	FB_BA_FdbBinary	Template for monitoring the flow monitor.
DlyOff	FB_BA_DlyOff	The template serves as a delayed pump shutdown.
Abk	FB_BA_AntiBlocking [► 739]	Anti-blocking protection.

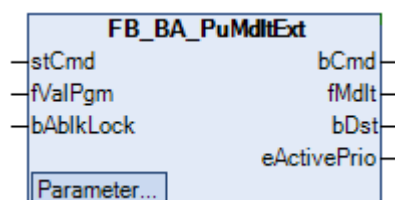
Variables

Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_B04	The priority switch <i>PrioSwiPgm</i> uses the switch-off delay <i>DlyOff</i> , the anti-blocking protection function Abk [► 739] and the priority program of the command structure <i>stCmd</i> of the base class FB_BA_Mot1st [► 756] to determine the current switch value for the base class FB_BA_Mot1st [► 756].
tonAbkLock	TON	Switch-off delay of the blocking protection pulse.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.5.3.2 FB_BA_PuMdlExt



The template is used to control a speed-controlled pump with analog and binary inputs and outputs. It mainly consists of the base class [FB_BA_MotMdlExt](#) [► 771], the switch-off delay *DlyOff* and the anti-blocking protection function [Abk](#) [► 739]. The template *FlowMonit* stands for the monitoring of the flow monitor *FlowSwi*.

The pump is switched on externally via the priorities of the command structure *stCmd* of the base class [FB_BA_MotMdlExt](#) [► 771] or internally by the blocking protection function [Abk](#) [► 739]. The external request via the priority program is switched off with a delay by the template *DlyOff*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

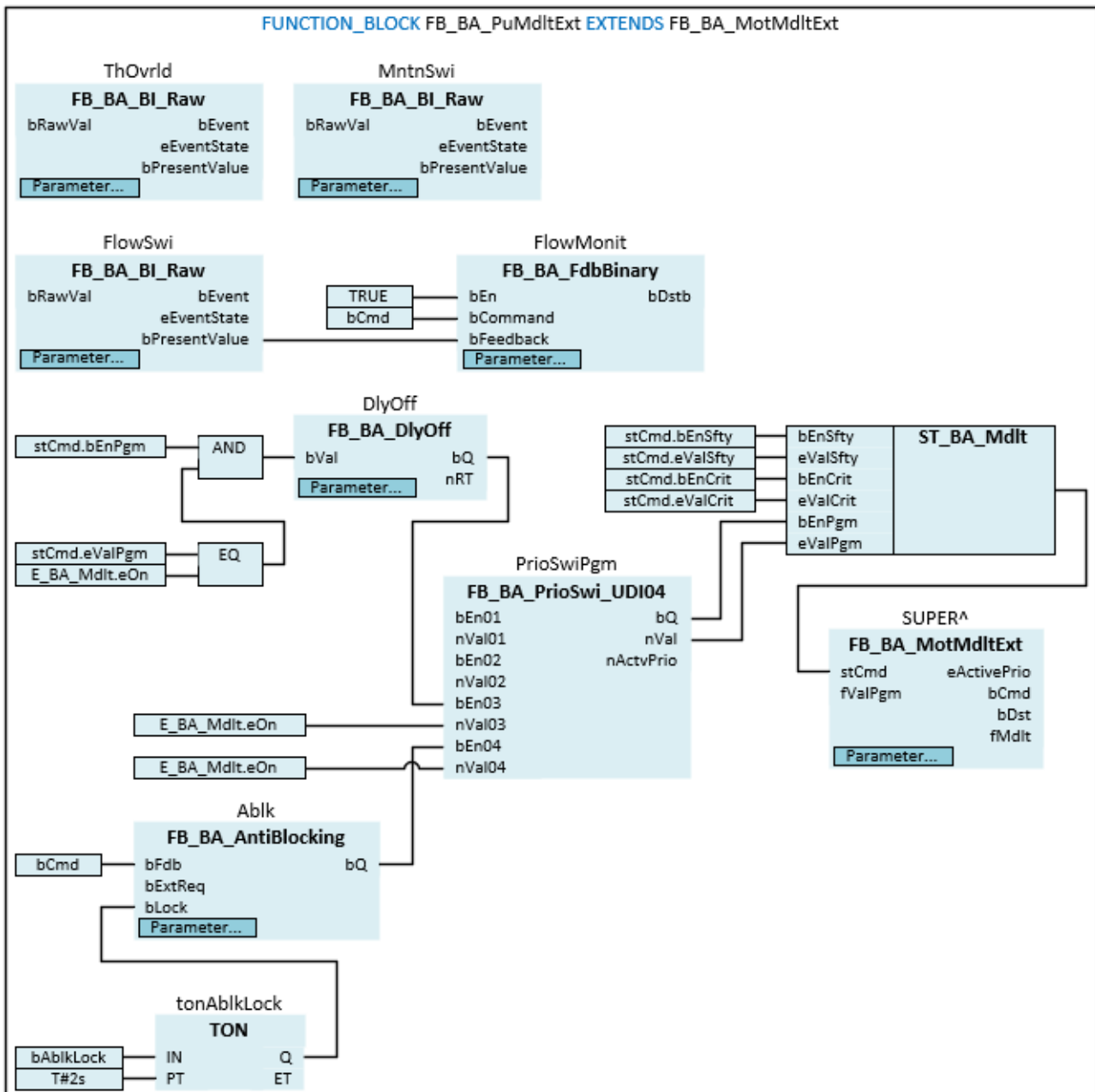
FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

FB_BA_MotMdlExt [► 771]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_PuMdlT EXTENDS FB_BA_MotMdlT
VAR_INPUT
    bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
    ThOvrlD       : FB_BA_BI_Raw;
    MntnSwi       : FB_BA_BI_Raw;
    FlowSwi       : FB_BA_BI_Raw;
    FlowMonit     : FB_BA_FdbBinary;
    DlyOff        : FB_BA_DlyOff;
    Ablk         : FB_BA_AntiBlocking;
END_VAR
VAR
    PrioSwiPgm    : FB_BA_PrioSwi_B04;
    tonAblkLock   : TON;
END_VAR

```

Inputs

Name	Type	Description
bAblkLock	BOOL	A TRUE at this input variable interrupts the blocking protection function Ablk [▶ 739]. It should be prevented that pumps and valves get a blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
ThOvrlD	FB_BA_BI_Raw [▶ 140]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 140]	The binary input object is used to process a maintenance switch.
FlowSwi	FB_BA_BI_Raw [▶ 140]	The binary input object is used to process a flow monitor.
FlowMonit	FB_BA_FdbBinary	Template for monitoring the flow monitor.
DlyOff	FB_BA_DlyOff	The template serves as a delayed pump shutdown.
Ablk	FB_BA_AntiBlocking [▶ 739]	Anti-blocking protection.

Variables

Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_B04	The priority switch <i>PrioSwiPgm</i> uses the switch-off delay DlyOff , the anti-blocking protection function Ablk [▶ 739] and the priority program of the command structure <i>stCmd</i> of the base class FB_BA_Mot1st [▶ 756] to determine the current switch value for the base class FB_BA_Mot1st [▶ 756].
tonAblkLock	TON	Switch-off delay of the blocking protection pulse.

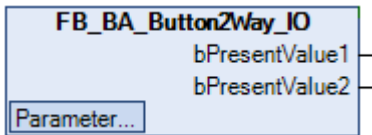
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.6 Sensor

Templates for sensors and data acquisition.

6.1.4.2.2.6.1 *FB_BA_Button2Way_IO*



The template is used to directly link the inputs of a dual push button. The states are represented on a hierarchy level with two objects and made available to the project via the outputs *bPresentValue1* and *bPresentValue2*. This template is used, for example, for the integration of blind buttons.

- ▲ ROM (BACnet Structured View Object)
 - RoomGeneral (BACnet Structured View Object)
 - ▲ SampleRoom (BACnet Structured View Object)
 - PrcDetc (BACnet Structured View Object)
 - WdwCon1 (BACnet Structured View Object)
 - WdwCon2 (BACnet Structured View Object)
 - DewPointSensor (BACnet Structured View Object)
 - RmTSen1 (BACnet Structured View Object)
 - RmTSen2 (BACnet Structured View Object)
 - RmTAdj (BACnet Structured View Object)
 - Ctrl (BACnet Structured View Object)
 - VlvHtg (BACnet Structured View Object)
 - VlvCol (BACnet Structured View Object)
 - ▲ SwitchBld (BACnet Structured View Object)
 - Swi1 (BACnet Binary Input Object)
 - Swi2 (BACnet Binary Input Object)
 - SunBld1 (BACnet Structured View Object)
 - SunBld2 (BACnet Structured View Object)



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [[▸ 195](#)]

FB_BA_View [[▸ 173](#)]

Illustration

```
FUNCTION_BLOCK FB_BA_Button2Way_IO EXTENDS FB_BA_VIEW
VAR_INPUT
END_VAR
VAR_INPUT CONSTANT
    Swi1          : FB_BA_BI_IO;
    Swi2          : FB_BA_BI_IO;
END_VAR
VAR_OUTPUT
    bPresentValue1 : BOOL;
    bPresentValue2 : BOOL;
END_VAR
```

🔧 Inputs CONSTANT

Name	Type	Description
Swi1	FB_BA_BI_IO [► 139]	Binary input object for switch1.
Swi2	FB_BA_BI_IO [► 139]	Binary input object for switch2.

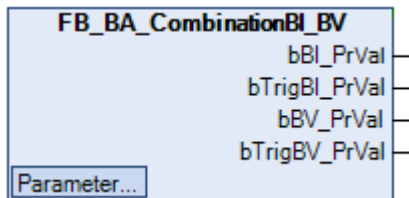
🔧 Outputs

Name	Type	Description
bPresentValue1	BOOL	Output: state switch 1.
bPresentValue2	BOOL	Output: state switch 2.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.6.2 *FB_BA_CombinationBI_BV*



The template is a combination of a BI and BV object.

The binary input object *Input* logs a binary input value from a bus terminal and outputs it as a boolean process value. In addition, the process value *Input* is output as a rising edge *bTrigBI_PrVal*.

The binary value object *Value* represents a boolean process value. It can be parameterized as a switch or push button by the parameter *E_BA_ToggleMode*. The output value of the Value object is also output as a rising edge *bTrigBV_PrVal*.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

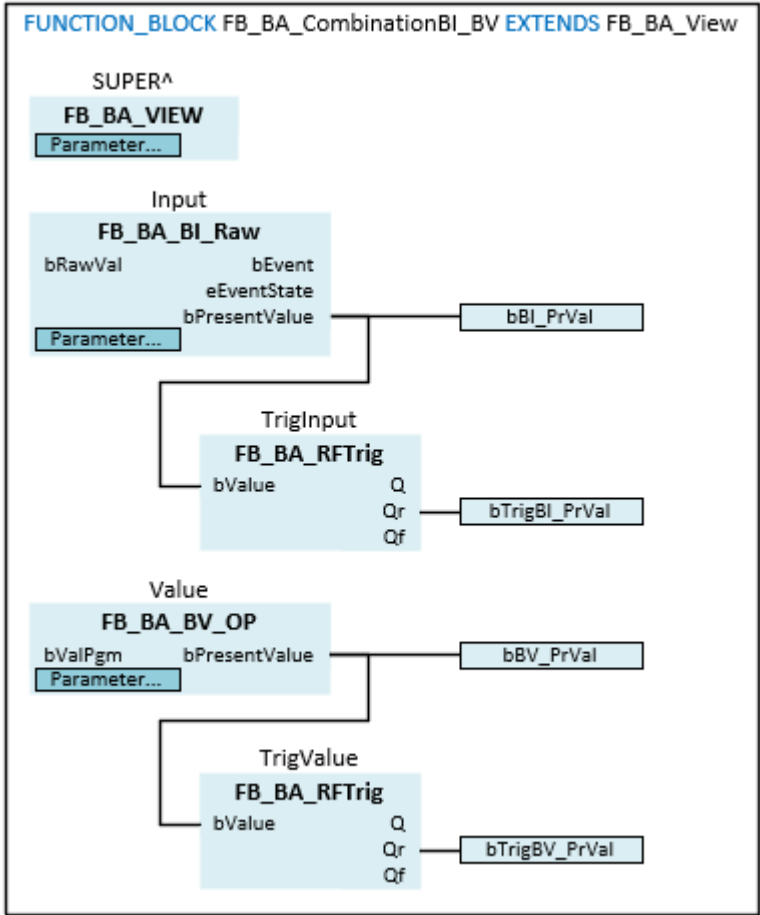
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_CombinationBI_BV EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    Input      : FB_BA_BI_Raw;
    Value      : FB_BA_BV_Op;
END_VAR
VAR_OUTPUT
    bBI_PrVal  : BOOL;
    bTrigBI_PrVal : BOOL;
    bBV_PrVal  : BOOL;
    bTrigBV_PrVal : BOOL;
END_VAR
VAR
    TrigInput  : FB_BA_RFTrig;
    TrigValue  : FB_BA_RFTrig;
END_VAR
```

📌 Inputs CONSTANT

Name	Type	Description
Input	FB_BA_BI_Raw [► 140]	Binary input object for displaying a process value.
Value	FB_BA_BV_Op [► 150]	Binary value object for displaying a process value. It can be used as a switch or push button.

🔌 Outputs

Name	Type	Description
bBI_PrVal	BOOL	Current value of the binary input object <i>Input</i> .
bTrigBI_PrVal	BOOL	Current value of the output <i>Qr</i> of the function block <i>TrigInput</i> .
bBV_PrVal	BOOL	Current value of the binary value object <i>Value</i> .
bTrigBV_PrVal	BOOL	Current value of the output <i>Qr</i> of the function block <i>TrigValue</i> .

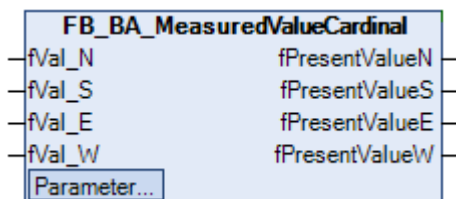
Variables

Name	Type	Description
TrigInput	FB_BA_RFTrig	The function block generates a rising edge from the output signal of the binary input object <i>Input</i> .
TrigValue	FB_BA_RFTrig	The function block generates a rising edge from the output signal of the binary value object <i>Value</i> .

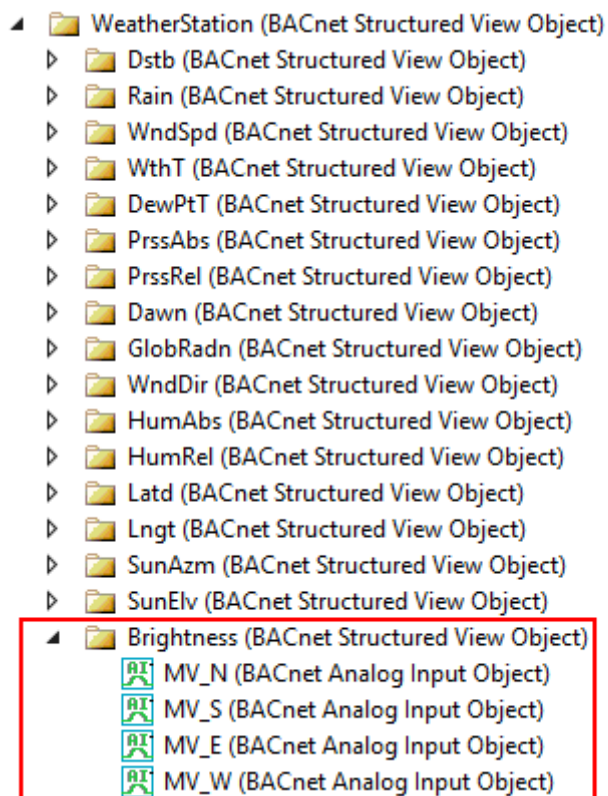
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.6.3 *FB_BA_MeasuredValueCardinal*



This template represents at the output, in a hierarchical representation, a collection of four values, which are assigned to the cardinal points. This template can be used to display the external brightnesses for all four cardinal points on one display layer, as shown in the following graphic.



The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [► 195]

FB_BA_View [► 173]

Illustration

```
FUNCTION_BLOCK FB_BA_MeasuredValueCardinal EXTENDS FB_BA_View
VAR_INPUT
    fVal_N      : REAL;
    fVal_S      : REAL;
    fVal_E      : REAL;
    fVal_W      : REAL;
END_VAR
VAR_INPUT CONSTANT
    MV_N        : FB_BA_AI;
    MV_S        : FB_BA_AI;
    MV_E        : FB_BA_AI;
    MV_W        : FB_BA_AI;
END_VAR
VAR_OUTPUT
    fPresentValueN : REAL;
    fPresentValueS : REAL;
    fPresentValueE : REAL;
    fPresentValueW : REAL;
END_VAR
```

Inputs

Name	Type	Description
fVal_N	REAL	Analog input value for "North".
fVal_S	REAL	Analog input value for "South".
fVal_E	REAL	Analog input value for "East".
fVal_W	REAL	Analog input value for "West".

Inputs CONSTANT

Name	Type	Description
MV_N	FB_BA_AI [▶ 125]	Analog input object for the "North" value.
MV_S	FB_BA_AI [▶ 125]	Analog input object for the "South" value.
MV_E	FB_BA_AI [▶ 125]	Analog input object for the "East" value.
MV_W	FB_BA_AI [▶ 125]	Analog input object for the "West" value.

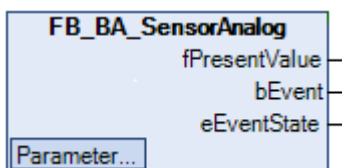
Outputs

Name	Type	Description
fPresentValueN	REAL	Output value for "North".
fPresentValueS	REAL	Output value for "South".
fPresentValueE	REAL	Output value for "East".
fPresentValueW	REAL	Output value for "West".

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.6.4 *FB_BA_SensorAnalog*



The function block *MV*, within the template, records an analog input value from an I/O bus terminal and converts it into a real process value.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.



The initialization of the template takes place within the method *FB_Init*.

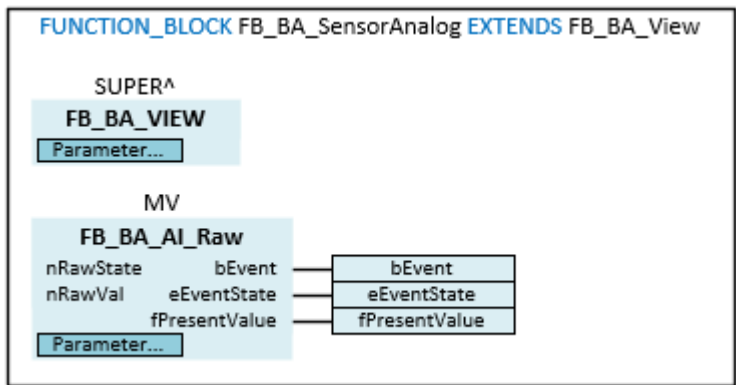
Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

[FB_BA_Object \[\[▶ 195\]\(#\)\]](#)

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_SensorAnalog EXTENDS FB_BA_View
VAR_INPUT CONSTANT
    MV
END_VAR
VAR_OUTPUT
    fPresentValue : REAL;
    bEvent        : BOOL;
    eEventState   : E_BA_EventState;
END_VAR
```

🔌 Inputs CONSTANT

Name	Type	Description
MV	FB_BA_AI_Raw [► 129]	Analog input object for displaying a process value.

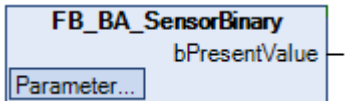
🔌 Outputs

Name	Type	Description
fPresentValue	REAL	Current value of the analog input object MV for displaying a process value.
bEvent	BOOL	A TRUE indicates that an event is pending.
eEventState	E_BA_EventState	Outputs the event state of the analog input object MV.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.6.5 *FB_BA_SensorBinary*



The template records the binary input value *bVal* and outputs it as a boolean process value.

The template is available in 2 different configuration variants regarding the connection to the I/O process level.

Under I/O process level the function blocks regarding the connection to the process image are listed, which exist in the different configuration variants.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

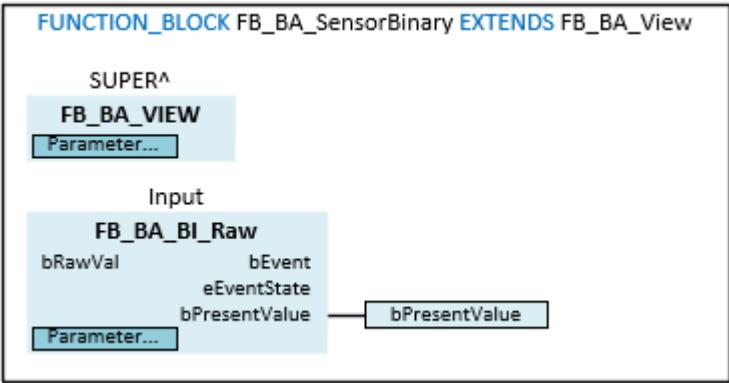


The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - FB_BA_Object [▶ 195]
 - FB_BA_View [▶ 173]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_SensorBinary EXTENDS FB_BA_View
VAR_INPUT
    bVal          : BOOL;
END_VAR
VAR_INPUT CONSTANT
    Input          : FB_BA_BI_Raw;
END_VAR
VAR_OUTPUT
    bPresentValue  : BOOL;
END_VAR
```

🔧 Inputs

Name	Type	Description
bVal	BOOL	Binary input value from the I/O process level.

🔧 Inputs CONSTANT

Name	Type	Description
Input	FB_BA_BI_Raw [▶ 140]	Binary input object for displaying a process value

🔧 Outputs

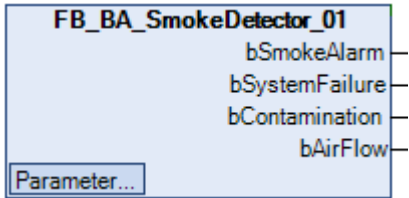
Name	Type	Description
bPresentValue	BOOL	Current value of the binary input object <i>Input</i> to display a process value.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.7 Smoke detector

6.1.4.2.2.7.1 *FB_BA_SmokeDetector_01*



The template is used to display faults and warnings of a duct smoke detector.

The binary output object *RstHw* can be used to reset these messages by remote release.

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

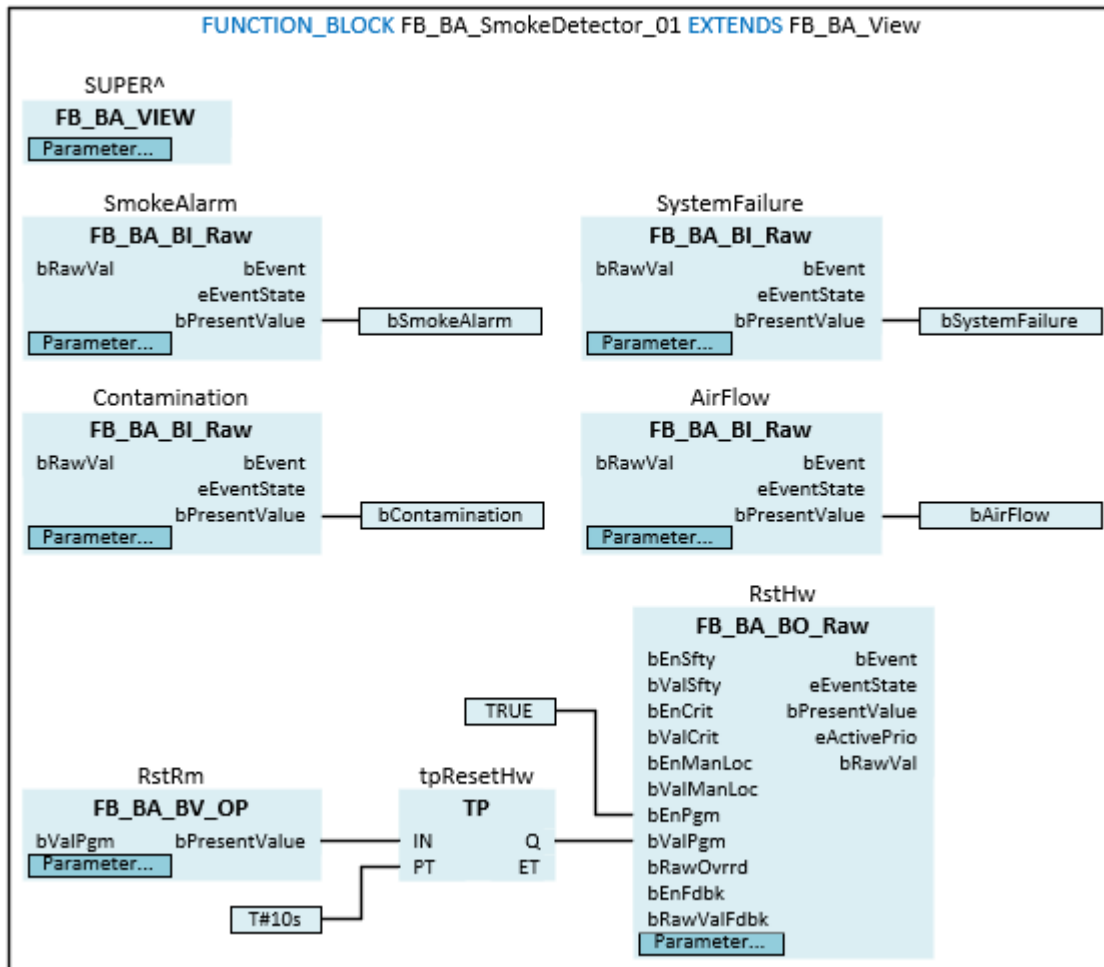


The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - [FB_BA_Object \[► 195\]](#)
 - [FB_BA_View \[► 173\]](#)

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_SmokeDetector_01 EXTENDS FB_BA_View
VAR_OUTPUT
    bSmokeAlarm      : BOOL;
    bSystemFailure   : BOOL;
    bContamination   : BOOL;
    bAirFlow         : BOOL;
END_VAR
VAR_INPUT CONSTANT
    SmokeAlarm      : FB_BA_BI_Raw;
    SystemFailure    : FB_BA_BI_Raw;
    Contamination    : FB_BA_BI_Raw;
    AirFlow         : FB_BA_BI_Raw;
    RstHw           : FB_BA_BO_Raw;
    RstRm           : FB_BA_BV_OP;
END_VAR
VAR
    tpResetHw       : TP;
END_VAR

```

 Outputs

Name	Type	Description
bSmokeAlarm	BOOL	Current value of the binary input object <i>SmokeAlarm</i> .
bSystemFailure	BOOL	Current value of the binary input object <i>SystemFailure</i> .
bContamination	BOOL	Current value of the binary input object <i>Contamination</i> .
bAirFlow	BOOL	Current value of the binary input object <i>AirFlow</i> .

🚩 Inputs CONSTANT

Name	Type	Description
SmokeAlarm	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process a smoke alarm.
SystemFailure	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process a system fault of the duct smoke detector.
Contamination	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process a pollution of the duct smoke detector.
AirFlow	FB_BA_BI_Raw [▶ 140]	Binary input object is used to process insufficient air flow.
RstHw	FB_BA_BO_Raw [▶ 145]	Binary output object which triggers a remote release on the smoke detector.
RstRm	FB_BA_BV_Op [▶ 150]	The binary value object <i>RstRm</i> triggers a remote release on the smoke detector from the management level. The <i>RstRm</i> object is initialized as a button object by the additional parameter <i>eToggleMode</i> .

Variables

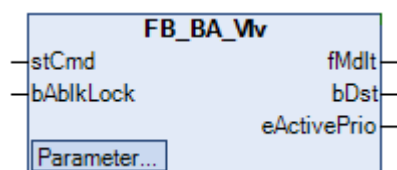
Name	Type	Description
tpResetHw	TP	The timing element extends the acknowledgement pulse for interfacing relay circuits (e.g. frost protection relay).

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.8 Valve

6.1.4.2.2.8.1 FB_BA_Vlv



The template is used to control a continuous valve with analog inputs and outputs. It mainly consists of the base class *FB_BA_ActuatorAnalog* [▶ 712], the anti-blocking protection function *Ablk* [▶ 739] and the AI object *Fdb* for recording the position feedback from the valve.

The valve is switched on externally by the priorities of the command structure *stCmd* of the base class *FB_BA_ActuatorAnalog* [▶ 712] or internally by the blocking protection function *Ablk* [▶ 739].

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

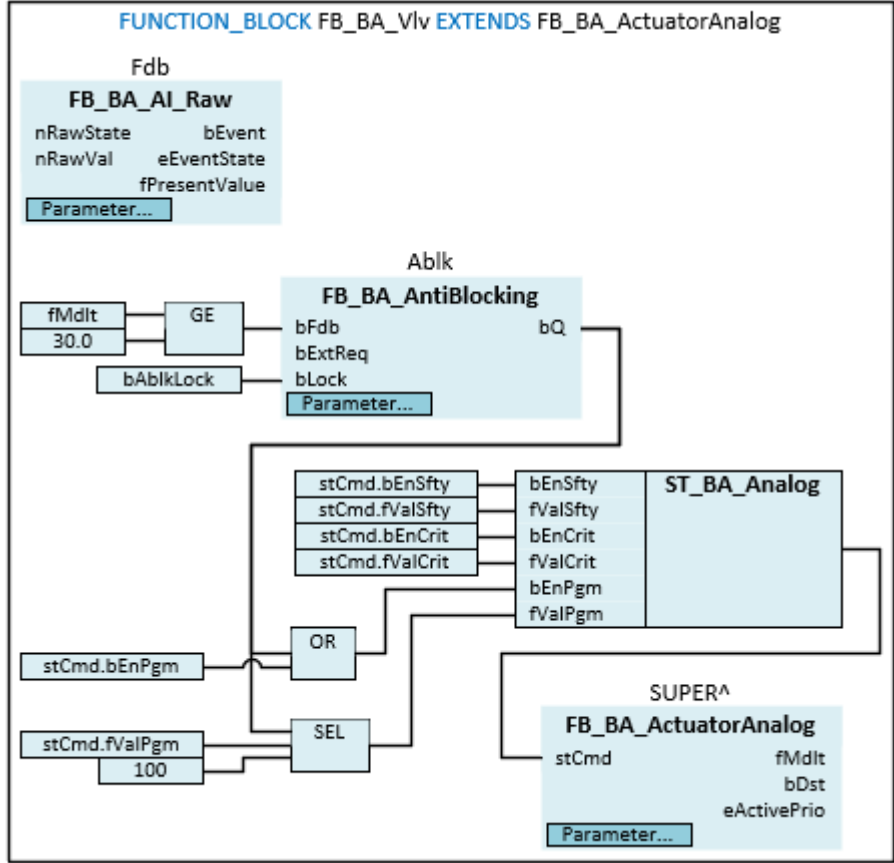


The initialization of the template takes place within the method *FB_Init*.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - FB_BA_Object [▶ 195]
 - FB_BA_View [▶ 173]
 - FB_BA_ActuatorAnalog [▶ 712]

Block diagram



Illustration

```
FUNCTION_BLOCK FB_BA_Vlv EXTENDS FB_BA_ActuatorAnalog
VAR_INPUT
    bAbkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
    Ablk          : FB_BA_AntiBlocking;
    Fdb           : FB_BA_AI_Raw;
END_VAR
```

🔧 Inputs

Name	Type	Description
bAbkLock	BOOL	A TRUE at this input variable interrupts the blocking protection function Ablk [▶ 739] . It should be prevented that pumps and valves get a blocking protection pulse at the same time.

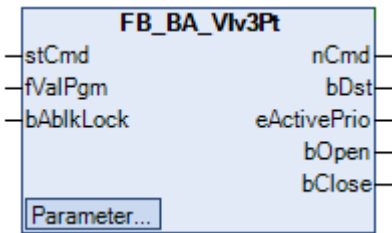
🚩 Inputs **CONSTANT**

Name	Type	Description
Ablk	FB_BA_AntiBlocking [▶ 739]	Anti-blocking protection.
Fdb	FB_BA_AI_Raw [▶ 129]	Analog input object for logging the position feedback of the valve.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.8.2 **FB_BA_Vlv3pt**



The template is used to control a three-point valve. It mainly consists of the base class FB_BA_ActuatorMO [▶ 722], the template Anlg3Pnt for the analog conversion of the input signal *fValPgm* into a three-point signal and the anti-blocking protection function Ablk [▶ 739].

The three-point valve is switched on externally by the priorities of the command structure *stCmd* of the base class FB_BA_ActuatorMO [▶ 722] or internally by the blocking protection function Ablk [▶ 739].

The template is available in 2 different configuration variants regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

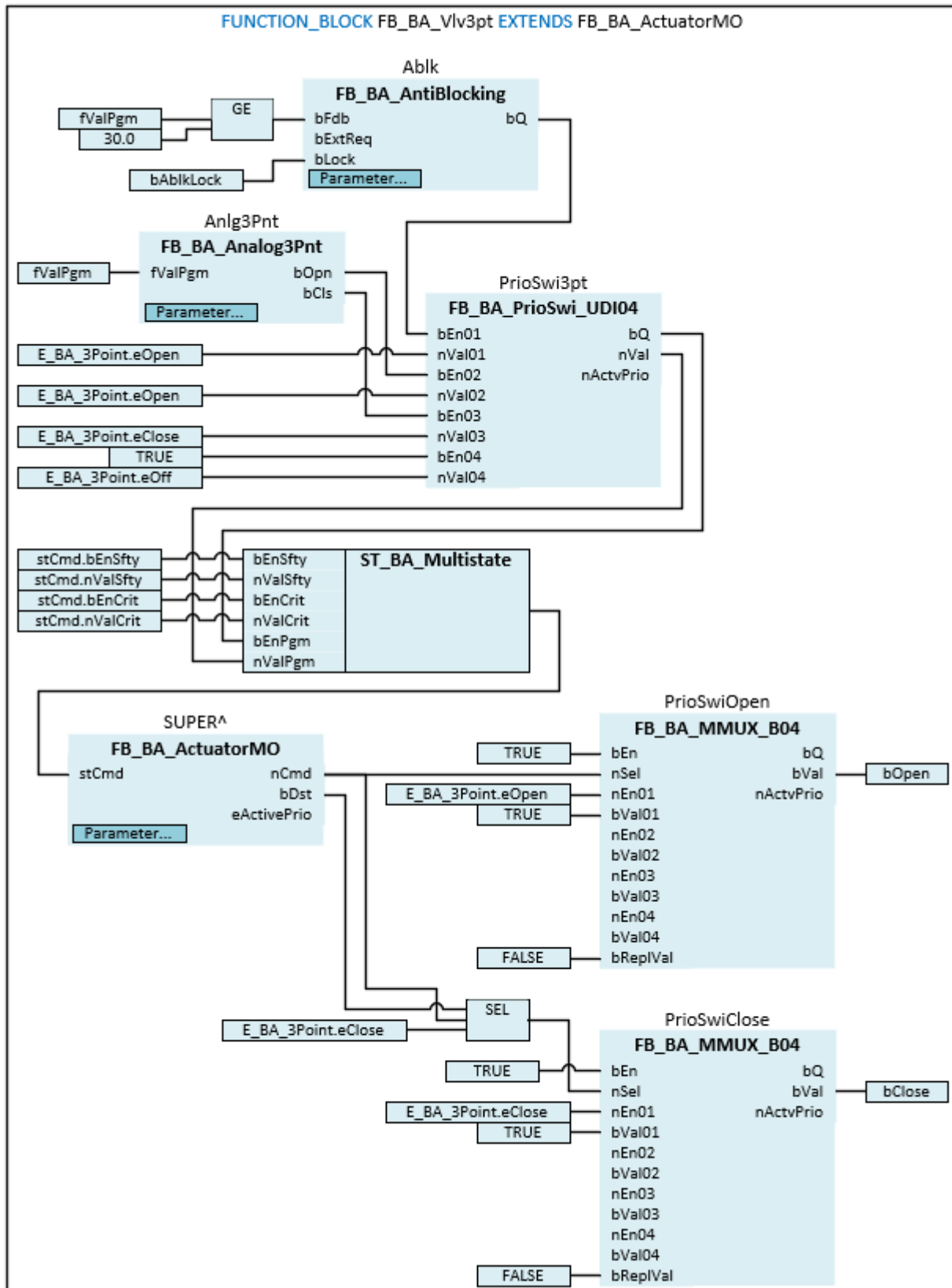


The initialization of the template takes place within the method `FB_Init`.

Inheritance hierarchy

- FB_BA_Base
 - FB_BA_BasePublisher
 - FB_BA_Object [▶ 195]
 - FB_BA_View [▶ 173]
 - FB_BA_ActuatorMO [▶ 722]

Block diagram



Illustration

```

FUNCTION_BLOCK FB_BA_Vlv3pt EXTENDS FB_BA_ActuatorMO
VAR_INPUT
    fValPgm      : REAL;
    bAblkLock    : BOOL;
END_VAR
VAR_OUTPUT

```

```

    bOpen      : BOOL;
    bClose     : BOOL;
END_VAR
VAR_INPUT CONSTANT
    Anlg3Pnt   : FB_BA_Analog3Pnt;
    Ablk       : FB_BA_AntiBlocking;
END_VAR
VAR
    PrioSwi3pt : FB_BA_PrioSwi_UDI04;
    PrioSwiOpen : FB_BA_MMUX_B04;
    PrioSwiClose : FB_BA_MMUX_B04;
END_VAR

```

Inputs

Name	Type	Description
fValPgm	REAL	Continuous input signal for analog conversion to a three-point signal. This can come from a PID controller and have a value from 0...100 %.
bAblkLock	BOOL	A TRUE at this input variable interrupts the blocking protection function Ablk [► 739]. It should be prevented that pumps and valves get a blocking protection pulse at the same time.

Outputs

Name	Type	Description
bOpen	BOOL	Variable for the control Open of the 3-point valve. This variable must be linked to a bus terminal.
bClose	BOOL	Variable for the control Close of the 3-point valve. This variable must be linked to a bus terminal.

Inputs CONSTANT

Name	Type	Description
Anlg3Pnt	FB_BA_Analog3Pnt [► 752]	The template <i>Anlg3Pnt</i> converts the analog input signal <i>fValPgm</i> into a three-point signal.
Ablk	FB_BA_AntiBlocking [► 739]	Anti-blocking protection.

Variables

Name	Type	Description
PrioSwi3pt	FB_BA_PrioSwi_UDI04 [► 369]	The priority switch <i>PrioSwi3pt</i> determines the current switching value for the priority "Program" of the command structure <i>stCmd</i> on the basis of the analog 3-point converter <i>Anlg3Pnt</i> and the anti-blocking protection function <i>Ablk</i> [► 739].
PrioSwiOpen	FB_BA_MMUX_B04 [► 364]	The multiplexer <i>PrioSwiOpen</i> receives the numeric switch value <i>nCmd</i> from the base class FB_BA_ActuatorMO [► 722] and converts the switch value into the 3-point signal <i>bOpen</i> .
PrioSwiClose	FB_BA_MMUX_B04 [► 364]	The multiplexer <i>PrioSwiClose</i> receives the numeric switch value <i>nCmd</i> from the base class FB_BA_ActuatorMO [► 722] and converts the switch value into the 3-point signal <i>bClose</i> .

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.9 Weather station

Templates for data acquisition from weather stations.

6.1.4.2.2.9.1 Thies**6.1.4.2.2.9.1.1 Integration Thies weather station**

This manual shows the integration of a Thies weather station Compact WSC11, 4.9056.10.000, using the template "FB_BA_Weatherstation_Thies".

Required hardware:

- Thies Weather Station Compact WSC11, 4.9056.10.000 (ASCII format)
- KL6041, preconfigured to 22 bytes process image

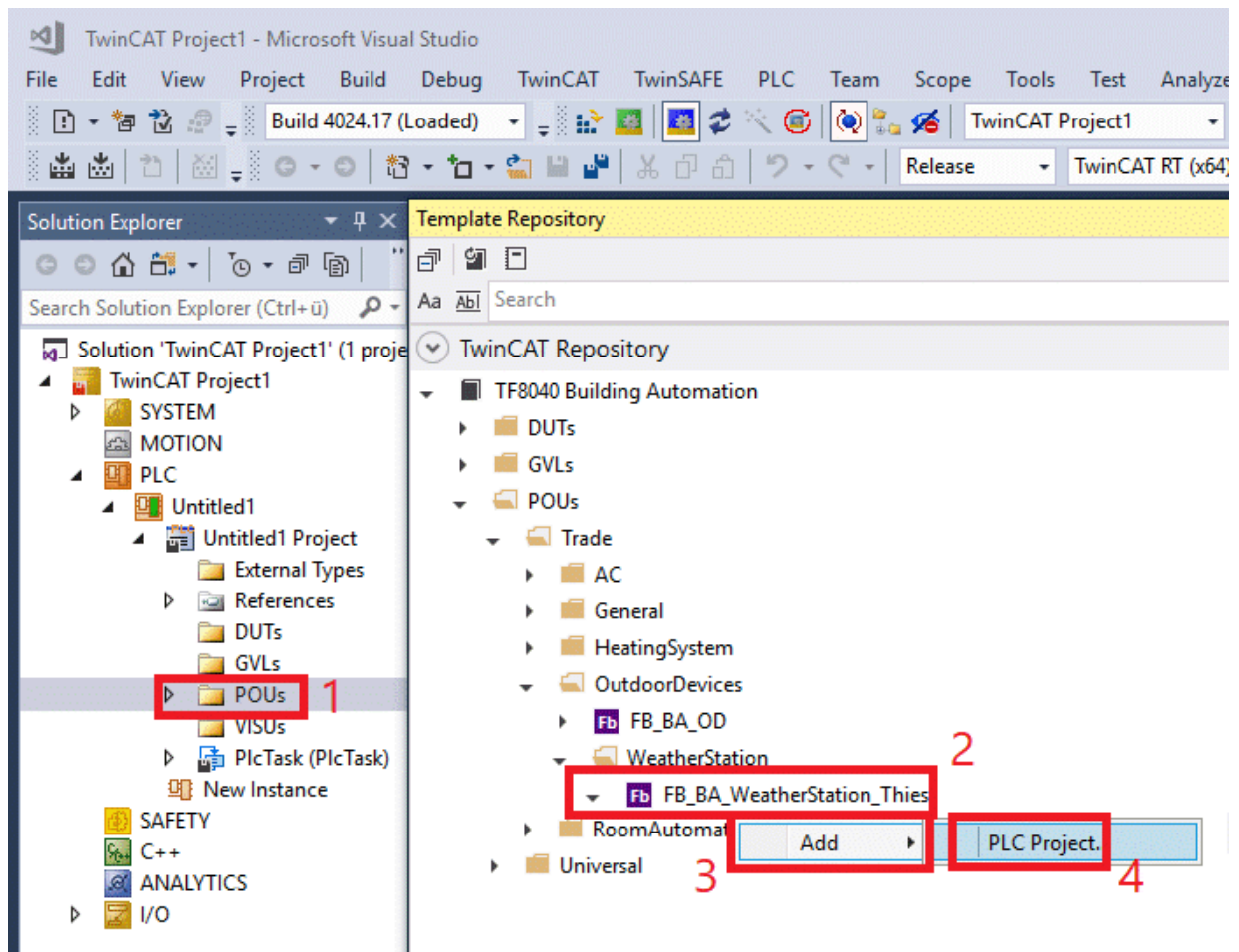
Required additional library:

- Tc2_SerialCom

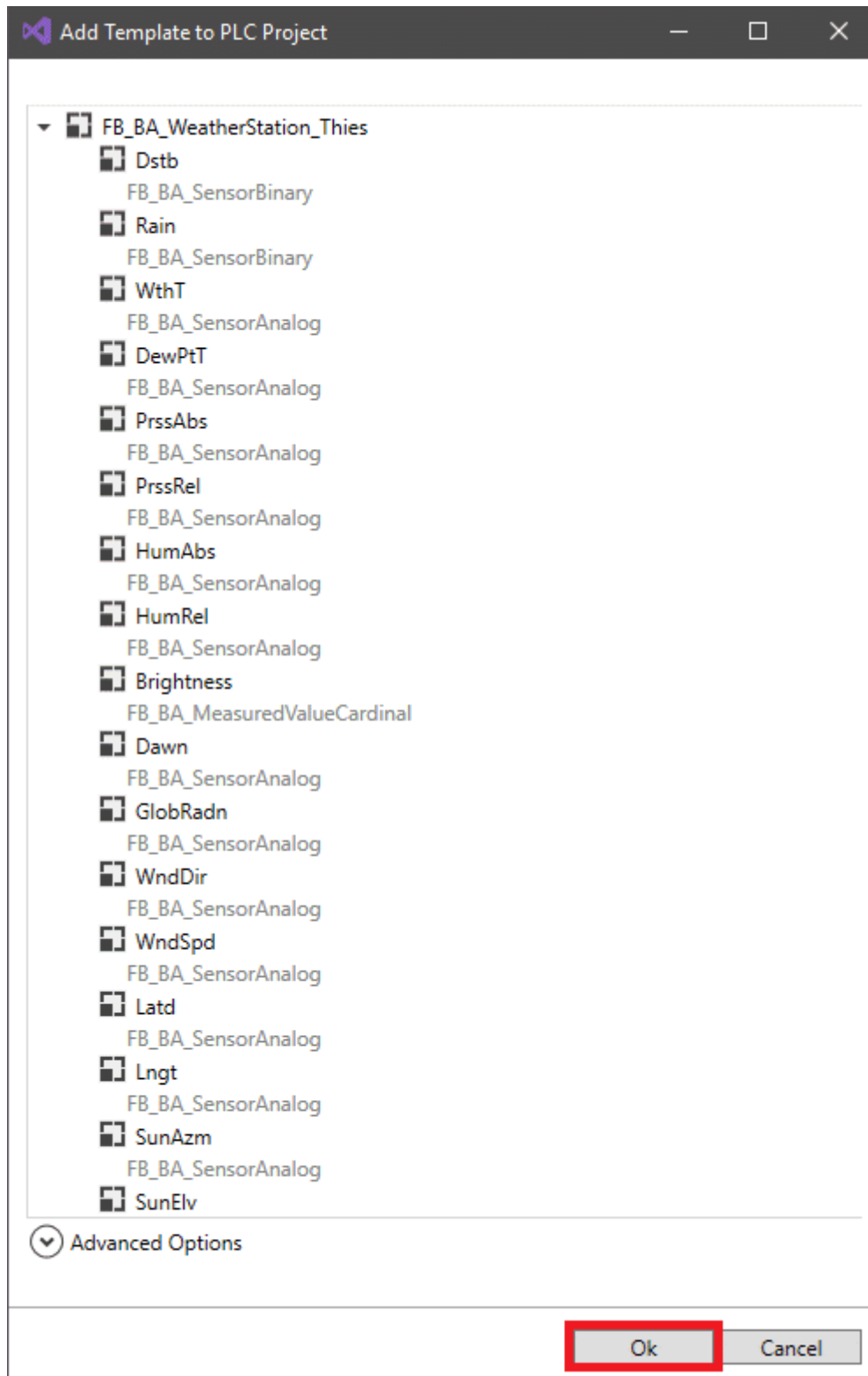
Adding the template:

- ✓ Click on the desired PLC project (1).
- 1. In the template repository, right-click on the template **FB_BA_WeatherStation_Thies** (2)

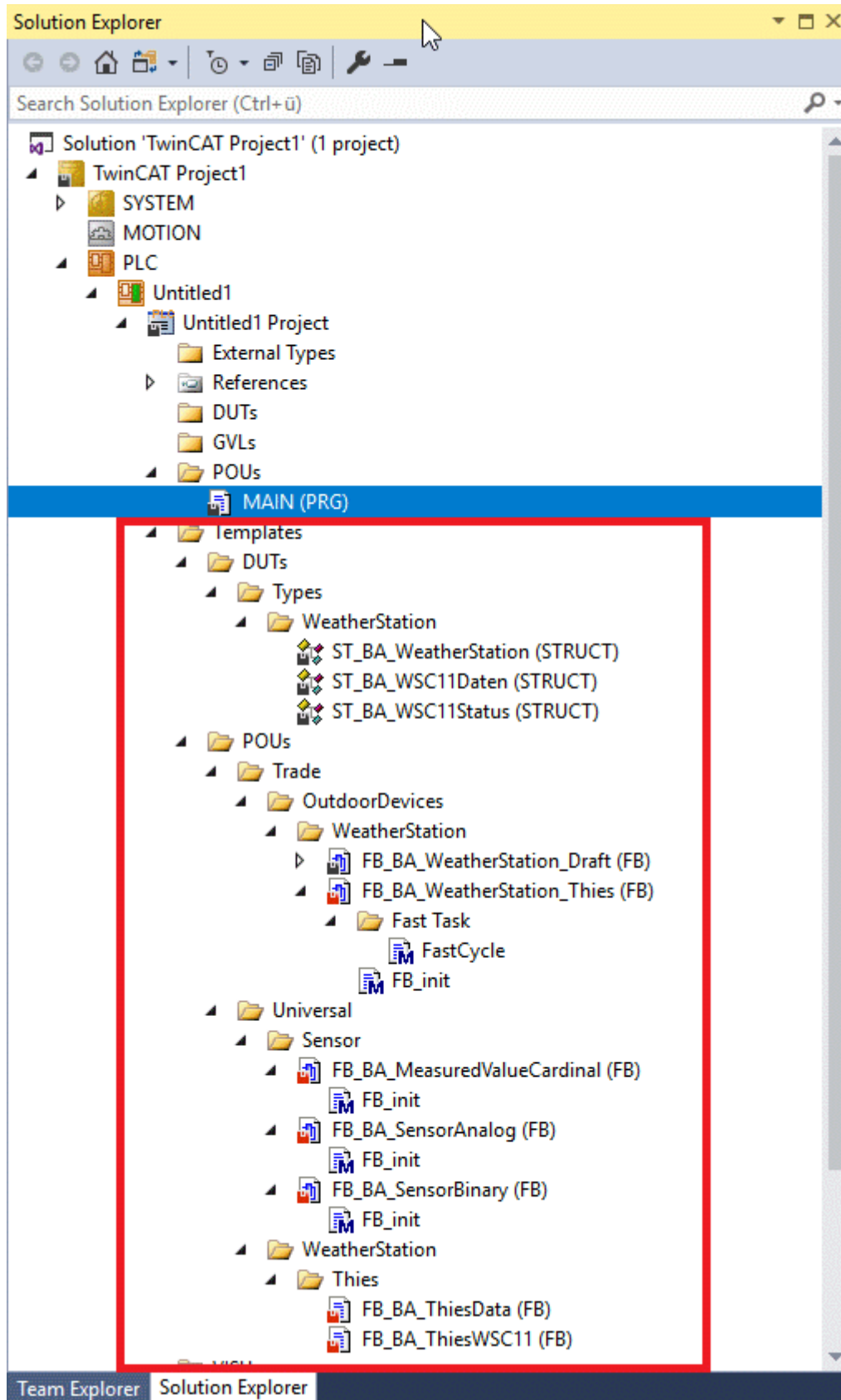
2. Add the template (3) and (4).



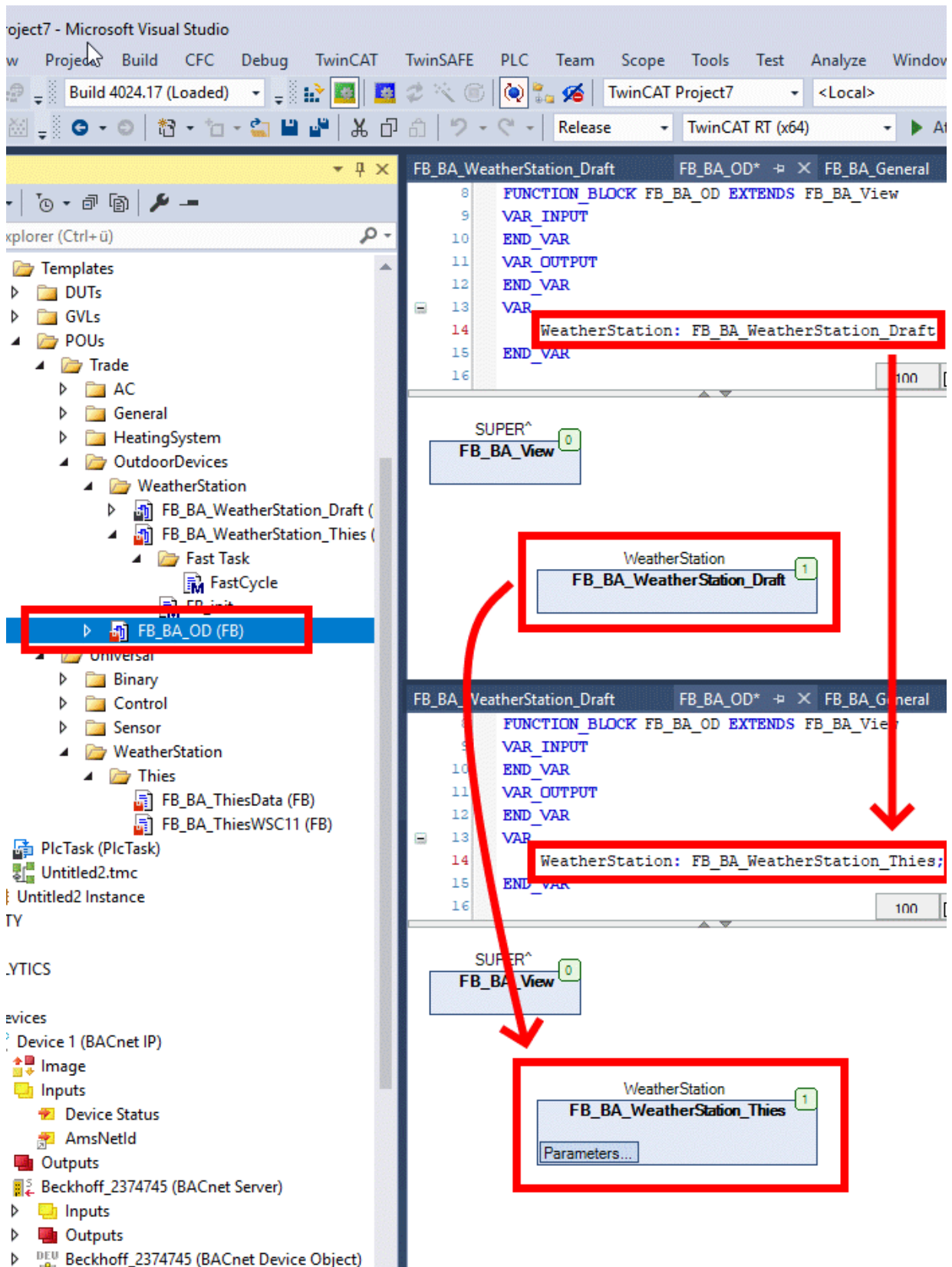
3. The following dialog with the list of additionally implemented function blocks opens. Close the dialog with **OK**.



4. All new PLC function blocks are now dragged into a folder **Templates** and can be distributed in the project if required.



- ⇒ In the standard PLC project a weather station template *FB_BA_Weatherstation_Draft* is already included, which contains all necessary display objects and is intended for individual linking.



Adding a fast task for serial communication

The serial terminal (KL6041) is configured to the following communication parameters during a PLC restart:

- Baud rate: 9600

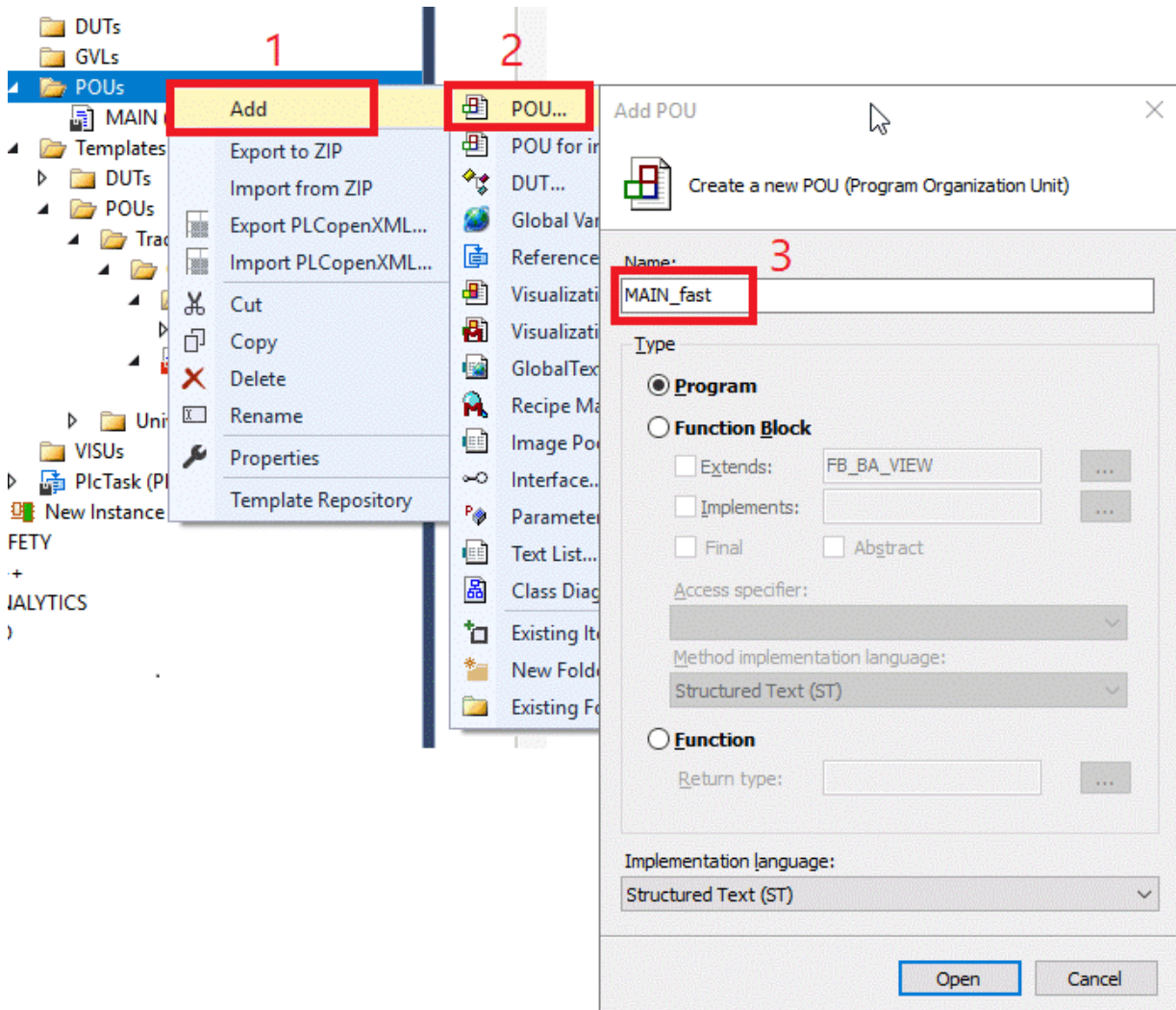
- Data bits: 8
- Parity: none
- Stop bits: 1
- Handshake: RS485 HALFDUPLEX

The process image must be set to 22 bytes beforehand.

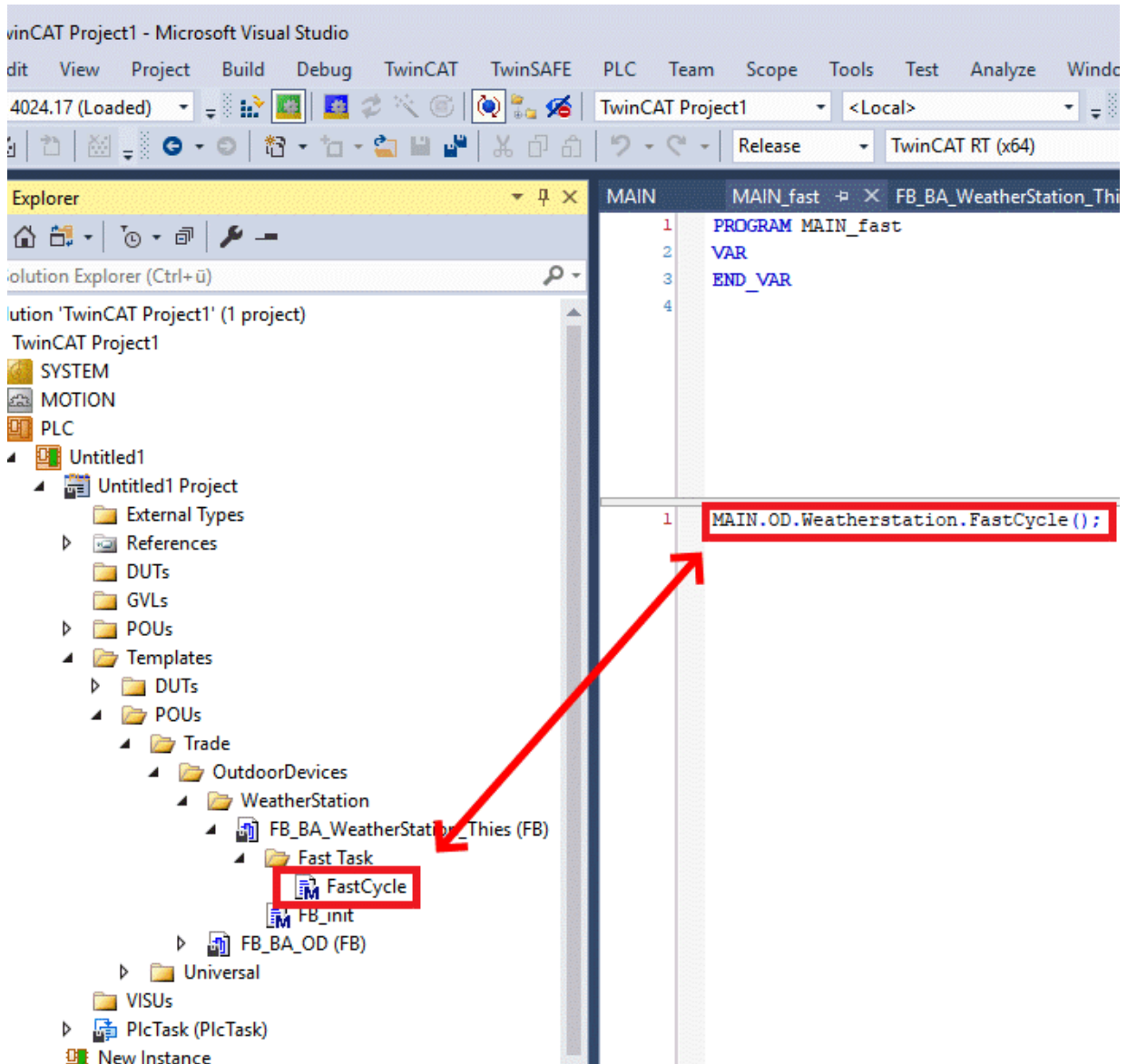
After that the actual communication between terminal and weather station starts.

The function blocks required for this are available in the Tc2_SerialCom library and are called in the "FastCycle" method of the "FB_BA_Weatherstation_Thies" template. This call must be assigned to a faster task than the normal PLC cycle task.

1. Right-click POU to open the new window, select **add** (1) and **POU** (2).

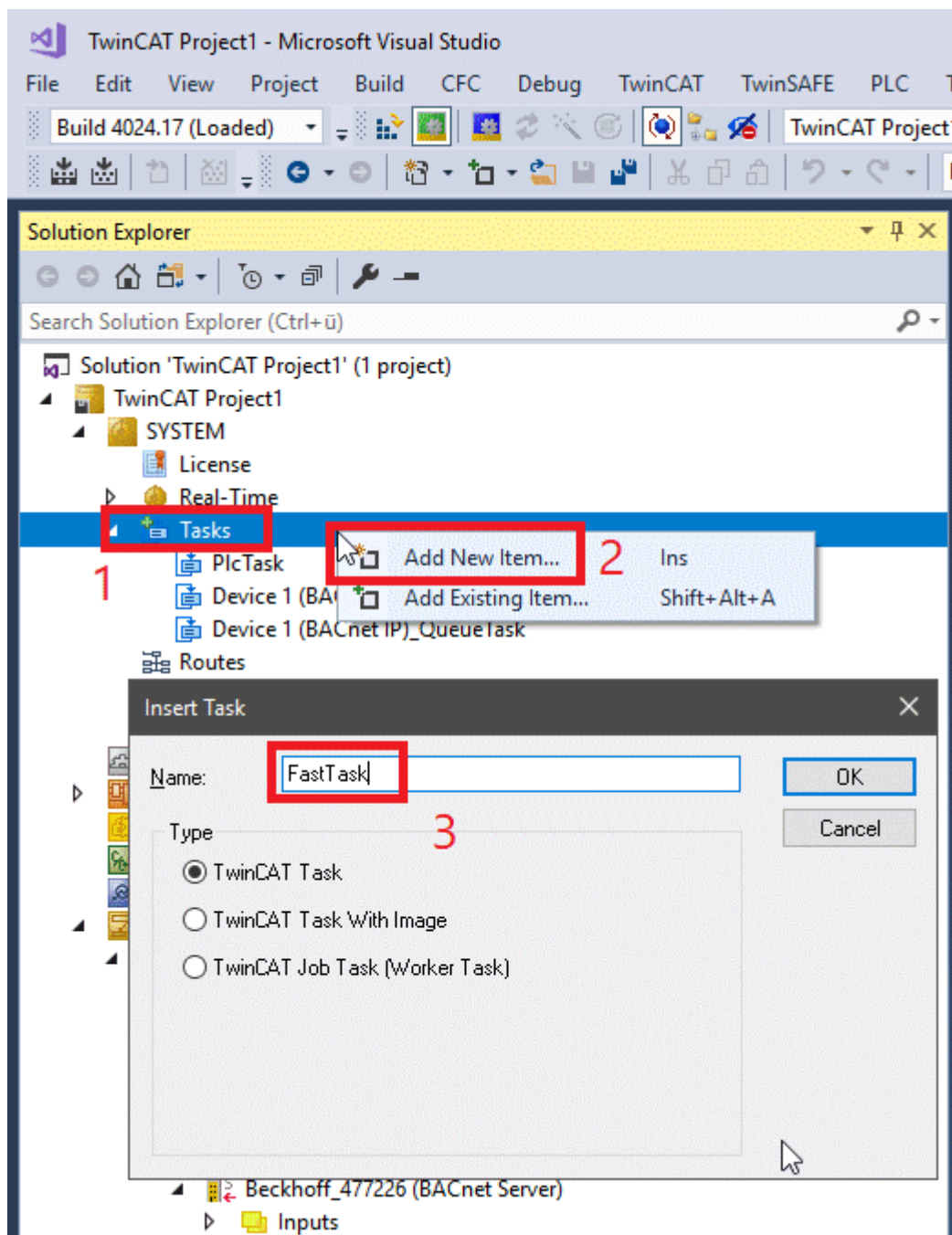


2. A window opens. Enter the name there (example **MAIN_fast**) (3). The POU type is a program in structured text (ST).

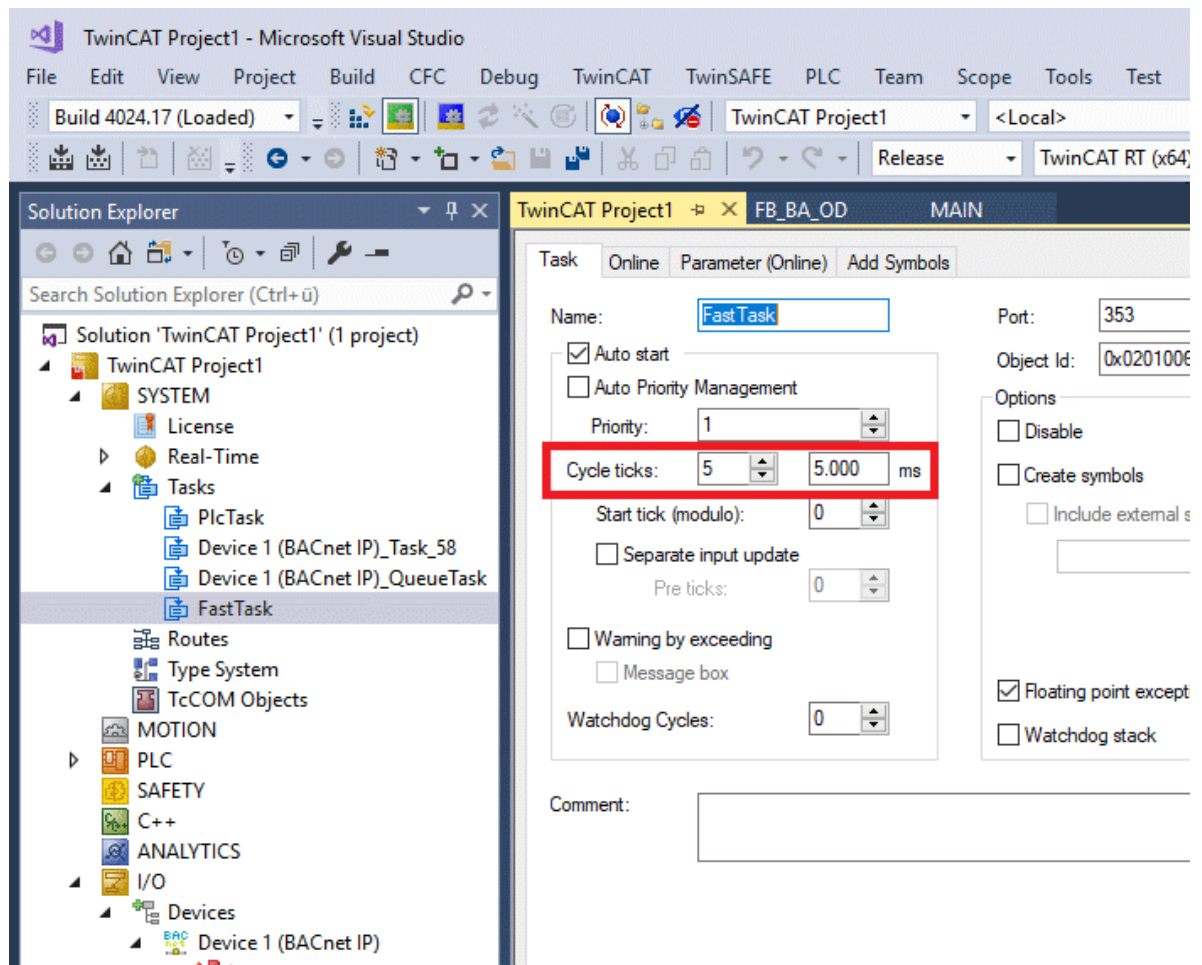


⇒ In this program, the method **FastCycle** is now called.

3. Add a new item (2) by right-clicking on **Tasks** (1). Name it with a meaningful name, for example **FastTask** (3).

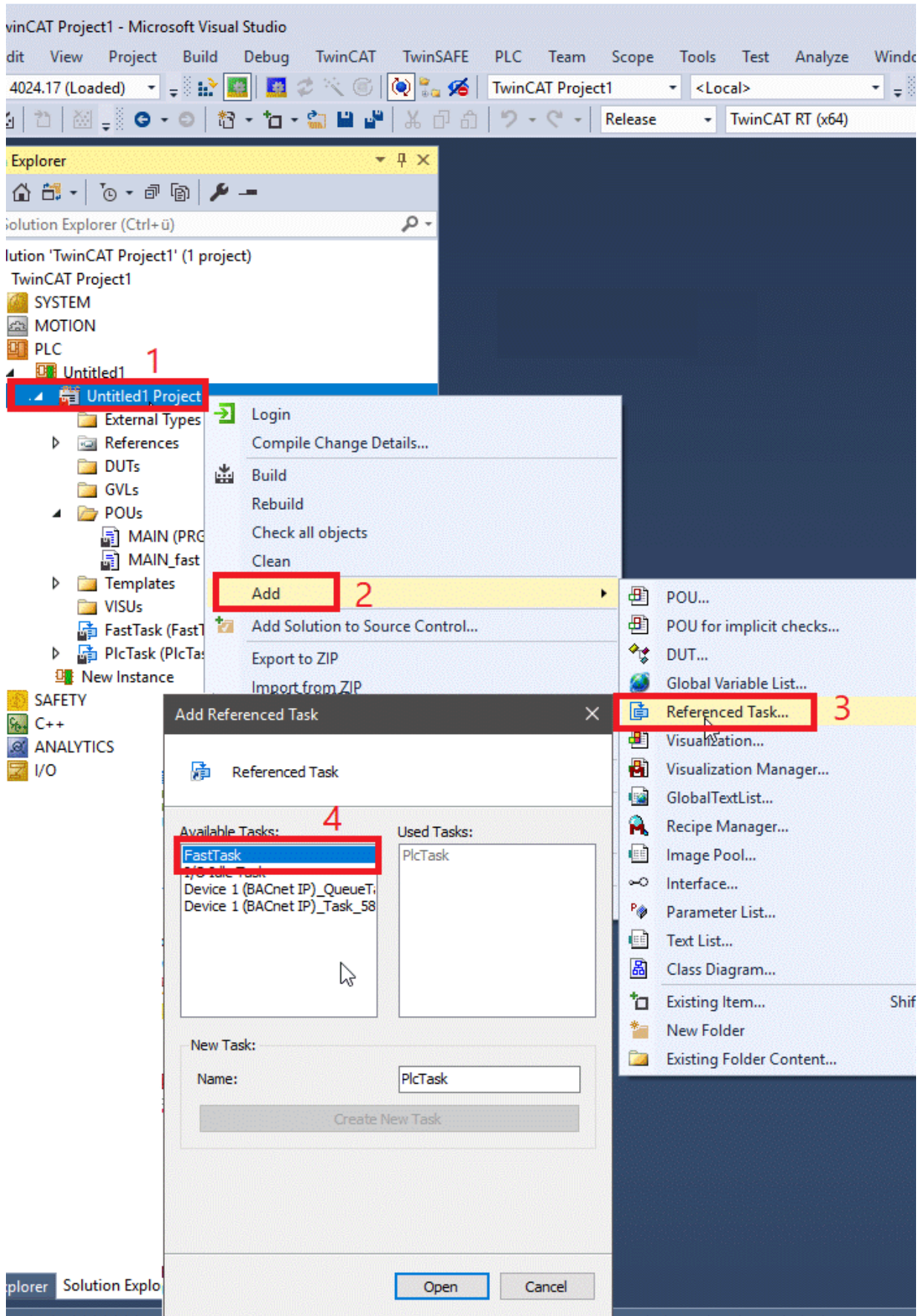


⇒ This new task must now be set to a small cycle time. The 5 ms shown here are a recommendation



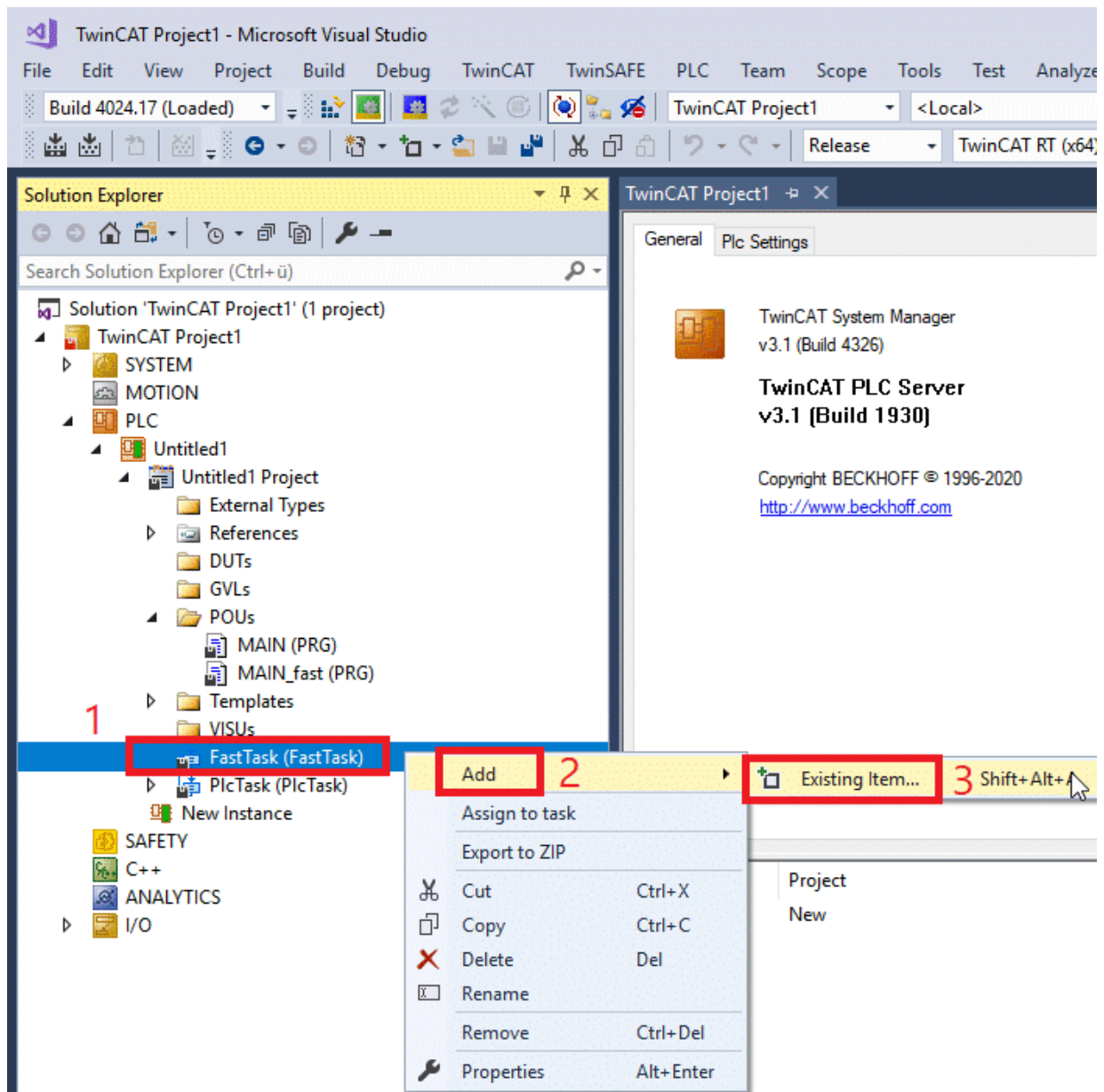
⇒ A task reference must be created so that this task is available to the program part:

4. Right-click on the project (1), select **add** (2) then **taskreference** (3). A window opens. There, select the Task to which the reference should refer (4).

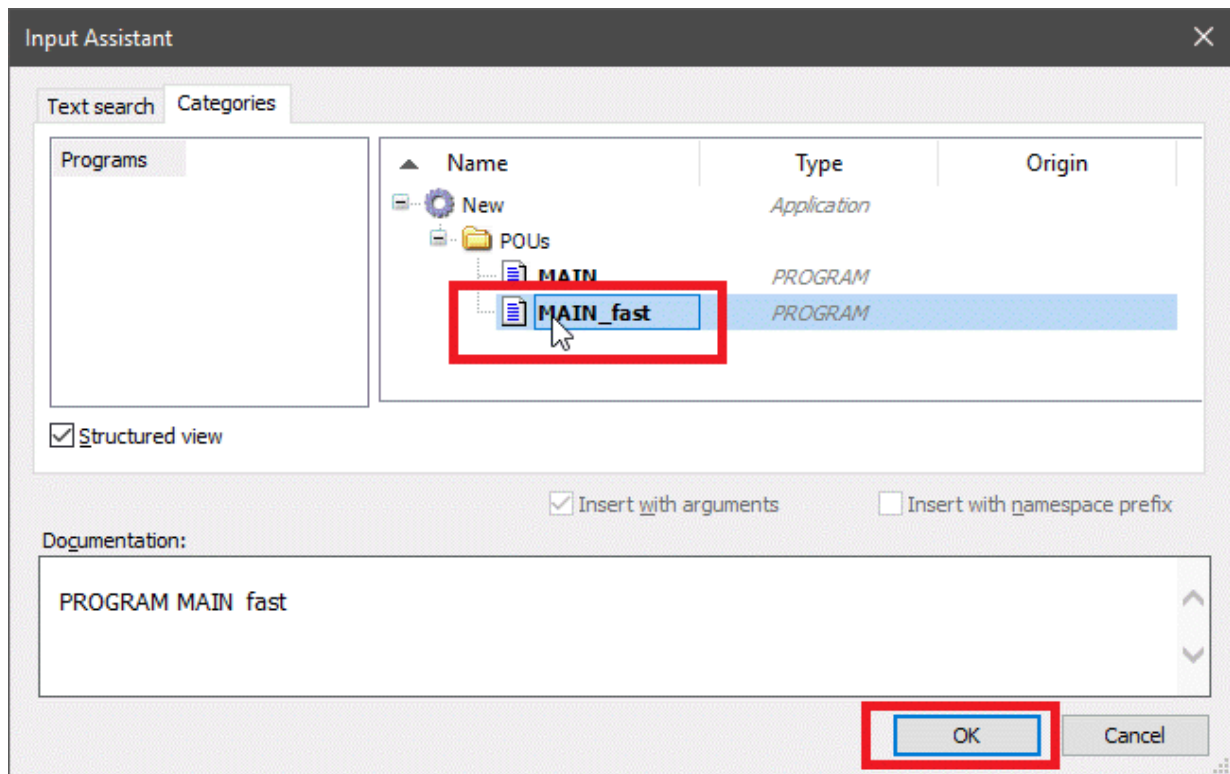


⇒ The task reference now appears in the PLC part below. The function block that is to be called in this task can now be assigned to it.

5. Open a dialog by right-clicking on the task **Fast Task(1)**, select **add** (2) and then **existing item** (3).

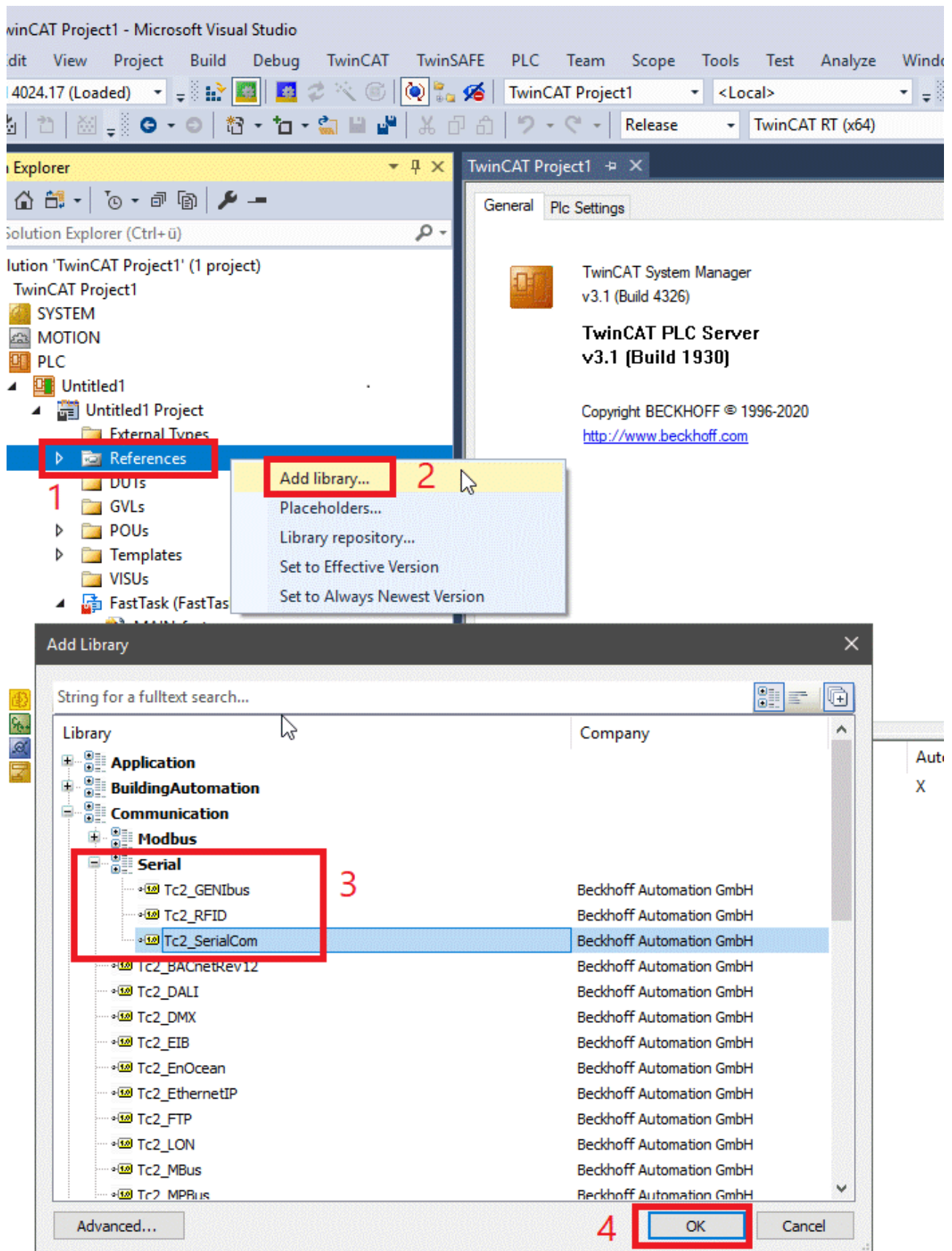


⇒ A window will open where you can select the call program.



Adding the serial communication library

Dragging in the template "FB_BA_WeatherStation_Thies" does not automatically add the required serial communication library. This must be inserted manually:

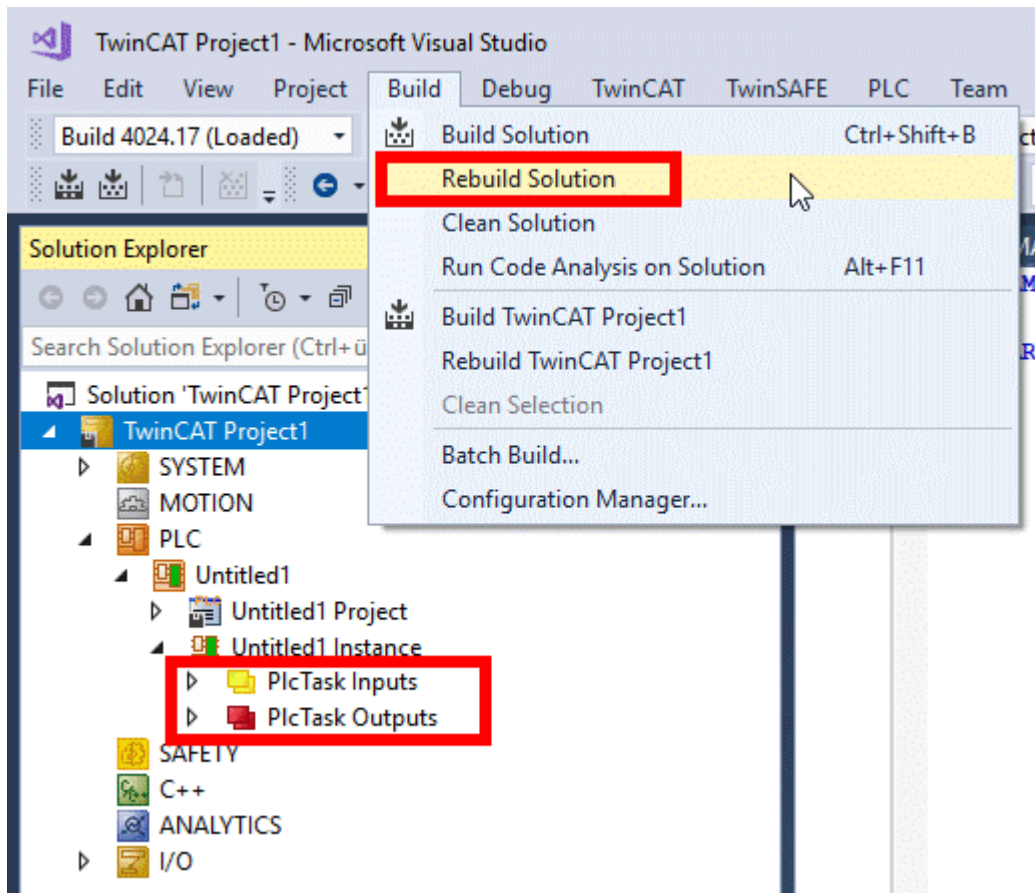


Right-click on "References" (1) and select "Add Library" (2). In the window that opens, select the "Tc2_SerialCom" library (3) and confirm with "OK" (4).

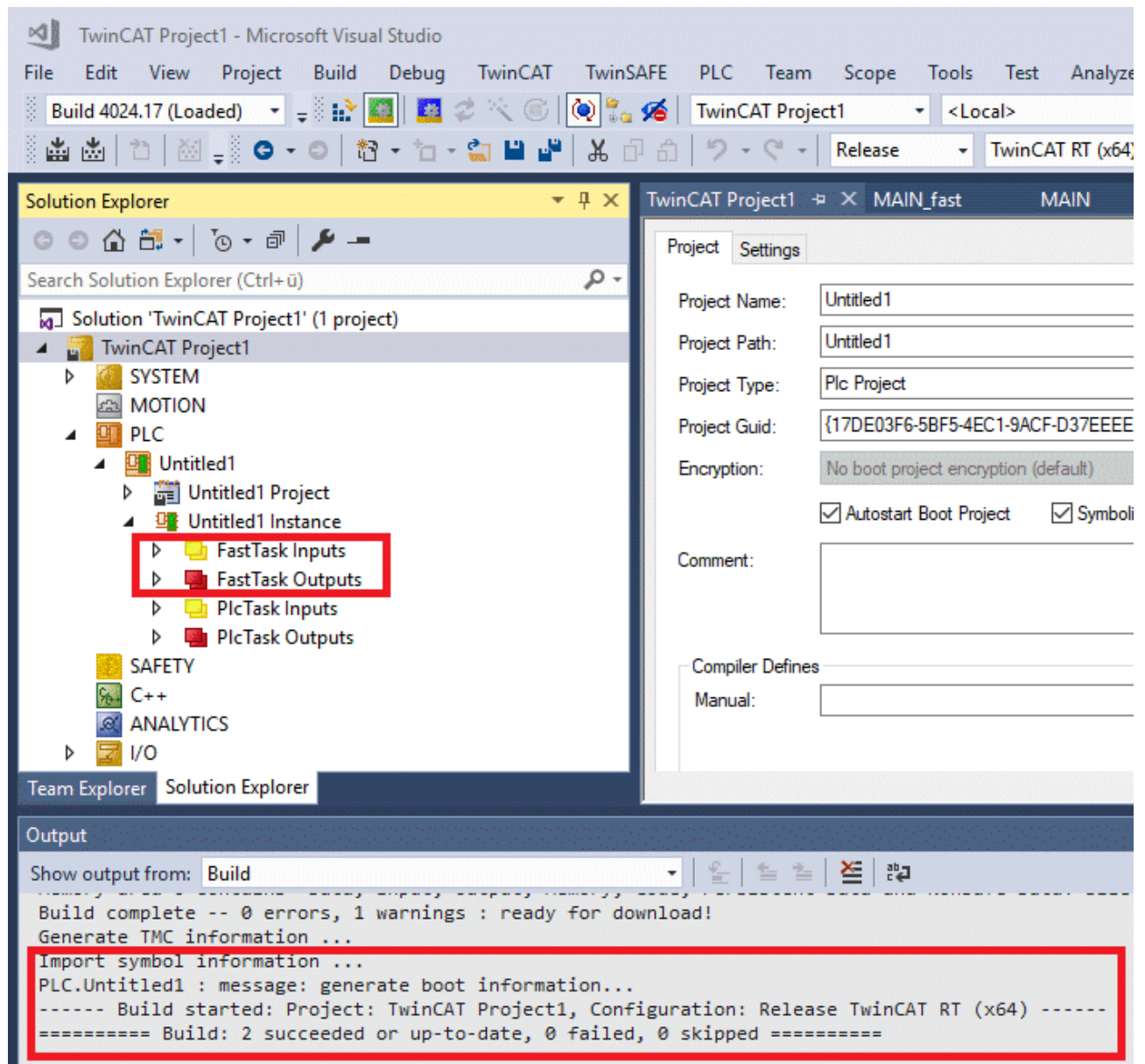
Creating program links

The link variables for the process image of the serial terminal are located in the "FB_BA_WeatherStation_Thies" template. However, just because of the implementation of the template, they are not automatically available for linking. To do this, the solution must be rebuilt once.

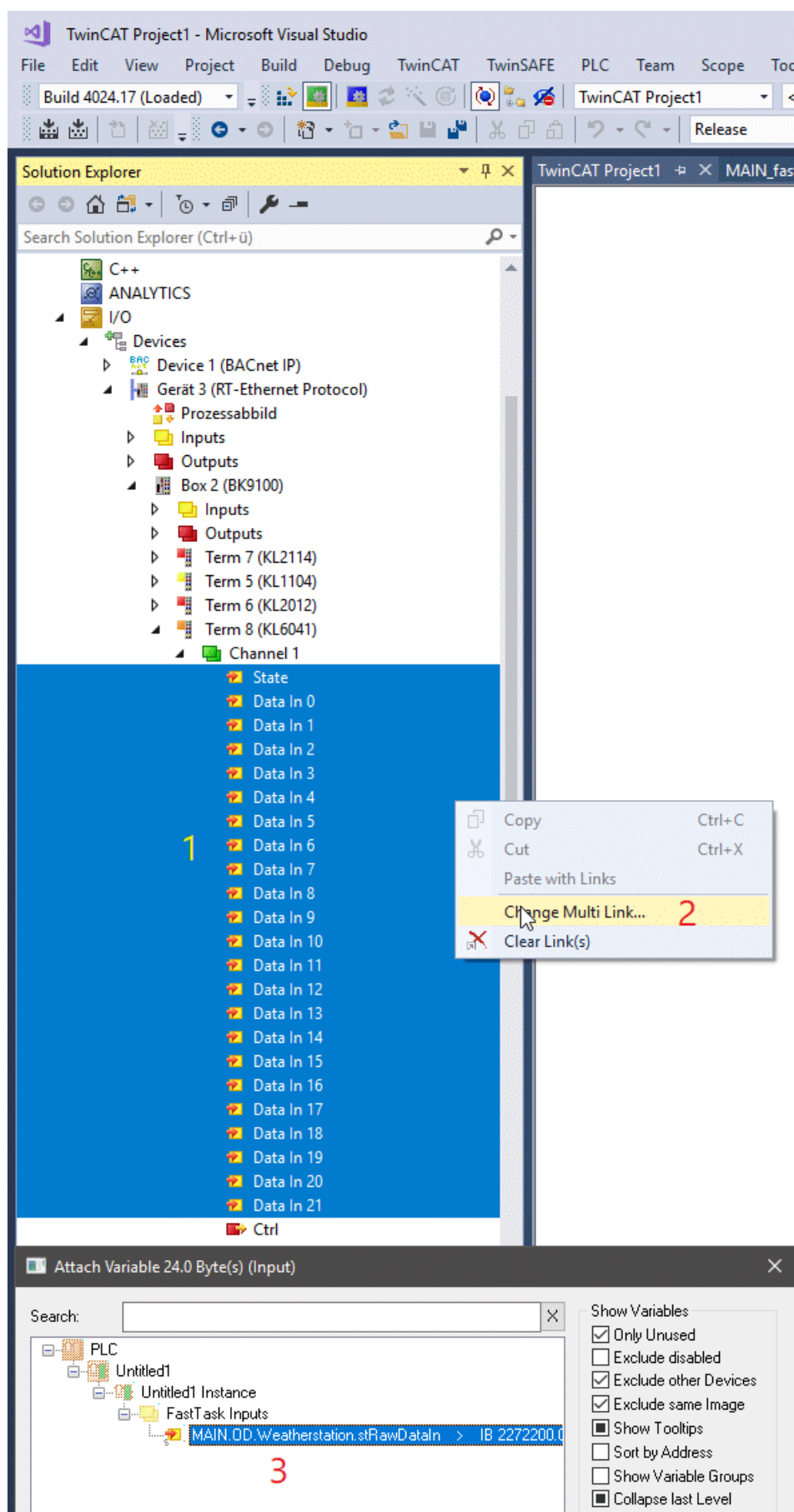
It is important at this point that no other compilation errors occur in the program.



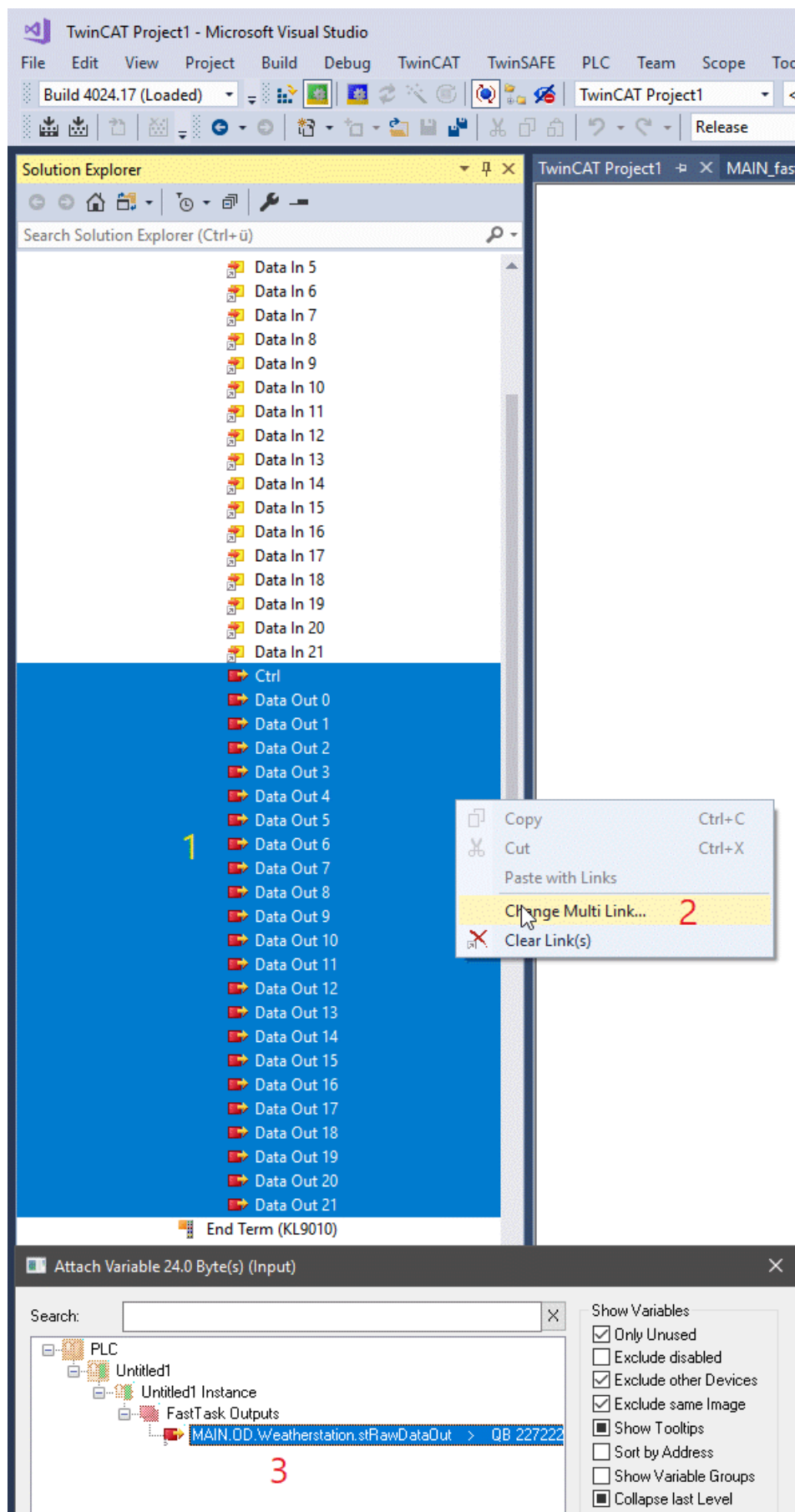
Under "Build", select "Rebuild Solution". At this point, only the already existing PLC task has a process image.



After an error-free creation, the link variables of the template are available and the Fast Task also holds a process image area.

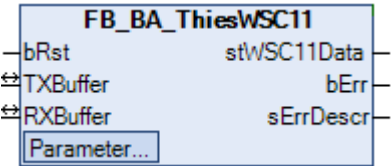


First, all input variables of the terminal process image are marked (click on status and then press the arrow down key while holding down the shift key) (1), then right-click and select "Multi-link" (2). Link to "stRawDataIn" of the template.



The linking of the output data is done analogously.

6.1.4.2.2.9.1.2 FB_BA_ThiesWSC11



Function block for cyclic reading and processing of serial data from a Thies WSC11 weather station.

This function block converts the serial data into a defined structure.

Illustration

```
VAR_INPUT
    bRst          : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
    nUpdateTime      : UDINT;
    nConfigTimeout   : UDINT;
    nMaxCyclesOldData : UDINT;
    nWndDatAvrgIntVal : UDINT;
    bWndSpdLEDOn     : BOOL;
    bTwilgtCalcAvrg  : BOOL;
    nWndDirNorthOffs : UDINT;
    nStHgtAMSL       : UDINT;
    nRainOffDly      : UDINT;
    nDistMsgDly      : UDINT;
END_VAR
VAR_OUTPUT
    stWSC11Data      : ST_BA_WSC11Data;
    bErr             : BOOL;
    sErrDescr        : T_MaxString;
END_VAR
VAR_IN_OUT
    TXBuffer         : ComBuffer;
    RXBuffer         : ComBuffer;
END_VAR
```

🔧 Inputs

Name	Type	Description
bRst	BOOL	A rising edge at this input triggers a software reset within the weather station.

🔧 Inputs CONSTANT PERSISTENT

Name	Type	Description
nUpdateTime	UDINT	Data retrieval interval [500...60000 ms].
nConfigTimeout	UDINT	Timeout for the configuration routine [s].
nMaxCyclesOldData	UDINT	Maximum number of read cycles with identical data: beyond this, it is considered an error - possibly caused by wire break.
nWndDatAvrgIntVal	UDINT	Averaging interval of wind direction and wind speed [1...10 min], 0 = off.
bWndSpdLEDOOn	BOOL	A TRUE indicates the event "Wind" via the blue LED on the weather station.
bTwiLgtCalcAvrg	BOOL	Selection of the value calculation for twilight: FALSE: the sum of the 4 light sensors is used. TRUE: the average value of the 4 light sensors is used.
nWndDirNorthOffs	UDINT	Offset for wind direction [0...360°]. This can be used to correct the north direction.
nStHgtAMSL	UDINT	Station height above sea level [0...3000 m].
nRainOffDly	UDINT	Release delay of rain detection [0...3600 s].
nDistMsgDly	UDINT	Delay of error messages [0...60 s].

🔧 Outputs

Name	Type	Description
stWSC11Data	ST_BA_WSC11Data [► 616]	Output of the data.
bErr	BOOL	Error.
sErrDescr	T_MaxString	Textual error description.

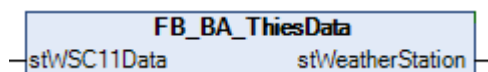
🔧/🔧 Inputs/outputs

Name	Type	Description
TXBuffer / RXBuffer	ComBuffer	Serial data exchange with the reading function block <u>SerialLineControl</u> in the fast task.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.9.1.3 FB_BA_ThiesData



In this function block, data from the Thies weather station are converted so that they can be represented with BACnet objects.

This concerns only the brightness values of the 4 cardinal points: while the weather station outputs these in kilolux, BACnet only provides the unit lux for the representation, therefore the values of the weather station have to be multiplied by 1000.

Illustration

```

FUNCTION_BLOCK FB_BA_ThiesData
VAR_INPUT
    stWSC11Data      : ST_BA_WSC11Data;
END_VAR
VAR_OUTPUT
    stWeatherStation : ST_BA_WeatherStation;
END_VAR

```

 Inputs

Name	Type	Description
stWSC11Data	ST_BA_WSC11Data [► 616]	Data from the Thies weather station.

 Outputs

Name	Type	Description
stWeatherStation	ST_BA_WeatherStation [► 615]	Converted data.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 TwinCAT Building Automation from V5.0.0.0

6.1.4.3 GVLs**6.1.4.3.1 Site**

The site GVL is a list of global variables which are required system-wide by all automation stations of a GA network. The data is used to control and regulate the plant and room automation.

The distribution of the data within the GA network is organized by means of the templates [FB_BA_AdsComClient \[► 620\]](#) and [FB_BA_AdsComServer \[► 622\]](#).

Regardless of whether it is the server or a client of the data within the GA network, all templates that create or read data always read or write to the GVL site.
Thus, the same GVL site is located in each automation station.

In the clients, e.g. of a floor controller for room automation, the data within the template [FB_BA_AdsComClient \[► 620\]](#) is read from the server via ADS and copied to the GVL. All room automation templates then access this data from the GVL site.

In the IPC, which provides the data within the GA network, the template [FB_BA_AdsComServer \[► 622\]](#) is called.

The system topology of the TF8040 templates provides that this data is generated by an IPC within the building and distributed with the function blocks [FB_BA_RawPublisher](#) and [FB_BA_RawSubscriber](#).

Illustration

```

VAR_GLOBAL
    Self_NetId      : T_BA_MedString := '127.0.0.1.1.1';
    ACE01_NetId     : T_BA_MedString := Self_NetId;
    ACE02_NetId     : T_BA_MedString := Self_NetId;
    ACE03_NetId     : T_BA_MedString := Self_NetId;
    ACE04_NetId     : T_BA_MedString := Self_NetId;
    ACE05_NetId     : T_BA_MedString := Self_NetId;

    GeneralSettings_Subject : T_BA_MedString := 'GeneralSettings';
    GeneralSettings_NetId   : T_BA_MedString := Self_NetId;

```



```

WeatherStation_Subject      : T_BA_MedString := 'WeatherStation';
WeatherStation_NetId       : T_BA_MedString := Self_NetId;

FacadeNorth_NetId          : T_BA_MedString := ACE01_NetId;
FacadeNorth_Subject        : T_BA_MedString := 'FcdNorth.FacadeSunBlind';
FacadeEast_NetId           : T_BA_MedString := ACE01_NetId;
FacadeEast_Subject         : T_BA_MedString := 'FcdEast.FacadeSunBlind';
FacadeSouth_NetId          : T_BA_MedString := ACE01_NetId;
FacadeSouth_Subject        : T_BA_MedString := 'FcdSouth.FacadeSunBlind';
FacadeWest_NetId           : T_BA_MedString := ACE01_NetId;
FacadeWest_Subject         : T_BA_MedString := 'FcdWest.FacadeSunBlind';

Building_NetId             : T_BA_MedString := ACE01_NetId;
BuildingAlarms_Subject     : T_BA_MedString := 'BuildingAlarms';
BuildingMode_Subject       : T_BA_MedString := 'BuildingMode';
BuildingEnergyLevel_Subject : T_BA_MedString := 'BuildingEnergyLevel';
BuildingSpRmT_Subject      : T_BA_MedString := 'BuildingSpRmT';
BuildingSunProtection_Subject : T_BA_MedString := 'BuildingSunProtection';

stGeneralSettings          : ST_BA_GeneralSettings; // published by FB_BA_Settings
bGeneralSettings_Error     : BOOL;
eBuildingEnergyLevel       : E_BA_EnergyLvlEx; // published by FB_BA_BuildingEnergyLevel
bBuildingEnergyLevel_Error : BOOL;
eBuildingMode              : E_BA_BuildingMode; // published by FB_BA_BuildingMode
bBuildingMode_Error        : BOOL;
stBuildingSunBlind         : ST_BA_BuildingSunBlind; // published by FB_BA_BuildingSunprotecti
on
bBuildingSunBlind_Error    : BOOL;
stBuildingAlarms           : ST_BA_BuildingAlarms; // published by FB_BA_BuildingAlarms
bBuildingAlarms_Error      : BOOL;
stWeatherStation           : ST_BA_WeatherStation; // published by FB_BA_WeatherStation_xxx
bWeatherStation_Error      : BOOL;
stFacadeNorthSunBlind      : ST_BA_Facade; // published by FB_BA_Facade
bFacadeNorthSunBlind_Error : BOOL;
stFacadeEastSunBlind       : ST_BA_Facade; // published by FB_BA_Facade
bFacadeEastSunBlind_Error  : BOOL;
stFacadeSouthSunBlind      : ST_BA_Facade; // published by FB_BA_Facade
bFacadeSouthSunBlind_Error : BOOL;
stFacadeWestSunBlind       : ST_BA_Facade; // published by FB_BA_Facade
bFacadeWestSunBlind_Error  : BOOL;
stBuildingSpRmT            : ST_BA_SpRmT; // Published by FB_BA_BuildingSpRmT
bBuildingSpRmT_Error       : BOOL;
END_VAR

```

6.1.4.3.2 SysTime

The global variable list contains the local NT system time of the TwinCAT system.

The variables may only be written once per controller.

```

{attribute 'qualified_only'}
VAR_GLOBAL
  stSystemTime      : TIMESTRUCT;
  stSystemTimeUTC   : TIMESTRUCT;
  dtSystemTime      : DT;
  dtSystemTimeUTC   : DT;
END_VAR

```

6.1.4.3.3 EventClassesID

EC_ID

The global variable list specifies the event class number for the message class objects and the alarm-capable objects.

Illustration

```

{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
  NC10 : UDINT := 10;
  NC20 : UDINT := 20;
  NC30 : UDINT := 30;
  NC40 : UDINT := 40;
  NC50 : UDINT := 50;
  NC51 : UDINT := 51;

```



```

NC60      : UDINT := 60;
NC70      : UDINT := 70;
NC80      : UDINT := 80;
END_VAR

```

Name	Type	Description
NC10	UDINT	Danger to life
NC20	UDINT	Safety message
NC30	UDINT	Alarm message
NC40	UDINT	Fault message
NC50	UDINT	Maintenance message
NC51	UDINT	Maintenance message
NC60	UDINT	System message
NC70	UDINT	Manual intervention
NC80	UDINT	Other messages

6.1.4.3.4 BA2_Param

The global variable list contains general parameters to initialize objects.

Illustration

```

{attribute 'qualified_only'}
{attribute 'strict'}
VAR_GLOBAL CONSTANT
    fAP                                     : REAL := 1013.25;
    nDefTimeDelay_ToAbnormal               : UDINT := 1;
    nDefTimeDelay_ToNormal                 : UDINT := 1;
    fDefCOVIncrement                       : REAL := 0.1;
    fDefLimitDeadband                     : REAL := 0.0;
    nLoop_DefOpMode                       : E_BA_PIDMode := E_BA_PIDMode.ePID;
    stTrend_DefStartTime                   : ST_BA_DateTime := ();
    stTrend_DefStopTime                   : ST_BA_DateTime := ();
    nTrend_BufferSize                     : UDINT := 500;
    bTrend_DefLogEnable                    : BOOL := FALSE;
    bTrend_DefStopOnFull                   : BOOL := FALSE;
    nTrend_DefLogInterval                  : UDINT := 90;
    nTrend_DefNotificationThreshold        : UDINT := 50;
    eTrend_DefLoggingType                  : E_BA_LoggingType := E_BA_LoggingType.ePolled;
END_VAR
VAR_GLOBAL
    bBlink                                 : BOOL;
    bDlyStartPLC                           : BOOL := TRUE;
END_VAR

```

VAR_GLOBAL CONSTANT

Name	Type	Description
fAP	REAL	Global constant Hydrostatic air pressure. The mean air pressure of the earth's atmosphere at sea level is 1013.25 hPa.
nDefTimeDelay_ToAbnormal	UDINT	
nDefTimeDelay_ToNormal	UDINT	
fDefCOVIncrement	REAL	
fDefLimitDeadband	REAL	
nLoop_DefOpMode	E_BA_PIDMode	
stTrend_DefStartTime	ST_BA_DateTime	
stTrend_DefStopTime	ST_BA_DateTime	
nTrend_BufferSize	UDINT	
bTrend_DefLogEnable	BOOL	
bTrend_DefStopOnFull	BOOL	
nTrend_DefLogInterval	UDINT	
nTrend_DefNotificationThreshold	UDINT	
eTrend_DefLoggingType	E_BA_LoggingType	

VAR_GLOBAL

Name	Type	Description
bBlink	BOOL	Global blink pulse. The blink pulse is generated in the template <code>FB_BA_ControlCabinetBasic</code> [► 677].
bDlyStartPLC	BOOL	The variable is set to TRUE delayed after a restart of the PLC. In the template <code>FB_BA_Device</code> [► 672] this delay is implemented by the timing element <code>tonDlyStartPLC</code> .

6.2 HMI

6.2.1 TcHmiBa

TcHmiBa is an extension of the [TwinCAT HMI](#) for building automation applications. The integrator should be made as easy as possible to create an HMI, for example for a heating system, ventilation system or the entire HVAC technology of a building. With this solution it is also possible to map and operate the room automation in a building. TcHmiBa supports some functions of a classic building control system (BMS).



Note that TcHmiBa is **not** a BMS.

Contents

The easy creation of HMIs for building automation is enabled by TcHmiBa by providing various [controls](#) [► 822] and [icons](#) [► 984].

Generic HMI

For integrators using the complete TF8040 solution, there is also the option of accessing generic functions [► 44] from TcHmiBa, which further simplify and accelerate engineering.

System requirements

- TF8040 (current version)
- [TE2000](#)
- Google Chrome / Chromium (support for other browsers to follow)

Open source software

Various open source software is used in the components of TcHmiBa.
The license texts can be found in the respective folders:

- Development system:
 - TcHmiBaFramework: *\$ProjectPath\$/Packages/Beckhoff.TwinCAT.HMI.BA.Framework/runtimes/native/Legal*
 - BaSiteExtension: *\$ProjectPath\$/Packages/Beckhoff.TwinCAT.HMI.BA.BaSite/runtimes/any/native/Legal*
- Target system:
 - TcHmiBaFramework: *C:\ProgramData\Beckhoff\TF2000 TwinCAT 3 HMI Server\service\TcHmiProject\www\Beckhoff.TwinCAT.HMI.BA.Framework\Legal*
 - BaSiteExtension: *C:\ProgramData\Beckhoff\TF2000 TwinCAT 3 HMI Server\service\TcHmiProject\BaSite\Legal*

6.2.1.1 Controls

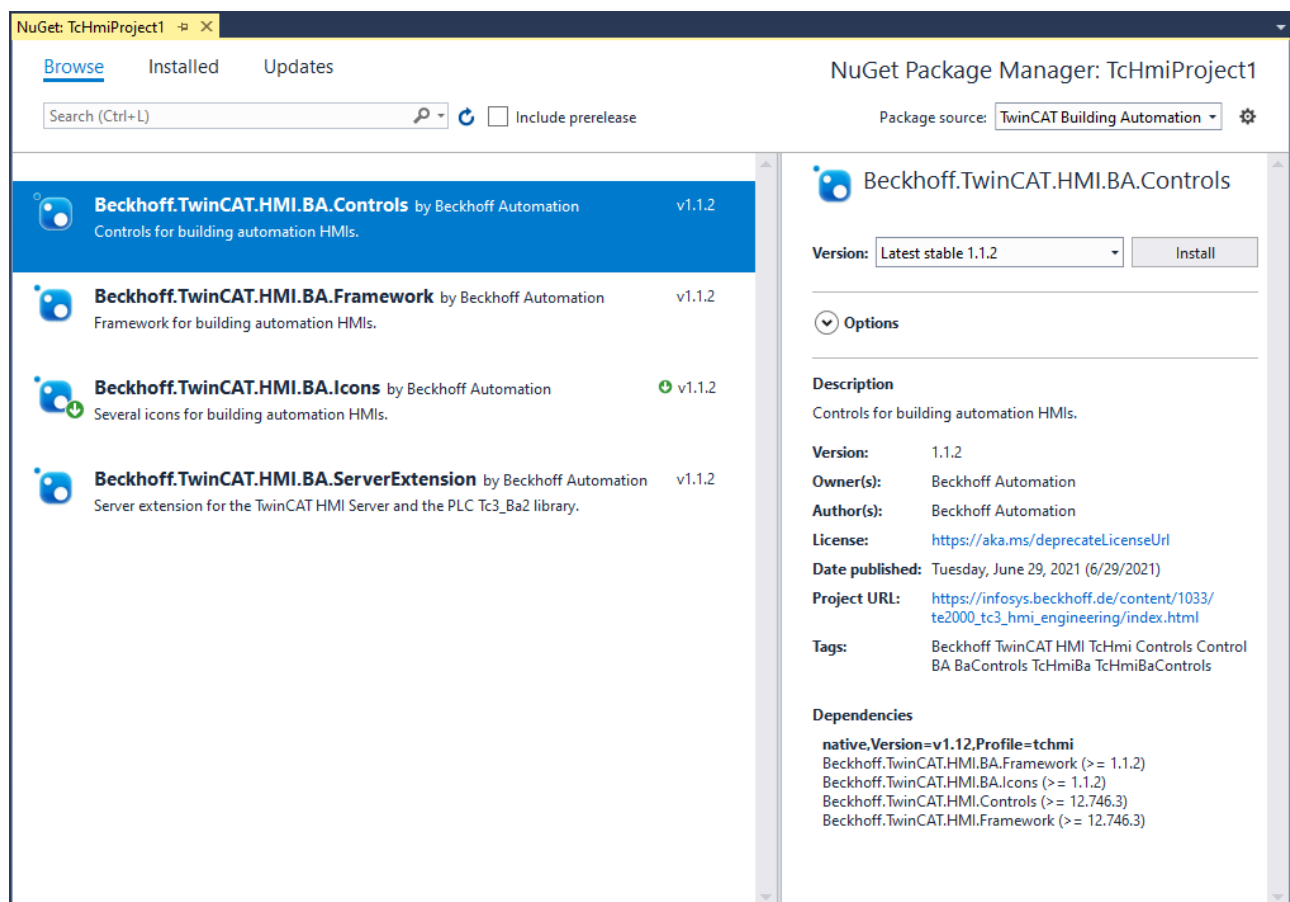
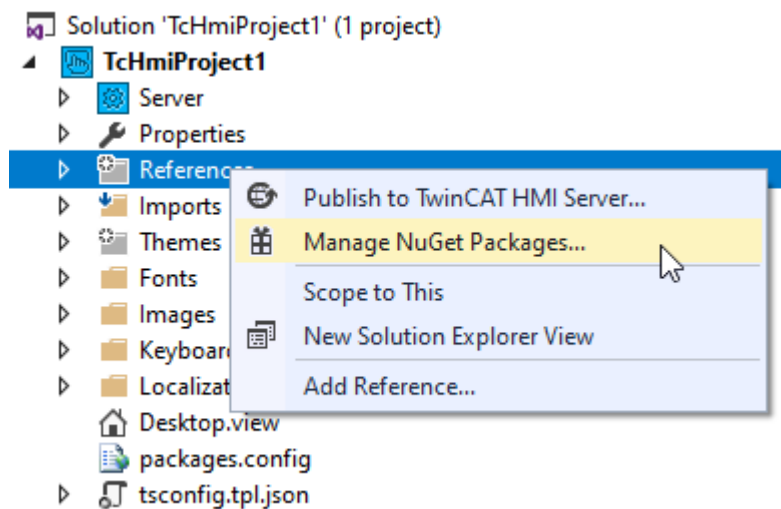
The following pages describe the content and functions of the NuGet package **Beckhoff.TwinCAT.HMI.BA.Controls**. The main focus is on the attributes and functions of the controls that can be used in the Designer.

System requirements

- TF8040 (current version)
- [TE2000](#)
- Google Chrome / Chromium (support for other browsers to follow)

Installation

The NuGet package **Beckhoff.TwinCAT.HMI.BA.Controls** must be installed in order to use the controls and functions.



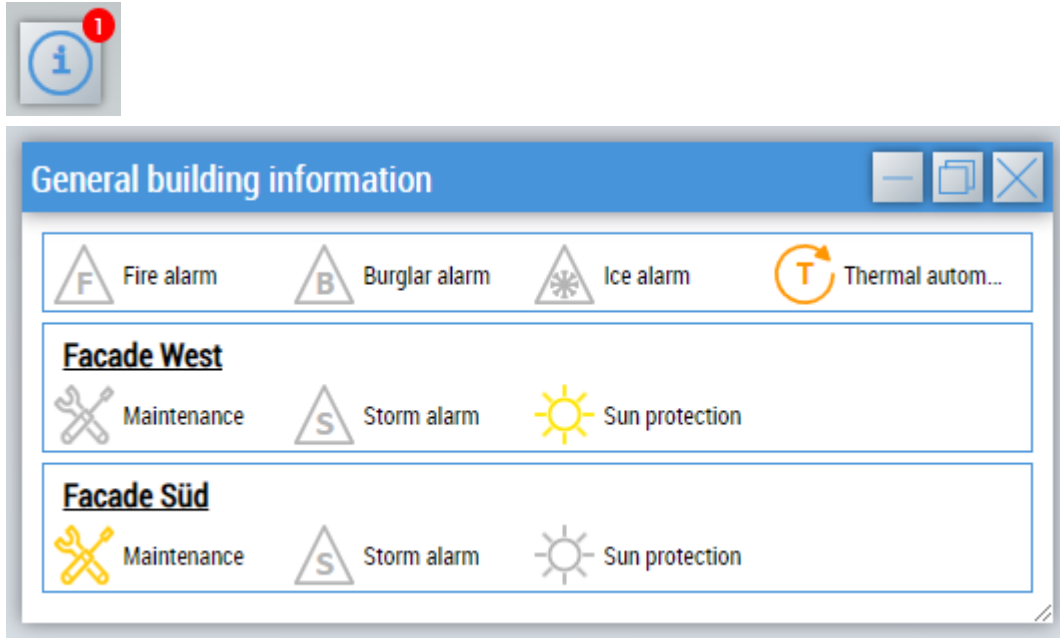
The package depends on the following packages and therefore installs them as well:

- [Beckhoff.TwinCAT.HMI.Framework](#)
- [Beckhoff.TwinCAT.HMI.Controls](#)
- [Beckhoff.TwinCAT.HMI.BA.Framework](#) [► 996]
- [Beckhoff.TwinCAT.HMI.BA.Icons](#) [► 984]

6.2.1.1.1 BuildingGeneral

6.2.1.1.1.1 BuildingInformation

The **BuildingInformation** control initially appears like a normal [button](#) [► 884]. It is designed to display various building or facade specific information.



Use

Use on any page (e.g. in the header).

Features

If the window is not open, the amount of relevant information is displayed in the corner of the button. This includes:

- Fire alarm
- Burglar alarm
- Ice alarm
- Maintenance
- Storm alarm

The status of the sun protection or the thermal automatic can only be viewed when the window is open. If an alarm becomes active, all controls that depend on building information are notified. This applies to:

- [Sunblind](#) [► 961]
- [Window](#) [► 954]

Attributes

The control inherits from the [button](#) [► 884] and thus has the same attributes. In addition, there are the following attributes.

Common

FireAlarm

```
tchmi:general#/definitions/Boolean
```

If TRUE, the fire alarm is displayed (see [Boolean](#)).

BurglarAlarm

tchmi:general#/definitions/Boolean

If TRUE, the burglar alarm is displayed (see [Boolean](#)).

IceAlarm

tchmi:general#/definitions/Boolean

If TRUE, the ice alarm is displayed (see [Boolean](#)).

ThermalAutomatic

tchmi:general#/definitions/Boolean

If TRUE, the thermal automatic is displayed (see [Boolean](#)).

Facades

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingInformation.Facades

Create different facades that can then be referenced by corresponding controls (see above) (see [Facades](#) [► 825]).

6.2.1.1.1.1 Types**Facades**

Collection of [FacadeInfos](#) [► 825].

```
"TcHmi.BuildingAutomation.Controls.BuildingInformation.Facades": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "array",
  "title": "Facades",
  "items": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.BuildingInformation.FacadeInfo"
  }
}
```

FacadeInfo

Schema for the attribute [Facades](#) [► 825] of the [BuildingInformation](#) [► 824] - Controls.

```
"TcHmi.BuildingAutomation.Controls.BuildingInformation.FacadeInfo": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "title": "FacadeInfo",
  "description": "Defines a facade",
  "engineeringColumns": [
    "name"
  ],
  "propertiesMeta": [
    {
      "name": "name",
      "category": "Common",
      "displayName": "Name",
      "description": "The name of the facade (e.g. west).",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "maintenance",
      "category": "Common",
      "displayName": "Maintenance",
      "description": "Flag if maintenance is active.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "stormProtection",
      "category": "Common",

```

```

    "displayName": "Storm Protection",
    "description": "Flag if storm protection is active.",
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "sunProtection",
    "category": "Common",
    "displayName": "Sun Protection",
    "description": "Flag if sun protection is active.",
    "defaultValue": null,
    "defaultValueInternal": null
  }
],
"properties": {
  "name": {
    "type": "string"
  },
  "maintenance": {
    "type": "boolean"
  },
  "stormProtection": {
    "type": "boolean"
  },
  "sunProtection": {
    "type": "boolean"
  }
}
}
}

```

6.2.1.1.1.2 Legend

The **Legend** control displays all icons on the active web page with their description.

Use

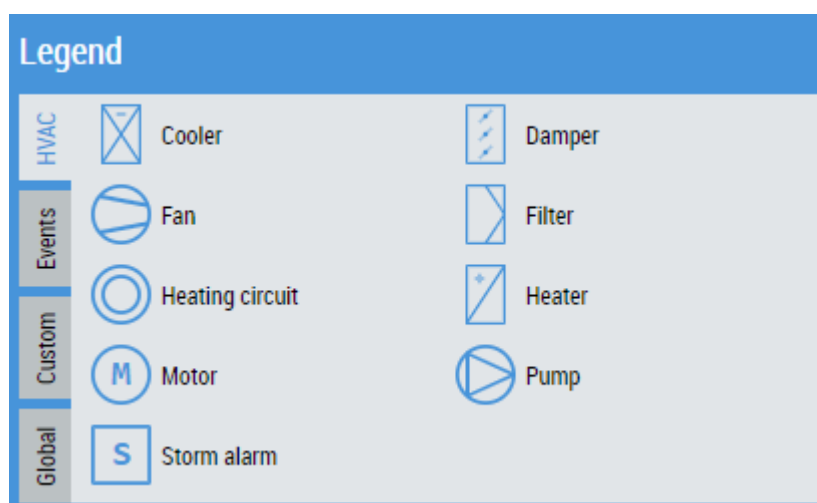
Use on any page where you want to display an explanation of the icons.

Features

Provides static and dynamic display of icons and allows adding additional icons.

Display

The icons can be displayed statically on the web page or dynamically (event-driven) by calling the `OpenLegendDialog` function.



Additional icons

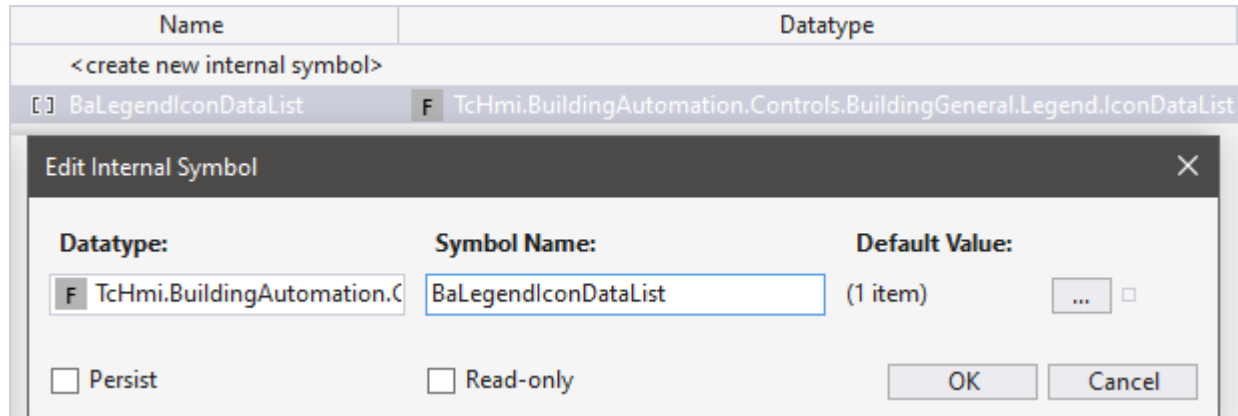
Local

Additional icons can be added to the respective instance of the legend via the `IconDataCustom` attribute. When the function is called, the attribute is in the parameter list.

Global

Icons to be displayed by each legend instance must be added via a TwinCAT HMI Internal Symbol.

The symbol must have the name `BaLegendIconDataList` and be of type `TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList`.



Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

Common

ShowHeadline

`tchmi:general#/definitions/Boolean`

Sets the visibility of the headline (see [Boolean](#)).

EntryWidth

`tchmi:framework#/definitions/MeasurementValue`

Width of an entry.

EntryWidthUnit

`tchmi:framework#/definitions/MeasurementUnit`

Unit of width of an entry.

TabPosition

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position`

Position of the tabs (see [Position](#) [► 925]).

IconDataSource

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataSource`

Selection of entries to be displayed (see [IconDataSource](#) [► 828]).

IconDataCustom

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList`

List with additional entries (see [IconDataList](#) [► 828]).

6.2.1.1.1.2.1 Types

IconDataList

Schema for the attribute `IconDataCustom` [► 827] of the `Legend` [► 826] control.

```
"TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList": {
  "title": "IconDataList",
  "type": "array",
  "items": {
    "oneOf": [
      {
        "title": "Icon",
        "type": "object",
        "additionalProperties": false,
        "engineeringColumns": [
          "Name"
        ],
        "propertiesMeta": [
          {
            "name": "path",
            "displayName": "Path",
            "category": "Common",
            "displayPriority": 10,
            "description": "Path of the icon.",
            "defaultValue": null,
            "defaultValueInternal": null
          },
          {
            "name": "category",
            "displayName": "Category",
            "category": "Common",
            "displayPriority": 10,
            "description": "Category of the icon.",
            "defaultValue": null,
            "defaultValueInternal": null
          },
          {
            "name": "description",
            "displayName": "Description",
            "category": "Common",
            "displayPriority": 10,
            "description": "Description of the icon.",
            "defaultValue": null,
            "defaultValueInternal": null
          }
        ],
        "properties": {
          "path": {
            "type": [ "string", "null" ]
          },
          "category": {
            "type": [ "string", "null" ]
          },
          "description": {
            "type": [ "string", "null" ]
          }
        }
      }
    ]
  }
}
```

IconDataSource

Schema for the attribute `IconDataSource` [► 827] of the `Legend` [► 826] control.

```
"TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataSource": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "integer",
  "enum": [ 0, 1, 2, 3, 4, 5, 6 ],
  "options": [
    {
      "label": "All",
      "value": 0
    },
    {
      "label": "OnlyBa",
```

```
    "value": 1
  },
  {
    "label": "OnlyCustom",
    "value": 2
  },
  {
    "label": "OnlyCustomGlobal",
    "value": 3
  },
  {
    "label": "BaAndCustom",
    "value": 4
  },
  {
    "label": "BaAndCustomGlobal",
    "value": 5
  },
  {
    "label": "CustomAndCustomGlobal",
    "value": 6
  }
]
```

6.2.1.1.1.3 WeatherStation

The **WeatherStation** template displays the weather data.

Use

Use on any page where a template of the **WeatherStation** type is to be displayed.

Compatibility

The [BaTemplateDescription \[► 963\]](#) supports the following BaObjects.

Temperature	
Temperature Outside	0.7 °C
Temperature Dewpoint	-0.5 °C

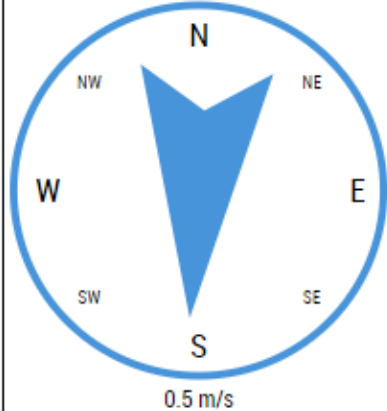
Humidity	
Absolute Humidity	4.64 g/kg
Relative Humidity	91.3 %
Rain	No

Brightness	
Brightness North	1800 lx
Brightness East	2100 lx
Brightness South	1800 lx
Brightness West	1600 lx

Air Pressure	
Absolute Air Pressure	1005 hPa
Relative Air Pressure	1016 hPa

Sun	
Sun Azemut	136.6 °
Sun Elevation	7.7 °
Dawn	999 °C
Global Radiation	41 W/m²

Wind	
Wind Direction	184 °
Wind Speed	0.5 m/s

Wind compass	
	

Subelements:

Symbol name	PLC template	Required	Description
Dstb	FB_BA_SensorBinary		Fault
Rain	FB_BA_SensorBinary		Rain
WthT	FB_BA_SensorAnalog	X	Outside temperature
DewPtT	FB_BA_SensorAnalog	X	Dew point
PrssAbs	FB_BA_SensorAnalog	X	Air pressure (absolute)
PrssRel	FB_BA_SensorAnalog	X	Air pressure (relative)
HumAbs	FB_BA_SensorAnalog	X	Air humidity (absolute)
HumRel	FB_BA_SensorAnalog	X	Air humidity (relative)
Brightness	FB_BA_MeasuredValueCardinal		Brightness
Dawn	FB_BA_SensorBinary	X	Dawn
GlobRadn	FB_BA_SensorAnalog	X	Solar radiation
WndDir	FB_BA_SensorAnalog	X	Wind direction
WndSpd	FB_BA_SensorAnalog	X	Wind speed
Latd	FB_BA_SensorAnalog	X	Latitude
Lngt	FB_BA_SensorAnalog	X	Longitude
SunAzm	FB_BA_SensorAnalog	X	Sun direction
SunElv	FB_BA_SensorAnalog	X	Sun elevation

Hierarchy:

- BaObject
 - Dstb
 - Rain
 - WthT
 - DewPtT
 - PrssAbs
 - PrssRel
 - HumAbs
 - HumRel
 - Brightness
 - Dawn
 - GlobRadn
 - WndDir
 - WndSpd
 - Latd
 - Lngt
 - SunAzm
 - SunElv

Corresponds to the PLC templates:

- FB_BA_WeatherStation_Draft
- FB_BA_WeatherStation_Thies

Attributes

The control inherits from BaseTemplate and thus has the same attributes. In addition, there are the following attributes.

Common**ShowPosition**

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the position can be seen (see [Boolean](#)).

ShowTemperature

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the temperature can be seen (see [Boolean](#)).

ShowAirPressure

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the air pressure can be seen (see [Boolean](#)).

ShowHumidity

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for humidity can be seen (see [Boolean](#)).

ShowSun

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the sun can be seen (see [Boolean](#)).

ShowBrightness

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for brightness can be seen (see [Boolean](#)).

ShowWind

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the wind can be seen (see [Boolean](#)).

ShowWindCompass

```
tchmi:general#/definitions/Boolean
```

Determines whether the wind compass can be seen (see [Boolean](#)).

6.2.1.1.1.3.1 Types**BaTemplateDescription**

Schema for the [BaTemplateDescription](#) [► 933] attribute of the [WeatherStation](#) [► 829] template.

```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "definitions": {
    "Tchmi.BuildingAutomation.Controls.BuildingGeneral.WeatherStation": {
      "type": "object",
      "frameworkInstanceOf": "Tchmi.BuildingAutomation.Controls.System.BaseTemplate",
      "frameworkControlType": "WeatherStation",
      "frameworkControlNamespace": "Tchmi.BuildingAutomation.Controls.BuildingGeneral"
    },
    "Tchmi.BuildingAutomation.Controls.BuildingGeneral.WeatherStation.BaTemplateDescription": {
      "$schema": "http://json-schema.org/draft-04/schema",
      "title": "BaTemplateDescription",
      "type": "object",
      "propertiesMeta": [
        {
          "name": "Dstb",
```

```

        "displayName": "Dstb",
        "description": "The disturbance of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Rain",
        "displayName": "Rain",
        "description": "The rain of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "WthT",
        "displayName": "WthT",
        "description": "The outisde temperature of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "DewPtT",
        "displayName": "DewPtT",
        "description": "The dew point of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "PrssAbs",
        "displayName": "PrssAbs",
        "description": "The absolute pressure of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "PrssRel",
        "displayName": "PrssRel",
        "description": "The relative pressure of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "HumAbs",
        "displayName": "HumAbs",
        "description": "The absolute humidty of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "HumRel",
        "displayName": "HumRel",
        "description": "The relative humidty of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Brightness",
        "displayName": "Brightness",
        "description": "The brightness of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Dawn",
        "displayName": "Dawn",
        "description": "The dawn of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "GlobRadn",
        "displayName": "GlobRadn",
        "description": "The global radiation of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "WndDir",
        "displayName": "WndDir",
        "description": "The wind direction of the weather station.",
        "defaultValue": null,

```

```

        "defaultValueInternal": null
    },
    {
        "name": "WndSpd",
        "displayName": "WndSpd",
        "description": "The wind speed of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Latd",
        "displayName": "Latd",
        "description": "The latitude of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Lngt",
        "displayName": "Lngt",
        "description": "The longitude of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "SunAzm",
        "displayName": "SunAzm",
        "description": "The sun azemut of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "SunElv",
        "displayName": "SunElv",
        "description": "The sun elevation of the weather station.",
        "defaultValue": null,
        "defaultValueInternal": null
    }
],
"properties": {
    "Dstb": {
        "type": "string"
    },
    "Rain": {
        "type": "string"
    },
    "WthT": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "DewPtT": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "PrssAbs": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "PrssRel": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "HumAbs": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "HumRel": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "Brightness": {
        "type": "string"
    },
    "Dawn": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "GlobRadn": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    }
},

```

```

        "WndDir": {
            "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
        },
        "WndSpd": {
            "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
        },
        "Latd": {
            "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
        },
        "Lngt": {
            "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
        },
        "SunAzm": {
            "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
        },
        "SunElv": {
            "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
        }
    }
}
}
}
}

```

6.2.1.1.2 Common

6.2.1.1.2.1 Calendar

The control **Calendar** is used to display and manage exceptions to a schedule and select a date.

Use

Use on any page where a date is to be selected.

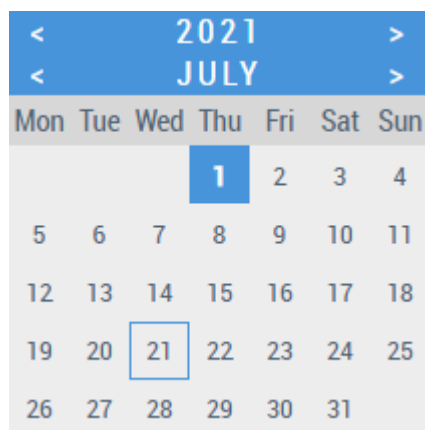
If a Schedule object is passed to the [BaObject](#) [► 837] attribute in the EventCalendar display mode, the exceptions of a time schedule can also be displayed or edited.

Features

Provides two different display modes, the ability to manage exceptions and a color highlighting of related exceptions when the mouse pointer is over them.

DatePicker

A space-saving view of a calendar. Returns the selected date via an event.

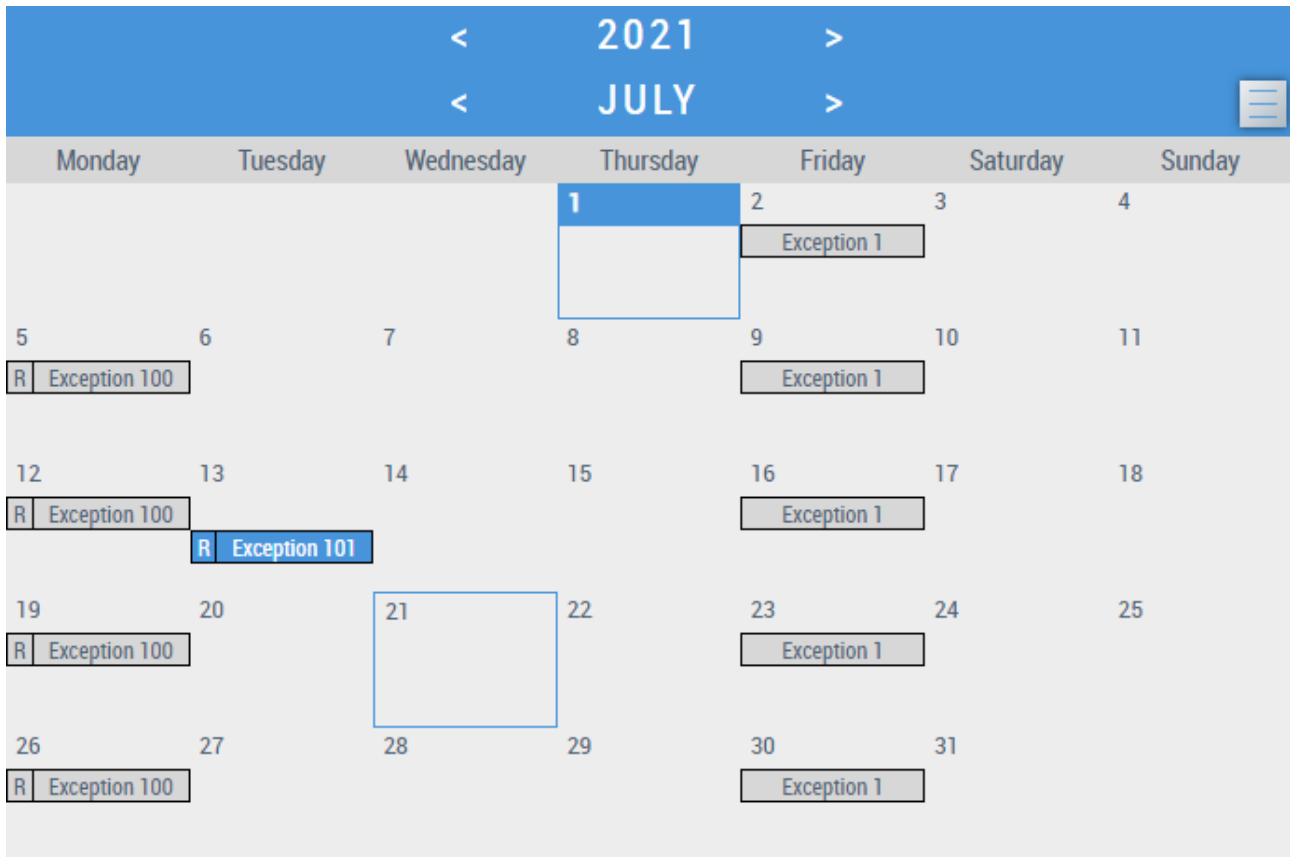


EventCalendar

The EventCalendar is available in the month view and in the year view.

Month view

An event-oriented view of a calendar. It returns the selected date via an event and offers the option to manage an exception by clicking on it.



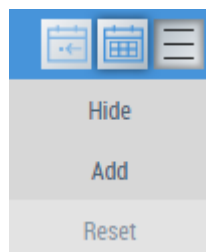
Year view

An overview of all days of the year with marking of the days on which at least one exception is active. Selecting a month changes to the month view.

< 2021 >																							
Mon Tue Wed Thu Fri Sat Sun							Mon Tue Wed Thu Fri Sat Sun							Mon Tue Wed Thu Fri Sat Sun									
January	4	5	6	7	1	2	3	February	1	2	3	4	5	6	7	March	1	2	3	4	5	6	7
	11	12	13	14	8	9	10		8	9	10	11	12	13	14		8	9	10	11	12	13	14
	18	19	20	21	15	16	17		15	16	17	18	19	20	21		15	16	17	18	19	20	21
	25	26	27	28	22	23	24		22	23	24	25	26	27	28		22	23	24	25	26	27	28
April	5	6	7	8	1	2	3	May	3	4	5	6	7	8	9	June	7	8	9	10	11	12	13
	12	13	14	15	9	10	11		10	11	12	13	14	15	16		14	15	16	17	18	19	20
	19	20	21	22	16	17	18		17	18	19	20	21	22	23		21	22	23	24	25	26	27
	26	27	28	29	23	24	25		24	25	26	27	28	29	30		28	29	30				
July	5	6	7	8	1	2	3	August	2	3	4	5	6	7	8	September	6	7	1	2	3	4	5
	12	13	14	15	9	10	11		9	10	11	12	13	14	15		13	14	8	9	10	11	12
	19	20	21	22	16	17	18		16	17	18	19	20	21	22		20	21	15	16	17	18	19
	26	27	28	29	23	24	25		23	24	25	26	27	28	29		27	28	22	23	24	25	26
October	4	5	6	7	1	2	3	November	1	2	3	4	5	6	7	December	6	7	1	2	3	4	5
	11	12	13	14	8	9	10		8	9	10	11	12	13	14		13	14	8	9	10	11	12
	18	19	20	21	15	16	17		15	16	17	18	19	20	21		20	21	15	16	17	18	19
	25	26	27	28	22	23	24		22	23	24	25	26	27	28		27	28	22	23	24	25	26

Menu

In the upper right corner there are buttons for quick access and other actions.



Quick access:

- Today: Jumps to the current date.
- View: Switches between the month and year view.

Other actions:

- Show/Hide: Shows or hides exceptions.
- Add: Adds local exceptions.
- Reset: Discards all unconfirmed changes.

Local exceptions

Entries in the aException collection of a Schedule object (e.g. `FB_BA_SchedM [159]`) are regarded as local exceptions.

In the upper area, you can set the date or the repetition type of the local exception. Below this, the time periods with the applicable values can be set.

The automatic numbering of the local exceptions starts at 1.

Global exceptions (calendar reference)

Entries in the aCalendar collection of a Schedule object (e.g. [FB_BA_SchedM \[► 159\]](#)) are regarded as global exceptions.

For global exceptions, only the time periods and values to be applied can be defined. The date or the repetition type must be configured in the referenced Calendar object (e.g. [FB_BA_Cal \[► 151\]](#)).

The automatic numbering of the global exceptions starts at 100.

Attributes

The control inherits from [BaseControl \[► 923\]](#) and thus has the same attributes. In addition, there are the following attributes.

BA

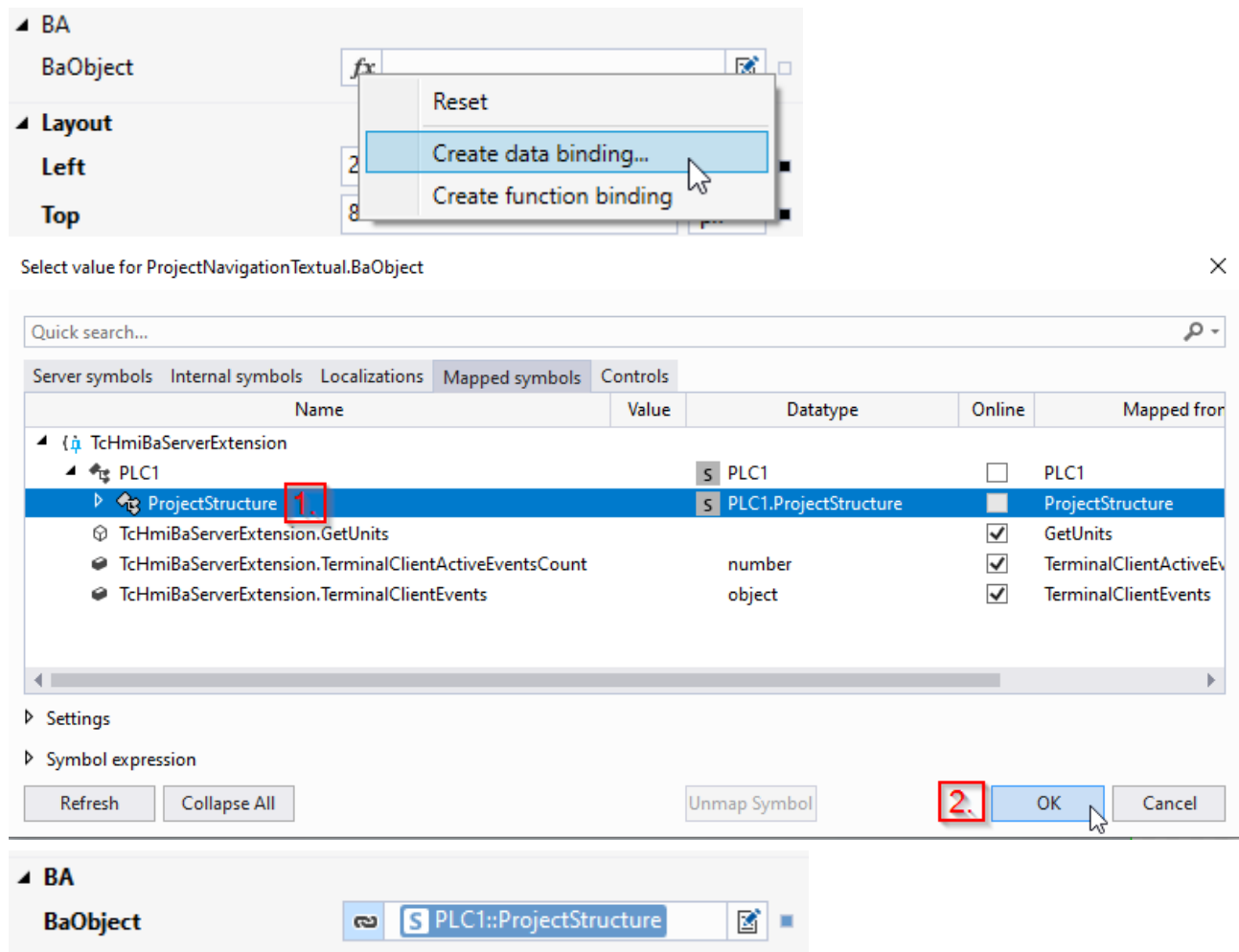
BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here \[► 63\]](#). The attribute is not applicable to all controls.



Common

DisplayMode

Defines the display mode of the calendar.

DisplayView

Sets the view of the calendar in the EventCalendar display mode.

ShowMenu

Sets the visibility of the menu.

Events

Event	Description
onDateChanged	Returns the selected date.

6.2.1.1.2.1.1 Enumerations

DisplayMode

Schema for the attribute `DisplayMode` [► 838] of the `Calendar` [► 834] control.

```
"TcHmi.BuildingAutomation.Controls.Common.Calendar.CalendarDisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [ 0, 1 ],
```

```

    "options": [
      {
        "label": "DatePicker",
        "value": 0
      },
      {
        "label": "EventCalendar",
        "value": 1
      }
    ]
  }
}

```

DisplayView

Schema for the attribute [DisplayView](#) [► 838] of the [Calendar](#) [► 834] control.

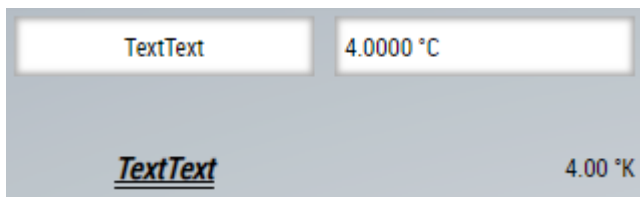
```

"TcHmi.BuildingAutomation.Controls.Common.Calendar.CalendarDisplayView": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [ 0, 1 ],
  "options": [
    {
      "label": "Month",
      "value": 0
    },
    {
      "label": "Year",
      "value": 1
    }
  ]
}

```

6.2.1.1.2.2 InputBox

The **InputBox** is used to display and edit numerical or textual values.



Use

Use on any page where numerical or textual values are to be displayed or edited.

Features

Numerical input

If the `DataType` is equal to number, the user input is checked for the following criteria:

- purely numerical input (letters and special characters are not allowed)
- minimum value (if `MinValue` is set)
- maximum value (if `MaxValue` is set)

The unit and number of decimal places for a numerical value can also be specified with the attributes `Unit` and `Digits`.

Attributes

The control inherits from [TextControl](#) [► 935] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

BaObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

The screenshot illustrates the configuration of the **BaObject** property in the TcHmiBa control. The left sidebar shows the control tree with **BA** expanded, and **BaObject** selected. A context menu is open over the **BaObject** property, with **Create data binding...** highlighted. Below this, a dialog box titled "Select value for ProjectNavigationTextual.BaObject" is shown. The dialog has tabs for "Server symbols", "Internal symbols", "Localizations", "Mapped symbols", and "Controls". The "Mapped symbols" tab is active, displaying a table of mapped symbols. The **ProjectStructure** symbol is selected, indicated by a red box and the number "1.". At the bottom of the dialog, the **OK** button is highlighted with a red box and the number "2.". The final state shows the **BaObject** property in the control tree set to **PLC1::ProjectStructure**.

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Common

Data Type

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.InputBox.InputDataType

Data type of the InputBox. If *auto* is selected, the default value or the first input is analyzed and the data type is set accordingly (see [InputDataType](#) [► 842]). If *number* is selected *Value* does **not** contain the unit.

BaData

BaInterface

tchmi:framework#/definitions/Symbol

Allows linking a symbol (see [symbol](#)) that satisfies the [BaInterface](#) [► 999] of the control. With this symbol all necessary data points of the control can be linked with only one binding.

BalInterfaceSymbolNames

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.InputBox.BaInterfaceSymbolNames

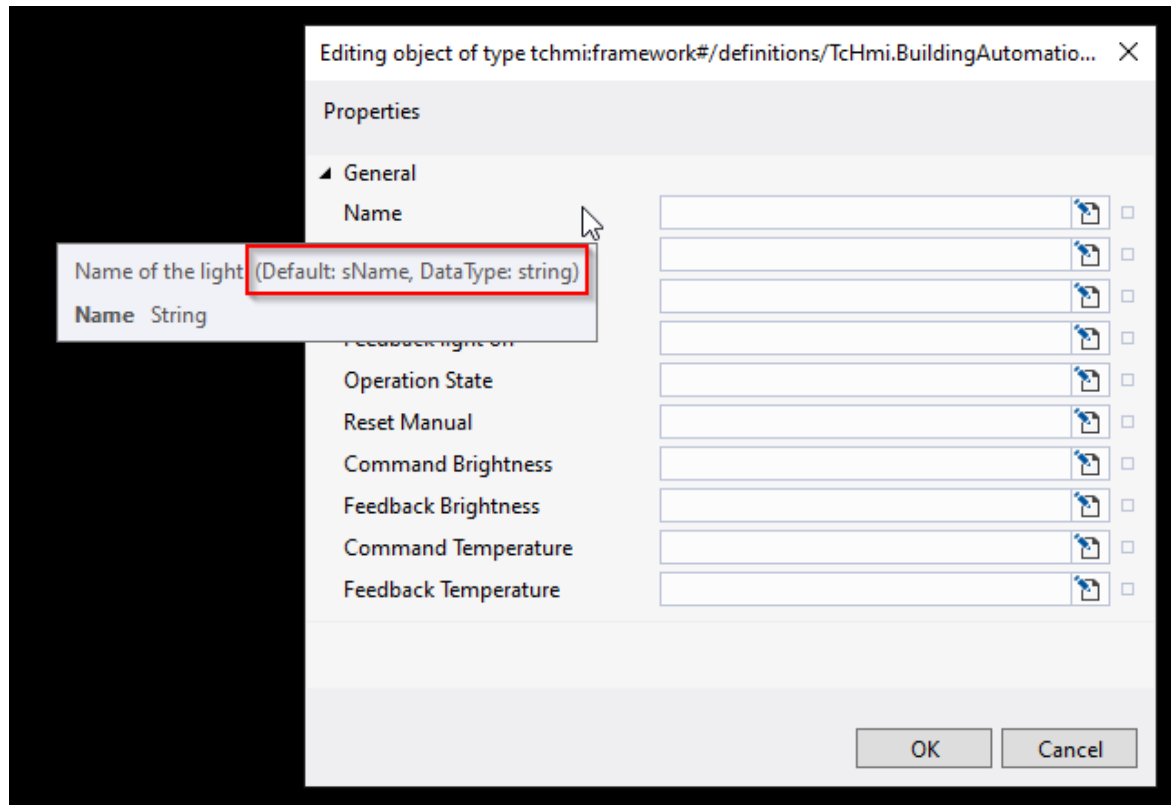
(see [BalInterfaceSymbolNames](#) [► 843])

tchmi:general#/definitions/Object

Allows editing the [BalInterfaceSymbolNames](#) [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the [BalInterfaceSymbolNames](#) are overwritten by all controls of a type.

Value

tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber

Current Value. Depending on the selected DataType the value is numerical or textual (see [StringOrNumber](#) [► 843]).

ValueFeedback

tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber

Feedback for the value (see [StringOrNumber](#) [► 843]).

UseValueFeedback

tchmi:general#/definitions/Boolean

If TRUE, then attention is paid to the ValueFeedback attribute (see [Boolean](#)).

Number

MinValue

tchmi:general#/definitions/Number

Lowest permissible input value (if DataType is equal to *number*) (see [Number](#)).

MaxValue

tchmi:general#/definitions/Number

Largest allowed input value (if DataType is equal to *number*) (see [Number](#)).

Unit

tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber

Determines the unit after *Value* (if DataType is equal to *number*) (see [StringOrNumber](#) [► 843]). Possible values:

- textual (e.g. "°C")
- numerical (enumeration value of E_BA_Unit)

Digits

tchmi:general#/definitions/Number

Number of decimal places (if DataType is equal to *number*) (see [Number](#)).

Events

Event	Description
onStateChanged	Triggered when the value has changed.
onUserInteractionFinished	Triggered when the user exits the input. This means: • Enter key is pressed • InputBox loses focus

6.2.1.1.2.2.1 Enumerations

InputDataType

Schema for the attribute [DataType](#) [► 840] of the [InputBox](#) [► 839] control.

```
"TcHmi.BuildingAutomation.Controls.InputBox.InputDataType": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "auto",
      "value": 0
    },
    {
      "label": "string",
      "value": 1
    },
    {
      "label": "number",
      "value": 2
    }
  ]
}
```

6.2.1.1.2.2.2 Types

StringOrNumber

Schema for the attribute [Value \[► 841\]](#) of the [InputBox \[► 839\]](#) control.

```
"TcHmi.BuildingAutomation.StringOrNumber": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "oneOf": [
    {
      "title": "String",
      "type": "string"
    },
    {
      "title": "Number",
      "type": "number"
    }
  ]
}
```

BaInterfaceSymbolNames

Schema for the attribute [BaInterfaceSymbolNames \[► 841\]](#) of the [InputBox \[► 839\]](#) control.

```
"TcHmi.BuildingAutomation.Controls.InputBox.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "value",
      "displayName": "Value",
      "description": "The symbol to set the value. (Default: Value, DataType: number, string)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "valueFeedback",
      "displayName": "Value feedback",
      "description": "The feedback for the value. (Default: ValueFeedback, DataType: number, string)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "unit",
      "displayName": "Unit",
      "description": "Unit for an analog value. (Default: Unit, DataType: string)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "minVal",
      "displayName": "Minimum value",
      "description": "Minimum value for an analog value. (Default: MinVal, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "maxVal",
      "displayName": "Maximum value",
      "description": "Maximum value for an analog value. (Default: MaxVal, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "value": {
      "type": "string"
    },
    "valueFeedback": {
      "type": "string"
    },
    "unit": {
      "type": "string"
    },
    "minVal": {
      "type": "string"
    }
  }
}
```



```

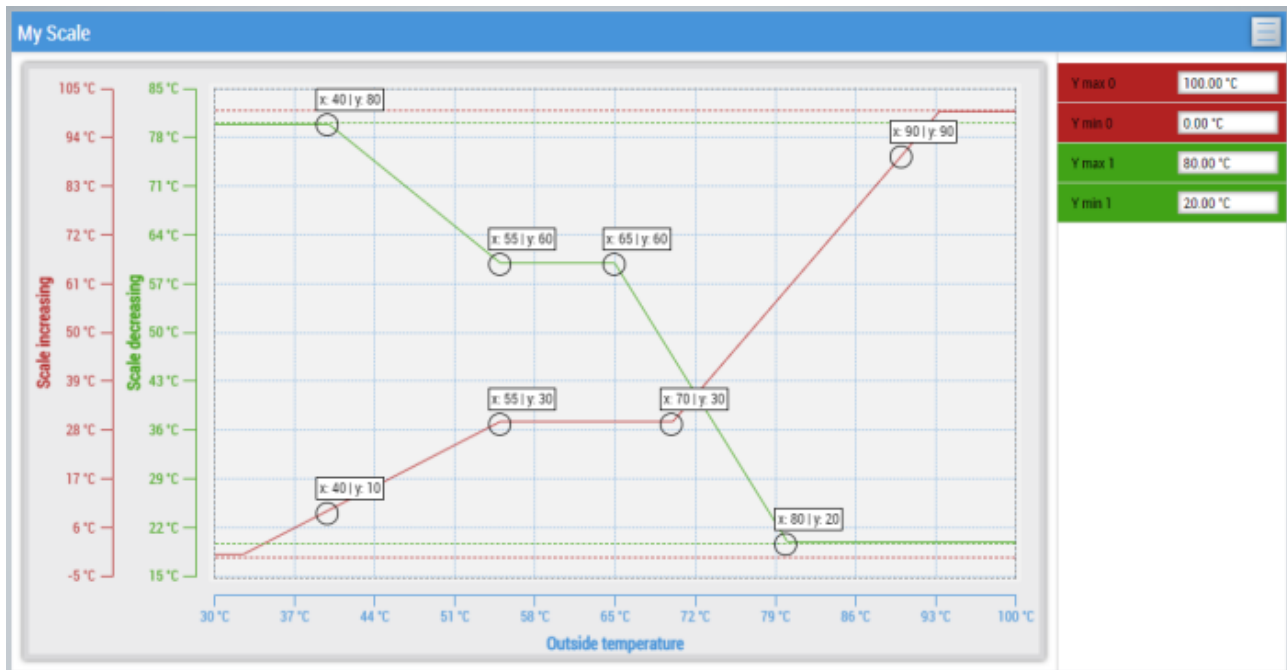
    },
    "maxVal": {
      "type": "string"
    }
  }
}

```

6.2.1.1.2.3 Scale

With the **Scale** control different types of scales can be displayed and edited.

The focus of the control is the visualization of a heating curve with a defined number of interpolation points.



Use

Can be used on any page where scales are to be displayed or edited.

Features

A scale can be edited by drag and drop and the maximum and minimum value can be set.

Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

BA

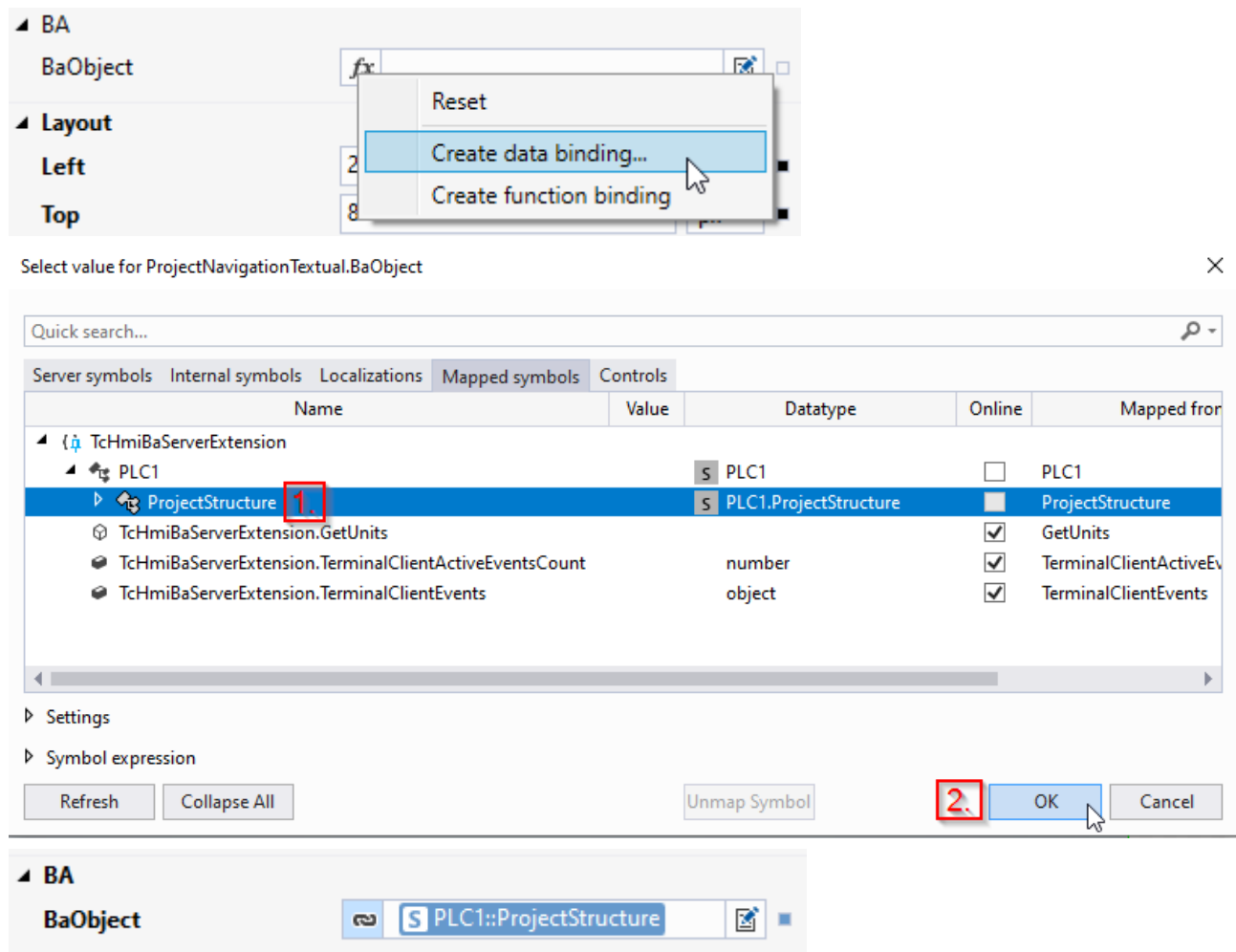
BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.



Common

Data

tchmi:framework#/definitions/TchHmi.BuildingAutomation.Controls.Scale.Scales

Determines the data for the different scales (see [Scales](#) [► 846]).

XAxisExtension

tchmi:general#/definitions/Number

Determines how much longer the X-axis is displayed, depending on XMin and XMax (see [number](#)).

Title

tchmi:general#/definitions/String

Determines the title that will be displayed in the header of the control (see [string](#)).

ShowHeader

tchmi:general#/definitions/Boolean

Determines whether the header of the control is displayed or not (see [Boolean](#)).

6.2.1.1.2.3.1 Enumerations

ScaleEnding

Schema for the Ending property of the [Data](#) [► 845] attribute of the [Scale](#) [► 844] control.

```
"TcHmi.BuildingAutomation.Controls.Scale.ScaleEnding": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1,
    2,
    3
  ],
  "options": [
    {
      "label": "direct",
      "value": 0
    },
    {
      "label": "infinity",
      "value": 1
    },
    {
      "label": "limits",
      "value": 2
    },
    {
      "label": "cutOff",
      "value": 3
    }
  ]
}
```

ScaleBehavior

Schema for the Behaviour property of the [Data \[► 845\]](#) attribute of the [Scale \[► 844\]](#) control.

```
"TcHmi.BuildingAutomation.Controls.Scale.ScaleBehavior": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "any",
      "value": 0
    },
    {
      "label": "onlyDecreasing",
      "value": 1
    },
    {
      "label": "onlyIncreasing",
      "value": 2
    }
  ]
}
```

6.2.1.1.2.3.2 Types

Scales

Schema for the attribute [Data \[► 845\]](#) of the [Scale \[► 844\]](#) control.

```
"TcHmi.BuildingAutomation.Controls.Scale.Scales": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Scales",
  "type": "array",
  "items": {
    "type": "object",
    "title": "Scale",
    "description": "Defines a scale.",
    "engineeringColumns": [
      "Title"
    ],
    "propertiesMeta": [
      {
        "name": "ScaleSetpoints",

```

```

        "category": "Common",
        "displayName": "ScaleSetpoints",
        "displayPriority": 10,
        "description": "",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "YMax",
        "category": "Common",
        "displayName": "YMax",
        "displayPriority": 10,
        "description": "",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "YMin",
        "category": "Common",
        "displayName": "YMin",
        "displayPriority": 10,
        "description": "",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Title",
        "category": "Common",
        "displayName": "Title",
        "displayPriority": 10,
        "description": "Title of the scale -> used for the axis description.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Behavior",
        "category": "Common",
        "displayName": "Behavior",
        "displayPriority": 10,
        "description": "Behavior of the scale",
        "defaultValue": 0,
        "defaultValueInternal": 0
    },
    {
        "name": "Ending",
        "category": "Common",
        "displayName": "Ending",
        "displayPriority": 10,
        "description": "Ending of the scale",
        "defaultValue": 0,
        "defaultValueInternal": 0
    },
    {
        "name": "SnapPeriode",
        "category": "Common",
        "displayName": "SnapPeriode",
        "displayPriority": 10,
        "description": "The periode the Setpoints should snap to after moving.",
        "defaultValue": 1,
        "defaultValueInternal": 1
    },
    {
        "name": "AllowVerticalMovement",
        "category": "Common",
        "displayName": "AllowVerticalMovement",
        "displayPriority": 10,
        "description": "Defines if vertical movement of the Setpoints is allowed.",
        "defaultValue": true,
        "defaultValueInternal": true
    },
    {
        "name": "AllowHorizontalMovement",
        "category": "Common",
        "displayName": "AllowHorizontalMovement",
        "displayPriority": 10,
        "description": "Defines if horizontal movement of the Setpoints is allowed.",
        "defaultValue": true,
        "defaultValueInternal": true
    },
    {

```

```

        "name": "YAxisExtension",
        "category": "Common",
        "displayName": "YAxisExtension",
        "displayPriority": 10,
        "description": "Defines how much the y axis is extended from YMin and YMax.",
        "defaultValue": 0,
        "defaultValueInternal": 0
    },
    {
        "name": "ReadOnly",
        "category": "Common",
        "displayName": "ReadOnly",
        "displayPriority": 10,
        "description": "Defines if horizontal movement of the Setpoints is allowed.",
        "defaultValue": true,
        "defaultValueInternal": true
    },
    {
        "name": "Unit",
        "category": "Common",
        "displayName": "Unit",
        "displayPriority": 10,
        "description": "Defines the unit of the y axis.",
        "defaultValue": null,
        "defaultValueInternal": null
    }
],
"properties": {
    "ScaleSetpoints": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Scale.ScaleSetpoints"
    },
    "YMax": {
        "type": "number"
    },
    "YMin": {
        "type": "number"
    },
    "Title": {
        "type": "string"
    },
    "Behavior": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Scale.ScaleBehavior"
    },
    "Ending": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Scale.ScaleEnding"
    },
    "SnapPeriode": {
        "type": "number"
    },
    "AllowVerticalMovement": {
        "type": "boolean"
    },
    "AllowHorizontalMovement": {
        "type": "boolean"
    },
    "YAxisExtension": {
        "type": "number"
    },
    "ReadOnly": {
        "type": "boolean"
    },
    "Unit": {
        "type": "string"
    }
}
}
}

```

ScaleSetpoints

Schema for the ScaleSetpoints property of the [Data \[► 845\]](#) attribute of the [Scale \[► 844\]](#) control.

```

"TcHmi.BuildingAutomation.Controls.Scale.ScaleSetpoints": {
    "type": "array",
    "items": {
        "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.ICoordinateXY",
    }
}

```

```

    "title": "ScaleSetpoint",
    "description": "Defines a scale setpoint"
  }
}

```

6.2.1.1.2.4 Slider

The **slider** can be used to display and edit numerical values.

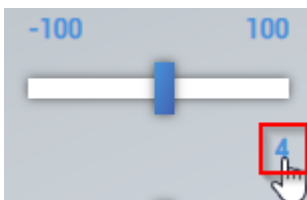


Use

Can be dragged to any page where numerical values are to be edited.

Features

The value can be set by drag and drop or by clicking on the slider. You can also click on the display that shows the current value and then enter the desired value.



It can be set whether the min. and max. values or the current value are displayed.

Different areas can be colored for the slider. Here you have the possibility to set color gradients or exact color areas.



If the [feedback concept](#) [► 849] is used, the value of the feedback is displayed with a slight shadow. Thus, for example, both values can be visualized at the same time for an object that has a target value and an actual value.



Attributes

The control inherits from [TextControl](#) [► 935] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

BAObject

```
tchmi:framework#/definitions/Symbol
```

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

The screenshot illustrates the process of creating a data binding for the `ProjectStructure` property of the `TcHmiBaServerExtension` object.

Top Panel: The `Layout` section is expanded, showing the `Left` and `Top` views. A context menu is open over the `ProjectStructure` property, with the `Create data binding...` option selected.

Bottom Panel: The `Symbol expression` field is shown with the value `S PLC1::ProjectStructure`. The `OK` button is highlighted with a red box and the number 2.

Symbol Selection Dialog: A dialog box titled "Select value for ProjectNavigationTextual.BaObject" is shown. It contains a search bar and a table of symbols.

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

The `ProjectStructure` symbol is highlighted with a red box and the number 1.

Common

ShowValue

tchmi:general#/definitions/Boolean

Determines whether the current value is displayed (see [Boolean](#)).

ShowScale

tchmi:general#/definitions/Boolean

Determines whether the `MinValue` [► 853] and `MaxValue` [► 853] are displayed (see [Boolean](#)).

Orientation

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Orientation

Determines the orientation of the slider (horizontal or vertical) (see [Orientation](#) [► 926]).

SwitchMinMax

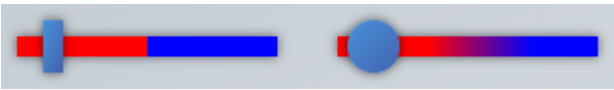
tchmi:general#/definitions/Boolean

If active, the positions of `MinValue` and `MaxValue` are swapped (see [Boolean](#)).

Ranges

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Slider.SliderRanges
```

Specifies different color areas or color gradients to be displayed in the slider (see [SliderRanges](#) [► 855]).



TcHmi_BuildingAutomation_Controls_Slider_2 | Ranges* ✕

Elements

Start	End
-100	0
0	100

✕ ↑ ↓ SliderRange Add

Properties

Colors

Color

#FFFF0000

R

255

G

0

B

0

A

255

#FFFF0000

Common

Start

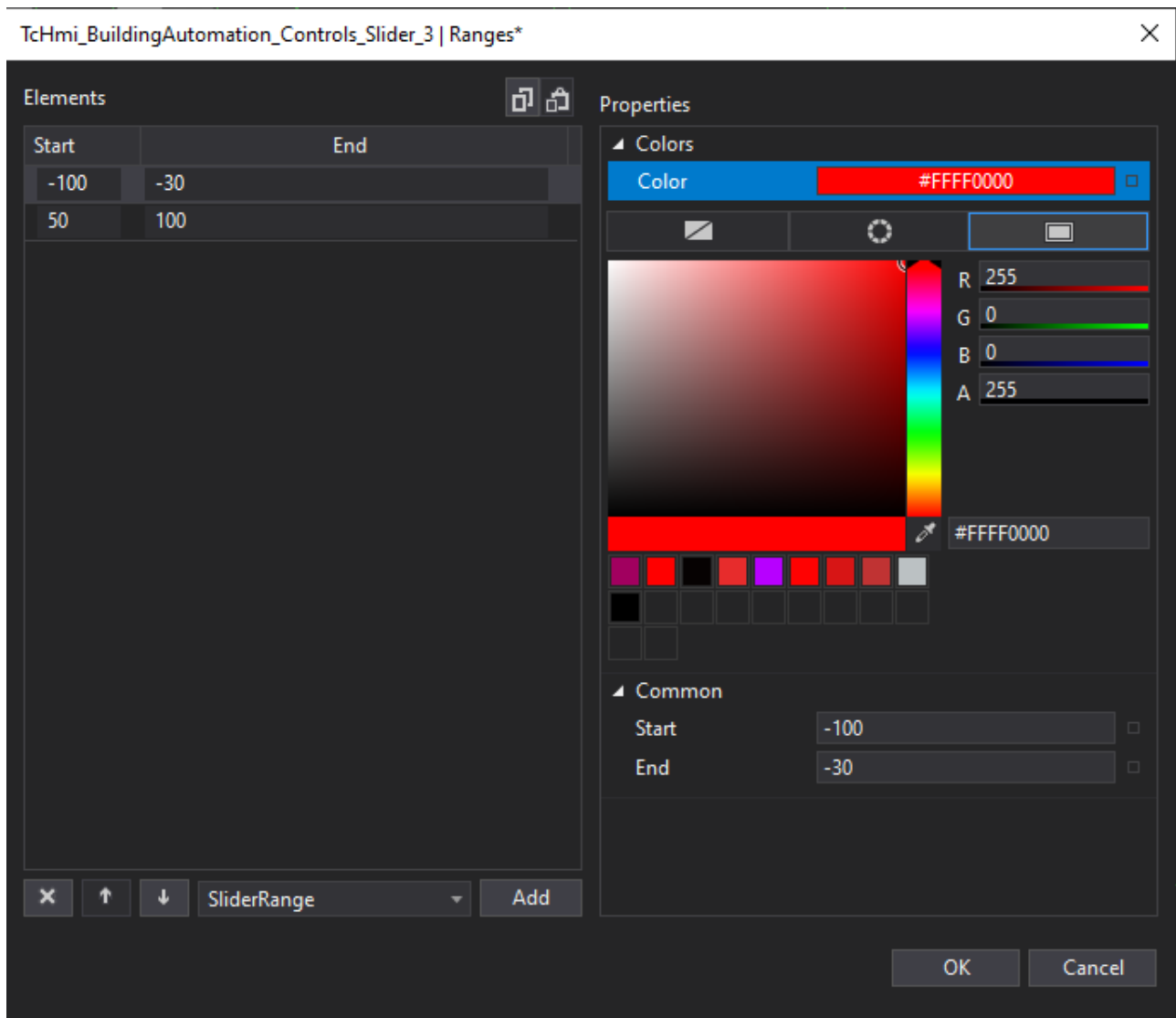
-100

End

0

OK

Cancel



KnobAppearance

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Slider.KnobAppearance

Determines the appearance of the slider's knob (see [KnobAppearance](#) [► 854]).

BaData

BaInterface

tchmi:framework#/definitions/Symbol

Allows linking a symbol (see [symbol](#)) that satisfies the [BaInterface](#) [► 999] of the control. With this symbol all necessary data points of the control can be linked with only one binding.

BaInterfaceSymbolNames

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Slider.BaInterfaceSymbolNames

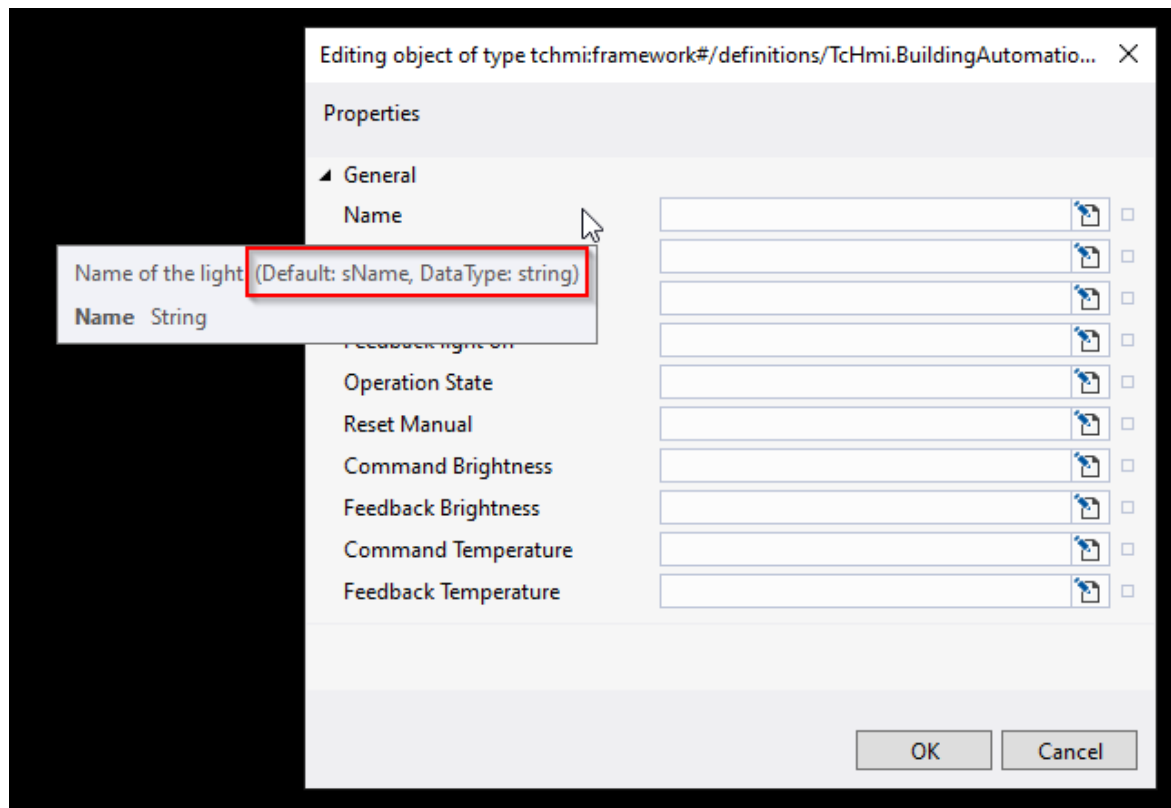
(see [BaInterfaceSymbolNames](#) [► 854])

tchmi:general#/definitions/Object

Allows editing the [BaInterfaceSymbolNames](#) [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the **BalInterfaceSymbolNames** are overwritten by all controls of a type.

Value

`tchmi:general#/definitions/Number`

The current value of the slider (see [Number](#)).

ValueFeedback

`tchmi:general#/definitions/Number`

The feedback for the value of the slider (see [Number](#)).

Number

Unit

`tchmi:general#/definitions/String`

Defines the unit that is displayed after [Value](#) [▶ 853] (see [String](#)).

MinValue

`tchmi:general#/definitions/Number`

The minimum value of the slider (see [Number](#)).

MaxValue

`tchmi:general#/definitions/Number`

The maximum value of the slider (see [Number](#)).

Step

tchmi:general#/definitions/Number

Determines the accuracy with which the value can be set with the slider (e.g. 0.01) (see [Number](#)).

Events

Event	Description
onUserInteractionFinished	The event is triggered when the value change has been completed by the user. This happens with drag and drop, when the user releases the slider again or after clicking on an area of the slider.
OnValueChanged	The event is triggered every time the value of the slider changes, for example, when the slider is moved.

6.2.1.1.2.4.1 Enumerations**KnobAppearance**

Schema for the attribute [KnobAppearance](#) [► 852] of the [slider](#) [► 849].

```
"TchHmi.BuildingAutomation.Controls.Slider.KnobAppearance": {
  "type": "string",
  "enum": [
    0,
    1
  ],
  "options": [
    {
      "label": "rectangle",
      "value": 0
    },
    {
      "label": "round",
      "value": 1
    }
  ]
}
```

6.2.1.1.2.4.2 Types**BaInterfaceSymbolNames**

Schema for the attribute [BaInterfaceSymbolNames](#) [► 852] of the [Slider](#) [► 849] control.

```
"TchHmi.BuildingAutomation.Controls.Slider.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "value",
      "displayName": "Value",
      "description": "The symbol to set the value. (Default: Value, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "valueFeedback",
      "displayName": "Value feedback",
      "description": "The feedback for the value. (Default: ValueFeedback, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "unit",
      "displayName": "Unit",
      "description": "Unit for an analog value. (Default: Unit, DataType: string)",
      "defaultValue": null,

```

```

    "defaultValueInternal": null
  },
  {
    "name": "minVal",
    "displayName": "Minimum value",
    "description": "Minimum value for an analog value. (Default: MinVal, DataType: number)",
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "maxVal",
    "displayName": "Maximum value",
    "description": "Maximum value for an analog value. (Default: MaxVal, DataType: number)",
    "defaultValue": null,
    "defaultValueInternal": null
  }
],
"properties": {
  "value": {
    "type": "string"
  },
  "valueFeedback": {
    "type": "string"
  },
  "unit": {
    "type": "string"
  },
  "minVal": {
    "type": "string"
  },
  "maxVal": {
    "type": "string"
  }
}
}
}

```

SliderRanges

Schema for the attribute [Ranges](#) [► 851] of the [Slider](#) [► 849] control.

```

"TcHmi.BuildingAutomation.Controls.Slider.SliderRanges": {
  "type": "array",
  "title": "SliderRanges",
  "items": {
    "type": "object",
    "title": "SliderRange",
    "description": "Defines a slider range",
    "engineeringColumns": [
      "start",
      "end"
    ],
  },
  "propertiesMeta": [
    {
      "name": "color",
      "category": "Color",
      "displayName": "Color"
    },
    {
      "name": "start",
      "category": "Common",
      "displayName": "Start ",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "end",
      "category": "Common",
      "displayName": "End",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "color": {
      "$ref": "tchmi:framework#/definitions/SolidColor"
    },
    "start": {
      "type": "number"
    },
  },
}

```

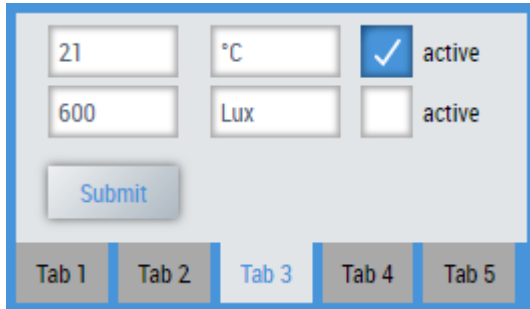
```

    "end": {
      "type": "number"
    },
    "required": [ "start", "color" ]
  }
}

```

6.2.1.1.2.5 TabWindow

The **TabWindow** is used to display different content in different tabs.



Use

Use on any page where a TabWindow is to be displayed.

Features

Pages of type *.content or programmatically created HTML can be assigned to the tabs via the Data attribute.

Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

Common

Data

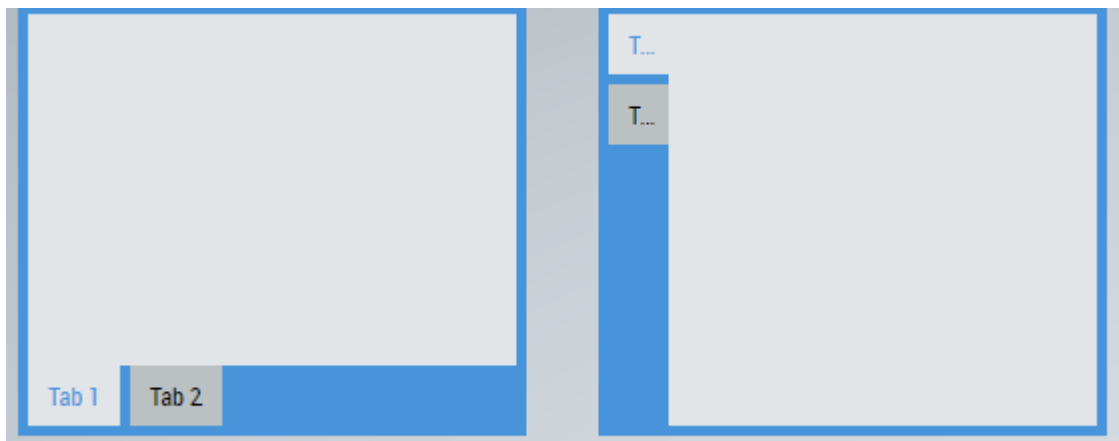
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TabWindow.TabWindowData

Contents for the different tabs (see [TabWindowData](#) [► 857]).

TabPosition

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position

Position of the tabs (see [Position](#) [► 925]).

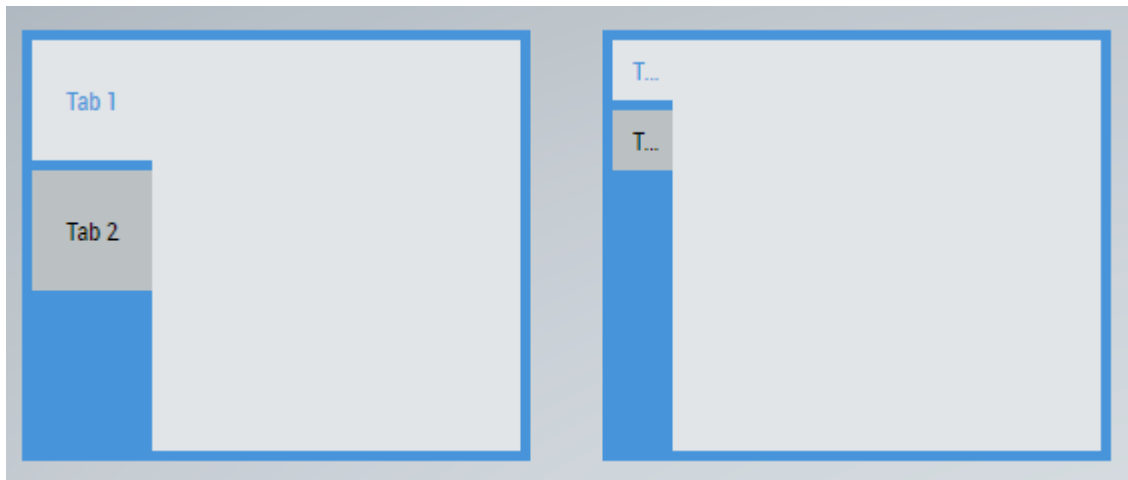


TabDistance

Tchmi:framework#/definitions/Number

Distance between tabs (see [Number](#)).**TabContainerDistance**

tchmi:general#/definitions/Number

The size of the tab container (see [Number](#)).**TabSizeAuto**

tchmi:general#/definitions/Boolean

Defines whether the tabs should occupy the complete width of the tab container or not (see [Boolean](#)).**6.2.1.1.2.5.1 Types****TabWindowData**Schema for the attribute [Data](#) [► 856] of the [TabWindow](#) [► 856] control.

```

"Tchmi.BuildingAutomation.Controls.TabWindowData": {
  "type": "array",
  "items": {
    "type": "object",
    "title": "TabData",
    "engineeringColumns": [
      "ID",
      "Text"
    ],
  },
  "propertiesMeta": [
    {
      "name": "ID",
      "displayName": "ID",
      "description": "The unique ID of the tab.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "Text",

```

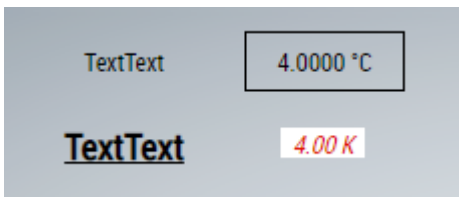
```

        "displayName": "Text",
        "description": "The text that will be displayed in the tab.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "Content",
        "displayName": "Content",
        "description": "The content that will be displayed when the tab was selected.",
        "defaultValue": null,
        "defaultValueInternal": null
    }
],
"properties": {
    "ID": {
        "type": "string"
    },
    "Text": {
        "type": "string"
    },
    "Content": {
        "$ref": "tchmi:framework#/definitions/ContentPath"
    }
},
"required": [ "ID", "Text", "Content" ]
}
}

```

6.2.1.1.2.6 Text block

The control **Textblock** is used to display text.



Use

Use on any page where text is to be displayed.

Features

A specialized text block with reduced resource requirements.

Attributes

The control inherits from [TextControl](#) [► 935] and thus has the same attributes. In addition, there are the following attributes.

BA

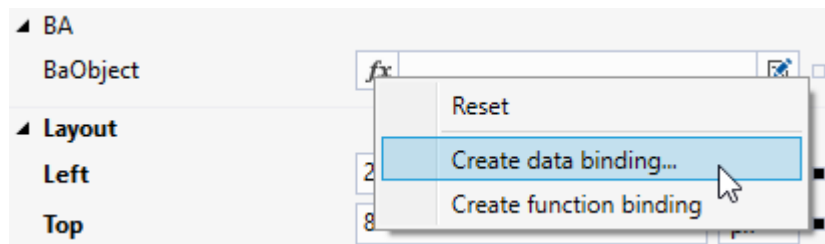
BAObject

tchmi:framework#/definitions/Symbol

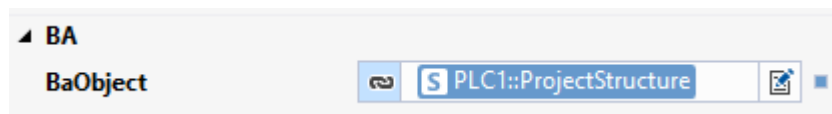
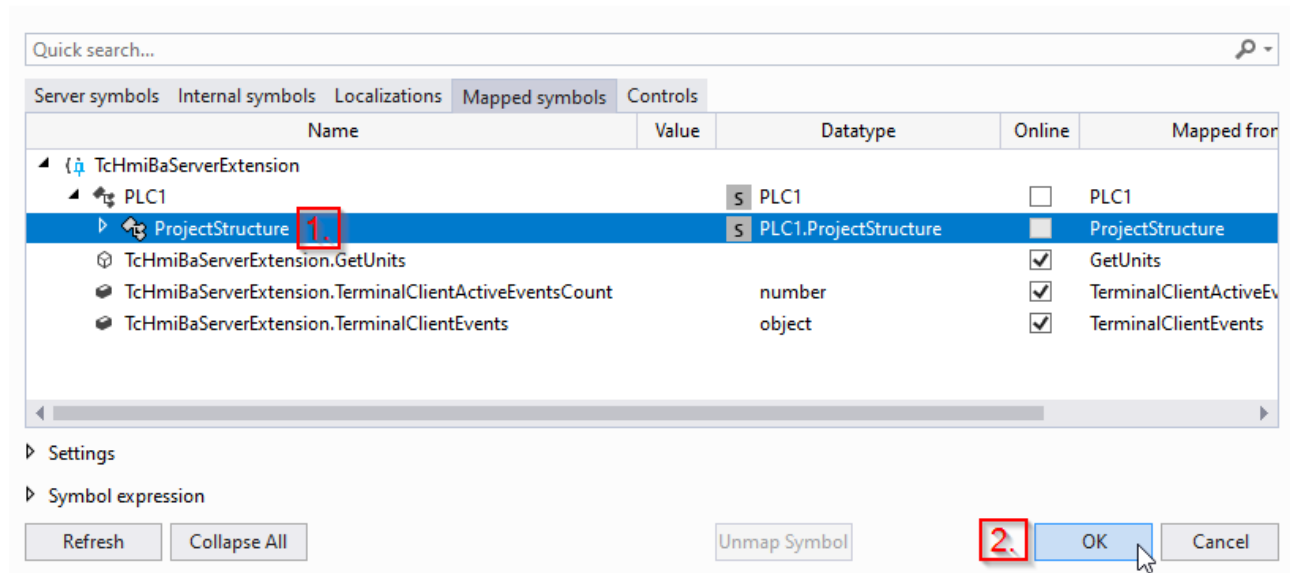
To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.



Select value for ProjectNavigationTextual.BaObject



Common

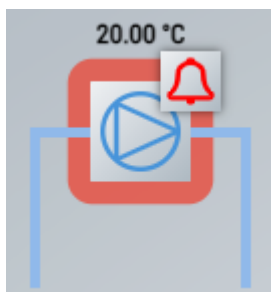
Text

tchmi:general#/definitions/String

Text for the text block (see [String](#)).

6.2.1.1.2.7 Uilcon

The **Uilcon** control can be used to display events and values. It looks like a normal [button](#) [▶ 884] and can be filled with different icons.



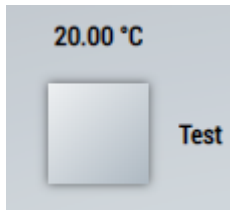
Use

Suitable for creating P&I diagrams to represent various plant components (e.g. pump). The attribute Connections can be used to create suitable connections to connect the Uilcon with a main line, for example.

Features

Value displays

Various displays can be added to the **Uilcon** via the attribute DisplaysData.

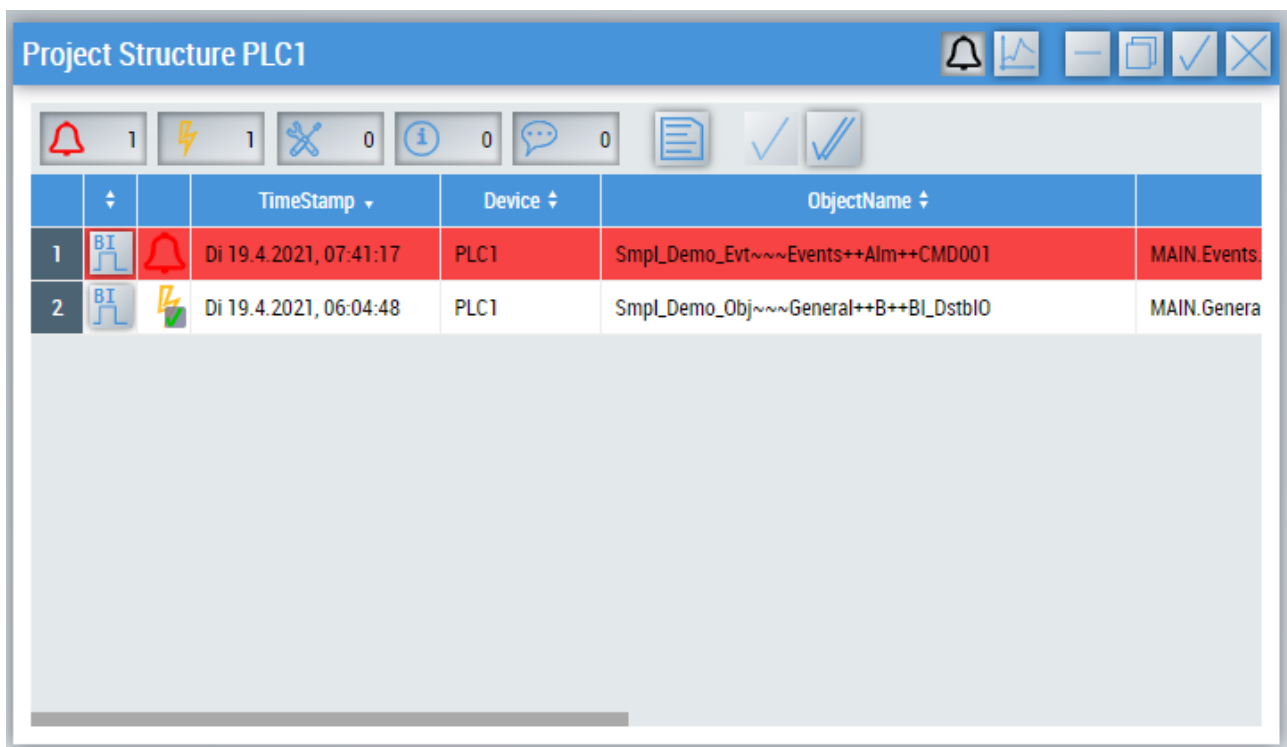


Event displays

The attribute EventsData can be used to display various events around the **Uilcon**.



If the generic approach [► 63] of TcHmiBa is used and a BaObject / BaView is linked to the control, active events are displayed automatically. When the **Uilcon** is actuated, the project navigation [► 889] of the linked object opens and, in the case of an event, the parameter window with the event view opens accordingly.



Attributes

The control inherits from the button [► 884] and thus has the same attributes. In addition, there are the following attributes.

BA

BAObject

```
tchmi:framework#/definitions/Symbol
```

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

The screenshot illustrates the process of creating a data binding for the `BaObject` property. A context menu is shown with the option **Create data binding...** selected. Below, the 'Select value for ProjectNavigationTextual.BaObject' dialog is displayed, showing a list of symbols under the 'Mapped symbols' tab. The `ProjectStructure` symbol is highlighted with a red box labeled '1.'. At the bottom, the 'OK' button is highlighted with a red box labeled '2.'. Finally, the `BaObject` property in the configuration window is shown with the value `S PLC1::ProjectStructure`.

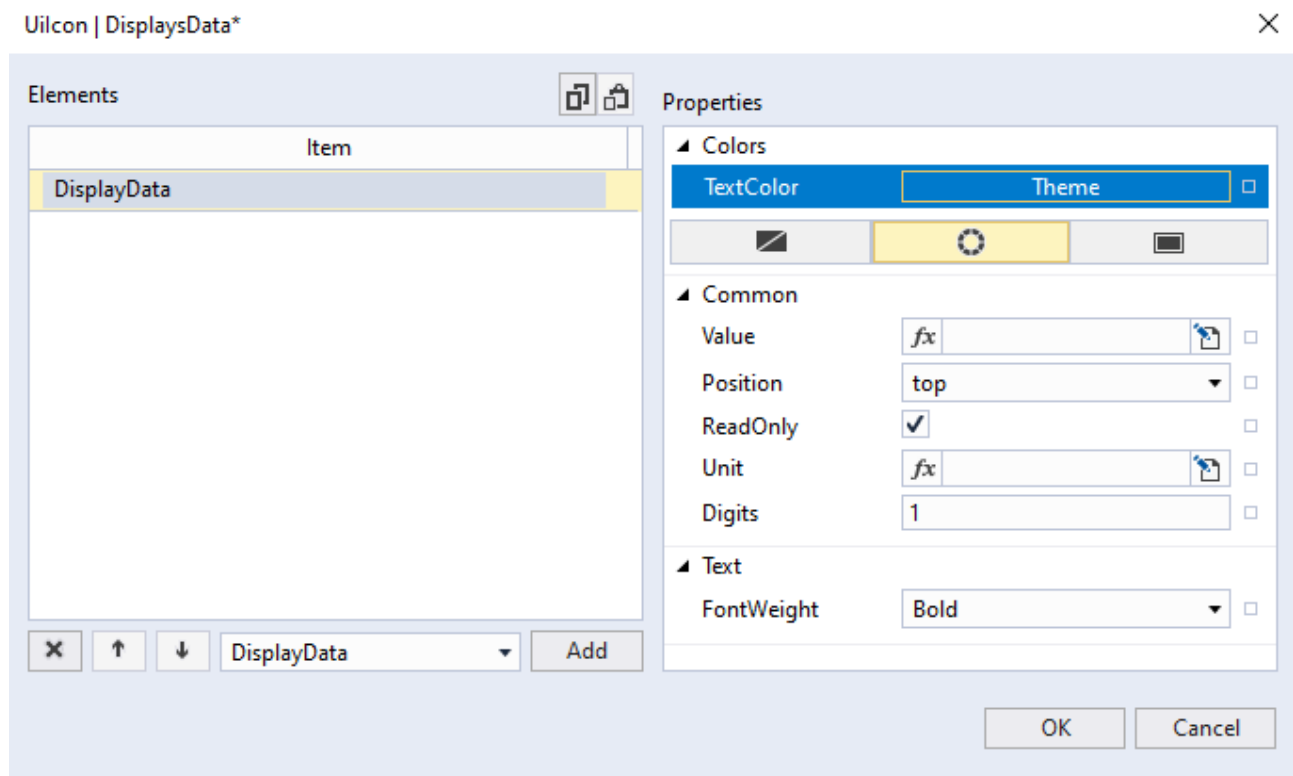
Name	Value	Datatype	Online	Mapped from
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Common

DisplaysData

TcHmi.BuildingAutomation.Controls.UiIcon.DisplaysData

The attribute allows to create different displays via an editor (see [DisplaysData](#) [► 865]).



The following properties can be set for each display:

Name	Description
TextColor	Font color of the display.
Value	Display value in the display. If a binding exists and ReadOnly is disabled, the value is written to this binding when the user ends the input.
Position	Position of the display. Several displays created at the same position are arranged on top of each other.
ReadOnly	Determines whether the display is editable or read-only.
Unit	Unit to be appended to the value (if it is a number).
Digits	Number of decimal places.
FontWeight	Font weight of the text.

EventsData

TcHmi.BuildingAutomation.Controls.UiIcon.EventsData

The attribute allows to create different events via an editor (see [EventsData](#) [► 866]).

Uilcon | EventsData*

Elements

Item
EventData

Properties

Common

Activate ☒

Event condition eTypeAlarm

Event state ePresentAcked

Event count 2

Other

Acknowledge symbol fx

Is Acknowledgable ☐

Icon

Enable pulse ☐

OK Cancel

The following properties can be set for each event:

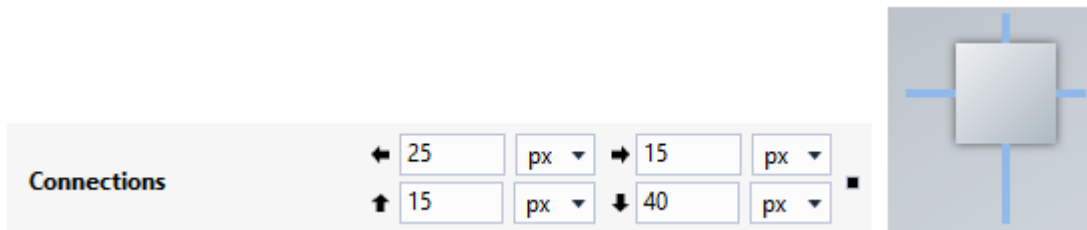
Name	Description
Activate	Determines whether the event is active or not.
Event condition	Determines the type (priority) of the event. The icons are arranged according to their priority in a clockwise direction. Top right is the highest priority.
Event state	Current state of the event.
Event count	Determines how many events of this type and state are active.
Acknowledge symbol	Writes TRUE to the symbol when the event is pressed.
Is Acknowledgable	Determines whether the event can be pressed.
Icon	Icon to use if no <i>event condition</i> is selected to allow user specific icons.
Enable pulse	Evaluation is done only if <i>Event condition</i> and <i>Event state</i> are not used. When activated, a red pulse is displayed around the Uilcon .

Connections

Connections can be used to represent connections to other lines in an P&I diagram.

tchmi:framework#/definitions/Padding

Connections can be created here that go vertically or horizontally away from the **Uilcon** (see [Padding](#)).



The length of the connection must be specified in each case.

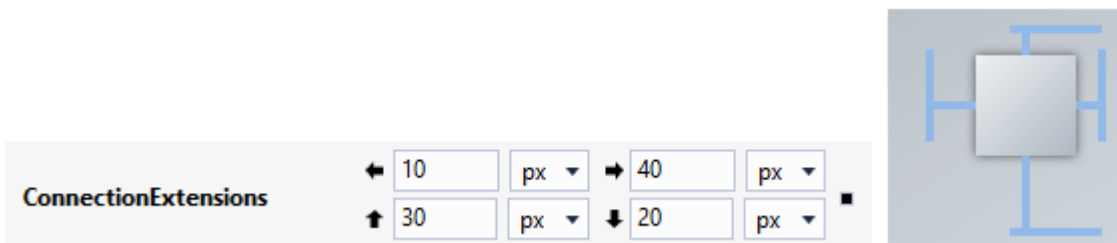


The unit pixel is always used. Percent is **not** supported at this point.

ConnectionExtensions

tchmi:framework#/definitions/Padding

Extensions for the connections created above can be created here (see [Padding](#)).



The length of the extension must be specified in each case.



The unit pixel is always used. Percent is **not** supported at this point.

ConnectionsWidth

tchmi:framework#/definitions/PositiveNumber

Specification of the width in pixels for the connections (see [PositiveNumber](#)).

ConnectionsColor

tchmi:framework#/definitions/SolidColor

Specification of the color for the connections (see [SolidColor](#)).

ConnectionsColorPerSide

tchmi:framework#/definitions/TcHmi.BuildingAutomation.FourSidedColor

Determines the color for different connections (see [FourSidedColor](#) [► 926]). The attribute `ConnectionsColor` must be set to `NULL` or `NONE`.

6.2.1.1.2.7.1 Types

DisplayPosition

Schema for the Position property of the [DisplaysData](#) [► 865] attribute from [UiIcon](#) [► 859] control.

```
"TcHmi.BuildingAutomation.Controls.UiIcon.DisplayPosition": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "DisplayPosition",
  "type": "string",
  "enum": [
```

```

0,
1,
2,
3,
4,
5,
6,
7
],
"options": [
  {
    "label": "left",
    "value": 0
  },
  {
    "label": "top",
    "value": 1
  },
  {
    "label": "right",
    "value": 2
  },
  {
    "label": "bottom",
    "value": 3
  },
  {
    "label": "topLeft",
    "value": 4
  },
  {
    "label": "topRight",
    "value": 5
  },
  {
    "label": "bottomRight",
    "value": 6
  },
  {
    "label": "bottomLeft",
    "value": 7
  }
]
}

```

DisplaysData

Schema for the attribute `DisplaysData` [► 861] of the `UiIcon` [► 859] control.

```

"TcHmi.BuildingAutomation.Controls.UiIcon.DisplaysData": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "array",
  "items": {
    "type": "object",
    "title": "DisplayData",
    "description": "Defines a display for the object.",
    "propertiesMeta": [
      {
        "name": "value",
        "category": "Common",
        "displayName": "Value",
        "description": "Value that is displayed in the display.",
        "displayPriority": 10,
        "defaultValue": null,
        "defaultValueInternal": null
      },
      {
        "name": "position",
        "category": "Common",
        "displayName": "Position",
        "description": "Position of the display.",
        "displayPriority": 10,
        "defaultValue": 1,
        "defaultValueInternal": 1
      },
      {
        "name": "readOnly",
        "category": "Common",
        "displayName": "ReadOnly",

```

```

        "description": "Defines if the display is editable or read only.",
        "displayPriority": 10,
        "defaultValue": true,
        "defaultValueInternal": true
    },
    {
        "name": "unit",
        "category": "Common",
        "displayName": "Unit",
        "description": "The unit will be added to the displayed value if the type of Value is
'number'.",
        "displayPriority": 10,
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "digits",
        "category": "Common",
        "displayName": "Digits",
        "description": "Number of displayed digits.",
        "displayPriority": 10,
        "defaultValue": 1,
        "defaultValueInternal": 1
    },
    {
        "name": "textColor",
        "category": "Colors",
        "displayName": "TextColor",
        "description": "Color of the displayed text.",
        "displayPriority": 10,
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "fontWeight",
        "category": "Text",
        "displayName": "FontWeight",
        "description": "Font weight of the displayed text.",
        "displayPriority": 10,
        "defaultValue": "Bold",
        "defaultValueInternal": "Bold"
    }
],
"properties": {
    "value": {
        "type": "string"
    },
    "position": {
        "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.PositionEx"
    },
    "readOnly": {
        "type": "boolean"
    },
    "unit": {
        "type": "string"
    },
    "digits": {
        "type": "number"
    },
    "textColor": {
        "$ref": "tchmi:framework#/definitions/SolidColor"
    },
    "fontWeight": {
        "$ref": "tchmi:framework#/definitions/FontWeight"
    }
},
"required": [ "value", "position" ]
}
}

```

EventsData

Schema for the attribute [EventsData \[► 862\]](#) of the [UiIcon \[► 859\]](#) control.

```

"Tchmi.BuildingAutomation.Controls.UiIcon.EventsData": {
    "$schema": "http://json-schema.org/draft-04/schema",
    "type": "array",
    "items": {
        "type": "object",

```

```

"title": "EventData",
"description": "Defines an event for the object.",
"propertiesMeta": [
  {
    "name": "activate",
    "category": "Common",
    "displayName": "Activate",
    "description": "If true the event will be displayed on the UiIcon.",
    "defaultValue": false,
    "defaultValueInternal": false
  },
  {
    "name": "eventCondition",
    "category": "Common",
    "displayName": "Event condition",
    "description": "BA event condition. If this is used the icon of the event will be selected automatically.",
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "eventIconState",
    "category": "Common",
    "displayName": "Event state",
    "description": "BA event icon state. If this is used the icon state of the event will be selected automatically.",
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "eventCount",
    "category": "Common",
    "displayName": "Event count",
    "description": "If larger than 0 a small number will be displayed on the event. This indicates the number of active events with the same icon.",
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "acknowledgeSymbol",
    "category": "Other",
    "displayName": "Acknowledge symbol",
    "description": "Boolean symbol that should be set to true when event icon was pressed.",
    "displayPriority": 200,
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "isAcknowledgeable",
    "category": "Other",
    "displayName": "Is Acknowledgeable",
    "description": "If true the event can be clicked, when false it is not enabled.",
    "displayPriority": 200,
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "icon",
    "category": "Other",
    "displayName": "Icon",
    "description": "Icon path for the displayed icon if 'EventCondition' and 'EventIconState' is not set.",
    "displayPriority": 200,
    "defaultValue": null,
    "defaultValueInternal": null
  },
  {
    "name": "enablePulse",
    "category": "Other",
    "displayName": "Enable pulse",
    "description": "Activates or deactivated the pulse if 'EventCondition' and 'EventIconState' is not set.",
    "displayPriority": 200,
    "defaultValue": null,
    "defaultValueInternal": null
  }
],
"properties": {
  "activate": {
    "type": "boolean"
  }
}

```



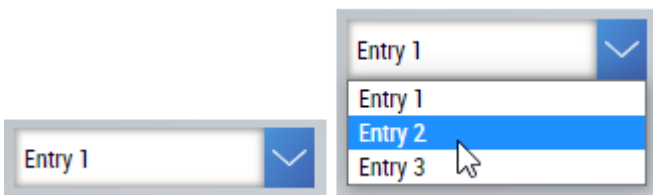
```

    },
    "eventCondition": {
      "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.EventCondition"
    },
    "eventIconState": {
      "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.EventIconState"
    },
    "eventCount": {
      "type": "number"
    },
    "acknowledgeSymbol": {
      "$ref": "tchmi:framework#/definitions/Symbol"
    },
    "isAcknowledgeable": {
      "type": "boolean"
    },
    "icon": {
      "$ref": "tchmi:framework#/definitions/Path"
    },
    "enablePulse": {
      "type": "boolean"
    }
  },
  "required": [ "activate" ]
}

```

6.2.1.1.2.8 Combo box

The **Combobox** shows or edits multistate values.



Use

Can be used on any page where multistate values are to be displayed or edited.

Features

Possibility to link a [BaObject](#) [► 868] to have to create only a single binding. All the required attributes are then linked via this binding and changes to the value are automatically written back to the PLC.

Attributes

The control inherits from [TextControl](#) [► 935] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

BAObject

```
tchmi:framework#/definitions/Symbol
```

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here \[► 63\]](#).
The attribute is not applicable to all controls.

BA

BaObject

Layout

Left

Top

fx

Reset

Create data binding...

Create function binding

Select value for ProjectNavigationTextual.BaObject

Quick search...

Server symbols Internal symbols Localizations Mapped symbols Controls

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1	PLC1	☐	PLC1
ProjectStructure	S PLC1.ProjectStructure	PLC1.ProjectStructure	☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount		number	☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents		object	☒	TerminalClientEvents

Settings

Symbol expression

Refresh Collapse All Unmap Symbol

2. OK Cancel

BA

BaObject

S PLC1::ProjectStructure

Common

Data

framework#/definitions/TcHmi.BuildingAutomation.Controls.Combobox.ComboboxItems

Data for the combobox (see [ComboboxItems \[► 871\]](#)).

Colors

ButtonColor

tchmi:framework#/definitions/SolidColor

Color of the button that opens the dropdown list (see [SolidColor](#)).

ButtonArrowColor

tchmi:framework#/definitions/SolidColor

Color of the arrow in the button (see [SolidColor](#)).

BaData

BaInterface

tchmi:framework#/definitions/Symbol

Allows linking a symbol (see [symbol](#)) that satisfies the [BaInterface](#) [► 999] of the control. With this symbol all necessary data points of the control can be linked with only one binding.

BaInterfaceSymbolNames

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Combobox.BaInterfaceSymbolNames

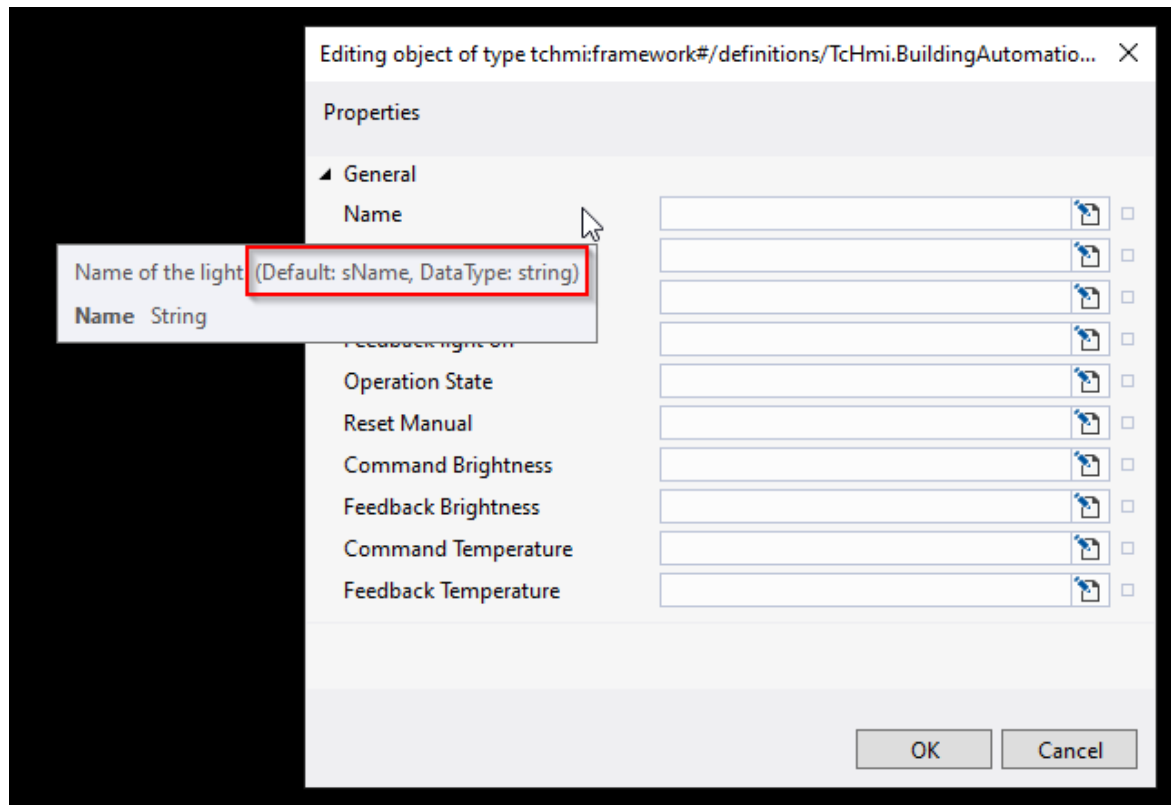
(see [BaInterfaceSymbolNames](#) [► 872])

tchmi:general#/definitions/Object

Allows editing the [BaInterfaceSymbolNames](#) [► 1000].



The default values of **BaInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BaInterfaceSymbolNames**:



Here is described how the **BaInterfaceSymbolNames** are overwritten by all controls of a type.

SelectedData (read-only)

framework#/definitions/TcHmi.BuildingAutomation.Controls.Combobox.ComboboxItem

Currently selected data (see [ComboboxItem](#) [► 871]).

SelectedValue

tchmi:general#/definitions/Number

Value of the currently selected data (see [number](#)).

UseSelectedValueFeedback

tchmi:general#/definitions/Boolean

Determines whether or not the attribute **SelectedValueFeedback** is observed (see [Boolean](#)).

SelectedValueFeedback

tchmi:general#/definitions/Number

Feedback for the selected value (see [number](#)).

Events

onChanged

Triggered when the selected value has changed. This happens when the user selects a new entry.

Events

Event	Description
onSelectedValueChanged	Triggered when the selected value has changed.
onUserInteractionFinished	Triggered when the user interaction with the combo box has finished.

6.2.1.1.2.8.1 Types

ComboboxItems

Collection of [ComboboxItem](#) [[▶ 871](#)].

```
"TcHmi.BuildingAutomation.Controls.Combobox.ComboboxItems": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "array",
  "title": "ComboboxItems",
  "items": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Combobox.ComboboxItem"
  }
}
```

ComboboxItem

Schema for the attribute [Data](#) [[▶ 869](#)] of the [Combobox](#) [[▶ 868](#)] control.

```
"TcHmi.BuildingAutomation.Controls.Combobox.ComboboxItem": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "title": "ComboboxItem",
  "description": "Defines a combobox item",
  "engineeringColumns": [
    "text",
    "value"
  ],
  "propertiesMeta": [
    {
      "name": "text",
      "category": "Common",
      "displayName": "Text",
      "description": "",
      "defaultValue": null,
      "defaultValueInternal": "Entry"
    },
    {
      "name": "value",
      "category": "Common",
      "displayName": "Value",
      "description": "",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "text": {
      "type": "string"
    },
    "value": {
      "type": "number"
    }
  }
}
```

```

    }
  }
}

```

BaInterfaceNames

Schema for the attribute [BaInterfaceSymbolNames](#) [► 870] of the [Combobox](#) [► 868] control.

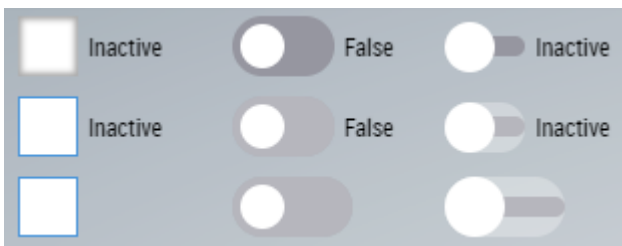
```

"TcHmi.BuildingAutomation.Controls.Combobox.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "value",
      "displayName": "Value",
      "description": "The symbol to set the value. (Default: Value, DataType: integer)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "valueFeedback",
      "displayName": "Value feedback",
      "description": "The feedback for the value." (Default: ValueFeedback, DataType: integer),
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "stateTexts",
      "displayName": "State texts",
      "description": "Array for the texts of the different values. (Default: StateTexts, DataType: s
tring[])",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "value": {
      "type": "string"
    },
    "valueFeedback": {
      "type": "string"
    },
    "stateTexts": {
      "type": "string"
    }
  }
}

```

6.2.1.1.2.9 Checkbox

The **checkbox** shows or edits binary values.



Use

Can be used on any page where binary values are to be displayed or edited.

Special features

The active and inactive text can be set (e.g. "On" / "Off").

The appearance can be customized using the [Appearance](#) [► 873] attribute (see image above).

Possibility to link a [BaObject](#) [► 873] to have to create only a single binding. All the required attributes are then linked via this binding and changes to the value are automatically written back to the PLC.

Attributes

The control inherits from [TextControl](#) [► 935] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

BA
BaObject

Layout
Left
Top

Reset
Create data binding...
Create function binding

Select value for ProjectNavigationTextual.BaObject

Quick search...

Server symbols Internal symbols Localizations Mapped symbols Controls

Name	Value	Datatype	Online	Mapped from
- TcHmiBaServerExtension - PLC1 ProjectStructure 1. - TcHmiBaServerExtension.GetUnits - TcHmiBaServerExtension.TerminalClientActiveEventsCount - TcHmiBaServerExtension.TerminalClientEvents	S PLC1	PLC1	☐	PLC1
	S PLC1.ProjectStructure		☐	ProjectStructure
			☒	GetUnits
		number	☒	TerminalClientActiveEv
		object	☒	TerminalClientEvents

Settings
Symbol expression

Refresh Collapse All Unmap Symbol

2. OK Cancel

BA
BaObject

S PLC1::ProjectStructure

Common

ActiveText

tchmi:general#/definitions/String

Determines the text that is displayed when *State* is TRUE (see [String](#)).

InactiveText

```
tchmi:general#/definitions/String
```

Determines the text that is displayed when *State* is FALSE (see [String](#)).

Appearance

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Checkbox.Appearance
```

Determines how the checkbox appears at the top (see [Appearance](#) [► 876]).

Colors

CheckBackgroundColor

```
tchmi:general#/definitions/SolidColor
```

Background color of the checkbox if *State* is TRUE (see [SolidColor](#)).



This attribute has no effect if *Appearance* is set to *ToggleSlider*.



CheckmarkColor

```
tchmi:general#/definitions/SolidColor
```

Color of the hook or toggle (see [SolidColor](#)).



BaData

BalInterface

```
tchmi:framework#/definitions/Symbol
```

Allows linking a symbol (see [symbol](#)) that satisfies the [BalInterface](#) [► 999] of the control. With this symbol all necessary data points of the control can be linked with only one binding.

BalInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Checkbox.BalInterfaceSymbolNames
```

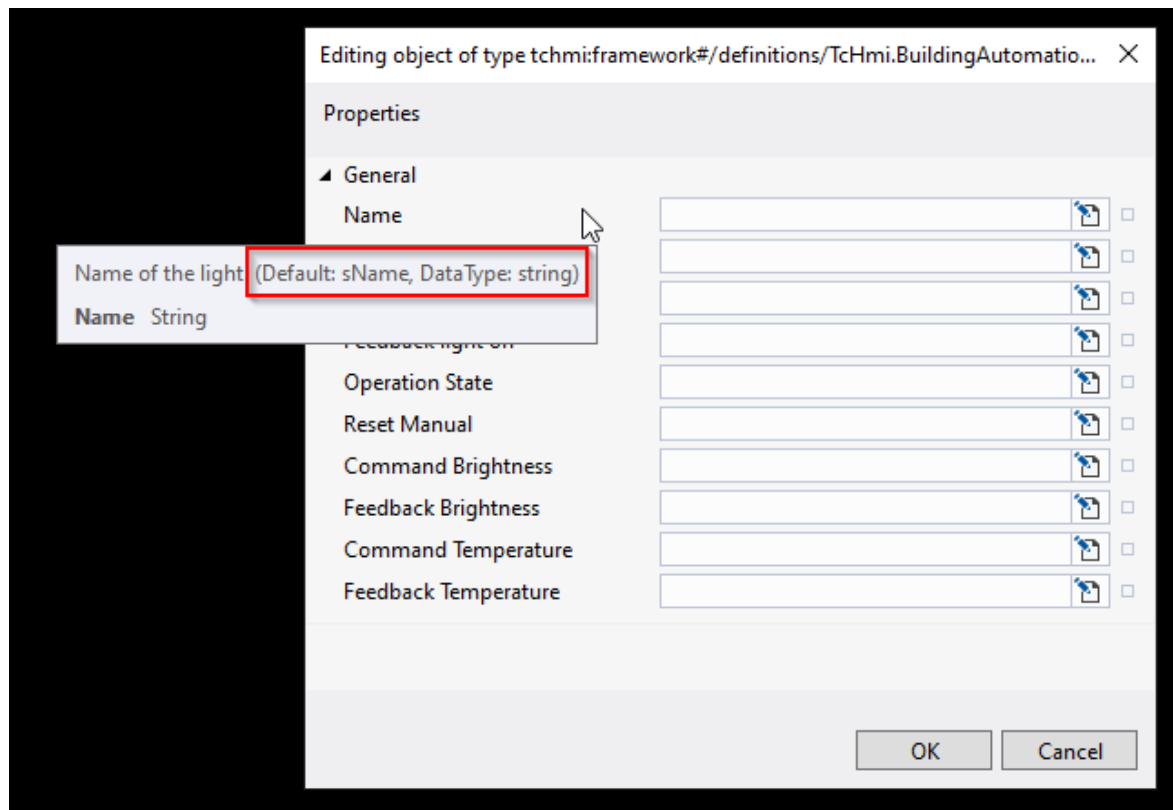
(see [BalInterfaceSymbolNames](#) [► 876])

```
tchmi:general#/definitions/Object
```

Allows editing the [BalInterfaceSymbolNames](#) [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the **BalInterfaceSymbolNames** are overwritten by all controls of a type.

State

tchmi:general#/definitions/Boolean

State of the checkbox (see [Boolean](#)).

StateFeedback

tchmi:general#/definitions/Boolean

Feedback for the state of the checkbox (see [Boolean](#)).

UseStateFeedback

tchmi:general#/definitions/Boolean

Determines whether or not the attribute **StateFeedback** is observed (see [Boolean](#)).

Events

Event	Description
onStateChanged	Triggered when the value of State has changed.
onUserInteractionFinished	Triggered when the user interaction with the checkbox has finished.

6.2.1.1.2.9.1 Types

BaInterfaceSymbolNames

Schema for the attribute BaInterfaceSymbolNames [► 874] of the Checkbox [► 872] control.

```
"TcHmi.BuildingAutomation.Controls.Checkbox.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "state",
      "displayName": "State",
      "description": "The symbol to set the state. (Default: State, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "stateFeedback",
      "displayName": "State feedback",
      "description": "The feedback for the state. (Default: StateFeedback, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "activeText",
      "displayName": "Active text",
      "description": "Text when the checkbox is active. (Default: ActiveText, DataType: string)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "inactiveText",
      "displayName": "Inactive text",
      "description": "Text when the checkbox is inactive. (Default: InactiveText, DataType: string)"
    },
    {
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "state": {
      "type": "string"
    },
    "stateFeedback": {
      "type": "string"
    },
    "activeText": {
      "type": "string"
    },
    "inactiveText": {
      "type": "string"
    }
  }
}
```

6.2.1.1.2.9.2 Enumerations

Appearance

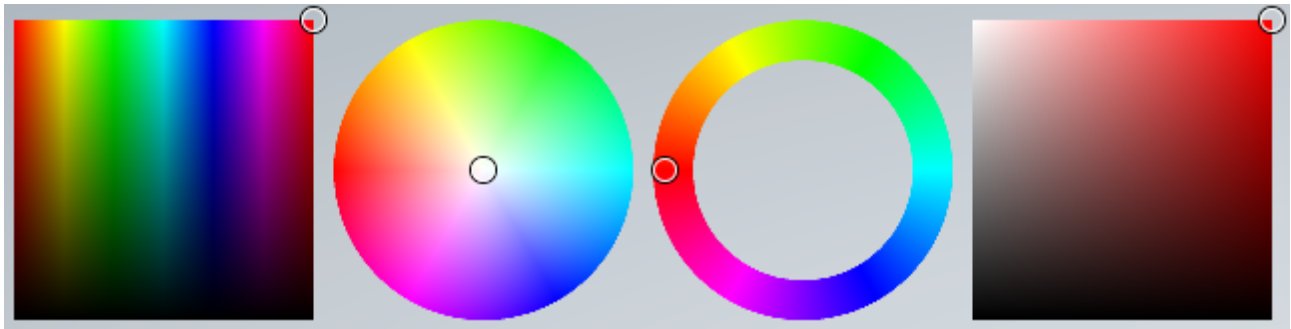
Schema for the attribute Appearance [► 874] of the Checkbox [► 872] control.

```
"TcHmi.BuildingAutomation.Controls.Checkbox.Appearance": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "checkbox",
      "value": 0
    }
  ],
}
```

```
{
  {
    "label": "toggleSwitch",
    "value": 1
  },
  {
    "label": "toggleSlider",
    "value": 2
  }
}
```

6.2.1.1.2.10 ColorPicker

The **ColorPicker** can be used to select a color from various color palettes.



Use

Can be used on any page where it is necessary to select a color.

Features

Color selection can be done using different color spaces.

HSL Ring

All colors are selected without shading.



HSV Circle

Selection of all colors up to white.



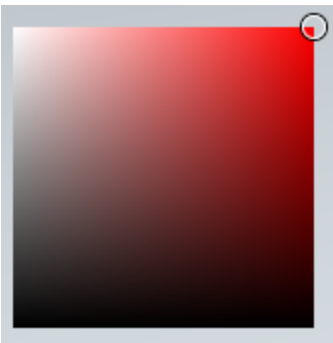
HSL Rect All color

Selection of all colors up to black.



HSL Rect single

Selection of all shades of a color. To change the color, change the attribute BackgroundColor.



Attributes

The control inherits from [TchHmiControl](#) and thus has the same attributes. In addition, there are the following attributes.

ColorPlateType

tchmi:framework#/definitions/TchHmi.BuildingAutomation.Controls.ColorPlateType

Defines the color palette used (see [ColorPlateType](#) [► 879]).

BackgroundColor

tchmi:framework#/definitions/SolidColor

Defines the background color when the HslRect1 color palette is selected (see [SolidColor](#)).

SelectedSolidColor (read-only)

tchmi:framework#/definitions/SolidColor

Selected color (see [SolidColor](#)).

SelectedRgbaColor (read-only)

tchmi:framework#/definitions/TchHmi.BuildingAutomation.RGBAColor

Selected color in RGBA format (see [RGBAColor](#) [► 880]).

Events

Event	Description
onSelectedColorChanged	Triggered when the selected color has changed.

6.2.1.1.2.10.1 Enumerations

ColorPlateType

Schema for the attribute `ColorPlateType` [► 878] of the `ColorPicker` [► 877].

```
"TcHmi.BuildingAutomation.Controls.ColorPicker.ColorPlateType": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [ 0, 1, 2, 3 ],
  "options": [
    {
      "label": "hslRect1",
      "value": 0
    },
    {
      "label": "hslRectAll",
      "value": 1
    },
    {
      "label": "hsvCircle",
      "value": 2
    },
    {
      "label": "hslRing",
      "value": 3
    }
  ]
}
```

6.2.1.1.2.10.2 Types

HSLColor

Schema for a color in HSL format.

```
"TcHmi.BuildingAutomation.HSLColor": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "title": "HSLColor",
  "propertiesMeta": [
    {
      "name": "h",
      "category": "Common",
      "displayName": "Hue",
      "displayPriority": 10,
      "description": "Hue 0-1",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "s",
      "category": "Common",
      "displayName": "Saturation",
      "displayPriority": 10,
      "description": "Saturation 0-1",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "l",
      "category": "Common",
      "displayName": "Lightness",
      "displayPriority": 10,
      "description": "Lightness 0-1",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "h": {
      "type": "number"
    },
    "s": {
      "type": "number"
    }
  },
}
```

```

    "1": {
      "type": "number"
    }
  }
}

```

RGBAColor

Schema for a color in RGBA format.

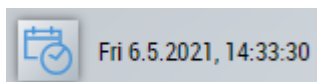
```

"TcHmi.BuildingAutomation.RGBAColor": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "title": "RGBAColor",
  "propertiesMeta": [
    {
      "name": "r",
      "category": "Common",
      "displayName": "Red",
      "displayPriority": 10,
      "description": "Red 0-255",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "g",
      "category": "Common",
      "displayName": "Green",
      "displayPriority": 10,
      "description": "Green 0-255",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "b",
      "category": "Common",
      "displayName": "Blue",
      "displayPriority": 10,
      "description": "Blue 0-255",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "a",
      "category": "Common",
      "displayName": "Alpha",
      "displayPriority": 10,
      "description": "Alpha 0-1",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "r": {
      "type": "number"
    },
    "g": {
      "type": "number"
    },
    "b": {
      "type": "number"
    },
    "a": {
      "type": "number"
    }
  }
}

```

6.2.1.1.2.11 DateTimeField

The **DateTimeField** can be used to select or display a date and time.

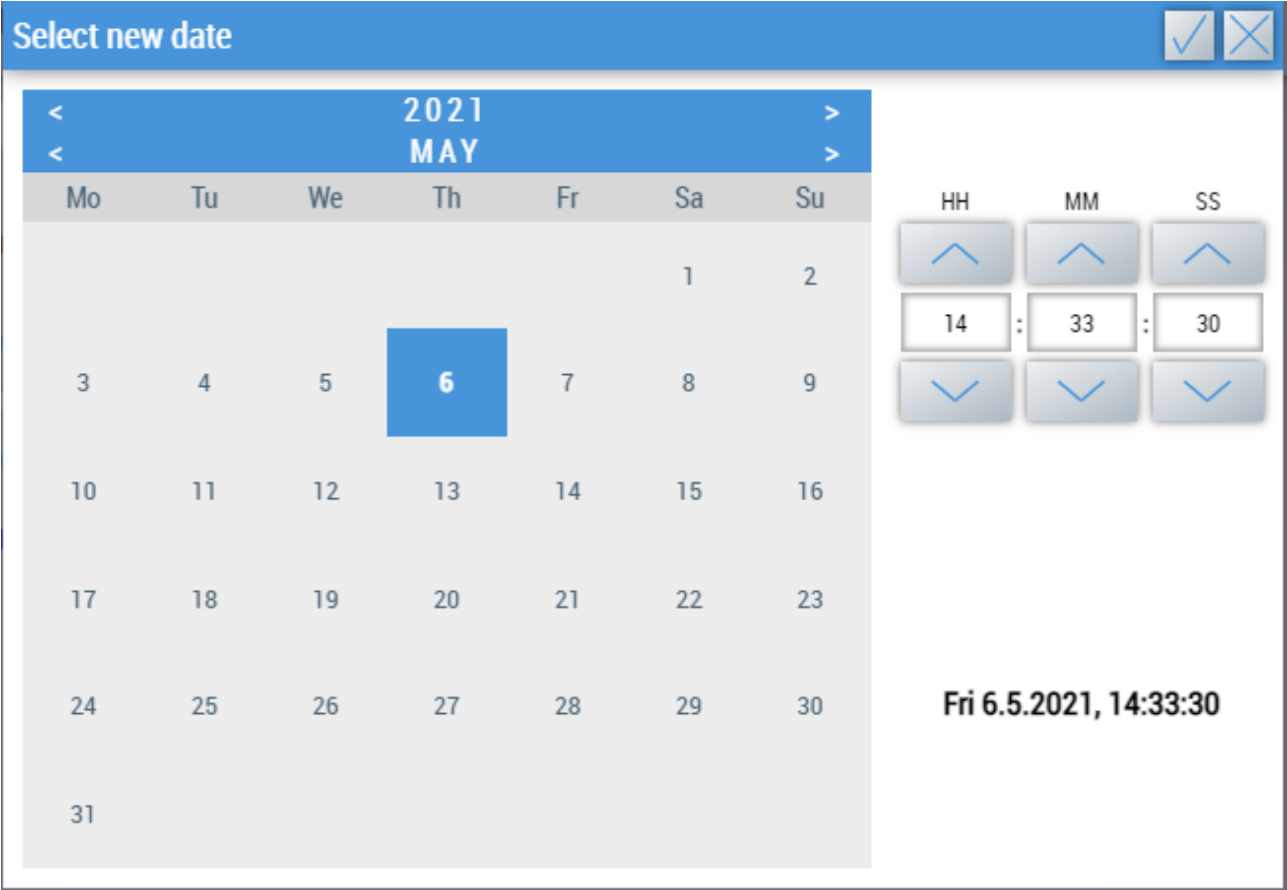


Use

Use on any page where date values are to be displayed or edited.

Features

If the attribute `ReadOnly` [► 924] is FALSE, the **DateTimePicker** can be opened via the button to select a new date or time.



Attributes

The control inherits from `TextControl` [► 935] and thus has the same attributes. In addition, there are the following attributes.

Common

DateTime

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BaDateTime`

Current value of time and date (see `DateTime` [► 929]).

Events

Event	Description
onDateChanged	Triggered when the value of the date or time has changed.
onUserInteractionFinished	Triggered when the user interaction with the <code>DateTimeField</code> has finished.

6.2.1.1.2.12 Paginator

The **Paginator** can be used to navigate between different content pages.

Use

Use on any page where you want to switch between different content pages in a gallery.

Features

Devices with touch screen

Switch between configured content pages with a swipe gesture to the left or right.

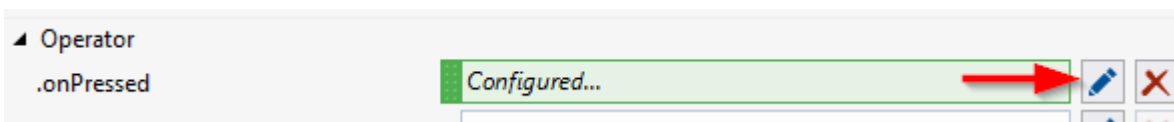
Devices without touch screen

Navigation through the content pages is done via various controls. For this purpose, the functions GoForward and GoBackward are called by events of the controls (e.g. OnPressed).

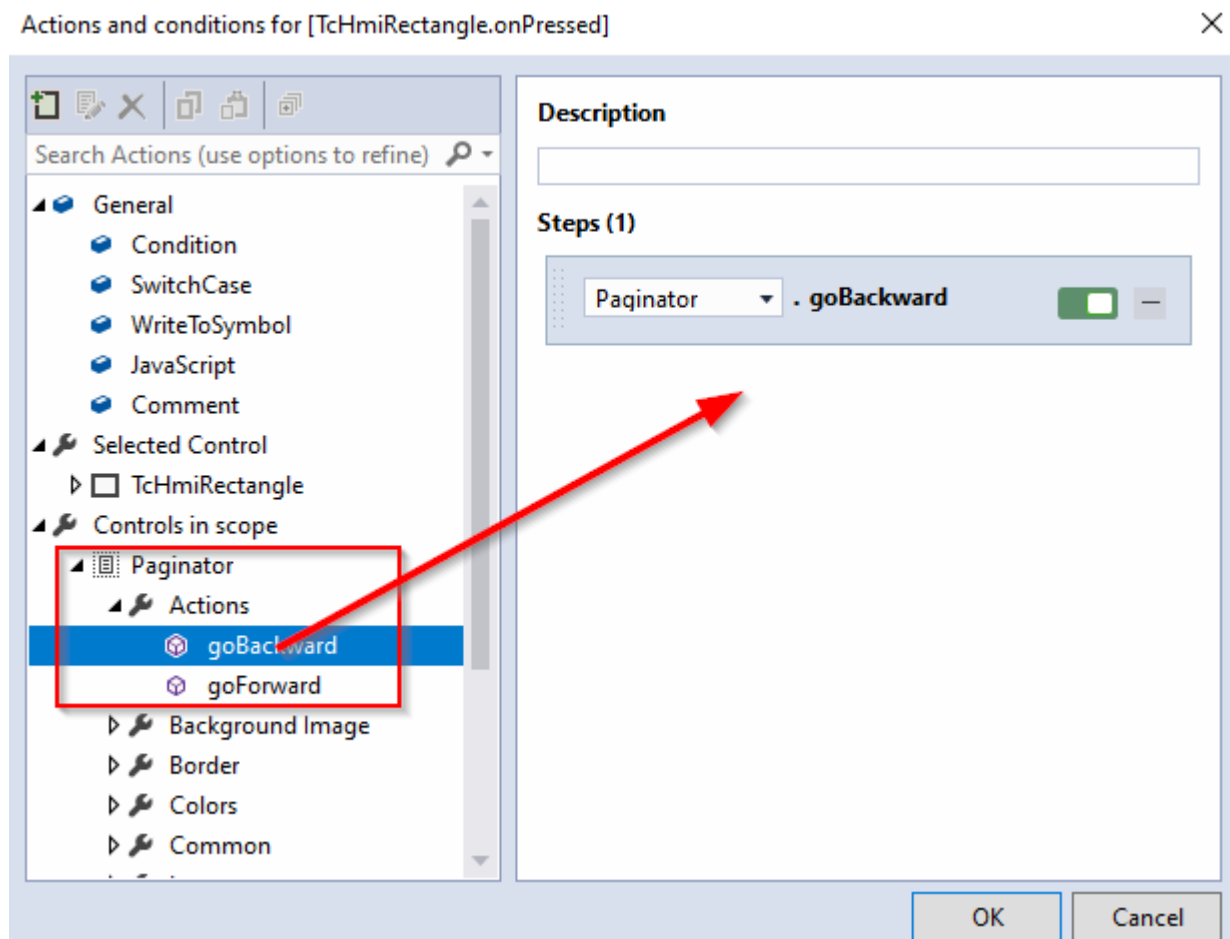
Explanation using the example of two rectangles, which are positioned to the left and right of the Paginator respectively.



The properties window of the respective rectangle is used to configure the **OnPressed** event.



Then the function **GoForward / GoBackward** is added.



Attributes

The control inherits from `TcHmiRegion` and thus has the same attributes. The `TargetContent` attribute is replaced by the `Pages` attribute.

Pages

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Paginator.Pages
```

Creating the pages to be navigated through.

Functions

GoForward

When the function is called, it navigates to the next page. If the navigation is currently on the last page, there is a change to the first page.

GoBackward

When the function is called, it navigates to the previous page. If the navigation is currently on the first page, there is a change to the last page.

6.2.1.1.2.12.1 Types

Pages

Schema for the attribute `Pages` [► 883] of the `Paginator` [► 882] control.

```
"TcHmi.BuildingAutomation.Controls.Paginator.Pages": {
  "type": "array",
  "items": {
```



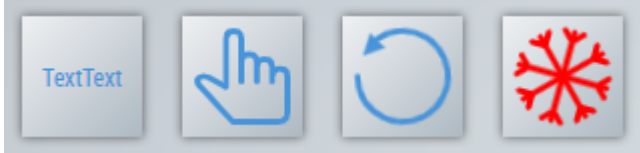
```

    "$ref": "tchmi:framework#/definitions/ContentPath"
  }
}

```

6.2.1.1.2.13 Button

The **button** is essentially the same as the [TcHmiButton](#). The only difference is extended functionalities for the icon, because the options of the icon package can be used here.



Use

Can be used wherever a button with extended icon functionality is required.

Features

Provides advanced functionality for icons from the NuGet package [TcHmiBalcons](#) [► 984].

Attributes

The control inherits from [TextControl](#) [► 935] and thus has the same attributes. In addition, there are the following attributes.

Icon

Path to the icon.

IconWidth

```
tchmi:general#/definitions/Number
```

Width of the icon (see [Number](#)).

IconWidthUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Unit of the width of the icon (see [DimensionUnit](#)).

IconHeight

```
tchmi:general#/definitions/Number
```

Height of the icon (see [Number](#)).

IconHeightUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Unit of the icon's height (see [DimensionUnit](#)).

IconHorizontalAlignment

```
tchmi:general#/definitions/HorizontalAlignment
```

Definition of the horizontal alignment of the icon within the button (see [HorizontalAlignment](#)).

IconVerticalAlignment

```
tchmi:general#/definitions/VerticalAlignment
```

Definition of the vertical alignment of the icon within the button (see [VerticalAlignment](#)).

IconRotation`tchmi:general#/definitions/Number`

Determines by how many degrees the icon should be rotated (see [Number](#)).

IconRotationSpeed`tchmi:general#/definitions/Number`

Determines the speed at which the icon should rotate (see [Number](#)).

IconRotationDirection`tchmi:general#/definitions/TcHmi.BuildingAutomation.Controls.Direction`

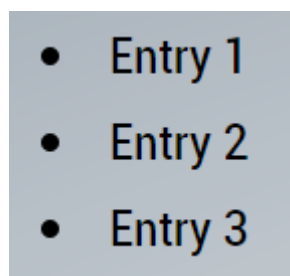
Determines the direction in which the icon rotates if the attribute *IconRotationSpeed* is defined. The default value is right (see [Direction](#)).

Events

Event	Description
onButtonPressed	Triggered when the button is pressed.
onButtonDoublePressed	Triggered when a double click on the button has been performed. The velocity for detecting a double click can be set globally [▶ 1014].

6.2.1.1.2.14 BulletPointList

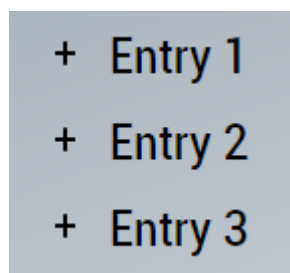
The **BulletPointList** lists various texts in bullet point form.

**Use**

Use on any page where a bullet point list is to be displayed.

Features

Change of the bullet by the attribute `ListStyleImage`.

**Attributes**

The control inherits from [TextControl](#) [[▶ 935](#)] and thus has the same attributes. In addition, there are the following attributes.

Entries

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BulletPointList.Entries
```

Bullet points to be displayed in the BulletPointList (see [Entries](#) [► 886]).

ListStyleImage

```
tchmi:framework#/definitions/Path
```

Path to the listing icon image.

6.2.1.1.2.14.1 Types

Entries

Collection of [String](#).

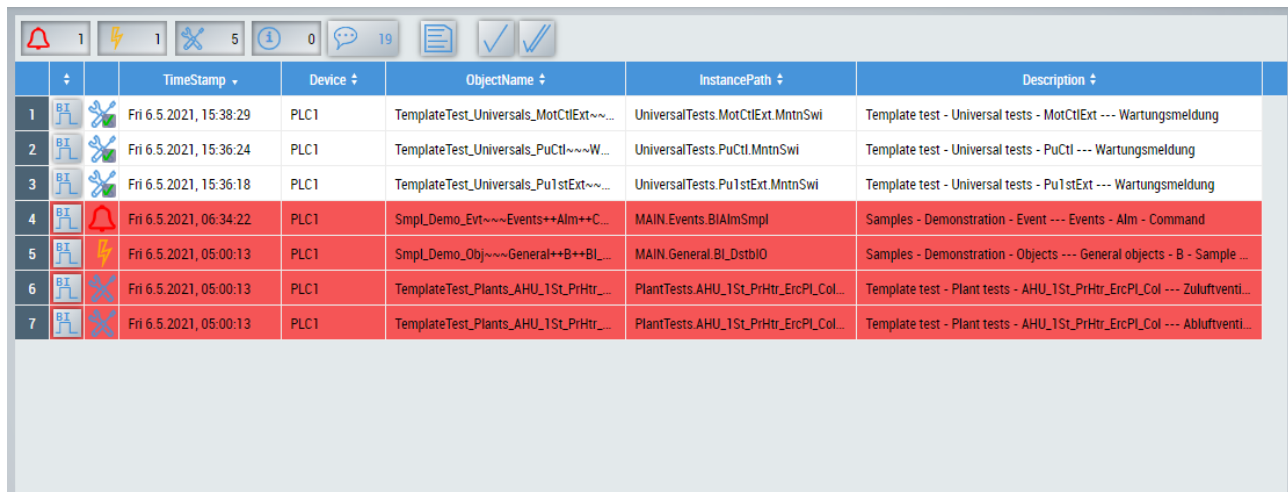
```
"TcHmi.BuildingAutomation.Controls.BulletPointList.Entries": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "array",
  "title": "Entries",
  "items": {
    "$ref": "tchmi:general#/definitions/String"
  }
}
```

6.2.1.1.3 Management

6.2.1.1.3.1 EventList

The **EventList** displays events in list form.

To use the control, the [generic functionalities](#) [► 63] of TcHmiBa must be used.



	TimeStamp	Device	ObjectName	InstancePath	Description
1	Fri 6.5.2021, 15:38:29	PLC1	TemplateTest_Universals_MotCtlExt~...	UniversalTests.MotCtlExt.MntnSwi	Template test - Universal tests - MotCtlExt --- Wartungsmeldung
2	Fri 6.5.2021, 15:36:24	PLC1	TemplateTest_Universals_PuCtl~...W...	UniversalTests.PuCtl.MntnSwi	Template test - Universal tests - PuCtl --- Wartungsmeldung
3	Fri 6.5.2021, 15:36:18	PLC1	TemplateTest_Universals_Pu1stExt~...	UniversalTests.Pu1stExt.MntnSwi	Template test - Universal tests - Pu1stExt --- Wartungsmeldung
4	Fri 6.5.2021, 06:34:22	PLC1	SmpL_Demo_Evt~...Events++Alm++C...	MAIN.Events.BIAImSmpL	Samples - Demonstration - Event --- Events - Alm - Command
5	Fri 6.5.2021, 05:00:13	PLC1	SmpL_Demo_Obj~...General++B++BL...	MAIN.General.BL_DstbID	Samples - Demonstration - Objects --- General objects - B - Sample ...
6	Fri 6.5.2021, 05:00:13	PLC1	TemplateTest_Plants_AHU_1St_PrHtr...	PlantTests.AHU_1St_PrHtr_ErcPL_Col...	Template test - Plant tests - AHU_1St_PrHtr_ErcPL_Col --- Zuluftventi...
7	Fri 6.5.2021, 05:00:13	PLC1	TemplateTest_Plants_AHU_1St_PrHtr...	PlantTests.AHU_1St_PrHtr_ErcPL_Col...	Template test - Plant tests - AHU_1St_PrHtr_ErcPL_Col --- Abluftventi...

Use

Use on any page where events are to be listed.

Features

Displays the events from a specific BaObject or BaView. It can also display the events of all connected controllers.

Using the buttons in the upper area, events can be filtered by different event types.



Allows you to acknowledge one or all events. The event currently selected in the list is acknowledged.



The button **History** shows or hides the event history.

Attributes

The control inherits from [BaseRoomControl](#) [► 934] and thus has the same attributes. In addition, there are the following attributes.

BA

BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

BA
BaObject

Layout
Left
Top

Reset
Create data binding...
Create function binding

Select value for ProjectNavigationTextual.BaObject

Quick search...

Server symbols Internal symbols Localizations Mapped symbols Controls

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Settings
Symbol expression

Refresh Collapse All Unmap Symbol

2 OK Cancel

BA
BaObject

S PLC1::ProjectStructure

Common

IsGlobalEventList

tchmi:general#/definitions/Boolean

Determines whether the events of all connected controllers should be displayed (see [Boolean](#)).

ActiveEventsCount (read-only)

tchmi:general#/definitions/Number

Number of active events that the user can acknowledge (see [number](#)).

ColumnSorting

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Management.EventList.ColumnSorting

Sets the order of the columns (see [ColumnSorting](#) [► 889]).

The default attribute setting can also be overwritten globally for all EventList controls in [CodeBehind](#).

```
TcHmi.EventProvider.register('onInitialized', function (e, data) {
    e.destroy();
    TcHmi.BuildingAutomation.Controls.Management.EventList.DefaultColumnSorting = [
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.baIdentifier,
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.event,
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.eventClass,
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.timestamp,
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.device,
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.objectName,
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.instancePath,
        TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.description
    ];
});
```

Events

Event	Description
onEventsChanged	This is triggered when the events collection has changed.
onEventAcknowledged	Is triggered when an event has been acknowledged.
onAllEventsAcknowledged	Triggered when all events have been acknowledged.

6.2.1.1.3.1.1 Enumerations

Columns

Schema for enumeration of available columns of [EventList](#) [► 886].

```
"TcHmi.BuildingAutomation.Controls.Management.EventList.Columns": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Columns",
  "type": "integer",
  "enum": [
    0,
    1,
    2,
    3,
    4,
    5,
    6
  ],
  "options": [
    {
      "label": "baIdentifier",
      "value": 0
    },
    {
      "label": "event",
      "value": 1
    },
    {
      "label": "timestamp",
      "value": 2
    },
    {
      "label": "device",
```

```

    "value": 3
  },
  {
    "label": "objectName",
    "value": 4
  },
  {
    "label": "instancePath",
    "value": 5
  },
  {
    "label": "description",
    "value": 6
  }
]
}

```

6.2.1.1.3.1.2 Types

ColumnSorting

Schema for the attribute [ColumnSorting](#) [► 888] of the [EventList](#) [► 886].

```

"TcHmi.BuildingAutomation.Controls.Management.EventList.ColumnSorting": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "ColumnSorting",
  "type": "array",
  "items": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Management.EventList.Columns"
  }
}

```

6.2.1.1.3.2 ProjectNavigationTextual

The **ProjectNavigationTextual** is one of the generic controls and can be used to navigate through the entire project structure of a device. It shows the type, description, value and events of the objects.



More detailed information about the generic capabilities of TcHmiBa and how they can be used can be found [here](#) [► 63].

The screenshot shows the 'Project Structure PLC1' window. On the left, a tree view lists the following objects:

- Samples
- Demonstration
- Objects
- General objects
- A
- B

On the right, a detailed view of object 'B' is shown, listing various data points and their current states:

Object	Value	State
Binary Value	☐	Aus
Sample BV Display	☒	Ein
Sample BV Setpoint	☒	Ein
Sample BV Pushbutton	☐	Aus
Sample BI (Process Data)	☐	Normal
Sample BI (Raw Data)	☐	Normal
Sample BI Disturb (IO)	☒	Ausgelöst

Use

To be able to navigate through the **complete project structure** of a runtime, it must be linked with the attribute `BaObject` [► 893]. Furthermore, the control allows you to navigate through the children of any view. If only a single object (that is, no view) is linked, then only one entry is displayed with this object.

Features

Generic navigation

The navigation is configured generically based on the structure of the linked object / view. Thus it is possible to reach all objects with **only one binding**. For each object, the **type** (e.g. Analog Input, Structured View), the **description**, a **value** (if available) and the **events** are displayed.

Navigation to content page

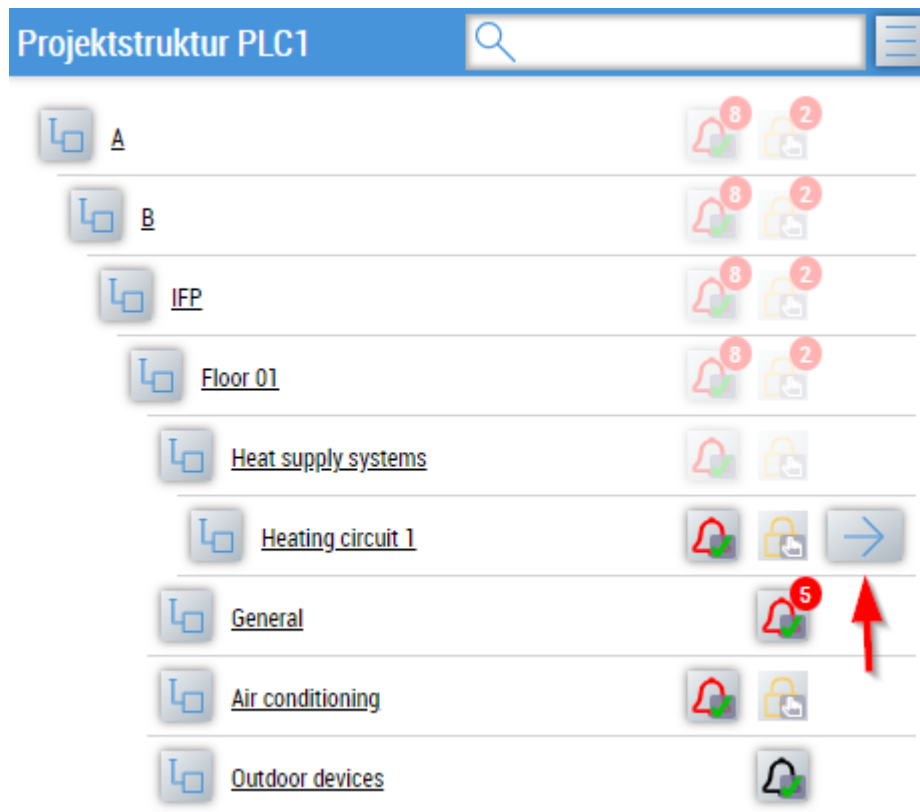
Sometimes it may be necessary to navigate directly to the content page of a system from the project navigation. This can be configured by assigning the SymbolPath of the system's view as the name of the content page to be navigated to.

Sample:

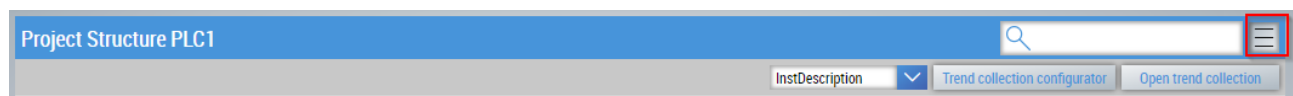
Heating circuit 1	
Acknowledge	☐ Inactive
Description	A - B - IFP - Floor 01 - Heat supply systems - Heating circuit 1
Description (Instance)	Heating circuit 1
Instance ID	105
Object Name	A-B-I-F01-HTG-HTC01
Object Name (Instance)	HTC01
Symbol Name	HTC01
Symbol Path	TF8040_Tutorial_04_PLC.MAIN.H.HTC01

The content page of the heating circuit must therefore be named
`TF8040_Tutorial_04_PLC_MAIN.H.HTC01.content` .

After that, there is an arrow at the end of the line of the heating circuit in the project navigation. Selecting this arrow will navigate to the corresponding content page:

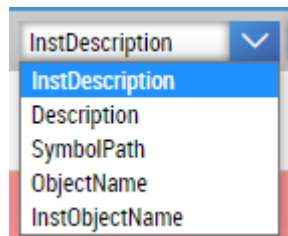


Header menu



The button in the header opens a menu in which:

- the label to be displayed for the entries can be selected.



- the [trend configurator](#) [► 45] can be opened and the generated trend configurations can be displayed.

The search field allows you to filter the list by a specific term.

Parameter window

The object type is identified by the icon of the button at the start of an entry. Pressing this button opens the parameter window of this object.

The 'Samples' dialog box displays the following parameters for a selected object:

Acknowledge	☐ Inactive
Description	Samples
Description (Instance)	Samples
Instance ID	100
Object Name	Smpl
Object Name (Instance)	Smpl
Symbol Name	Smpl
Symbol Path	Samples.MAIN.Smpl
Tag	

The content of this window differs, depending on the selected object. First, all parameters of the respective object are displayed. In the image above, a view is selected that has fewer parameters than an analog input, for example. Which parameters are displayed and which are writable or read-only depends on the rights of the logged-in user.

The changed parameters are written to the PLC when the dialog is closed via the **Confirm** button. When you click the **Close** button, the settings you have made will be lost.

Depending on the selected object, the header shows one or two additional buttons.

The button with the event bell is available for every event-enabled object and replaces the contents of the window with the event list of the object when clicked.

The 'Samples' dialog box displays the event list for a selected object. The header shows the following buttons: 1, 1, 0, 0, 0, 1, 1, 1.

		TimeStamp	Device	ObjectName	InstancePath
1		Di 19.4.2021, 07:41:17	PLC1	Smpl_Demo_Evt~~~Events++Alm++CMD001	MAIN.Events.BIAlmSmpl
2		Di 19.4.2021, 06:04:48	PLC1	Smpl_Demo_Obj~~~General++B++BI_DstbIO	MAIN.General.BI_DstbIO

The second button is intended for various Trend functionalities and is available only for objects that support them or for views that contain objects with trend functions.

Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

The screenshot shows the configuration of the **BaObject** attribute for the **ProjectNavigationTextual** control. A context menu is open over the **BaObject** property, with options: **Reset**, **Create data binding...** (highlighted), and **Create function binding**.

Below the menu, a dialog titled "Select value for ProjectNavigationTextual.BaObject" is shown. It features a search bar and tabs for "Server symbols", "Internal symbols", "Localizations", "Mapped symbols", and "Controls". The "Mapped symbols" tab is active, displaying a table of available symbols.

Name	Value	Datatype	Online	Mapped from
- TcHmiBaServerExtension - PLC1 ProjectStructure (highlighted with a red box and '1') - TcHmiBaServerExtension.GetUnits - TcHmiBaServerExtension.TerminalClientActiveEventsCount - TcHmiBaServerExtension.TerminalClientEvents	- PLC1 - PLC1.ProjectStructure - GetUnits - TerminalClientActiveEv - TerminalClientEvents	- PLC1 - PLC1.ProjectStructure - GetUnits - TerminalClientActiveEv - TerminalClientEvents	☐ ☐ ☒ ☒ ☒	- PLC1 - ProjectStructure - GetUnits - TerminalClientActiveEv - TerminalClientEvents

At the bottom of the dialog, there are buttons for "Refresh", "Collapse All", "Unmap Symbol", and "OK" (highlighted with a red box and '2').

The final configuration shows the **BaObject** property set to **PLC1::ProjectStructure**.

Common

BaUsedTitle

TcHmi.BuildingAutomation.Controls.ProjectNavigationTextual.BaUsedTitle

Determines which parameter is used for the description in an entry (see [BaUsedTitle](#) [► 894]). The setting can be customized in the client.

Show header

tchmi:general#/definitions/Boolean

Determines whether the header is displayed or not (see [Boolean](#)).

6.2.1.1.3.2.1 Enumerations

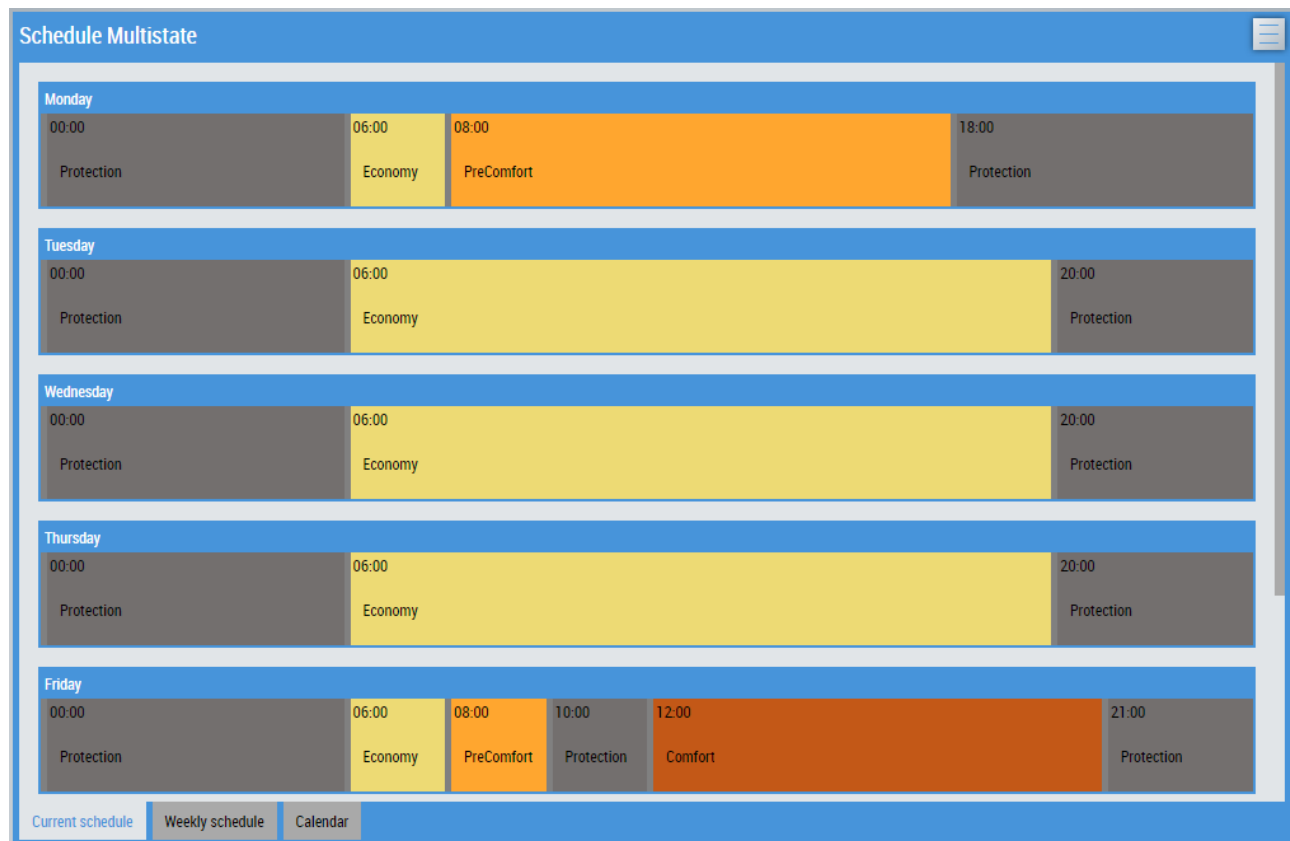
BaUsedTitle

Schema for the attribute [BaUsedTitle](#) [► 893] of the [ProjectNavigationTextual](#) [► 889] control.

```
"TcHmi.BuildingAutomation.Controls.ProjectNavigationTextual.BaUsedTitle": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    "InstDescription",
    "Description",
    "SymbolPath",
    "ObjectName",
    "InstObjectName"
  ]
}
```

6.2.1.1.3.3 Schedule

The **Schedule** can be used to display and operate schedules and calendar entries. The current schedule is created on the basis of the weekly schedule and exceptions.



Use

Use on any page where a schedule is to be managed.

If a Schedule object is passed to the [BaObject](#) [► 896] attribute, the [generic functions](#) [► 63] can be used.

Features

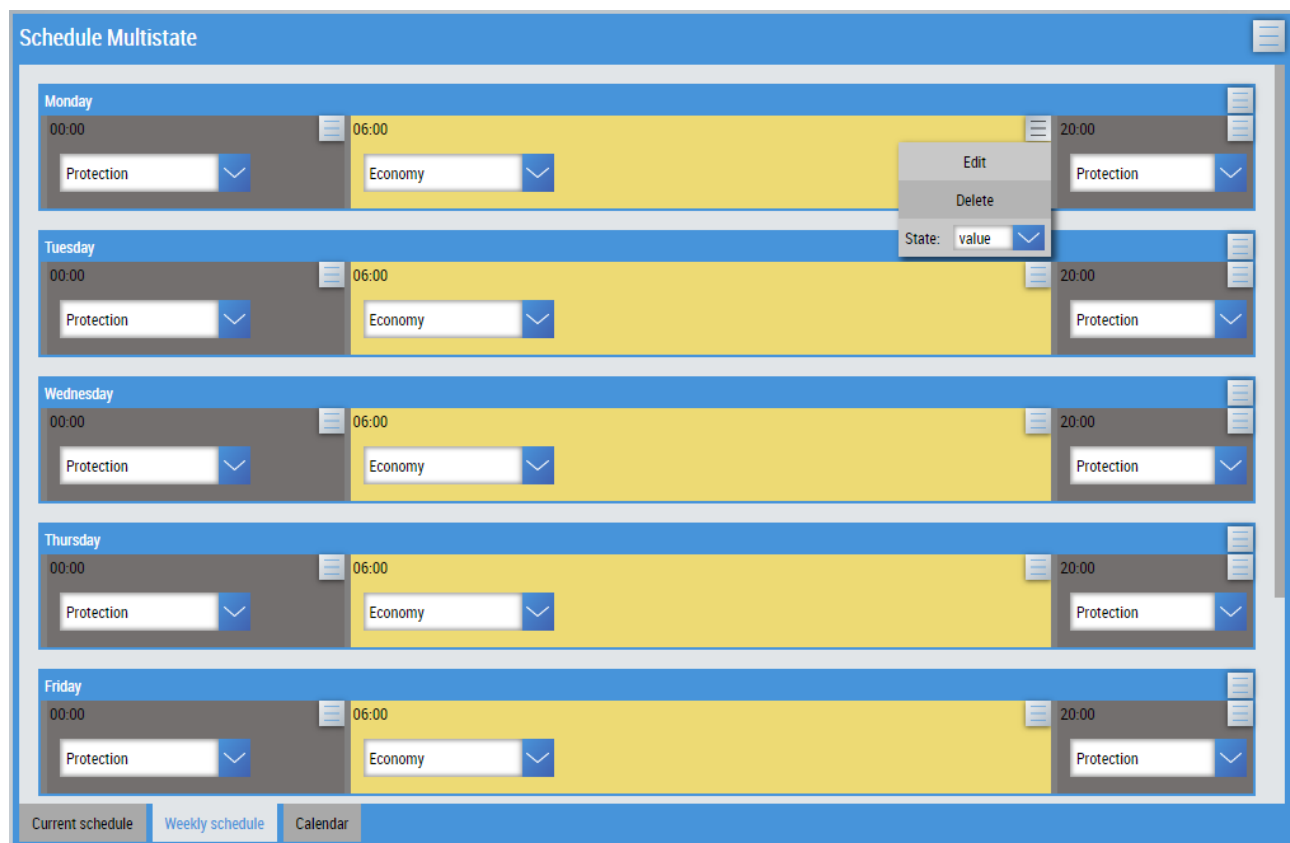
Resulting schedule

The first tab, Current schedule, displays the combination of the weekly schedule and the exceptions. The following hierarchy applies:

1. Local exceptions
2. Global exceptions
3. Weekly schedule

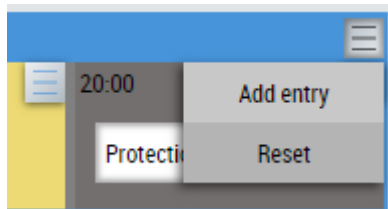
Editing the weekly schedule

On the **Weekly schedule** tab it is possible to edit the weekly schedule without taking into account exceptions that have already been defined.



In this view there is a schedule with different entries for each day. An entry can be edited or deleted via its menu. The start and end time or position can also be changed with the mouse or finger.

Each daily schedule also has a menu that can be used to add entries and reset changes.



Managing exceptions

The exceptions are managed on the **Calendar** tab. For more information on how to use it, see the [Calendar](#) [► 834] control.

Menu

Using the menu in the upper right-hand area of the schedule, you can either transfer all changes made to the PLC or discard them.



Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

BA

BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

Select value for ProjectNavigationTextual.BaObject

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure 1.	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Settings

Symbol expression

Refresh Collapse All Unmap Symbol **2.** OK Cancel

BA

BaObject S PLC1::ProjectStructure

Common

Orientation

TcHmi.BuildingAutomation.Controls.Orientation

Determines the orientation of the weekly schedule (see [Orientation](#) [► 926]).

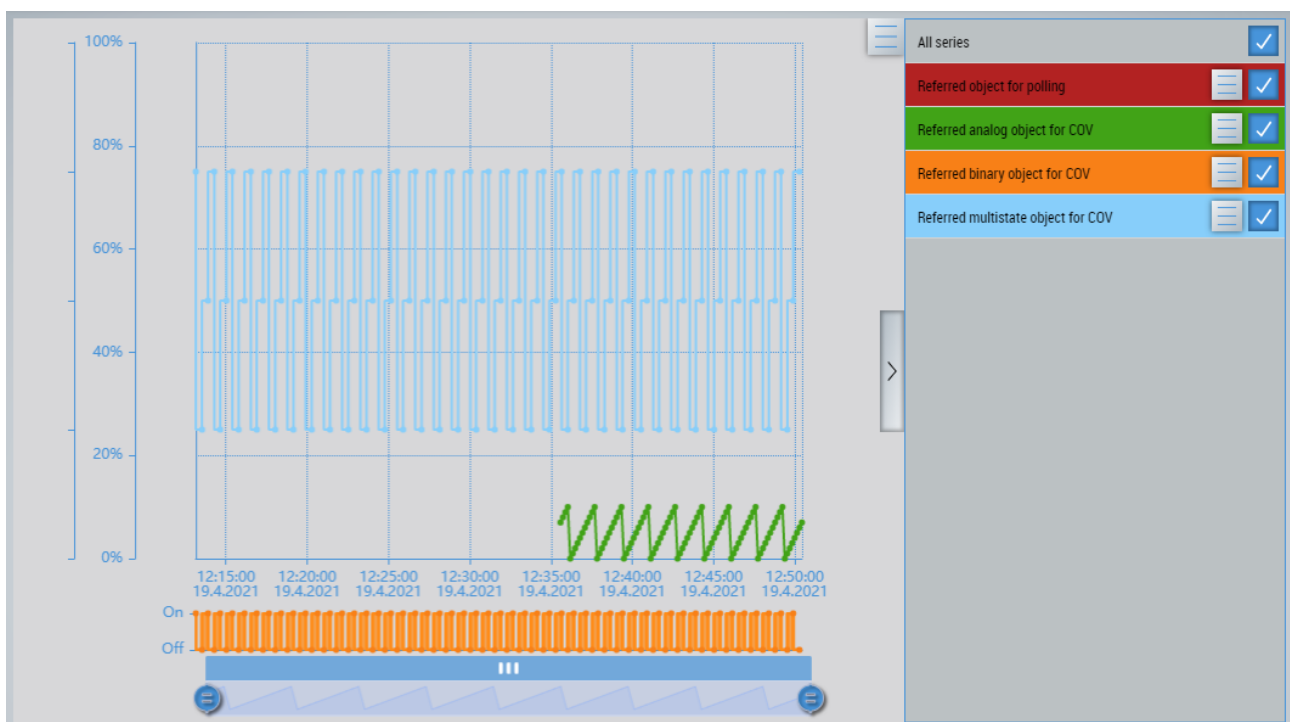
SnapPeriod

tchmi:general#/definitions/Number

Determines how precisely schedule entries can be set (see [Number](#)). If *SnapPeriode* is set to 15, for example, entries can be set to the nearest quarter of an hour.

6.2.1.1.3.4 Trend

The **Trend** control can display multiple trend curves. It allows you to select different trend curves and change the settings of all axes.



Use

Can be used on any page where a trend is to be displayed. Allows linking to a [BaObject](#) [► 900] of type Trend object or View.



For more information, see the documentation on [Trending](#) [► 45].

Features

Multiple trend curves

If the BaObject is a **trend object**, then only the associated trend curve is displayed. It is not possible to select from different trend curves.

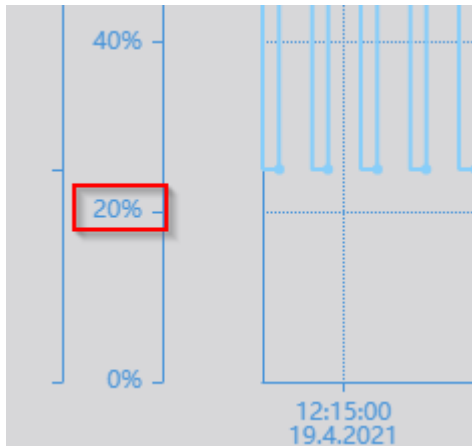
If the BaObject is a **View**, it is searched for trend objects and existing trend curves are displayed accordingly. It is possible to select from different trend curves if more than two trend objects are found.



In the listing, the trend curves to be displayed in the chart can be selected via the checkboxes. The adjacent button opens the [parameter window](#) [► 891] of the respective trend object.

Axis parameterization

The settings of a y-axis can be opened by selecting the respective scale values.



Axis configuration	
Auto scale	☐
Min	0.00
Max	100.00

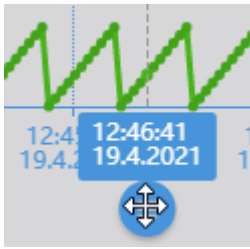
Menu

The **menu** allows further settings for the trend.

☐	Cursor
☒	Datenzoom
Wiederherstellen	
Update	
☐	Auto update
Ref Object	▼
InstDescription	▼

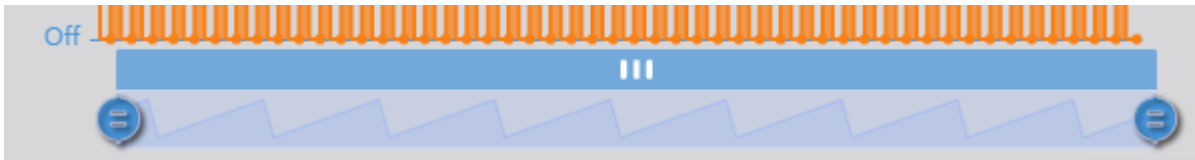
Cursor

If **Cursor** is activated, a cursor is displayed under the x-axis. By default, this function is disabled.



Data zoom

The zoom can be shown and hidden via the **checkbox**. By default, the zoom is shown.



Redo

Restores the default settings.

Update

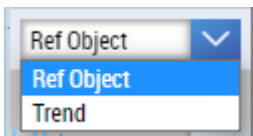
The trend curves can be updated once.

Auto Update

If the checkbox is checked, the trend curves are automatically updated as soon as new trend entries are available.

Displayed objects

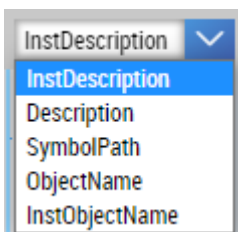
Determines the objects to be displayed in the listing.



- RefObject: Displays recorded values.
- Trend: Displays all trend objects that record a value.

Displayed label

Selection of the label to be used in the listing.



Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

BA

BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.

i More detailed information about the generic possibilities of TcHmiBa can be found [here](#) **[▶ 63]**. The attribute is not applicable to all controls.

BA

BaObject

Layout

Left

Top

fx

Reset

Create data binding...

Create function binding

Select value for ProjectNavigationTextual.BaObject

Quick search...

Server symbols

Internal symbols

Localizations

Mapped symbols

Controls

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Settings

Symbol expression

Refresh

Collapse All

Unmap Symbol

2. OK

Cancel

BA

BaObject

S PLC1::ProjectStructure

BA

BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.

i More detailed information about the generic possibilities of TcHmiBa can be found [here](#) **[▶ 63]**. The attribute is not applicable to all controls.

BA

BaObject

Layout

Left

Top

Reset

Create data binding...

Create function binding

Select value for ProjectNavigationTextual.BaObject

Quick search...

Server symbols Internal symbols Localizations Mapped symbols Controls

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Settings

Symbol expression

Refresh Collapse All Unmap Symbol

2. OK Cancel

BA

BaObject

S PLC1::ProjectStructure

6.2.1.1.4 Plants

6.2.1.1.4.1 Motor

The **motor** template shows the status of a motor graphically and, if available, the feedback in text form.



Use

Use on any page where a template of the **Motor** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
Cmd	FB_BA_BO_IO	x	Command
MdlIt	FB_BA_BO_IO		Feedback

Hierarchy:

- BaObject

- Cmd
- Mdlr

Corresponds to the PLC templates

- FB_BA_MotCtl
- FB_BA_MotCtlExt
- FB_BA_Pu1st
- FB_BA_Pu1stExt
- FB_BA_PuCtl

Attributes

This template inherits from the [UilconFdbStp \[► 938\]](#) control. In addition, there is the following attribute.

Common

DisplayMode

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Templates.Universal.Motor.DisplayMode

Determines the icon to be displayed (see [DisplayMode \[► 902\]](#)).

6.2.1.1.4.1.1 Enumeration

DisplayMode

Schema for the [DisplayMode \[► 902\]](#) attribute of the [Motor \[► 901\]](#) template.

```
"TcHmi.BuildingAutomation.Templates.Universal.Motor.DisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Type",
  "type": "integer",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "fan",
      "value": 0
    },
    {
      "label": "motor",
      "value": 1
    },
    {
      "label": "pump",
      "value": 2
    }
  ]
}
```

6.2.1.1.4.1.2 Types

SubBaTemplateDescription

Schema of a [Motor \[► 901\]](#) template for the BaTemplateDescription.

```
"TcHmi.BuildingAutomation.Templates.Universal.Motor.SubTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "SubTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {

```

```

        "name": "identifier",
        "displayName": "Identifier",
        "description": "Identifier of the motor.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "subTemplate",
        "displayName": "Motor",
        "description": "The template description of the motor.",
        "defaultValue": null,
        "defaultValueInternal": null
    }
],
"properties": {
    "identifier": {
        "type": "string"
    },
    "subTemplate": {
        "$ref": "tchmi:framework#/definitions/
TchHmi.BuildingAutomation.Templates.Universal.Motor.TemplateDescription"
    }
}
}

```

BaTemplateDescription

Schema for the attribute [BaTemplateDescription \[▸ 933\]](#) of the [Motor \[▸ 901\]](#) template.

```

"TchHmi.BuildingAutomation.Controls.Plants.Motor.BaTemplateDescription": {
    "$schema": "http://json-schema.org/draft-04/schema",
    "title": "BaTemplateDescription",
    "type": "object",
    "propertiesMeta": [
        {
            "name": "feedback",
            "displayName": "Feedback",
            "description": "The feedback of the motor. (Default: Fdb, DataType: Analog)",
            "defaultValue": null,
            "defaultValueInternal": null
        },
        {
            "name": "setpoint",
            "displayName": "Setpoint",
            "description": "The setpoint of the motor. (Default: Mdlt, DataType: Analog)",
            "defaultValue": null,
            "defaultValueInternal": null
        },
        {
            "name": "command",
            "displayName": "Command",
            "description": "The command of the motor. (Default: Cmd, DataType: Binary)",
            "defaultValue": null,
            "defaultValueInternal": null
        }
    ],
    "properties": {
        "feedback": {
            "type": "string"
        },
        "setpoint": {
            "type": "string"
        },
        "command": {
            "type": "string"
        }
    },
    "required": [ "command" ]
}

```

6.2.1.1.4.2 Pump

The **Pump** template shows the state of a pump graphically and, if available, the feedback in text form.



Use

Use on any page where a template of the **Pump** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
Cmd	FB_BA_BO_IO	x	Command
Mdlt	FB_BA_BO_IO		Feedback

Hierarchy:

- BaObject
 - Cmd
 - Mdlt

Corresponds to the PLC template

- FB_BA_MotCtl
- FB_BA_MotCtlExt
- FB_BA_Pu1st
- FB_BA_Pu1stExt
- FB_BA_PuCtl

Attribute

This template inherits from the [UilconFdbStp](#) [► 938] control. In addition, there is the following attribute.

Common

DisplayMode

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Templates.Universal.Pump.DisplayMode

Determines the icon to be displayed (see [DisplayMode](#) [► 904]).

6.2.1.1.4.2.1 Enumeration

DisplayMode

Schema for the attribute [DisplayMode](#) [► 904] of the [Pump](#) [► 903] template.

```
"TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.DisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Type",
  "type": "integer",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
```

```

        "label": "temperature",
        "value": 0
    },
    {
        "label": "pressure",
        "value": 1
    },
    {
        "label": "humidity",
        "value": 2
    }
]
}

```

6.2.1.1.4.2.2 Types

SubBaTemplateDescription

Schema of a [Pump \[► 903\]](#) template for the BaTemplateDescription.

```

"Tchmi.BuildingAutomation.Templates.Universal.Pump.SubTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "SubTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "identifier",
      "displayName": "Identifier",
      "description": "Identifier of the pump.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "subTemplate",
      "displayName": "Pump",
      "description": "The template description of the pump.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "identifier": {
      "type": "string"
    },
    "subTemplate": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomatin.Templates.Universal.Pump.TemplateDescription"
    }
  }
}

```

BaTemplateDescription

Schema for the [BaTemplateDescription \[► 933\]](#) attribute of the [Pump \[► 903\]](#) template.

```

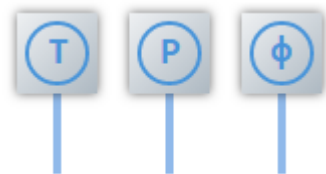
"Tchmi.BuildingAutomation.Templates.Universal.Pump.TemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "TemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "The feedback of the pump.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "command",
      "displayName": "Command",
      "description": "The command of the pump.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "feedback": {

```

```
    "type": "string"
  },
  "command": {
    "type": "string"
  }
},
"required": [ "command" ]
}
```

6.2.1.1.4.3 **SensorAnalog**

The **SensorAnalog** template displays the feedback of an analog value and, if linked, a setpoint.



Use

Use on any page where a template of the **SensorAnalog** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
MV	FB_BA_AI_IO	x	Feedback

Hierarchy:

- BaObject
 - MV

Corresponds to the PLC template

- FB_BA_SensorAnalog_IO

Attribute

This template inherits from the [UilconFdbStp](#) [► 938] control. In addition, there is the following attribute.

Common

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Templates.Universal.SensorAnalog.DisplayMode
```

Determines the icon to be displayed (see [DisplayMode](#) [► 906]).

6.2.1.1.4.3.1 **Enumeration**

DisplayMode

Schema for the [DisplayMode](#) [► 906] attribute of the [SensorAnalog](#) [► 906] template.

```
"TcHmi.BuildingAutomation.Templates.Universal.SensorAnalog.DisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Type",
  "type": "integer",
  "enum": [
    0,
```

```

    1,
    2
  ],
  "options": [
    {
      "label": "temperature",
      "value": 0
    },
    {
      "label": "pressure",
      "value": 1
    },
    {
      "label": "humidity",
      "value": 2
    }
  ]
}

```

6.2.1.1.4.3.2 Types

SubBaTemplateDescription

Schema of a [SensorAnalog](#) [► 906] template for the BaTemplateDescription.

```

"TcHmi.BuildingAutomation.Templates.Universal.SensorAnalog.SubTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "SubTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "identifier",
      "displayName": "Identifier",
      "description": "Identifier of the sensor.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "subTemplate",
      "displayName": "Sensor Analog",
      "description": "The template description of the sensor.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "identifier": {
      "type": "string"
    },
    "subTemplate": {
      "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Templates.Universal.SensorAnalog.TemplateDescription"
    }
  }
}

```

BaTemplateDescription

Schema for the [BaTemplateDescription](#) [► 933] attribute of the [SensorAnalog](#) [► 906] template.

```

"TcHmi.BuildingAutomation.Templates.Universal.SensorAnalog.TemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "TemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "The feedback of the sensor. (Default: MV, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "feedback": {
      "type": "string"
    }
  }
}

```



```

    },
    "required": [ "feedback" ]
  }
}

```

6.2.1.1.4.4 Valve

The **Valve** template shows the feedback of a valve and, if available, the setpoint.



Use

Use on any page where a template of **Valve** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Three-point

Subelements:

Symbol name	PLC template	Required	Description
Anlg3Pnt	FB_BA_Analog3Pnt	x	
Pos	FB_BA_AV_Op	x	Feedback

Hierarchy:

- BaObject
 - Anlg3Pnt
 - Pos

Corresponds to the PLC template

- FB_BA_Vlv3pt

Analog value

Subelements:

Symbol name	PLC template	Required	Description
Fdb	FB_BA_AI_IO	x	Feedback
MdIt	FB_BA_AO_IO		Setpoint

Hierarchy:

- BaObject
 - Fdb
 - MdIt

Corresponds to the PLC template

- FB_BA_Vlv

Attribute

This template inherits from the [UilconFdbStp \[▸ 938\]](#) control. In addition, there is the following attribute.

Common

DisplayMode

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Templates.Universal.Valve.DisplayMode

Determines the icon to be displayed (see [DisplayMode \[▸ 909\]](#)).

6.2.1.1.4.4.1 Enumeration

DisplayMode

Schema for the [DisplayMode \[▸ 909\]](#) attribute of the [Valve \[▸ 908\]](#) template.

```
"TcHmi.BuildingAutomation.Templates.Universal.Valve.DisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Type",
  "type": "integer",
  "enum": [
    0,
    1,
    2,
    3,
    4,
    5,
    6,
    7
  ],
  "options": [
    {
      "label": "twoDirection_0",
      "value": 0
    },
    {
      "label": "twoDirection_1",
      "value": 1
    },
    {
      "label": "twoDirection_2",
      "value": 2
    },
    {
      "label": "twoDirection_3",
      "value": 3
    },
    {
      "label": "threeDirection_0",
      "value": 4
    },
    {
      "label": "threeDirection_1",
      "value": 5
    },
    {
      "label": "threeDirection_2",
      "value": 6
    },
    {
      "label": "threeDirection_3",
      "value": 7
    }
  ]
}
```

6.2.1.1.4.4.2 Types

SubBaTemplateDescription

Schema of a [Valve \[► 908\]](#) template for the BaTemplateDescription.

```
"TchHmi.BuildingAutomation.Controls.Plants.Valve.SubBaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "SubBaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "identifier",
      "displayName": "Identifier",
      "description": "Identifier of the sensor.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "subTemplate",
      "displayName": "Valve",
      "description": "The template description of the valve.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "identifier": {
      "type": "string"
    },
    "subTemplate": {
      "$ref": "tchmi:framework#/definitions/
TchHmi.BuildingAutomation.Controls.Plants.Valve.BaTemplateDescription"
    }
  }
}
```

BaTemplateDescription

Schema for the [BaTemplateDescription \[► 933\]](#) attribute of the [Valve \[► 908\]](#) template.

```
"TchHmi.BuildingAutomation.Controls.Plants.Valve.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "The feedback of the valve. (Default: Fdb / Anlg3Pnt::Pos, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "setpoint",
      "displayName": "Setpoint",
      "description": "The setpoint of the valve. (Default: Mdlt, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "command",
      "displayName": "Command",
      "description": "The command of the valve. (Default: Cmd, DataType: Binary)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "feedback": {
      "type": "string"
    },
    "setpoint": {
      "type": "string"
    },
    "command": {
      "type": "string"
    }
  }
}
```

```

}
}
}

```

6.2.1.1.4.5 Damper

The **Damper** template displays the damper position graphically and in text form.



Use

Use on any page where a template of the **Damper** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Two-point

Subelements:

Symbol name	PLC template	Required	Description
SwiCls	FB_BA_BI_IO	x	Switch close
SwiOpn	FB_BA_BI_IO	x	Switch open

Hierarchy:

- BaObject
 - SwiCls
 - SwiOpn

Corresponds to the PLC template:

- FB_BA_Dmp2P

Analog

Subelements:

Symbol name	PLC template	Required	Description
MdIt	FB_BA_AO_IO	x	Feedback

Hierarchy:

- BaObject
 - MdIt

Corresponds to the PLC template

- FB_BA_ActuatorAnalog

Attribute

This template inherits from the [UilconFdbStp](#) [► 938] control. In addition, there are the following attributes.

Common

DisplayMode

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Templates.Universal.Damper.DisplayMode

Determines the icon to be displayed (see [DisplayMode](#) [► 912]).

FlapPosition

tchmi:general#/definitions/Number

Position of the flaps in percent (0 is closed) (see [number](#)).

6.2.1.1.4.5.1 Enumeration

DisplayMode

Schema for the [DisplayMode](#) [► 912] attribute of the [Damper](#) [► 911] template.

```
"TcHmi.BuildingAutomation.Templates.Universal.Damper.DisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Type",
  "type": "integer",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "damper",
      "value": 0
    },
    {
      "label": "fireDamper",
      "value": 1
    },
    {
      "label": "shutOffValve",
      "value": 2
    }
  ]
}
```

6.2.1.1.4.5.2 Types

SubBaTemplateDescription

Schema of a [Damper](#) [► 911] template for the BaTemplateDescription.

```
"TcHmi.BuildingAutomation.Templates.Universal.Damper.SubTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "SubTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "identifier",
      "displayName": "Identifier",
      "description": "Identifier of the damper.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "subTemplate",
      "displayName": "Damper",
      "description": "The template description of the damper.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "identifier": {
```

```

        "type": "string"
      },
      "subTemplate": {
        "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Templates.Universal.Damper.TemplateDescription"
      }
    }
  }
}

```

TemplateDescription

Schema for the BaTemplateDescription [► 933] attribute of the Damper [► 911] template.

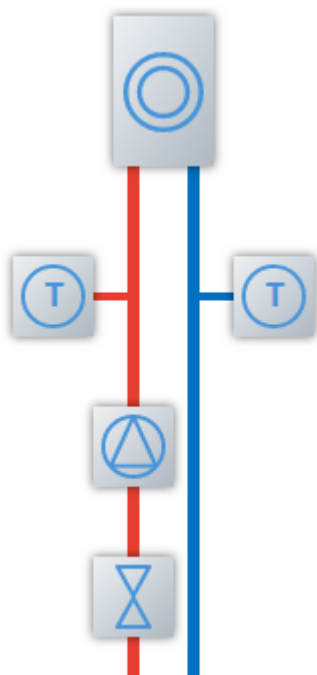
```

"TcHmi.BuildingAutomation.Controls.Plants.Damper.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "The feedback of the damper. (Default: Fdb, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "setpoint",
      "displayName": "Setpoint",
      "description": "The setpoint of the damper. (Default: Mdlt, DataType: Setpoint)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "command",
      "displayName": "Command",
      "description": "The command of the damper. (Default: Cmd, DataType: Binary)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "swiOpn",
      "displayName": "Switch open",
      "description": "The switch open of the damper. (Default: SwiOpn, DataType: Binary)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "swiCls",
      "displayName": "Switch close",
      "description": "The switch close of the damper. (Default: SwiCls, DataType: Binary)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "feedback": {
      "type": "string"
    },
    "setpoint": {
      "type": "string"
    },
    "command": {
      "type": "string"
    },
    "swiOpn": {
      "type": "string"
    },
    "swiCls": {
      "type": "string"
    }
  }
}

```

6.2.1.1.4.6 HeatingCircuit

The **FB_BA_H_HtgCir01** template represents a tankless water heater.



Use

Use on any page where a template of the **HtgCir01** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
Pu	FB_BA_Pu1st	Pump
Sp	FB_BA_H_HtgCir_Sp	Setpoint
SpFIWT	FB_BA_AV_Op	Setpoint flow temperature sensor
TFI	FB_BA_SensorAnalog_IO	Flow temperature sensor
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv	Valve

Hierarchy:

- BaObject
 - Pu
 - Sp
 - SpFIWT
 - TFI
 - TRt
 - Vlv

Corresponds to the PLC template:

- FB_BA_H_HtgCir01

Attributes

This template inherits from the template [FB_BA_AC_ColT_02](#) [► 915].

6.2.1.1.4.6.1 Types

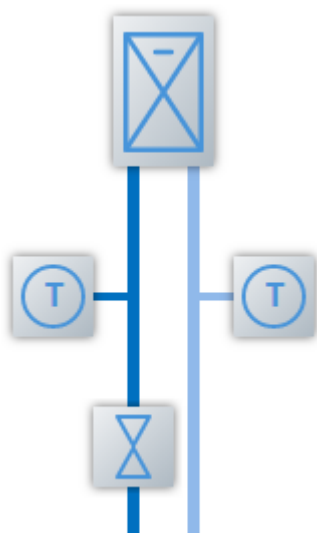
BaTemplateDescription

Schema for the `BaTemplateDescription` [► 933] attribute of the `HeatingCircuit` [► 913] template.

```
"Tchmi.BuildingAutomation.Controls.Plants.HeatingCircuit.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "tempFlow",
      "displayName": "Flow temperature sensor",
      "description": "The flow temperature sensor of the heating circuit. (Default: TFl, DataType: S
ensorAnalog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "tempReturn",
      "displayName": "Return temperature sensor",
      "description": "The return temperature sensor of the heating circuit. (Default: TRt, DataType: S
ensorAnalog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "valve",
      "displayName": "Valve",
      "description": "The valve of the heating circuit. (Default: Vlv, DataType: Valve)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "pump",
      "displayName": "Pump",
      "description": "The pump of the heating circuit. (Default: Pu, DataType: Pump)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "tempFlowSp",
      "displayName": "Flow temperature sensor setpoint",
      "description": "The flow temperature sensor setpoint of the heating circuit. (Default: Sp::SpF
lWT, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "tempFlow": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "tempReturn": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "valve": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.Valve.SubBaTemplateDescription"
    },
    "pump": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.Pump.SubBaTemplateDescription"
    },
    "tempFlowSp": {
      "type": "string"
    }
  }
}
```

6.2.1.1.4.7 Cooler

The `FB_BA_AC_CoIT_02` template represents a cooler.



Use

Use on any page where a template of the **FB_BA_CoIT_02** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
TFI	FB_BA_SensorAnalog_IO	Flow temperature sensor
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv	Valve

Hierarchy:

- BaObject
 - TFI
 - TRt
 - Vlv

Corresponds to the PLC template:

- FB_BA_AC_CoIT_02

Attributes

This template inherits from the [BaseTemplate](#) [► 932] control. In addition, there is the following attribute.

Valve

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Templates.Universal.Valve.DisplayMode
```

Determines the icon to be displayed (see [Valve](#) [► 908]).

6.2.1.1.4.7.1 Types

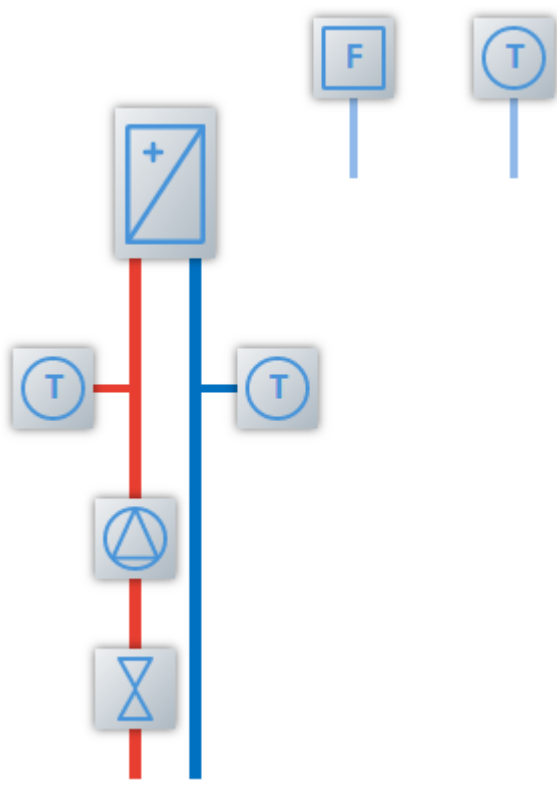
BaTemplateDescription

Schema for the attribute `BaTemplateDescription` [► 933] of the `Cooler` [► 915] template.

```
"Tchmi.BuildingAutomation.Controls.Plants.Cooler.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "tempFlow",
      "displayName": "Flow temperature sensor",
      "description": "The flow temperature sensor of the cooler. (Default: TFl, DataType: SensorAnal
og)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "tempReturn",
      "displayName": "Return temperature sensor",
      "description": "The return temperature sensor of the cooler. (Default: TRt, DataType: SensorAn
alog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "valve",
      "displayName": "Valve",
      "description": "The valve of the cooler. (Default: Vlv, DataType: Valve)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "tempFlow": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "tempReturn": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
    },
    "valve": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.Valve.SubBaTemplateDescription"
    }
  }
}
```

6.2.1.1.4.8 PreHeater

The `FB_BA_AC_PreHtr` template represents a preheater.



Use

Use on any page where a template of the **PreHtr** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
FrostThermostat	FB_BA_SensorBinary_IO	Frost protection thermostat
Pu	FB_BA_Pu1st	Pump
Sp	FB_BA_H_HtgCir_Sp	Setpoint
SpFIWT	FB_BA_AV_Op	Setpoint flow temperature sensor
TFI	FB_BA_SensorAnalog_IO	Flow temperature sensor
TFrost	FB_BA_AO_IO	Temperature frost
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv	Valve

Hierarchy:

- BaObject
 - FrostThermostat
 - Pu
 - Sp
 - SpFIWT
 - TFI
 - TFrost
 - TRt

- Vlv

Corresponds to the PLC template:

- FB_BA_AC_PreHtr

Attributes

This template inherits from the template [FB_BA_AC_CoIT_02 \[► 915\]](#).

6.2.1.1.4.8.1 Types

TemplateDescription

Schema for the attribute [BaTemplateDescription \[► 933\]](#) of the [PreHeater \[► 917\]](#) template.

```
"TcHmi.BuildingAutomation.Controls.Plants.PreHeater.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "tempFlow",
      "displayName": "Flow temperature sensor",
      "description": "The flow temperature sensor of the pre-
heater. (Default: TFl, DataType: SensorAnalog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "tempReturn",
      "displayName": "Return temperature sensor",
      "description": "The return temperature sensor of the pre-
heater. (Default: TRt, DataType: SensorAnalog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "valve",
      "displayName": "Valve",
      "description": "The valve of the pre-heater. (Default: Vlv, DataType: Valve)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "pump",
      "displayName": "Pump",
      "description": "The pump of the pre-heater. (Default: Pu, DataType: Pump)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "tempFlowSp",
      "displayName": "Flow temperature sensor setpoint",
      "description": "The flow temperature sensor setpoint of the heating circuit. (Default: Sp::SpF
lWT, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "frostThermostat",
      "displayName": "Frost thermostat",
      "description": "The frost thermostat of the pre-
heater. (Default: FrostThermostat, DataType: SensorBinary)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "tFrost",
      "displayName": "Temperature frost",
      "description": "The temperature frost of the pre-
heater. (Default: TFrost, DataType: SensorAnalog)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ]
}
```

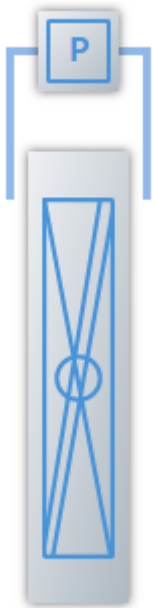
```

"properties": {
  "tempFlow": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
  },
  "tempReturn": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
  },
  "valve": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.Valve.SubBaTemplateDescription"
  },
  "pump": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.Pump.SubBaTemplateDescription"
  },
  "tempFlowSp": {
    "type": "string"
  },
  "frostThermostat": {
    "type": "string"
  },
  "tFrost": {
    "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.SubBaTemplateDescription"
  }
}
}

```

6.2.1.1.4.9 ErcRotation

The **FB_BA_AC_ErcRot_01** template represents an energy recovery.



Use

Use on any page where a template of the **ErcRot_01** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
DiffPrssSwi	FB_BA_SensorBinary_IO	Differential pressure
Mdlt	FB_BA_AO_IO	Motor feedback
Motor	FB_BA_MotMdlt	Motor

Hierarchy:

- BaObject
 - DiffPrssSwi
 - Motor
 - Mdlt

Corresponds to the PLC template:

- FB_BA_AC_ErcRot_01

Attributes

This template inherits from the [BaseTemplate \[► 932\]](#) control.

6.2.1.1.4.9.1 Types

BaTemplateDescription

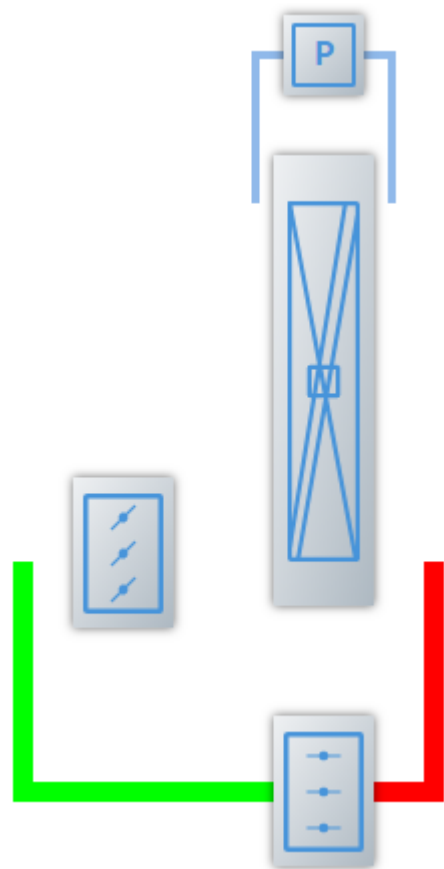
Schema for the attribute [BaTemplateDescription \[► 933\]](#) of the [ErcRotation \[► 920\]](#) template.

```
"TcHmi.BuildingAutomation.Controls.Plants.ErcRotation.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "diffPrssSwi",
      "displayName": "Differential pressure",
      "description": "The differential pressure of the energy recovery. (Default: DiffPrssSwi, DataType: SensorBinary)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "The feedback of the energy recovery. (Default: Motor::Fdb, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "setpoint",
      "displayName": "Setpoint",
      "description": "The setpoint of the energy recovery. (Default: Motor::Mdlt, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "command",
      "displayName": "Command",
      "description": "The command of the energy recovery. (Default: Motor::Cmd, DataType: Binary)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "diffPrssSwi": {
      "type": "string"
    },
    "feedback": {
      "type": "string"
    }
  }
}
```

```
"setpoint": {
  "type": "string"
},
"command": {
  "type": "string"
}
}
```

6.2.1.1.4.10 ErcPlate

The **FB_BA_AC_ErcPI_02** template represents an energy recovery.



Use

Use on any page where a template of the **ErcPI_02** type is to be displayed.

Compatibility

The [BaTemplateDescription](#) [► 40] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
ByDmp	FB_BA_ActuatorAnalog	Bypass damper
DiffPrssSwi	FB_BA_SensorBinary_IO	Differential pressure
Dmp	FB_BA_ActuatorAnalog	Damper

Hierarchy:

- BaObject
 - ByDmp
 - DiffPrssSwi

- Dmp

Corresponds to the PLC templates:

- FB_BA_AC_ErcPI_02

Attributes

This template inherits from the [BaseTemplate \[► 932\]](#) control.

6.2.1.1.4.10.1 Types

BaTemplateDescription

Schema for the attribute [BaTemplateDescription \[► 933\]](#) of the [ErcPlate \[► 922\]](#) template.

```
"Tchmi.BuildingAutomation.Controls.Plants.ErcPlate.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaTemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "damper",
      "displayName": "Damper",
      "description": "The damper of the energy recovery. (Default: Dmp, DataType: Damper)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "damperBy",
      "displayName": "Damper Bypass",
      "description": "The damper bypass of the energy recovery. (Default: ByDmp, DataType: Damper)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "diffPrssSwi",
      "displayName": "Differential pressure",
      "description": "The differential pressure of the energy recovery. (Default: DiffPrssSwi, DataT
type: SensorBinary)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "damper": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.Damper.SubBaTemplateDescription"
    },
    "damperBy": {
      "$ref": "tchmi:framework#/definitions/
Tchmi.BuildingAutomation.Controls.Plants.Damper.SubBaTemplateDescription"
    },
    "diffPrssSwi": {
      "type": "string"
    }
  }
}
```

6.2.1.1.5 System

6.2.1.1.5.1 BaseControl

The BaseControl is the basis for various controls, it contains methods and attributes that other controls also need. This prevents redundant implementations.

Use

This is only used for inheritance and is therefore not available in the toolbox.

Features

Implements functionalities that run in the background and take over the management of various tasks. These include, for example:

- Busy handling
- log out various watches

Attributes

The control inherits from [TcHmiControl](#) and thus has the same attributes. In addition, there are the following attributes.

Common

ReadOnly

tchmi:general#/definitions/Boolean

Determines whether the user has read-only or write permission (see [boolean](#)).



The attribute is not applicable to all controls.

Layout

ContentPadding

tchmi:framework#/definitions/Padding

Sets the padding for the content of the control (see [Padding](#)).



The attribute is not applicable to all controls.

6.2.1.1.5.1.1 Enumerations

DateTimeChoice

Schema for a selection of time and date.

```
"TcHmi.BuildingAutomation.DateTimeChoice": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "DateTimeChoice",
  "type": "integer",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "Date",
      "value": 0
    },
    {
      "label": "Time",
      "value": 1
    },
    {
      "label": "DateTime",
      "value": 2
    }
  ]
}
```

DialogWindowButtons

Schema for a selection of dialog buttons.

```
"TcHmi.BuildingAutomation.Controls.DialogWindowButtons": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "DialogWindowButtons",
  "type": "string",
  "enum": [
    0,
    1
  ],
  "options": [
    {
      "label": "Cancel",
      "value": 0
    },
    {
      "label": "CancelSize",
      "value": 1
    },
    {
      "label": "OkCancel",
      "value": 2
    },
    {
      "label": "OkCancelSize",
      "value": 3
    }
  ]
}
```

Position

Schema for a selection of positions.

```
"TcHmi.BuildingAutomation.Position": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "Position",
  "type": "string",
  "enum": [
    0,
    1,
    2,
    3
  ],
  "options": [
    {
      "label": "left",
      "value": 0
    },
    {
      "label": "top",
      "value": 1
    },
    {
      "label": "right",
      "value": 2
    },
    {
      "label": "bottom",
      "value": 3
    }
  ]
}
```

Direction

Schema for a selection of directions.

```
"TcHmi.BuildingAutomation.Direction": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1
  ],
  "options": [
```

```

    {
      "label": "left",
      "value": 0
    },
    {
      "label": "right",
      "value": 1
    }
  ]
}

```

Orientation

Schema for a selection of orientations.

```

"Tchmi.BuildingAutomation.Orientation": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1
  ],
  "options": [
    {
      "label": "Vertical",
      "value": 0
    },
    {
      "label": "Horizontal",
      "value": 1
    }
  ]
}

```

6.2.1.1.5.1.2 Types

FourSidedColor

Schema for an attribute to define a color per page.

```

"Tchmi.BuildingAutomation.FourSidedColor": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "title": "FourSidedColor",
  "propertiesMeta": [
    {
      "name": "left",
      "displayName": "Left",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "top",
      "displayName": "Top",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "right",
      "displayName": "Right",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "bottom",
      "displayName": "bottom",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "left": {
      "$ref": "tchmi:framework#/definitions/SolidColor"
    },
    "top": {
      "$ref": "tchmi:framework#/definitions/SolidColor"
    }
  }
}

```

```

    "right": {
      "$ref": "tchmi:framework#/definitions/SolidColor"
    },
    "bottom": {
      "$ref": "tchmi:framework#/definitions/SolidColor"
    }
  }
}

```

NumberOrBoolean

Schema for attributes that can have as data type number or boolean.

```

"TcHmi.BuildingAutomation.NumberOrBoolean": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": [ "number", "boolean" ]
}

```

ICoordinateXY

Schema for an xy-coordinate pair.

```

"TcHmi.BuildingAutomation.ICoordinateXY": {
  "type": "object",
  "title": "ICoordinateXY",
  "description": "Defines a XY coordinate",
  "engineeringColumns": [
    "x",
    "y"
  ],
  "propertiesMeta": [
    {
      "name": "x",
      "category": "Common",
      "displayName": "X Value"
    },
    {
      "name": "y",
      "category": "Common",
      "displayName": "Y Value"
    }
  ],
  "properties": {
    "x": {
      "type": "number"
    },
    "y": {
      "type": "number"
    }
  },
  "required": [ "x", "y" ]
}

```

DialogWindowOptions

Schema for options of a dialog box.

```

"TcHmi.BuildingAutomation.DialogWindowOptions": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "content",
      "displayName": "Content",
      "description": "The content that should be displayed in the dialog window.",
      "defaultValue": null
    },
    {
      "name": "buttons",
      "displayName": "Buttons",
      "description": "Buttons for the dialog.",
      "defaultValue": 0
    },
    {
      "name": "modal",
      "displayName": "Modal",
      "description": "Selects if the dialog is opened modal or not.",
      "defaultValue": true
    }
  ],

```

```

    {
      "name": "headline",
      "displayName": "Headline",
      "description": "The headline of the dialog.",
      "defaultValue": "Dialog headline"
    },
    {
      "name": "width",
      "displayName": "Width",
      "defaultValue": 1000,
      "defaultValueInternal": 1000
    },
    {
      "name": "widthUnit",
      "displayName": "Width unit",
      "refTo": "width",
      "defaultValue": "px",
      "defaultValueInternal": "px"
    },
    {
      "name": "height",
      "displayName": "Height",
      "defaultValue": 500,
      "defaultValueInternal": 500
    },
    {
      "name": "heightUnit",
      "displayName": "Height unit",
      "refTo": "height",
      "defaultValue": "px",
      "defaultValueInternal": "px"
    }
  ],
  "properties": {
    "content": {
      "$ref": "tchmi:framework#/definitions/ContentPath"
    },
    "buttons": {
      "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.DialogWindowButtons"
    },
    "modal": {
      "type": "boolean"
    },
    "headline": {
      "type": "string"
    },
    "width": {
      "type": "number"
    },
    "widthUnit": {
      "$ref": "tchmi:framework#/definitions/MeasurementUnit"
    },
    "height": {
      "type": "number"
    },
    "heightUnit": {
      "$ref": "tchmi:framework#/definitions/MeasurementUnit"
    }
  }
}

```

UserContents

Schema for user name and content for an editor entry.

```

"Tchmi.BuildingAutomation.UserContents": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "array",
  "items": {
    "type": "object",
    "title": "UserContent",
    "description": "Defines a relation between a user and a content.",
    "engineeringColumns": [
      "userName",
      "content"
    ],
    "propertiesMeta": [
      {
        "name": "userName",

```

```

        "displayName": "UserName ",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "content",
        "displayName": "Content",
        "defaultValue": null,
        "defaultValueInternal": null
    }
],
"properties": {
    "userName": {
        "type": "string"
    },
    "content": {
        "$ref": "tchmi:framework#/definitions/ContentPath"
    }
},
"required": [ "userName", "content" ]
}
}

```

6.2.1.1.5.1.2.1 DateTime

BaDateTime

Schema for time and date in BA format.

```

"TchHmi.BuildingAutomation.BaDateTime": {
    "$schema": "http://json-schema.org/draft-04/schema",
    "type": "object",
    "propertiesMeta": [
        {
            "name": "stDate",
            "defaultValue": null,
            "defaultValueInternal": null
        },
        {
            "name": "stTime",
            "defaultValue": null,
            "defaultValueInternal": null
        }
    ],
    "properties": {
        "stDate": {
            "$ref": "tchmi:framework#/definitions/TchHmi.BuildingAutomation.BaDate"
        },
        "stTime": {
            "$ref": "tchmi:framework#/definitions/TchHmi.BuildingAutomation.BaTime"
        }
    }
}

```

BaDate

Schema for a date in BA format.

```

"TchHmi.BuildingAutomation.BaDate": {
    "$schema": "http://json-schema.org/draft-04/schema",
    "type": "object",
    "propertiesMeta": [
        {
            "name": "eDayOfWeek",
            "defaultValue": null,
            "defaultValueInternal": null
        },
        {
            "name": "eMonth",
            "defaultValue": null,
            "defaultValueInternal": null
        },
        {
            "name": "nDay",
            "defaultValue": null,
            "defaultValueInternal": null
        }
    ],

```

```

    {
      "name": "nYear",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "eDayOfWeek": {
      "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.Weekday"
    },
    "eMonth": {
      "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.Month"
    },
    "nDay": {
      "type": "number"
    },
    "nYear": {
      "type": "number"
    }
  }
}

```

BaTime

Schema for a time in BA format.

```

"TcHmi.BuildingAutomation.BaTime": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "nHour",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "nMinute",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "nSecond",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "nHour": {
      "type": "number"
    },
    "nMinute": {
      "type": "number"
    },
    "nSecond": {
      "type": "number"
    }
  }
}

```

Month

Enumeration for a date specification in BACnet format.

```

"TcHmi.BuildingAutomation.Month": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    255,
    1,
    2,
    3,
    4,
    5,
    6,
    7,
    8,
    9,
    10,

```

```

    11,
    12
  ],
  "options": [
    {
      "label": "Unspecified",
      "value": 255
    },
    {
      "label": "eJanuary",
      "value": 1
    },
    {
      "label": "eFebruary",
      "value": 2
    },
    {
      "label": "eMarch",
      "value": 3
    },
    {
      "label": "eApril",
      "value": 4
    },
    {
      "label": "eMay",
      "value": 5
    },
    {
      "label": "eJune",
      "value": 6
    },
    {
      "label": "eJuly",
      "value": 7
    },
    {
      "label": "eAugust",
      "value": 8
    },
    {
      "label": "eSeptember",
      "value": 9
    },
    {
      "label": "eOctober",
      "value": 10
    },
    {
      "label": "eNovember",
      "value": 11
    },
    {
      "label": "eDecember",
      "value": 12
    }
  ]
}

```

Weekday

Schema for specifying a day of the week in BACnet format.

```

"TcHmi.BuildingAutomation.Weekday": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    255,
    1,
    2,
    3,
    4,
    5,
    6,
    7
  ],
  "options": [
    {
      "label": "Unspecified",

```



```

    "value": 255
  },
  {
    "label": "eMonday",
    "value": 1
  },
  {
    "label": "eTuesday",
    "value": 2
  },
  {
    "label": "eWednesday",
    "value": 3
  },
  {
    "label": "eThursday",
    "value": 4
  },
  {
    "label": "eFriday",
    "value": 5
  },
  {
    "label": "eSaturday",
    "value": 6
  },
  {
    "label": "eSunday",
    "value": 7
  }
]
}

```

6.2.1.1.5.2 BaseTemplate

The **BaseTemplate** is the basis for all more sophisticated template controls (e.g. PreHtr, HtgCir), which are more than just a Uilcon. It provides methods and attributes to prevent redundant implementations.

Use

This is only used for inheritance and is therefore not available in the toolbox.

Features

Enables the use of [BaTemplates](#) [► 39] for all inheriting controls.

Attributes

The control inherits from [TcHmiControl](#) and thus has the same attributes. In addition, there are the following attributes.

BA

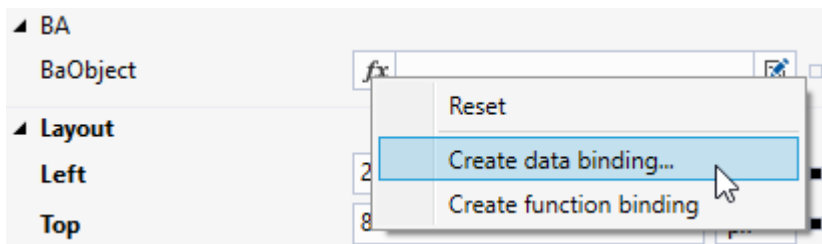
BAObject

tchmi:framework#/definitions/Symbol

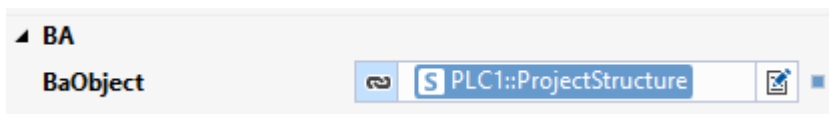
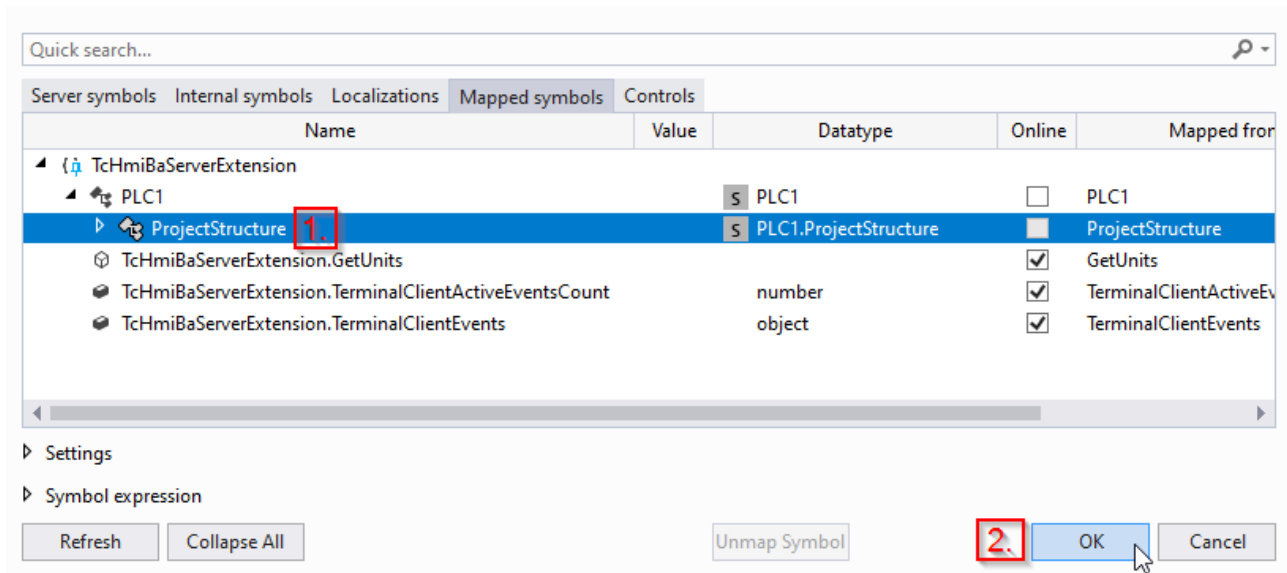
To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.



Select value for ProjectNavigationTextual.BaObject



BaTemplateDescription

tchmi:general#/definitions/Object

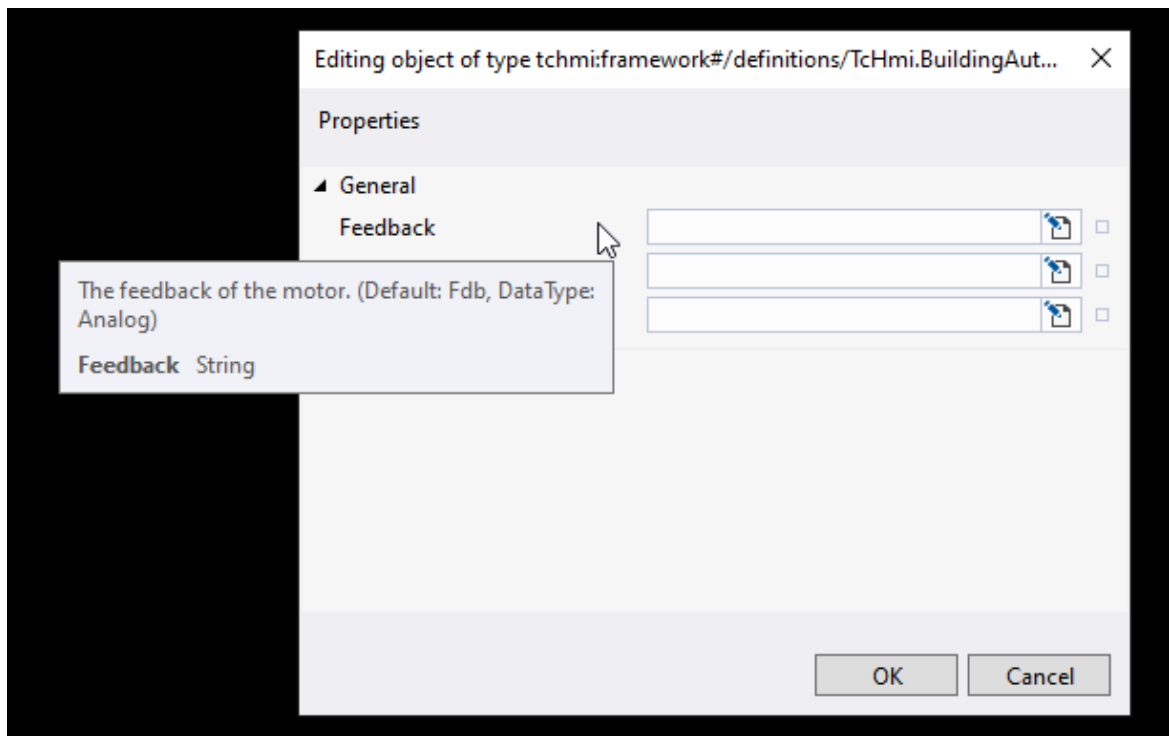
Allows editing the BaTemplateDescription (see [object](#)).



More detailed information about the *BaTemplate* can be found [here](#) [► 39].



The default values of the BaTemplateDescription, as well as the expected data types can be found in the tooltip of the dialog for setting the BaTemplateDescription:



6.2.1.1.5.3 BaseRoomControl

The **BaseRoomControl** is the basis for all room controls (e.g. Light, Sunblind, etc.). It provides methods and attributes to prevent redundant implementations.

Use

This is only used for inheritance and is therefore not available in the toolbox.

Features

Enables the use of [BaTemplates](#) [► 39] for all inheriting controls.

Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

BA

BaTemplateDescription

tchmi:general#/definitions/Object

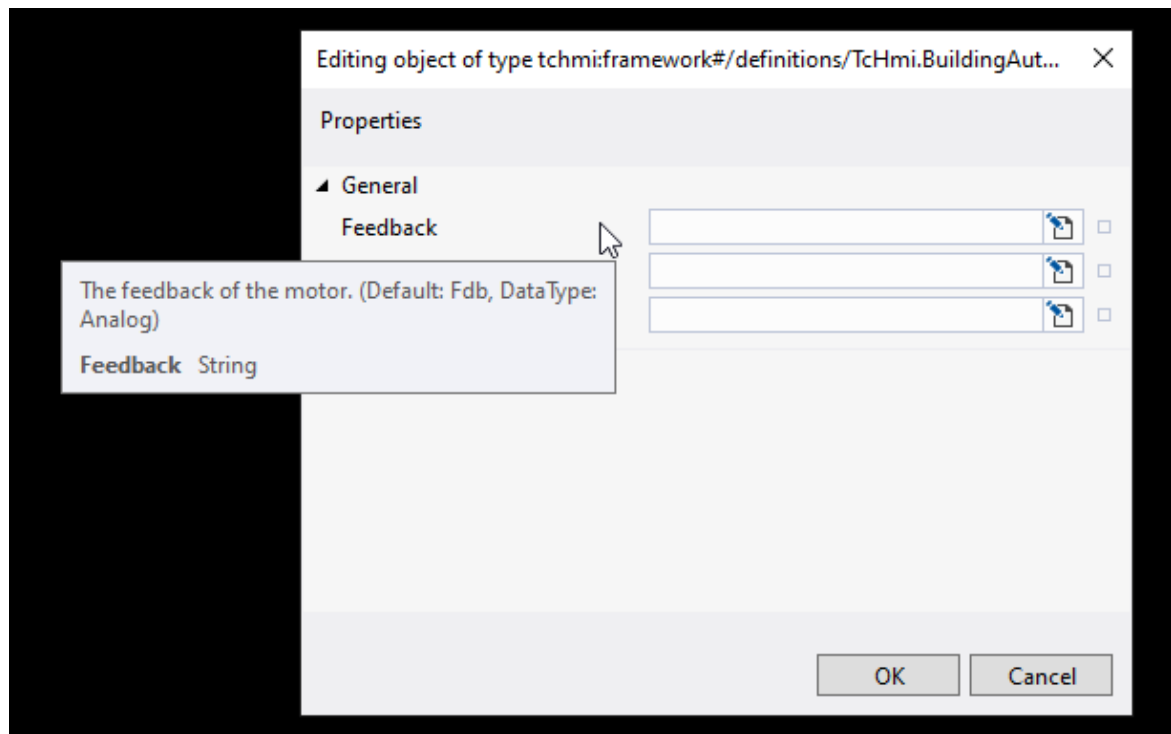
Allows editing the BaTemplateDescription (see [object](#)).



More detailed information about the *BaTemplate* can be found [here](#) [► 39].



The default values of the BaTemplateDescription, as well as the expected data types can be found in the tooltip of the dialog for setting the BaTemplateDescription:



BaData

BaInterface

tchmi:framework#/definitions/Symbol

Allows linking a symbol (see [symbol](#)) that satisfies the [BaInterface](#) [► 999] of the control. With this symbol all necessary data points of the control can be linked with only one binding.

6.2.1.1.5.4 TextControl

The **TextControl** offers various attributes, all of which are valid for text manipulation.

Use

This is only used for inheritance and is therefore not available in the toolbox.

Features

The following text manipulations are possible:

- change position horizontally and vertically
- influence font, size and thickness
- add various decorations to the text (e.g. underlined)
- define how the text should be displayed if the available space is not sufficient

Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

Colors

TextColor

```
tchmi:framework#/definitions/SolidColor
```

Color of texts (see [SolidColor](#)).

TextDecorationColor

```
tchmi:framework#/definitions/SolidColor
```

Color of text decorations (see [SolidColor](#)).

Text

TextVerticalAlignment

```
tchmi:framework#/definitions/VerticalAlignment
```

Vertical alignment of texts (see [VerticalAlignment](#)).

TextHorizontalAlignment

```
tchmi:framework#/definitions/HorizontalAlignment
```

Horizontal alignment of texts (see [HorizontalAlignment](#)).

TextFontSize

```
tchmi:framework#/definitions/MeasurementValue
```

Font size of texts. Percentage values are relative to the font size of the parent element (see [MeasurementValue](#)).

TextFontSizeUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit for the font size of texts. Can be absolute (px) or relative (%) (see [MeasurementUnit](#)).

TextFontFamily

```
tchmi:framework#/definitions/FontFamily
```

Font of texts (see [FontFamily](#)).

TextFontStyle

```
tchmi:framework#/definitions/FontStyle
```

Font style of texts (see [FontStyle](#)).

TextFontWeight

```
tchmi:framework#/definitions/FontWeight
```

Font weight of texts (see [FontWeight](#)).

TextDecorationLine

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationLine
```

Position of the text decoration (see [TextDecorationLine](#) [► 937]).

TextDecorationStyle

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationStyle
```

Text decoration style (see [TextDecorationStyle](#) [► 937])

UserSelect

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.UserSelect
```

Behavior when the text is selected by a user (see [UserSelect](#) [► 937]).

TextOverflow

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextOverflow
```

Determines how to display text that is wider than the control (see [TextOverflow](#) [► 937]).

6.2.1.1.5.4.1 Enumerations**TextDecorationLine**

Schema for the TextDecorationLine attribute of the [TextControl](#) [► 935] control.

```
"TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationLine": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [ "none", "underline", "overline", "line-through", "initial", "inherit" ]
}
```

TextDecorationStyle

Schema for the TextDecorationStyle attribute of the [TextControl](#) [► 935] control.

```
"TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationStyle": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [ "solid", "double", "dotted", "dashed", "wavy", "initial", "inherit" ]
}
```

UserSelect

Schema for the UserSelect attribute of the [TextControl](#) [► 935] control.

```
"TcHmi.BuildingAutomation.Controls.TextControl.UserSelect": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [ "auto", "none", "text", "all" ]
}
```

TextOverflow

Schema for the TextOverflow attribute of the [TextControl](#) [► 935] control.

Name	Description
None	Text is not truncated and displayed beyond the control.
Ellipsis	The text is truncated (e.g. "A long te...")
marquee	Text moves back and forth in the available space (marquee).

```
"TcHmi.BuildingAutomation.Controls.TextControl.TextOverflow": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "integer",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "none",
      "value": 0
    },
    {
      "label": "ellipsis",
      "value": 1
    },
    {
      "label": "marquee",
      "value": 2
    }
  ]
}
```

```
}  
]  
}
```

6.2.1.1.5.5 UilconFdbStp

The UilconFdbStp is a specialized form of the [Uilcon \[► 859\]](#) and serves as a basis for simple TcHmiBa controls (e.g. Valve, SensorAnalog). It provides methods and attributes to prevent redundant implementations.

Use

This is only used for inheritance and is therefore not available in the toolbox.

Features

Facilitates the handling of the setpoint and actual value.

Attributes

The control inherits from [Uilcon \[► 859\]](#) and thus has the same attributes. In addition, there are the following attributes.

BA

Setpoint

tchmi:framework#/definitions/Symbol

BaObject for the setpoint (see [symbol](#)).

Common

DisplayMode

tchmi:general#/definitions/Number

Determines the icon to be displayed (see [Number](#)).

ShowFeedback

tchmi:general#/definitions/Boolean

Determines whether the feedback is visible (see [Boolean](#)).

ShowSetpoint

tchmi:general#/definitions/Boolean

Determines whether the setpoint is visible (see [Boolean](#)).

ShowDisplays

tchmi:general#/definitions/Boolean

Determines whether the setpoint and actual value are visible (see [Boolean](#)).

DisplayPosition

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.UiIcon.DisplayPosition

Position of setpoint and actual value (see [DisplayPosition \[► 864\]](#)).

6.2.1.1.6 RoomAutomation

6.2.1.1.6.1 HeatingCooling

The **HeatingCooling** control displays the operation mode, set and actual temperature in a room and can change the setpoint.



Use

Use on any page where controls are needed to control air conditioning systems.

Features

Operation modes

It is displayed whether the system is currently:

- in heating mode,



- in cooling mode or



- inactive.



If the system is inactive, the last active state is always displayed.

Displays

The user level determines the information available in the display.

For users with Advanced level or higher, the current values for temperature and setpoint are visible.

21.0 °C → 20.0 °C

For Default users, only the current setpoint change is displayed.

2.0 °K

Operation

Clicking on the **HeatingCooling** control changes the visibility of the menu for setting the temperature.



Attributes

The control inherits from [BaseRoomControl](#) [► 934] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

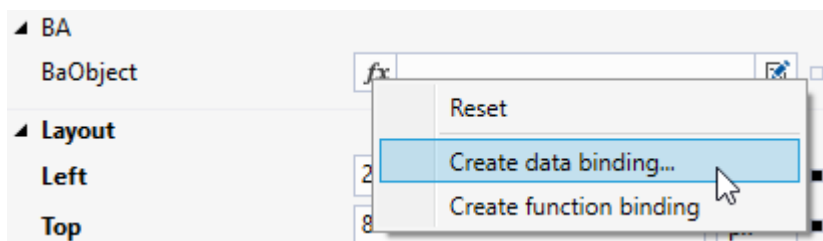
BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

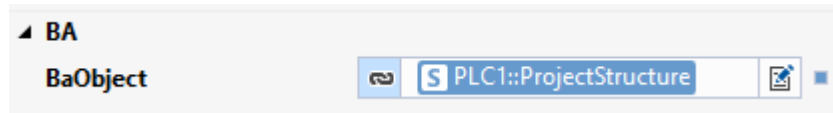


Select value for ProjectNavigationTextual.BaObject



Server symbols	Internal symbols	Localizations	Mapped symbols	Controls
Name	Value	Datatype	Online	Mapped from
- TcHmiBaServerExtension - PLC1 ProjectStructure 1. - TcHmiBaServerExtension.GetUnits - TcHmiBaServerExtension.TerminalClientActiveEventsCount - TcHmiBaServerExtension.TerminalClientEvents	- PLC1 - PLC1.ProjectStructure	- number - object	☐ ☒ ☒ ☒	- PLC1 - ProjectStructure - GetUnits - TerminalClientActiveEv - TerminalClientEvents

Settings
 Symbol expression



BaTemplateDescription

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.AirConditioning.TemplateDescription

(see [TemplateDescription](#) [► 944])

tchmi:general#/definitions/Object

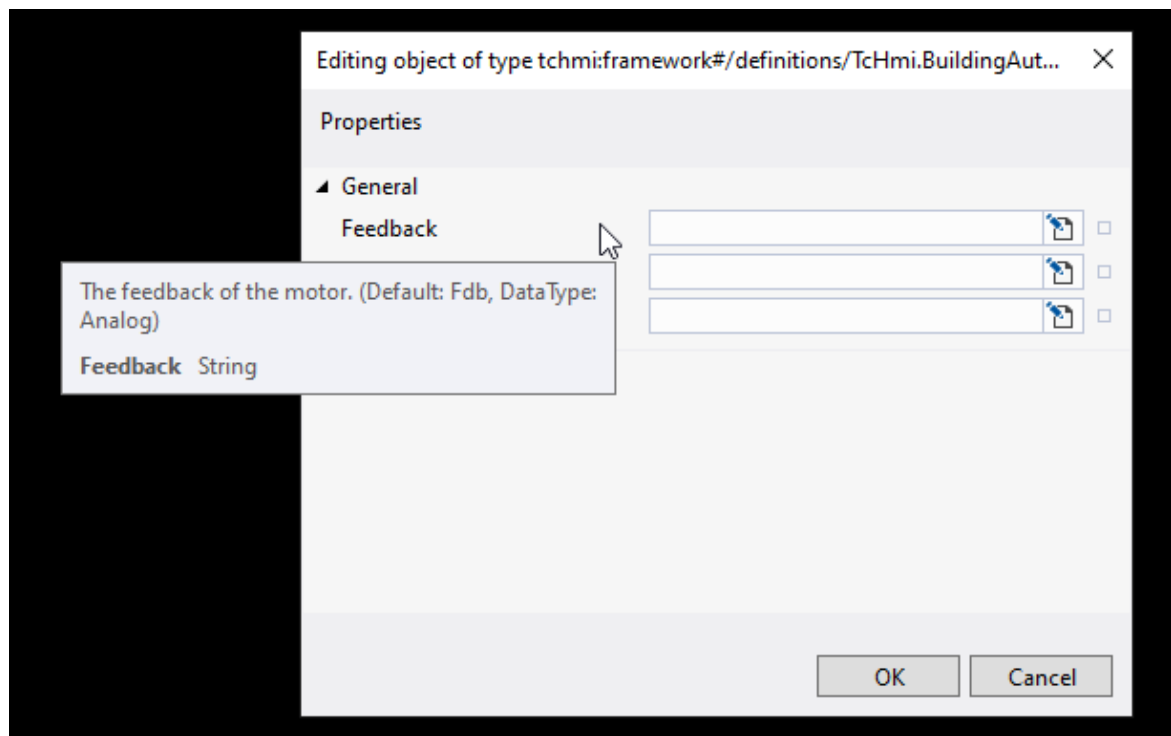
Allows editing the BaTemplateDescription (see [object](#)).



More detailed information about the *BaTemplate* can be found [here](#) [► 39].



The default values of the BaTemplateDescription, as well as the expected data types can be found in the tooltip of the dialog for setting the BaTemplateDescription:



Common

ShowTemperatures

tchmi:general#/definitions/Boolean

If TRUE, then the temperatures are displayed (see [Boolean](#)).

BaData

BaInterfaceSymbolNames

tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.AirConditioning.BaInterfaceSymbolNames

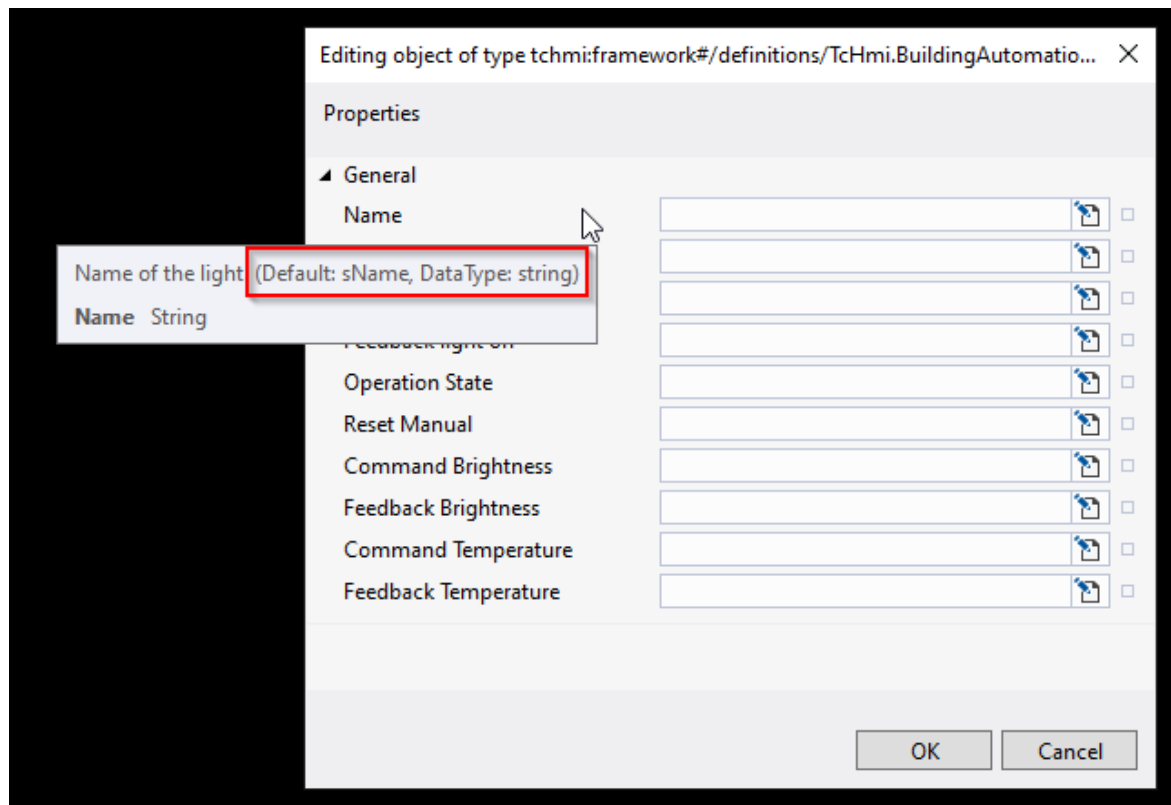
(see [BaInterfaceSymbolNames](#) [► 943])

tchmi:general#/definitions/Object

Allows editing the `BalInterfaceSymbolNames` [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the `BalInterfaceSymbolNames` are overwritten by all controls of a type.

Temperature

CurrentTemperature

`tchmi:general#/definitions/Number`

Current measured room temperature (see [Number](#)).

TempAdjust

`tchmi:general#/definitions/Number`

Current temperature adjustment (see [Number](#)).

TempAdjustFeedback

`tchmi:general#/definitions/Number`

Feedback for the current temperature adjustment (see [Number](#)).

TempAdjustRange

`tchmi:general#/definitions/Number`

Determines the scope of the temperature adjustment (see [Number](#)).

HeatingSetpoint

`tchmi:general#/definitions/Number`

Current setpoint for heating mode (see [Number](#)).

CoolingSetpoint

tchmi:general#/definitions/Number

Current setpoint for cooling mode (see [Number](#)).

HeatingActive

tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean

If TRUE or greater than 0, then heating is displayed (see [NumberOrBoolean](#) [[► 927](#)]).

CoolingActive

tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean

If TRUE or greater than 0, then cooling is displayed (see [NumberOrBoolean](#) [[► 927](#)]).

Unit

tchmi:general#/definitions/String

Unit for displaying temperatures (see [String](#)).

Events

Event	Description
onTempAdjustChanged	Triggered when the user changes the temperature adjustment.

6.2.1.1.6.1.1 Types

BaInterfaceSymbolNames

Schema for the attribute [BaInterfaceSymbolNames](#) [[► 941](#)] of the [HeatingCooling](#) [[► 939](#)] control.

```
"TcHmi.BuildingAutomation.Controls.AirConditioning.BaTemplateDescription.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "name",
      "displayName": "Name",
      "description": "Name of the light the light. (Default: sName, DataType: string)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "roomTemp",
      "displayName": "Current temperature",
      "description": "The current temperature. (Default: RoomTemp, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "roomTempAdj",
      "displayName": "Temperature adjustment",
      "description": "The adjustment of the temperature setpoint. (Default: RoomTempAdj, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "heatingSetpoint",
      "displayName": "Heating setpoint",
      "description": "The setpoint for heating. (Default: HeatingSetpoint, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
}
```

```

    {
      "name": "heatingActive",
      "displayName": "Heating active",
      "description": "Defines if the heating is active or not. (Default: HeatingActive, DataType: boolean / number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "coolingSetpoint",
      "displayName": "Cooling setpoint",
      "description": "The setpoint for cooling. (Default: CoolingSetpoint, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "coolingActive",
      "displayName": "Cooling active",
      "description": "Defines if the cooling is active or not. (Default: CoolingActive, DataType: boolean / number)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "name": {
      "type": "string"
    },
    "roomTemp": {
      "type": "string"
    },
    "roomTempAdj": {
      "type": "string"
    },
    "heatingSetpoint": {
      "type": "string"
    },
    "heatingActive": {
      "type": "string"
    },
    "coolingSetpoint": {
      "type": "string"
    },
    "coolingActive": {
      "type": "string"
    }
  }
}

```

BaTemplateDescription

Schema for the [BaTemplateDescription](#) [► 941] attribute of the [HeatingCooling](#) [► 939] control.

```

"TcHmi.BuildingAutomation.Controls.AirConditioning.BaTemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "TemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "currentTemperature",
      "displayName": "Current temperature",
      "description": "The current temperature. (Default: RoomTempSen1, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "temperatureAdjust",
      "displayName": "Temperature adjustment",
      "description": "The adjustment of the temperature setpoint. (Default: RoomTempAdj, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "controlHeating",
      "displayName": "Heating controller",
      "description": "The controller for heating mode. (Default: HeatCoolPID::ControlHeating, DataType: Loop)",
      "defaultValue": null,

```

```

        "defaultValueInternal": null
    },
    {
        "name": "controlCooling",
        "displayName": "Cooling controller",
        "description": "The controller for cooling mode. (Default: HeatCoolPID::ControlCooling, DataTy
        pe: Loop)",
        "defaultValue": null,
        "defaultValueInternal": null
    }
],
"properties": {
    "currentTemperature": {
        "type": "string"
    },
    "temperatureAdjust": {
        "type": "string"
    },
    "controlHeating": {
        "type": "string"
    },
    "controlCooling": {
        "type": "string"
    }
}
}
}

```

6.2.1.1.6.2 Light

The **Light** control is used to display and control the brightness of a light source.



Use

Use on any page where controls are needed to control light sources.

Features

Brightness specification

The brightness can be adjusted using an analog (dimnable) or binary (switchable) value. Buttons with predefined brightness values are available for dimmable lamps.



The display of these buttons can be set as follows:

```

TcHmi.EventProvider.register('onInitialized', function (e, data) {
    e.destroy();
    TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.ShowQuickLinks = true;
}).

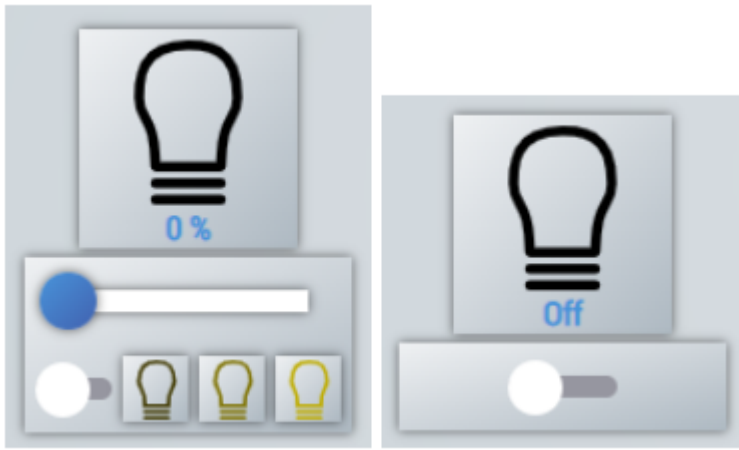
```

Operation modes

Display of different modes via the attribute `OperationState`.

Operation

Clicking on the **Light** control changes the visibility of the menu for adjusting the brightness.



Attributes

The control inherits from [BaseRoomControl](#) [► 934] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

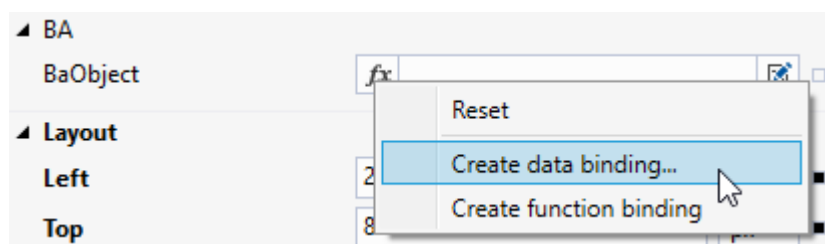
BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.



Select value for ProjectNavigationTextual.BaObject

✕

Quick search...

Server symbols Internal symbols Localizations **Mapped symbols** Controls

Name	Value	Datatype	Online	Mapped from
└─ TcHmiBaServerExtension				
└─ PLC1	S PLC1		☐	PLC1
└─ ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
└─ TcHmiBaServerExtension.GetUnits			☒	GetUnits
└─ TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
└─ TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Settings

Symbol expression

Refresh Collapse All Unmap Symbol

2. OK Cancel

BA

BaObject S PLC1::ProjectStructure

BaTemplateDescription

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Light.TemplateDescription

(see [TemplateDescription](#) [► 953])

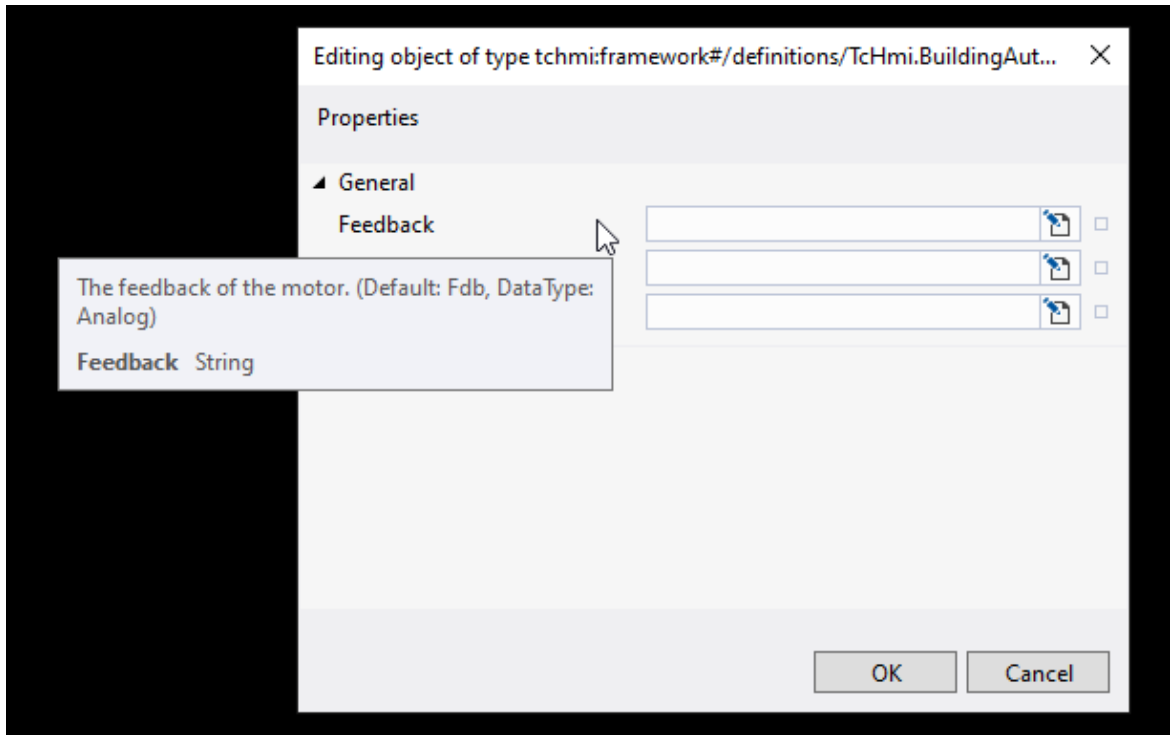
tchmi:general#/definitions/Object

Allows editing the BaTemplateDescription (see [object](#)).



More detailed information about the *BaTemplate* can be found [here](#) [► 39].

i The default values of the BaTemplateDescription, as well as the expected data types can be found in the tooltip of the dialog for setting the BaTemplateDescription:



Common

DisplayMode

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.DisplayMode

Defines the display mode of the lamp (see [DisplayMode](#) [► 951]).

Name	Representation
lightBulb	
lightBulbFilled	
filles	

ShowValue

tchmi:general#/definitions/Boolean

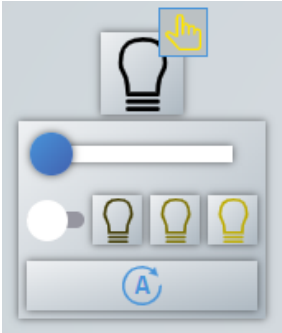
If TRUE, the brightness value is displayed (see [Boolean](#)).

BaData

OperationState

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.OperationState

Defines the display mode for the lamp (see [OperationState](#) [► 951]).

Name	Description	Representation
autoActive	Automatic for the lighting is active.	
autoInactive	Automatic for the lighting is switched off or not available.	
hand	Automatic was overwritten by a manual intervention. Display of a button for resetting the "hand" mode. When pressed, the <i>onHandModeReset</i> event is triggered.	

BaInterfaceSymbolNames

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Light.BaInterfaceSymbolNames

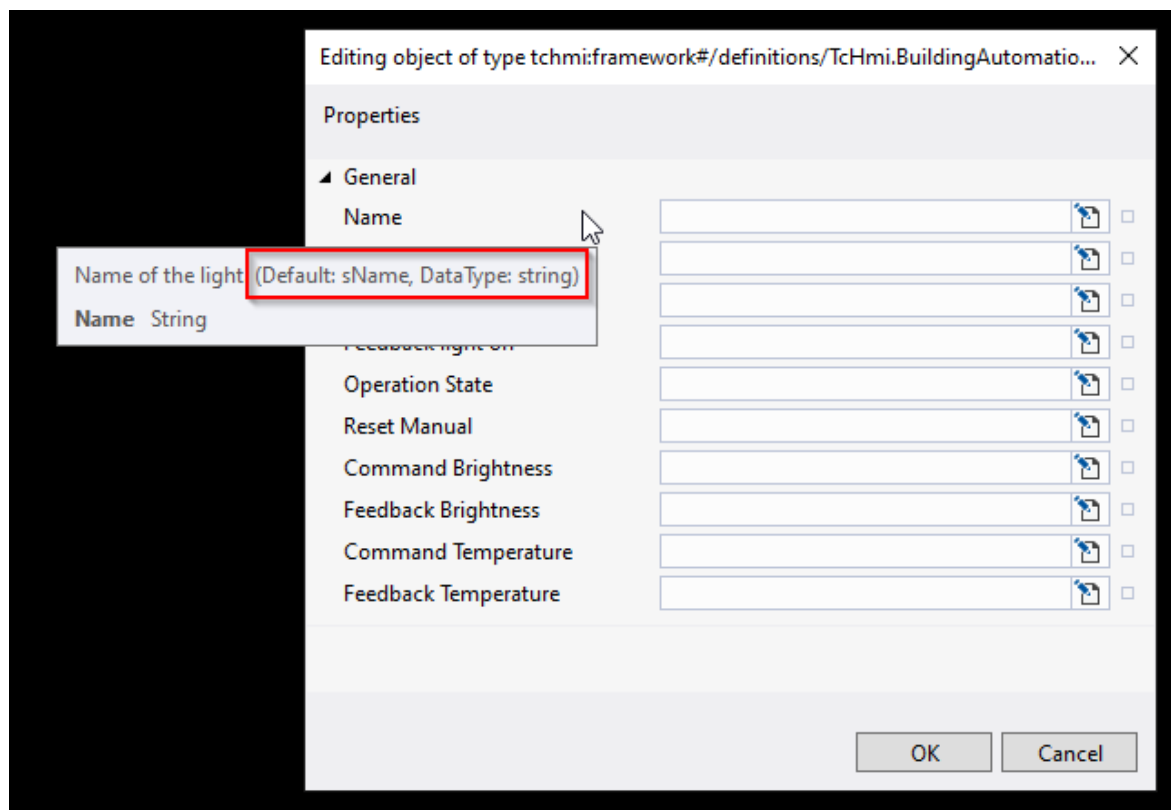
(see [BaInterfaceSymbolNames](#) [► 951])

tchmi:general#/definitions/Object

Allows editing the [BaInterfaceSymbolNames](#) [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the **BalInterfaceSymbolNames** are overwritten by all controls of a type.

Brightness

tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean

Current brightness value (see [NumberOrBoolean](#) [► 927]).

BrightnessFeedback

tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean

Feedback for the brightness value (see [NumberOrBoolean](#) [► 927]).

MinBrightness

tchmi:general#/definitions/Number

Lowest permissible brightness value (see [Number](#)).

MaxBrightness

tchmi:general#/definitions/Number

Maximum permissible brightness value (see [Number](#)).

Events

Event	Description
onBrightnessChanged	Triggered when the user changes the brightness.
onHandModeReset	Triggered when the button for resetting from "hand" to automatic mode is pressed.

6.2.1.1.6.2.1 Enumerations

OperationState

Schema for the attribute [OperationState \[► 949\]](#) of the [Light \[► 945\]](#) control.

```
"TcHmi.BuildingAutomation.Controls.Light.OperationState": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "integer",
  "enum": [
    1,
    2,
    3
  ],
  "options": [
    {
      "label": "autoActive",
      "value": 1
    },
    {
      "label": "hand",
      "value": 2
    },
    {
      "label": "autoInactive",
      "value": 3
    }
  ]
}
```

DisplayMode

Schema for the attribute [DisplayMode \[► 948\]](#) of the [Light \[► 945\]](#) control.

```
"TcHmi.BuildingAutomation.Controls.Light.DisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1,
    2
  ],
  "options": [
    {
      "label": "lightBulb",
      "value": 0
    },
    {
      "label": "lightBulbFilled",
      "value": 1
    },
    {
      "label": "filled",
      "value": 2
    }
  ]
}
```

6.2.1.1.6.2.2 Types

BalInterfaceSymbolNames

Schema for editing the [BalInterfaceSymbolNames \[► 949\]](#) of the [Light \[► 945\]](#) template in an editor.

```

"\"TcHmi.BuildingAutomation.Controls.Light.BaInterfaceSymbolNames\": {
  \"$schema\": \"http://json-schema.org/draft-04/schema\",
  \"title\": \"BaInterfaceSymbolNames\",
  \"type\": \"object\",
  \"propertiesMeta\": [
    {
      \"name\": \"name\",
      \"displayName\": \"Name\",
      \"description\": \"Name of the light. (Default: sName, DataType: string)\",
      \"defaultValue\": null,
      \"defaultValueInternal\": null
    },
    {
      \"name\": \"initialize\",
      \"displayName\": \"Initialize Light\",
      \"description\": \"Flag to initialize the light. (Default: bInitialize_In, DataType: boolean)\",
      \"defaultValue\": null,
      \"defaultValueInternal\": null
    },
    {
      \"name\": \"cmdToggle\",
      \"displayName\": \"Command Toggle\",
      \"description\": \"Toggle to switch the light on and off. (Default: bToggle_In, DataType: boolean)\"
    },
    {
      \"name\": \"fdbLightOn\",
      \"displayName\": \"Feedback light on\",
      \"description\": \"Feedback if the light is on or off. (Default: bLightOn_Out, DataType: boolean)\"
    },
    {
      \"name\": \"operationState\",
      \"displayName\": \"Operation State\",
      \"description\": \"The state of the light (autoActive, hand, autoInactive). (Default: eOperationa
lState_Out, DataType: integer)\",
      \"defaultValue\": null,
      \"defaultValueInternal\": null
    },
    {
      \"name\": \"resetManual\",
      \"displayName\": \"Reset Manual\",
      \"description\": \"Symbol to reset the light into automatic mode. (Default: bResetManual_In, Data
Type: boolean)\",
      \"defaultValue\": null,
      \"defaultValueInternal\": null
    },
    {
      \"name\": \"cmdBrightness\",
      \"displayName\": \"Command Brightness\",
      \"description\": \"The command for the brightness. (Default: fLightValue_In, DataType: number)\",
      \"defaultValue\": null,
      \"defaultValueInternal\": null
    },
    {
      \"name\": \"fdbBrightness\",
      \"displayName\": \"Feedback Brightness\",
      \"description\": \"The feedback for the brightness. (Default: fLightValue_Out, DataType: number)\"
    },
    {
      \"name\": \"cmdTemperature\",
      \"displayName\": \"Command Temperature\",
      \"description\": \"The command for the light temperature. (Default: fLightTemperature_In, DataTyp
e: number)\",
      \"defaultValue\": null,
      \"defaultValueInternal\": null
    },
    {
      \"name\": \"fdbTemperature\",
      \"displayName\": \"Feedback Temperature\",
      \"description\": \"The feedback for the light temperature. (Default: fLightTemperature_Out, DataT
ype: number)\",
      \"defaultValue\": null,
      \"defaultValueInternal\": null
    }
  ]
}

```

```

    }
  ],
  "properties": {
    "name": {
      "type": "string"
    },
    "initialize": {
      "type": "string"
    },
    "operationState": {
      "type": "string"
    },
    "resetManual": {
      "type": "string"
    },
    "cmdToggle": {
      "type": "string"
    },
    "fdbLightOn": {
      "type": "string"
    },
    "cmdBrightness": {
      "type": "string"
    },
    "fdbBrightness": {
      "type": "string"
    },
    "cmdTemperature": {
      "type": "string"
    },
    "fdbTemperature": {
      "type": "string"
    }
  }
}

```

BaTemplateDescription

Schema for the attribute **BaTemplateDescription** [► 947] of the **Light** [► 945] control.

```

"TcHmi.BuildingAutomation.Controls.Light.TemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "TemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "command",
      "displayName": "Command",
      "description": "The command for the light. (Default: Command, DataType: Binary/Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "The feedback for the light. (Default: Feedback, DataType: Binary/Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "reset",
      "displayName": "Reset",
      "description": "The object to reset the light into automatic mode. (Default: ResetState, DataT
type: Binary)
",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "state",
      "displayName": "State",
      "description": "The state of the light (autoActive, hand, autoInactive). (Default: StateMI, Da
taType: Multistate)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "command": {

```

```

    "type": "string"
  },
  "feedback": {
    "type": "string"
  },
  "reset": {
    "type": "string"
  },
  "state": {
    "type": "string"
  }
}
}

```

6.2.1.1.6.3 Window

The **Window** control can display and control the position of windows or roof domes with drives.



Use

Use on any page where controls are needed to control windows.

Features

Position specification

The position of the window can be specified via an analog or binary value if the drive supports it. For analog values, buttons with predefined position values can be displayed.



The display of these buttons can be set as follows:

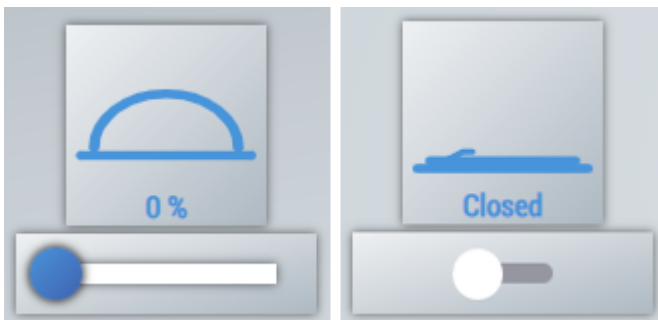
```

TcHmi.EventProvider.register('onInitialized', function (e, data) {
  e.destroy();
  TcHmi.BuildingAutomation.Controls.RoomAutomation.Window.ShowQuickLinks = true;
}).

```

Operation

Clicking on the **Window** control changes the visibility of the menu for setting the position.



Attributes

The control inherits from [BaseRoomControl](#) [► 934] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

BaObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.

The screenshot illustrates the configuration of the **BaObject** property in the TcHmiBa control. The left sidebar shows the control's structure with **Layout** expanded. The main area displays a context menu with options: **Reset**, **Create data binding...** (highlighted), and **Create function binding**. Below this, a dialog titled "Select value for ProjectNavigationTextual.BaObject" is shown. The dialog has tabs for **Server symbols**, **Internal symbols**, **Localizations**, **Mapped symbols**, and **Controls**. The **Mapped symbols** tab is active, showing a table of available symbols. The **ProjectStructure** symbol is selected, indicated by a red box labeled "1.". At the bottom of the dialog, the **OK** button is highlighted with a red box labeled "2.". The final result shows the **BaObject** property in the control's configuration field set to **PLC1::ProjectStructure**.

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

BaTemplateDescription

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Window.TemplateDescription

(see [TemplateDescription](#) [► 961])

tchmi:general#/definitions/Object

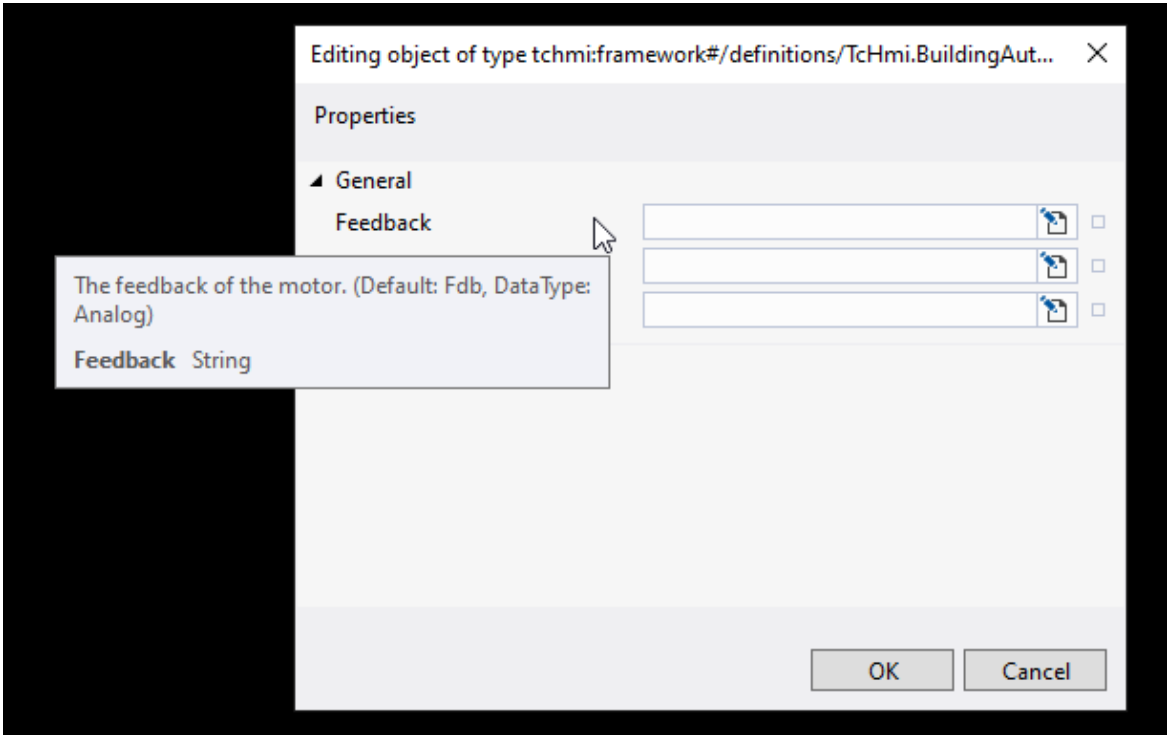
Allows editing the BaTemplateDescription (see [object](#)).



More detailed information about the *BaTemplate* can be found [here](#) [► 39].



The default values of the BaTemplateDescription, as well as the expected data types can be found in the tooltip of the dialog for setting the BaTemplateDescription:



Common

FacadeName

tchmi:general#/definitions/String

Name of the facade to which the window is assigned (see [string](#)). It can be set in [BuildingInformation control](#) [[► 824](#)].

ShowValue

tchmi:general#/definitions/Boolean

If TRUE, the value for the position is displayed (see [Boolean](#)).

DisplayMode

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Window.DisplayMode

Defines the display mode of the window (see [DisplayMode](#) [[► 959](#)]).

Name	Representation
roofDome	
window	

IconRotation

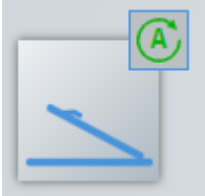
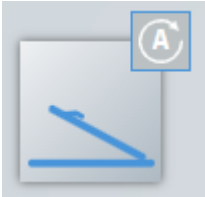
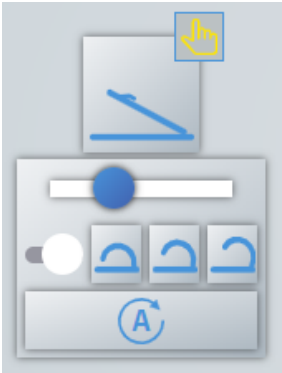
tchmi:general#/definitions/Number

Rotates the icon. Angle specification in degrees (see [Number](#)).

BaData**OperationState**

tchmi:framework#/definitions/TcHmi.BuildingAutomation.RoomAutomation.Window.OperationState

Displays the mode of the window (see [OperationState](#) [► 959]).

Name	Description	Representation
autoActive	Automatic for the window is active.	
autoInactive	Automatic for the window is switched off or not available.	
hand	Automatic was overwritten by a manual intervention.	
maintenance	Window is in maintenance position.	
safetyPosition	Window is in safety position.	

BaInterfaceSymbolNames

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Window.BaInterfaceSymbolNames

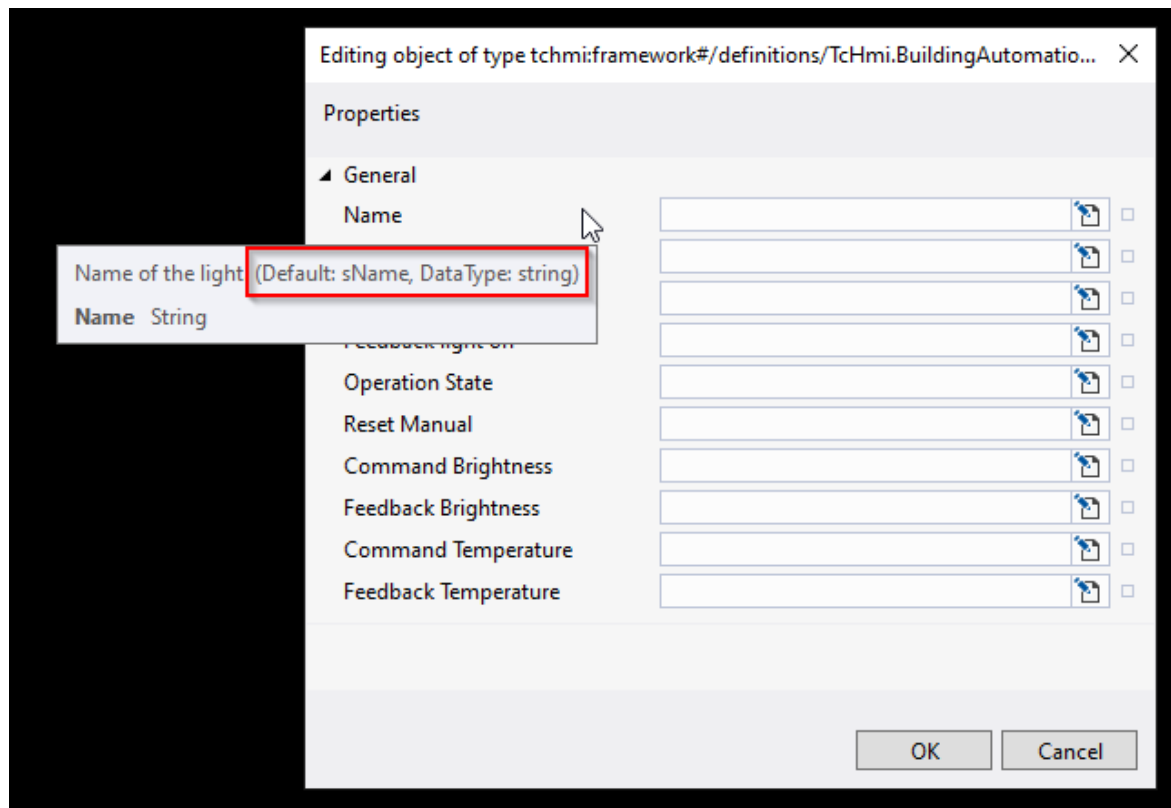
(see [BaInterfaceSymbolNames](#) [► 960])

tchmi:general#/definitions/Object

Allows editing the [BaInterfaceSymbolNames](#) [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the **BalInterfaceSymbolNames** are overwritten by all controls of a type.

Position

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean`

Current position (see [NumberOrBoolean](#) [► 927]).

PositionFeedback

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean`

Feedback for the position (see [NumberOrBoolean](#) [► 927]).

MinPosition

`tchmi:general#/definitions/Number`

Lowest permissible position value (see [Number](#)).

MaxPosition

`tchmi:general#/definitions/Number`

Largest permissible position value (see [Number](#)).

Events

Event	Description
onPositionChanged	Triggered when the user changes the position of the window or a roof dome.
onHandModeReset	Triggered when the button for resetting from "hand" to automatic mode is pressed.

6.2.1.1.6.3.1 Enumerations

OperationState

Schema for the attribute [OperationState \[► 957\]](#) of the [Window \[► 954\]](#) control.

```
"TcHmi.BuildingAutomation.Controls.Window.OperationState": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "integer",
  "enum": [
    1,
    2,
    3,
    4,
    5
  ],
  "options": [
    {
      "label": "autoActive",
      "value": 1
    },
    {
      "label": "autoInactive",
      "value": 2
    },
    {
      "label": "hand",
      "value": 3
    },
    {
      "label": "maintenance",
      "value": 4
    },
    {
      "label": "safetyPosition",
      "value": 5
    }
  ]
}
```

DisplayMode

Schema for the attribute [DisplayMode \[► 956\]](#) of the [Window \[► 954\]](#) control.

```
"TcHmi.BuildingAutomation.Controls.Window.DisplayMode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    0,
    1
  ],
  "options": [
    {
      "label": "standard",
      "value": 0
    },
    {
      "label": "roofDome",
      "value": 1
    }
  ]
}
```

6.2.1.1.6.3.2 Types

BaInterfaceSymbolNames

Schema for the attribute `BaInterfaceSymbolNames` [► 957] of the `Window` [► 954] control.

```
"TcHmi.BuildingAutomation.Controls.Window.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "name",
      "displayName": "Name",
      "description": "Name of the light the light. (Default: sName, DataType: string)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "commandOpen",
      "displayName": "Command open",
      "description": "Command to open the window. (Default: cmdOpen, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "commandClose",
      "displayName": "Command close",
      "description": "Command to close the window. (Default: cmdClose, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "Feedback about the window position. (Default: feedback, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "operationState",
      "displayName": "Operation state",
      "description": "The operation state of the window. (Default: operationState, DataType: integer)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "reset",
      "displayName": "Reset",
      "description": "Symbol to reset the window into automtmc mode. (Default: reset, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "name": {
      "type": "string"
    },
    "commandOpen": {
      "type": "string"
    },
    "commandClose": {
      "type": "string"
    },
    "feedback": {
      "type": "string"
    },
    "operationState": {
      "type": "string"
    },
    "reset": {
      "type": "string"
    }
  }
}
```

BaTemplateDescription

Schema for the attribute **BaTemplateDescription** [► 955] of the **Window** [► 954] control.

```
"TcHmi.BuildingAutomation.Controls.Window.TemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "TemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "commandOpen",
      "displayName": "Command open",
      "description": "The command that opens the window. (Default: CommandOpen, DataType: Binary)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "commandClose",
      "displayName": "Command close",
      "description": "The command that closes the window. (Default: CommandClose, DataType: Binary)"
    },
    {
      "name": "feedback",
      "displayName": "Feedback",
      "description": "The feedback for the window. (Default: Feedback, DataType: Binary/Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "commandOpen": {
      "type": "string"
    },
    "commandClose": {
      "type": "string"
    },
    "feedback": {
      "type": "string"
    }
  }
}
```

6.2.1.1.6.4 Sunblind

The **Sunblind** control can display and control the position and angle of sunblinds.



Use

Use on any page where controls are needed to control sunblinds.

Features

Operation modes

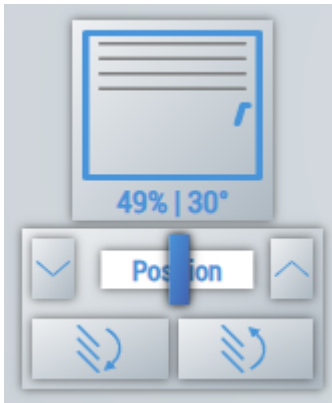
Display of different modes via the attribute **OperationState**.

Angle setting

The angle setting is optional and can only be used if the sunblind supports it. This function is enabled or disabled via the attribute **UseAngle**.

Operation

Clicking on the **Sunblind** control changes the visibility of the menu for setting the position or angle.



Attributes

The control inherits from [BaseRoomControl](#) [► 934] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

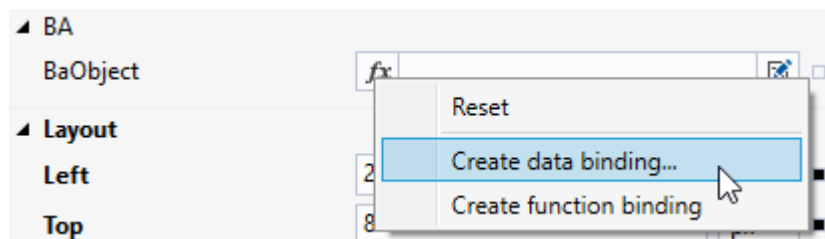
BAObject

tchmi:framework#/definitions/Symbol

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here](#) [► 63]. The attribute is not applicable to all controls.



Select value for ProjectNavigationTextual.BaObject

✕

Quick search...

Server symbols Internal symbols Localizations **Mapped symbols** Controls

Name	Value	Datatype	Online	Mapped from
└─ TcHmiBaServerExtension				
└─ PLC1	S PLC1		☐	PLC1
└─ ProjectStructure 1.	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Settings

Symbol expression

Refresh Collapse All Unmap Symbol **2.** OK Cancel

BA

BaObject S PLC1::ProjectStructure

BaTemplateDescription

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Sunblind.TemplateDescription

(see [TemplateDescription](#) [► 971])

tchmi:general#/definitions/Object

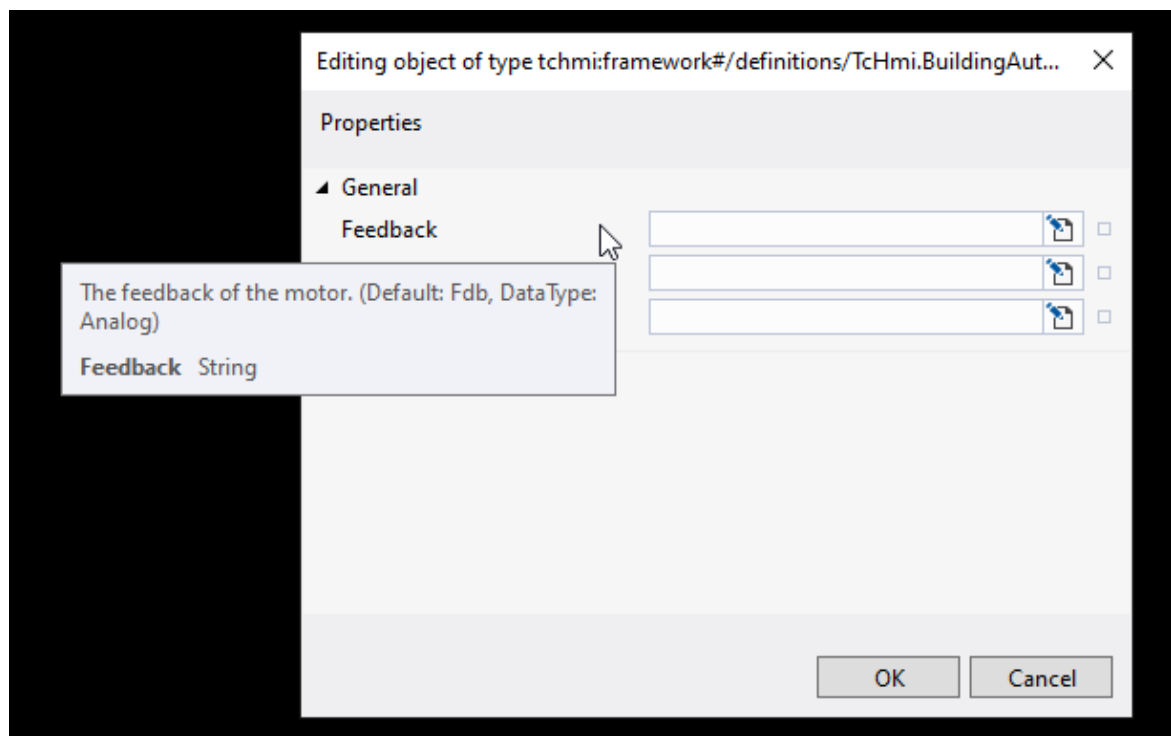
Allows editing the BaTemplateDescription (see [object](#)).



More detailed information about the *BaTemplate* can be found [here](#) [► 39].



The default values of the BaTemplateDescription, as well as the expected data types can be found in the tooltip of the dialog for setting the BaTemplateDescription:



Common

FacadeName

tchmi:general#/definitions/String

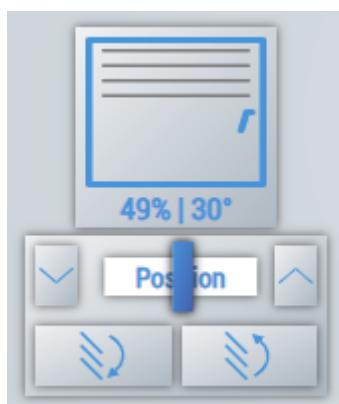
Name of the facade to which the sunblind is assigned (see [String](#)). It can be set in [BuildingInformation control](#) [► 824].

Controls

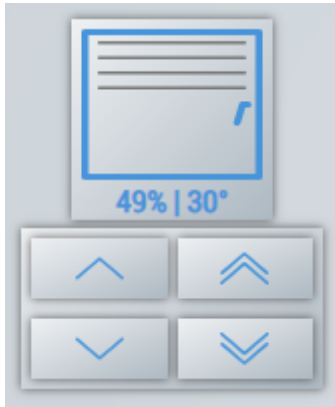
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Sunblind.Controls

Specifies the type of controls to be used for position and angle settings (see [Controls](#) [► 969]).

- sliderHorizontal



- buttons



ShowValue

```
tchmi:general#/definitions/Boolean
```

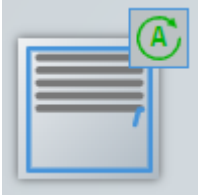
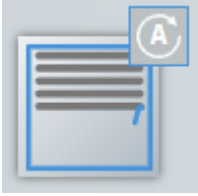
If TRUE, the values for angle and position are displayed (see [Boolean](#)).

BaData

OperationState

```
tchmi:framework#/definitions/  
TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.OperationState
```

Defines the displayed mode for the sunblind (see [OperationState](#) [[► 968](#)]).

Name	Description	Representation
autoActive	Automatic for the sunblind is active.	
autoInactive	Automatic for the sunblind is switched off or not available.	
hand	Automatic was overwritten by a manual intervention. Display of a button for resetting the "hand" mode. When pressed, the <i>onHandModeReset</i> event is triggered.	
maintenance	Sunblind is in the maintenance position.	
safetyPosition	Sunblind is in the safety position.	

BaInterfaceSymbolNames

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Sunblind.BaInterfaceSymbolNames

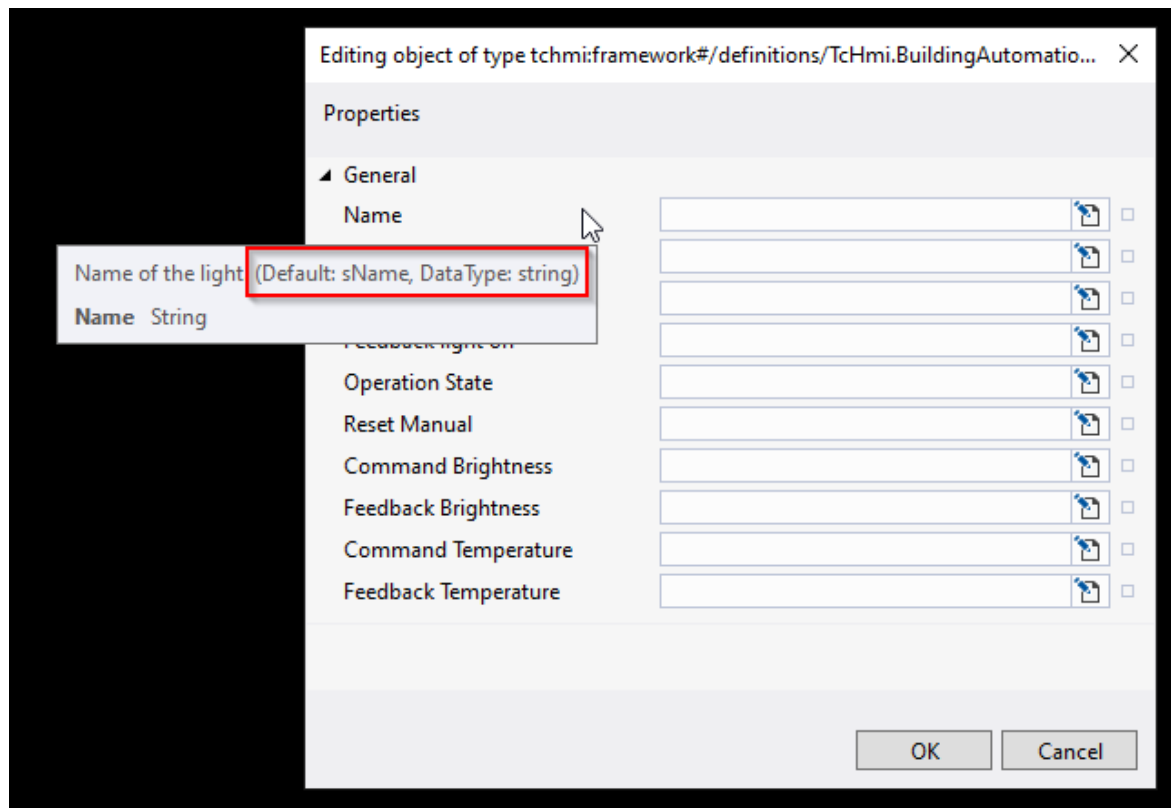
(see [BaInterfaceSymbolNames](#) [► 969])

tchmi:general#/definitions/Object

Allows editing the [BaInterfaceSymbolNames](#) [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the **BalInterfaceSymbolNames** are overwritten by all controls of a type.

Position

tchmi:general#/definitions/Number

Current position (see [Number](#)).

PositionFeedback

tchmi:general#/definitions/Number

Feedback for the position (see [Number](#)).

MinPosition

tchmi:general#/definitions/Number

Position value for the lower end position (see [Number](#)).

MaxPosition

tchmi:general#/definitions/Number

Position value for the upper end position (see [Number](#)).

Angle

UseAngle

tchmi:general#/definitions/Boolean

If TRUE, then the controls for controlling the angle are displayed (see [Boolean](#)).

Angle

tchmi:general#/definitions/Number

Current angle (see [Number](#)).**AngleFeedback**

tchmi:general#/definitions/Number

Feedback for the angle (see [Number](#)).**MinAngle**

tchmi:general#/definitions/Number

Lowest permissible angle (see [Number](#)).**MaxAngle**

tchmi:general#/definitions/Number

Maximum permissible angle (see [Number](#)).**AngleStep**

tchmi:general#/definitions/Number

Defines the step size with which the angle is adjusted via the angle buttons (see [Number](#)).**Events**

Event	Description
onPositionChanged	Triggered when the user changes the position of the sunblind.
onAngleChanged	Triggered when the user changes the angle of the sunblind.
onHandModeReset	Triggered when the button for resetting from "hand" to automatic mode is pressed.

6.2.1.1.6.4.1 Enumerations**OperationState**Schema for the attribute [OperationState](#) [► 965] of the [Sunblind](#) [► 961] control.

```

"TchHmi.BuildingAutomation.Controls.Sunblind.Mode": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "integer",
  "enum": [
    1,
    2,
    3,
    4,
    5
  ],
  "options": [
    {
      "label": "autoActive",
      "value": 1
    },
    {
      "label": "autoInactive",
      "value": 2
    },
    {
      "label": "hand",
      "value": 3
    }
  ]
}

```

```

        "label": "maintenance",
        "value": 4
    },
    {
        "label": "safetyPosition",
        "value": 5
    }
]
}

```

Controls

Schema for the attribute [Controls \[► 964\]](#) of the [Sunblind \[► 961\]](#) control.

```

"TcHmi.BuildingAutomation.Controls.Sunblind.Controls": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "string",
  "enum": [
    "buttons",
    "sliderHorizontal"
  ]
}

```

6.2.1.1.6.4.2 Types

BaInterfaceSymbolNames

Schema for the attribute [BaInterfaceSymbolNames \[► 966\]](#) of the [Sunblind \[► 961\]](#) control.

```

"TcHmi.BuildingAutomation.Controls.Sunblind.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "name",
      "displayName": "Name",
      "description": "Name of the sunblind. (Default: sName, DataType: string)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "cmdUp",
      "displayName": "Command Up",
      "description": "The command to drive the sunblind up. (Default: bUp_In, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "cmdDown",
      "displayName": "Command down",
      "description": "The command to drive the sunblind down. (Default: bDown_In, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "fdbOpen",
      "displayName": "Feedback open",
      "description": "The feedback if the sunblind is opened. (Default: bOpened_Out, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "fdbClose",
      "displayName": "Feedback close",
      "description": "The feedback if the sunblind is closed. (Default: bClosed_Out, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "initialize",
      "displayName": "Initialize",
      "description": "When set to true the sunblind will be initialized. (Default: bInitialize_In, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ]
}

```

```

        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "operationState",
        "displayName": "Operation state",
        "description": "Operation state of the sunblind. (Default: eOperationalState_Out, DataType: integer)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "resetManual",
        "displayName": "Reset manual",
        "description": "Symbol to reset the sunblind into automatic mode. (Default: bResetManual_In, DataType: boolean)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "positionCmd",
        "displayName": "Position command",
        "description": "The command to set the position of the sunblind. (Default: fSunblindPosition_In, DataType: number)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "positionFdb",
        "displayName": "Position feedback",
        "description": "The position feedback of the sunblind. (Default: fSunblindPosition_Out, DataType: number)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "angleCmd",
        "displayName": "Angle command",
        "description": "The command to set the angle of the sunblind. (Default: fSunblindAngle_In, DataType: number)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "angleFdb",
        "displayName": "Angle feedback",
        "description": "The angle feedback of the sunblind. (Default: fSunblindAngle_Out, DataType: number)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "minAngle",
        "displayName": "Minimum angle",
        "description": "The minimum angle of the sunblind. (Default: fAnglLmtDwn, DataType: number)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "maxAngle",
        "displayName": "Maximum angle",
        "description": "The maximum angle of the sunblind. (Default: fAnglLmtUp, DataType: number)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    ],
    "properties": {
        "name": {
            "type": "string"
        },
        "cmdUp": {
            "type": "string"
        },
        "cmdDown": {
            "type": "string"
        },
        "initialize": {
            "type": "string"
        },
        "operationState": {

```

```

    "type": "string"
  },
  "resetManual": {
    "type": "string"
  },
  "positionCmd": {
    "type": "string"
  },
  "positionFdb": {
    "type": "string"
  },
  "angleFdb": {
    "type": "string"
  },
  "angleCmd": {
    "type": "string"
  },
  "minAngle": {
    "type": "string"
  },
  "maxAngle": {
    "type": "string"
  },
  },
}
}

```

BaTemplateDescription

Schema for the attribute [BaTemplateDescription \[► 963\]](#) of the [Sunblind \[► 961\]](#) control.

```

"TcHmi.BuildingAutomation.Controls.Sunblind.TemplateDescription": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "TemplateDescription",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "positionFeedback",
      "displayName": "Position feedback",
      "description": "The position feedback of the sunblind. (Default: Feedback, DataType: Binary/Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "positionCommand",
      "displayName": "Position command",
      "description": "The command to set the position of the sunblind. (Default: ActlPos, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "angleFeedback",
      "displayName": "Angle feedback",
      "description": "The angle feedback of the sunblind. (Default: ActlAngl, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "angleCommand",
      "displayName": "Angle command",
      "description": "The command to set the angle of the sunblind. (Default: LocInputAngl, DataType: Analog)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "positionFeedback": {
      "type": "string"
    },
    "positionCommand": {
      "type": "string"
    },
    "angleFeedback": {
      "type": "string"
    },
    "angleCommand": {
      "type": "string"
    }
  }
}

```

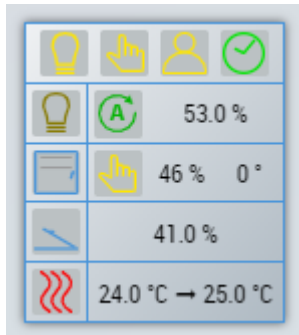


```
}  
}  
}
```

6.2.1.1.6.5 RoomControl

The **RoomControl** can combine the various controls of the room automation. Available components are:

- [AirConditioning \[► 939\]](#)
- [Light \[► 945\]](#)
- [Sunblind \[► 961\]](#)
- [Window \[► 954\]](#)



Use

The **RoomControl** can be used to automate a room or area with only one control. It is possible, for example, to combine only one area with lamps so that the overview in the visualization is maintained.

Features

All settings that are possible with the individual controls can be made.

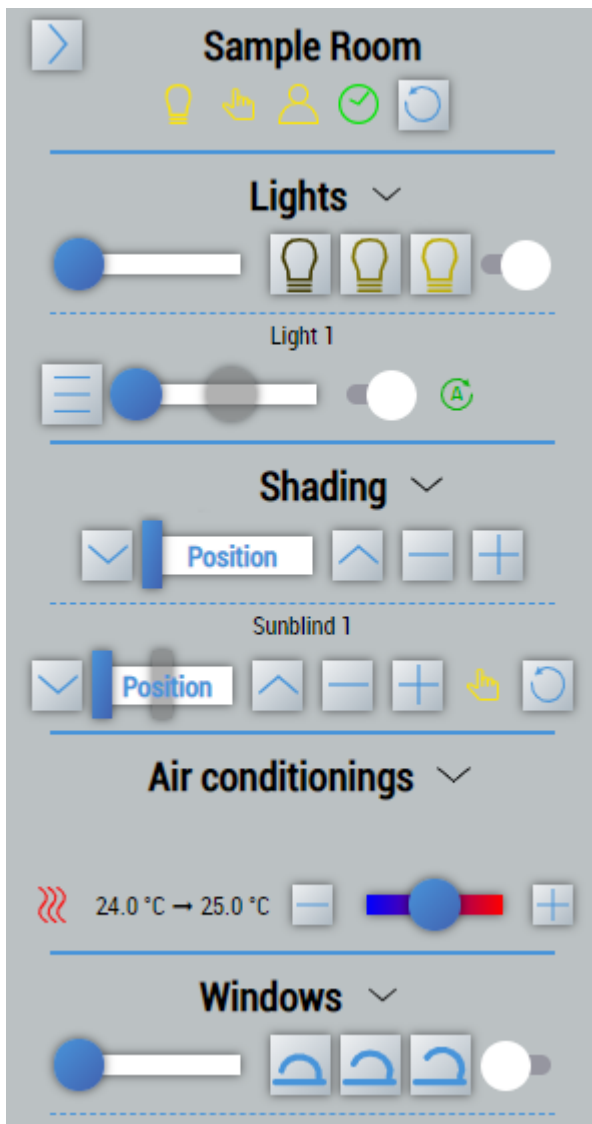
Room status

The general room information is displayed in the header of the control:

- Light on or off
- Mode or message with the highest priority
- Occupancy
- Overrun time

Side menu for control

Clicking on the control opens the side menu, where the controls for the individual areas can be found.



Distributed control

Controls are located above each area, e.g. to control all lamps simultaneously. This makes it possible to set all lamps to the same brightness value.

Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the [feedback concept](#) [► 43].

BA

BAObject

```
tchmi:framework#/definitions/Symbol
```

To use the generic functionalities of TcHmiBa (see [Symbol](#)). It links a single object or a complete view (including children) to the control.



More detailed information about the generic possibilities of TcHmiBa can be found [here \[► 63\]](#). The attribute is not applicable to all controls.

The screenshot illustrates the process of creating a data binding in the TcHmiBa development environment. It shows the 'BA' project structure with 'BaObject' and 'Layout' (Left, Top) elements. A context menu is open over the 'Left' element, with 'Create data binding...' selected. Below this, a dialog box titled 'Select value for ProjectNavigationTextual.BaObject' is shown. The 'Mapped symbols' tab is active, displaying a table of symbols. The 'ProjectStructure' symbol is highlighted, and its value is set to 'PLC1.ProjectStructure'. The 'OK' button is highlighted with a red box and the number '2'.

Name	Value	Datatype	Online	Mapped from
TcHmiBaServerExtension				
PLC1	S PLC1		☐	PLC1
ProjectStructure	S PLC1.ProjectStructure		☐	ProjectStructure
TcHmiBaServerExtension.GetUnits			☒	GetUnits
TcHmiBaServerExtension.TerminalClientActiveEventsCount	number		☒	TerminalClientActiveEv
TcHmiBaServerExtension.TerminalClientEvents	object		☒	TerminalClientEvents

Below the table, the 'Symbol expression' window shows the 'PLC1::ProjectStructure' symbol selected. The 'OK' button is highlighted with a red box and the number '2'.

Common

ControlUnits

tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomControl.ControlList

Determines which components ([AirConditioning \[► 939\]](#), [Light \[► 945\]](#), [Sunblind \[► 961\]](#), [Window \[► 954\]](#)) should be added (see [ControlList \[► 976\]](#)).

Name

tchmi:general#/definitions/String

The room name (see [String](#)).

HideRoomStatus

tchmi:general#/definitions/Boolean

If TRUE, then the room information (header) is not visible (see [Boolean](#)).

ShowRoomName

tchmi:general#/definitions/Boolean

If TRUE, the [room name \[► 974\]](#) is displayed instead of the room information (see [Boolean](#)).

BaData

Presence

tchmi:general#/definitions/Boolean

If TRUE, then presence was detected in the room (see [Boolean](#)).

- Presence active



- Presence inactive



DelayActive

tchmi:general#/definitions/Boolean

If TRUE, then the overrun time for automatic control is active (see [Boolean](#)).

- Overrun time active



- Overrun time inactive



Lights

ShowLights

tchmi:general#/definitions/Boolean

If TRUE, the lights are displayed in the control. The components are always visible in the side menu (see [Boolean](#)).



Further information about the attributes can be found in the documentation for the [Light](#) [► 945] control.

Sunblinds

ShowSunblinds

tchmi:general#/definitions/Boolean

If TRUE, the Sunblinds are displayed in the control (see [Boolean](#)). The components are always visible in the side menu.



Further information about the attributes can be found in the documentation for the [Sunblind](#) [► 961] control.

AirConditionings

ShowAirConditionings

tchmi:general#/definitions/Boolean

If TRUE, the AirConditionings are displayed in the control (see [Boolean](#)). The components are always visible in the side menu.



Further information on the attributes can be found in the documentation for the [AirConditioning \[► 939\]](#) control.

Windows

ShowWindows

tchmi:general#/definitions/Boolean

If TRUE, then the Windows are displayed in the control (see [Boolean](#)). The components are always visible in the side menu.



Further information about the attributes can be found in the documentation for the [Window \[► 954\]](#) control.

6.2.1.1.6.5.1 Types

ControlList

Schema for the attribute [ControlUnits \[► 974\]](#) of the [RoomControl \[► 972\]](#) controls.

```
"TcHmi.BuildingAutomation.Controls.RoomControl.ControlList": {
  "title": "ControlList",
  "type": "array",
  "items": {
    "oneOf": [
      {
        "title": "Light",
        "type": "object",
        "additionalProperties": false,
        "engineeringColumns": [
          "Name"
        ],
      },
      {
        "name": "name",
        "displayName": "Name",
        "category": "Common",
        "displayPriority": 10,
        "description": "Name of the light.",
        "defaultValue": null,
        "defaultValueInternal": null
      },
      {
        "name": "brightness",
        "displayName": "Brightness",
        "category": "Common",
        "displayPriority": 10,
        "description": "Brightness of the light. When used as data binding the brightness will be written when changed by the user.",
        "defaultValue": 0,
        "defaultValueInternal": 0
      },
      {
        "name": "brightnessFeedback",
        "displayName": "BrightnessFeedback",
        "category": "Common",
        "displayPriority": 10,
        "description": "Feedback for the brightness of the light.",
        "defaultValue": null,
        "defaultValueInternal": null
      }
    ]
  },
  "propertiesMeta": [
    {
      "name": "baObject",
      "displayName": "BaObject",
      "category": "BA",
      "displayPriority": 1,
      "description": "BaView of the light.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "name",
      "displayName": "Name",
      "category": "Common",
      "displayPriority": 10,
      "description": "Name of the light.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "brightness",
      "displayName": "Brightness",
      "category": "Common",
      "displayPriority": 10,
      "description": "Brightness of the light. When used as data binding the brightness will be written when changed by the user.",
      "defaultValue": 0,
      "defaultValueInternal": 0
    },
    {
      "name": "brightnessFeedback",
      "displayName": "BrightnessFeedback",
      "category": "Common",
      "displayPriority": 10,
      "description": "Feedback for the brightness of the light.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ]
}
```

```

    },
    {
      "name": "mode",
      "displayName": "Mode",
      "category": "Common",
      "displayPriority": 10,
      "description": "Mode of the light.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "resetHandMode",
      "displayName": "ResetHandMode",
      "category": "Common",
      "displayPriority": 10,
      "description": "Symbol which is set to true if the hand mode was reseted.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "baObject": {
      "type": [ "string", "null" ]
    },
    "name": {
      "type": [ "string", "null" ]
    },
    "brightness": {
      "type": [ "number", "boolean", "string", "null" ]
    },
    "brightnessFeedback": {
      "type": [ "number", "boolean", "string", "null" ]
    },
    "mode": {
      "oneOf": [
        { "$ref": "tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Light.Mode" },
        { "type": [ "string", "null" ] }
      ]
    },
    "resetHandMode": {
      "type": [ "string", "null" ]
    }
  }
},
{
  "title": "Sunblind",
  "type": "object",
  "additionalProperties": false,
  "engineeringColumns": [
    "Name"
  ],
  "propertiesMeta": [
    {
      "name": "baObject",
      "displayName": "BaObject",
      "category": "BA",
      "displayPriority": 1,
      "description": "BaView of the sunblind.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "name",
      "displayName": "Name",
      "category": "Common",
      "displayPriority": 10,
      "description": "Name of the sunblind.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "facadeName",
      "displayName": "FacadeName",
      "category": "Common",
      "displayPriority": 10,
      "description": "Defines the facade where the sunblind is mounted. (Equal to the name i
n BuildingInformation control)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ]
}

```

```

    },
    {
      "name": "position",
      "displayName": "Position",
      "category": "Common",
      "displayPriority": 10,
      "description": "Position of the sunblind. When used as data binding the position will
be written when changed by the user.",
      "defaultValue": null,
      "defaultValueInternal": 0
    },
    {
      "name": "positionFeedback",
      "displayName": "PositionFeedback",
      "category": "Common",
      "displayPriority": 10,
      "description": "Feedback for the position of the sunblind.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "angle",
      "displayName": "Angle",
      "category": "Common",
      "displayPriority": 10,
      "description": "Angle of the sunblind. When used as data binding the angle will be wri
tten when changed by the user.",
      "defaultValue": null,
      "defaultValueInternal": 0
    },
    {
      "name": "angleFeedback",
      "displayName": "AngleFeedback",
      "category": "Common",
      "displayPriority": 10,
      "description": "Feedback for the angle of the sunblind.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "mode",
      "displayName": "Mode",
      "category": "Common",
      "displayPriority": 10,
      "description": "Mode of the sunblind.",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "resetHandMode",
      "displayName": "ResetHandMode",
      "category": "Common",
      "displayPriority": 10,
      "description": "Symbol which is set to true if the hand mode was reseted.",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "baObject": {
      "type": [ "string", "null" ]
    },
    "name": {
      "type": [ "string", "null" ]
    },
    "facadeName": {
      "type": [ "string", "null" ]
    },
    "position": {
      "type": [ "number", "null", "string" ]
    },
    "positionFeedback": {
      "type": [ "number", "null", "string" ]
    },
    "angle": {
      "type": [ "number", "null", "string" ]
    },
    "angleFeedback": {
      "type": [ "number", "null", "string" ]
    }
  },

```

```

        "mode": {
            "oneOf": [
                { "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Sunblind.Mode" },
                { "type": [ "string", "null" ] }
            ]
        },
        "resetHandMode": {
            "type": [ "string", "null" ]
        }
    },
    {
        "title": "Air conditioning",
        "type": "object",
        "additionalProperties": false,
        "engineeringColumns": [
            "Name"
        ],
        "propertiesMeta": [
            {
                "name": "baObject",
                "displayName": "BaObject",
                "category": "BA",
                "displayPriority": 1,
                "description": "BaView of the air conditioning.",
                "defaultValue": null,
                "defaultValueInternal": null
            },
            {
                "name": "name",
                "displayName": "Name",
                "category": "Common",
                "displayPriority": 10,
                "description": "Name of the light.",
                "defaultValue": null,
                "defaultValueInternal": null
            },
            {
                "name": "currentTemp",
                "displayName": "CurrentTemperature",
                "category": "Common",
                "displayPriority": 10,
                "description": "The current temperature.",
                "defaultValue": 0,
                "defaultValueInternal": 0
            },
            {
                "name": "tempAdjust",
                "displayName": "TemperatureAdjustment",
                "category": "Common",
                "displayPriority": 10,
                "description": "The adjustment for the current temperature setpoint.",
                "defaultValue": null,
                "defaultValueInternal": null
            },
            {
                "name": "tempAdjustFeedback",
                "displayName": "TempAdjustFeedback",
                "category": "Common",
                "displayPriority": 10,
                "description": "Feedback for the temp adjust.",
                "defaultValue": null,
                "defaultValueInternal": null
            },
            {
                "name": "heatingSetpoint",
                "displayName": "HeatingSetpoint",
                "category": "Common",
                "displayPriority": 10,
                "description": "Setpoint of the heating controller. If no cooling setpoint is used it
is possible to use the CoolingActive flag to switch between heating and cooling mode.",
                "defaultValue": null,
                "defaultValueInternal": null
            },
            {
                "name": "coolingSetpoint",
                "displayName": "CoolingSetpoint",
                "category": "Common",
                "displayPriority": 10,

```



```

        "description": "Setpoint of the cooling controller.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "heatingActive",
        "displayName": "HeatingActive",
        "category": "Common",
        "displayPriority": 10,
        "description": "Flag that defines if heating mode is active or not.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "coolingActive",
        "displayName": "CoolingActive",
        "category": "Common",
        "displayPriority": 10,
        "description": "Flag that defines if cooling mode is active or not.",
        "defaultValue": null,
        "defaultValueInternal": null
    }
},
"properties": {
    "baObject": {
        "type": [ "string", "null" ]
    },
    "name": {
        "type": [ "string", "null" ]
    },
    "currentTemp": {
        "type": [ "number", "null", "string" ]
    },
    "tempAdjust": {
        "type": [ "number", "null", "string" ]
    },
    "tempAdjustFeedback": {
        "type": [ "number", "string", "null" ]
    },
    "heatingSetpoint": {
        "type": [ "number", "null", "string" ]
    },
    "coolingSetpoint": {
        "type": [ "number", "null", "string" ]
    },
    "heatingActive": {
        "type": [ "boolean", "number", "null", "string" ]
    },
    "coolingActive": {
        "type": [ "boolean", "number", "null", "string" ]
    }
},
},
{
    "title": "Window",
    "type": "object",
    "additionalProperties": false,
    "engineeringColumns": [
        "Name"
    ],
    "propertiesMeta": [
        {
            "name": "baObject",
            "displayName": "BaObject",
            "category": "BA",
            "displayPriority": 1,
            "description": "BaView of the light.",
            "defaultValue": null,
            "defaultValueInternal": null
        },
        {
            "name": "name",
            "displayName": "Name",
            "category": "Common",
            "displayPriority": 10,
            "description": "Name of the window.",
            "defaultValue": null,
            "defaultValueInternal": null
        }
    ],
    {

```

```

        "name": "facadeName",
        "displayName": "FacadeName",
        "category": "Common",
        "displayPriority": 10,
        "description": "Defines the facade where the sunblind is mounted. (Equal to the name i
n BuildingInformation control)",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "windowPosition",
        "displayName": "Position",
        "category": "Common",
        "displayPriority": 10,
        "description": "Position of the light. When used as data binding the position will be
written when changed by the user.",
        "defaultValue": null,
        "defaultValueInternal": 0
    },
    {
        "name": "windowPositionFeedback",
        "displayName": "PositionFeedback",
        "category": "Common",
        "displayPriority": 10,
        "description": "Feedback for the position of the window.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "mode",
        "displayName": "Mode",
        "category": "Common",
        "displayPriority": 10,
        "description": "Mode of the window.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "resetHandMode",
        "displayName": "ResetHandMode",
        "category": "Common",
        "displayPriority": 10,
        "description": "Symbol which is set to true if the hand mode was reseted.",
        "defaultValue": null,
        "defaultValueInternal": null
    },
    {
        "name": "displayMode",
        "displayName": "DisplayMode",
        "category": "Common",
        "displayPriority": 10,
        "description": "Defines the display mode of the window.",
        "defaultValue": 0,
        "defaultValueInternal": 0
    },
    {
        "name": "iconRotation",
        "displayName": "IconRotation",
        "category": "Common",
        "displayPriority": 10,
        "description": "The rotation of the icon in degree (0°-360°).",
        "defaultValue": null,
        "defaultValueInternal": null
    }
],
"properties": {
    "baObject": {
        "type": [ "string", "null" ]
    },
    "name": {
        "type": [ "string", "null" ]
    },
    "facadeName": {
        "type": [ "string", "null" ]
    },
    "windowPosition": {
        "type": [ "number", "boolean", "string", "null" ]
    },
    "windowPositionFeedback": {
        "type": [ "number", "boolean", "string", "null" ]
    }
}

```

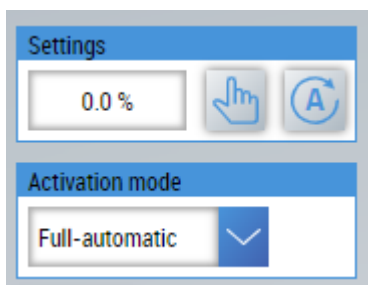
```

    },
    "mode": {
      "oneOf": [
        { "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Window.Mode" },
        { "type": [ "string", "null" ] }
      ]
    },
    "resetHandMode": {
      "type": [ "string", "null" ]
    },
    "displayMode": {
      "oneOf": [
        { "$ref": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Window.DisplayMode" },
        { "type": [ "string", "null" ] }
      ]
    },
    "iconRotation": {
      "type": [ "number", "null" ]
    }
  }
}
]
}
}
}

```

6.2.1.1.6.6 LightZone

The **LightZone** control is used to display and control a light zone.



Use

Use on any page where controls are needed to control light zones.

Features

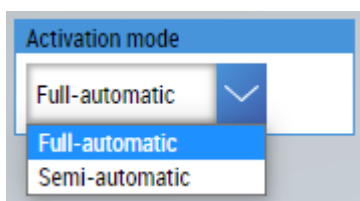
Settings

The brightness value of the light zone can be specified via the **Settings** category. The brightness value is activated by pressing the **Hand** button.

This switches the light zone to manual mode. The light zone can then be switched back to automatic mode via the **Automatic** button.

Activation mode

Via the category **Activation mode** it is possible to set how the automatic of the light zone should be activated.



If **Full-automatic** is selected, the light zone automatic is activated by presence detection.

If **Semi-automatic** is selected, the light zone automatic is activated, in which the light was activated via the on switch.

Attributes

The control inherits from [BaseControl](#) [► 923] and thus has the same attributes. In addition, there are the following attributes.

BaData

BalInterface

```
tchmi:framework#/definitions/Symbol
```

Allows linking a symbol (see [symbol](#)) that satisfies the [BalInterface](#) [► 999] of the control. With this symbol all necessary data points of the control can be linked with only one binding.

BalInterfaceSymbolNames

```
tchmi:framework#/definitions/  
TcHmi.BuildingAutomation.Controls.RoomAutomation.LightZone.BalInterfaceSymbolNames
```

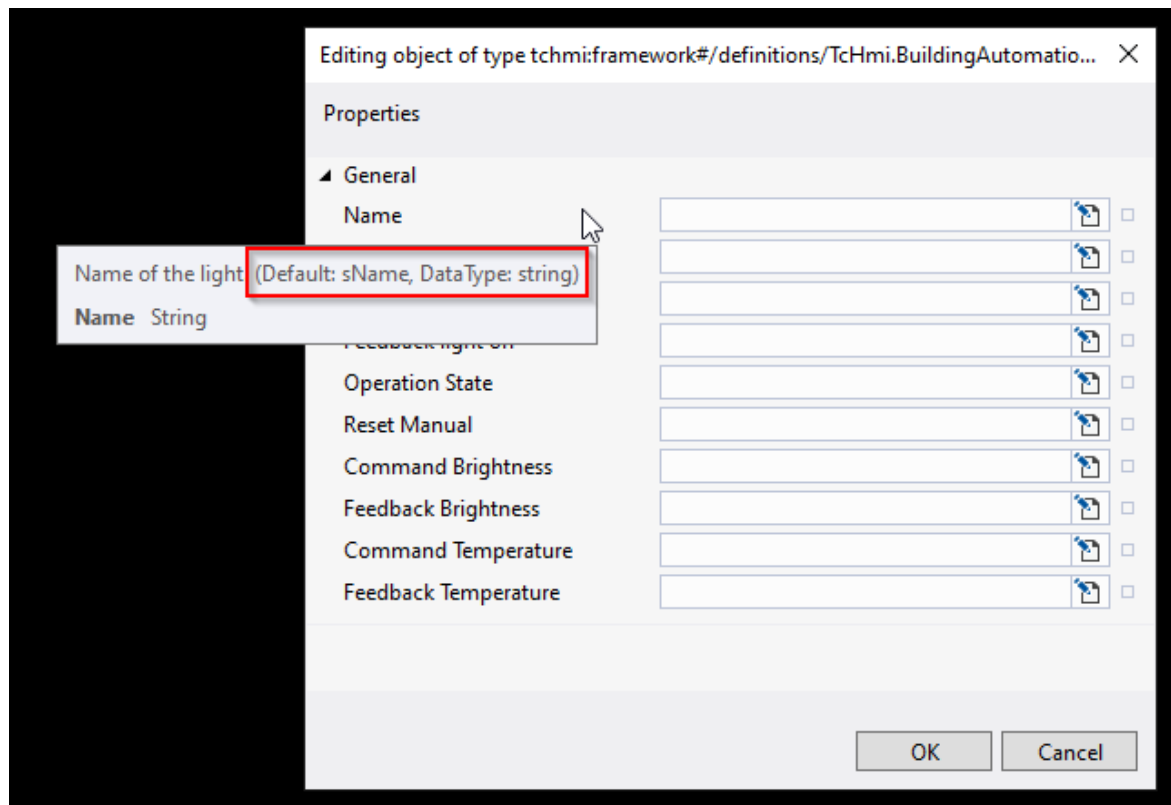
(see [BalInterfaceSymbolNames](#) [► 984])

```
tchmi:general#/definitions/Object
```

Allows editing the [BalInterfaceSymbolNames](#) [► 1000].



The default values of **BalInterfaceSymbolNames**, as well as the expected data types can be found in the tooltip of the dialog for setting **BalInterfaceSymbolNames**:



Here is described how the **BalInterfaceSymbolNames** are overwritten by all controls of a type.

6.2.1.1.6.6.1 Types

BaInterfaceSymbolNames

Schema for the attribute `BaInterfaceSymbolNames` [► 983] of the `LightZone` [► 982] control.

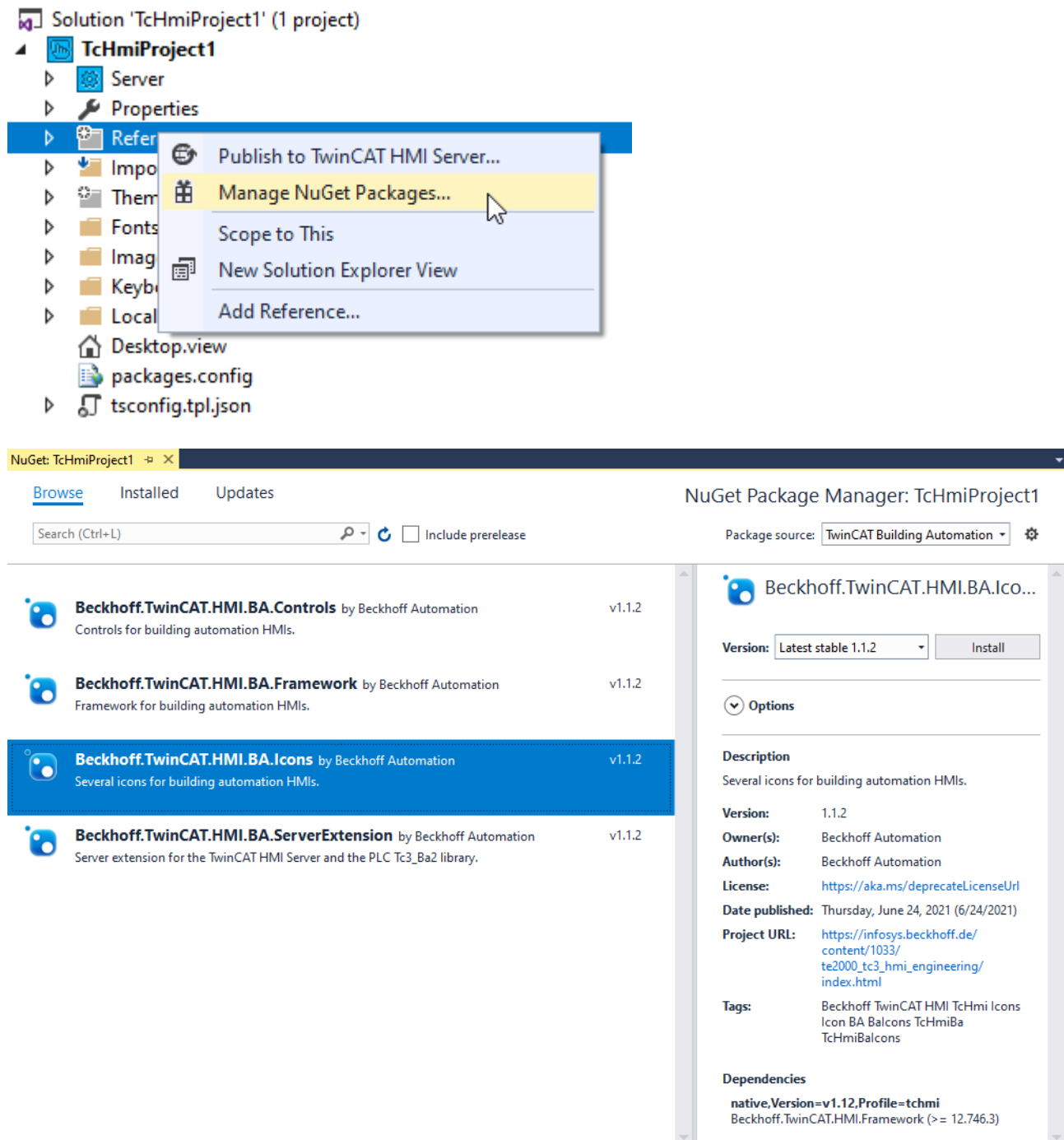
```
"TcHmi.BuildingAutomation.Controls.RoomAutomation.LightZone.BaInterfaceSymbolNames": {
  "$schema": "http://json-schema.org/draft-04/schema",
  "title": "BaInterfaceSymbolNames",
  "type": "object",
  "propertiesMeta": [
    {
      "name": "manualValue",
      "displayName": "Manual value",
      "description": "Manual value of the light zone. (Default: fSetValMan_In, DataType: number)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "writeManualValue",
      "displayName": "Write manual value",
      "description": "Write the manual value to the lights of the zone. (Default: bSetValMan_In, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "controlModeActive",
      "displayName": "Control mode active",
      "description": "Defines if the control mode is active or inactive. (Default: bCtrlMod_Out, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "activateControlMode",
      "displayName": "Activate control mode",
      "description": "Set to true to activate the alarm mode. (Default: bSetCtrlMod_In, DataType: boolean)",
      "defaultValue": null,
      "defaultValueInternal": null
    },
    {
      "name": "activationMode",
      "displayName": "Activation mode",
      "description": "Activation mode of the light zone. (Default: eLightActivationMode_In, DataType: integer)",
      "defaultValue": null,
      "defaultValueInternal": null
    }
  ],
  "properties": {
    "manualValue": {
      "type": "string"
    },
    "writeManualValue": {
      "type": "string"
    },
    "controlModeActive": {
      "type": "string"
    },
    "activateControlMode": {
      "type": "string"
    },
    "activationMode": {
      "type": "string"
    }
  }
}
```

6.2.1.2 Icons

TcHmiBa contains various icons that are necessary for the implementation of visualizations for building automation. The icons are created in *.svg format and are intended for use on the web.

Installation

In order to use the icons, the NuGet package **Beckhoff.TwinCAT.HMI.BA.Icons** must be installed.



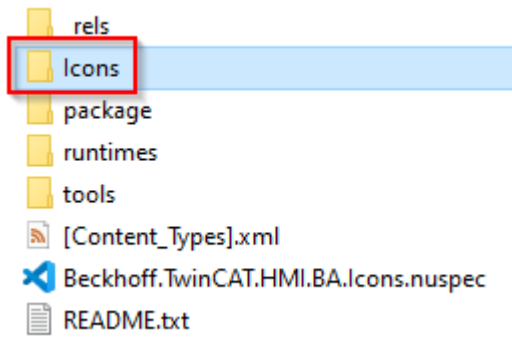
Since there is a dependency on the Beckhoff.TwinCAT.HMI.Framework, this is also installed.

Use

There are three different application options.

ZIP archive

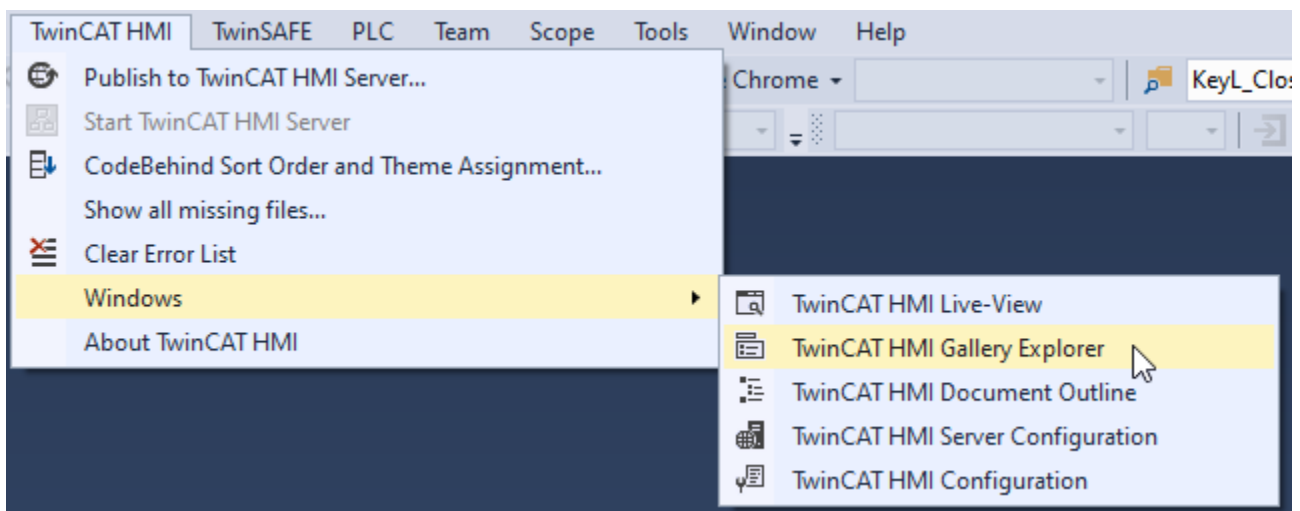
A NuGet package is basically a ZIP archive that allows direct use of the icons after unpacking. The unpacked content looks like this:

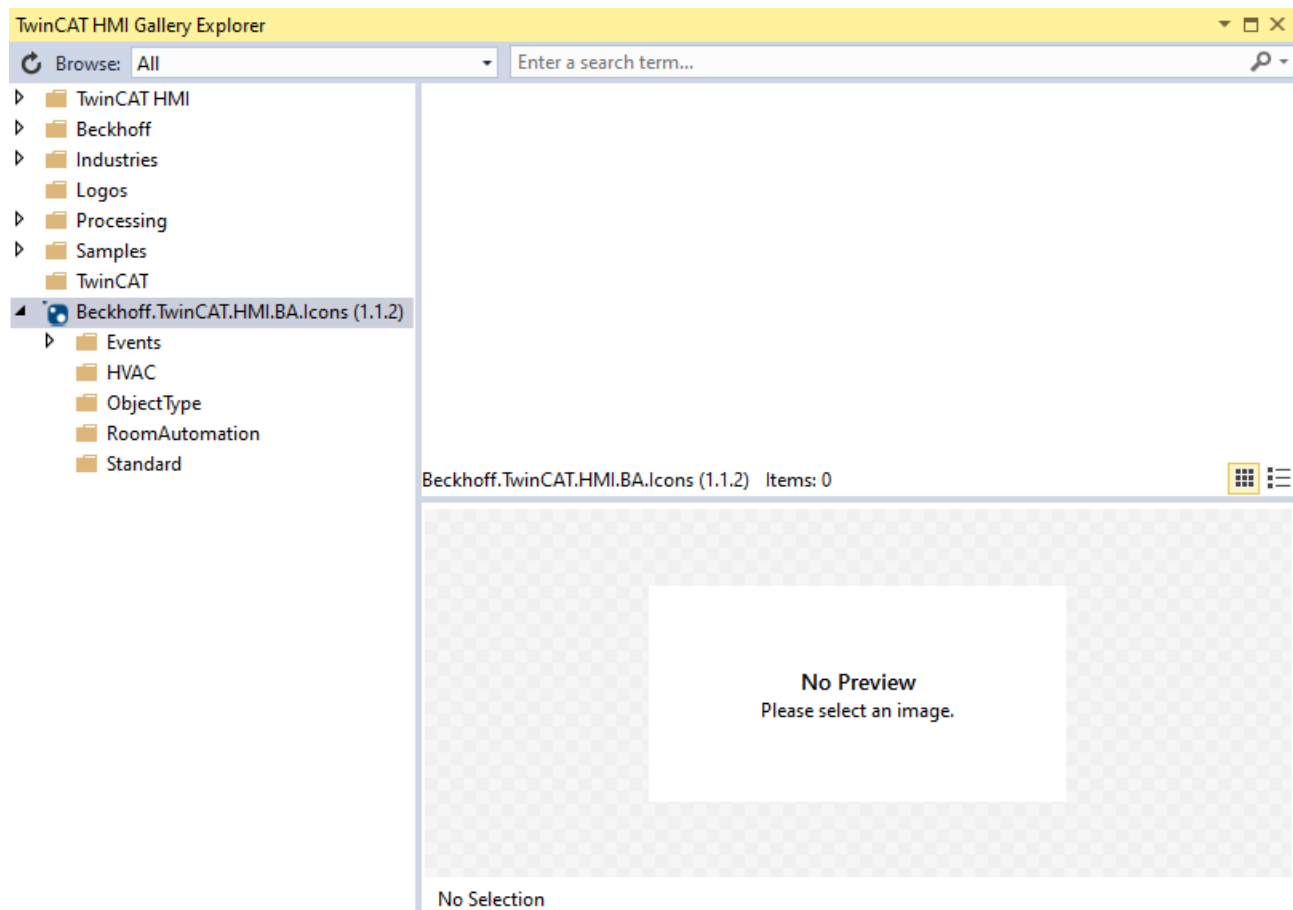


Only the folder **Icons** is relevant. It contains the various icons divided according to content categories.

GalleryExplorer

After installing the NuGet package in a **TwinCAT HMI project**, the icons integrate into the **GalleryExplorer**.





The icons here are arranged according to function. To use an icon from the **GalleryExplorer**, it must be dragged and dropped into a folder in the project.



The icon is created as a copy in the project directory, not as a reference.

Use as reference

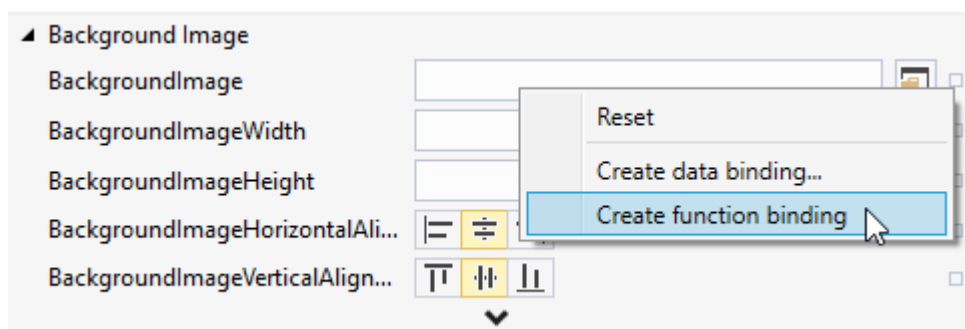
When using the icons as reference, their extended functionalities can be used. In addition, the icons benefit directly from updates to the NuGet package.

TcHmi project

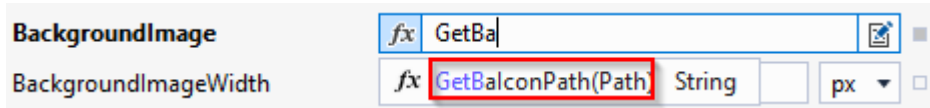
The function `TcHmi.Functions.GetBalconPath()` can be used to reference the icons in a TcHmi project.

The following steps are required:

1. Drag a control (e.g. a button) to the content/view.
2. Create the **Function Binding**.



3. In the field **Backgroundimage** enter **GetBalconPath** (function is suggested).

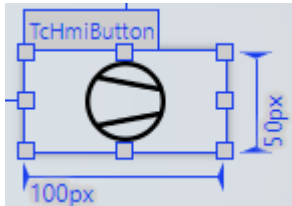


4. Path specification to the icon, e.g. "HVAC/fan" (quotation marks must be observed).
 5. Adjust size and position.



The path information can be taken from the structure of the *GalleryExplorer*.

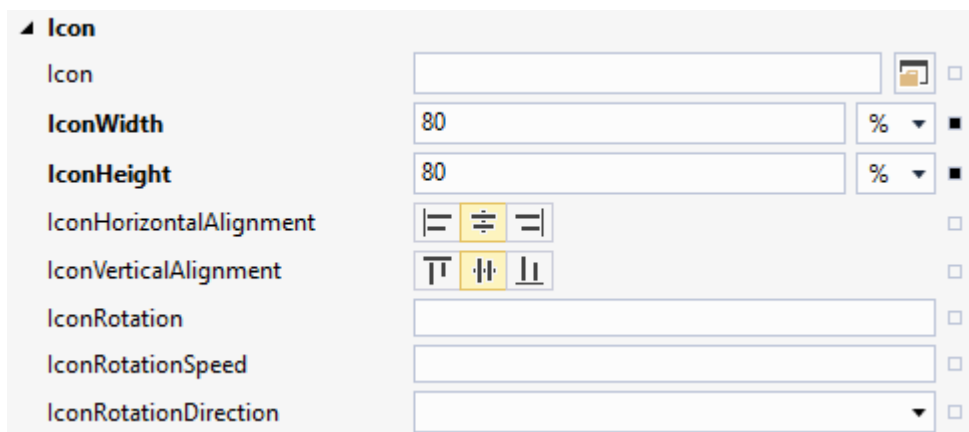
⇒ After that the button should look like this:



The extended functions of the icons can only be used with controls from the NuGet package **Beckhoff.TwinCAT.HMI.BA.Controls**.

The controls that have a category *Icon* have extended setting options.

As an example, the button from the category **BA | Base** is used below.



The Icon attribute can be set again using the `TcHmi.Functions.GetBalconPath()` function. This form of implementation allows the icon to be changed dynamically. The following attributes are available:

- IconRotation
- IconRotationSpeed
- IconRotationDirection
- IconColor (see Colors category)

In the code

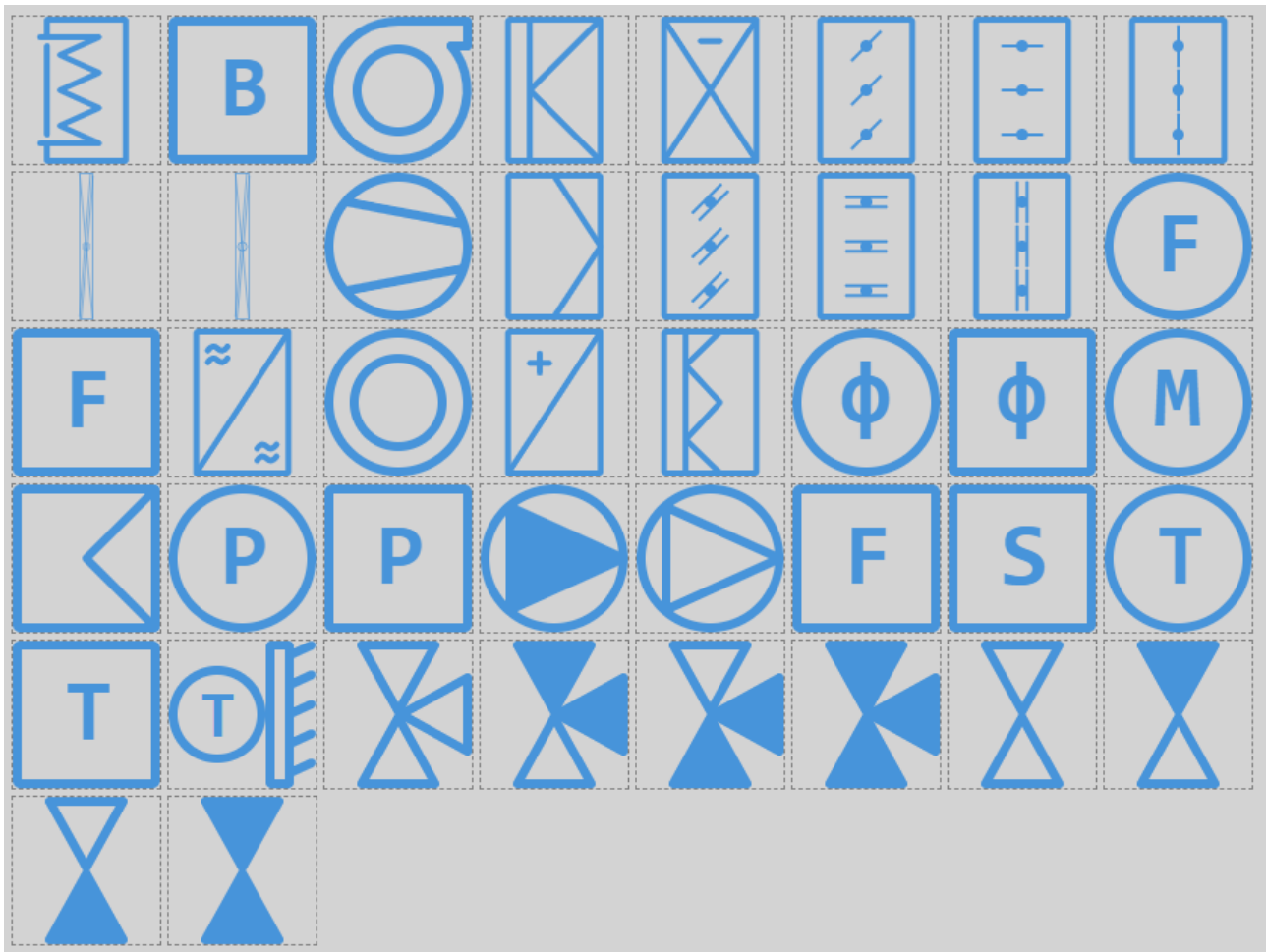
For using the icons in the code, e.g. when developing framework controls, the icon paths can be accessed even more easily. There are constants in the `TcHmi.BuildingAutomation.Icons` namespace that point to the respective icons in the NuGet package (e.g. `TcHmi.BuildingAutomation.Icons.HVAC.fan.svg`).

HVAC symbols

Symbols for P&I diagrams.



The icons are drawn with appropriate size ratios for an P&I diagram and are only shown differently in this listing.



Event symbols

Symbols to represent alarms, events or notifications.

Events

The concept of [alarms](#) [► 28] uses the following icons.

Icon	Name
	Alarm
	Fault
	Maintenance
	Notification
	Miscellaneous

The [events](#) [► 28] are displayed in different states.

Flag

The flag icons are displayed when one of the StatusFlags of an object is active.

Icon	Name
	InAlarm
	Fault
	Overridden
	OutOfservice

Priorities

Icon	Name
	LifeSafety
	Critical
	ManualLocal
	ManualRemote

Lock

Icon	Name
	High
	Medium

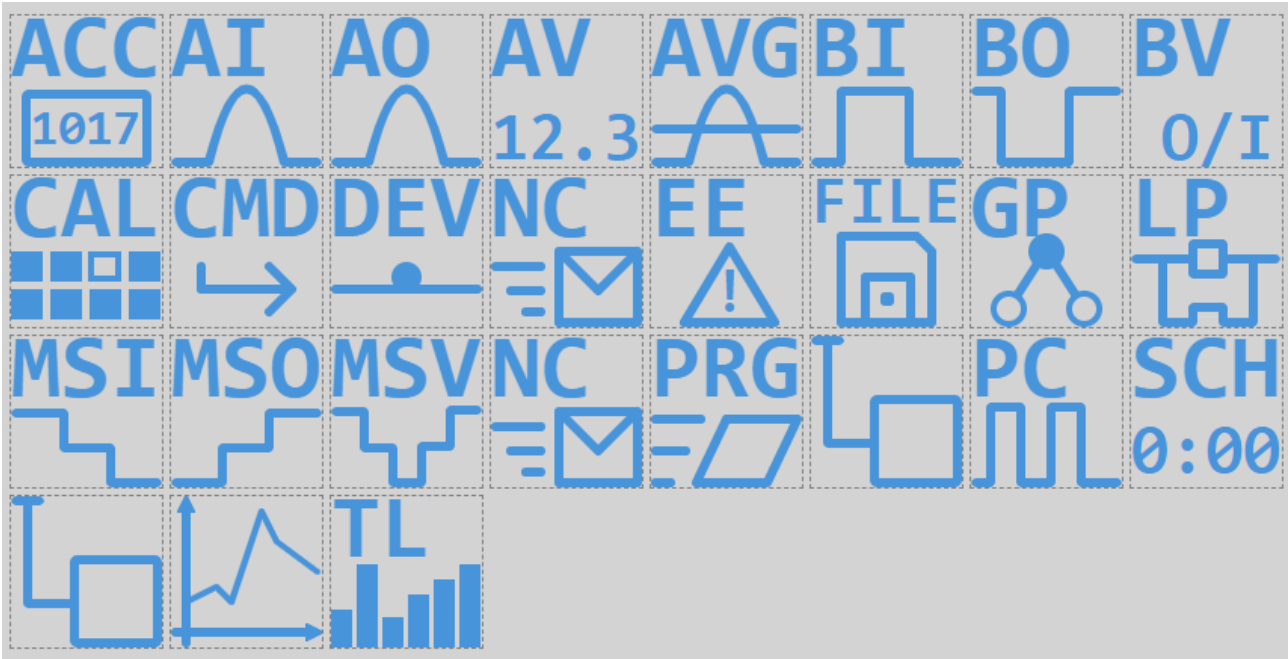
NodeType

Icon	Name
	Aggregate
	Buildings
	Building element
	Component
	Control cabinet
	Floor
	Information focus
	Property
	Plant

Icon	Name
	Room
	Technical system

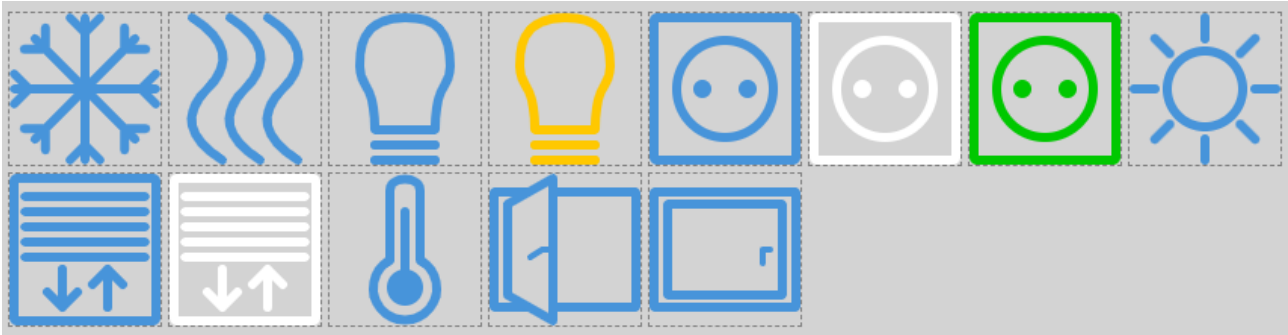
ObjectType

Symbols for the different object types.



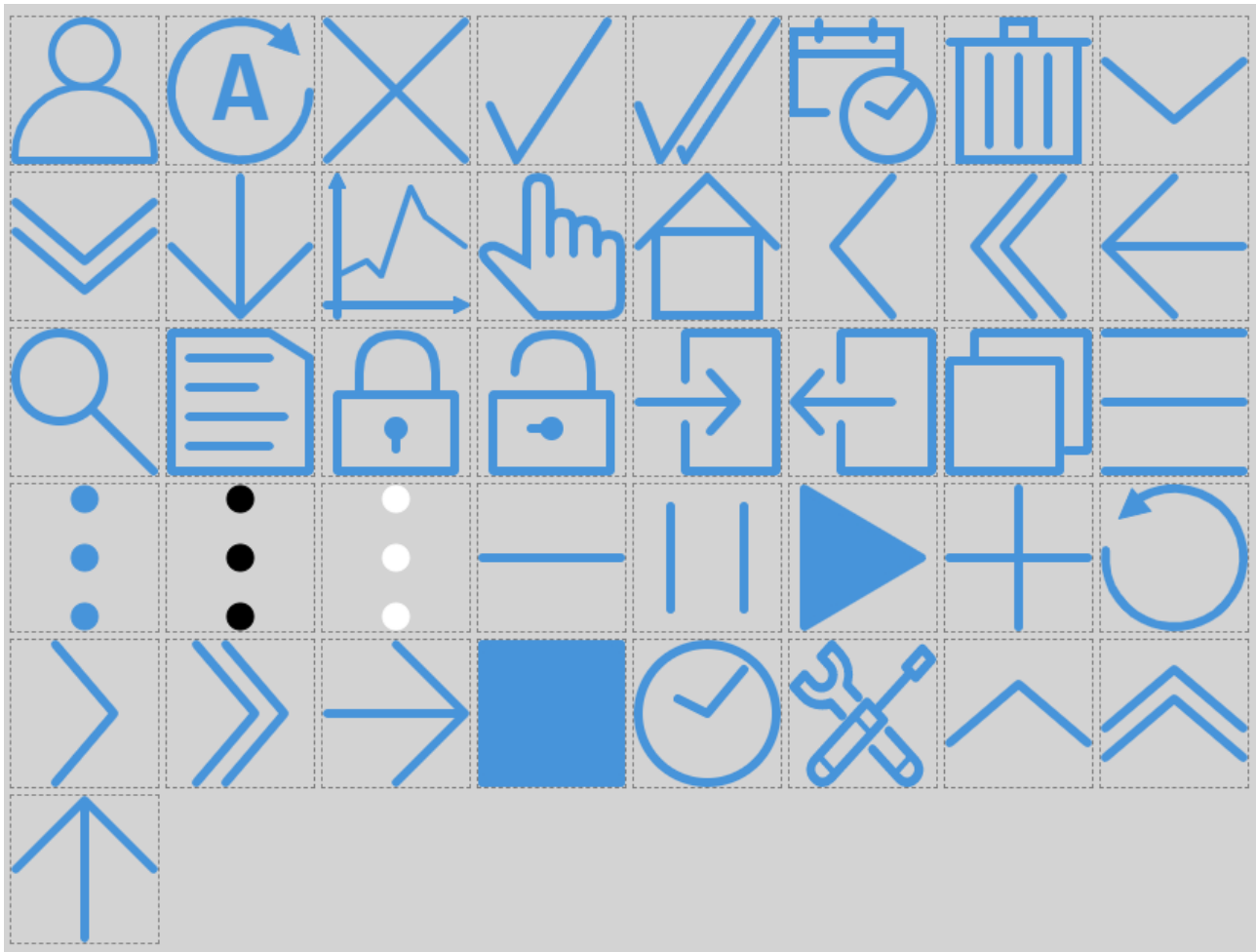
Room Automation

Symbols for room automation.



Standard

Standard symbols for visualizations.



6.2.1.3 Framework

The framework contains various classes, enumerations, interfaces and types that are used to create the [controls](#) [► 822]. Various helper methods facilitate server-client communication, access to different content and positioning in a control. There are **no controls** in the package because it is intended for use in framework control projects.

The framework is written in [TypeScript](#).

If the complete solution [TF8040](#) [► 8] is used, the framework also takes over many management functions to realize the [Generic](#) [► 44].

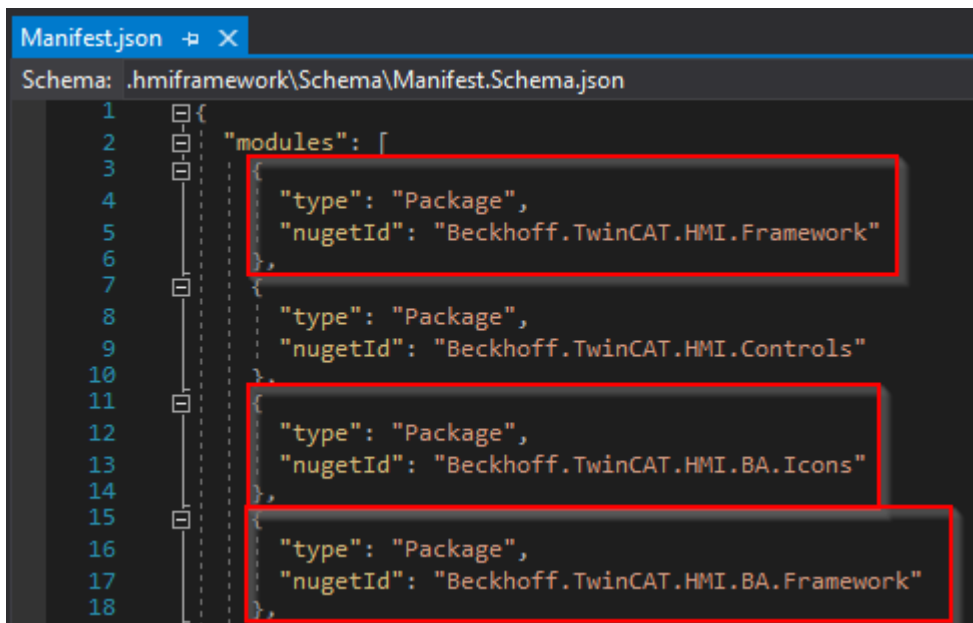
Installation

In order to use the framework, the NuGet package **Beckhoff.TwinCAT.HMI.BA.Framework** must be installed.

In addition, further functions are required, which originate from the following NuGet packages that are automatically installed as well:

- [Beckhoff.TwinCAT.HMI.BA.Icons](#) [► 984]
- [Beckhoff.TwinCAT.HMI.Framework](#)

Ensure that the files from the framework are loaded in the project. When used in a TcHmi framework project, the appropriate packages must be entered in the *Manifest.json*.



IntelliSense support for the framework in Visual Studio can be achieved by adding the following entries to *tsconfig.tpl.json*:

```
"include": [
  "$(Beckhoff.TwinCAT.HMI.Framework).InstallPath/TcHmi.d.ts",
  "$(Beckhoff.TwinCAT.HMI.BA.Icons).InstallPath/index.d.ts",
  "$(Beckhoff.TwinCAT.HMI.BA.Framework).InstallPath/index.d.ts"
]
```

6.2.1.3.1 BA

6.2.1.3.1.1 BAObjectHandler

The BaObjectHandler takes over the management of the BaObject.

Use

The implementation is done via the respective interfaces.

IUsesBaObject

Provision of the BaObjectHandler.

Requires:

- TcHmiBaFramework

```
module MyNamespace {
  export class MyClass<A extends MyClass.IAttributes = MyClass.IAttributes> extends
    TcHmi.BuildingAutomation.Base implements
    TcHmi.BuildingAutomation.BaObjectHandler.IUsesBaObject {

    public baObjectHandler: TcHmi.BuildingAutomation.BaObjectHandler;

    constructor(id: string, parent: TcHmi.BuildingAutomation.IBaseNode | null, attr?: A) {
      super(id, parent, attr);

      this.baObjectHandler = new TcHmi.BuildingAutomation.BaObjectHandler(this);
    }

    public processBaObject() {
      if (this.baObjectHandler.baObject == null) return;
      // do work
    }
  }

  export module MyClass {
    export interface IAttributes extends TcHmi.BuildingAutomation.Base.IAttributes {
```

```

        // optional additional attributes
    }
}

```

IFCUsesBaObject

Providing the BaObjectHandler with BaObject attribute for the TcHmi Designer.

Requires:

- TcHmiBaFramework
- TcHmiBaControls

```

module MyNamespace {
    export class MyControl extends TcHmi.BuildingAutomation.Controls.System.BaseControl
    implements TcHmi.BuildingAutomation.BaObjectHandler.IFCUsesBaObject {

        public baObjectHandler: TcHmi.BuildingAutomation.BaObjectHandler;

        constructor(element: JQuery, pcElement: JQuery, attrs:
TcHmi.Controls.ControlAttributeList) {
            super(element, pcElement, attrs);

            this.baObjectHandler = new TcHmi.BuildingAutomation.BaObjectHandler(this);
        }

        public processBaObject() {
            if (this.baObjectHandler.baObject == null) return;
            // do work
        }

        public setBaObject(p: TcHmi.BuildingAutomation.BA.BaBasicObject |
TcHmi.BuildingAutomation.BA.BaBasicObject.IBaBasicObjectAttributes | TcHmi.Symbol | null |
undefined): this {
            this.baObjectHandler.setBaObject(p);

            return this;
        }

        public getBaObject() {
            return this.baObjectHandler.baObject;
        }
    }
}

```

In the interface IFCUsesBaObject the setters and getters for the BaObject are already defined, so in the *Description.json* only the BaObject attribute has to be defined.

```

"attributes": [
    {
        "name": "data-tchmi-ba-object",
        "displayName": "BaObject",
        "propertyName": "BaObject",
        "propertySetterName": "setBaObject",
        "propertyGetterName": "getBaObject",
        "visible": true,
        "themeable": "None",
        "type": "tchmi:framework#/definitions/Symbol",
        "category": "BA",
        "description": "BA object of the control.",
        "requiredOnCompile": false,
        "readOnly": false,
        "bindable": true,
        "heritable": true,
        "defaultValue": null,
        "defaultValueInternal": null
    }
]

```

 Properties

Name	Description
loadChildren	Determines whether all child elements are loaded when setting the <i>BaObject</i> (as <i>BaView</i>).
loadTexts	Determines whether all texts are loaded when setting the <i>BaObject</i> (as <i>BaView</i>).
enableParentBaObjectProcessor	Determines whether the <i>BaObject</i> processor of the parent control is called.
baObject	Returns the <i>BaObject</i> .
baObjectSymbolExpression	Returns the <i>SymbolExpression</i> from the <i>BaObject</i> .
isLoadingBaObject	Checks if the <i>BaObject</i> is set but still loading.

 Methods

Name	Description
setBaObject	Sets the <i>BaObject</i> .
resolveBaObject	Resolves the passed information to create a <i>BaObject</i> .
readBaObject	Reads the <i>BaObject</i> from the server.
watchBaVariable	Adds a watch to the passed <i>BaVariable</i> . The Watch is also destroyed when the class is destroyed.
tryWatchBaVariable	Attempts to monitor a <i>BaVariable</i> .
tryWatchChildrenBaVariable	Attempts to monitor a <i>BaVariable</i> of a child element of a <i>BaView</i> .
watchValueRange	Monitors the value range of a specific variable.
checkBaObjectAccess	Checks the <i>OperationType</i> and the write access of the <i>BaVariable ValueRm</i> .

Events

Name	Description
onBaObjectChanged	Triggered when the <i>BaObject</i> has changed.

6.2.1.3.2 Helper**6.2.1.3.2.1 BaInterfaceHandler**

The *BaInterfaceHandler* is a helper class that handles the management of the *BaInterfaces* of a control.

With the help of the *BaInterfaces* it is possible to link several data points from one control with only one binding.

BaInterface

The *BaInterface* attribute is associated with a TcHmi symbol. This symbol must have a certain structure, which is described by the *BaInterface* definition.

Sample

The *BaInterface* of a checkbox has the following structure:

```
export type BaInterface = {
  state: boolean,
  stateFeedback?: boolean,
  activeText?: string,
  inactiveText?: string,
}
```

For the connected symbol to be valid for the *BaInterface* of the checkbox, the symbol must have the described subsymbols with the corresponding data type.

BaInterfaceDefinition

A control that uses the BaInterfaceHandler must implement the interface `IUsesBaInterface<T>` [► 1001].

The type parameter `T` should describe the structure of the used *BaInterfaces*.

Here using the data type `Checkbox.BaInterface`:

```
export class Checkbox implements IUsesBaInterface<Checkbox.BaInterface>
```

Since TypeScript types cannot be interpreted by JavaScript at runtime, it is necessary to define the `BaInterfaceDefinition` as a constant.

Here the data type of the respective element must be specified and optional elements of the *BaInterfaces* can be defined via the property *optional*.

Optional elements are not considered in the later validation of the *BaInterfaces*. If required elements are not found during validation, error messages will occur and the *BaInterface* will not be processed further.

```
export const BaInterfaceDef: BaInterfaceDefinition<BaInterface> = {
  state: {
    type: 'boolean'
  },
  stateFeedback: {
    type: 'boolean',
    optional: true
  },
  activeText: {
    type: 'string',
    optional: true
  },
  inactiveText: {
    type: 'string',
    optional: true
  }
};
```

It is possible to specify multiple data types.

```
command: {
  type: ['number', 'boolean']
}
```

BaInterfaceSymbolNames

So that the BaInterfaceHandler can access the corresponding sub-symbols of the *BaInterfaces*, it is necessary to specify a symbol name for each element of the interface. The symbol name corresponds to the name of the variable in the connected function block / structure. This can be done, for example, by a variable `Checkbox.BaInterfaceSymbolNames` :

```
export let BaInterfaceSymbolNames: BaInterfaceSymbolNames<BaInterface> = {
  state: {
    symbolName: 'State'
  },
  stateFeedback: {
    symbolName: 'StateFeedback'
  },
  activeText: {
    symbolName: 'ActiveText'
  },
  inactiveText: {
    symbolName: 'InactiveText'
  }
}
```

The symbol name should be able to be adapted for later use if the symbol is called differently in the connected function block/structure (e.g. `symbolName: "bState"`).

It is also possible to use subsymbols. If the state is e.g. in a structure or a function block within the connected symbol, it can be accessed (e.g. `symbolName: "Command::bValueRm"`).

There are two possibilities for overwriting these default symbol names, which are explained below.

Initialization

The `BaInterfaceHandler` must be initialized in the `__prevInit()` method of the control. The following steps must be performed during initialization.

```
// Create instance of BaInterfaceHandler
this.baInterfaceHandler = new BaInterfaceHandler<Checkbox.BaInterface>(this);
// Set the BaInterfaceDefinition
this.baInterfaceHandler.baInterfaceDefinition = Checkbox.BaInterfaceDef;
// Set the default symbol names
this.setBaInterfaceSymbolNames(Checkbox.BaInterfaceSymbolNames);
```

Use

The implementation of the setter for the `BaInterface` attribute can look like this:

```
public setBaInterface(p: BaInterfaceSymbol<Checkbox.BaInterface> | null | undefined): this {
    this.baInterfaceHandler.setBaInterfaceSym(p, () => {
        // do work with the validated BaInterface
    });
    return this;
}

public getBaInterface() {
    return this.baInterfaceHandler.getBaInterfaceSym();
}
```

Here you can see that for the setter the method `setBaInterfaceSym()` of the `BaInterfaceHandler` is used. In addition to the symbol, this also expects a `Processor` method that is called when the `BaInterfaceSymbol` has been validated or the `BaInterfaceSymbolNames` have changed.

The setters and getters for the `BaInterfaceSymbolNames` attribute can be implemented as follows:

```
public setBaInterfaceSymbolNames(p: BaInterfaceSymbolNames<Checkbox.BaInterface> |
    BaInterfaceSymbolNamesDesigner | null | undefined): this {
    if (p != null)
        this.baInterfaceHandler.updateSymbolNames(BaInterfaceHandler.convertToBaInterfaceSymbolNames
(p));
    return this;
}

public getBaInterfaceSymbolNames(): BaInterfaceSymbolNames<Checkbox.BaInterface> | null |
    undefined {
    return this.baInterfaceHandler.baInterfaceDescription;
}
```

Methods

The most important methods of the `BaInterfaceHandler` are described below.

Name	Description
<code>hasSubSymbol</code>	Checks if the <code>BaInterface</code> has a specific subelement. This method must be used if optional elements are to be read or written.
<code>writeSubSymbol</code>	Writes the value of a subelement.
<code>watchSubSymbol</code>	Creates a subscription for a subelement.
<code>updateSymbolNames</code>	Updates the symbol names of the <code>BaInterface</code> subelements.
<code>convertToBaInterfaceSymbolNames</code>	Converts the data type <code>BaInterfaceSymbolNamesDesigner</code> to <code>BaInterfaceSymbolNames</code> . If the data type <code>BaInterfaceSymbolNames</code> is already passed to the method, no conversion is performed and the object is returned directly.

6.2.1.3.2.1.1 *IUsesBaInterface<T>*

The interface `IUsesBaInterface<T>` must be implemented by controls that are to use the `BaInterface` attribute.

Structure

The interface prescribes the following properties and methods for the corresponding control:

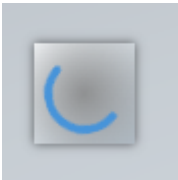
```

export interface IUsesBaInterface<I extends IBaInterfaceStructure> extends IJQueryNode {
  /**
   * Handler for the BaInterface.
   * @category Public
   */
  baInterfaceHandler: BaInterfaceHandler<I>;
  /**
   * Sets the BaInterface attribute.
   * @category Attribute setter
   * @param p New BaInterface or null.
   * @returns Returns the control.
   */
  setBaInterface(p: Symbol | null | undefined): this
  /**
   * Gets the BaInterface attribute.
   * @category Attribute getter
   * @returns Returns the BaInterface attribute.
   */
  getBaInterface(): Symbol | null | undefined;
  /**
   * Sets the BaInterfaceSymbolNames attribute.
   * @category Attribute setter
   * @param p New BaInterfaceSymbolNames or null.
   * @returns Returns the control.
   */
  setBaInterfaceSymbolNames(p: BaInterfaceSymbolNames<I> | null | undefined): this;
  /**
   * Gets for the BaInterfaceSymbolNames attribute.
   * @category Attribute getter
   * @returns Returns the BaInterfaceSymbolNames attribute.
   */
  getBaInterfaceSymbolNames(): BaInterfaceSymbolNames<I> | null | undefined
}

```

6.2.1.3.3 BusyHandler

The BusyHandler is a class of [TcHmiBaFramework \[► 996\]](#). It provides information about whether a control is still busy (e.g. waiting for information from the TwinCAT HMI Server). Recognizable by the loading animation.



Functions

logTimerResultsOfControl

Checks which actions on a control lead to loading times.

Namespace: `TcHmi.BuildingAutomation.BusyHandler.logTimerResultsOfControl`



Is only applicable to controls that implement the `TcHmi.BuildingAutomation.BusyHandler.IBusyHandler` interface.

Preparation

Before use, the recording of timer results must be activated. This can be done, for example, at the project level with a code-behind function:

```

let TcHmi.EventProvider.register('onInitialized', function (e, data) {
  e.destroy();
  TcHmi.BuildingAutomation.BusyHandler.RecordTimerResults = true;
})

```

Use

The call is made in the console of the browser after a control has been loaded. The ID of the control is necessary for this.

```
TcHmi.BuildingAutomation.BusyHandler.logTimerResultsOfControl('DieControlId')
```

Evaluation

The results of the timers are available in the console window:

```
TcHmi.BuildingAutomation.BusyHandler.logTimerResultsOfControl('Checkbox_Sp_2')
Timer results of 'Checkbox_Sp_2' ▼ {timerResults: Array(1), children: {}} ⓘ
  ▼ children:
    ▼ Checkbox_Sp_2-ba-fc:
      ▼ timerResults: Array(3)
        ▶ 0: {requiredTime: '0.19 s', action: 'Reading BaObject'}
        ▶ 1: {requiredTime: '0.00 s', action: "Watching BaVariable 'ePresentValue'"}
        ▶ 2: {requiredTime: '0.10 s', action: "Watching value range of 'ePresentValue'"}
          length: 3
        ▶ [[Prototype]]: Array(0)
      ▶ [[Prototype]]: Object
    ▼ timerResults: Array(1)
      ▶ 0: {requiredTime: '0.19 s', action: 'Waiting for busy children'}
        length: 1
      ▶ [[Prototype]]: Array(0)
    ▶ [[Prototype]]: Object
```

It is recognizable what the control was busy with.

In this case, a sub-element with ID *Checkbox_Sp_2-ba-fc* was waited for, which was busy loading the BaObject most of the time.

6.2.1.4 Functions

6.2.1.4.1 OpenTrendCollectionView

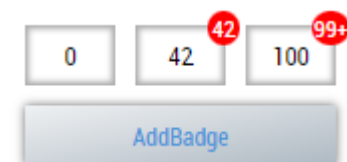
Opens a dialog for [Monitoring trend collections](#) [► 47].

Use

The function is called event-driven and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

6.2.1.4.2 AddBadge

Adds a number display to a control in the upper right corner. The display is optimized for TcHmiBa Controls.



Use

The function is called event-driven and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

Control

```
tchmi:framework#/definitions/Control
```


Control for the badge extension (see [Control](#)).

Count

```
tchmi:general#/definitions/Number
```

Number to be displayed. Is visible when the value is greater than 0. Restriction to "99+" from three-digit values (see [Number](#)).

6.2.1.4.3 ConvertHexToRgbColor

Converts a hex color to an RGBA color.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

Hex

```
tchmi:general#/definitions/String
```

Hex color to convert (see [String](#)).

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

[RGBA color](#) [[► 880](#)].

6.2.1.4.4 ConvertHslToRgbColor

Converts an HSL color to an RGBA color.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

HSL

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.HSLColor
```

HSL color to be converted (see [HSL-Color](#) [[► 879](#)]).

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

[RGBA color](#) [[► 880](#)].

6.2.1.4.5 ConvertRgbToHexColor

Converts an RGBA color to a hex color.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

RGBA

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA color to be converted (see [RGBAcolor](#) [► 880]).

Return value

```
tchmi:general#/definitions/String
```

Hex color (see [String](#)).

6.2.1.4.6 ConvertRgbaToHslColor

Converts an RGBA color to an HSL color.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

RGBA

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA color to be converted (see [RGBAcolor](#) [► 880]).

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.HSLColor
```

HSL color [► 879].

6.2.1.4.7 ConvertRgbaToSolidColor

Converts an RGBA color to a TcHmi solid color.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

RGBA

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA color to be converted (see [RGBAcolor](#) [► 880]).

Return value

```
tchmi:framework#/definitions/SolidColor
```

TcHmi Solid color (see [SolidColor](#)).

6.2.1.4.8 ConvertSolidToRgbaColor

Converts a TcHmi solid color to an RGBA color.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

Solid

```
tchmi:general#/definitions/SolidColor
```

TcHmi Solid color to be converted (see [SolidColor](#)).

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA color [[▶ 880](#)].

6.2.1.4.9 ConvertUnitToString

Converts the value of a BA.Unit enumeration into a unit.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter

Unit

```
tchmi:general#/definitions/Number
```

Partial path from the icon (see [Number](#)).

Return value

```
tchmi:general#/definitions/String
```

Unit (see [String](#)).

6.2.1.4.10 GetBaIconPath

Returns the full path of an icon for a partial path.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

The partial path for the function parameter corresponds to the structure of the NuGet package Beckhoff.TwinCAT.HMI.BA.Icons, as it can be seen in the [TwinCAT HMI Gallery Explorer](#).

Valid call options:

```
GetBaIconPath('HVAC/Cooler.svg')
```

```
GetBaIconPath('HVAC/cooler.svg')
```

```
GetBaIconPath('HVAC/Cooler')
```

```
GetBaIconPath('HVAC/cooler')
```

Parameter**Path**

```
tchmi:general#/definitions/String
```

Partial path from the icon (see [String](#)).

Return value

```
tchmi:general#/definitions/String
```

Full path from the icon (see [String](#)).

6.2.1.4.11 GetCurrentUserNames

Returns the name of the active user (no authentication corresponds to `__SystemGuest`) or null if unknown (e.g. when loading).

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Return value

```
tchmi:general#/definitions/String
```

User name (see [String](#)).

6.2.1.4.12 GetFadeColor

Calculates a color between two colors to a corresponding value.

Use

The function can be called event-driven as well as via the [Function Binding](#) and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter**StartColor**

```
tchmi:framework#/definitions/SolidColor
```

Start color of the cross-fade (see [SolidColor](#)).

StartValue

```
tchmi:framework#/definitions/Number
```

Start value of the cross-fade (see [Number](#)).

EndColor

```
tchmi:framework#/definitions/SolidColor
```

End color of the cross-fade (see [SolidColor](#)).

EndValue

```
tchmi:framework#/definitions/Number
```

End value of the cross-fade (see [Number](#)).

Value

tchmi:framework#/definitions/Number

Value used to calculate the color. (see [Number](#)).

Return value

tchmi:general#/definitions/SolidColor

Calculated color (see [SolidColor](#)).

6.2.1.4.13 LoadUserDependentContent

Loads a specified content when the specified user is logged in.

Use

The function is called event-driven and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter**HostRegion**

tchmi:framework#/definitions/TcHmi.Controls.System.TcHmiRegion

The host region to which the content should be loaded (see [Region](#)).

UserContents

tchmi:framework#/definitions/TcHmi.BuildingAutomation.UserContents

Collection of users with their associated content (see [UserContents](#) [► 928]).

StoreLastContent

tchmi:general#/definitions/Boolean

Determines whether the last content will be reloaded on the user's next visit (see [Boolean](#)).

6.2.1.4.14 OpenDialogWindow

Opens a content page in a dialog.

Use

The function is called event-driven and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter**Content**

tchmi:framework#/definitions/ContentPath

Path to the content to be displayed.

Buttons

tchmi:framework#/definitions/TcHmi.BuildingAutomation.DialogWindowButtons

Buttons for closing or confirming the dialog (see [DialogWindowButtons](#) [► 925]).

Modal

```
tchmi:general#/definitions/Boolean
```

Determines whether the dialog is opened modally or not (see [Boolean](#)).

Scrolling

```
tchmi:framework#/definitions/ScrollMode
```

Determines whether the content can be scrolled (see [ScrollMode](#)).

Headline

```
tchmi:general#/definitions/String
```

Title of the dialog (see [String](#)).

Width

```
tchmi:general#/definitions/Number
```

Width of the dialog (see [Number](#)).

WidthUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit of the width of the dialog.

Height

```
tchmi:general#/definitions/Number
```

Height of the dialog (see [Number](#)).

HeightUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit of the height of the dialog.

6.2.1.4.15 OpenFileDialog

Opens an instance of the Legend control in a dialog.

Use**Use**

The function is called event-driven and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter**Modal**

```
tchmi:general#/definitions/Boolean
```

Determines whether the dialog is opened modally or not (see [Boolean](#)).

Buttons

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.DialogWindowButtons
```

Buttons for closing or confirming the dialog (see [DialogWindowButtons](#) [► 925]).

Width

```
tchmi:framework#/definitions/MeasurementValue
```

Width of the dialog.

WidthUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit of the width of the dialog.

Height

```
tchmi:framework#/definitions/MeasurementValue
```

Height of the dialog.

HeightUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit of the height of the dialog.

EntryWidth

```
tchmi:framework#/definitions/MeasurementValue
```

Width of an entry.

EntryWidthUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit of width of an entry.

TabPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position
```

Position of the tabs (see [Position](#) [► 925]).

IconDataSource

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataSource
```

Selection of entries to be displayed (see [IconDataSource](#) [► 828]).

IconDataCustom

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList
```

List with additional entries (see [IconDataList](#) [► 828]).

6.2.1.4.16 OpenLightZoneDialog

Opens an instance of the [LightZone](#) [► 982] control in a dialog.

Use

The function is called event-driven and can be found in the [Actions and Conditions](#) editor under *Functions > BuildingAutomation*.

Parameter**BaObject**

```
tchmi:framework#/definitions/Symbol
```

BaObject for the control (see [Symbol](#)).

6.2.1.4.17 UpdateObjectInfo

Updates the object information of all objects in all BA devices.

Use

The function is called event-driven and can be found in the Actions and Conditions editor under *Functions > BuildingAutomation*.

6.2.1.4.18 UseBaObjectsInUserControl

Description

By default, a UserControl reads out the complete structure behind a linked parameter, which can lead to a high communication traffic. With a BaObject (e.g. BaView) this can quickly become a lot of data. This function reduces the server communication to a minimum when using a BaObject as a parameter in a UserControl.

Use

The parameter from the UserControl for the BaObject must be of type *Symbol*. By this definition the parameter is not read, but only passed on. Likewise the name of the parameter must be *BaObject!*

Edit/Define Parameters

Description:

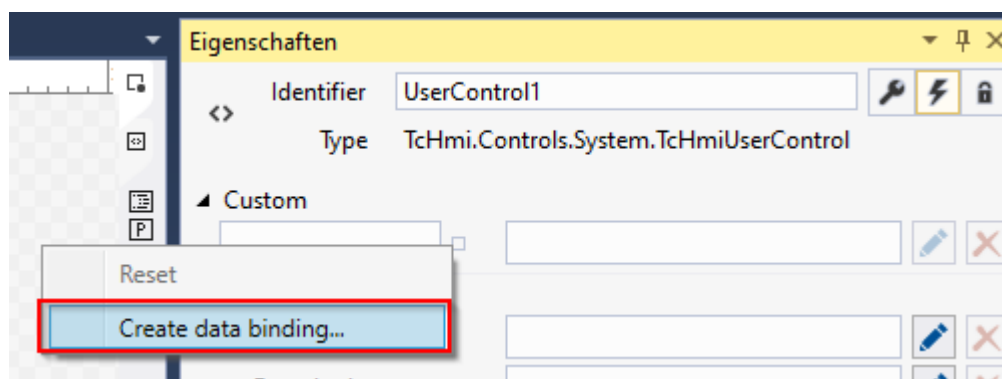
Parameters:

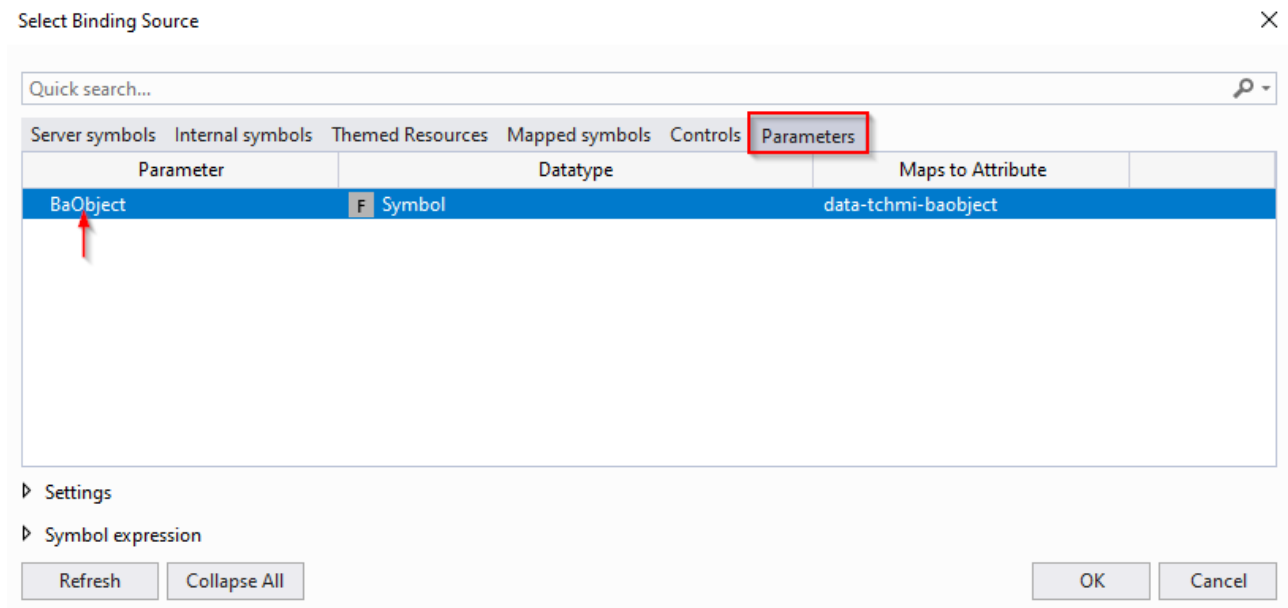
Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility	Category
BaObject	F Symbol			☒	☒	



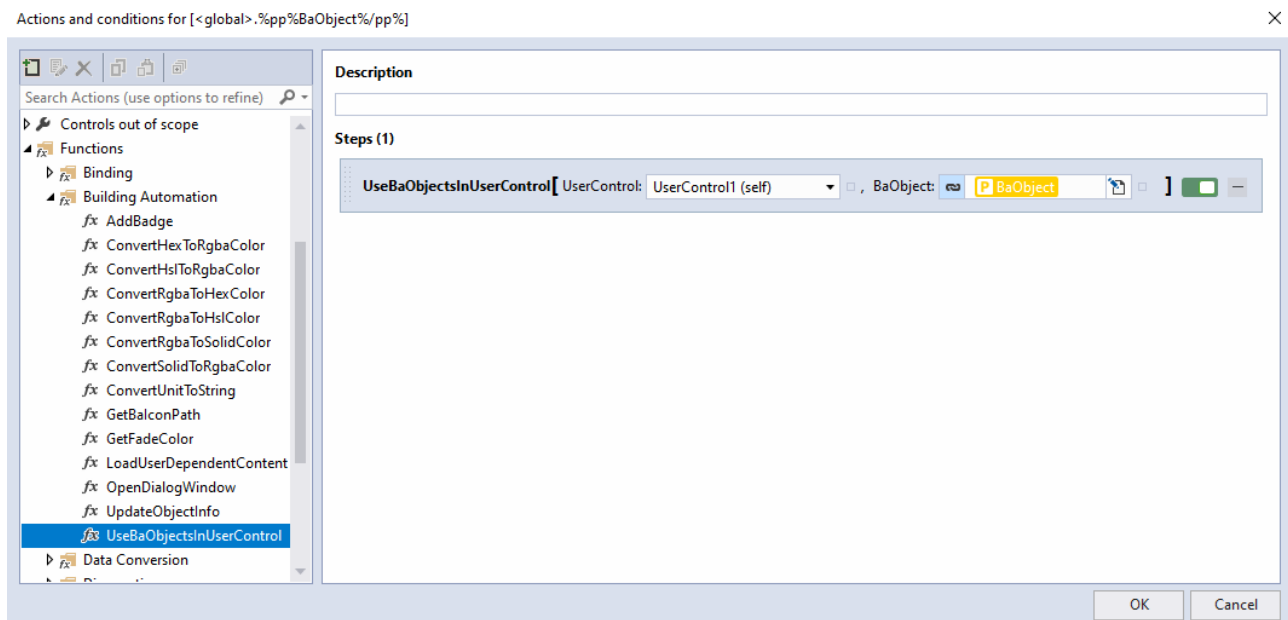
In this way, access to the underlying symbols of the BaObject is no longer possible.

The function should be called whenever the BaObject parameter has been changed. For this purpose, a new event *.BaObject* is created.





In the configuration window of the event, the function is selected in the folder "Functions > BuildingAutomation > UseBaObjectInUserControl".



The controls in the UserControl are then linked to the BaObject or its subelements via the identifiers of the controls.

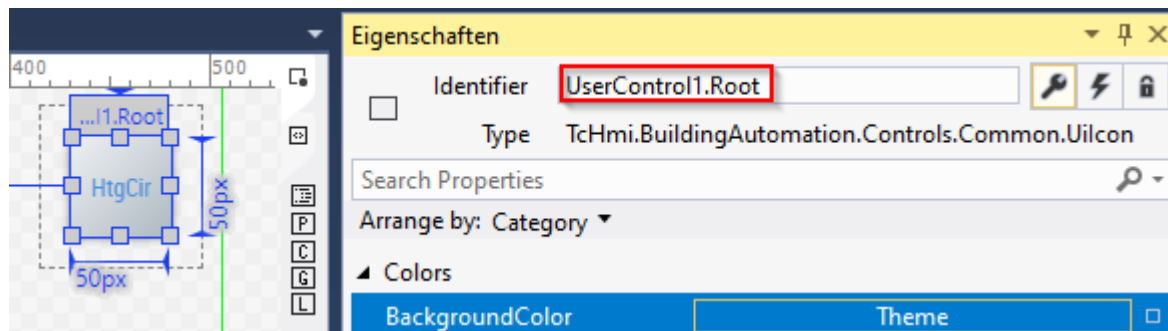


It is also possible to connect UserControls within a UserControl with BaObjects. The identifier must be assigned as explained in the next paragraph. Likewise, the inner UserControl must then also have the parameter *BaObject*.

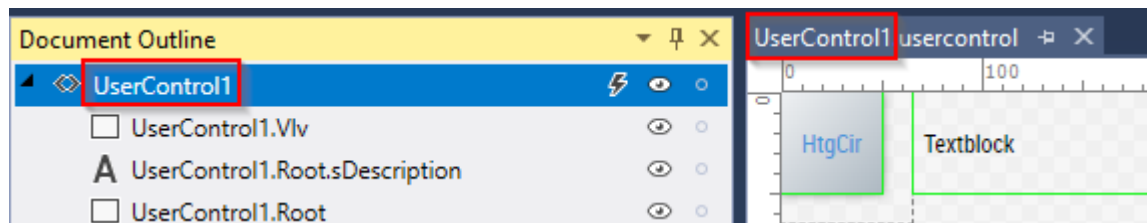
Linking

BaObject

To use the BaObject directly, the control must have the identifier *\$UserControlName\$.Root*. *\$UserControlName\$* is replaced with the name of the UserControl.



Make sure that the root element of the UserControl has the same name as the file of the UserControl.

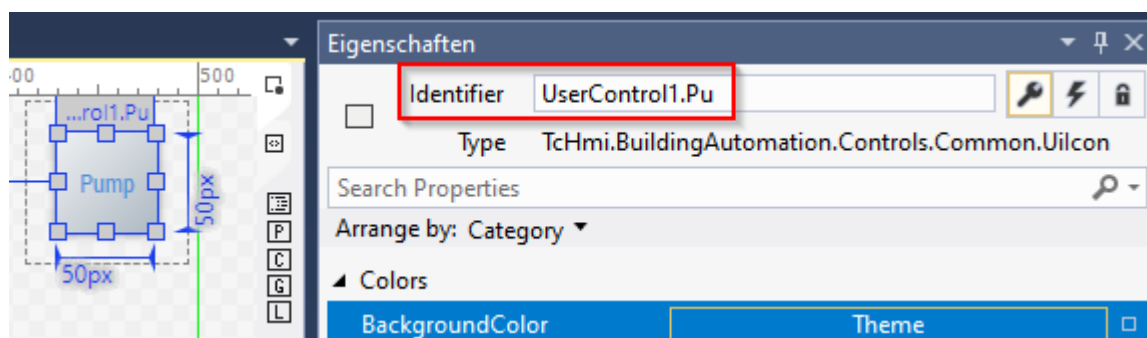


Subelements of BaObject

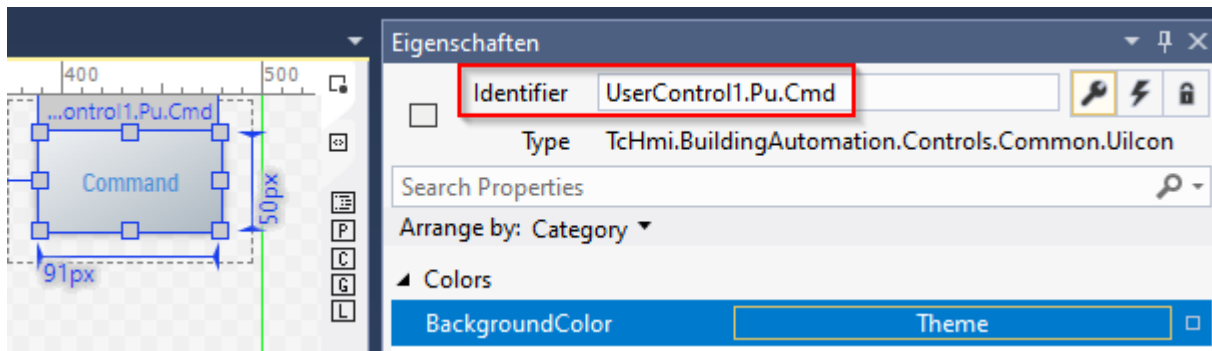
To use a subelement, the control must carry the symbol path as identifier. Explanation using the example of a BaView with the following structure.



Access to the element *Pu* is via the symbol path *\$UserControlName\$.Pu* as identifier.



Access to the element *Cmd* is via the symbol path *\$UserControlName\$.Pu.Cmd* as identifier.



Parameters from BaObject

Access to the parameter of an element is done via the symbol path.

Example:

- *\$UserControlName\$.Root.sDescription*
- *\$UserControlName\$.Pu.Cmd.bPresentValue*
- *\$UserControlName\$.OpMod.OpModMan.nPresentValue*
- *\$UserControlName\$.HtgLmt.Sp.fPresentValue*

If the control is of type [Checkbox](#) [► 872], [ComboBox](#) [► 868] or [InputBox](#) [► 839], then the value of the parameter is written to the PLC after the user interaction is terminated.

Without BaObject or BaParameter

If the control or UserControl within the UserControl is not to work with a BaObject or BaParameter, then the identifier of the control must not contain *\$UserControlName\$*.

Parameter

UserControl

```
tchmi:framework#/definitions/TcHmi.Controls.System.TcHmiUserControl
```

The UserControl in which the BaObjects are to be used. Mostly this parameter is the UserControl itself (self).

BaObject

```
tchmi:general#/definitions/Object
```

The UserControl parameter to which the BaObject is connected from the outside must be linked with this parameter.

6.2.1.5 Global settings

Global settings of the different TcHmiBa NuGet packages are described below.

TcHmiBaFramework

DoublePressDuration

Defines the time that may elapse between two clicks until the two clicks are recognized as a double click.

```
TcHmi.EventProvider.register('onInitialized', function (e, data) {
    e.destroy();
    TcHmi.BuildingAutomation.Components.Button.DoublePressDuration = 200;
})
```

DisableNodeTypelcons

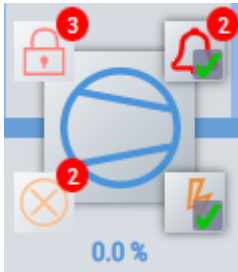
Defines whether the NodeTypelcons are used in the ProjectNavigationTextual.

```
TcHmi.EventProvider.register('onInitialized', function (e, data) {
    e.destroy();
    TcHmi.BuildingAutomation.Navigation.ProjectNavigationList.DisableNodeTypeIcons = true;
})
```

MaximumEventCondition

Defines the maximum [EventCondition](#) [► 92], which is displayed in the HMI.

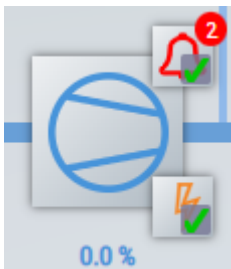
Default setting displays all [events](#) [► 989], [flags](#) [► 990], [priorities](#) [► 992] and [locks](#) [► 992].



If, for example, only [events](#) [► 989] are to be displayed, the default value can be overwritten in the CodeBehind as follows.

```
TcHmi.EventProvider.register('onInitialized', function (e, data) {
    e.destroy();
    TcHmi.BuildingAutomation.BA.EventHelper.MaximumEventCondition = TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther;
})
```

Thereupon the control from above is as follows:



TcHmiBaControls

EnablePopUp

Defines whether the global keyboard should be displayed automatically when an input field is selected.

The default value is *true*.

```
TcHmi.EventProvider.register('onInitialized', function (e, data) {
    e.destroy();
    TcHmi.BuildingAutomation.Helper.Keyboard.EnablePopUp = false;
})
```

6.2.1.6 Project templates

Project templates are intended to make it easier to get started with a *TcHmiBa* project by pre-installing required dependencies and providing additional elements (e.g. navigation).

Installation

The installation of the project templates into the available development environments is **not** performed by the TF8040 but by the batch file *InstallProjectTemplates.bat*.



After installing the TwinCAT 3 HMI and TF8040, the batch file is located in the directory:
C:\TwinCAT\Functions\TF8040 Building Automation\HMI\ProjectTemplates

A double click on the *InstallProjectTemplates.bat* executes it and a console window opens.

```

C:\Windows\system32\cmd.exe
This batch program installs the project template TcHmiBaProject to VS2017, VS2019 and TcXaeShell.
.
1 Datei(en) kopiert.
Installed TcHmiBaProject in VS 2017 and TcXaeShell.
.
1 Datei(en) kopiert.
Installed TcHmiBaProject in VS 2019.
.
Drücken Sie eine beliebige Taste . . .

```



The project templates currently have some limitations. The following section describes these in more detail and how they can be remedied.

User management

In order to use the integrated user management of TF8040, it is necessary that certain user groups and matching users exist in the TcHmi project. The project template does **not** include these users. The program *CreateDefaultUserManagement.exe* can be used to create the required settings for a project.



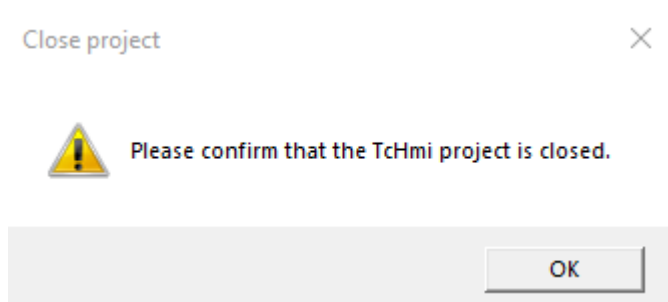
More detailed information on the various user groups can be found [here \[► 15\]](#).



After installing the TwinCAT 3 HMI and TF8040, the program is located in the directory:
C:\TwinCAT\Functions\TF8040 Building Automation\HMI\Tools\CreateDefaultUserManagement

✓ Double-clicking on the *CreateDefaultUserManagement.exe* will execute it.

1. First select the HMI project.
2. Close the project (if not already closed).



3. Check output

```
Please select TcHmi project file:
Selected project file: 'C:\temp\TcHmiBaProject1\TcHmiBaProject1.hmiproj'

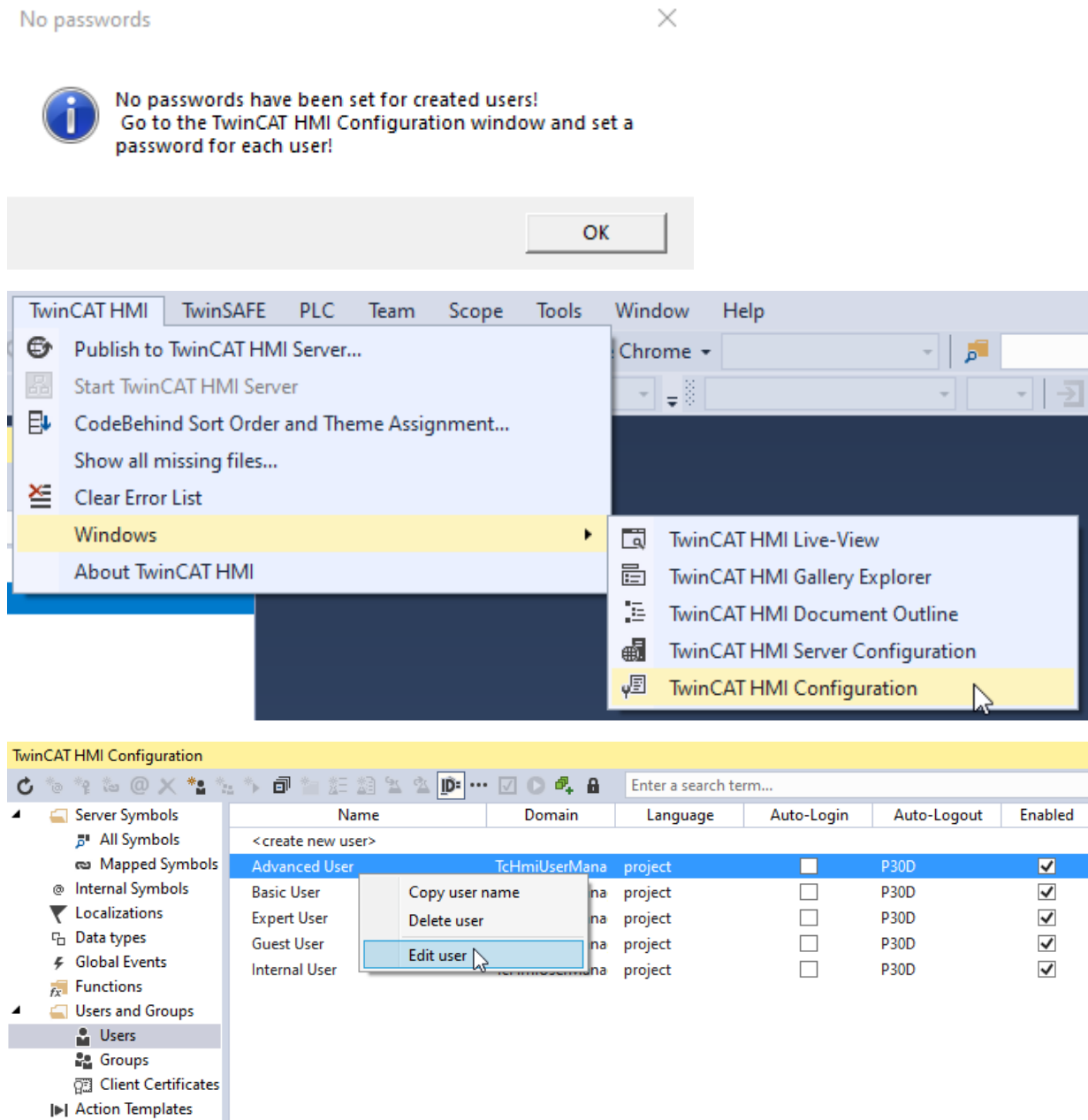
Start editing TcHmiUserManagement.Config.default.json
'Advanced User' added to Users.
'Default User' added to Users.
'Expert User' added to Users.
'Internal User' added to Users.
Finished editing TcHmiUserManagement.Config.default.json

Writing to TcHmiUserManagement.Config.default.json ...
Finished writing to TcHmiUserManagement.Config.default.json.

Start editing TcHmiSrv.Config.default.json
'Advanced User' added to UserGroupUsers.
'Default User' added to UserGroupUsers.
'Expert User' added to UserGroupUsers.
'Internal User' added to UserGroupUsers.
'Advanced' added to UserGroups.
'Default' added to UserGroups.
'Expert' added to UserGroups.
'Internal' added to UserGroups.
Finished editing TcHmiSrv.Config.default.json

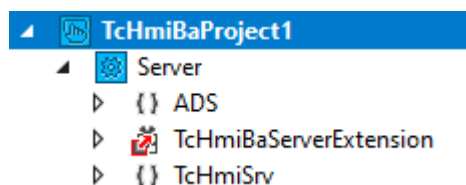
Writing to TcHmiSrv.Config.default.json ...
Finished writing to TcHmiSrv.Config.default.json.
```

- Open the project and create passwords for the created users.

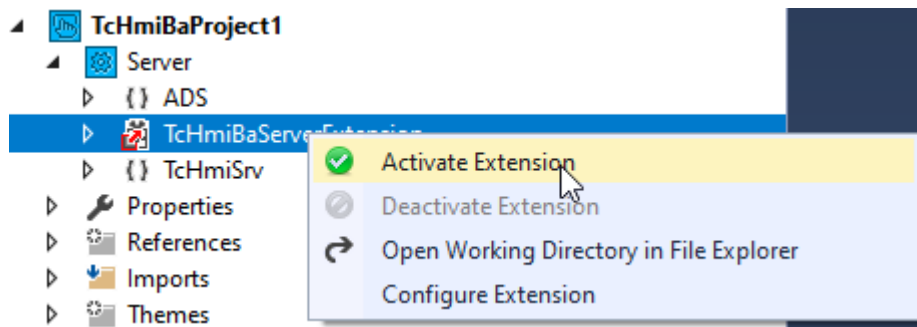


TcHmiBaServerExtension

The server extension will start inactive because the configuration files are not loaded with the project template.

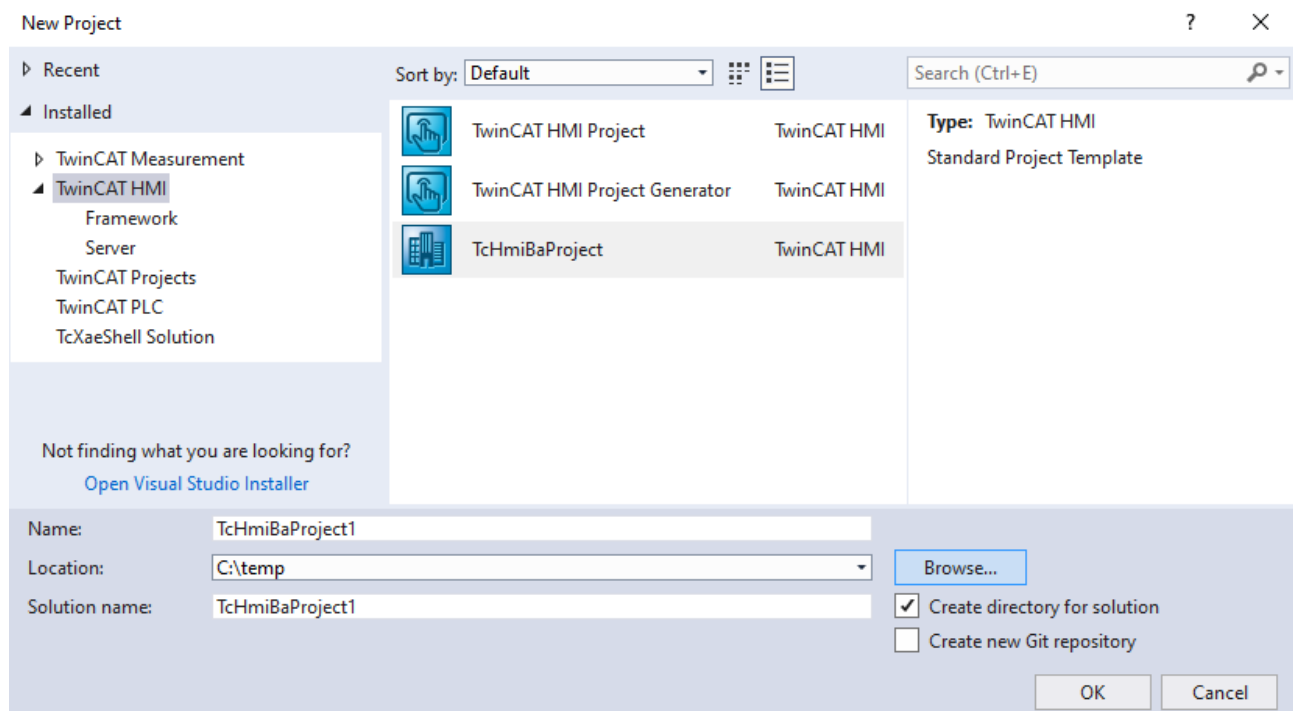


Therefore a manual start is necessary.



6.2.1.6.1 TcHmiBaProject

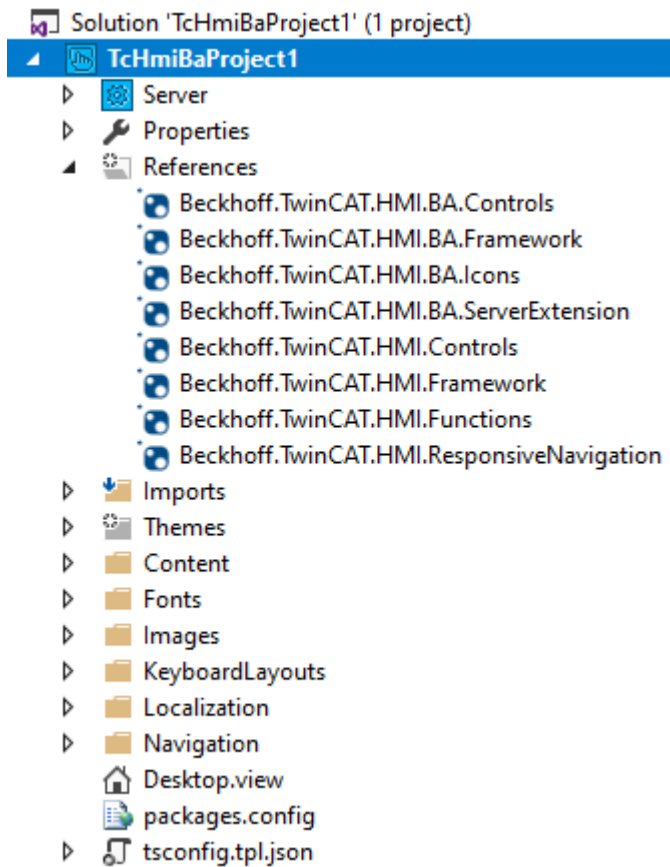
The project template is used to create a TcHmi project quickly and easily.



The [notes \[1016 \]](#) for project templates must be observed.

Contents

Overview of the contents of the project template.

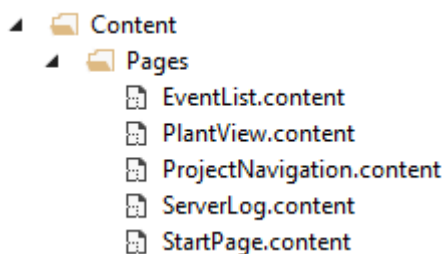


References

The project template contains the following NuGet packages:

- Beckhoff.TwinCAT.HMI.BA.Icons
- Beckhoff.TwinCAT.HMI.BA.Framework
- Beckhoff.TwinCAT.HMI.BA.Controls
- Beckhoff.TwinCAT.HMI.BA.ServerExtension
- Beckhoff.TwinCAT.HMI.Framework
- Beckhoff.TwinCAT.HMI.Controls
- Beckhoff.TwinCAT.HMI.Functions
- Beckhoff.TwinCAT.HMI.ResponsiveNavigation

Content



To make it easier to get started, the project template already contains some *content* pages, such as a navigation in the [header](#) [[▶ 1021](#)].

The following pages are required for the header to function correctly:

- EventList.content
- ServerLog.content

- StartPage.content

The navigation in the header refers to the following pages:

- PlantView.content
- ProjectNavigation.content

These pages are customizable, and further pages can be added.



If pages are added, renamed or removed, the attribute `MenuData ▶ 1022` of the header must be updated.

6.2.1.6.1.1 Header

The header is a `UserControl` and serves as an entry point for users. It offers an easy way to configure a navigation for the HMI and several other features.

Features

The features at a glance (from left to right).

- Logo (1)
- Responsive navigation (2)
- User settings and further information (3)
- Event list (4)
- Building information (5)
- Outdoor temperature (6)
- Date and time (7)



User settings and further information

In this menu, users can make various settings that affect them. In addition, diagnostic data and the server log can be displayed.

Close X

Language	User	Others
English (US)	Logout	TcHmiBaServerExtension diagnostics
German	Reload	Calendar list
	Theme	> Server log

Event list

The event list can be called up via the button with the bell symbol. The button also displays the number of active events, if the number was linked via the EventCount attribute.

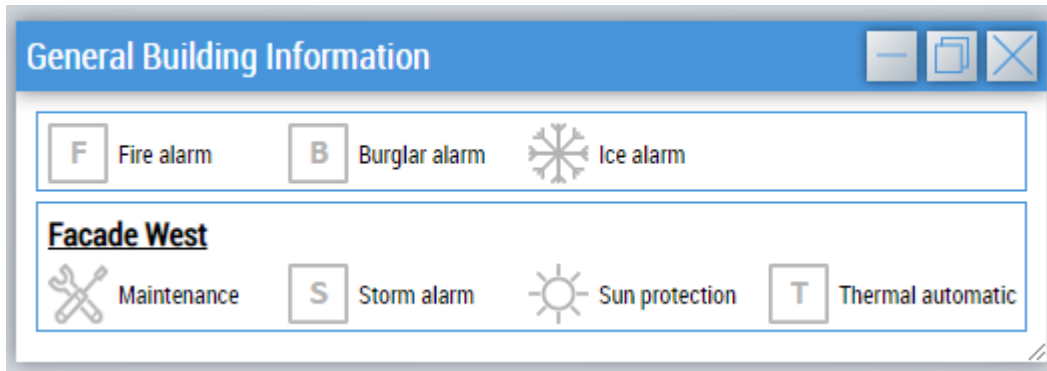


Building information

The button with the info symbol is the BuildingInformation control.



It can be used to open a window with information about the building and the facades.



Attributes

The control inherits from `TcHmiControl` and thus has the same attributes. In addition, there are the following attributes.

Logo

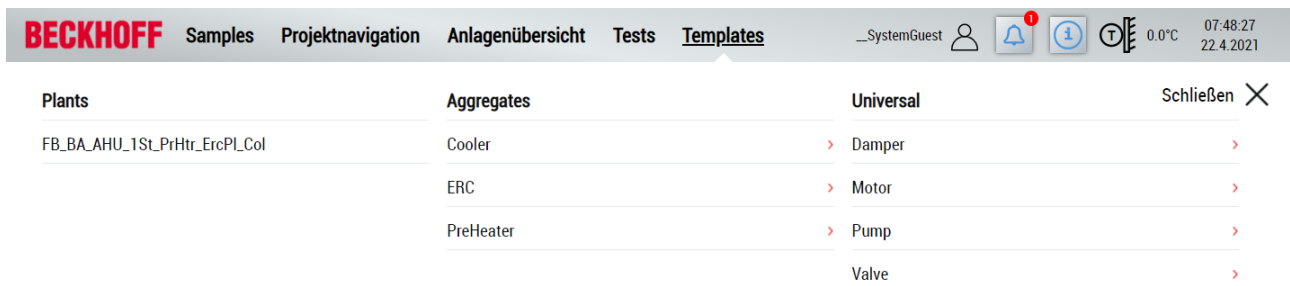
`tchmi:framework#/definitions/ContentPath`

Path specification to the image with the logo that is displayed at the start of the header.

MenuData

`TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationBar.MenuItemList`

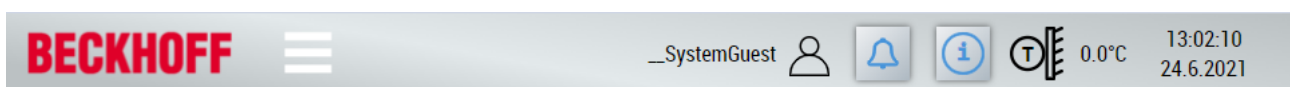
Defines the structure and hierarchy of the navigation. Entries in the header can be linked directly to content pages, or submenus can be created that expand when selected.



SwitchBreakpoint

`tchmi:general#/definitions/Number`

Determines at which pixel width the navigation changes to the Hamburger symbol.



EventCount

`tchmi:general#/definitions/Number`

Here you can link a symbol that contains the number of active events in the event list. This number is then displayed in a badge on the button for the event list.

**CurrentTemperature**

tchmi:general#/definitions/Number

Current temperature to be displayed in the header. Typically, the outdoor temperature is displayed here.

CurrentTemperatureUnit

tchmi:general#/definitions/String

Unit of the current temperature.

CloseMenu

tchmi:general#/definitions/Boolean

If the attribute has the value TRUE, the header menus can be closed (this can be changed during runtime).

NavContent

TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent

Here a control of type *TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent* must be linked, in which the content of the responsive navigation is displayed.

UserContent

TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent

Here a control of type *TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent* must be linked, in which the content of the user menu is displayed.

TargetRegion

tchmi:framework#/definitions/TcHmiRegion

Here the *TcHmiRegion* must be linked, which is used to display the content pages selected from the menu.

EventContent

tchmi:framework#/definitions/ContentPath

Content on which the event list is located.

StartPage

tchmi:framework#/definitions/ContentPath

Start page of the HMI. This page is loaded into the TargetRegion when you click the logo.

7 Tools



For new projects we strongly recommend to use the version 5 of TF8040!

7.1 Site Explorer

The Terminal Explorer clearly displays all objects in the project structure. It supports the commissioning and adjustment of plants.

System requirements

Windows:

- Windows 7 or higher
- Microsoft .NET Framework v6.0.14 or higher.

The current version of the .NET Runtime can be checked in the Control Panel under Programs and Features.

Name	Herausgeber	Installiert am	Größe	Version
Local Administrator Password Solution	Microsoft Corporation	04.05.2020	129 KB	6.2.0.0
Microsoft .NET Core Runtime - 2.1.28 (x64)	Microsoft Corporation	17.06.2021	89,6 MB	2.1.28.30015
Microsoft .NET Core SDK 2.1.526 (x64)	Microsoft Corporation	20.08.2021	478 MB	2.1.526
Microsoft .NET Runtime - 5.0.16 (x64)	Microsoft Corporation	02.05.2022	91,0 MB	5.0.16.31117
Microsoft .NET Runtime - 6.0.14 (x64)	Microsoft Corporation	15.02.2023	95,3 MB	6.0.14.32123

Beckhoff:

One of the following components must be installed to use Terminal Explorer:

- TC1000 | TC3.1 ADS
- TE1000 | TC3 Engineering

Application

Access rights

During the first application, the user is prompted to apply a role [► 15]. In this way, the access rights are firmly fixed.

Dialog boxes

Project



- Object [► 28]
- Object name
- Description
- State

- Value source
Indicates a setpoint or display value.
- Active event [▶ 28]
- Overridden
- Out of service
- Active priority

- Manual override
- Current Value
- Trend
- Commissioning condition

Displays the properties of the selected entry.

- **Control**
Among other things, the properties of a control list components (such as services or supplements) that are executed at runtime.

Properties

▼ Misc	
Active	True
Address	5.57.161.218.1.1:851
▼ Communication	
Connections	
Connection	Connected
Description	CX-39A1DA
▼ Device information	
▼ Components	
BACnet Revision	14
BACnet Stack	4.0.1.13
Tc3_BA2	4.9.0.36
Tc3_BA2_Common	2.1.14
Tc3_BACnetRev14	4.1.21.7
Terminal Server	1.3
Enabled	True
Namespace	Tc3_BA2
Name	CX-39A1DA (5.57.161.218.1.1:851)
▼ Statistics	
Objects	244
Plants	7
Read variables	0

Name

Example: Properties of a control

- **Objects**

Properties

DPAD

Level 6

Identification

Identifier AV:102

Subject information

Hash 1284

Identifier AVSp

Index 1

Type hash 42323

Misc

Data class Analog

Node type Function

Operational type Setpoint

Purpose Operation

References

Symbol

Symbol name AVSp

Symbol path Tc3_BA2.MAIN.General.AVSp

Title

Description Samples - Demonstration - Objects -- General

Description (Instance) Sample AV Setpoint

Labels

Object name Smpl_Demo_Obj~~~General++A++AVSp

Object name (Instance) AVSp

Data class

Example: Properties of an analog object.

Events

Active [events](#) [► 28] are listed in the event overview:

Project Events Event History

Device	TimeStamp	Path	Type	Symbol	Name	Event
5.57.161.218.1.1:851	16.12.2019 07:56:30	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	01.01.0001 00:00:00	Tc3_BA2.MAIN_TestDPAD.HtgCir02.Pump3.MntnSwi	BI	MntnSwi	PUM03_MTN001	
5.57.161.218.1.1:851	01.01.0001 00:00:00	Tc3_BA2.MAIN_TestDPAD.HtgCir02.Pump3.ThOvrid	BI	ThOvrid	PUM03_SM001	
5.57.161.218.1.1:851	01.01.0001 00:00:00	Tc3_BA2.MAIN_TestDPAD.HtgCir02.Pump3.Dst	BI	Dst	PUM03_SM002	

Sonstiges

Increment 1719

State eIndicatd

TimeStamp 16.12.2019 07:54

Type eAlarm

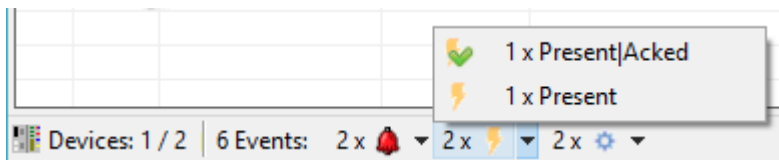
Increment

In addition, a summary of all active events is displayed in the status bar at the bottom.

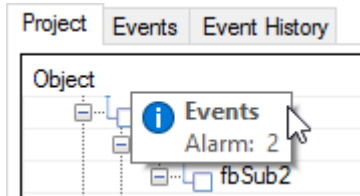
For each event type, the highest priority event symbol is indicated:

Devices: 1 / 2 | 6 Events: 2 x 2 x 2 x

Individual states of the combined [events](#) [► 28] can be viewed via the drop-down menu:



In the project view, a summary of all active events [► 28] of a view can also be displayed by moving the mouse over the entry:



Event history

All occurred events [► 28] of a view are listed in the event history:

Device	TimeStamp	Path	Type	Symbol	Name	Event
5.57.161.218.1.1:851	16.12.2019 07:54:21	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:25	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:26	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:30	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:31	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:35	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:36	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:40	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	
5.57.161.218.1.1:851	16.12.2019 07:54:41	Tc3_BA2.MAIN_Simulation.BI	BI	BI	BI	

Sonstiges
 Increment 1739
 State eIndicatd
 TimeStamp 16.12.2019 07:55
 Type eAlarm
Increment

Functions

Acknowledge

Events [► 28] can be acknowledged from different views via the context menu:

Object	Object Name	Description	Man.	State	Value	Trend	Commissioning State
BI_Proc	BI_Proc	Sample BI (Process Data)			Normal		Unknown
BI_Raw	BI_Raw	Sample BI (Raw Data)			Normal		Unknown
BI_DetbIO	BI_DetbIO	Sample BI Disturb (IO)			Ausgelöst		Unknown
BI_Dsp_SwIO	BI_Dsp_SwIO	BI Schalter Display (IO)			Ausgelöst		Unknown
BO_CmdProc	BO_CmdProc	BO Befehl (Process Data)			Aus		Unknown
BO_CmdRaw	BO_CmdRaw	BO Befehl (Raw Data)			Aus		Unknown
BO_CmdIO	BO_CmdIO	BO Befehl (IO)			Aus		Unknown
BODisp_CmdIO	BODisp_CmdIO	BO Befehl Display (IO)			Aus		Unknown

Example: Acknowledge fault in the project view.

Device	TimeStamp	Path	Type	Symbol	Name	Event	Lock
CX-39A1DA (5.57.161.218.1.1:851)	30.06.2021 06:37:49	Tc3_BA2.MAIN.General.BV	BV	BV	Smpl_Demo_Obj~~~General++B++BV		
CX-39A1DA (5.57.161.218.1.1:851)		Tc3_BA2.MAIN.General.BI_Proc	BI_Proc	BI_Proc	Smpl_Demo_Obj~~~General++B++BI_Proc		
CX-39A1DA (5.57.161.218.1.1:851)		Tc3_BA2.MAIN.General.BI_Raw	BI_Raw	BI_Raw	Smpl_Demo_Obj~~~General++B++BI_Raw		
CX-39A1DA (5.57.161.218.1.1:851)	30.06.2021 06:37:49	Tc3_BA2.MAIN.General.BI_DetbIO	BI_DetbIO	BI_DetbIO	Smpl_Demo_Obj~~~General++B++BI_DetbIO		

Example: Acknowledge alarm in the event overview.

Navigate to an object

The context menu can be used to navigate directly to the selected object [► 28] (in the project view [► 1024]):

Project	Events	Event History		
	Device	TimeStamp	Path	Type
▶	CX-39A1DA (5.57.161.218.1.1:851)		A2.MAIN.General.BV	BV 0/3
	CX-39A1DA (5.57.161.218.1.1:851)		A2.MAIN.General.BI_Proc	BI
	CX-39A1DA (5.57.161.218.1.1:851)		A2.MAIN.General.BI_Raw	BI
	CX-39A1DA (5.57.161.218.1.1:851)	30.06.2021 06:37:48	Tc3_BA2.MAIN.General.BI_DstbIO	BI

Example: Navigation to the selected object from the event history.

Commissioning:

Commissioning states of individual objects can be adjusted in the corresponding column:

Project

Events

Event History

Object	Object Name	Description	Man.	State	Value	Trend	Commissioning State
AVSp	AVSp	AVSp			21,5	☒	
AI_Proc	AI_Proc	AI_Proc			0	☐	Checked
AI_Raw	AI_Raw	AI_Raw			0	☐	Defect device
AI_IO	AI_IO	AI_IO			0	☐	Checked
AI_IO_2_10V	AI_IO_2_10V	AI_IO_2_10V			0	☐	Defect wiring

Description	Man.	State	Value	Trend	Commissioning State
Sample AV Setpoint 2			3 °C	☐	
Sample AI (Process Data)			20 %	☐	Checked
Sample AI (Raw)				☐	Checked
Sample AI (IO)				☐	Defect I/O
Sample AI (IO)				☐	Checked
Sample AI Temperature				☐	Checked
Sample AI Display (IO)				☐	Unknown
Analog Befehl (Process)				☐	Unknown

Symbol:

MAIN.General.AI_Raw.eCommissioningState

Checked

Ok

Cancel

Example: Interface for entering the commissioning state of an object.

Details

Commissioning details of selected objects are displayed in the properties.



Subsequent editing is possible for some details.

Value	Trend	Commissioning State
3 °C	☐	
20 %	☐	Checked
0 %	☐	Defect device
0 %	☐	Defect I/O
0 %	☐	Checked
0 °C	☐	Checked
0 %	☐	Unknown

Properties	Trend
Commissioning Details Comment State Timestamp User The "On" LED remains dark. Defect device 30.06.2021 13:07 MD	

Example: Commissioning details of a defective field device.

• Online

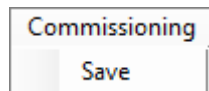
The following details are stored online (per [object](#) [▶ 281]) in the controller.

- State: Current commissioning state.
- **Offline**

The following details are stored offline (per [object \[► 28\]](#)) in the site settings.

- Timestamp: Time of the last change of the commissioning state (if the change was made using Site Explorer).
- Comment: Optional comment.

Menu



Save: Saves the commissioning details of all connected devices.



At the latest when the connection to a device is terminated, the commissioning details are automatically saved.

Trend

Various information can be read in the project view:

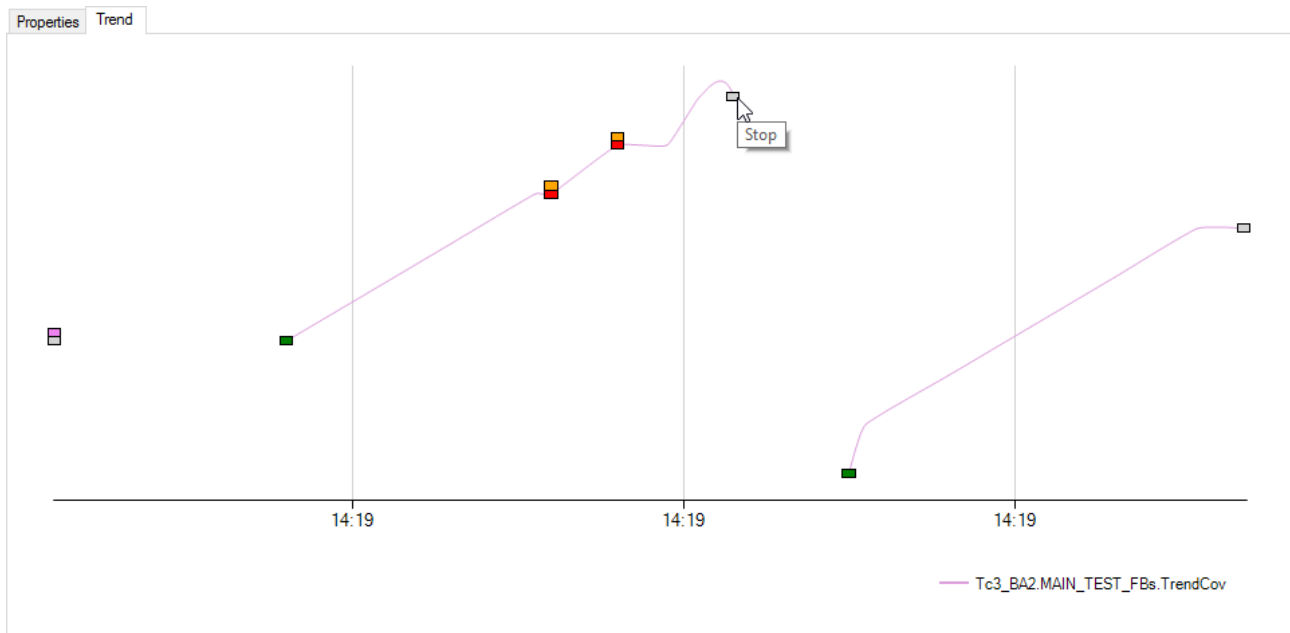
1. Indication of a referencing [Trend object \[► 160\]](#).
2. Current number of data sets of the referencing [Trend object \[► 160\]](#).
3. Start or stop recording the current value of an [object \[► 28\]](#) (**Online trend**).
4. Display or remove the log buffer of a [Trend object \[► 160\]](#) (**Offline trend**).

Object	Object Name	Description	Man.	State	Value	Trend	Commissioning State
Tmd	Tmd	Tmd					
PollAV	PollAV	Referred object for polling			3 °C	☐ -	
TrendPoll	TrendPoll	Sample Trend				☐ [1]	
CovAI	CovAI	Referred analog object for COV			3 %	☐ *	Unknown
TrendCovAI	TrendCovAI	Sample Analog Trend COV				☐ [100]	
CovBI	CovBI	Referred binary object for COV			Off	☐ *	Unknown
TrendCovBI	TrendCovBI	Sample Binary Trend COV				☐ [100]	
CovMI	CovMI	Referred multistate object for COV				☐ *	Unknown

Example: Representation of different trends.

Offline trend

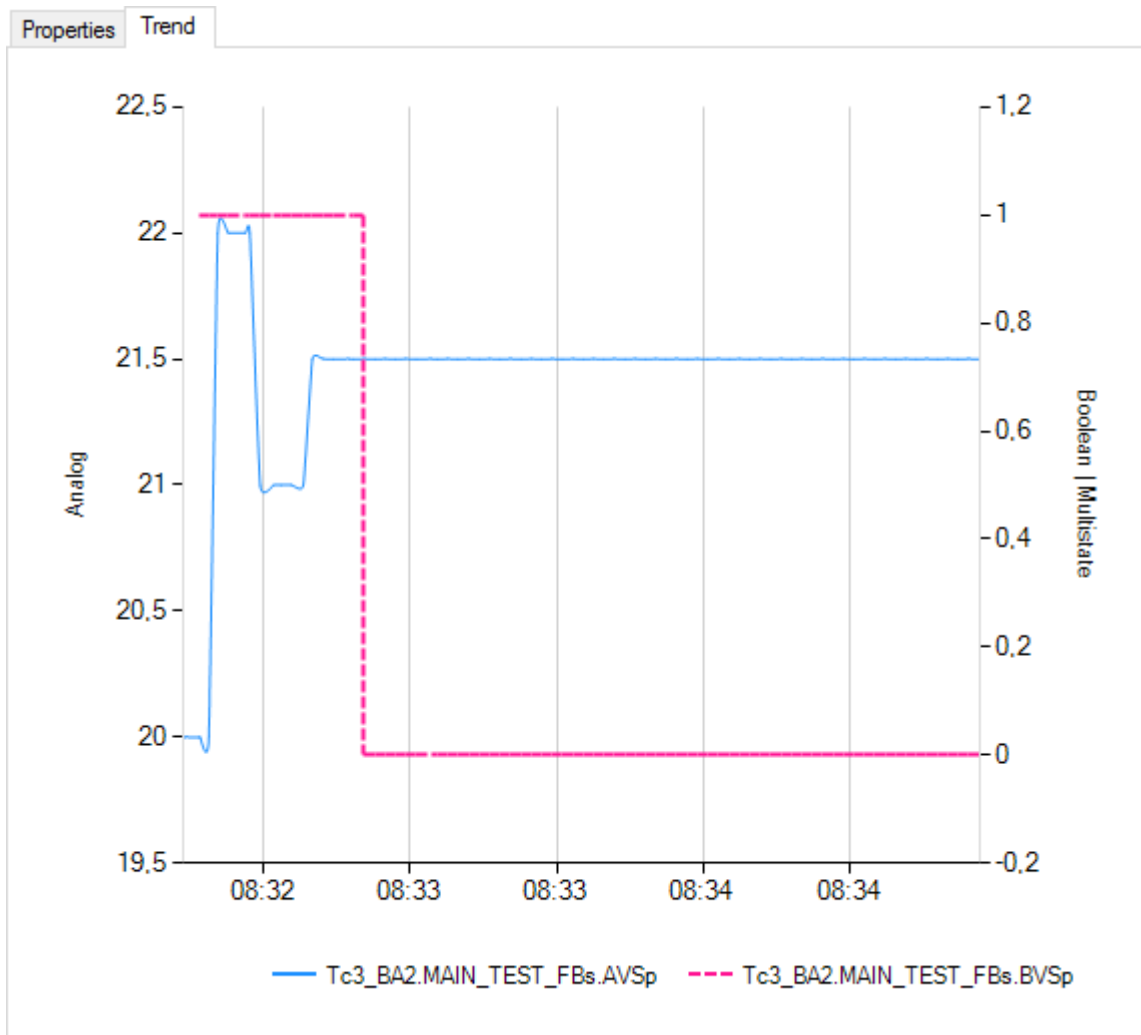
The **offline trend** reads the log buffer of a [Trend object \[► 160\]](#) and displays it in the **Trend view**.



Entries are not updated automatically!

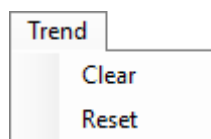
Online trend

The **online trend** records the current value of an [object](#) [► 28] in the **Trend view**.

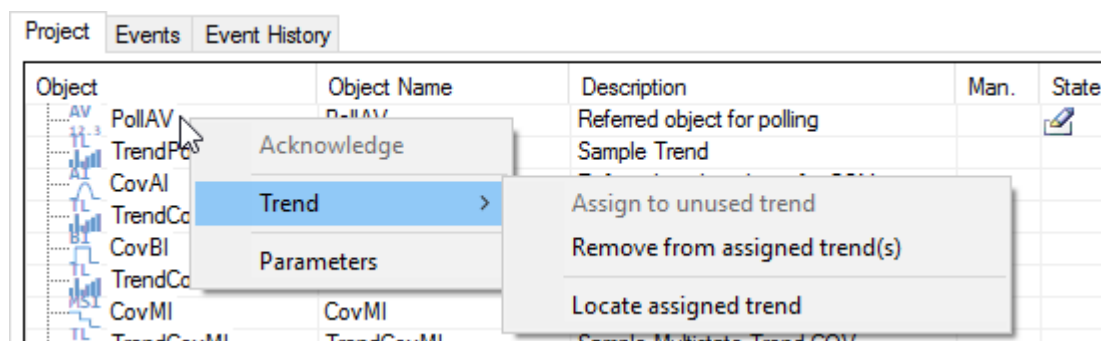


NOTICE**Loss of data**

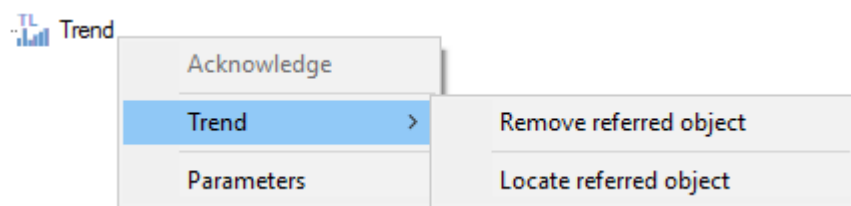
Recorded values are not saved after ending the recording or closing the trend view!

Menu

Name	Description
Cleaning	Removes all entries from the trend view (equivalent to restarting the current recordings).
Reset	Not only removes all entries from the Trend view, but also stops recording all <u>objects</u> [► 28].

Referencing

Name	Description
Assign free trend	Assigns an <u>object</u> [► 28] to the next, unused <u>Trend object</u> [► 160].
Remove reference from assigned trend object(s)	Removes the <u>Objects</u> [► 28] reference from all referencing <u>Trend object(s)</u> [► 160].
Navigate to assigned trend	Navigates to the assigned <u>Trend object</u> [► 160] (in the project view).

Referencing trend objects

Name	Description
Remove referred object	Removes the current <u>object</u> [► 28] reference.
Locate referred object	Navigates to the assigned <u>object</u> [► 28] (in the project view).

Editing values

Current values (if writable) can be edited directly in the project view.

Project Events Event History

Object	Object Name	Description	Man.	State	Value	Trend
General	General	General objects				
A	A					
B	B					
M	M					
MV	MV					
MVDsp	BAS01001					
MVSp	BAM01001					
MVPushButton	MVPushButton					
MI StateProc	MI StateProc					

Edit value

Symbol:
MAIN.General.MVSp.nValueRm

Auto

Ok
Cancel

Value	Trend
Aus	☐
Auto	☐
Auto	☐
Aus	☐
Aus	☐

Editing parameters

Project Events Event History

Object	Object Name
M	M
MV	MV
MVDsp	BAS01001
MVSp	BAM01001
MVPushButton	MVPushButton
MI_State	MI_State
MI_State	MI_State
MI_State	MI_State
MIDsp_S	MIDsp_S
MODisp_IO	MODisp_IO

Acknowledge

Trend >

Parameters

MODisp_IO (Sample MO Befehl (IO))

Object	Value	R/W
☐ Acknowledge	Inactive	A A
☐ Enable Plant-Lock	No Lock	A I
☐ Event Detection Enable	Disable	A E
☒ Event Enable		
☐ Event Class ID	182	A E
☒ Event Transition Text		
☐ Event State	eNormal	* -
☒ Status Flags		
☐ Reliability	No Fault Detected	A -
☐ Out Of Service	Disable	A E
☒ State Text		
☐ State Count	5	* -
☐ Present Value	Stufe 2	* -
☐ Default Value	Aus	A E
☐ Tag		* A
☐ Assign as Trend-Reference	None	E E
☐ Device Type		A E
☐ Mapping-Mode	Index 1:N	A E
☐ Feedback Mapping-Mode	Index 1:N	E E
☐ Commissioning State	Unknown	A E
☐ Symbol Path	Tc3_BA2.MAIN....	E -
☐ Symbol Name	MODisp_IO	E -
☐ Instance ID	100	A -
☐ Object Name	Smpl_Demo_Obj...	A I
☐ Description	Samples - Demon...	* E
☐ Manual Enable	Disable	A A
☐ Manual Value	<Invalid #0>	A A

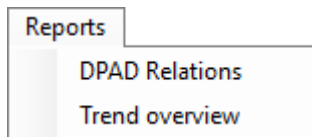
Access rights: The column **R|W** (Read, Write) lists the required access rights per parameter.

Role	Abbreviation
Default	*
Advanced	A
Expert	E
Internal	I
Locked	-

Reports

Reports are intended to provide a general overview of a project. Furthermore, the following requirements can also be met:

- Project documentation for the operator
- Project status tracking



All reports can be exported to the following formats:

- *.csv

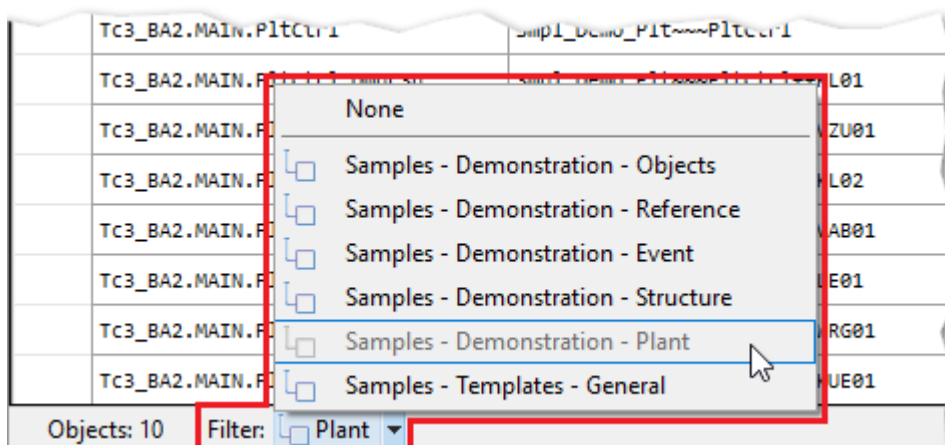
Object summary

Overview of all objects [► 28] located in the connected site.

Symbol	Object Name	Description
Tc3_BA2.MAIN.Smp1	Smp1	Samples
Tc3_BA2.MAIN.Demo	Smp1_Demo	Samples - Demonstration
Tc3_BA2.MAIN.Obj	Smp1_Demo_Obj	Samples - Demonstration - Objects
Tc3_BA2.MAIN.General	Smp1_Demo_Obj~~~General	Samples - Demonstration - Objects --- General objects
Tc3_BA2.MAIN.General.A	Smp1_Demo_Obj~~~General++A	Samples - Demonstration - Objects --- General objects - Analog objects
Tc3_BA2.MAIN.General.AV	Smp1_Demo_Obj~~~General++A++AV	Samples - Demonstration - Objects --- General objects - Analog objects - Analog...
Tc3_BA2.MAIN.General.AVDsp	Smp1_Demo_Obj~~~General++A++MyAvDsp001	Samples - Demonstration - Objects --- General objects - Analog objects - Sampl...
Tc3_BA2.MAIN.General.AVSp	Smp1_Demo_Obj~~~General++A++AVSp	Samples - Demonstration - Objects --- General objects - Analog objects - Sampl...
Tc3_BA2.MAIN.General.AVSp2	Smp1_Demo_Obj~~~General++A++AVSp2	Samples - Demonstration - Objects --- General objects - Analog objects - Sampl...

Filter

To increase the overview, the view can be filtered according to plants:



Trend overview

Overview of available trends [▶ 160] and recorded or referred objects [▶ 28].

Trend overview			
Export			
Trend	Referred Object		
Tc3_BA2 MAIN.Msc.TrendPol	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Sample Trend	Tc3_BA2 MAIN.Msc.PolAV	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Referred object for polling
Tc3_BA2 MAIN.Msc.TrendCovAI	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Sample Analog Trend COV	Tc3_BA2 MAIN.Msc.CovAI	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Referred analog object for COV
Tc3_BA2 MAIN.Msc.TrendCovBI	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Sample Binary Trend COV	Tc3_BA2 MAIN.Msc.CovBI	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Referred binary object for COV
Tc3_BA2 MAIN.Msc.TrendCovMI	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Sample Multistate Trend COV	Tc3_BA2 MAIN.Msc.CovMI	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Referred multistate object for COV
Tc3_BA2 MAIN.Msc.Trend[1]	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Multiple trend objects - Trend	Tc3_BA2 PRG_Obj_General_Ext.TrendViaPpf_Sp	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Test AV Setpoint
Tc3_BA2 MAIN.Msc.Trend[2]	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Multiple trend objects - Trend		
Tc3_BA2 MAIN.Msc.Trend[3]	Samples - Demonstration - Objects -- Miscellaneous objects - Tmd - Multiple trend objects - Trend		
Entries: 16 Unused trends: 10			

DPAD Relations

Create a comparison to show DPAD relations.

DPAD Relations								
Export								
Symbol	Object Name	Description	Level	Aggregate Name	Aggregate Description	Function Name	Function Description	
Tc3_BA2 PRG_Obj_General_Ext.Pf_AO_Cmd	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Pf++VLV01++CMD001	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Pf - Ventil - Command	5	VLV	Ventil	CMD	Command	
Tc3_BA2 PRG_Obj_General_Ext.Pf_InterpolPX1	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Pf++STX001	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Pf - Stützpunkt X[Idx]	5	STX	Stützpunkt X[Idx]			
Tc3_BA2 PRG_Obj_General_Ext.Pf_MV01	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Pf++CMD001	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Pf - Command	5	CMD	Command			
Tc3_BA2 PRG_Obj_General_Ext.Pf_MV02	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Pf++CMD002	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Pf - Command	5	CMD	Command			
Tc3_BA2 PRG_Obj_General_Ext.Pf_ProfileLessObj1	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Pf++PF_ProfileLessObj1	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Pf - Pf_ProfileLessObj1	5					
Tc3_BA2 PRG_Obj_General_Ext.Pf_ProfileLessObj2	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Pf++T2 UOA001	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Pf - Test 2 uöa	5					
Tc3_BA2 PRG_Obj_General_Ext.Pf_ProfileLessObj3	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Pf++T3001	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Pf - Test 3	5					
Tc3_BA2 PRG_Obj_General_Ext.Lbl	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Lbl	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Lbl	4					
Tc3_BA2 PRG_Obj_General_Ext.Lbl_Cmd	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Lbl++CMD001	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Lbl - Command	5	CMD	Command			
Tc3_BA2 PRG_Obj_General_Ext.Lbl_TR	Smpl_Demo_Obj~~~PRG_Obj_General_Ext.Lbl++TFL01++MW001	Samples - Demonstration - Objects -- PRG_Obj_General_Ext.Lbl - Vorlauftemperatur - Messwert	5	TFL	Vorlauftemperatur	MW	Messwert	
Entries: 149								

Included information per object [▶ 28]:

Symbol path

Properties

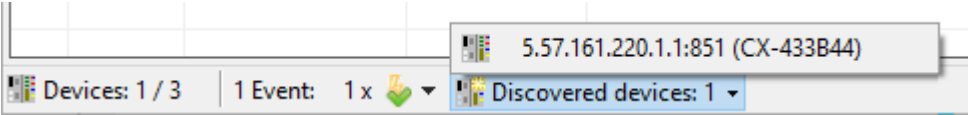
- ObjectName
- Description

Label (if used)

- **Aggregate information**
 - Name
 - Description
- **Function information**
 - Name
 - Description

Discovered devices

Each time a connection to the Site is established, connection information of all available devices is updated. If there are connections to unknown devices among them (which are not part of the current Site configuration), they are listed as discovered devices at the bottom of the application:



To add a discovered device to the Site configuration, just click on the corresponding menu item.



Discovered devices are possible suggestions. They are displayed because it can be assumed that they belong to the current Site configuration.

Log and status

Messages and states are displayed in the log or in the status bar at the bottom of the application:

	Code	Time	Process	Message
	OL52	14.06.2021 16:29:39		Started Terminal Explorer (Beta) v1.0.11.2
	LS408	14.06.2021 16:29:39	LoadSettings	Scope=Internal

Devices: 1 / 3 | 1 Event: 1 x  Discovered devices: 1

Log entries may contain additional information ().

These can be called up by moving the mouse pointer over the *.
The following functions can be called up via the context menu:

- **Copy selected entries:** Copies selected entries to the clipboard.
- **Copy all entries:** Copies all entries to the clipboard.
- **Export:** Copies all entries to a file.
- **Clean:** Removes all entries from the view.

Version

Details of versions are output in the log when the application is started:

	Code	Time	Process	Message
	OL52	14.06.2021 16:48:52		Started Terminal Explorer (Beta) v1.0.11.2
				Loaded TwinCAT ADS v4.4.0.0
				Loaded Terminal Client API v1.2.2.1 (Compatible to Terminal Server v1.0.12.0)

- **Terminal Explorer**

[OL52] Started TerminalExplorer v1.0.11.2

- **Terminal Client API**

Loaded terminal client API v1.2.2.1 (Compatible to terminal server v1.0.12.0)

The entry contains two pieces of information to be distinguished:

- Version of the loaded **Terminal Client API** DLL
- Version of the Terminal Server to which the loaded Terminal Client API is compatible.

NOTICE

Observe version

Only connections to ADS devices working with this version can be established!

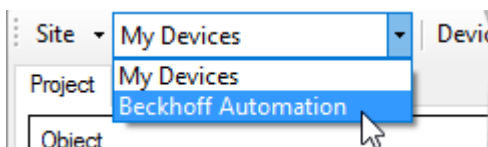
- **TwinCAT ADS**

Loaded TwinCAT ADS v4.4.0.0

Toolbar

Sites

Selection of configured Sites.



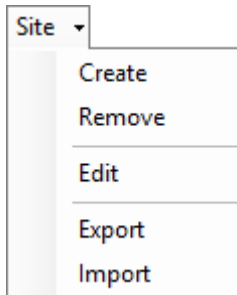
Use cases

You can choose between two use cases for Site deployment:

- **Local:** For own purposes (e.g. for tests) Site and device configurations can be stored locally (in the application directory).

- **Reference:** To improve collaboration in teams, references allow saving Site and device configurations in arbitrary directories (e.g. network drives or Git repositories).
Involved persons thus work on the same basis and avoid side effects such as:
 - Different selection and configuration of devices
 - Inconsistent commissioning states

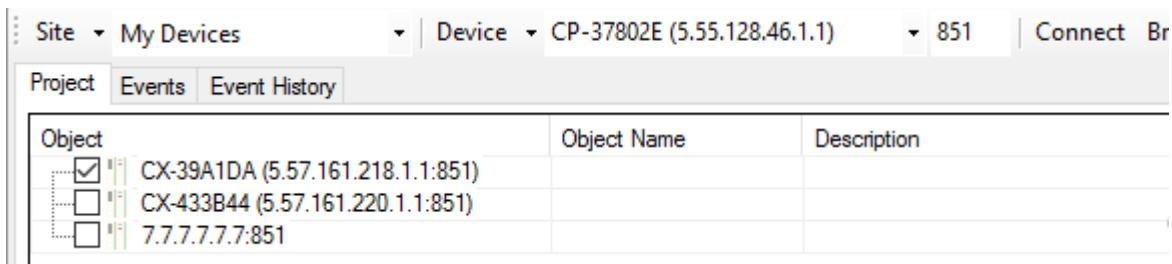
Site management



Name	Description
Create	Creates a new Site.
Remove	Removes an existing Site.
Edit	Displays the Site properties for editing.
Export	Exports a Site to a specific directory for referencing.
Import	Imports a Site for use as a reference.

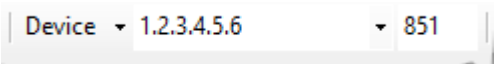
Connection

When the Site connects, communication is established to all selected devices:



Manage devices



Name	Description
Add	Adds a new device to the active Site. Adding devices by manually entering the <i>AMS NetID</i> is possible. All prerequisites (such as setting up the ADS route) must be met before a connection can be established. 
Remove	Removes a selected device from the active Site configuration.

Manage routes

- **Select route:** The selection box can be used to select routes that have already been created:

Device ▾ CX-2444B8 (5.36.68.184.1.1) ▾ 851

Object N CX-2444B8 (5.36.68.184.1.1) Man.

CX-25DF5E (5.37.223.94.1.1)

CX-39A1EA (5.57.161.234.1.1)

CX-3806B9 (5.56.6.185.1.1)

CX-39A1DA (5.57.161.218.1.1)

- **Create new route:** The dialog box for searching devices can be opened via the Browse button:

Connect Browse Diagn

Search target systems (Local) ...

Find target

Broadcast Search

Filter:

Name	Connected	Route	NetId	IPAddress	TwinCAT	OS
CX-39A1DA	X	X	5.57.161.218.1.1	172.21.97.12	3.1.4024	Win10 (1607)
MD-NB01	X	X	7.7.7.7.7.7	172.21.97.130	3.1.4024.7	Win8

Selected Route Target

Route Name:

NetId:

Protocol Type:

Address:

☐ HostName ☒ IP Address

Add Route

Remove Route

Close

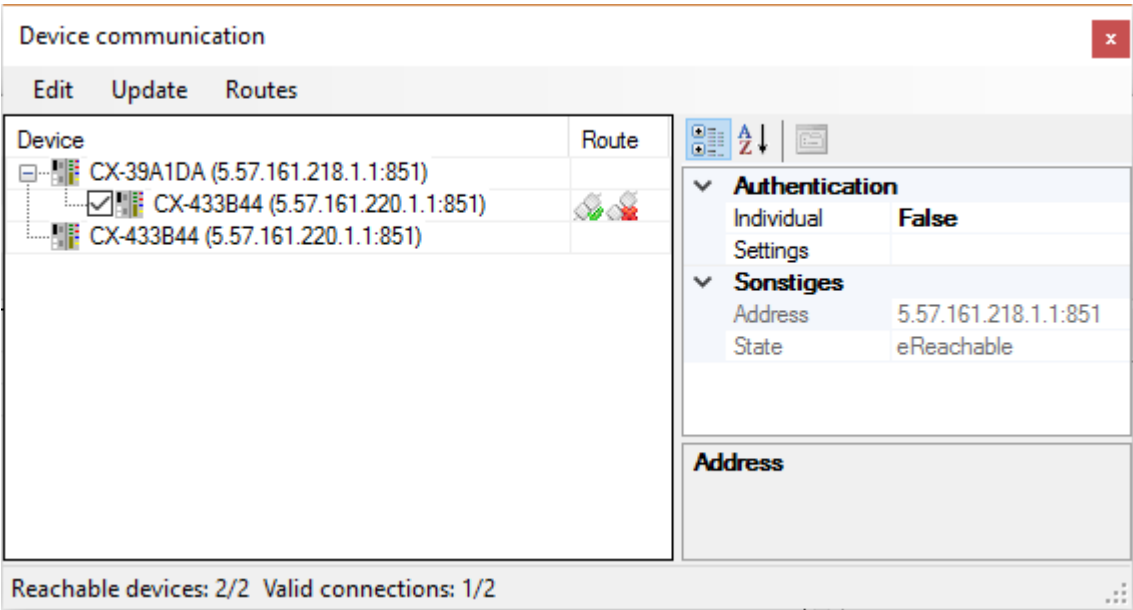
Diagnose

Diagnostic functions are accessible for configured devices of the active Site via the menu:

Diagnose ▾

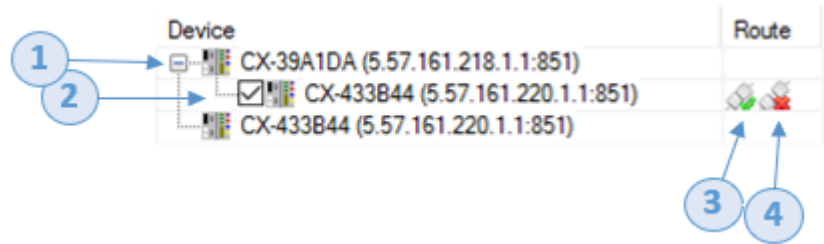
Device communication

Device communication



Connections and states

Displays a list of all configured devices to visualize the communication of individual devices with each other.



If a device (1) is designated for communication (using remote subscriptions), all **target devices** (2) will each appear as a connection under the **source device** (1).

i It also displays *target devices* (2) that are not part of the active Site configuration.

The individual states of the routes on the respective devices are indicated by corresponding symbols:

- The left symbol (3) represents the state of the route on the **source device** (1).
- The right symbol (4) represents the state of the route on the **target device** (2).

Key

- **Devices**

Graphic	Description
	Device unreachable.
	Device reachable.

- **Routes**

Graphic	Description
	Route unknown if the state could not be determined (e.g. if the device is unreachable).
	Route missing.
	Route valid.

Create route

Communication within a site is functional when the routes of all communicating devices are established with each other.

Example1: Routes valid on source and target device:

Device	Route
 CX-39A1DA (5.57.161.218.1.1:851)	
 CX-433B44 (5.57.161.220.1.1:851)	
 CX-433B44 (5.57.161.220.1.1:851)	

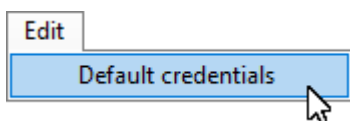
Example 2: Route valid on source device but not on target device:

Device	Route
 CX-39A1DA (5.57.161.218.1.1:851)	
 CX-433B44 (5.57.161.220.1.1:851)	
 CX-433B44 (5.57.161.220.1.1:851)	

Missing routes (both of all and for selected devices) can be configured at once:

✓ **Provide credentials:** In most cases, uniform credentials are used across all devices.

1. Open the properties for default credentials:



2. Make adjustments via properties:

▼	Login	
	Password	•
	UserName	Administrator
▼	Misc	
	Adressing	IPAddress
	Security	eNone

⇒ If individual credentials must be stored for different devices, the default credentials can be overwritten as follows:

3. Select device:

Device	Route
 CX-39A1DA (5.57.161.218.1.1:851)	
☒  CX-433B44 (5.57.161.220.1.1:851)	
 CX-433B44 (5.57.161.220.1.1:851)	

4. Activate and edit the individual credentials:

⇒ Select devices for route configuration

5. Devices with invalid route settings are automatically selected for configuration at the beginning:

6. If devices should not be provided for configuring the route, they can be deselected:

⇒ Apply configuration:

7. The route configuration of the selected devices can be rolled out via the menu:

⇒ All configuration operations are logged in the log:

	Code	Time	Process	Message
	AMR575	07.07.2021 13:57:40	AddRoutes	Adding route of connection 'CX-433B44 (5.57.161.220.1.1:851)' ~ 'CX-39A1DA (5.57.161.218.1.1:851)' successful.

⇒ For unavailable devices (e.g. not reachable or route not configured) neither route states can be displayed nor route configurations can be adopted:

Device	Route
CX-39A1DA (5.57.161.218.1.1:851)	
CX-433B44 (5.57.161.220.1.1:851)	
CX-433B44 (5.57.161.220.1.1:851)	

Fix connection

Despite existing route configuration, the connection between two devices may be faulty.

In this case, routes are set up on both sides, but with incorrect parameters (for example, if an IP address has changed).

Device	Route
CX-39A1DA (5.57.161.218.1.1:851)	
CX-433B44 (5.57.161.220.1.1:851)	
CX-433B44 (5.57.161.220.1.1:851)	

Example: Display of a valid connection due to routes set up on both sides.

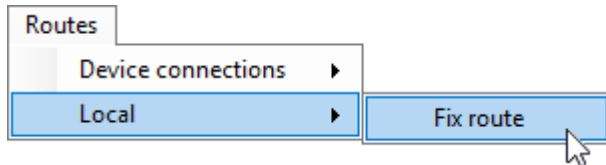
A (highlighted) connection can be repaired via the menu.
This reconfigures the routes on both devices.

Fix local route

Despite existing route configuration, the local connection to a target device may be faulty. In this case, the route is set up, but with faulty parameters (e.g., if the IP address of the target device has changed).

Device	Route
 CX-39A1DA (5.57.161.218.1.1:851)	
 CX-433B44 (5.57.161.220.1.1:851)	

Routes to unreachable devices can be fixed via the menu:



7.2 Symbol Explorer

7.2.1 Definitions

7.2.1.1 Symbol

A symbol is a type for describing variables of a Beckhoff controller. If variables are read from a controller using the Symbol Explorer, information such as name, type, size, subvariables and many other parameters are merged into one variable. A symbol is then formed from this quantity of parameters.

Symbol types

A symbol can describe different types of variables.

Complex symbols describe variables that can consist of function blocks, arrays and structures.

Primitive symbols describe variables that can consist of basic types, INT, REAL, BOOL, etc.

7.2.1.2 Snapshot

A snapshot describes a snapshot of a symbol or symbol structure.

The function to create a snapshot is provided for several use cases. You can create a snapshot of all the symbols and thus have a complete backup of the symbols on a controller. However, using the various representations and filter functions in the symbol list, you can also create just "sections of a symbol/symbols" and save them in a snapshot. You have thus created a template for the simplified duplication of symbol values.

7.2.2 Introduction

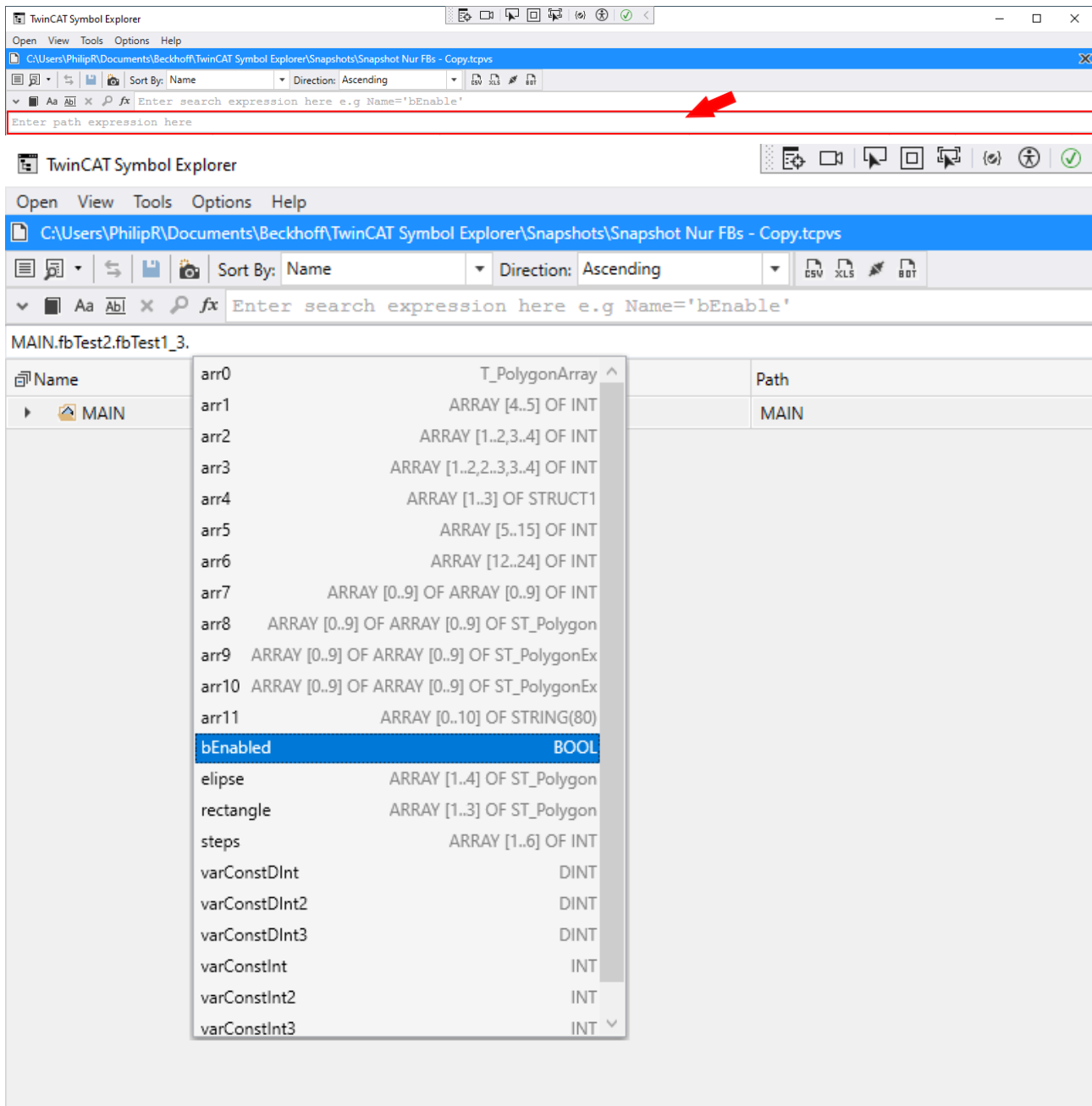
With the Symbol Explorer you can access controllers online via the ADS communication interface:

- You can read them as persistently declared variables.
- The Symbol Explorer provides functions with which variable backups can be created. These can be duplicated for follow-up projects and uploaded to other controllers.
- Variables can be checked for differences and merged using a compare function.

7.2.3 User interface

The interface is divided into various windows with different tasks for using the Symbol Explorer.

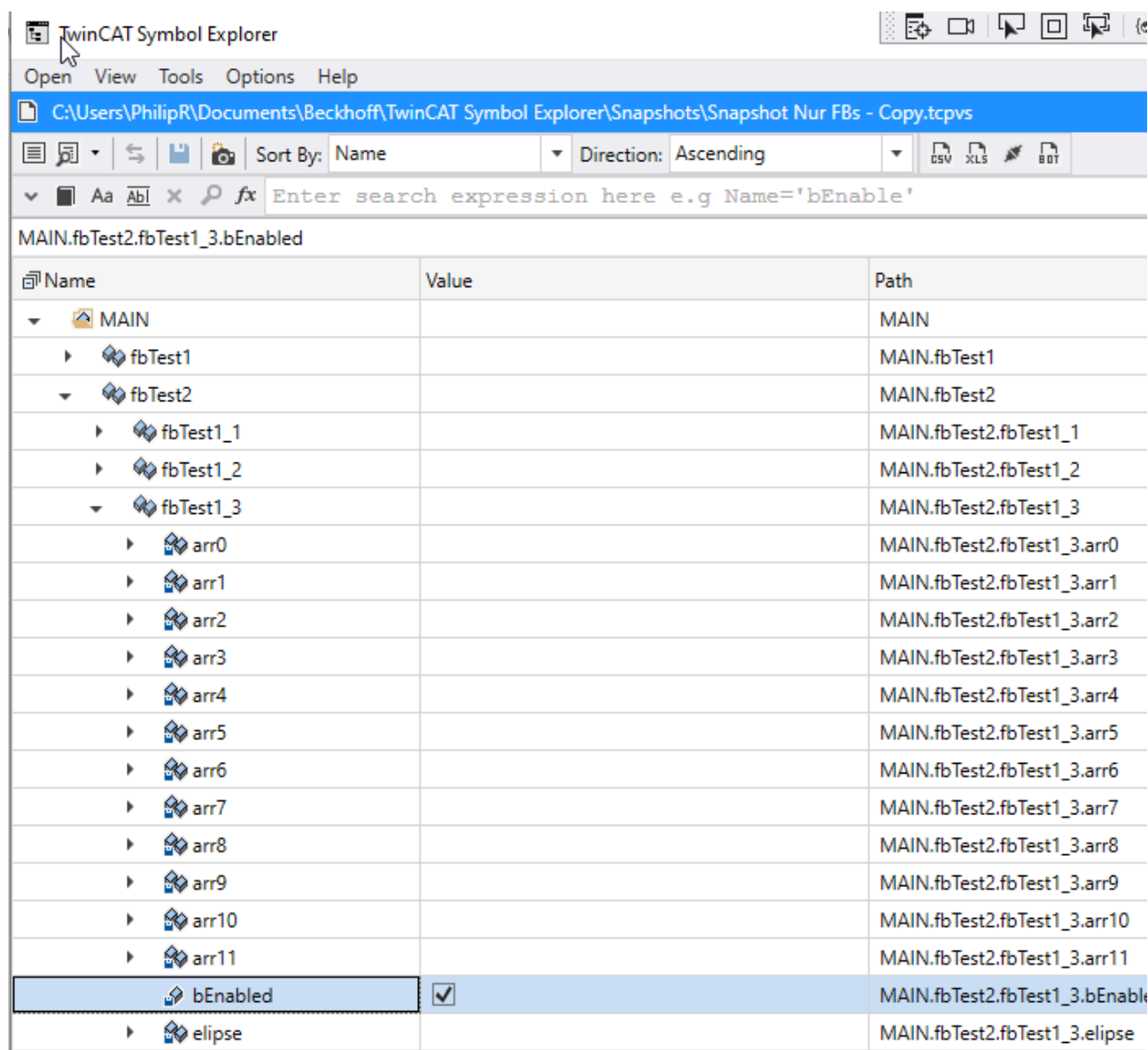
7.2.3.1 Symbol Quick Navigation Bar



The screenshot shows the TwinCAT Symbol Explorer interface. The top window has a search bar with the text "Enter search expression here e.g Name='bEnable'". A red arrow points to this search bar. Below it, the same window is shown with the search results for "bEnabled".

Search Results:

Name	Path
MAIN	MAIN
arr0	T_PolygonArray
arr1	ARRAY [4..5] OF INT
arr2	ARRAY [1..2,3..4] OF INT
arr3	ARRAY [1..2,2..3,3..4] OF INT
arr4	ARRAY [1..3] OF STRUCT1
arr5	ARRAY [5..15] OF INT
arr6	ARRAY [12..24] OF INT
arr7	ARRAY [0..9] OF ARRAY [0..9] OF INT
arr8	ARRAY [0..9] OF ARRAY [0..9] OF ST_Polygon
arr9	ARRAY [0..9] OF ARRAY [0..9] OF ST_PolygonEx
arr10	ARRAY [0..9] OF ARRAY [0..9] OF ST_PolygonEx
arr11	ARRAY [0..10] OF STRING(80)
bEnabled	BOOL
ellipse	ARRAY [1..4] OF ST_Polygon
rectangle	ARRAY [1..3] OF ST_Polygon
steps	ARRAY [1..6] OF INT
varConstDInt	DINT
varConstDInt2	DINT
varConstDInt3	DINT
varConstInt	INT
varConstInt2	INT
varConstInt3	INT



7.2.3.2 Main window

After opening the start page, the following options can be selected:

- Recent snapshots
- Recent routes
- Connect
- Snapshot

Main menu

The menu is divided into the following tabs.

Open

Command	Description
Choose route	Connect to a route.
Open snapshot	Open a snapshot.
Recent snapshots	List of frequently used snapshots.
Recent routes	List of frequently used routes.

View

Command	Description
Output	Switch Output window small / large.

Tools

Command	Description
Compare symbols	Comparison of two symbol lists.

Help

Command	Description
About TwinCAT Symbol Explorer	Version specification

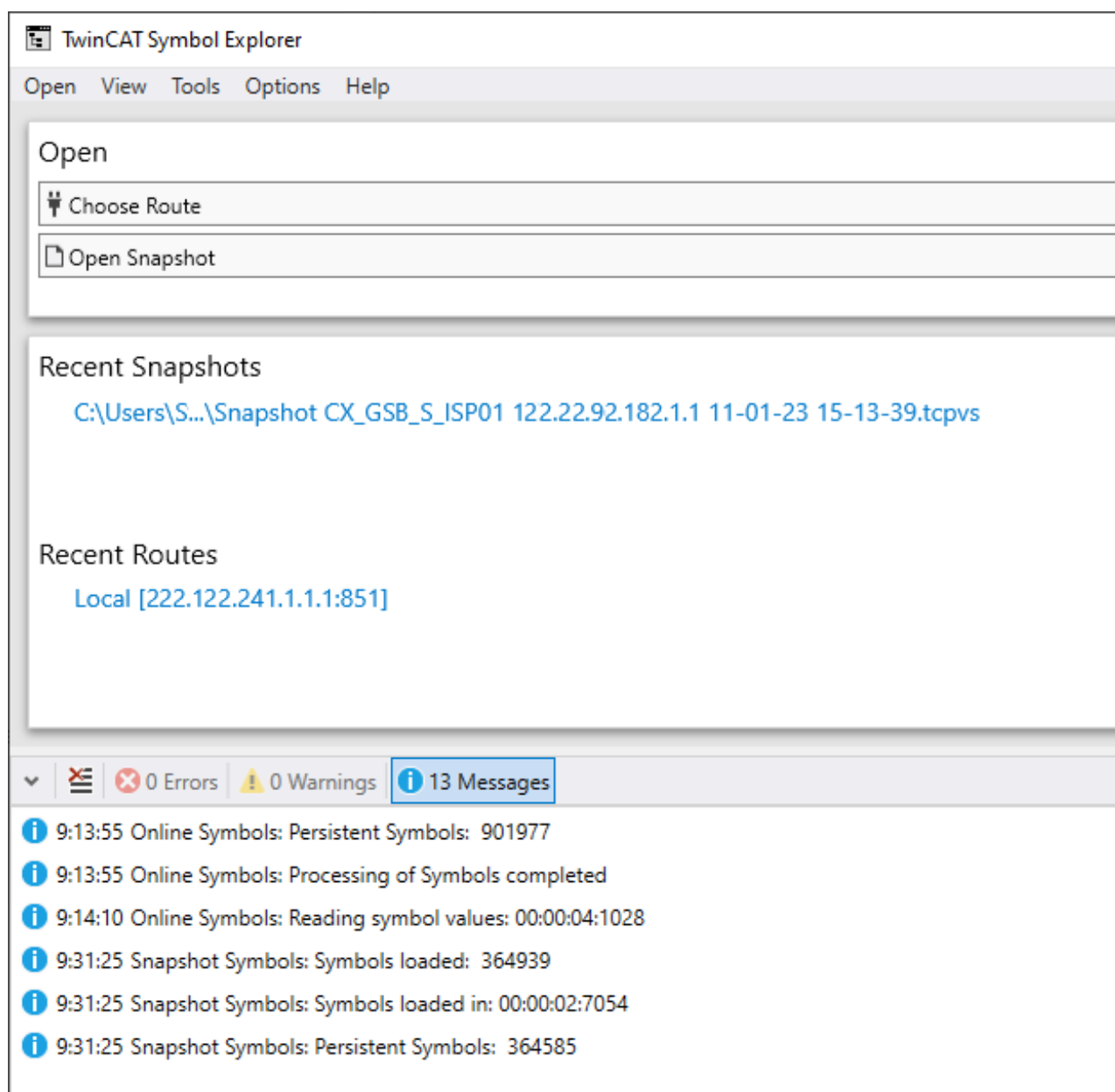
7.2.3.2.1 Start Page button

Symbol	Command	Description
	Connect route	Connect to the controller
	Open snapshot	Open a snapshot
	Recent snapshots	List of frequently used snapshots
	Recent routes	List of frequently used routes

7.2.3.3 Start page

The start page gives a quick overview of the last selected routes or opened snapshots.

Furthermore, one of the last routes or snapshots can be selected and opened directly with one click.



Open

Command	Description
Choose Route	Connect to a route.
Open snapshot	Open a snapshot

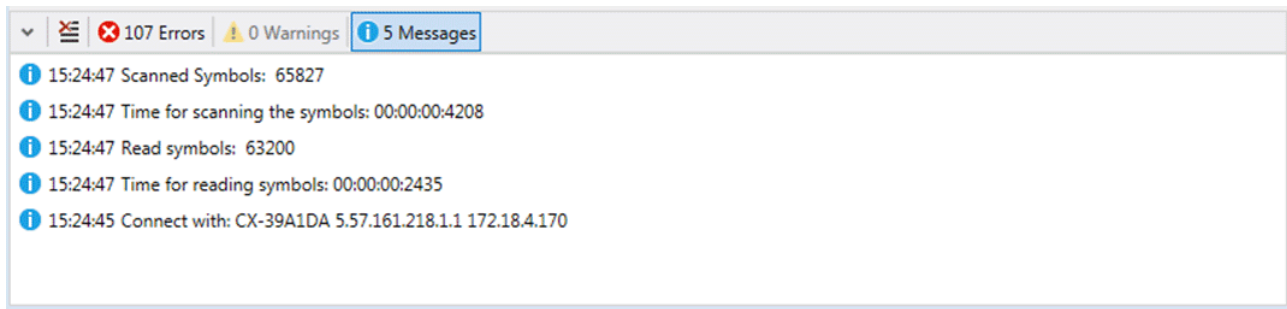
Recent

Command	Description
Recent snapshots	List of frequently used snapshots.
Recent routes	List of frequently used routes.

7.2.3.4 Output window

The Output window displays information, errors and warnings at runtime.

The functions of the toolbar are described in the following table.



Toolbar

Symbol	Command	Description
	Expand / Collapse	Expands and collapses the Output window
	Clear messages	Clears the messages
	Show / Hide errors	Shows and hides the error messages
	Show / Hide warnings	Shows and hides the warnings
	Show / Hide messages	Shows and hides the messages

7.2.3.5 Online window

The Online window displays the online symbols of a controller. Functions for creating backups, monitoring and editing symbols are available via the toolbar and the context menus.

Toolbar

Symbol	Command	Description
	Show symbols as list	Shows symbols as a flat list.
	Drop down box	Extended symbol lists.
 Show Instance Watch	Show instance watch	Symbol instances view.
	Synchronize symbols	Show dialog for synchronizing symbols.
Sort By: Name	Sort by	Sorting of symbols by e.g. name or size.
Direction: Ascending	Sort direction	Direction of sorting.
	Upload symbols	Writes changed symbols to the controller
	Take snapshot	Takes a snapshot of the symbols
	Refresh	Refreshes symbol value once
	Auto refresh	Refreshes symbol value every 2 seconds
 CSV	Export / Import CSV	Export or import symbols as CSV file
 XLS	Export symbols as Excel	Export symbols as an Excel spreadsheet.
	Close window	Close window

● Exclusions when refreshing

i Symbols that have been edited via the Symbol Explorer and symbols that are currently being edited are excluded from refreshing.

● Notes on ADS communication

i The Symbol Explorer uses the ADS protocol for the online communication. Note that ADS is only a transport layer; however, side effects can occur. Read these requirements and note the restrictions: ADS itself is only the transport layer, the requested ADS device must support the ADS command. When the PLC is processing an ADS request (reading/writing the symbol values), it will work completely on that single ADS request before starting a new PLC cycle. To keep the load on the controller low, the number of symbols to be read/written has been limited to a maximum of 250 per ADS request for this reason.

7.2.3.6 Snapshot window

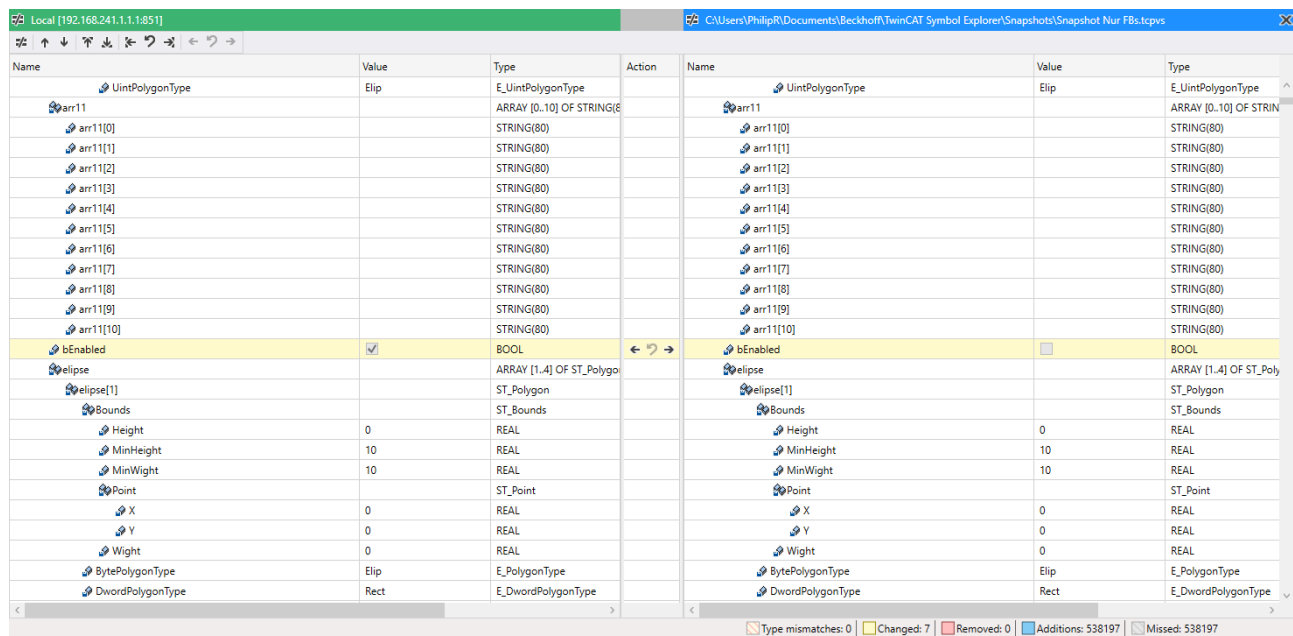
The Snapshot window displays the offline symbols from a snapshot. Functions for editing, uploading and creating copy templates are available via the toolbar and the context menus.

Toolbar

Symbol	Command	Description
	Show symbols as list	Shows symbols as a flat list.
	Drop down box	Extended symbol lists.
 Show Instance Watch	Show instance watch	Symbol instances view.
	Synchronize symbols	Show dialog for synchronizing symbols.
Sort By: Name	Sort by	Sorting of symbols by e.g. name or size.
Direction: Ascending	Sort direction	Direction of sorting.
	Save	Saves a snapshot.
	Take snapshot	Takes a snapshot of the symbols
	Export / Import CSV	Export or import symbols as CSV file
	Export symbols as Excel	Export symbols as an Excel spreadsheet.
	Close window	Close window

7.2.3.7 Comparison window

The Symbol Compare window displays the offline symbols from a snapshot. Functions for editing, uploading and creating copy templates of the symbols are available via the toolbar and the context menus.



Name	Value	Type	Action	Name	Value	Type
UintPolygonType	Elip	E_UintPolygonType		UintPolygonType	Elip	E_UintPolygonType
arr1		ARRAY [0..10] OF STRING(80)		arr1		ARRAY [0..10] OF STRING(80)
arr1[0]		STRING(80)		arr1[0]		STRING(80)
arr1[1]		STRING(80)		arr1[1]		STRING(80)
arr1[2]		STRING(80)		arr1[2]		STRING(80)
arr1[3]		STRING(80)		arr1[3]		STRING(80)
arr1[4]		STRING(80)		arr1[4]		STRING(80)
arr1[5]		STRING(80)		arr1[5]		STRING(80)
arr1[6]		STRING(80)		arr1[6]		STRING(80)
arr1[7]		STRING(80)		arr1[7]		STRING(80)
arr1[8]		STRING(80)		arr1[8]		STRING(80)
arr1[9]		STRING(80)		arr1[9]		STRING(80)
arr1[10]		STRING(80)		arr1[10]		STRING(80)
bEnabled	☒	BOOL	← →	bEnabled	☐	BOOL
ellipse		ARRAY [1..4] OF ST_Polygon		ellipse		ARRAY [1..4] OF ST_Polygon
ellipse[1]		ST_Polygon		ellipse[1]		ST_Polygon
Bounds		ST_Bounds		Bounds		ST_Bounds
Height	0	REAL		Height	0	REAL
MinHeight	10	REAL		MinHeight	10	REAL
MinWight	10	REAL		MinWight	10	REAL
Point		ST_Point		Point		ST_Point
X	0	REAL		X	0	REAL
Y	0	REAL		Y	0	REAL
Wight	0	REAL		Wight	0	REAL
BytePolygonType	Elip	E_PolygonType		BytePolygonType	Elip	E_PolygonType
DwordPolygonType	Rect	E_DwordPolygonType		DwordPolygonType	Rect	E_DwordPolygonType

Type mismatches: 0 | Changed: 7 | Removed: 0 | Additions: 538197 | Missed: 538197

The Symbol Compare window displays the following differences:

- Differences from symbol to symbol (value)
- Removed symbols
- Added symbols
- Missing symbols

- Types differences

Toolbar

Symbol	Command	Description
	Show changes	Shows value differences only.
	Previous change	Go to previous change
	Next change	Go to next change
	First change	Go to first change
	Undo all	Reset changes
	Last change	Go to last change
	Copy all changes to left	Copies all changes into the left-hand list
	Copy all changes to right	Copies all changes into the right-hand list
	Copy change to left	Copies the selected row into the left-hand list
	Copy change to right	Copies the selected row into the right-hand list
	Close window	Close window

7.2.3.8 Symbol list



Copying and pasting in the Symbol Explorer is list cell dependent!

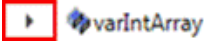
This means that if a cell is selected in the Symbol Explorer and copied, the contents of the cell are copied.

The only exception is the cell 'Name', if this is selected and copied, the entire symbol information is copied.

By means of the hotkeys and context menus, the symbol list provides functions such as Edit values or Copy/Paste. Furthermore, the symbol list offers a filter function, with which symbols to be edited can be purposefully filtered and highlighted.

Name	Value	Type	Comment
▶ BACnet_Globals_LIB			
▼ MAIN			
varInt	12	INT	INT Variable
varInt2	13	INT	INT Variable
varInt3	0	INT	INT Variable
varDInt	0	DINT	DINT Variable
varDInt2	450	DINT	DINT Variable
varDInt3	0	DINT	DINT Variable
▶ fbArray		ARRAY [0..10] OF FB_Test_0	
▶ fbTest0		FB_Test_0	

Hotkeys

Shortcut	Description
Ctrl + left-click 	Expands or collapses the symbol and its sub-symbols.
Ctrl + left-click	Selective multi-selection on a row.
Ctrl + C	Copies the selected symbol (and all sub-symbols).
Ctrl + V	Pastes the copied symbol values into the selected symbol (and the sub-symbols).
Ctrl + Shift + C	Copies the contents of the selected cell.
Ctrl + R	Resets the symbol value to the previous value.
Shift + left-click	Multi-selection
Double left-click	Starts the editing of a symbol.
F2	If a symbol had been selected beforehand, editing starts.

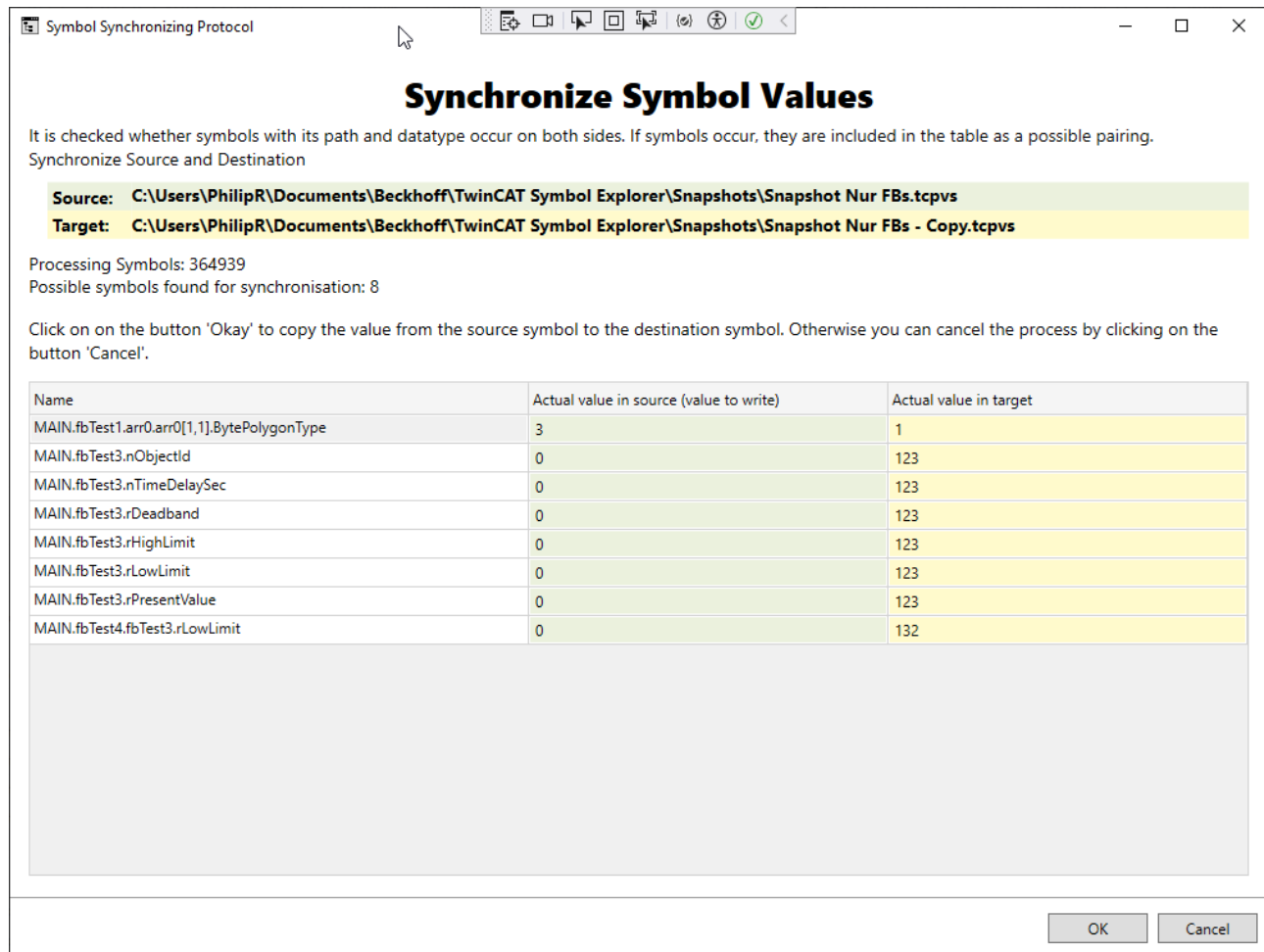
Context menu

Selection	Description
Read from PLC	Reads the current value from the PLC on the selected symbol.
Copy	Copies the selected symbol (and all subsystems).
Paste	Pastes the copied symbol values into the selected symbol (and the sub-symbols).
Search for	Suggestions for a search
Reset	Resets the symbol value to the previous value.
Add instance watch	Add symbol to the Instance Watch.
Details	Call symbol details



Copying and pasting behaves differently in the Symbol Explorer than in other software tools. If a symbol is copied in the Symbol Explorer and pasted to a different symbol, the symbol is not copied like a file, e.g. in Windows Explorer; instead, only the symbol values from the copying source are transferred to the target symbol.

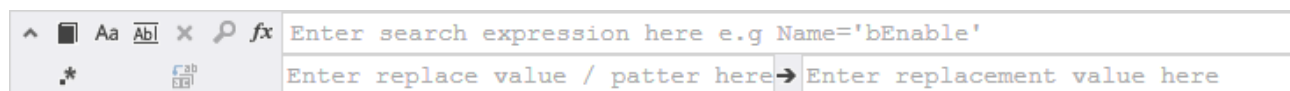
7.2.3.9 Synchronize Symbol window



7.2.4 Find and replace

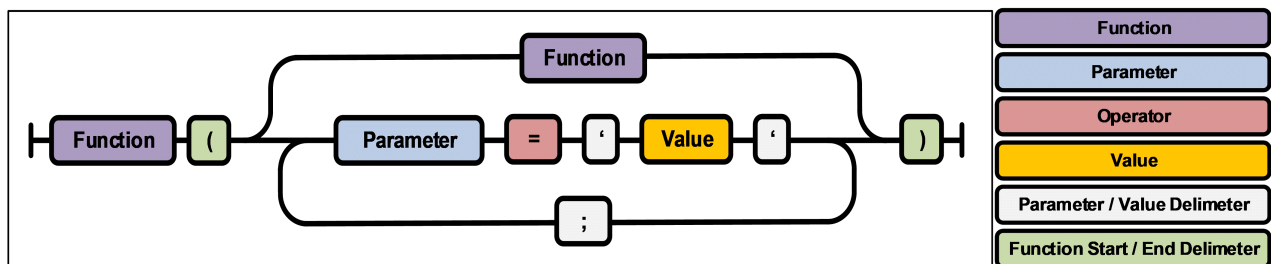
By means of the individual filtering of the symbols, it is possible to find the desired symbols faster. You can filter for the name, type or comment of a symbol.

Find and replace toolbar



Symbol	Command	Description
	Recent search terms	List of frequent searches
	Match case	Search is case-sensitive
	Match whole word	Search for the exact entry
	Close search	Closes the search
	Start search	Starts the search
	Use regular expressions	Switches the search to the use of regular expressions
	Replace all	Replace all
	Search Pattern	Search Pattern Templates

7.2.4.1 Filter equation overview



A function is a subordinate quantity of functions, parameters, operators and value pairs. A function begins and ends with brackets (). Each function, parameter, operator and value pair is followed by a semicolon.

AND function

The AND function is used to filter for symbols whose conditions are satisfied by a TRUE. Two examples of the use of the AND function and in conjunction with the OR function are shown below.

```
AND ( Type = BOOL;
      Name = 'bEnabled';
      Value = True )
```

The function filters for symbols with the type `BOOL` whose name is `bEnabled` and whose value is `TRUE`.

```
AND ( Name = nCounter;
      OR ( Value = 10;
           Value = 50 ) )
```

The function filters for symbols with the name `nCounter` whose value is '10' or '50'.

OR function

The OR function is used to filter for symbols where one condition is satisfied by a TRUE. Two examples of the use of the OR function and in conjunction with the AND function are shown below.

```
OR ( Value = 150;
     Value = -150 )
```

The function filters for symbols whose value is '150' or '-150'.

```
AND ( Name = 'nCounter';
      OR ( Value = 10;
           Value = 50 ) )
```

The function filters for symbols with the name *nCounter* whose value is '10' or '50'.

PARENT function

The PARENT function is used to filter for symbols whose direct parent symbol (higher-level symbol) is subject to an identical condition. The conditions specified in the PARENT function are logically ANDed.



The PARENT function can only be used in an AND function or in an OR function.

Here is a general sample of the use of the PARENT function:

```
AND (      Name = 'sObjectName';  
PARENT (  Name = 'Plant';  
          Value = TRUE ))
```

The function filters for symbols with the name '*sObjectName*' and their parent symbol with the name '*Plant*'. Furthermore, the value of this symbol must be TRUE.

ANCESTOR function

The ANCESTOR function is used to filter symbols whose ancestors in the parent chain of symbols are subject to a condition, e.g. a special name or type. The condition specified in the ANCESTOR function is logically ANDed.



The ANCESTOR function can only be used in an AND function or in an OR function.

Here are some general examples of the use of the ANCESTOR function:

```
AND (      Name = 'sObjectName';  
ANCESTOR ( Type = 'FB_BAC_AI'))
```

The function filters for symbols with the name '*sObjectName*'. The ancestors of these symbols come from the parent chain of symbols with the type '*FB_BAC_AI*'.

```
AND (      Name = 'sObjectName';  
ANCESTOR ( Name = 'Plant01'))
```

The function filters for symbols with the name '*sObjectName*'. Within the parent chain of these symbols, one of the ancestors bears the name '*Plant01*'.

CHILD function

The CHILD function is used to filter symbols whose child symbols are subject to a condition. The conditions specified in the CHILD function are logically ANDed.



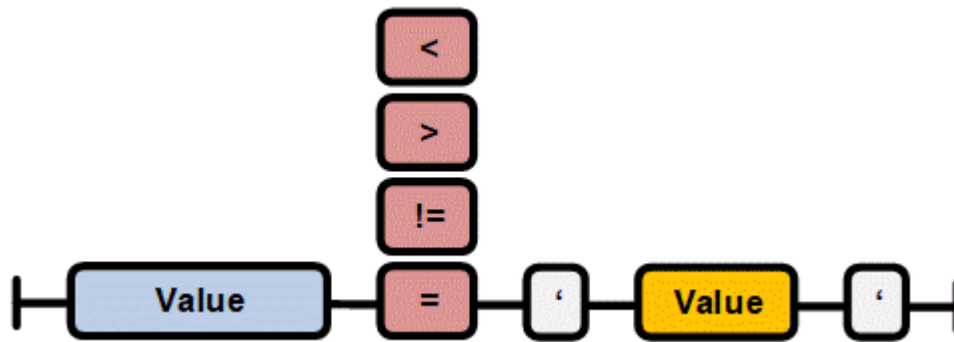
The CHILD function can only be used in an AND function or in an OR function.

Here is a general example of the use of the CHILD function:

```
AND (      Name = 'Plant';  
Child (  Name = 'sObjectName';  
          Value = 'P001' ))
```

Value Parameter

The function searches for symbols whose value satisfies a condition with a certain value.

**Examples:**

```
Value = '100'
```

Search for symbols whose value is '100'.

```
Value > '100'
```

Search for symbols whose value is greater than '100'.

```
Value < '100'
```

Search for symbols whose value is smaller than '100'.

```
Value != '100'
```

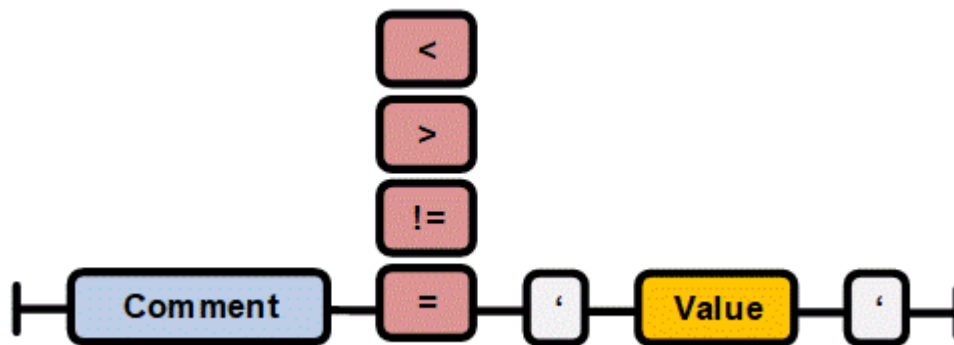
Search for symbols whose value is not '100'.

```
Value = ''
```

Search for symbols whose value is empty ''.

Comment Parameter

The function searches for symbols whose value satisfies a condition with a certain comment.

**Examples:**

```
Comment = 'Signal to detect'
```

Search for symbols whose comment is 'Signal to detect'.

```
Comment != 'Signal to detect'
```

Search for symbols whose comment is not 'Signal to detect'.

```
Comment > 'Signal'
```

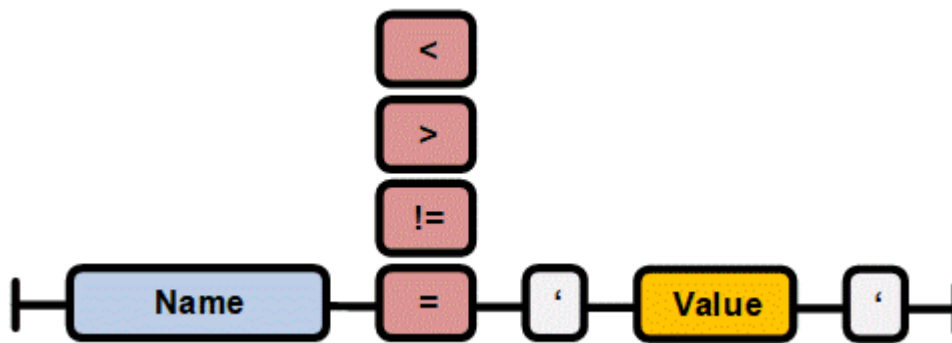
Search for symbols whose comment begins with 'Signal'.

```
Comment < 'detect'
```

Search for symbols whose comment ends with 'detect'.

Name Parameter

The function searches for symbols whose value satisfies a condition with a certain name.



Examples:

```
Name = 'bEnable'
```

Search for symbols whose name is 'bEnable'.

```
Name != 'bEnable'
```

Search for symbols whose name is not 'bEnable'.

```
Name > 'bEn'
```

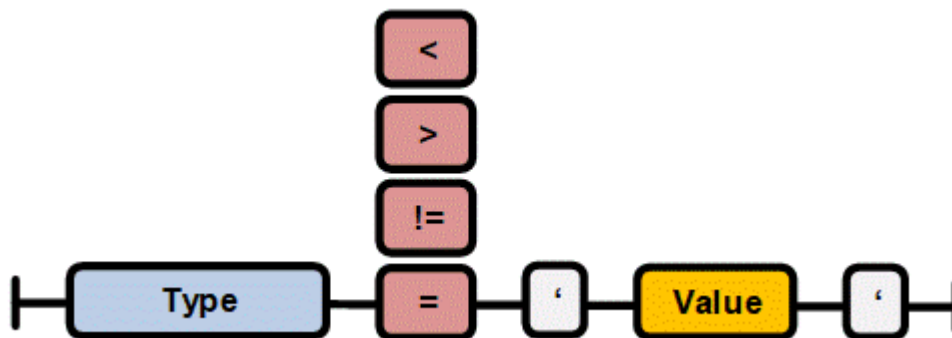
Search for symbols whose name begins with 'bEn'.

```
Name < 'le'
```

Search for symbols whose name ends with 'le'.

Type Parameter

The function searches for symbols whose value corresponds to a condition of a certain type.



Examples:

```
Type = 'FB_BACnet_Pump'
```

Search for symbols whose type is 'FB_BACnet_Pump'.

```
Type != 'FB_BACnet_Pump'
```

Search for symbols whose name is not 'FB_BACnet_Pump'.

```
Type > 'FB_BACnet'
```

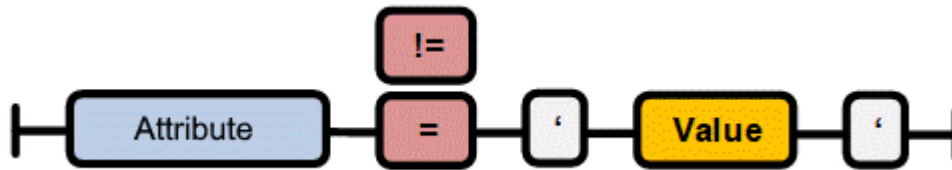
Search for symbols whose name begins with 'FB_BACnet'.

```
Type < 'BACnet_Pump'
```

Search for symbols whose name ends with 'BACnet_Pump'.

Attribute Parameter

The function searches for symbols that have an attribute whose name satisfies a condition with a certain value.



Samples:

```
Attribute = 'IsPersistent'
```

Search for symbols that have an attribute with the name 'IsPersistent'.

```
Attribute != 'IsPersistent'
```

Search for symbols that do not have an attribute with the name 'IsPersistent'.

7.2.5 Getting started

This chapter describes step by step how to work with the Symbol Explorer and is intended to provide an overview of its functions.

Connecting to a controller

The Symbol Explorer communicates with a controller via the ADS communication interface. So that communication with a controller can be successfully established, the following conditions must be satisfied:

- The controller can be reached via the network.
- TwinCAT is in Run Mode.
- An AMS route to the controller has been set up.
- A PLC runtime has been activated and started.

If the criteria are fulfilled, a connection can be established and the symbols read out. It is only possible to read out symbols that have been declared as persistent and symbols that contain persistently declared symbols.

Starting the connection to a controller

1. Start the Symbol Explorer and click the **Connect** button on the start page.
 2. In the **Choose Route** dialog, select the **Ams Route** belonging to the controller to which you wish to connect.
- ⇒ If you were able to successfully connect to the controller, the online window opens, showing you the read-out symbols in a list.

Taking a snapshot

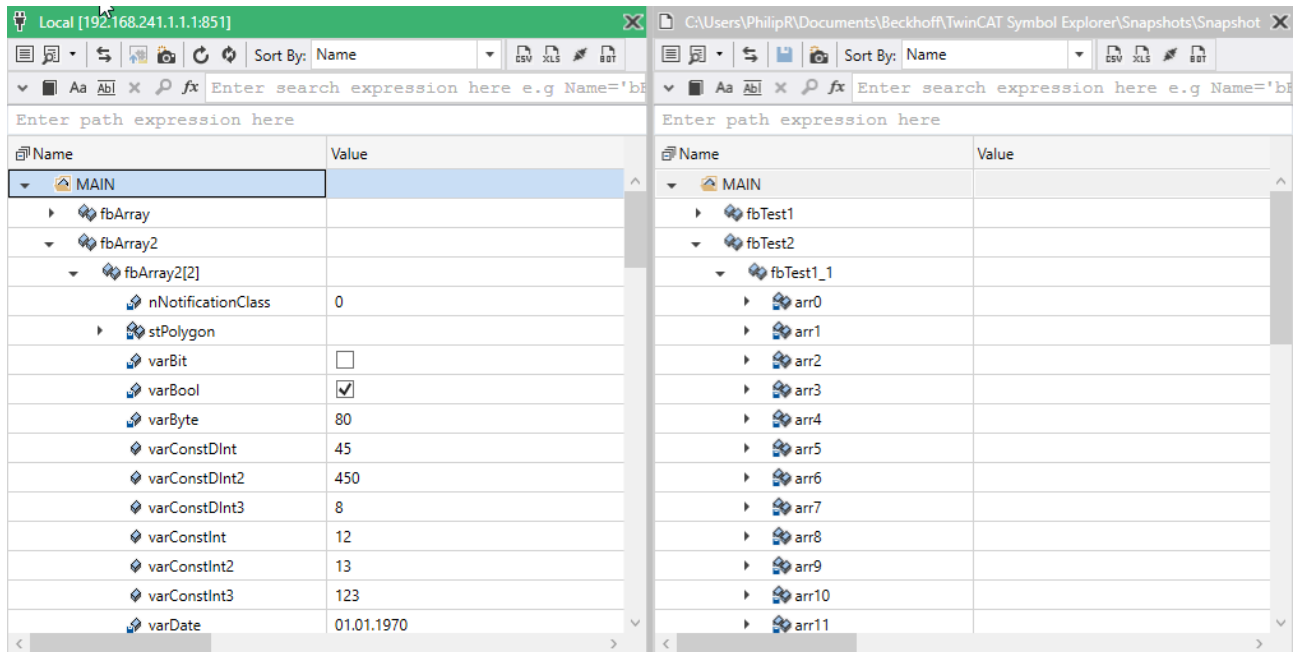
1. Start the **Symbol Explorer** and connect to a controller.
 2. Click the **Take Snapshot** button in the online window in order to take the snapshot.
 3. You will then be requested to select a location to save the snapshot file. Select a folder and confirm with **OK**.
- ⇒ All values of the symbols in the list are synchronized with the PLC and written to the file.

Loading a snapshot

1. Open the **Symbol Explorer** and click the **Snapshot** button on the start page.
 2. You will then be requested to select a snapshot file. Select a snapshot file to be loaded and confirm with **OK**.
- ⇒ The snapshot window then opens, showing you the loaded symbols in a list.

Viewing and editing symbol lists in parallel

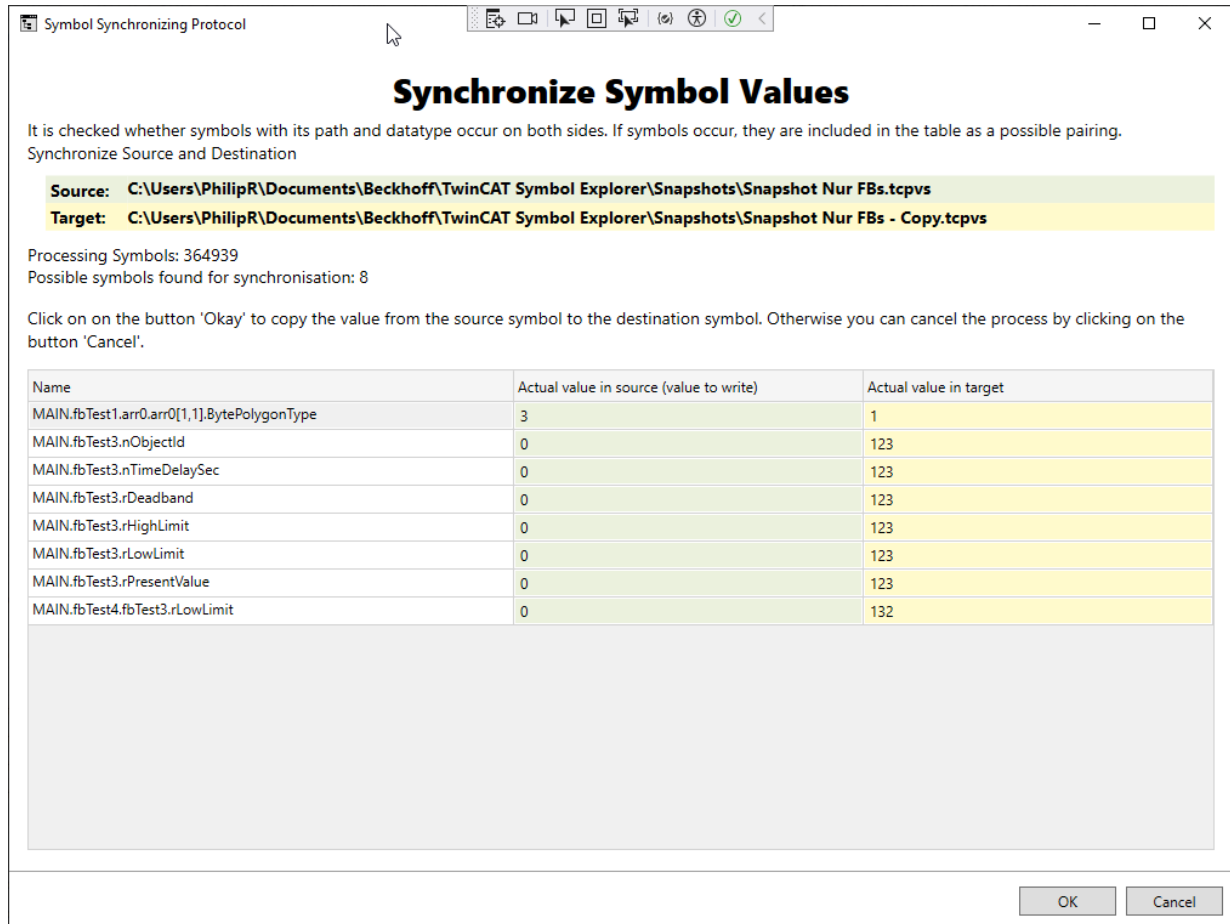
The Symbol Explorer can display two symbol lists in parallel. It is thus possible, for example, to place the current symbols (online) and symbols from a snapshot next to one another and to edit symbol values between these two lists. Simply connect to a controller and then open a snapshot. A different order of opening/connecting is also possible.



Copying a snapshot to a controller

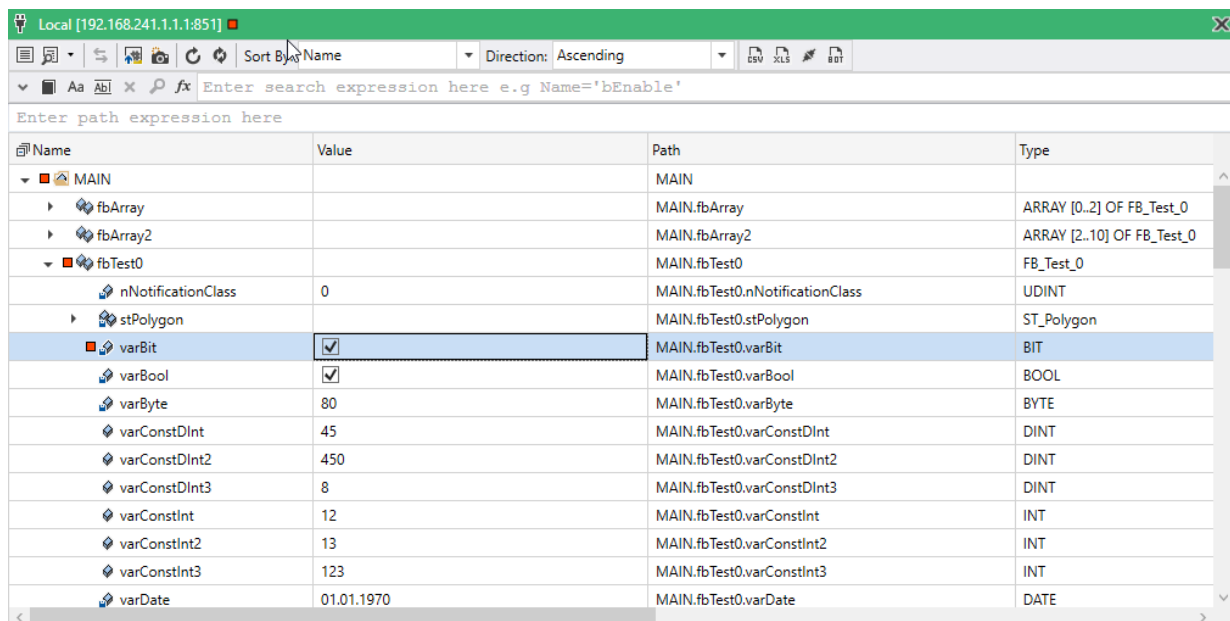
1. Open and connect the Symbol Explorer to a controller.
2. Then open a snapshot. You can now see the Online and Snapshot windows side by side.

3. Press the **Copy Symbols** button in the snapshot window. The **Symbol Transfer Protocol** dialog then opens:



⇒ The dialog informs you in detail about the copying procedure.

4. If the symbol values are correct, confirm the copying procedure by clicking **OK**, otherwise select **Cancel**. If you have confirmed the copying procedure, the symbol values from the snapshot are copied into the online symbol list.



⇒ Once the copy procedure is completed, the symbols with changed values are highlighted by a red square in front of the symbol name.

5. You can now load the changes from the Online window into the controller. To do this, press the **Upload Symbol** button in the toolbar.

Comparing symbols and synchronizing differences

With the Symbol Explorer it is possible to compare two lists of symbols. You can thus compare online symbols with symbols from a snapshot and check for differences.

Starting a comparison

1. Connect to a controller or load a snapshot.
 2. Repeat the procedure so that two symbol lists stand side by side.
 3. Then press the hotkey **[F8]** or select **Compare Symbols** in the Tools menu.
- ⇒ The comparison window then opens.

Synchronizing differences

During synchronization, differences are copied from one symbol to another, either from left to right or from right to left. This procedure differs from the direct editing of a symbol via the Online or Snapshot window.

Synchronization functions

The synchronization functions can be applied implicitly to differences on the basis of a row selection. For example, if you click a row with a difference and then the right arrow button on the central toolbar, the symbol value is copied from the left into the symbol on the right. This allows the easy merging of many small differences.

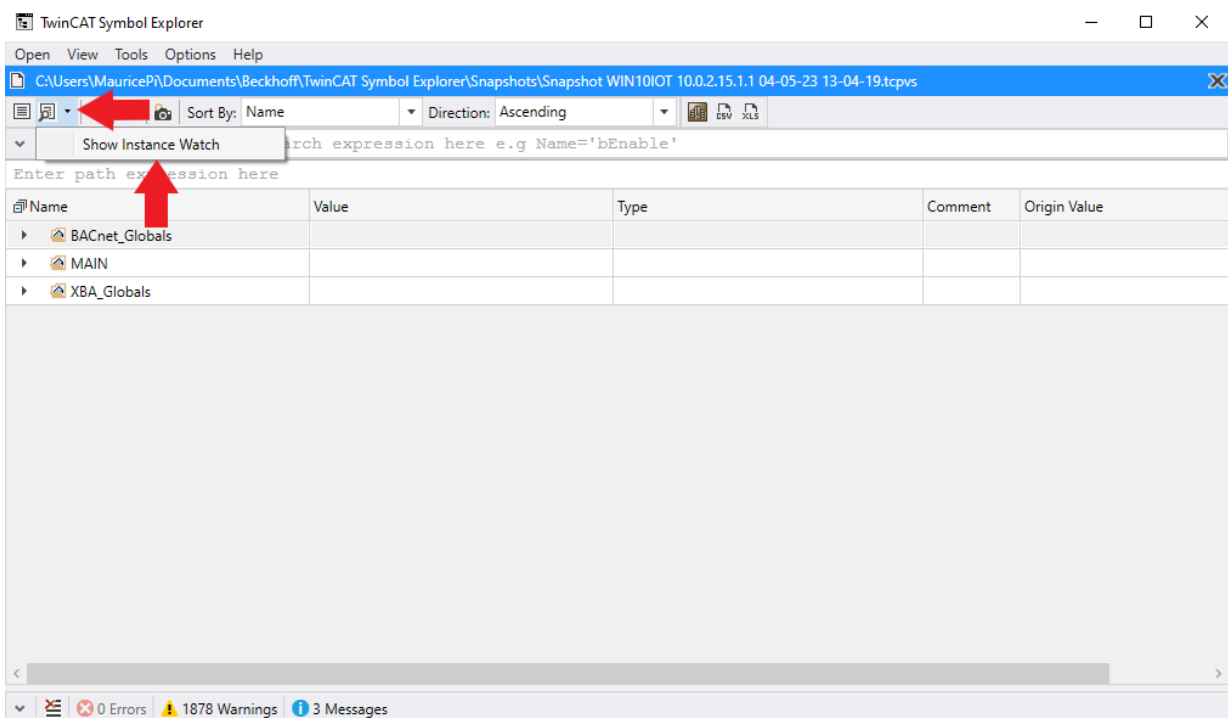
Local [192.168.241.1.1.1:851]			C:\Users\PhilipR\Documents\Beckhoff\TwinCAT Symbol Explorer\Snapshots\SnapX		
Name	Value	Action	Name	Value	
arr11[1]			arr11[1]		
arr11[2]			arr11[2]		
arr11[3]			arr11[3]		
arr11[4]			arr11[4]		
arr11[5]			arr11[5]		
arr11[6]			arr11[6]		
arr11[7]			arr11[7]		
arr11[8]			arr11[8]		
arr11[9]			arr11[9]		
arr11[10]			arr11[10]		
bEnabled	☒	← →	bEnabled	☐	
ellipse			ellipse		
ellipse[1]			ellipse[1]		
Bounds			Bounds		
Height	0		Height	0	
MinHeight	10		MinHeight	10	
MinWight	10		MinWight	10	

Type mismatches: 0 |
 Changed: 7 |
 Removed: 0 |
 Additions: 538197 |
 Missed: 538197

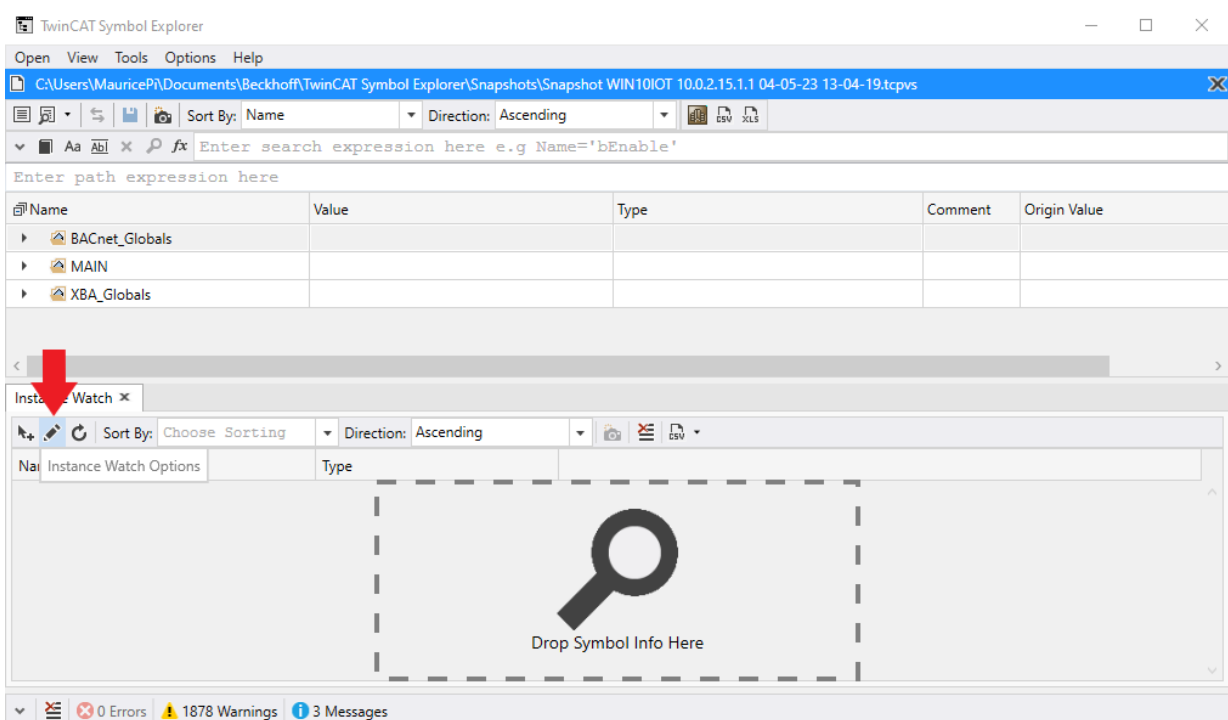
7.2.5.1 Open Instance Watch Template

- ✓ To load an Instance Watch template, the Symbol Explorer must first be started.

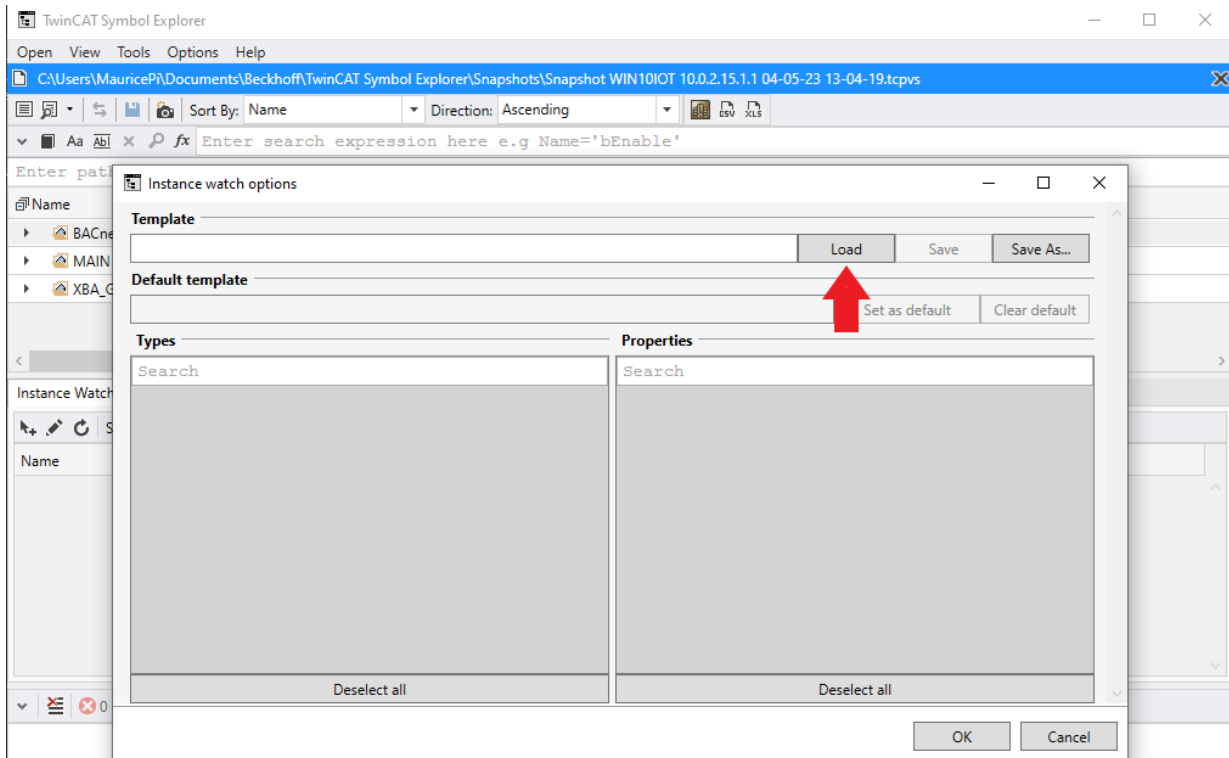
1. Open the Instance Watch



⇒ If no default template has been defined, the Instance Watch is empty.

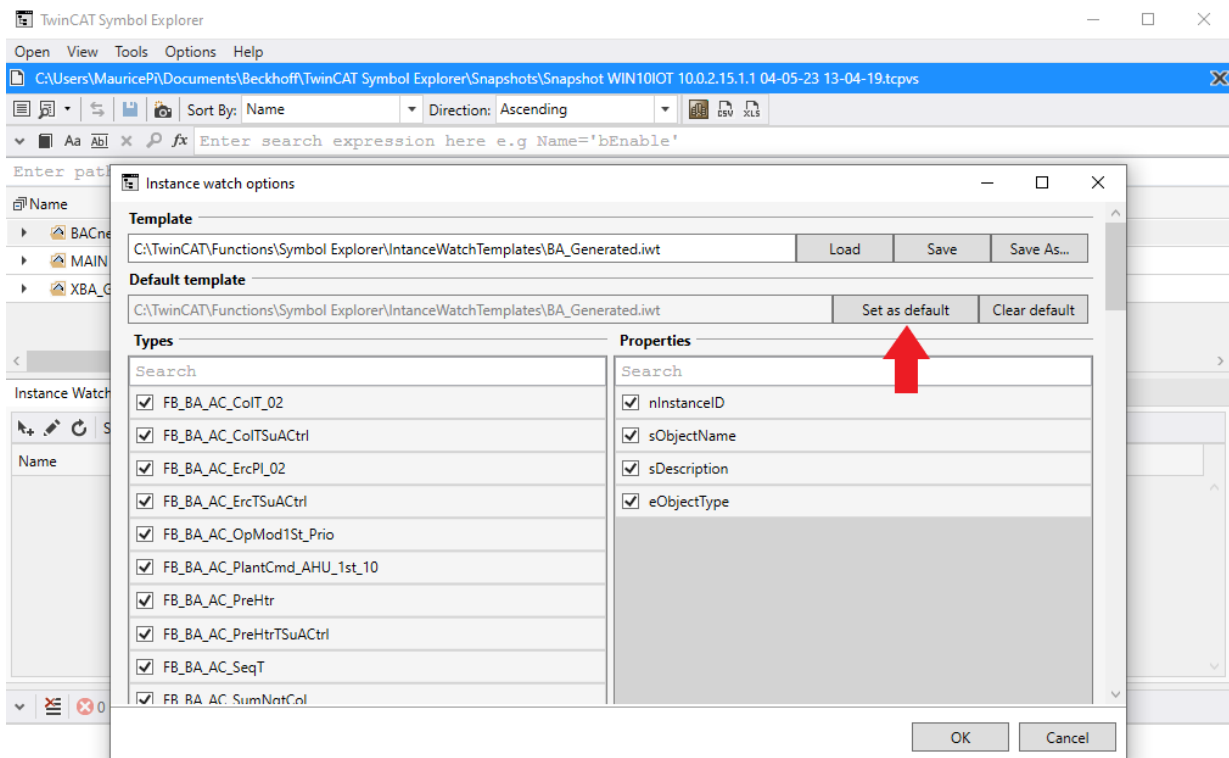
2. To load a template press the button **Instance Watch Options (pen)**.

3. A window opens. Load a template using the button **Load**. Here a file chooser dialog opens where you have to select the *.iwt file.



4. If you want this template to be loaded every time the instance watch is started, set this by pressing the button **Set as default**. The path of the *.iwt file should appear on the left, next to the button at **Default template**.

To undo this change click the button **Clear default**.



5. Confirm the **Instance watch options** dialog with **OK**.

⇒ The instance watch is loaded with the template.

7.2.6 Extensions

Extensions for Symbol Explorer are documented here.

7.2.6.1 BaExtension

This plugin for the symbol explorer creates a filtered object list of symbol types, which consist of types of the object and views of TF8040. The object list is saved as an Instance Watch Template and can be viewed in the Instance Watch Viewer in the Symbol Explorer.

This allows an Instance Watch to be created in a Live View or snapshot that shows all objects of the TF8040 in a filtered manner.

This list can now be exported to Excel, for example, properties can be changed and read in again.

These changes can be incorporated at runtime.

Setup requirements

OS Version: see Symbol Explorer

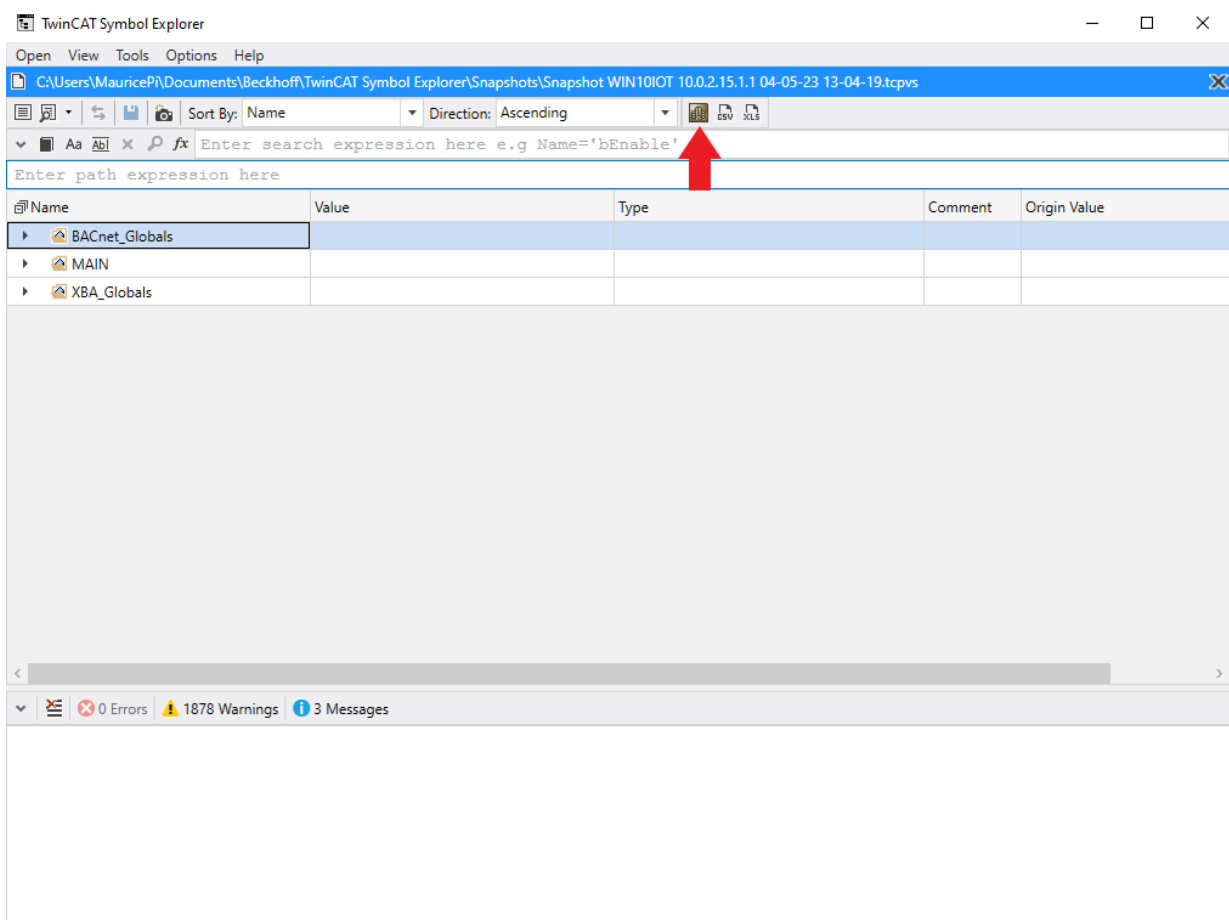
Symbol Explorer Version: V1.455.5.0

Description of the interface

Generation of the template

The extension adds a button to the Symbol Explorer toolbar.

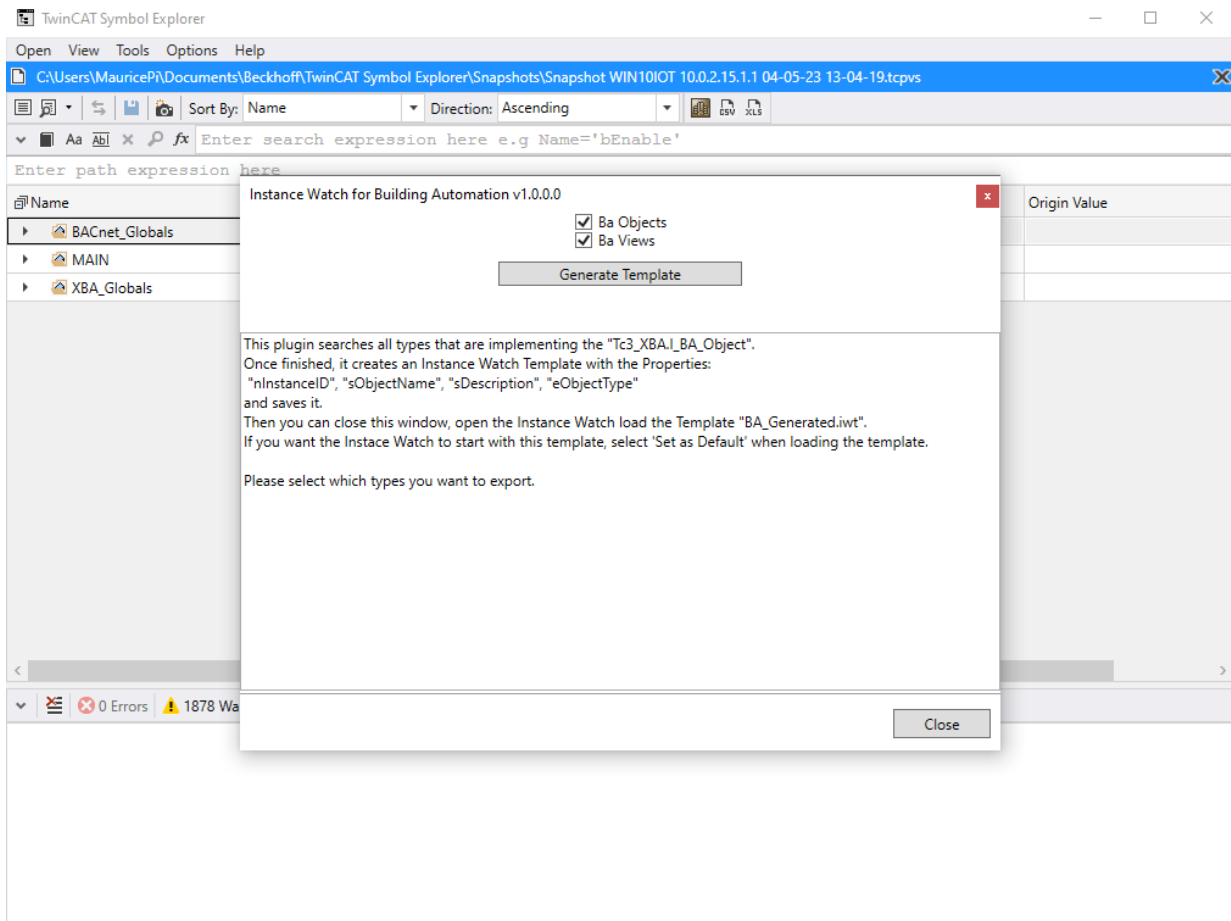
1. A click on this button opens a dialog.



2. The extension version number is displayed in the title. This dialog consists of two checkboxes, a button, an output field and a button to close the dialog. The checkboxes can be used to select whether objects that implement the interface of the **Ba Objects** or

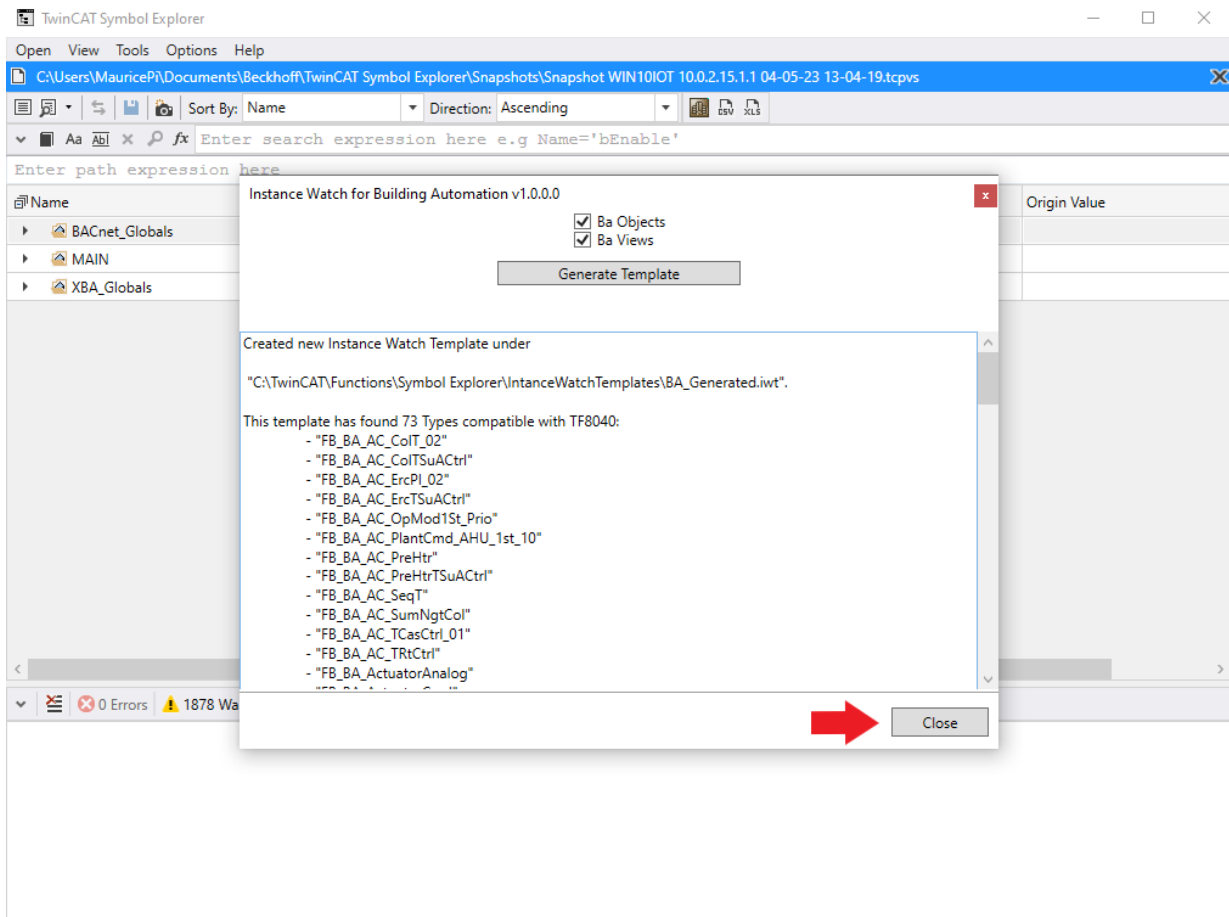
the **Ba Views** are to be included in the template. To add all TF8040 objects to the template, both checkboxes must be selected.

Press the **Generate Template** button to generate a template.

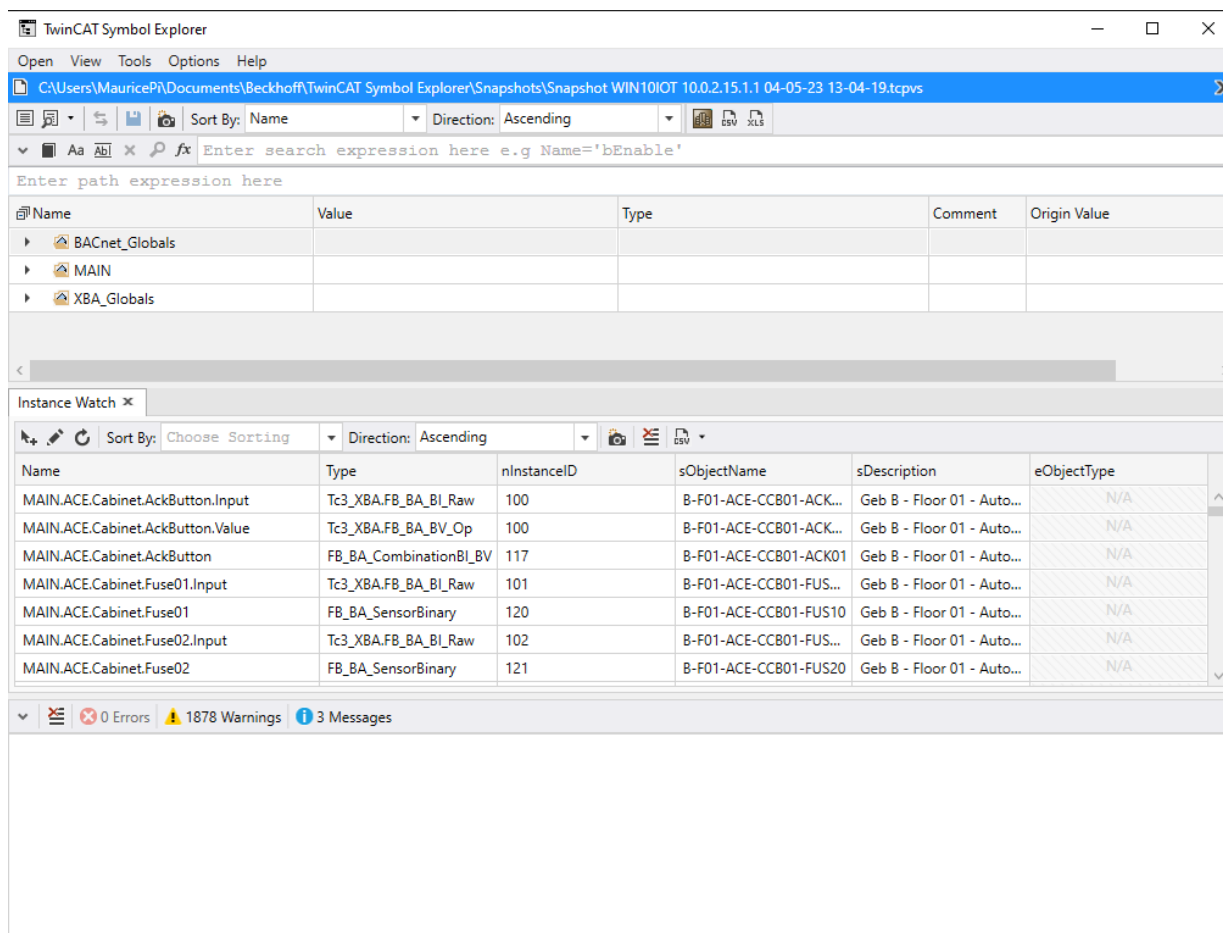


⇒ After generation the template is saved under: *C:\TwinCAT\Functions\Symbol Explorer\IntanceWatch Templates\BA_Generated.iwt.*

3. Close the dialog with **Close**.



⇒ The result looks like this:



Open the template:

see chapter "[Open Instance Watch Template \[► 1060\]](#)"

7.2.7 Examples of regular expressions

For an understanding of the expressions used for filtering and sorting, the following link opens a collection of frequently used regular expressions:

[Overview of regular expressions](#)

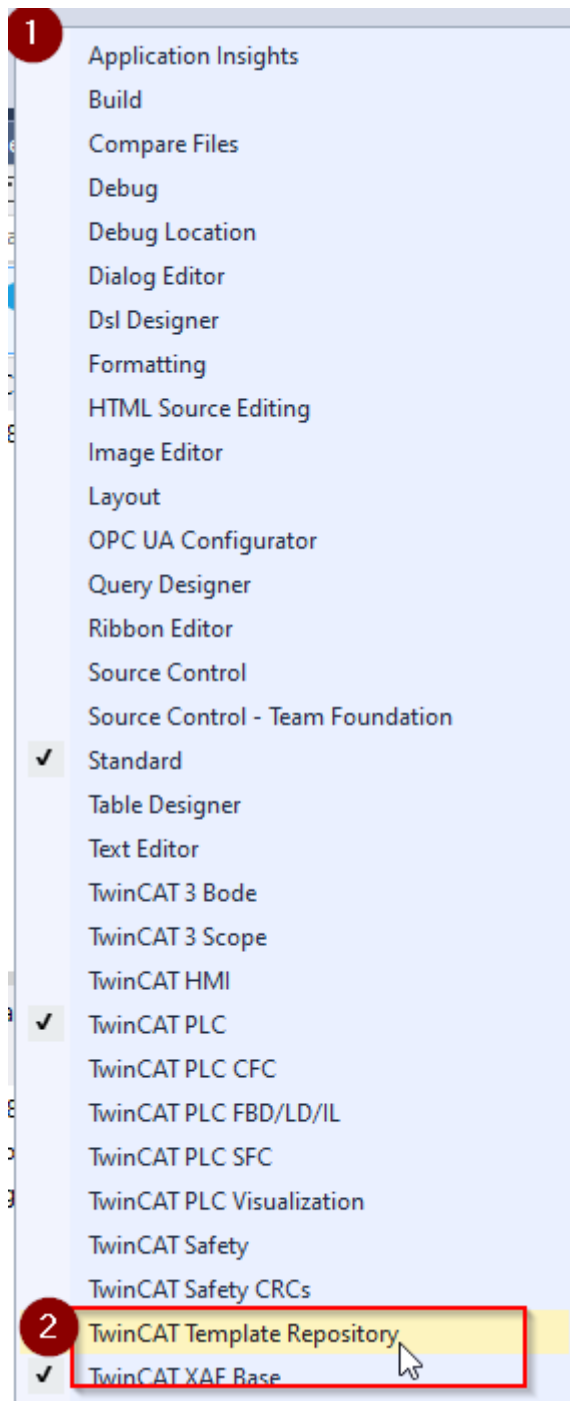
7.3 Template Repository

The Template Repository is an app for managing templates.

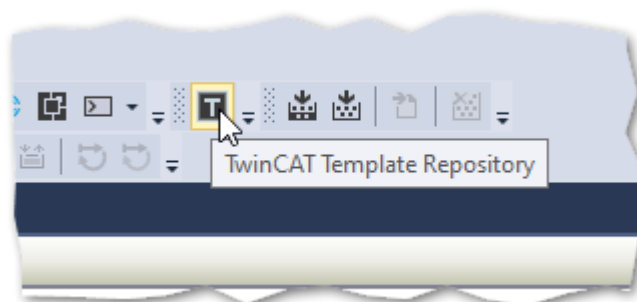
Starting the app in the development environment

- ✓ The TF8040 setup automatically installs the Template Repository in the TwinCAT development environment (XAE Shell or Visual Studio).
- 1. Right-click on a free field in the toolbar to open a context menu.

2. Select the app **Template Repository** from the context menu.

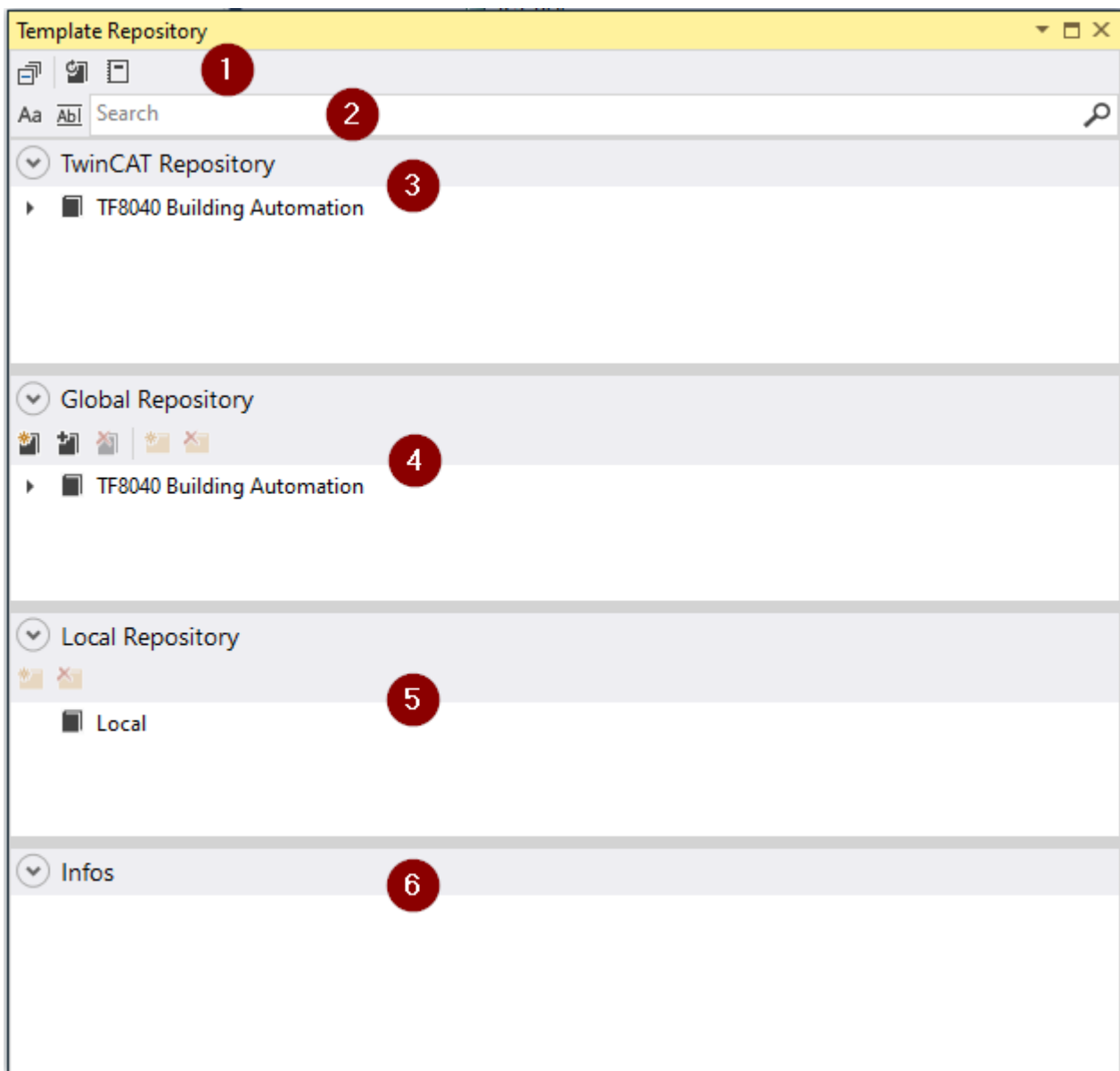


3. Start the app by clicking on the icon **TwinCAT Template Repository**.
⇒ The main window opens.



Main Window

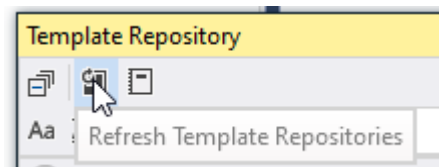
The main window is divided into 6 different areas:



1 Menu bar

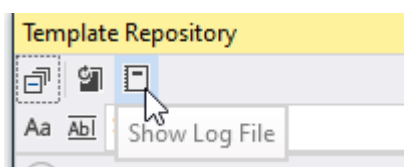
The following functions are available in the menu bar:

- **Refresh Template Repositories** updates all repositories



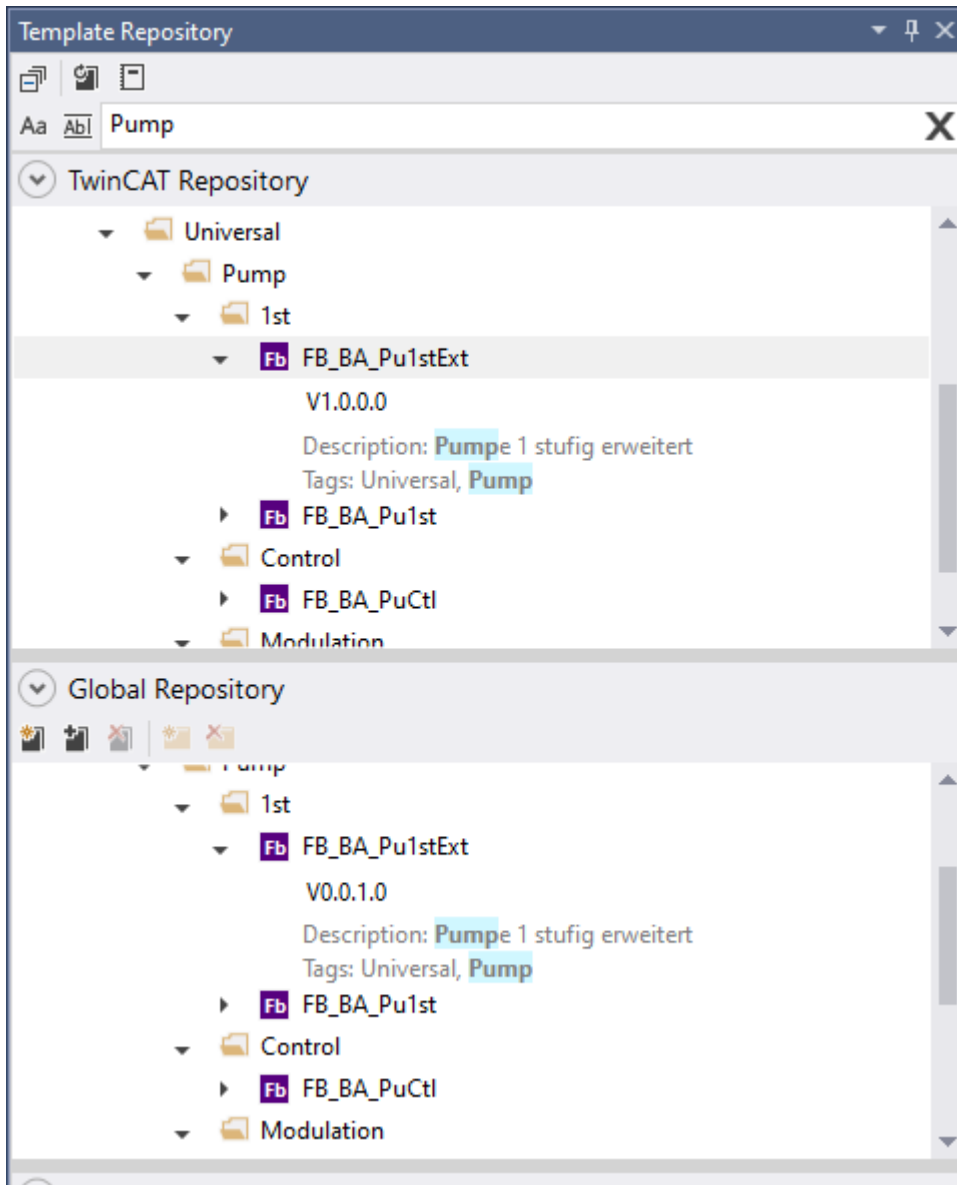
- **Show LogFile** opens the log file dialog

Detailed log information about the software version used and errors are displayed here.



2 Search bar

The search bar searches the included repositories for keywords (*tags*) or names within the templates. The search results are displayed in the respective areas.



3 TwinCAT Repository area

The TwinCAT repository area contains all templates that are supplied with the TF8040 installation. Templates can be taken from this area but not added.



With the TF8040 installation, you receive a repository with templates. In order to use them for your project-specific applications, you must temporarily load the repository into the global area.

4 Global Repository Area

The Global Repository area serves as a storage location for all customer-specific templates that can be used by several developers across projects. Here templates can be taken and added across projects. The storage location can be a network drive in the company network or any location on the local hard disk, for example.

5 Local Repository area

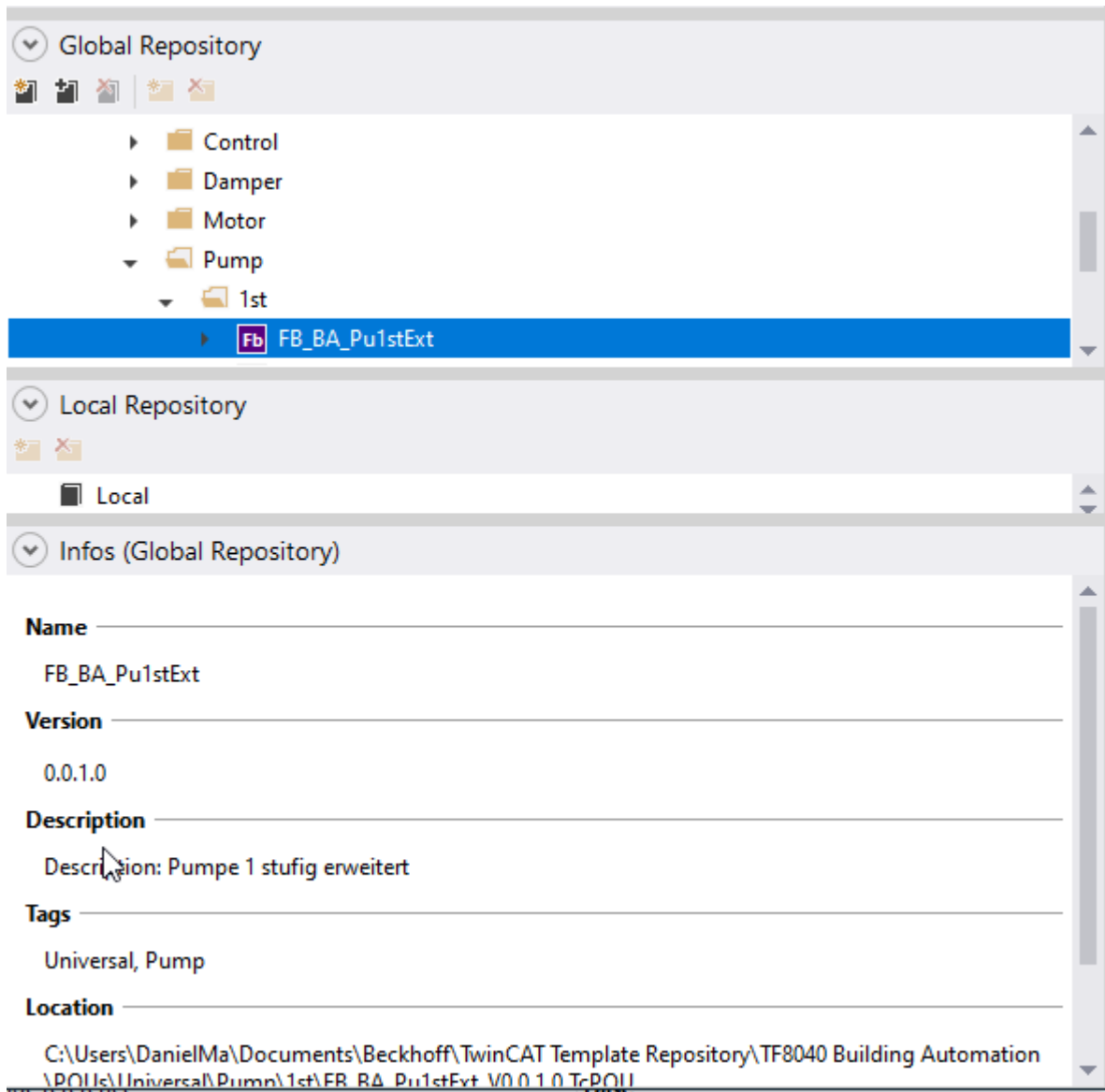
All templates that are used in the local solution are stored in the Local Repository area.



Templates can be exchanged between the local and global area.

6 Info area

The notification area displays detailed information about the currently selected templates.

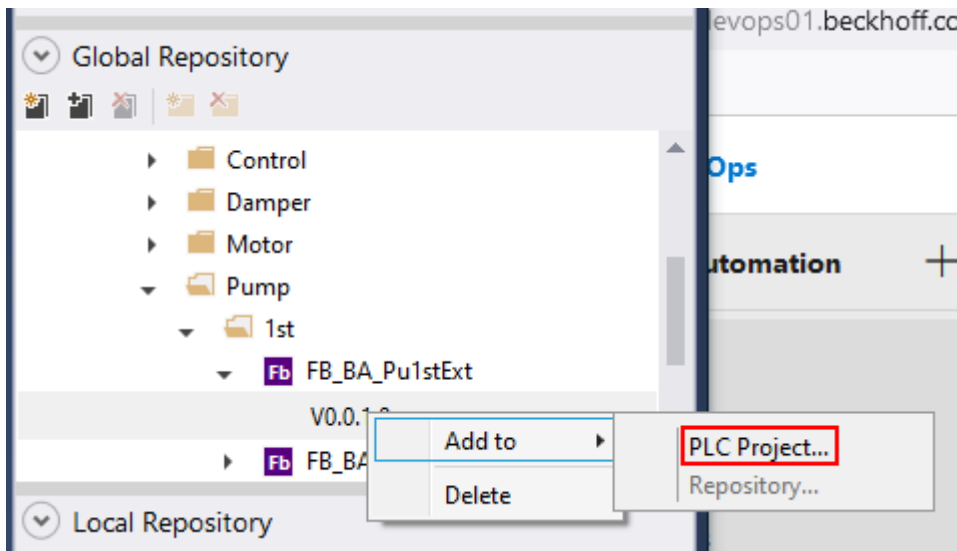


Adding templates to a PLC project

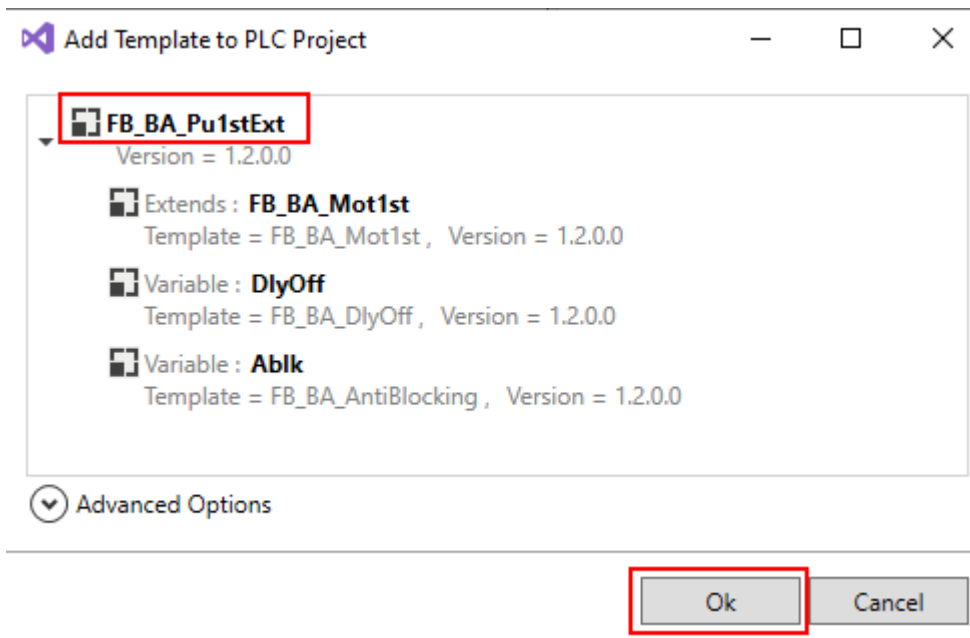
Templates can be integrated into the current solution in two ways:

- ✓ **Double click** on the desired template

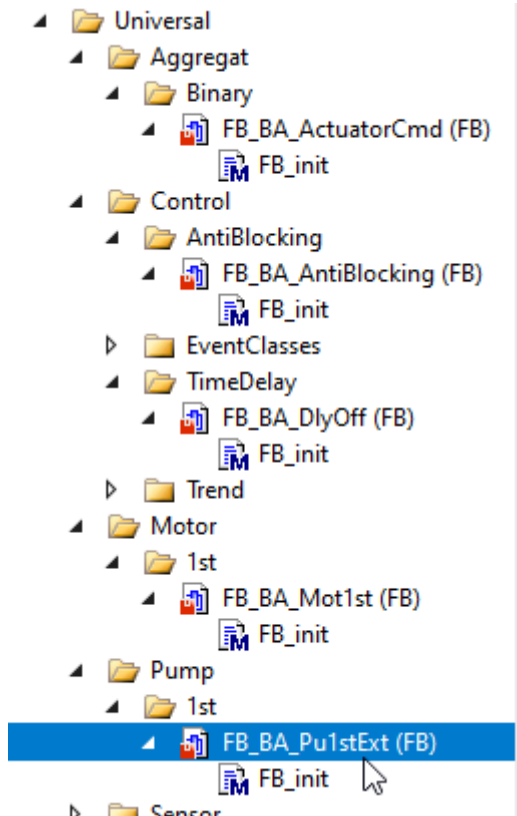
- ✓ Right click on the desired template > Add to > PLC Project



1. The dialog window **Add Template to PLC Project** opens. Select the template and confirm this with **OK**.



⇒ The desired template including all dependent subtemplates is now in the PLC project.



Call and instantiation of a template

After the desired template has been added to the project from the repository, it can be instantiated and called in the project applications.

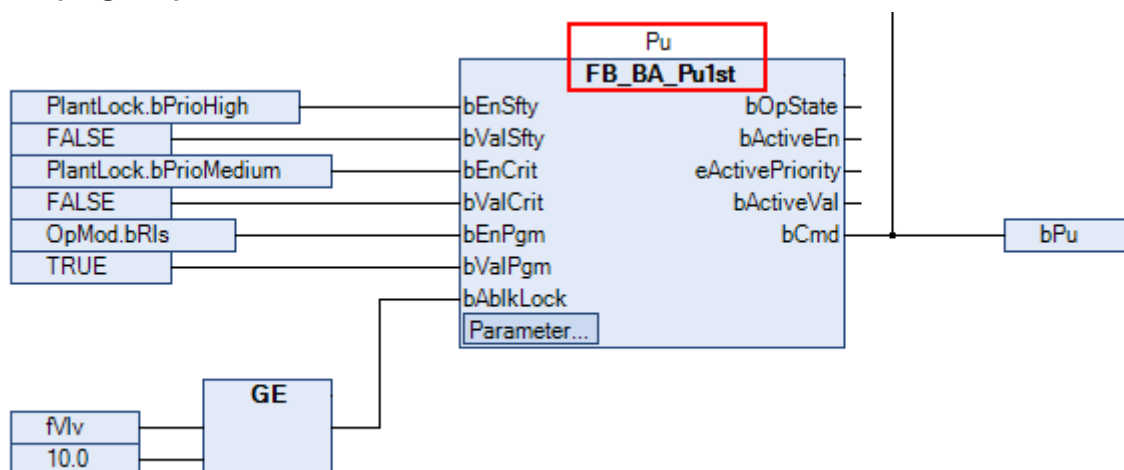
Instantiation in the declaration part:

```

22 | Pu : FB_BA_Pu1st;
23 | Sn : FB_BA_H_HrdCir_Sn;

```

Call in the program part:



8 Appendix

8.1 Third-party components

This software contains third-party components.

Please refer to the license file provided in the following folder for further information:

\\TwinCAT\\Functions\\SymbolExplorer\\Licenses.

8.2 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963-157
e-mail: support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963-460
e-mail: service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963-0
e-mail: info@beckhoff.com
web: www.beckhoff.com

More Information:
www.beckhoff.com/tf8040

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com

