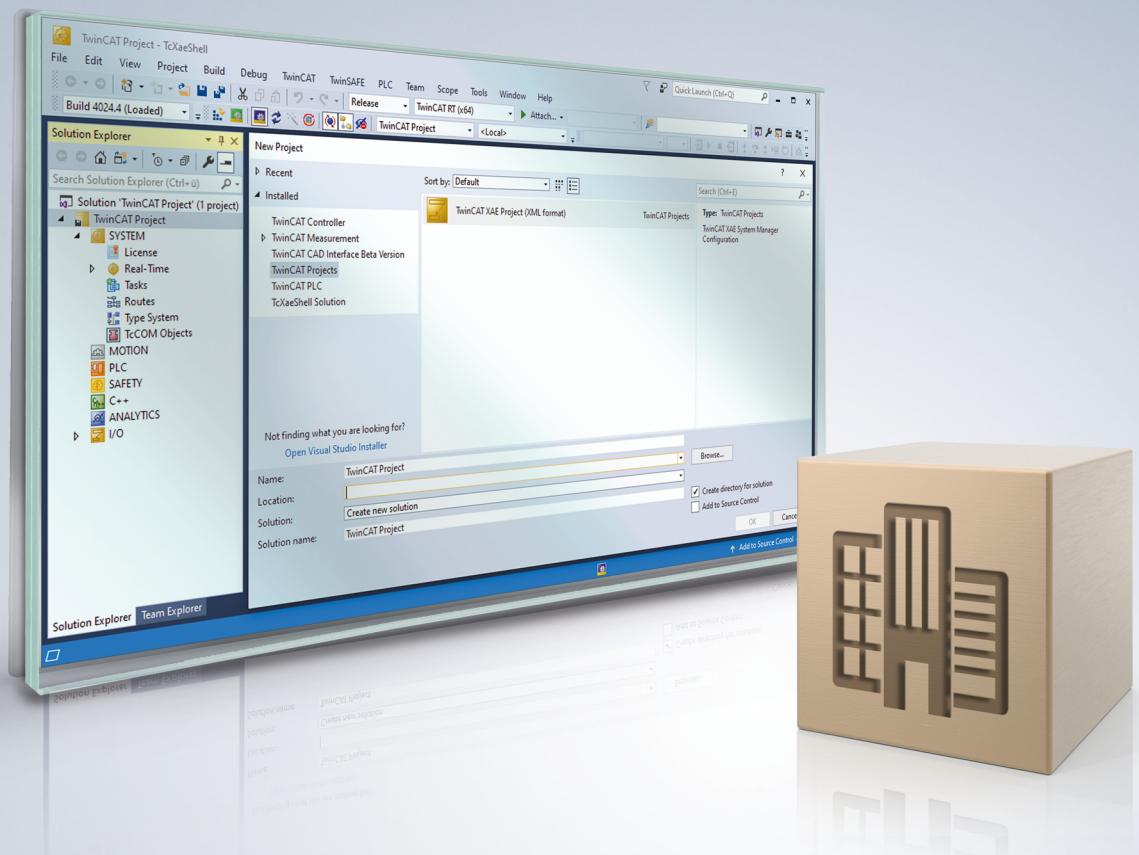
BECKHOFF New Automation Technology

Manual | EN

TF8040

TwinCAT 3 | Building Automation



2025-05-26 | Version: 1.14.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled 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 notice.
Claims to modify products that have already been supplied may not 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 and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

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

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

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 Overview

In order to meet the high demands of building automation, such as comfort, energy efficiency, low investment and operating costs, and a fast return on investment, it is essential to have a consistent, coordinated control system for the automation of all technical systems.

With TwinCAT 3 Building Automation, Beckhoff has developed a software product that reduces engineering time and integrates all essential functions for all technical systems of modern building automation. Extensive software libraries and tools continue the idea of the modular Beckhoff automation toolkit at the software level. The new software suite essentially comprises three basic functions:

TwinCAT 3 BA PLC Libraries [▶ 98]:

Basic functions for all technical systems.

TwinCAT 3 BA PLC Templates [▶ 696]:

Function templates for all technical systems.

TwinCAT 3 BA Tools [▶ 1285]:

Software tools to optimize the implementation of building automation projects in terms of data processing, commissioning, adjustment and maintenance.

By using the function TwinCAT 3 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 Installation

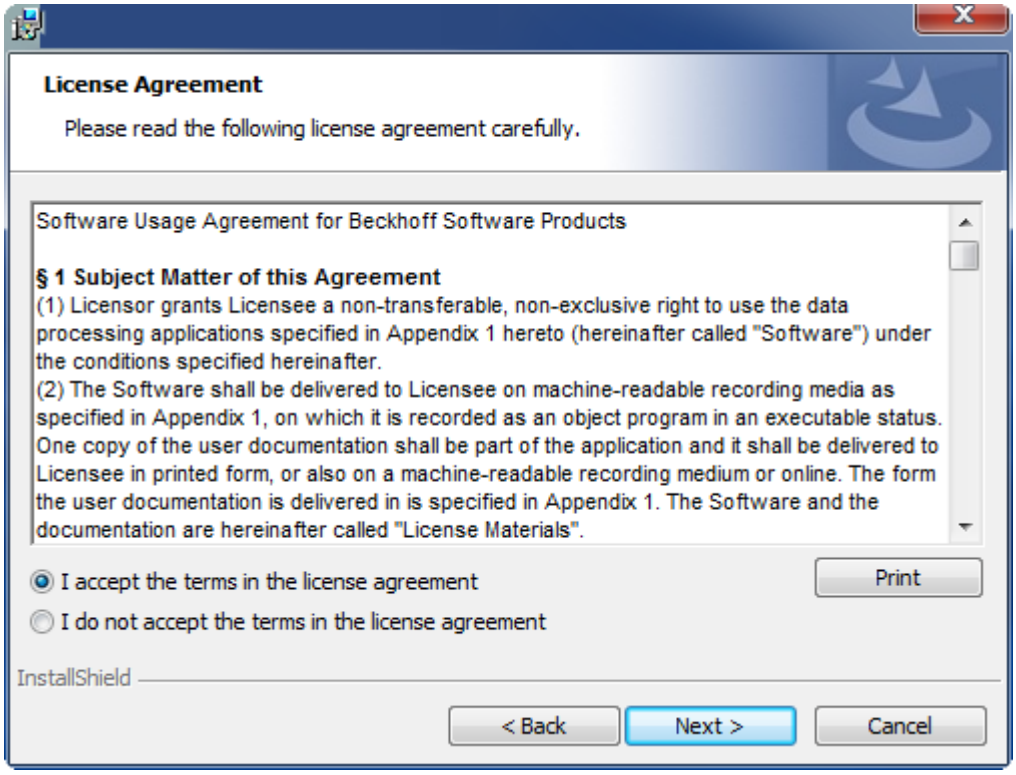
The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.

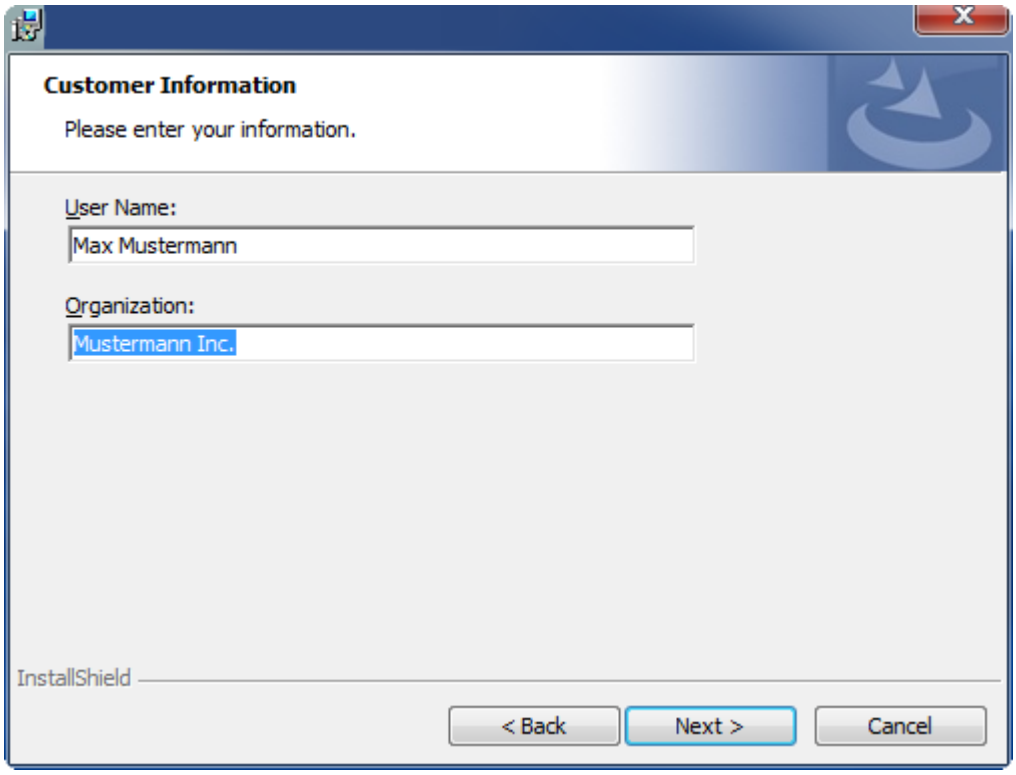
1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.

⇨ The installation dialog opens.

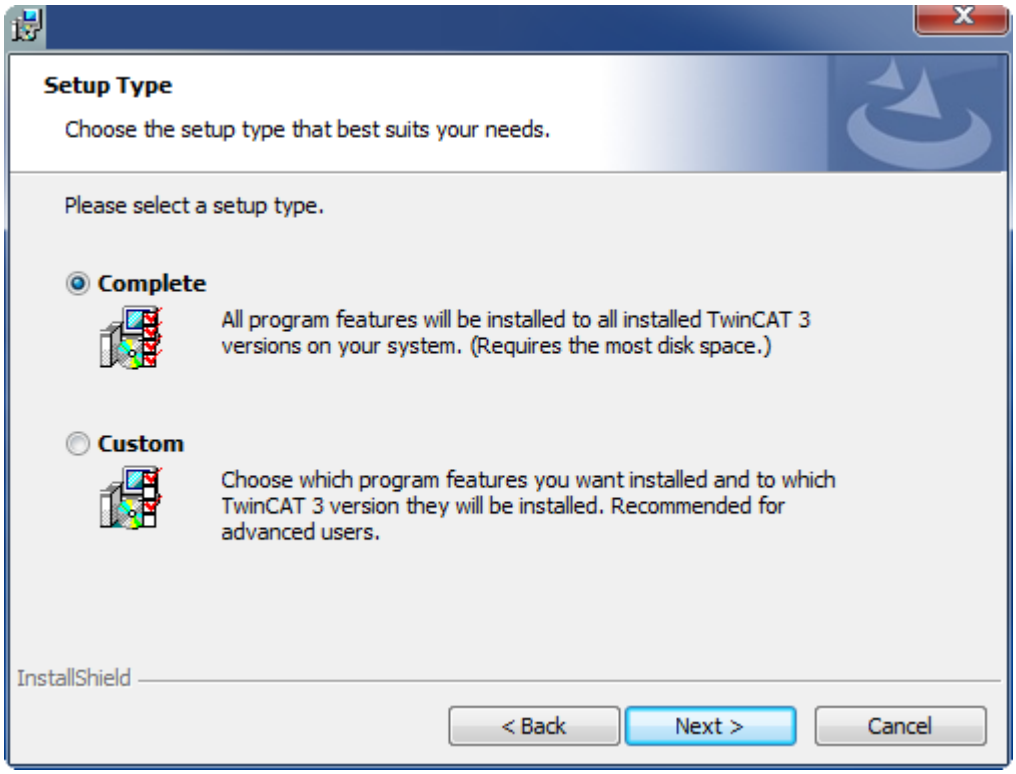
2. Accept the end user licensing agreement and click **Next**.



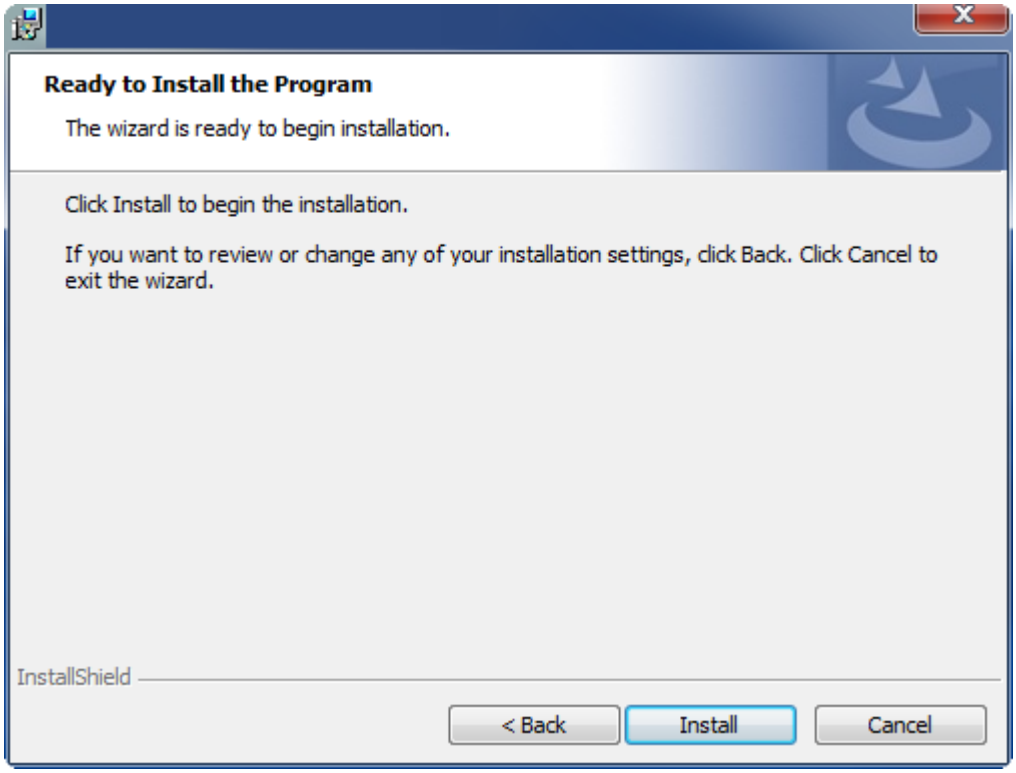
3. Enter your user data.



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**.

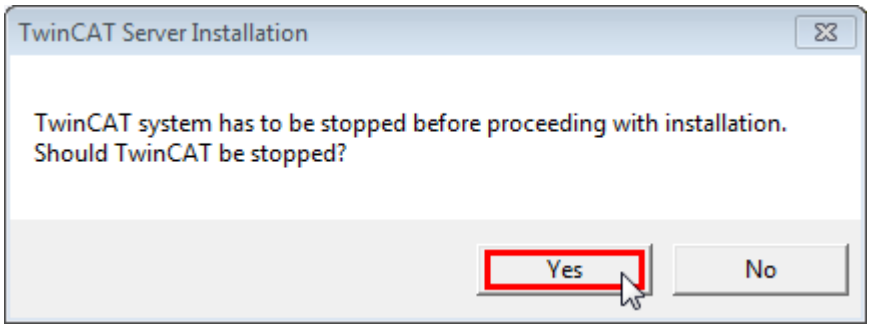


5. Select **Next**, then **Install** to start the installation.

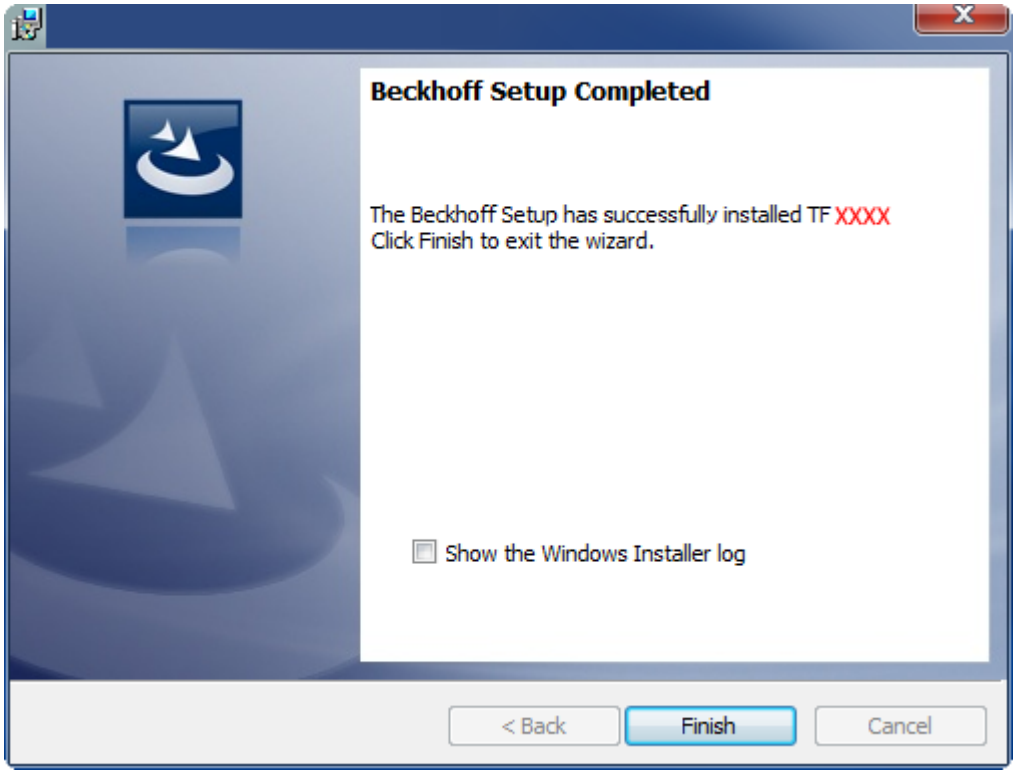


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

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



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



⇨ The TwinCAT 3 Function has been successfully installed.

2.4 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

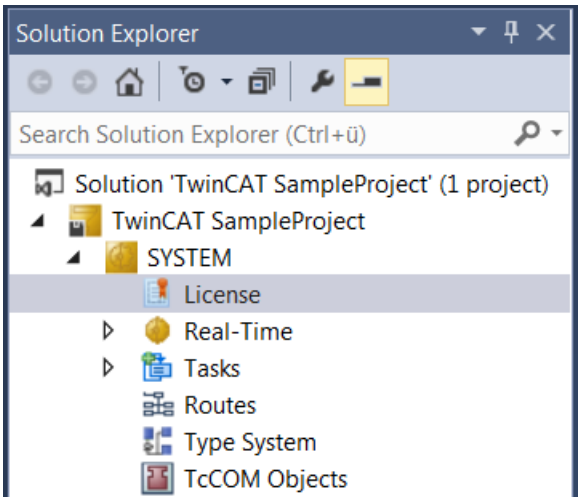
A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

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.

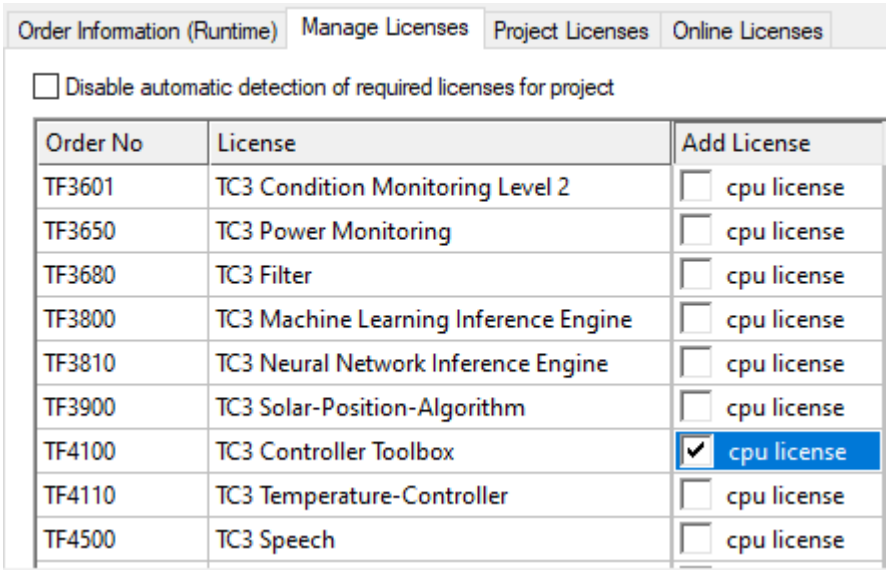
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

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").

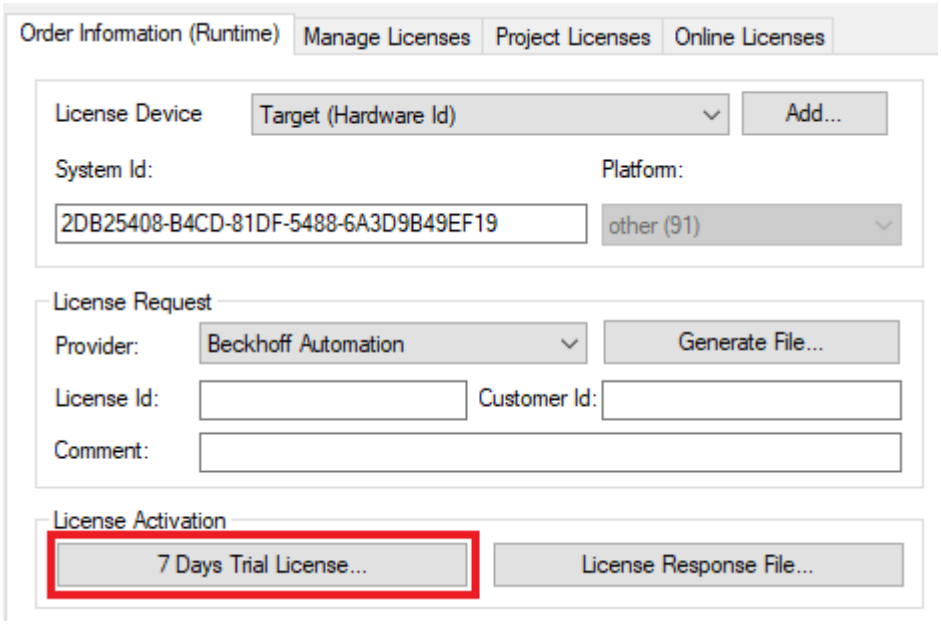


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu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

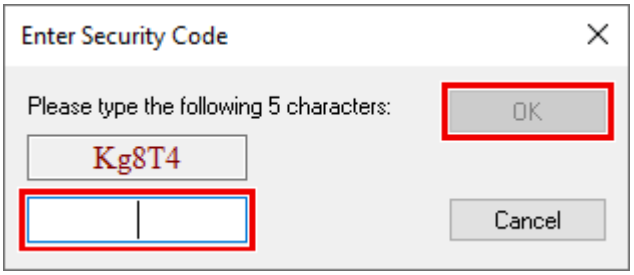
6. Open the **Order Information (Runtime)** tab.

   ⇨ 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.



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



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.

2.5 System requirements

Requirements

The TF8040 consists of several software components with different system requirements.

PLC libraries

The PLC libraries contained in TF8040 require TwinCAT version >= 3.1.4024.35 and run on the following operating systems:

- Windows 10
- Windows CE

A TF8020 license is required to use the PLC libraries.

Tools

The tools included in the TF8040 run on the **Windows 10** operating system.

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

TcHmiBa

Here you will find the system requirements for TcHmiBa [▶ 1121].

2.6 Energy efficiency

The influence of building automation and building management is described in DIN EN ISO 52120-1:2019-12.

This DIN EN ISO 52120 standard contains a structured list of building automation and technical building management functions that have an impact on the energy efficiency of buildings.

The functions of DIN EN ISO 52120 are categorized and listed in a structured manner according to technical systems and so-called Building Automation and Control (BAC) functions.

A factor-based method makes it easy to estimate the impact of these BA functions on the overall energy efficiency of a building. The application of this simplified procedure shows that energy savings of 30% are possible using the methods listed therein.

TwinCAT 3 Building Automation offers functions as described in DIN EN ISO 52120.

This makes it possible to estimate the potential energy savings through the use of plant and room automation with TwinCAT 3 Building Automation. Templates for plant and room functions in TF8040 facilitate the planning, execution and subsequent operation of buildings.

An essential prerequisite for the implementation of an energy efficient and sustainable building automation system is the integration of the building automation system. Plant and room automation with all its technical systems must be integrated into one system. The templates from TF8040 TwinCAT 3 Building Automation implement the energy efficiency functions described in the DIN EN ISO 52120 standard.

3 Concepts

This chapter contains introductory information to help you get started with TwinCAT 3 Building Automation. Concepts serve as drafts and suggestions for project planning of a building automation system with TwinCAT 3 Building Automation.

3.1 User roles

The base 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 [▶ 1120]) 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 and can only read current values that are in the Tc3_XBA [▶ 99] 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.

Adjustment of access rights from the PLC

It is possible to define access rights from the PLC.

However, this only works initially, i.e. not at runtime, and then refers to all objects that have this property.

To do this, an instance of FB_BA_Param [▶ 159] is created in a suitable part of the program, preferably in MAIN, and pre-initialized with the corresponding write and read rights.

The initialization is sufficient - there is no further call in the actual program part.

Sample:

In the following sample, the read and write rights are to be changed for two parameters:

- Acknowledge (*bAcknowledgeRm*): Read from Basic role, write from Expert role.
- DefaultValue (*bDefaultValue*, *fDefaultValue*, *nDefaultValue*): Read from Internal role, write from Advanced role.

This adjustment requires the instantiation of a FB_BA_Param [▶ 159], here called *Parameters*, which is structured as follows:

```
Parameters : FB_BA_Param :=  (DefReadAccess := F_BA_ParameterRolesBuilder()
            .Set(E_BA_Parameter.eAcknowledgeRm, E_BA_Role.eBasic)
            .Set(E_BA_Parameter.eDefaultValue, E_BA_Role.eInternal)
            .Build(),

        DefWriteAccess := F_BA_ParameterRolesBuilder()
            .Set(E_BA_Parameter.eAcknowledgeRm, E_BA_Role.eExpert)
            .Set(E_BA_Parameter.eDefaultValue, E_BA_Role.eAdvanced)
            .Build());
```

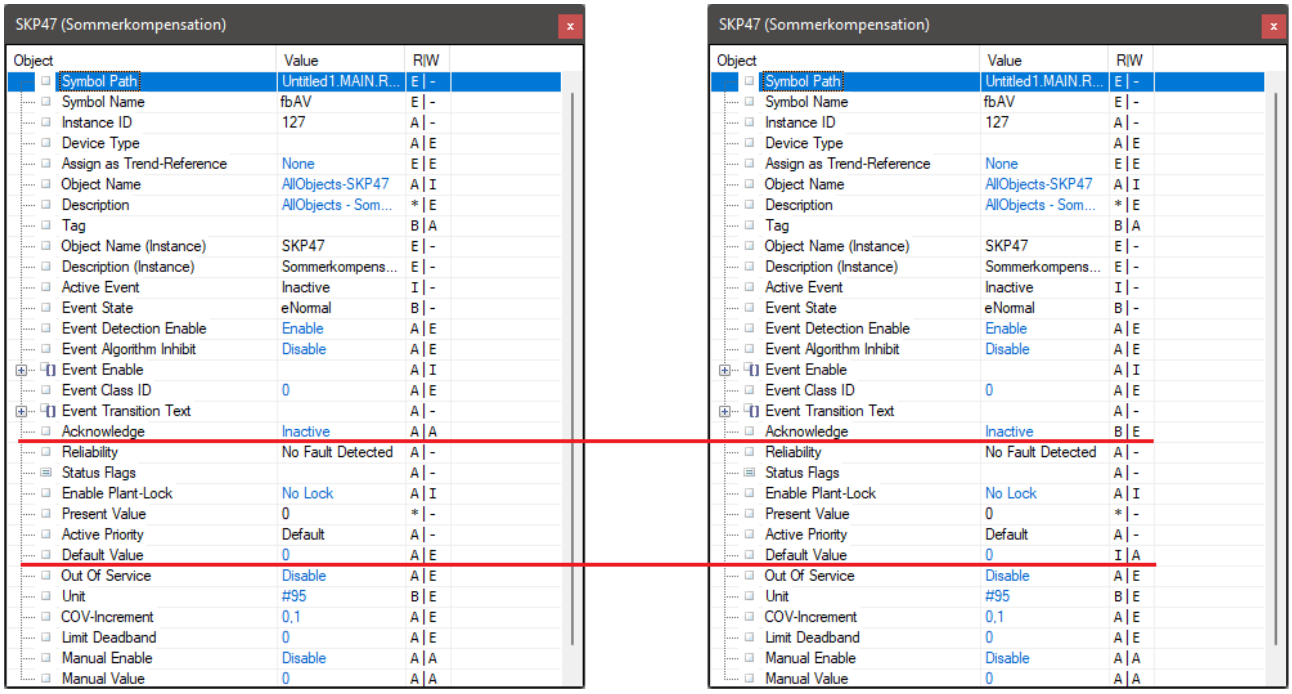
The *DefReadAccess* and *DefWriteAccess* properties each represent an array of the various parameters with the associated roles as minimum access and are only used to change the roles, which are, however, already predefined.

The Set methods change the minimum access role (E_BA_Role [▶ 99]) for a specific parameter (E_BA_Parameter [▶ 108]). A Set method must be added for each parameter whose access right is to be changed.

The parameterization of the properties is completed by calling the *Build* method. This applies to both the *DefReadAccess* read access property and the *DefWriteAccess* write access property.

The above implementation changes the read and write access of **all** objects that have the properties *bAcknowledgeRm* and DefaultValue (*bDefaultValue*, *fDefaultValue*, *nDefaultValue*). It is also possible to change only the read or only the write access; the change does not have to be made in pairs.

Using the example of an AV object, the change in the Site Explorer is as follows:



3.2 Communication

The communication protocols OPC UA; Modbus TCP; MQTT; and ADS are all used together to transfer information relating to the values of symbols and structures of the PLC.

In contrast to BACnet, they do not describe any standardized communication objects, meaning that the structure of the data in many projects is very specific and individual. Independent of BACnet, TwinCAT TF8040 is also very helpful for such projects, as the Tc3_BA2 [▶ 267] basic library contains a large number of function blocks that are required for a building automation project regardless of the communication protocol used.

The BACnet communication protocol has become the global standard for building automation. In contrast to the other protocols, BACnet provides a very detailed standard with its objects and services.

TwinCAT Building Automation makes use of the object-oriented structure of BACnet.

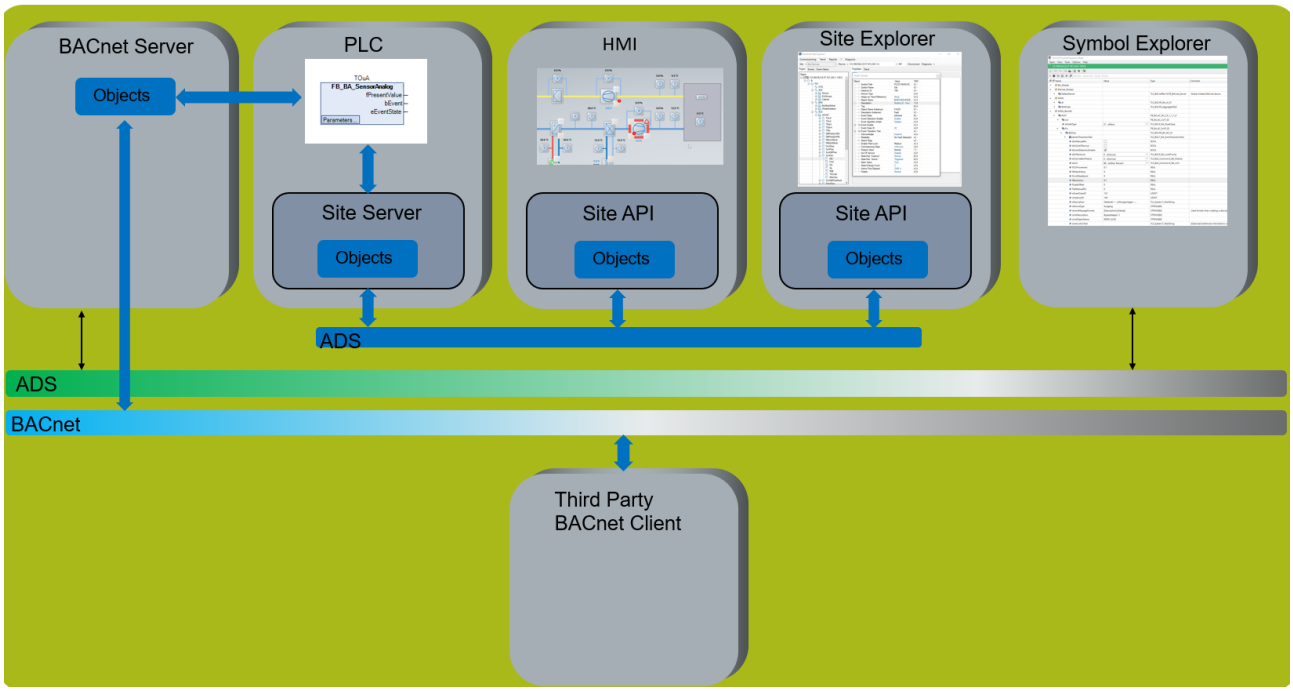
The library Tc3_XBA [▶ 99] contains a framework called the base framework [▶ 30].

It is used to map a project structure [▶ 38] and numerous other generic functions.

The Site Server within the PLC organizes the ADS communication to the TwinCAT HMI.

The Site API is the counterpart of the Site Sever and realizes the generic representations in the HMI.

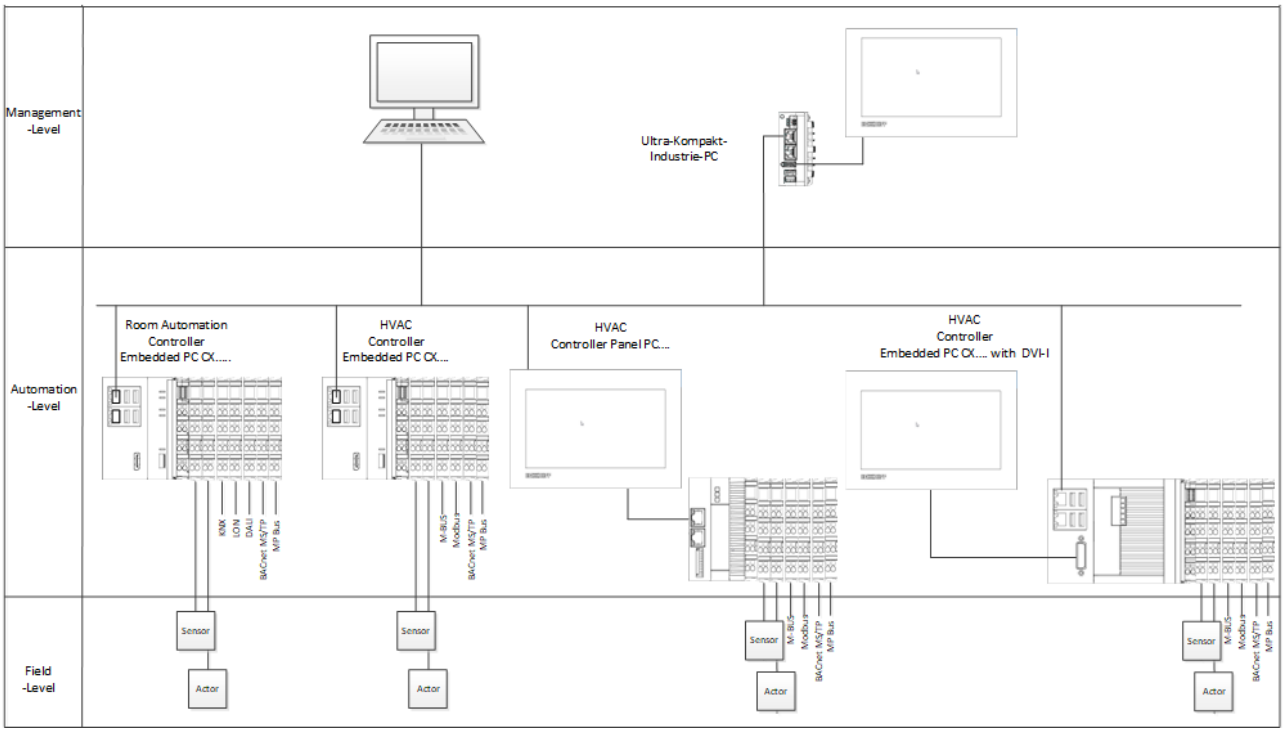
The Site Explorer service tool also communicates with the automation stations using the Site API via ADS.



3.3 System structure

The Beckhoff Industrial PCs meet all requirements with regard to the automation of heating, ventilation and air conditioning as well as room automation [▶ 21].

The large portfolio of Beckhoff Industrial PCs results in excellent scalability of the building automation system. The high industrial quality standard guarantees high system availability and investment security.



All PCs are programmed with TwinCAT Engineering TE1000 regardless of their use.

By using the TwinCAT functions TF8020 BACnet and TF8040 Building Automation, all industrial PCs in the Beckhoff portfolio become a powerful automation station for building automation. (The TwinCAT function TF8040 is currently not available for the small controllers of the CX7xxx series).

The choice of processor allows the automation station to be adapted to the requirements of the automation station to be configured with fine granularity in terms of its computing power.

The design of the industrial PC is also freely selectable.

The embedded PCs from the CX series are ideal for this purpose.

Bus terminals (K-bus) or EtherCAT Terminals (E-bus) can be attached on the right-hand side of the embedded PC.

Depending on the version of the embedded PC, it can be equipped with a DVI-I interface.

A control panel can be connected if required. The industrial PC can be used as an automation station for a control cabinet with local operation via a touch panel. The function TF2000 HMI server must then be installed on the IPC.

Very powerful ultra-compact PCs are available for higher-level tasks within the automation system, such as the MBE (management and operating unit).

The TwinCAT HMI can thus act not only as a local control of an IPC, but also as a web HMI for a large number of automation stations.

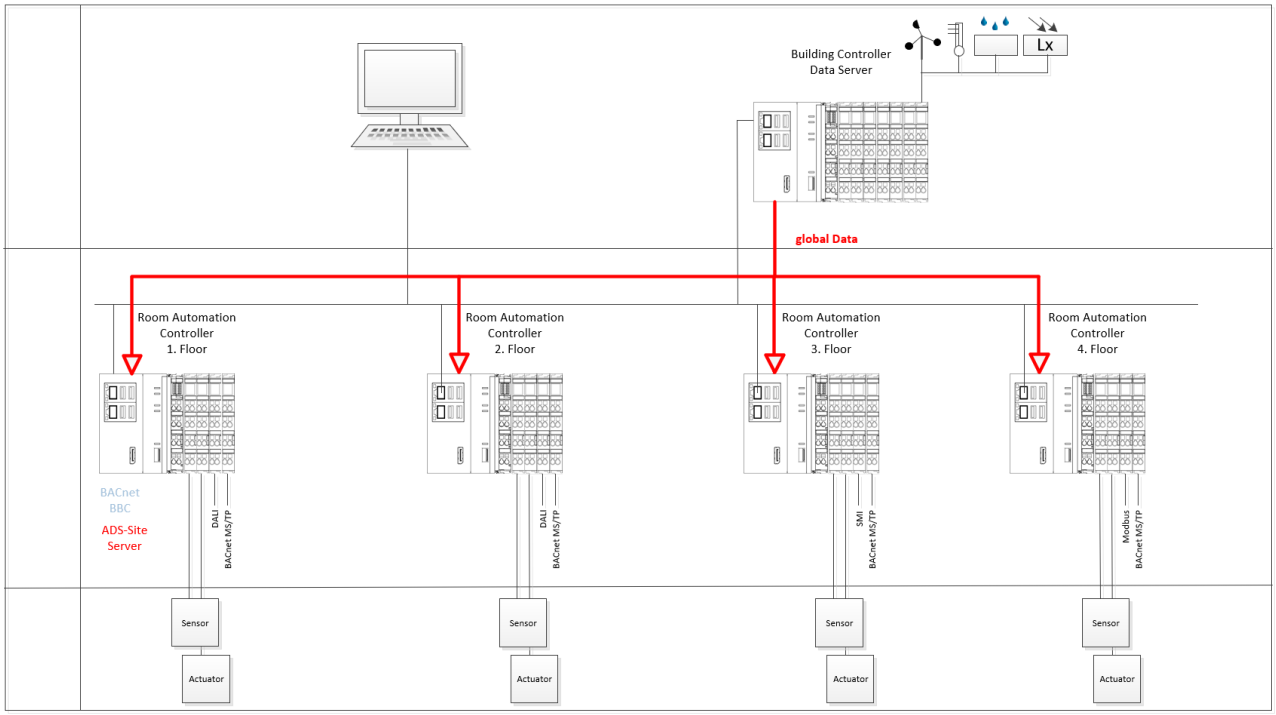
Whether it is plant or room automation, the same hardware and software is used in all applications. Interoperability of all building automation components is therefore ensured.

3.4 ADS communication within the templates

TwinCAT Building Automation offers a simple concept for data distribution within a building automation network.

Within this concept, the Building Controller serves as a central data distributor.

It should consist of a powerful IPC.



The following data is recorded, calculated and provided within the Building Controller.

Weather data

- Outside temperature
- Precipitation report
- Dew point temperature
- Air pressure
- Air humidity
- Brightness
- Global radiation
- Wind speed
- Wind direction

global events

- Burglary
- Fire

global setting parameters

- Frost setpoint
- Frost protection limit value
- damped value of the outside temperature
- heating enable of the unattenuated outside temperature
- heating enable of the attenuated outside temperature

global time schedule with the active energy levels

- Protection or Economy or PreComfort or Comfort

Setpoints of the energy levels

- Protection Heating
- Economy Heating
- PreComfort Heating
- Comfort Heating

- Protection Cooling
- Economy Cooling
- PreComfort Cooling
- Comfort Cooling

Operation modes

- Default (standard)
- Night watchman tour
- Cleaning mode

Sun protection for the entire building

- Positioning telegram resulting from the user functions: fire, burglary, icing,
  - Position
  - Angle
  - Priority
- Enabling the thermal automatic (global radiation-dependent)
- Enabling the twilight automatic (depending on outside light)
- Central reset of manual operation of all blind actuators

Sun protection selective per facade

- Positioning telegram resulting from the user functions:
  - Position
  - Storm protection (depending on wind direction)
  - Maintenance
  - Thermal automatic (global)
  - Position or slat angle with active sun protection
  - Enabling the sun protection dependent on outside light and sun position

In the building controller, all data-providing templates write their data to the Site GVL [▶ 1116].

The data is transferred from this data pool of the Site GVL [▶ 1116] to the clients via ADS.

The data is distributed according to technical system.

Building automation in general:

Weather data, global events and global setting parameters for HVAC systems are generated in the function block *FB_BA_BuildingAutomationServer* and written to the Site GVL [▶ 1116].

The function block *FB_BA_AdsComServer* publishes this data via ADS.

In a client, this data is in turn received via a Subscriber and written into the local Site GVL [▶ 1116] of the automation station.

Doors,Gates,Windows,Sun protection

Facade data and global sun protection data are generated in the function block *FB_BA_DGWSPServer* and written to the Site GVL [▶ 1116].

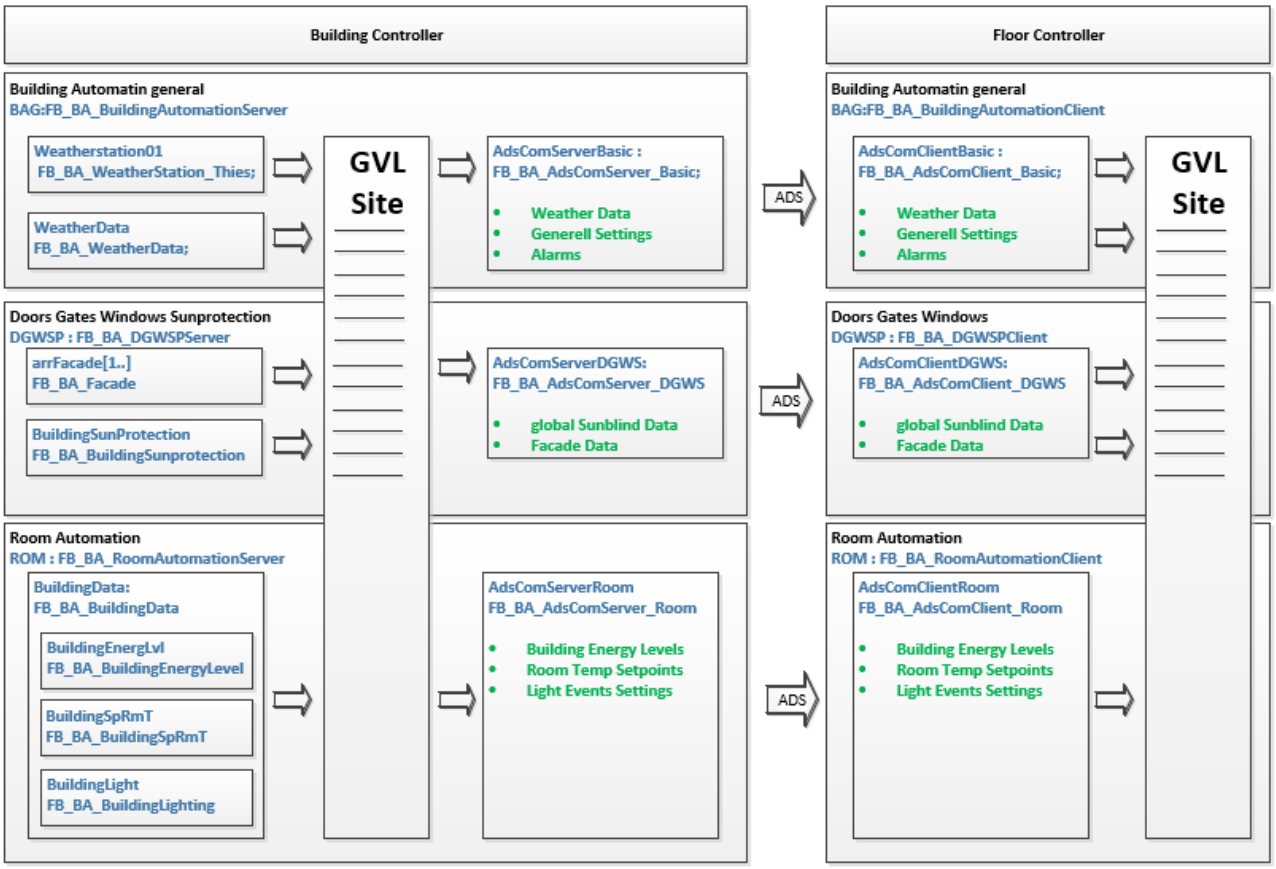
The function block *FB_BA_AdsComServer_DGWS* publishes this data via ADS.

In a client, this data is in turn received via a Subscriber and written into the local Site GVL [▶ 1116] of the automation station.

Room automation

Room automation data, e.g. global room temperature setpoints and energy levels, are generated in the function block *FB_BA_RoomAutomationServer* and written to the Site GVL [▶ 1116].

The function block *FB_BA_RoomAutomationClient* receives this data and writes it to the local Site GVL [▶ 1116] of the automation station.



3.5 Room automation

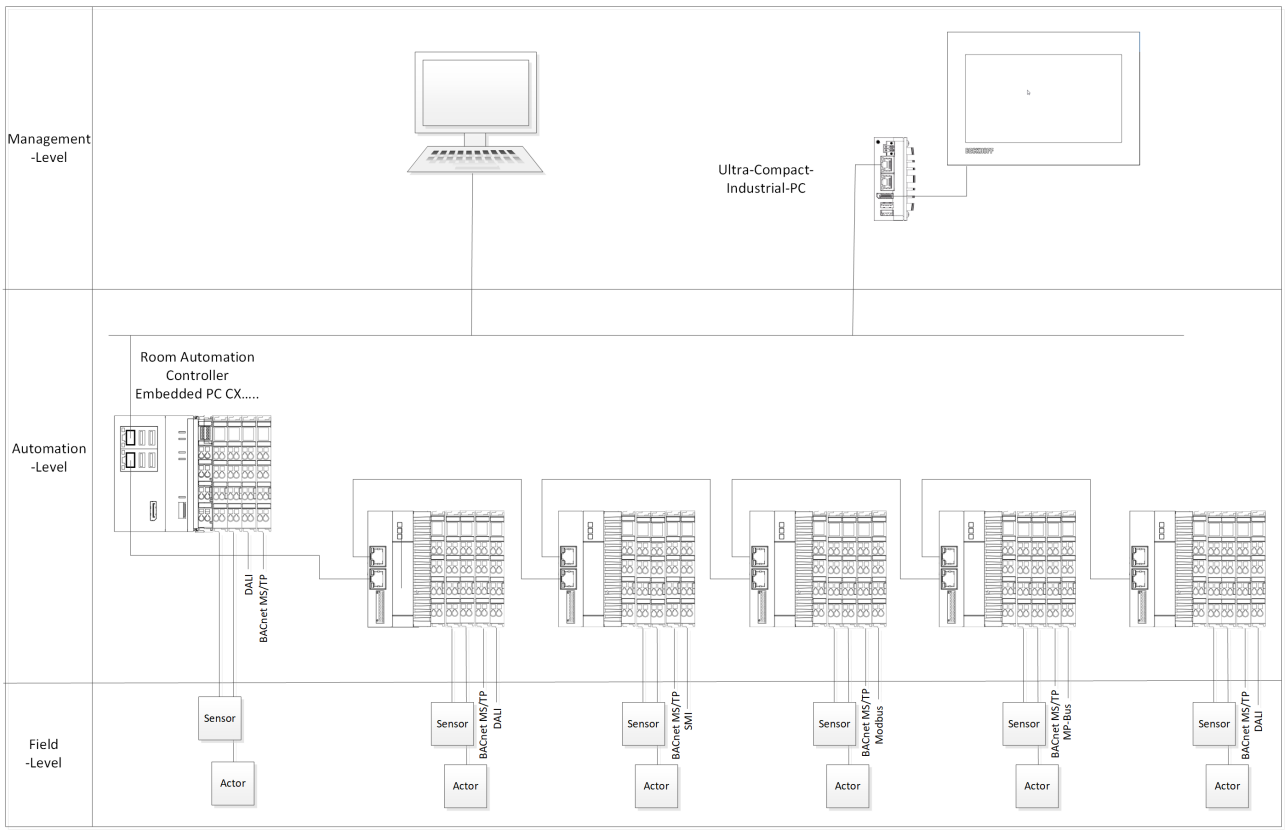
Room automation is integrated into the automation level.

The automation stations for room automation in TwinCAT Building Automation also consist of Beckhoff IPCs.

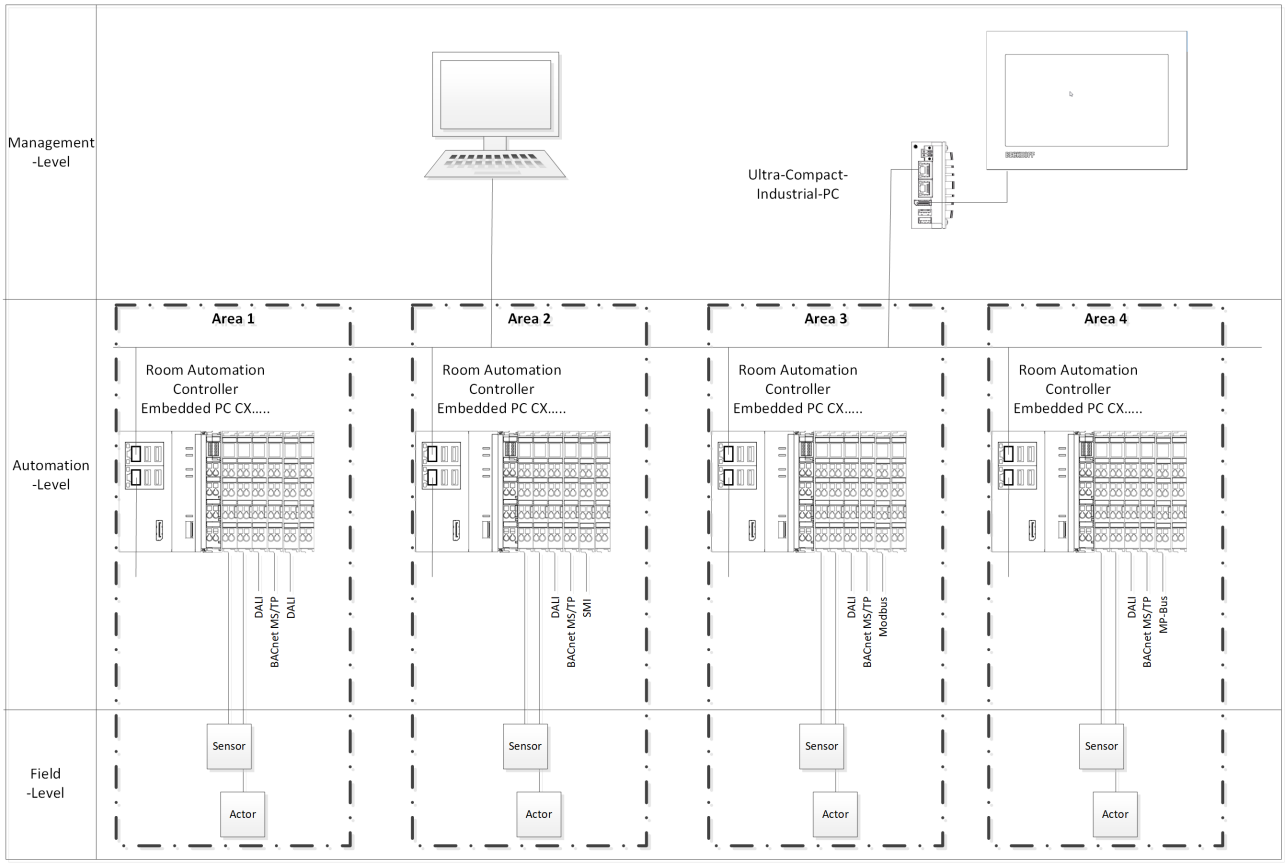
The choice and number of Beckhoff IPCs depends heavily on the building automation requirements.

Due to the extraordinarily high computing power of the IPCs, it is possible to automate very large areas up to an entire floor with a single IPC. Sensors and actuators at field level can be recorded using remote fieldbus couplers. This reduces the cabling effort and also the fire load.

Subsystems such as DALI for illumination can also be integrated into the fieldbus couplers and distributed throughout the floor.



If necessary, a floor can be divided into several areas. The areas are then each controlled by a smaller, separate IPC.



3.5.1 Shell model

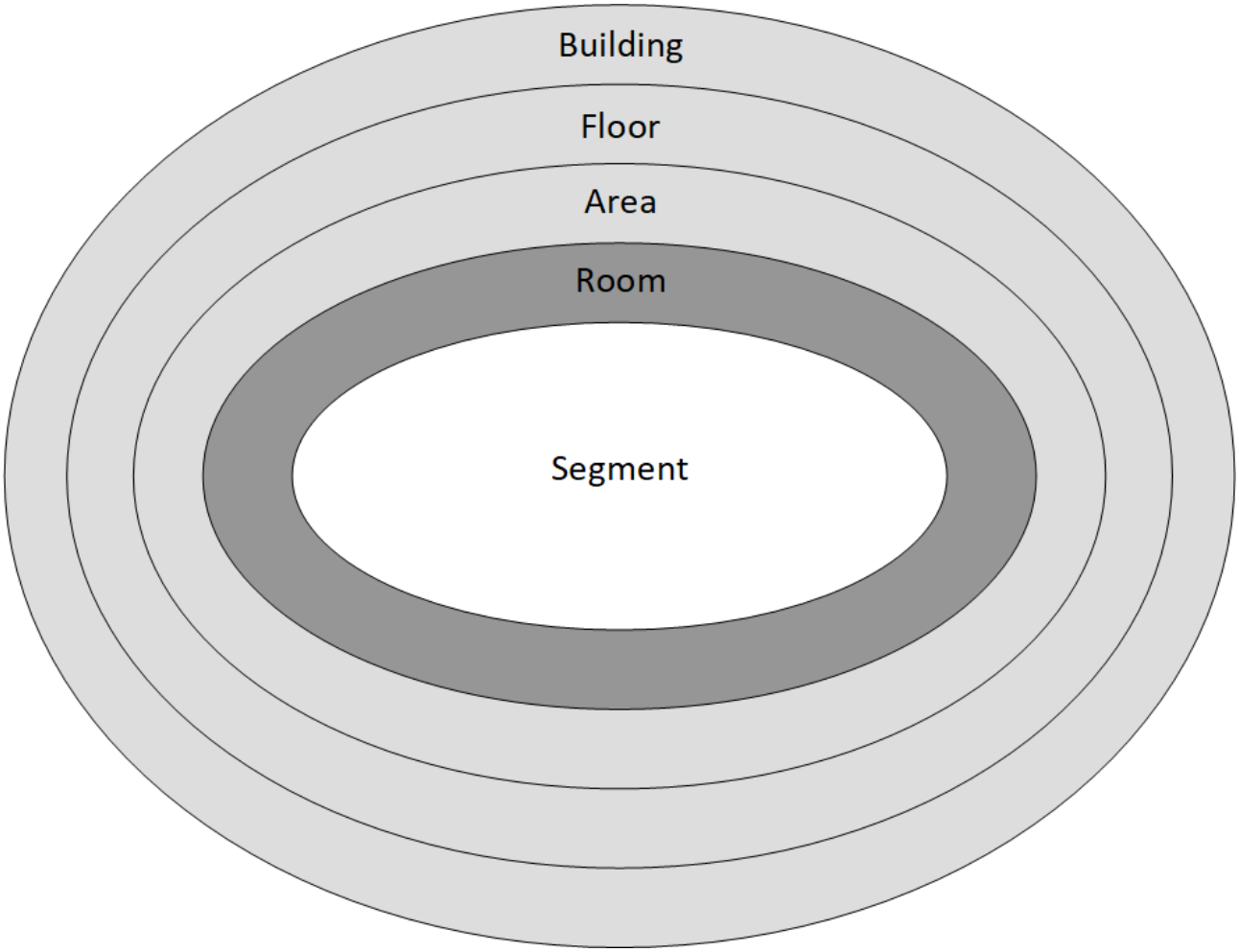
The VDI 3813 shell model is used to explain room automation in TF8040. The shell model describes levels within which the room functions [▶ 23], consisting of sensor, user or actuator functions, are used.

The user and sensor functions of one shell affect the actuators of all enclosed shells. Thus, the position of a user function in the shell model determines the subset of actuators on which this user function acts.

Example:

A schedule for central control of all the lights on a floor, e.g. for cleaning an office, is positioned in the Floor shell.

If the cleaning operation has to be planned separately in the sub-areas of the floor, for example if there are individual tenant areas in which the cleaning operation is carried out separately, a schedule must be placed for each area.



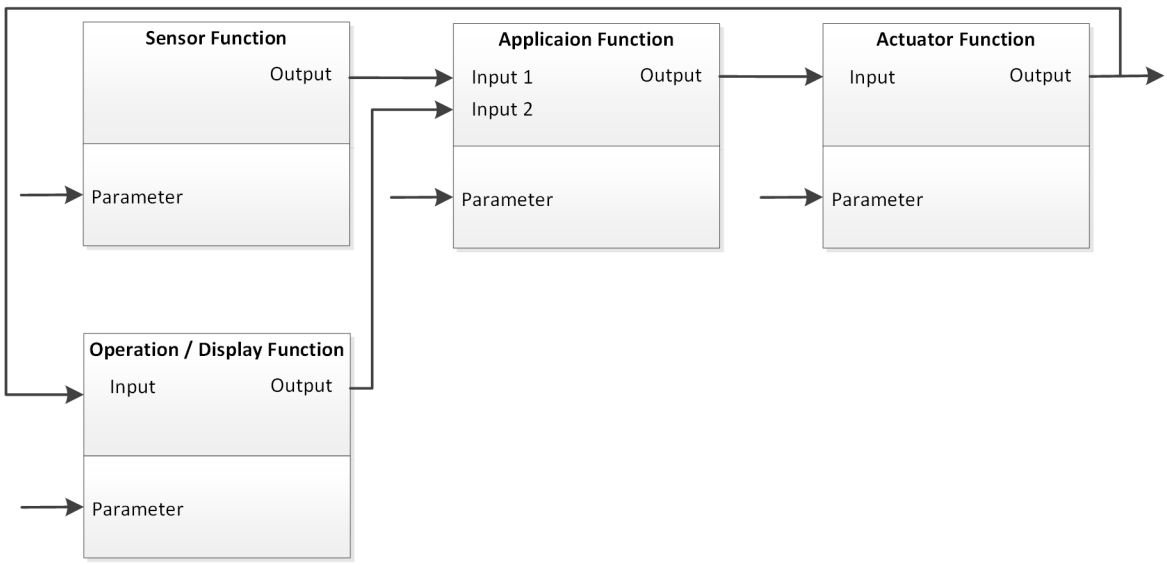
3.5.2 Room functions

The functions of a room are divided into sensor, application and actuator functions.

The signals from the sensors and actuators are either recorded or output directly by a terminal or integrated via a fieldbus system. The interface or communication of the sensor and actuator functions with the application functions of the room is always identical. This allows the room application functions to remain the same regardless of the fieldbus system selected.

When using fieldbus systems for room automation, the measuring signals of the sensors or the control commands of the actuators are integrated by incorporating the corresponding fieldbus terminal.

Room automation functions according to VDI 3813:



The room sensor functions not only serve the application functions of a technical system, but are also processed by the application functions of illumination, sun protection and room air condition control. The presence monitoring of a room, which is used for the automation of the three technical systems mentioned above, is particularly representative of this.

The type of application functions for the technical systems illumination and sun protection depend heavily on the use of the room. These should be determined early in the design process by the building automation designer in conjunction with the building operator.

3.5.2.1 Lighting

The lighting functions within TF8040 are based on the shell model [▶ 23] from the VDI 3813 standard for room automation.

There can be one or more room functions [▶ 23] in each shell of the lighting controller model. A user function of the light sends a telegram to control the lights.

```
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
```

A control value for the light intensity in % and the light temperature in Kelvin is transmitted within the telegram. The telegram also contains the variables *bActv* and *ePrio*.
To decide which of the functions in a shell has priority, each light telegram is given a priority *ePrio*.
A telegram selector of type FB_BA_LightingTgmSel4 [▶ 310] /FB_BA_LightingTgmSel8 [▶ 310] decides which of the telegrams is passed through to the next inner shell if there are several light functions in a shell.

Global lighting functions [▶ 25] with high priorities are positioned in the building, floor and area shells.

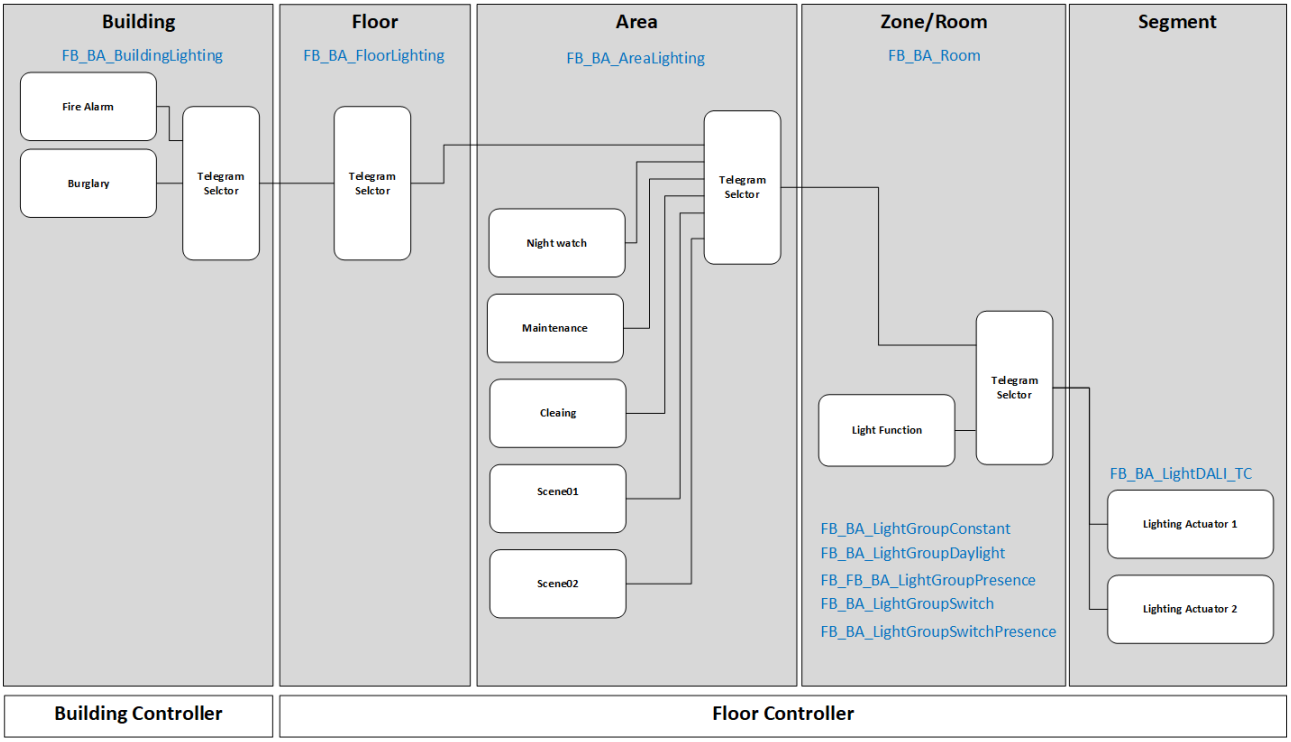
These lighting functions are implemented within the templates FB_BA_BuildingLighting [▶ 917] and FB_BA_FloorLighting [▶ 919].

Global lighting functions [▶ 25]

In the Zone/Room shell (local lighting functions [▶ 26]), the respective application function that meets the room's illumination requirements is selected. If there are several lighting groups in a room, a separate application function is placed for each one.

The sensor and user functions of the building shell are located in the Building Controller automation station. The functions of the shells floor to segment are located in the Floor Controller. Further details on this can be found in the ADS communication chapter within the templates [▶ 18]

The overall functionality of the lighting controllers is determined by the selection and position of the lighting functions in the shell model. The lighting concept is very individual and strongly dependent on the use of the building.



3.5.2.1.1 Global lighting functions

In addition to the functions within the room, luminaires are controlled by higher-level application functions. These must be adapted to the requirements for the operation of the building and determined in the planning phase.

Possible global lighting functions can be:

- **Burglary**
  A connection between the BA system and the burglary alarm system triggers a global command that switches on the illumination throughout the building.
  The implementation of this lighting function can be found in the template FB_BA_BuildingLighting [▶ 917].
- **Fire**
  The illumination is switched on by a connection between the BA system and the fire alarm system.
  The implementation of this lighting function can be found in the template FB_BA_BuildingLighting [▶ 917].
- **Maintenance mode**
  When maintenance mode is triggered via the management and operating device (MBE) or TwinCAT 3 HMI, illumination is switched on in a specific area of the building.
  The implementation of this lighting function can be found in the template FB_BA_FloorLighting [▶ 919].
- **Night watchman tour**.
  The illumination is switched on and dimmed to a certain value during the security staff's tours. Operation can be carried out via a time schedule or manually.
  The implementation of this lighting function is located in the template FB_BA_FloorLighting [▶ 919].

3.5.2.1.2 Local lighting functions

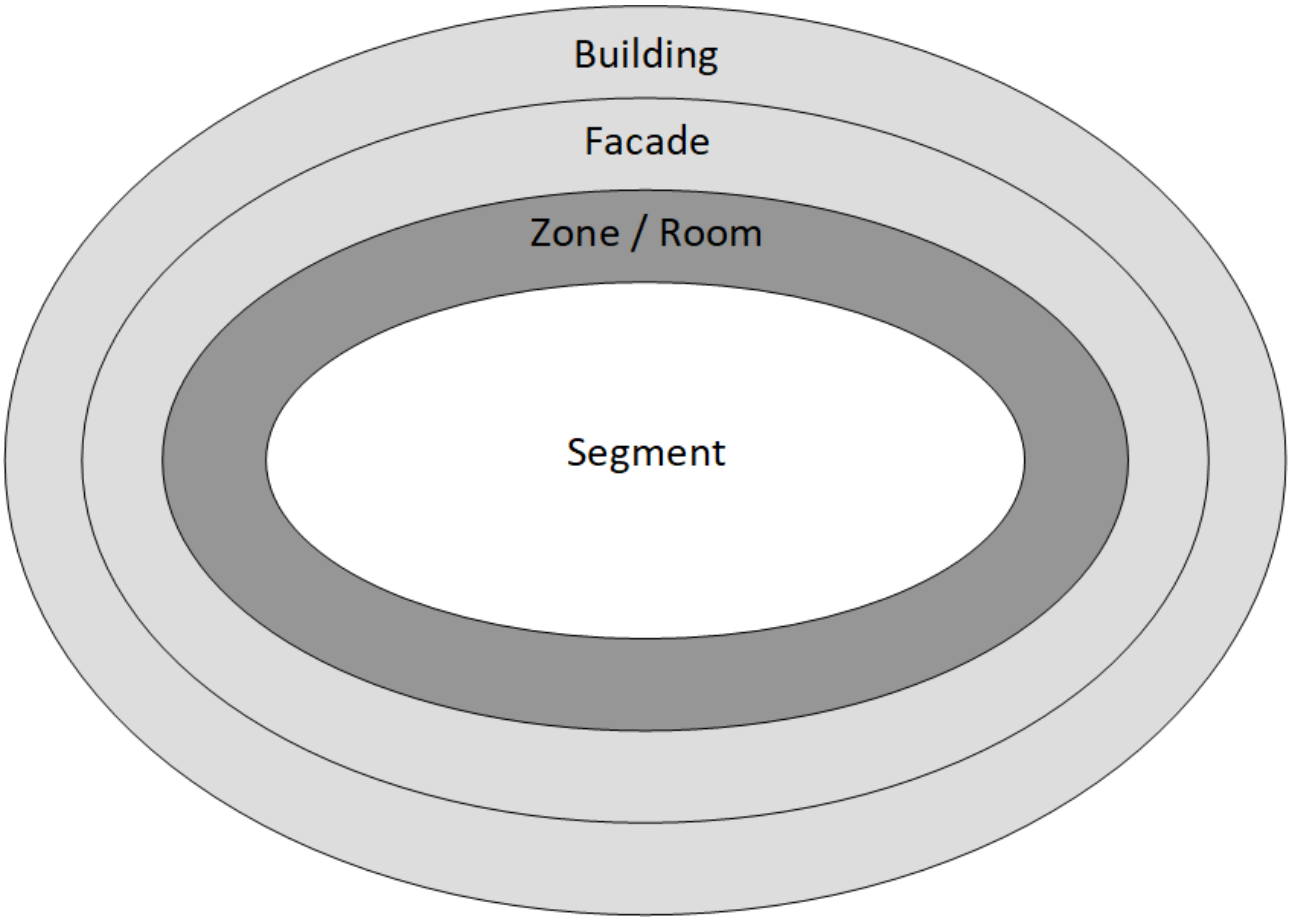
The application functions of the illumination in the room depend very much on the use or type of room.

The following application functions for illumination are available as templates in TwinCAT 3 Building Automation. All application functions relate to one or more luminaires.

- **Lighting group**
  With the lighting group application function, switchable or dimmable lighting devices can be switched on and off or dimmed. The lighting group can be operated via light switches, room control units or via graphical operation using TwinCAT 3 HMI.
  This lighting function is implemented in the template FB_BA_LightGroupSwitch [▶ 903].
- **Automatic light**
  The automatic light application function switches the room lighting on automatically when the room is occupied. Natural illumination by daylight is not taken into account. The light is only switched via presence detection. The function is particularly useful for energy-saving lighting in rooms with insufficient daylight, such as corridors or sanitary rooms.
  This lighting function is implemented in the template FB_BA_LightGroupPresence [▶ 900].
- **Constant light regulation**
  The constant light regulation application function automatically controls the room lighting or parts of it during occupancy so that it does not fall below a set minimum illuminance. This ensures high-contrast work with minimal energy consumption. While the on-delay and off-delay settings can be used to control switching on before the minimum light level is reached or switching off after the minimum light level is reached, a change in the occupancy status results in undelayed switching. Override by a push button or graphical operation via TwinCAT 3 HMI stops the constant light regulation. Instead, the input value of the push button or the TwinCAT 3 HMI is passed on to the luminaires.
  This light function is implemented in the template FB_BA_LightGroupConstant [▶ 894].

3.5.2.2 Sun protection

The sun protection functions are based on the VDI 3813 shell model.



There can be one or more sun protection application functions in each shell of the sun protection model. Each application function generates a positioning telegram for the sun protection actuators.

All positioning telegrams (ST_BA_SunBld [▶ 274]) of a shell are evaluated by a telegram selector. The telegram with the highest priority is forwarded to the inner shell.

```
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
```

A control value for the position of the blind in % and a slat angle in degrees are transmitted within the telegram. The telegram also contains the variables *bActv* and *ePrio*.

Global user functions of the sun protection system operate in the Building, Floor and Area shells.

Global sun protection [▶ 28]

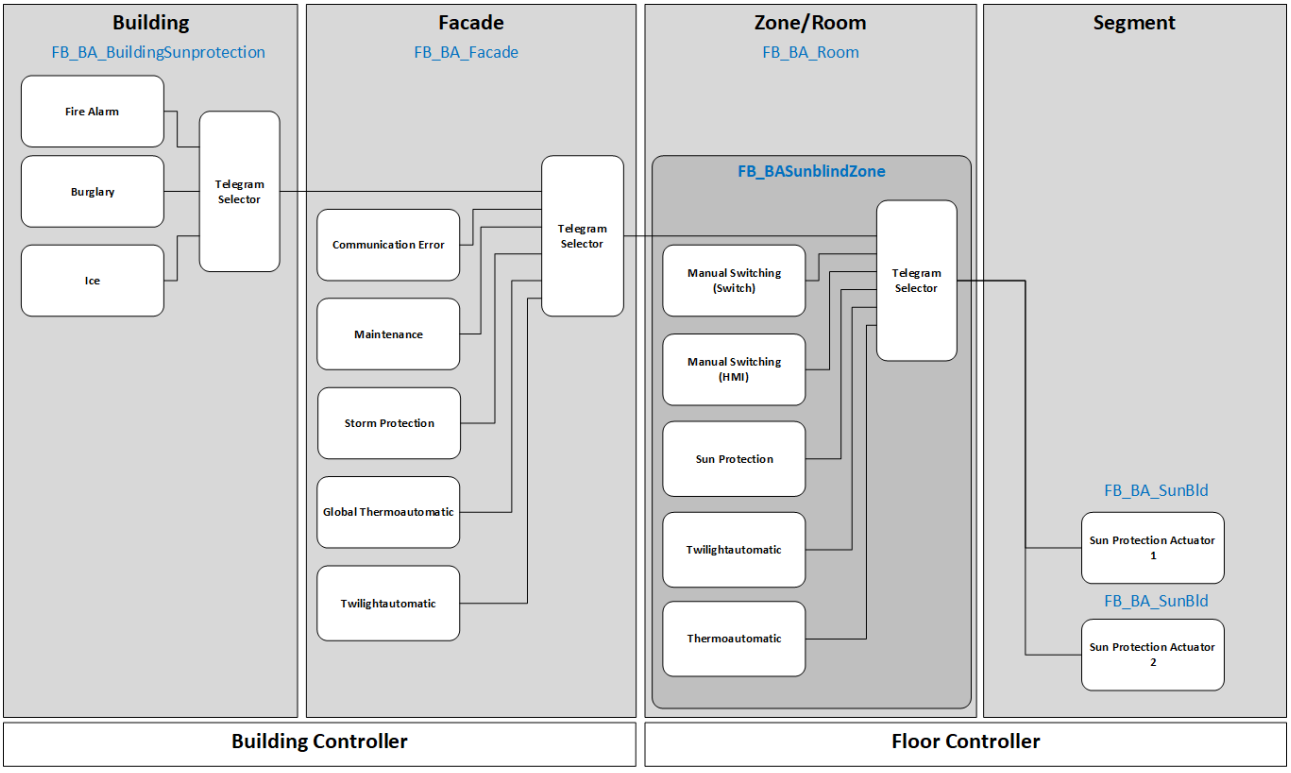
The room user functions work in the Room shell.

Local sun protection [▶ 28]

The overall functionality of the sun protection system is determined by the choice and position of the sun protection functions in the shell model.

The functions of the outer shell for the building affect all the building's sun protection actuators.

The functions of the inner shells only apply to the corresponding subset of the sun protection actuators that are located within the shells they enclose. The sun protection actuators themselves are located in the innermost shell segment.



3.5.2.2.1 Global sun protection

In addition to the local sun protection functions, there are other application functions in the system from which global positioning commands are sent to the sun protection actuators.

The following sun protection functions are located in the building level of the shell model.

The functions are implemented in the template FB_BA_BuildingSunprotection [▶ 937].

- **Fire**
  In the event of a fire, all the blinds in a building are raised.
- **Storm protection**
  The storm protection function prevents external sun protection devices from being damaged by wind.
- **Icing protection**
  The risk of the blinds icing up is predicted by combining the measured values for outside temperature and precipitation. To protect the blinds from mechanical damage, they move up when there is a risk of icing and remain there until a predicted icing time has elapsed.

There is one instance of the template FB_BA_Facade [▶ 940] for each facade of the building.

The template contains functions and calculations related to a facade.

- **Maintenance**
  To check that all the blinds on a facade are fully functional and correctly positioned, it is possible to raise and lower all the blinds on a facade synchronously using the maintenance function. In the case of facade cleaning, the blinds of a facade can all be raised and blocked in this position.
- **Global thermal automatic**
  The global thermal automatic has the same function as the local thermal automatic within the rooms. However, it controls all the blinds on a facade in the same way, so that a uniform image of all the blinds on a facade is created during longer periods of absence, e.g. at weekends. The global thermal automatic uses the measured room temperature value of a reference room.
- **Communication error**
  In the event of communication problems within the BA network, it is not ensured that the positioning telegrams are passed through to the sun protection actuators with a high priority. The blinds are then raised for safety reasons.

3.5.2.2.2 Local sun protection

The application functions can vary depending on the use and technical equipment of a room.

In the TF8040 sun protection templates, one or more groups of external venetian blinds or blinds are assumed.

The templates of the sun protection functions always refer to a group of blinds which are controlled in parallel. Function blocks for the integration of roller shutters are also available.

All room-related sun protection functions are located in the template FB_BA_SunblindZone [▶ 964].

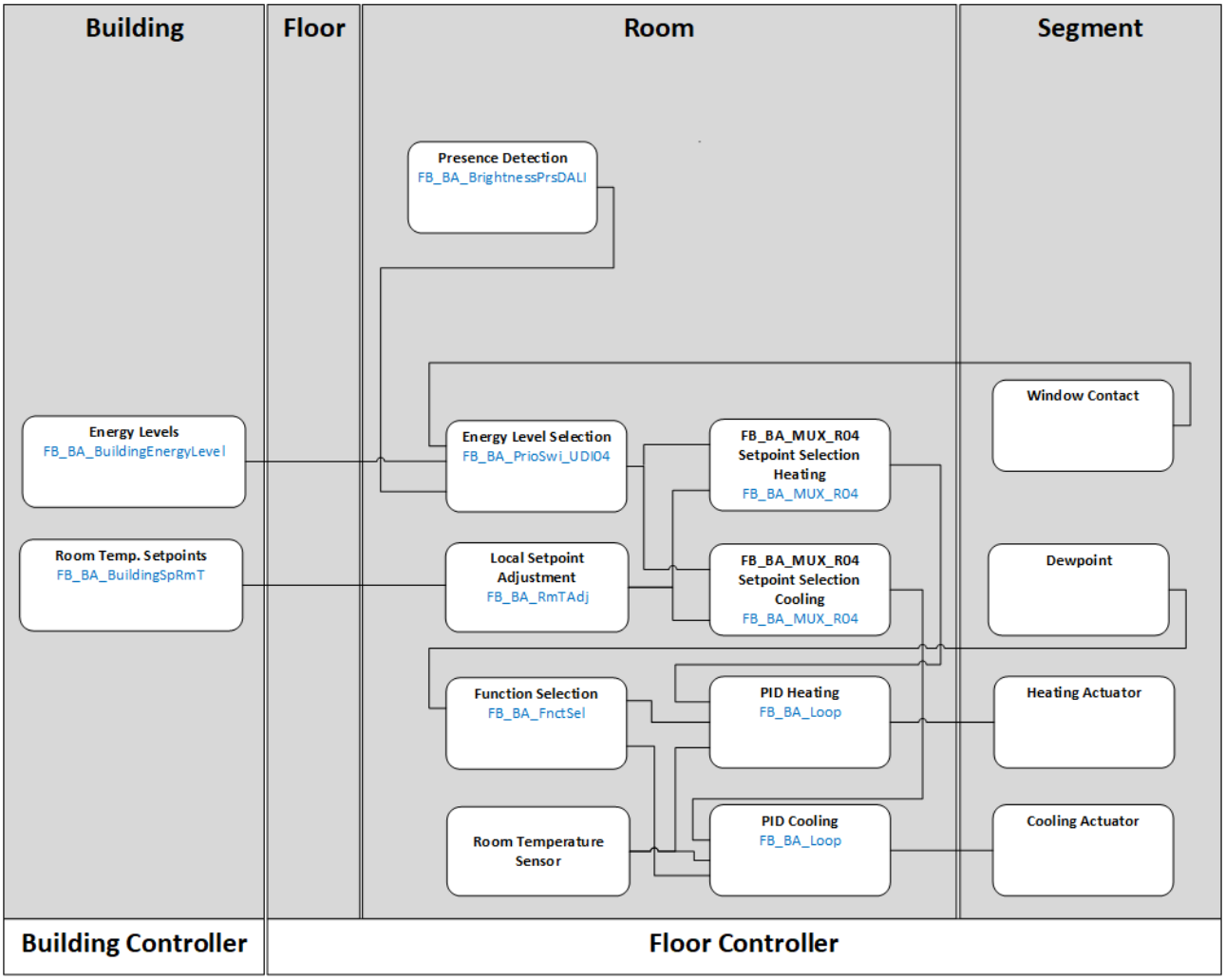
The room functions for a blind group are:

- **Sun protection with lamella setpoint tracing**
  Sun protection primarily serves as glare protection. The position of the slats is cyclically adjusted to the current position of the sun. As a result, every room is supplied with the best possible daylight despite the prevention of direct sunlight. The energy consumption for the artificial illumination of a room is kept as low as possible. The sun protection also prevents the room from overheating due to solar radiation, thus reducing the energy required for cooling. The sun protection is activated when someone is present in the room.
- **Thermal automatic**
  With the help of thermal automatic, the sun protection is used in unoccupied rooms to support heating or cooling by selectively allowing or blocking solar heat input. This prevents overheating in summer and reduces the load on the heating system in winter. The thermal automatic processes the room temperature and room temperature setpoint sensor function.

- **Twilight automatic**
  The twilight automatic function can be used to position sun protection devices depending on the outdoor brightness. This function allows, for example, the sun protection to be closed during the night, e.g. to reduce cooling through the windows or to reduce the building's light emissions and prevent unwanted views into the building from outside.
- **Manual control**
  The automatic functions of the sun protection system can be manually overridden using a push button, a room control unit that can be integrated via fieldbus, or a graphical control panel via TwinCAT 3 HMI.

3.5.2.3 Air conditioning

The room air conditioning functions are based on the shell model from the VDI 3813 standard for room automation.



There is a timer program in the Building shell. This timer program describes the periods during which the rooms in the building are in *Protection* mode, *Economy* mode, *Pre-Comfort* or *Comfort* mode.

The operation modes of the buildings are also referred to as energy levels.

The longer the building is unoccupied, the further the energy level can be reduced.

If the building is not used for a very long time, it can be put into Protection mode.

In winter, the room temperature is reduced or increased to a frost protection setpoint and in summer to an overheating protection setpoint.

For longer regular periods of absence, e.g. at weekends or during the night, the *Economy* energy level is planned.

With the *Pre-Comfort* level, the setpoints are raised to such an extent that they can be reached for *Comfort* mode in the short term.

The setpoints for the *Comfort* level can be changed slightly if necessary using a local setpoint adjuster in the room.

The central generation and distribution of the room temperature setpoints for the entire building are contained in the building controller, see sketch ADS communication within the templates [▶ 18].

3.6 Base framework

The base framework in TwinCAT 3 Building Automation is an essential property and basis of TF8040. The base framework contains functions that run in the background of a TF8040 solution in the PLC. The functions of the base framework are part of the Tc3_XBA [▶ 99] library.

The base framework is very useful when implementing a BA project (building automation project). This chapter explains these functions.

Creation of the project structure (PS)

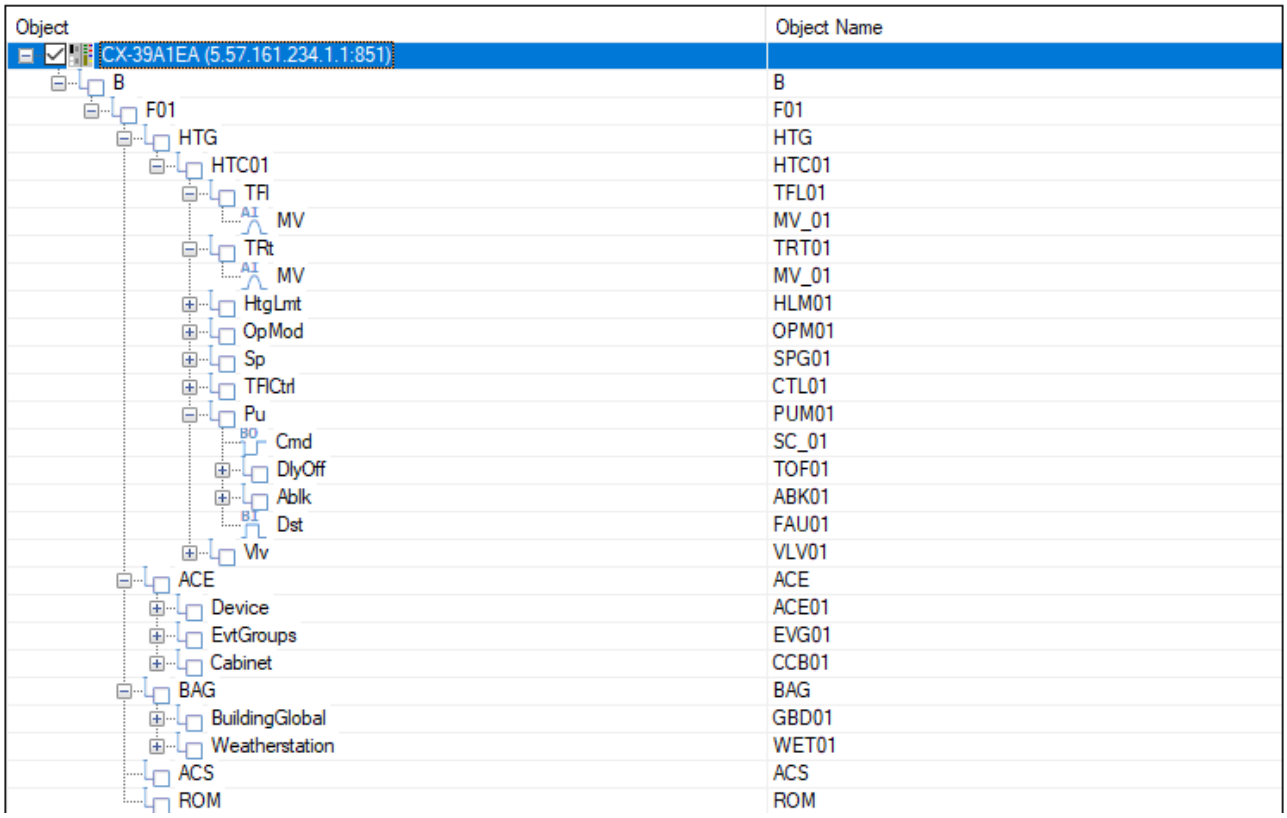
A building automation project is usually organized in levels. The levels start with the property or the location, for example. Further levels can be the building, the systems it contains, the associated aggregates and, finally, the data points it contains.

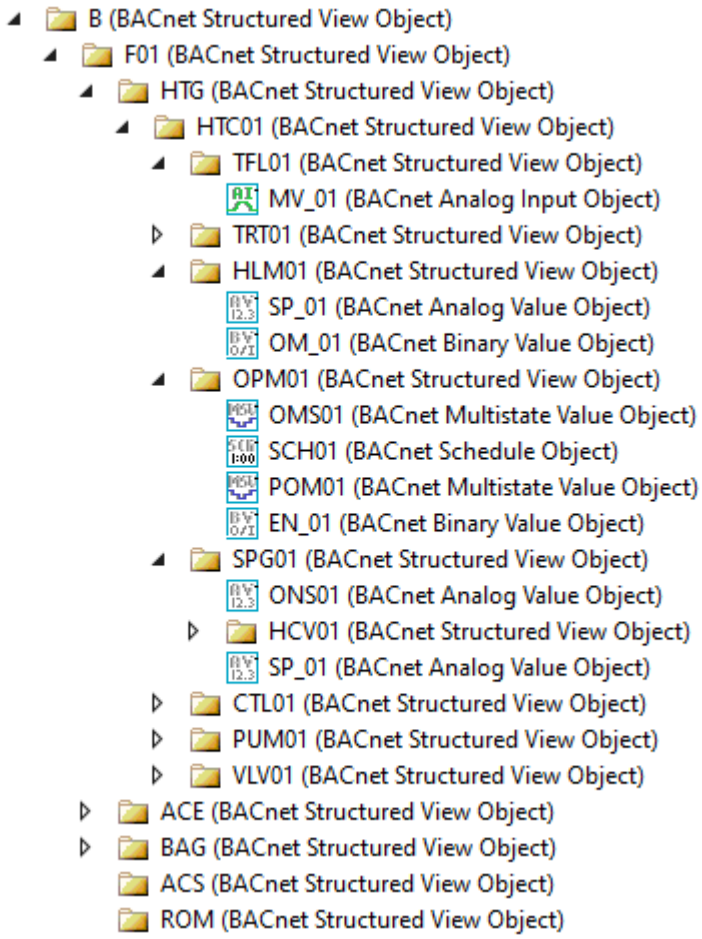
The most important task of the base framework is to map the structure of a project in TwinCAT. The project structure is a system consisting of folders and the objects they contain.

For large projects with a large number of objects, this project structure is an important basis for designing a project sustainably and clearly.

The image shows a project structure created using the base framework in the Site Explorer [▶ 1285].

The project structure is mapped in the Site Explorer tool, in the BACnet configuration and in a generic navigation of the TwinCAT HMI.





The benefits of the base framework, which generates the project structure in a clear form in the HMI, the BACnet and also in the BACnet MBE, are very great.

Everything from commissioning, parameterization, operation via HMI, subsequent maintenance and connection to a BACnet MBE is facilitated by the project structure.

The creation of a project structure is described in the chapter Project structure [▶ 38].

DPAD(user address key)

Another significant advantage is the processing of a user address key (BAS).

All levels of the project structure described above are given a name and a descriptive text.

The base framework of TF8040 can concatenate the object names and the description texts of the levels and generate the names and descriptions of the BACnet objects from them. In TwinCAT 3 Building Automation, the BAS is also referred to as DPAD Datapoint Addressing Description. Detailed information can be found in the chapter DPAD [▶ 42].

Events

The base framework records the events of the objects within the project structure.

A collection of events including all sub-elements is created at each level of the project structure.

A function block called FB_BA_PlantLock [▶ 135] can be used for plant shutdowns without having to program a collective error message.

The acknowledgement and resetting of events is also organized by the base framework. A reset or acknowledgement command is automatically passed through to the objects from the top level of the project structure down to the lowest level of the project structure. The processing of events is described in more detail in the Events [▶ 32] documentation.

Counting events

Each folder or view object is able to collect status information such as OutOfService, InAlarm, Overridden etc. from all subordinate objects and output their number. The EventConditionCount method counts the pending events of objects at the respective level of the project structure.

3.6.1 Objects

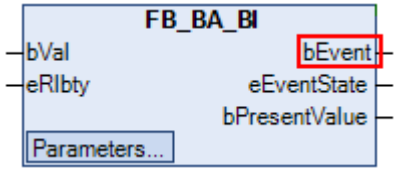
Objects are the elementary components of the base framework. An object offers various properties. All objects in TwinCAT 3 Building Automation refer to the objects of the TwinCAT Function TF8020 BACnet.

3.6.1.1 Events

The base framework of TwinCAT 3 Building Automation and the objects it contains offer extensive functions for processing events.

In TF8040, an event always refers to an object. Events occur when an object assumes an abnormal or faulty state. Event-capable objects in TwinCAT 3 Building Automation have the *bEvent* output for further processing of the event within the TwinCAT program.

Example:FB_BA_AI



The state of an object is described with the *EventState*.

Possible event states are:

State	Description
eNormal	The state of the object is normal.
eFault	The state of the object is faulty.
eOffnormal	The state of the object is abnormal.
eLowLimit	The upper limit value of an analog object has been exceeded.
eHighLimit	The value has fallen below the lower limit of an analog object.

Display of events

Events are displayed in the event list of the Site Explorer [▶ 1285] and the TcHmiBa [▶ 1120]. The events are also transmitted to BACnet clients via the BACnet server if required.

The display of an event depends on the following properties:

- 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):

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

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

The following representations result for each event type:

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

Event controllers

Critical events often require a control response, such as shutting down a ventilation system after a fire damper fails.

The desired control functionality is parameterized with the lock functionalities of the event-enabled objects.

For this purpose, the events within the levels in the project structure [▶ 30] are summarized and evaluated using the function block FB_BA_PlantLock [▶ 135].

Parameterizing events

An event can have different requirements with regard to its display, its control processing and its acknowledgement and resetting. Most of these properties are not parameterized on the object itself, but with the event class FB_BA_EC [▶ 218] assigned to the object.

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):

With the function block FB_BA_EventObserver [▶ 134] it is possible to carry out a collective acknowledgement or reset of all events within the project structure. Which objects are acknowledged or reset depends on the position of the FB_BA_EventObserver [▶ 134] in the project structure. In general, all objects that are located in the same folder or a subfolder in the project structure are acknowledged or reset.

- **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 extended alarm mode, an event (or an object) must not only be acknowledged, but also reset in order to restore an already passed 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 disabling events that, for example, have the desired effect on the FB_BA_PlantLock [▶ 135].

- **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.

3.6.1.2 Command

Several control or positioning commands of an object to be commanded have an influence on the output value *PresentValue*. All commands are stored in an array. The command with the highest priority determines the result at the output of the object.

A priority can be given a value between 1 and 16. The highest priority has the value 1.

The following objects in TwinCAT 3 Building Automation are objects to be commanded and thus have a priority array.

FB	Type	Description
FB_BA_BO_Raw	BO	Binary output with external declaration of variables for hardware mapping.
FB_BA_BO	BO	Binary.
FB_BA_BO_IO	BO	Binary output with internal declaration of variables for hardware mapping.
FB_BA_AO_Raw	AO	Analog output with external declaration of variables for hardware mapping.
FB_BA_AO	AO	Analog.
FB_BA_AO_IO	AO	Analog output with internal declaration of variables for hardware mapping.
FB_BA_MO_Raw	MO	Multi State output with external declaration of the variables for hardware mapping.
FB_BA_MO	MO	Multi State output.
FB_BA_MO_IO	MO	Multi State output with internal declaration of variables for hardware mapping.
FB_BA_BV	BV	Binary Value object.
FB_BA_MV	MV	Multistate Value object.
FB_BA_AV	AV	Analog Value object.

For an entry to be made in the priority array of an object, the associated enable input *bEn*... must be TRUE on the function block.

The Manual Remote priority is not activated by means of an input at the function block, but by writing to a parameter variable. The object is commanded either via the BACnet protocol, e.g. from an MBE, or from TwinCAT via ADS.

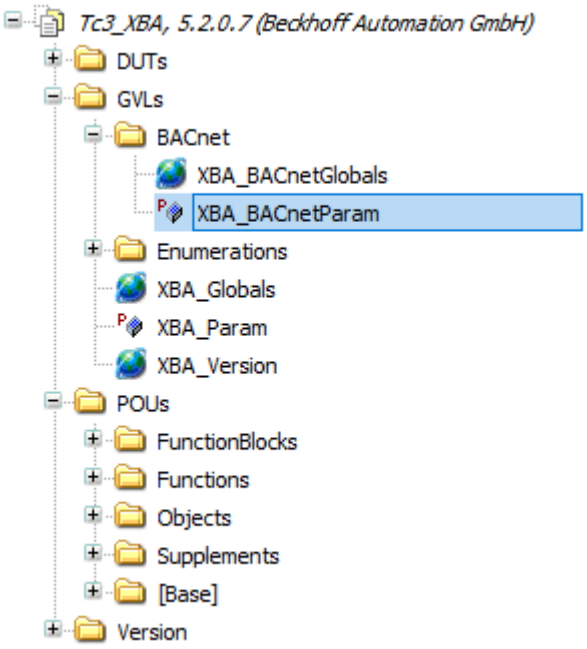
The following priorities are predefined at the objects by TwinCAT Building Automation:

Name	Description	Symbol release	Symbol value	Default Prio
Safety	Personal safety	bEnSafety	*ValSfty	1
Critical	Plant safety	bEnCrit	*ValCritical	3
ManLoc	Local manual override (LVB)	bEnManLoc	*ValManLoc	7
ManualRm	Manual override from a distance (MBE)	bEnManualRm	*ValManualRm	8
Pgm	Program control	bEnPgm	*ValPgm	15

Depending on whether the object type is Analog, Binary or Multi State, the value of a command is a REAL, BOOL or UDINT.

Changing priorities

The priorities can be changed in the variable list BA_BACnet_Param.


Tc3_XBA, 5.2.0.7 (Beckhoff Automation GmbH)
DUTs
GVLs
BACnet
XBA_BACnetGlobals
XBA_BACnetParam
Enumerations
XBA_Globals
XBA_Param
XBA_Version
POUs
FunctionBlocks
Functions
Objects
Supplements
[Base]
Version

```
{attribute 'qualified_only'}
// Supplement translation:
VAR_GLOBAL CONSTANT
    {region 'Objects'}
        {region 'Local'}
            {region 'EventConfig'}
                sEventMessageTextFormat             : STRING                                := '{Descr} - {EvtTrans}';
            {endregion}
        {endregion}
        {region 'Remote'}
            {region 'Analog Output'}
                fRM_AO_WriteIncrement               : REAL                                  := 0.0;
            {endregion}
            {region 'StructuredView'}
                eView_SubordinateAnnotationMode     : E_BACnet_AnnotationTitle              := E_BACnet_AnnotationTitle.eSymbolName;
            {endregion}
        {endregion}
    {endregion}
    {region 'Priorities'}
        aPriority                                   : ARRAY[E_BA_Priority.First .. E_BA_Priority.Last] OF E_BACnet_Priority := [
                                                        (*  eProgram                    *)  E_BACnet_Priority.eP15,
                                                        (*  eManualRemote               *)  E_BACnet_Priority.eP8,
                                                        (*  eManualLocal                *)  E_BACnet_Priority.eP7,
                                                        (*  eCritical                   *)  E_BACnet_Priority.eP3,
                                                        (*  eLifeSafety                 *)  E_BACnet_Priority.eP1
                                                      ];
    {endregion}
    {region 'Translation'}
        aNodeType                                   : ARRAY[E_BA_NodeType.First .. E_BA_NodeType.Last] OF E_BACnet_NodeType := [
                                                        (*  eUnknown                    *)  E_BACnet_NodeType.eUnknown,
                                                        (*  eOther                      *)  E_BACnet_NodeType.eOther,
                                                        (*  eGeneral                    *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eLocation                   *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eBuilding                   *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eBuildingElement            *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eInformationFocus           *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eControlCabinet             *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eTrade                      *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eFloor                      *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eRoom                       *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  ePlant                      *)  E_BACnet_NodeType.eOrganizational,
                                                        (*  eAggregate                  *)  E_BACnet_NodeType.eEquipment,
                                                        (*  eFunction                   *)  E_BACnet_NodeType.eFunctional
                                                      ];
        aNotifyType                                 : ARRAY[E_BA_EventType.First .. E_BA_EventType.Last] OF E_BACnet_NotifyType := [
                                                        (*  eAlarm                      *)  E_BACnet_NotifyType.eAlarm,
                                                        (*  eDisturb                    *)  E_BACnet_NotifyType.eAlarm,
                                                        (*  eMaintenance                *)  E_BACnet_NotifyType.eNotifyEvent,
                                                        (*  eNotification               *)  E_BACnet_NotifyType.eNotifyEvent,
                                                        (*  eOther                      *)  E_BACnet_NotifyType.eNotifyEvent
                                                      ];
        aEventState                                 : ARRAY[E_BA_EventState.First .. E_BA_EventState.Last] OF E_BACnet_EventState := [
                                                        (*  eNormal                     *)  E_BACnet_EventState.eNormal,
                                                        (*  eFault                      *)  E_BACnet_EventState.eFault,
                                                        (*  eOffnormal                  *)  E_BACnet_EventState.eOffnormal,
                                                        (*  eLowLimit                   *)  E_BACnet_EventState.eLowLimit,
                                                        (*  eHighLimit                  *)  E_BACnet_EventState.eHighLimit
                                                      ];
    {endregion}
END_VAR
```

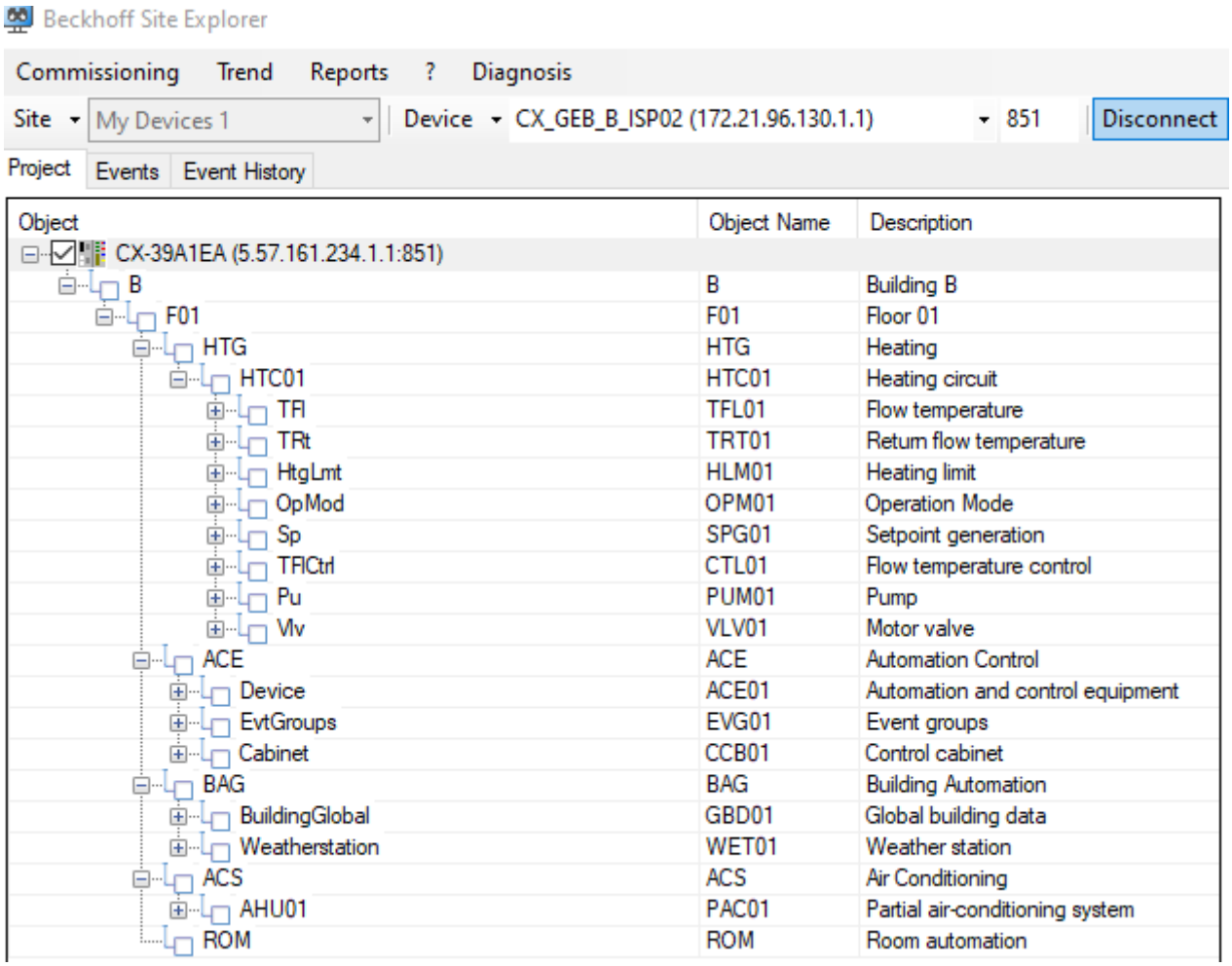
The active priority is displayed at the output of all objects to be commanded by means of the variable *eActivePrio*.

Variables

Name	Type	Description
bEnSafety	BOOL	Enabling the "Safety" priority.
fValSafety	REAL	Analog value for the "Safety" priority.
bValSafety	BOOL	Binary value for the "Safety" priority.
nValSafety	INT	Integer value for the "Safety" priority.
bEnCritical	BOOL	Enabling the "Critical" priority.
fValCritical	REAL	Analog value for the "Critical" priority.
bValCritical	BOOL	Binary value for the "Critical" priority.
nManCritical	INT	Integer value for the "Critical" priority.
bEnManLocal	BOOL	Enabling the "Manual Local" priority.
fValManLocal	REAL	Analog value for the "Manual Local" priority.
bValManLocal	BOOL	Binary value for the "Manual Local" priority.
nManLocal	INT	Integer value for the "Manual Local" priority.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.
fValManualRm	REAL	Analog value for the "Manual Remote" priority.
bValManualRm	BOOL	Binary value for the "Manual Remote" priority.
nManualRm	INT	Integer value for the "Manual Remote" priority.
bEnPgm	BOOL	Enabling the "Program" priority.
fValPgm	REAL	Analog value for the "Program" priority.
bValPgm	BOOL	Binary value for the "Program" priority.
nPgm	INT	Integer value for the "Program" priority.

3.6.2 Project structure

This section describes how to create a project structure, as shown in the following illustration.



Initialization of the project structure begins in the MAIN program of the automation station.

The first step is to determine how many levels are to be mapped in the project.

The project structure is not only used to organize a project, but also to process a BAS (user address key).

The levels of the project structure are therefore described with the initialization of the function block FB_BA_CarelessDPAD.

The function block is called in the MAIN program of the automation station. A more detailed description of BAS processing can be found in the chapter DPAD [▶ 42].

The graphics in this description refer to the standard TF8040 PLC.

The building, floor, technical systems, plant, aggregate and function levels are generated within the sample PLC.

```
PROGRAM MAIN
VAR
DPAD        : FB_BA_CarelessDPAD := (
                    aIdentifier := [
    (* Level 1 *)       (   eNodeType:=E_BA_NodeType.eBuilding,        sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 2 *)       (   eNodeType:=E_BA_NodeType.eFloor,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 3 *)       (   eNodeType:=E_BA_NodeType.eTrade,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 4 *)       (   eNodeType:=E_BA_NodeType.ePlant,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2),
    (* Level 5 *)       (   eNodeType:=E_BA_NodeType.eAggregate,       sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2),
    (* Level 6 *)       (   eNodeType:=E_BA_NodeType.eFunction,        sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2)
                    ]
                );
```

The initialization of the project structure (PS) is started in MAIN. A level within the project structure is represented by a folder. A folder is created by an instance of the function block FB_BA_View [▶ 241].

There may be further subfolders or objects within a folder. Objects are created using the function blocks from the chapter Objects [▶ 191] of the Tc3_XBA [▶ 99] library.

The project structure is edited by creating and concatenating folders. The concatenation is done by assigning a parent to each folder within the FB_BA_View [▶ 241].

The parent is assigned by transferring the symbol name of the next higher FB_BA_View [▶ 241] in the project structure to the variable *iParent*.

```
//  Level 1 Gebäude
//  Level 1 Building
        B           : FB_BA_View := (
                        sObjectName     := 'B',
                        sDescription    := 'Building B',
                        eDPADMode       := E_BA_DPADMode.eInclude
                    );

// Level 2 Stockwerk    ->  O01 = Obergeschoss 01, U01 = Untergeschoss 01,  E00 = Ergeschoss
// Level 2 floor -> F01 = floor 01, B01 = basement 01,  GFL = Ground floor
            F01         : FB_BA_View := (
                            iParent         := B,
                            sObjectName     := '{F01}',
                            sDescription    := '{Floor 01}',
                            eDPADMode       := E_BA_DPADMode.eInclude
                        );
// Level 3 Gewerk Automationsschwerpunkt
// Level 3 Trade AutomationFocalPoint
                ACE         : FB_BA_AutomationControl := (
                                iParent         := F01,
                                sObjectName     := '{ACE}',
                                sDescription    := '{Automation Control}',
                                eDPADMode       := E_BA_DPADMode.eInclude
                            );
// Level 3 Gewerk Gebäudeautomation Allgemein
// Level 3 Trade General
                BAG         : FB_BA_BuildingAutomation := (
                                iParent         := F01,
                                sObjectName     := '{BAG}',
                                sDescription    := '{Building Automation}',
                                eDPADMode       := E_BA_DPADMode.eInclude
                            );

// Level 3 Gewerk Luftechnische Anlagen
// Level 3 Trade ventilation system
                ACS         : FB_BA_AirConditioning := (
                                iParent         := F01,
                                sObjectName     := '{ACS}',
                                sDescription    := '{Air Conditioning}',
                                eDPADMode       := E_BA_DPADMode.eInclude
                            );
// Level 3 Gewerk Wärmeversorgungsanlagen
// Level 3 Trade Heat supply systems, heat distribution  -->    HD = heat distribution, HG = heat generation
                HTG         : FB_BA_Heating := (
                                iParent         := F01,
                                sObjectName     := '{HTG}',
                                sDescription    := '{Heating}',
                                eDPADMode       := E_BA_DPADMode.eInclude
                            );
// Level 3 Gewerk Raumautomation
// Level 3 Trade room automation
                ROM         : FB_BA_RoomAutomation := (
                                iParent         := F01,
                                sObjectName     := '{ROM}',
                                sDescription    := '{Room automation}',
                                eDPADMode       := E_BA_DPADMode.eInclude
                            );
```

The sample shows that the levels Building to Trade are created in the MAIN program.

There are the trades Heat supply systems, Automation, General, Ventilation system and Room automation.

The parent of all trades is F01. This creates the five subfolders for the trades below the F01 floor folder.

One method of creating a level of the project structure is to call a FB_BA_View [▶ 241] directly in the Main program.

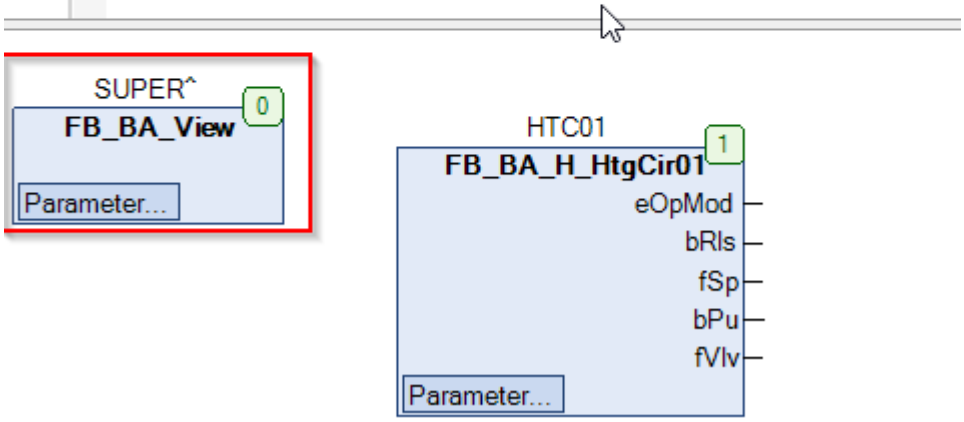
In our sample, this is implemented with the Building and Floor levels.

In the function blocks for the trades Heat supply systems, Automation, General, Ventilation systems and Room automation, the folders are created by calling a function block, which also represents a folder.

However, these folders are created because the function blocks ACE, BAG, ACS, HTG and Rome all inherit from the function block FB_BA_View [▶ 241].
This is done with the EXTENDS operator.

Here is a sample of the function block Heating supply systems.

```
9   FUNCTION_BLOCK FB_BA_Heating EXTENDS FB_BA_View
10  VAR_INPUT CONSTANT
11  HTC01               : FB_BA_H_HtgCir01;
12  END_VAR
13  VAR_INPUT
14  END_VAR
15  VAR_OUTPUT
16  END_VAR
17  VAR
18  END_VAR
```



The function block FB_BA_Heating is a folder and at the same time an FB within which all plants of this trade, e.g. heating circuits, can be called.

This mechanism of creating folders by inheriting from FB_BA_View [▶ 241] continues down to the deepest structures of the project structure.

In the case of heating circuit HTC01, which is called within FB HTG, the parent of the function block is the one in which HTC01 itself is called.

In this case, the parent of the heating circuit is initialized with the THIS^ operator.

```
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
```

```
1  HTC01.iParent                := THIS^;
2  HTC01.iLabel                 := LblPlt_Heating_circuit;
3  HTC01.sObjectName            := '{Idx=01}';
4  HTC01.eDPADMode              := E_BA_DPADMode.eInclude;
```

This rule continues through to the heating circuit aggregates and the objects they contain.

Please note the following when creating the project structure:

- The concatenation of all folders and objects of the PS must be closed.
- All mechanisms of the base framework will not work otherwise!
- All function blocks that map a folder within the project structure and in turn call function blocks from sublevels must inherit from FB_BA_View [▶ 241].
- The FB_BA_View [▶ 241] must be called with the Super^ operator at the beginning of the function block.
- The execution order in the CFC must be observed.
- All function blocks whose parent is the function block within which it is called must receive the parameter THIS^ as parent.
- All function blocks from the template Repository with parameters must be called within the declaration area of VAR_INPUT_CONSTANT.

3.6.3 DPAD

All objects within the TF8040 project structure are given a name and a description text. A common practice in building automation is to name the objects according to a user address key (BAS). Depending on the complexity of the project-specific BAS, TF8040 offers various options for processing the BAS. The BAS is referred to as DPAD (Data Point Addressing Description) in TwinCAT 3 Building Automation.

Generic generation

Generic generation [▶ 43] of ObjectName and Description with the DPAD (Data Point Addressing Description).

This method is suitable for a project in which the BAS can be generated from a concatenation of the levels of the project structure.

This requires that the order of the levels in the project structure matches the order of the BAS text concatenations.

Individual texts for e.g. circuit diagram pages, device identification from the circuit diagram etc. can only be processed with greater effort due to the generic approach of DPAD.

With the DPAD mechanism, it should be noted that the initial texts of the objects are only transferred to the persistent data of the controller in the first cycle of the controller. The initial data is only reloaded after an overall reset of the controller.

As the concatenation mechanism accesses the persistent data, changes do not take effect even after the texts have been concatenated again.

Editing with the Symbol Explorer

The Symbol Explorer [▶ 42] is an excellent tool for editing data.

All control parameters, including texts, can be conveniently edited using special lists. It is possible to export the data to a csv file, edit it and then load it back into the controller.

3.6.3.1 Editing with the Symbol Explorer

In some projects, there are specifications for the user address key that cannot be optimally realized with the DPAD algorithm.

As an alternative to the DPAD algorithm, the name and description texts of the objects can also be edited using the BaExtension of the Symbol Explorer.

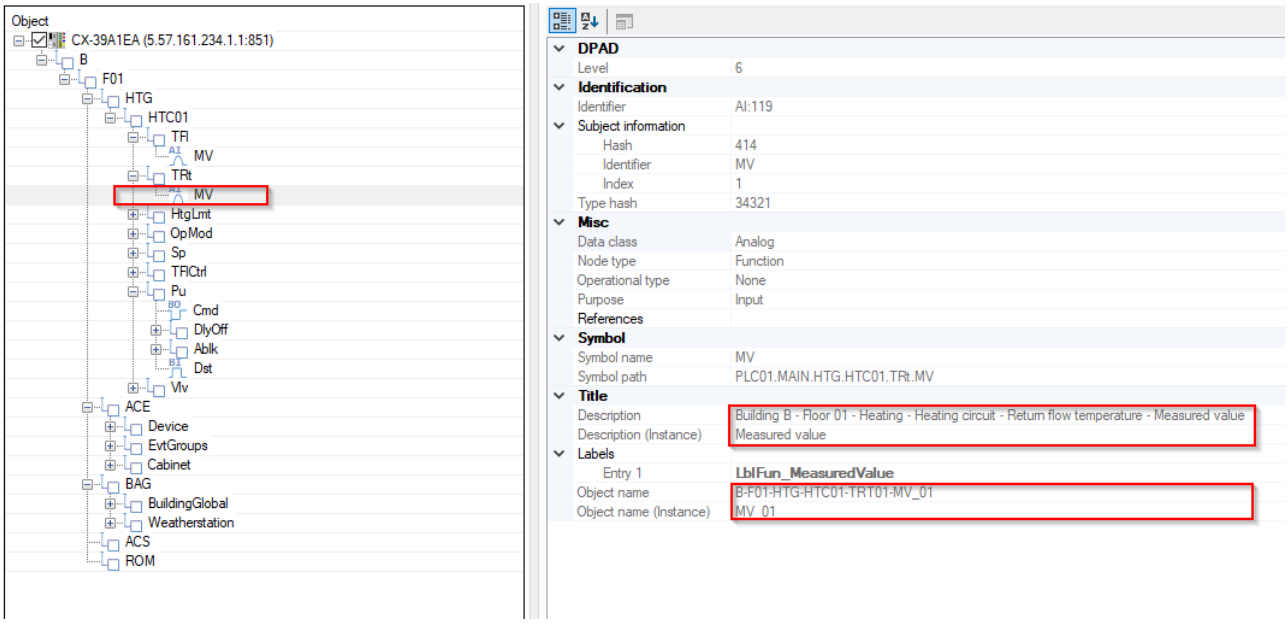
3.6.3.2 Generic generation

During the generic generation of the DPAD, the texts of the levels of the base framework are automatically concatenated together.

Complete object names and object descriptions are created from the names and descriptions of the objects and folders from the project structure.

Here is an example to explain the mechanism:

The illustration shows the measured value (*MV*) of the return temperature sensor (*TRt*) from heating circuit 01 HTC01.



An instance of the function block FB_FB_BA_CarelessDPAD is required for the generic generation of the DPAD. It is recommended to call the function block right at the beginning in MAIN.

The number of levels, the type of separators and the character width of the indexes are defined in the initialization of the DPAD function block. It is important that the number of initialized levels of the DPAD matches those of the project structure.

The indexes are mostly used for naming the aggregate and function levels of the data point. In the initialization sample, two digits (00 to 99) are provided for each index.

```
PROGRAM MAIN
VAR
DPAD        : FB_BA_CarelessDPAD := (
                    aIdentifier := [
    (* Level 1 *)       (   eNodeType:=E_BA_NodeType.eBuilding,        sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 2 *)       (   eNodeType:=E_BA_NodeType.eFloor,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 3 *)       (   eNodeType:=E_BA_NodeType.eTrade,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 4 *)       (   eNodeType:=E_BA_NodeType.ePlant,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2),
    (* Level 5 *)       (   eNodeType:=E_BA_NodeType.eAggregate,       sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2),
    (* Level 6 *)       (   eNodeType:=E_BA_NodeType.eFunction,        sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2)
                    ]
                );
```

The concatenation algorithm of the DPAD function block only recognizes the texts that are to be concatenated if they are defined as placeholders {} within the curly brackets.

Placeholders are very variable and versatile. A more detailed description of the placeholders can be found in the chapter: Object description texts

There are two methods for specifying the initialization texts for the object name and the description:

Direct writing of name and description

Sample:

sObjectname := 'Sample name';

sDescription := 'Sample description';

Use of placeholders

One variant for initializing these texts is the use of placeholders.

These are usually defined in the *FB_init* of the template (in this sample FB_BA_H_HtgCir01 [▶ 975]).

Only texts initialized in placeholders {} are concatenated to the other levels of the project structure by the DPAD algorithm.

The following samples show how the concatenation of texts can be influenced by placeholders.

All of the following DPAD initialization samples refer to the name of the pump in the heating circuit.

Object names and descriptions are described directly in these samples without the use of labels.

Building	Floor	Trade	Plant	Index	Aggregate	Index	Function	Index
B	F01	HTG	HTC	01	PUM	01	FAU	01
B	Floor 01	Heating	Heating Circuit	01	Pump	01	Fault	01

Name concatenated and fixed index

`Pu.sObjectName := '{Pu{Idx=99}}';`	= B-F01-HTG-HTC01-Pu99

Description text with object index

`Pu.sDescription := '{Pump{Idx}}';`	= B-Floor 01-Heat supply systems–Heating Circuit – Pump99

Name concatenated and automatic index

`Pu.sObjectName := '{Pu} ';`	= B-F01-HTG-HTC01-Pu04

Entry point for concatenation defined by !{} and ObjectIndex {Idx} activated

`Pu.sDescription := '!{Pump{Idx}}';`	= Pump04

Name concatenated without index

`Pu.sObjectName := '{Pu01{!Idx}}';`	= B-F01-HTG-HTC01-Pu01
`Pu.sDescription:= '{Pump01}';`	= B-Floor 01-Heat supply systems-Heating-Circuit–Pump01

User-defined name

`Pu.sObjectName := '{123M1{!Idx}}';`	= B-F01-HTG-HTC01-123M1
`Pu.sDescription:= '{Pump 123M1}';`	= B-Floor 01-Heat supply systems–Heating Circuit–Pump123M1

Name not concatenated and fixed index

`Pu.sObjectName := Pu{Idx=99}';`	= Pu99

Transfer of the object index to the description text

`Pu.sDescription:='Pump{Idx}';`	= Pump99

Name not concatenated and fixed index

`Pu.sObjectName :='Pu{Idx=99}';`	= Pu99

Defined own index in the description text

`Pu.sDescription :='Pumpe{Idx=123}';`	= Pumpe123

Name not concatenated and InstanceID of object

`Pu.sObjectName :='Pu{InstID}';`	= Pu201
`Pu.sDescription:='Pump{InstID}';`	= Pump201

Name not concatenated and automatic

`Pu.sObjectName:='Pu';`	= Pu04
`Pu.sDescription:='Pump{Idx}';`	= Pump04

Name not concatenated without index

`Pu.sObjectName := 'Pu{!Idx}';`	= Pu
`Pu.sDescription := 'Pump';`	= Pump

Labels for defining names and descriptions

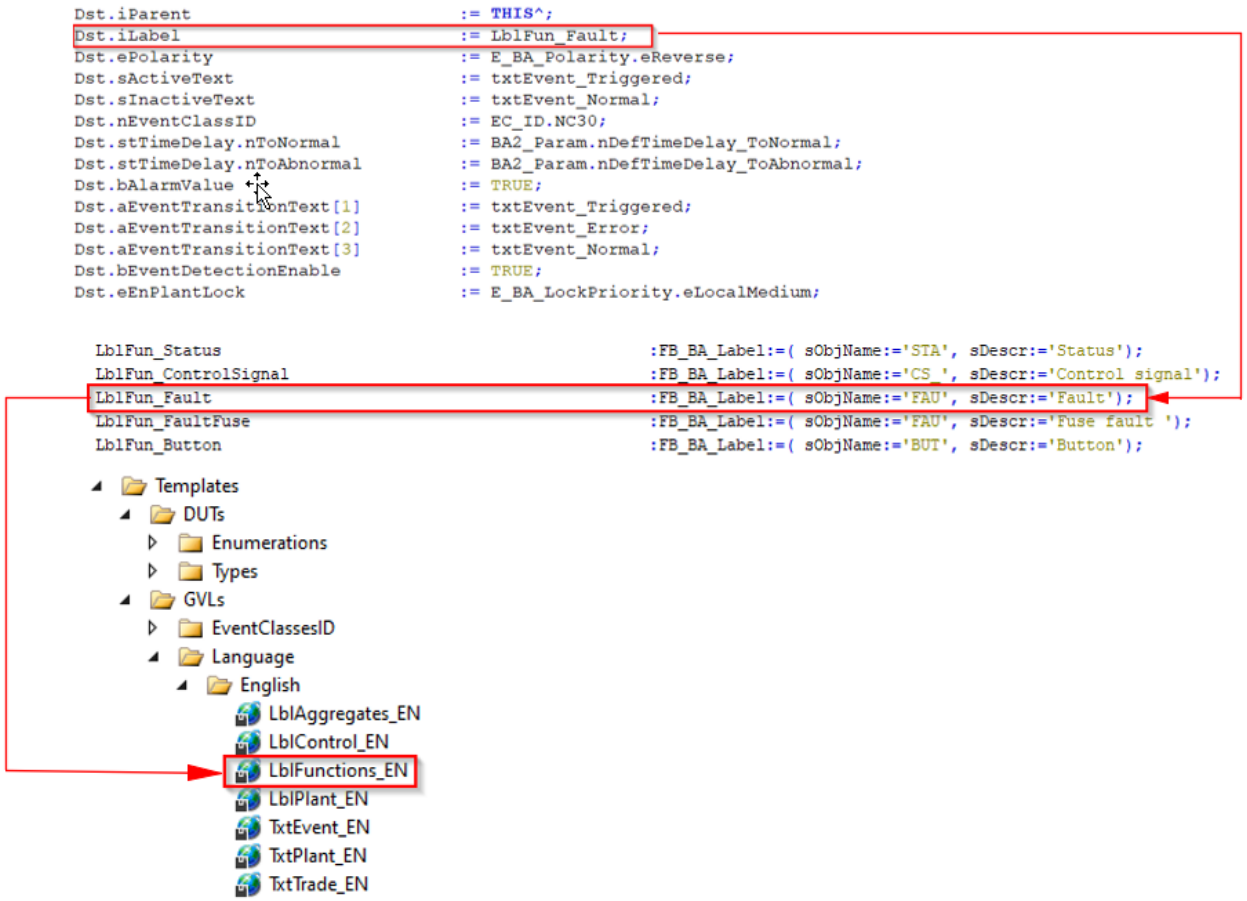
Labels are another option for initializing the texts of names and descriptions of objects.

A label is the paired summary of the texts for the name and description of an object. Initialization is carried out with the function block FB_BA_Label.

The function block FB_BA_Label2x is to be used if the texts of the levels aggregate and function of the data point are to be initialized together.

In TwinCAT 3 Building Automation, the labels are stored in a list of global variables.

Within the TwinCAT 3 Building Automation templates, they are used to initialize the text variables *sObjectname* and *sDescription*.



The label lists are available for naming aggregates (*LblAggregates_xx*), general control functions (*LblControll_xx*) and function names (*LblFunction_xx*).

The lists with the ending _EN contain labels in English and the lists with the ending _DE contain labels in German. For the choice of language, the language that is not desired must be commented out before compiling the controller for the first time.

DPAD mode

For each level of the project structure, the DPAD mode can be used to determine whether and which texts are to be taken into account in the concatenation.

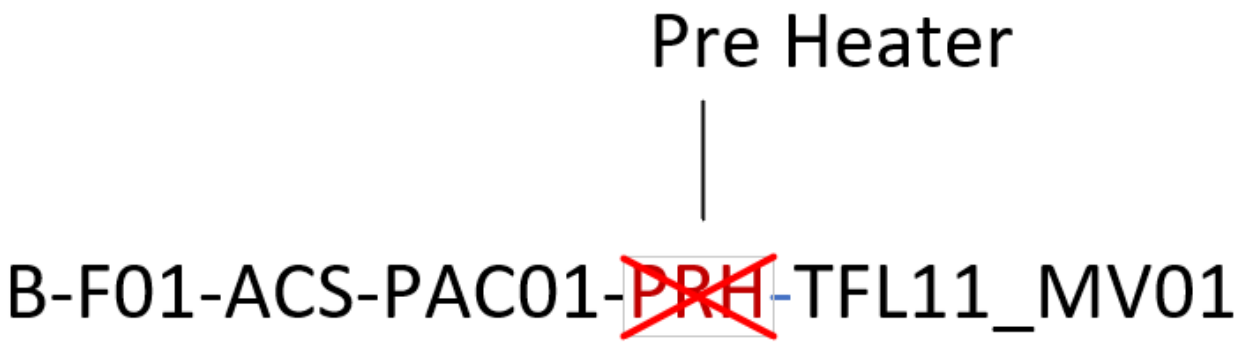
In the following sample, the explanation is based on a ventilation system from the TwinCAT 3 Building Automation template repository.

The main aggregates of the ventilation system, the heater (*PreHtr*), the energy recovery (*ERC*) and the cooler (*Col*), each have their own function block, within which all components or sub-aggregates are called. The main aggregates are not part of the levels in the DPAD. In the definition of the DPAD, the levels for the sub-aggregates and functions are defined directly after the Plant level.

```
PROGRAM MAIN
VAR
DPAD        : FB_BA_CarelessDPAD := (
                    aIdentifier := [
    (* Level 1 *)       (   eNodeType:=E_BA_NodeType.eBuilding,        sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 2 *)       (   eNodeType:=E_BA_NodeType.eFloor,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 3 *)       (   eNodeType:=E_BA_NodeType.eTrade,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 0),
    (* Level 4 *)       (   eNodeType:=E_BA_NodeType.ePlant,           sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2),
    (* Level 5 *)       (   eNodeType:=E_BA_NodeType.eAggregate,       sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2),
    (* Level 6 *)       (   eNodeType:=E_BA_NodeType.eFunction,        sSeparator_ObjectName := '-',   sSeparator_Description := ' - ',    nIndexDigits := 2)
                    ]
                );
```

This means that there is one more level in the PLC than there is in the DPAD.

The heater level must therefore be excluded when generating the object names.



This is done with the DPAD mode and the following initialization.

```
OpMod.iLabel                                    := LblCtl_OperationMode;
OpMod.sObjectName                               := '{Idx=01}';
OpMod.eDPADMode                                 := E_BA_DPADMode.eInclude;

PreHtr.iParent                                          := THIS^;
PreHtr.iLabel                                           := LblCtl_PreHeater;
PreHtr.sObjectName                                      := '{Idx=01}';
PreHtr.eDPADMode                                        := E_BA_DPADMode.eIncludeDescription;
PreHtr.eDPADMode                                        := E_BA_DPADMode.;


PreHtr.TFl.MV.bEventDetectionEnable                     := TRUE;
PreHtr.TRt.MV.eEnPlantLock                              := E_BA_LockPrior
PreHtr.TRt.MV.bEventDetectionEnable                     := TRUE;
PreHtr.TRtCtrl.PreRinseDst.eEnPlantLock                 := E_BA_LockPrior
PreHtr.TRtCtrl.PreRinseDst.bEventDetectionEnable        := TRUE;
PreHtr.TFrost.MV.eEnPlantLock                           := E_BA_LockPrior
PreHtr.TFrost.MV.bEventDetectionEnable                  := TRUE;
PreHtr.FrostThermostat.Input.eEnPlantLock               := E_BA_LockPriority.eMedium;
PreHtr.FrostThermostat.Input.bEventDetectionEnable      := TRUE;
PreHtr.Pu.Dst.eEnPlantLock                              := E_BA_LockPriority.eMedium;
PreHtr.Pu.Dst.bEventDetectionEnable                     := TRUE;
```

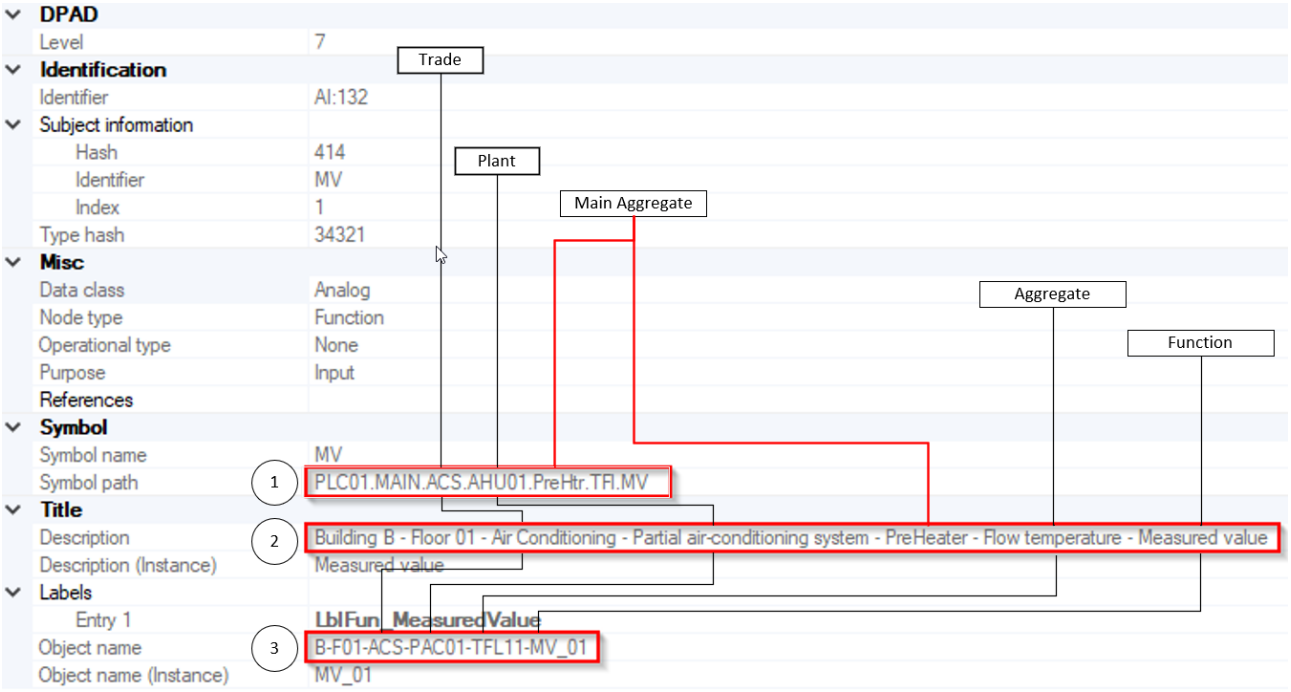
```
PreHtr.iParent                                          := THIS^;
PreHtr.iLabel                                           := LblCtl_PreHeater;
PreHtr.sObjectName                                      := '{Idx=01}';
PreHtr.eDPADMode                                        := E_BA_DPADMode.eIncludeDescription;
PreHtr.TFl.MV.bEventDetectionEnable                     := TRUE;
PreHtr.TRt.MV.eEnPlantLock                              := E_BA_LockPriority.eMedium;
PreHtr.TRt.MV.bEventDetectionEnable                     := TRUE;
PreHtr.TRtCtrl.PreRinseDst.eEnPlantLock                 := E_BA_LockPriority.eNoLock;
PreHtr.TRtCtrl.PreRinseDst.bEventDetectionEnable        := TRUE;
PreHtr.TFrost.MV.eEnPlantLock                           := E_BA_LockPriority.eMedium;
PreHtr.TFrost.MV.bEventDetectionEnable                  := TRUE;
PreHtr.FrostThermostat.Input.eEnPlantLock               := E_BA_LockPriority.eMedium;
PreHtr.FrostThermostat.Input.bEventDetectionEnable      := TRUE;
PreHtr.Pu.Dst.eEnPlantLock                              := E_BA_LockPriority.eMedium;
PreHtr.Pu.Dst.bEventDetectionEnable                     := TRUE;
```

The *eIncludeDescription* setting is selected for the heater. The description text of the heater is thus concatenated and the name of the heater is excluded when generating the object names.

The result of this parameterization can be seen in the Site Explorer.

The symbol path (1) has 7 levels, which means that the description text (2) and the object name (3) also have 7 levels, whereby one level is not displayed for the object name.

In the description text (2) of the flow temperature sensor of the air heater, the description text *'PreHeater'* is included and excluded in the object name (3).



3.6.3.3 Object description texts

Object description texts are used to create a unique name for all objects in the project structure [▶ 38]. (This is called DPAD [▶ 42] or BAS (user address key))

Each level within the project structure is given a name and a description text.

The concatenation of the partial names from the project structure results in the name and description text of an object. Concatenated texts are created using placeholders or labels.

The following object parameters are summarized as texts:

- ObjectName
- Description

The texts of a object [▶ 32] 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 saved persistently), they will no longer be generated again!

From this point on, texts can only be changed by the user!

Labels

A Label FB (e.g. in a GVL) can be used in the declaration of one or more objects [▶ 32] to generate its texts automatically.

A distinction is made between the following label types:

- Simple labels (FB_BA_Label)
  Provide 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 (simple or dual) label can be assigned to each object [▶ 32].

Each view [▶ 32] 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 ObjectName and Description parameters are initialized with standard placeholders as follows:

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

During object initialization, the text generation mechanism analyses a text for defined placeholders.

Only if a valid placeholder exists, the mechanism can replace it with an automatically generated text. On the other hand, a manually defined text would have replaced a placeholder and would not be overwritten.

As soon as the texts are generated, variables no longer contain placeholders.

Even if the device is restarted and parameters (persistent variables) have been saved, the current texts remain (automatically generated, manually defined or even changed by an operator 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**
  Automatic indexing is set in the project template for objects in the last two levels of the project structure. This means that the ObjectName parameter also receives a consecutive index number in addition to a defined name. 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**
  Indexing can be specified using placeholders and their corresponding index attributes.

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

A given index is also automatically used for parameters of subsequent objects [▶ 32] if they do not also explicitly define an index.

Sample:

As 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!

Concatenation

The texts of an object [▶ 32] are concatenated with texts of parent objects [▶ 30] under certain conditions:

- Parent settings [▶ 241]
- Settings in the DPAD-FB [▶ 42] used
- Manual overwriting using index attributes
- Global settings [▶ 284]
- 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

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 [▶ 696]) 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

- Placeholder:
  `{}`
- Placeholder that overwrites a parameter value (e.g. `sObjectName := '{MyValue}'`):
  `{Value}`
- Placeholder, as a representing attribute (e.g. `sDescription:= '{Idx}'`):
  `{Attribute}`
  `{Attribute1,Attribute(n),...}`
- Placeholder, as defining attribute (e.g. `sDescription:= '{Idx=10}'`):
  `{Attribute=Value}`
  `{Attribute1=Value,Attribute(n)=Value,...}`

- Placeholder, as a defining attribute which overwrites a parameter value (e.g. `sDescription:= '{Pump{Idx=10}}'`):
  `{Value{Attribute=Value}}`
- Placeholder with multiple attributes:
  `{Placeholder{Attribute1=Value,Attribute2=Value}}`
- Placeholder, as a representing parameter (e.g. `sDescription:= '{} {InstID}'`):
  `{Parameter}`
- Placeholder that denies concatenation with parent elements. The entry point of the concatenation is defined. From this point on, only placeholders of the child elements are concatenated:
  `!{}`
- Placeholder that denies certain attributes (switching off attributes):
  `{Value{!Attribute}}`

Samples

Parameter ObjectName with defined placeholder (concatenates automatically generated values):
```
sObjectName := '{}'
```

Concatenates parameter ObjectName with overwritten value:
```
sObjectName := '{AAA01_BB01}'
```

Concatenates parameter Description with placeholders and predefined additional information that is not replaced:
```
sDescription := '{} North'
```

Concatenates parameter Description with a placeholder and extends a substitute index attribute (the index value of the ObjectName is transferred):
```
sDescription := '{Idx}'
```

Concatenated parameter ObjectName with placeholder including defining index attribute:
```
sObjectName := '{Idx=10}'
```

Concatenated parameter ObjectName with placeholder including substitute (aggregate) and defined (function) index attribute:
```
sObjectName := '{Idx,Idx=10}'
```

Concatenated parameter Description with overwritten value and a defined (function) index attribute:
```
sDescription := '{Pump{Idx=5}}'
```

Concatenated parameter Description with placeholder and a substitute parameter:
```
sDescription := '{} - ID {InstID}'
```

ObjectName with default placeholder but without concatenation (concatenation denied):
```
sObjectName := '!{}'
```

Concatenated parameter ObjectName with overwritten value, but without index attribute (attribute denied):
```
sDescription := '{PU{!Idx}}'
```

Error messages

Name	Text	Description
[T23]	Internal error on parsing placeholder "...": Defined value and attributes at once!	Defining placeholder values and attributes at the same time is not allowed, because a Defined value prevents attributes from being applied.

Generating texts

The following sequence is used to decide how a placeholder is replaced:

1. Overwrite texts
   If defined, a manually initialized title is used:
   ```
   fbSensor : FB_BA_AO_IO := (sObjectName:='Tmp03', sDescription:='Temperature sensor 3');
   ```
2. Labels
   If titles are not overwritten, a label can be applied:
   ```
   SomeLabel : FB_BA_Label := (sName:='PU', sDescription:='Pumpe');
   Pump : FB_BA_View := (iLabel:=SomeLabel);
   ```
3. SymbolName
   If nothing is defined, the following default values are used:
   - ObjectName receives the information from the symbolpath.
   - Description receives the information from the symbolname.

Samples

Initialization by means of labels

Transfer of the texts defined in the label to objects [▶ 32].

```
{region 'Labels in GVL "MyLabels"'}
  MV    : FB_BA_Label    := (sName:='MW',  sDescription:='Messwert');
  TFl    : FB_BA_Label    := (sName:='TFL',  sDescription:='Vorlauftemperatur');
  TRt    : FB_BA_Label    := (sName:='TRL',  sDescription:='Rücklauftemperatur');
  PreHtr    : FB_BA_Label    := (sName:='LE',  sDescription:='Lufterhitzer');

  TFl_MV    : FB_BA_Label2x    := (iAggregate:=TFl,  iFunction:=MV);
  TRt_MV    : FB_BA_Label2x    := (iAggregate:=TRt,  iFunction:=MV);
{endregion}
{region 'Objects in some program'}
  PreHtr    : FB_BA_View    := (      iLabel:=MyLabels.PreHtr);
  TFl    : FB_BA_AI_Raw    := (iParent:=PreHtr,  iLabel:=MyLabels.TFl_MV);
  TRt    : FB_BA_AI_Raw    := (iParent:=PreHtr,  iLabel:=MyLabels.TRt_MV);
{endregion}
```

3.7 PLC

3.7.1 Control and regulation mechanisms

This chapter explains the general relationships between open-loop and closed-loop control.

3.7.1.1 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 [▶ 256] of the function blocks FB_BA_SchedA [▶ 222], FB_BA_SchedB [▶ 223] and FB_BA_SchedM [▶ 225].

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 [▶ 256] of the function blocks FB_BA_SchedA [▶ 222], FB_BA_SchedB [▶ 223] and FB_BA_SchedM [▶ 225].

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 [▶ 256].

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 [▶ 256] 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 [▶ 189] is provided for this purpose. The use of this parameter builder is described in its documentation.

Local exceptions

The parameter aException [▶ 256] 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 [▶ 181] is provided for this purpose. The use of this parameter builder is described in its documentation.

Global exceptions

The parameter aCalendar [▶ 256] 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 [▶ 188] is provided for this purpose. The use of this parameter builder is described in its documentation.

3.7.2 Fast task for serial communication

Various bus systems, such as DALI, MBus or EnOcean®, are used in building automation. Many of these systems are integrated by Beckhoff hardware and are based on serial communication. Communication with the serial hardware often requires a task that is processed in a significantly shorter time than the PLC task in which the normal application runs.

In addition, the fast task must have a higher priority than the PLC task.

This relationship is well described in the Introduction to serial communication:

Within TF8040, the following components currently require a fast task for processing serial information:

- the Thies weather station, which is based on standard serial communication at 9600 bps (see Integration Thies weather station [▶ 1097]).
- the DALI function blocks, which are based on the TC3_DALI library. The integration of communication is described in the following chapter (see DALI communication [▶ 868]).

3.8 HMI

3.8.1 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 [▶ 99] library for the PLC and the BaSite-Extension [▶ 1262] 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 [▶ 77].

Project structure

The generic functions within *TcHmiBa* are based on the project structure [▶ 38], which is established in the PLC by a child/parent relationship. With this structure it is possible to derive a generic navigation [▶ 77] through all objects on a controller.

Further information on the project structure [▶ 38] can be found in the documentation.

Events

The events of all linked controllers can be collected and displayed centrally in an event list [▶ 1166]. 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 [▶ 55].

3.8.2 Trending

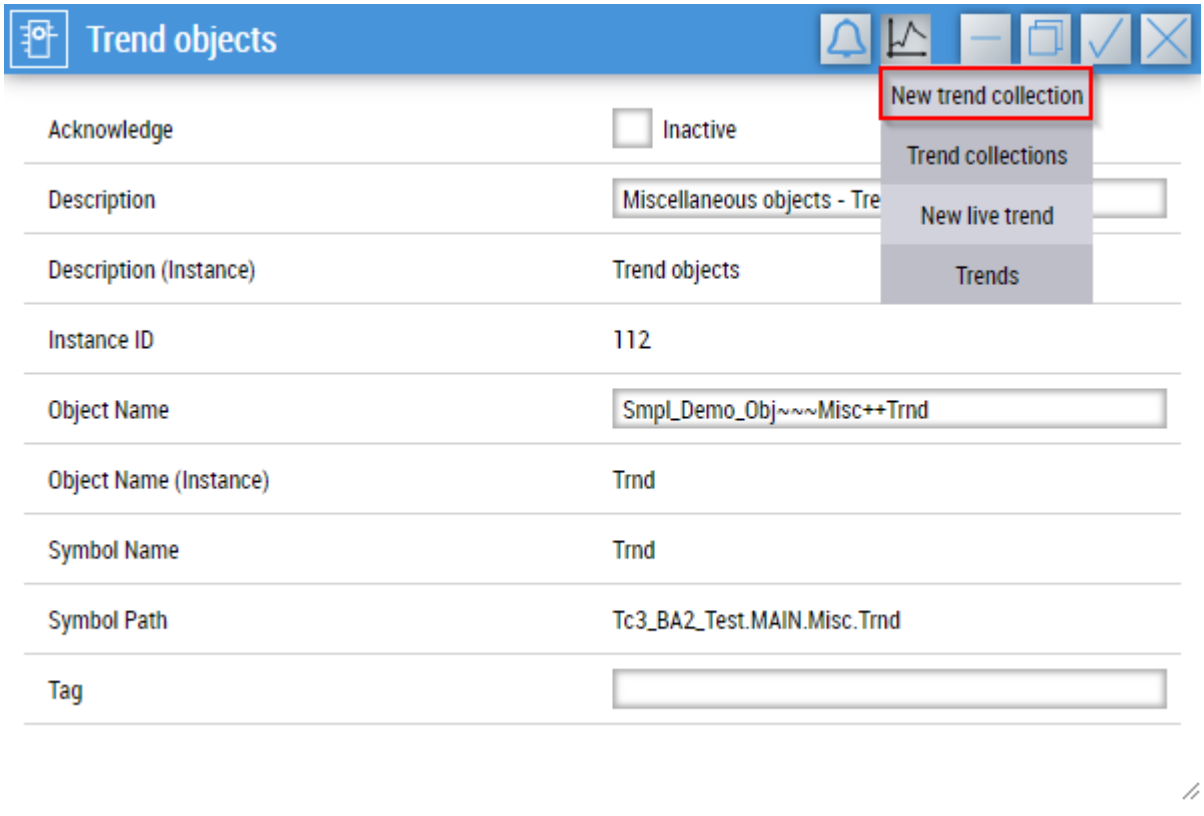
Trends can be used and managed in the HMI. The functions described here can only be used with the generic functions [▶ 77] of TcHmiBa.

Trend collections

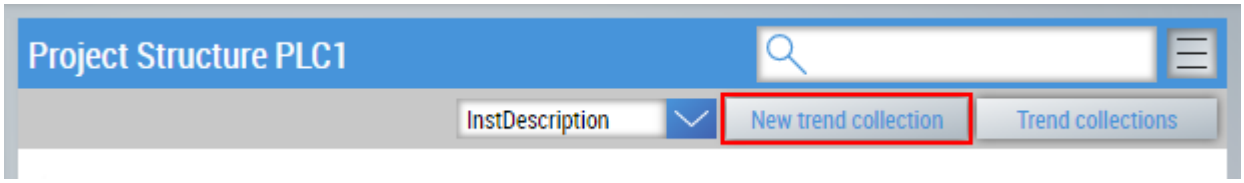
Displaying individual trend curves is often not sufficient, as they are difficult to compare in this way. For this reason there is an option to create trend collections in order to compare the trend curves in a chart.

Configurator

The configurator can be opened via the menu in the Parameter window [▶ 1170] of a view and then only displays objects from this level.

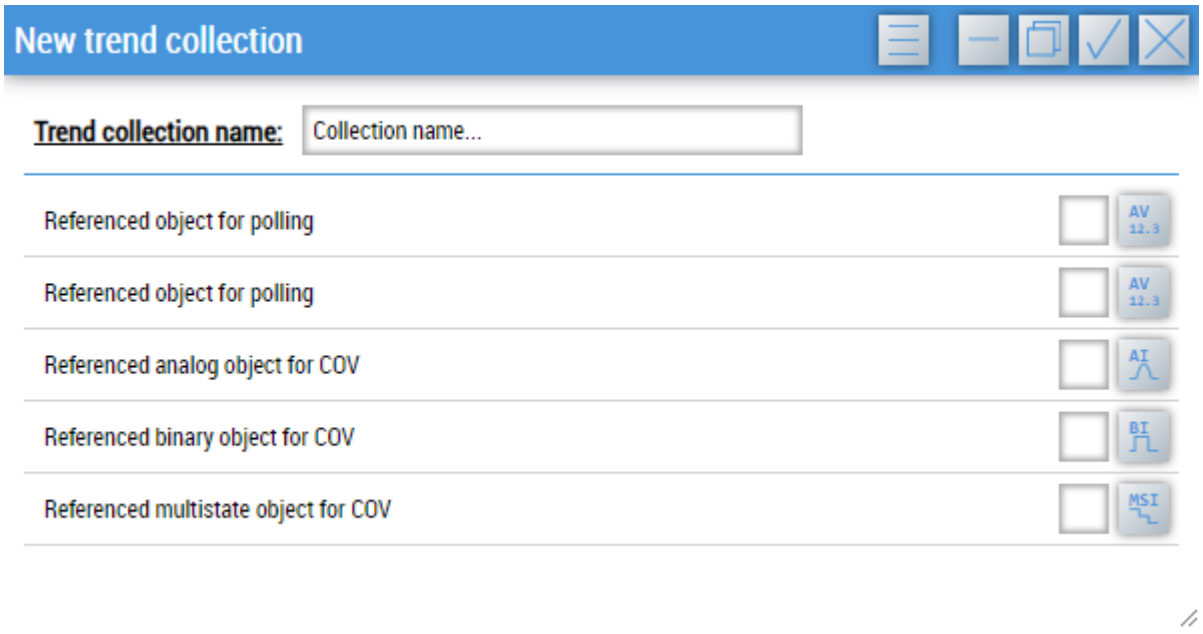


In addition, it can also be opened via the menu of ProjectNavigationTextual [▶ 1167].

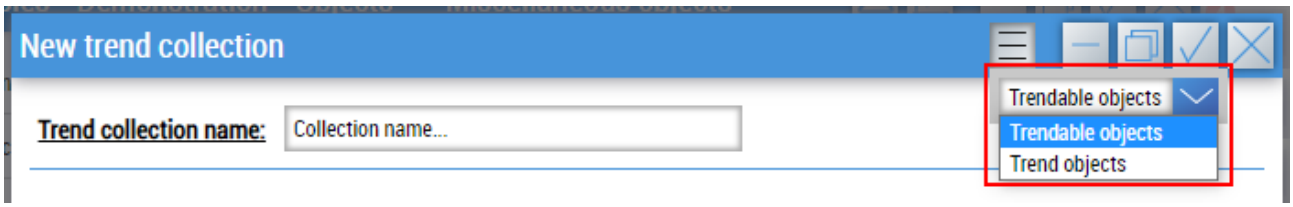


Creating trend collections

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

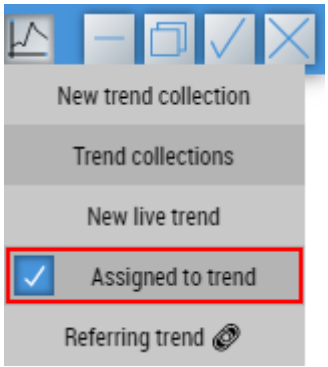


The menu offers the option to select the objects to be displayed.

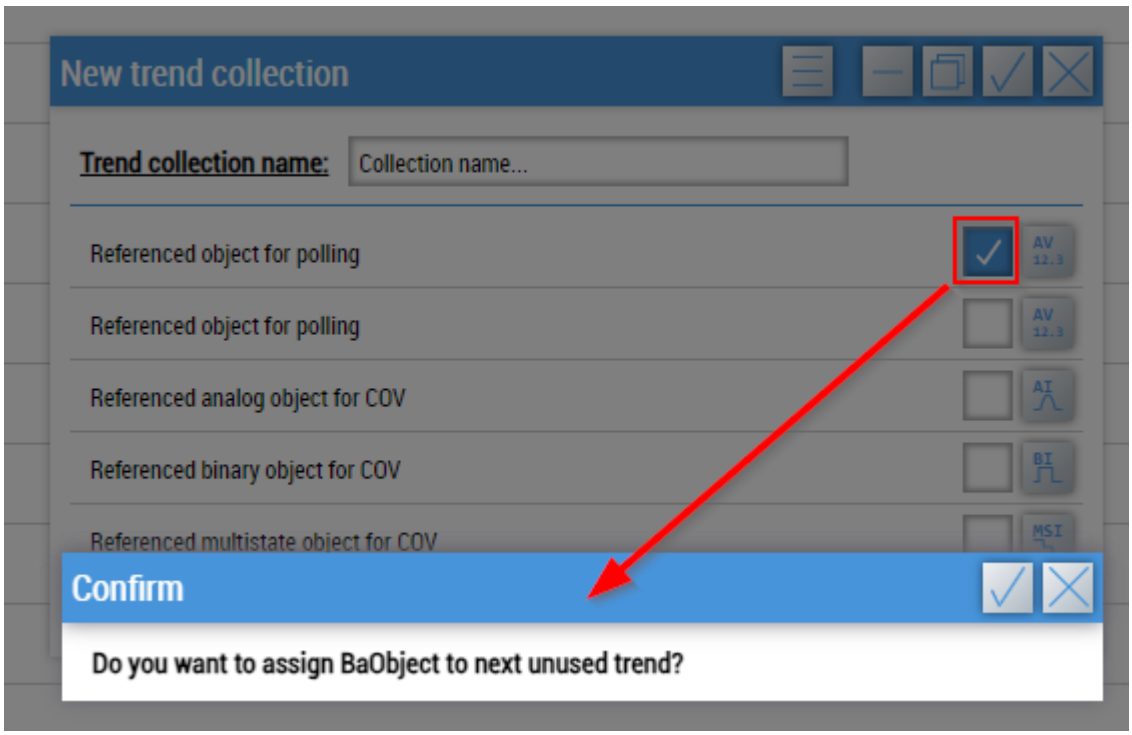


- **Trendable object**: Displays all trendable objects (e.g. *FB_BA_AV*).
- **Trend object**: Displays all trend objects (*FB_BA_Trend*).

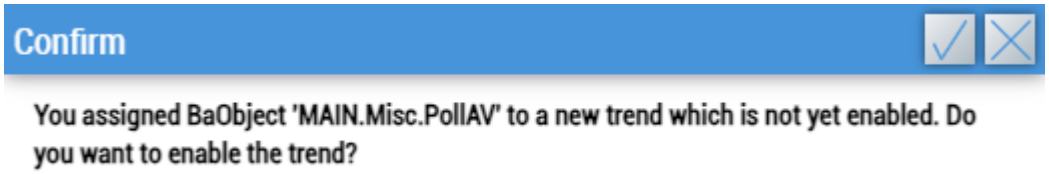
A **Trendable object** must be assigned to a trend object for use in the trend. The state for this can be viewed and also changed in the menu of the Parameter window [▶ 1170].



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**.



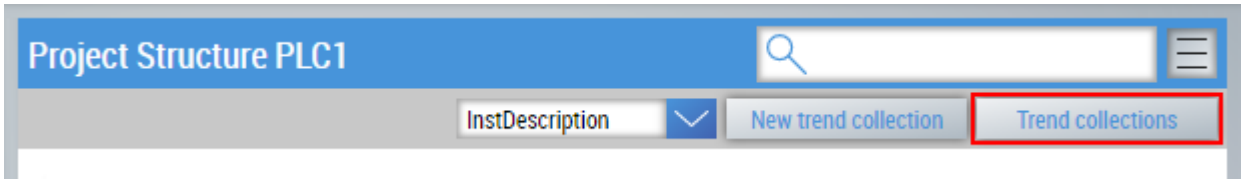
When the trend collection is complete, it will be available in the trend collection view after confirming the dialog.

Observing trend collections

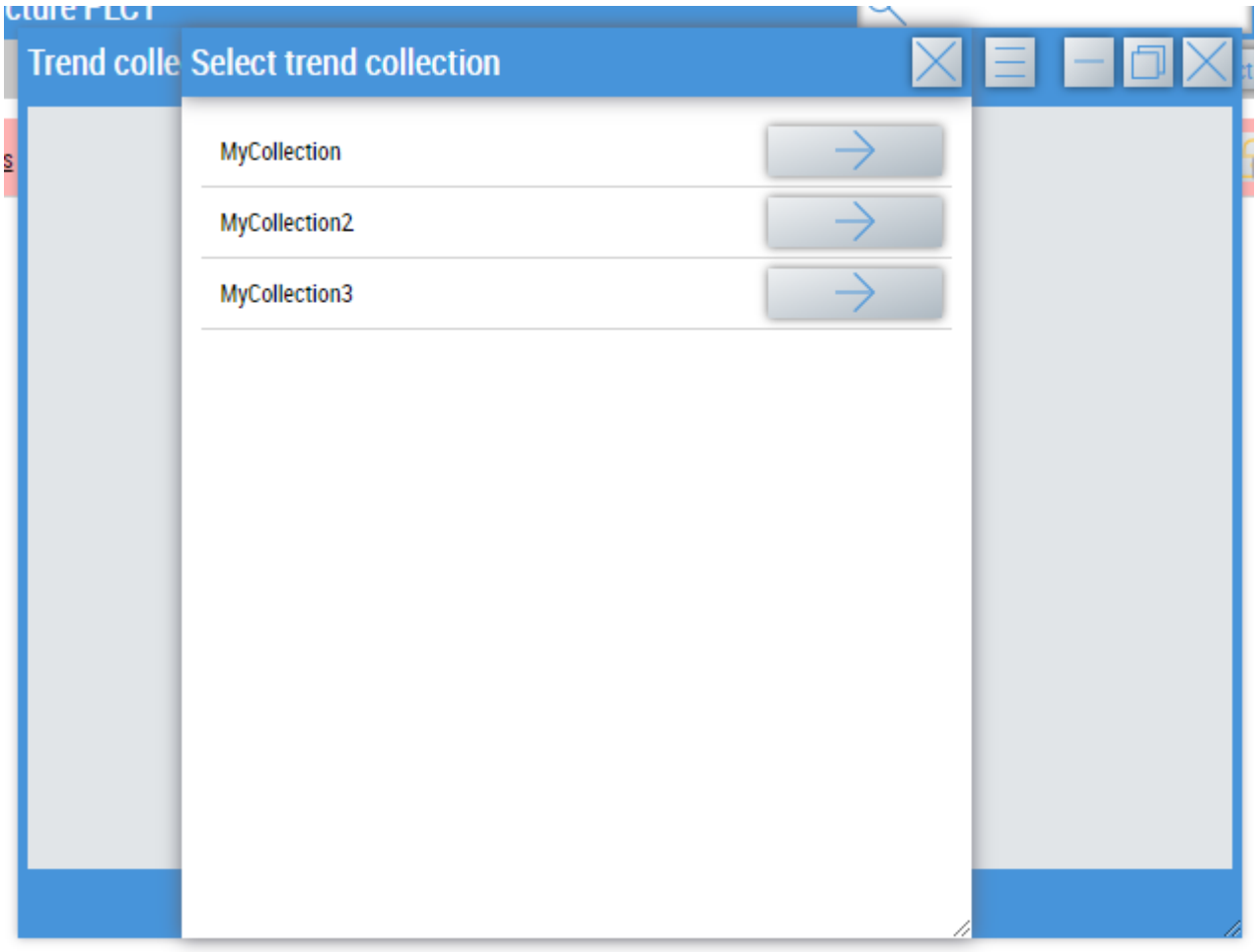
The created trend collections can be viewed and added in the trend collection view.

This can be called up in two ways.

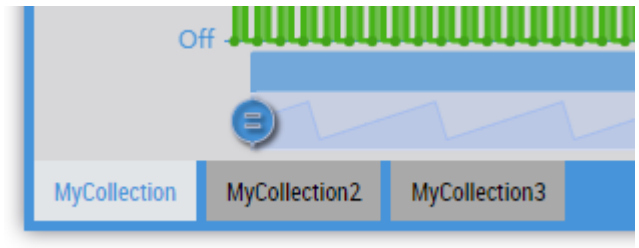
1. By calling the function OpenTrendCollectionView [▶ 1258].
2. From the ProjectNavigationTextual [▶ 1167].



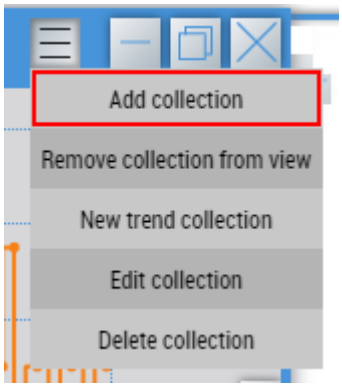
When the dialog is opened for the first time, an overview appears. A trend collection can be selected from it and then displayed in a Trend Control [▶ 1175].



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.

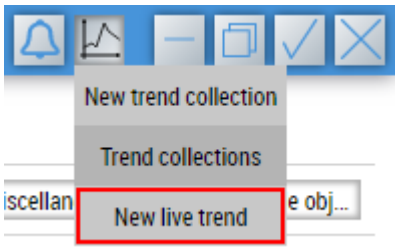
- **Remove collection from view**: Removes the trend collection in the active tab from the view (it will not be deleted).
- **Edit collection**: Opens the configurator to edit the trend collection in the active tab.

- **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 [▶ 1170].

This action is only available if the object itself is trendable (e.g. FB_BA_MV [▶ 237]) or is a view with trendable objects.

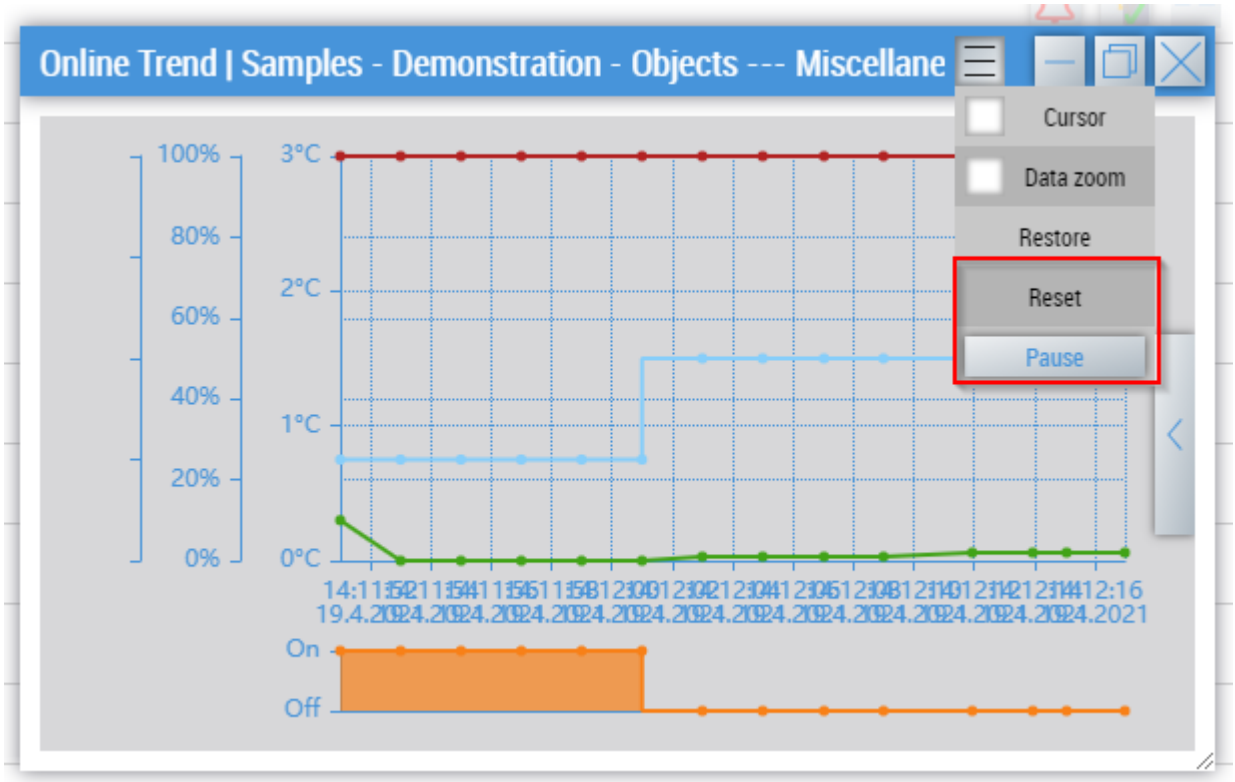


After pressing the button, the live trend opens, which records either only one trendable object or all trendable objects of the view in a FIFO memory. The functions in the menu are similar to those of the Trend Control [▶ 1175].

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.

3.8.3 Feedback

In some TcHmiBa controls [▶ 1122] 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.

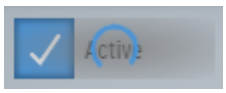
Accordingly, the TcHmiBa controls always display only the feedback value (e.g. *StateFeedback* of the *Checkbox*), which ensures that the current value is always displayed in the HMI.

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

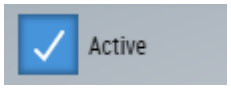
Procedure

The procedure using the example of the checkbox [▶ 1135] 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



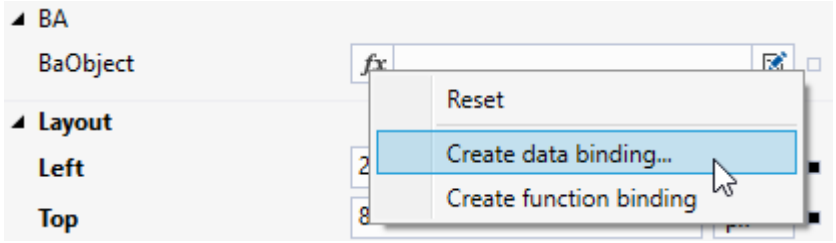
There are only two possible outputs.
a. Value of symbol linked to *StateFeedback* changes to the same value written to *State*.
b. Value of *StateFeedback* does not change.

3.8.4 BaObject

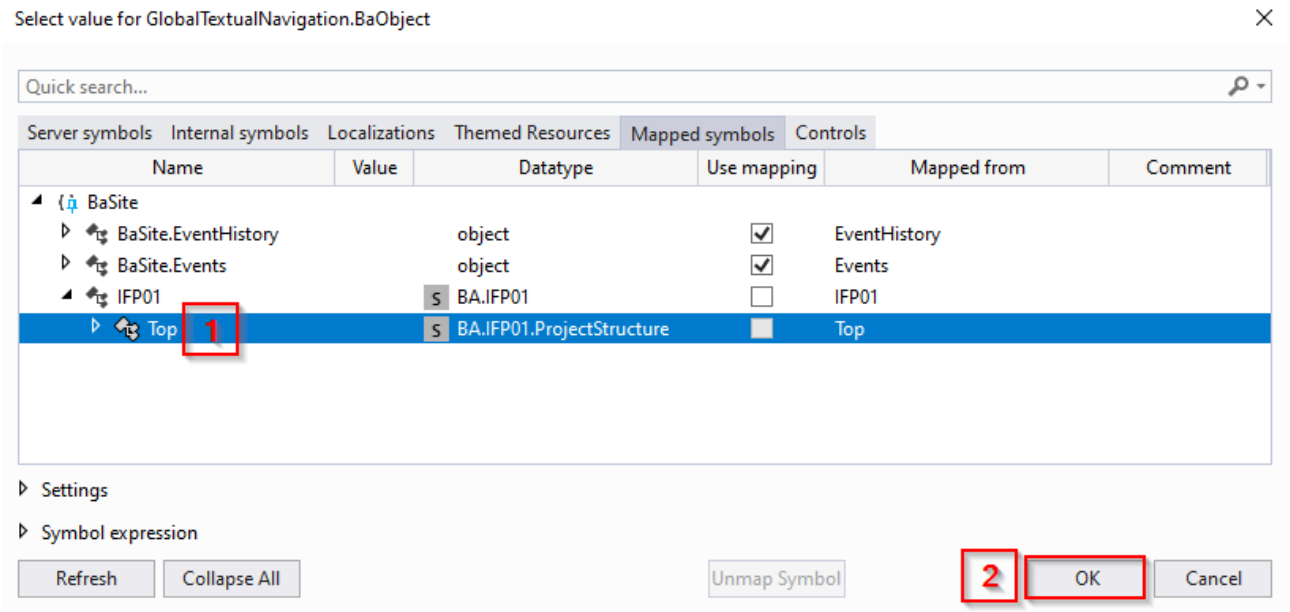
The BaSite-Extension [▶ 1262] provides the TcHmi with the structure of the PLC in a special form. BaObjects are created from this, which can then be linked to the BaObject attribute from the BA category of a TcHmiBa control [▶ 1122].

Create binding

Call up the action to create Binding from the menu.



Select symbol.



Symbol is assigned.

BaObject

A BaObject is the representation of an object from the project structure [▶ 38] of a PLC based on TF8040. The TcHmiBa controls generically use the data and functions from the BaObjects that they need for their work.

Further information can be found in the BaObject handling [▶ 1227] documentation.

3.8.5 BaTemplate

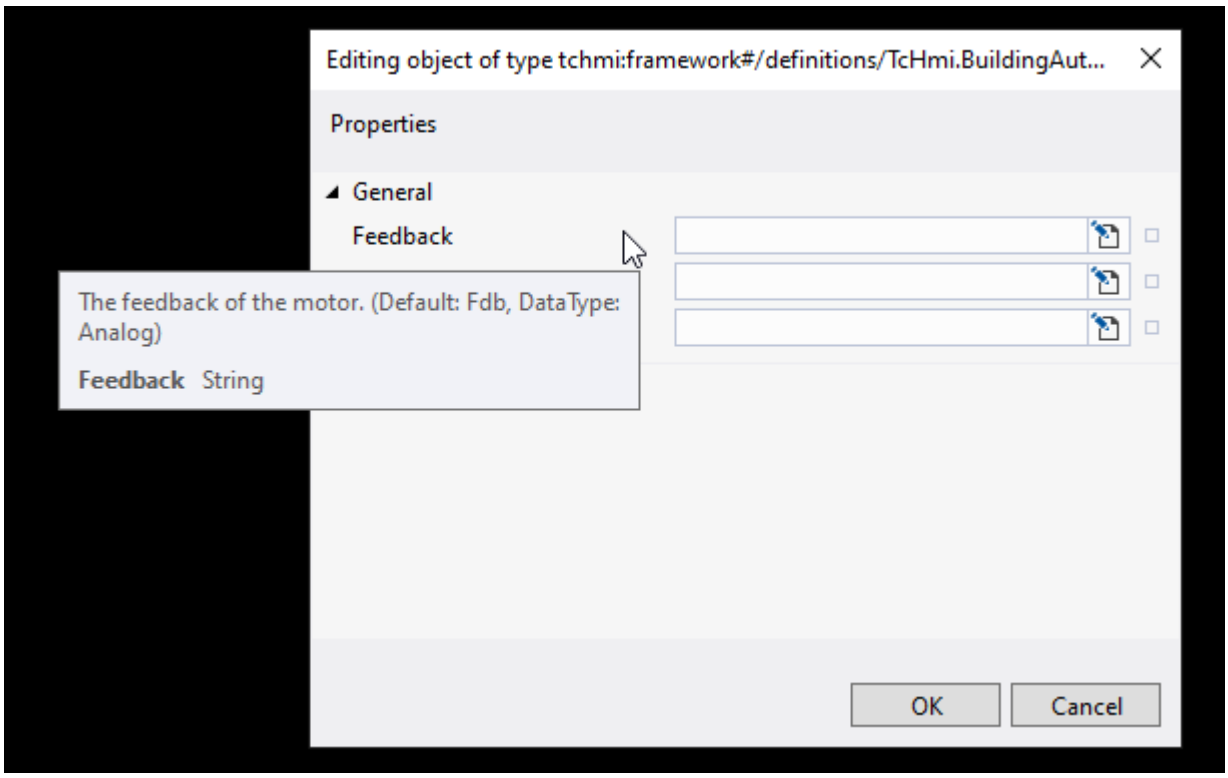
TcHmiBa templates are TcHmiBa controls [▶ 1122] that directly represent a TF8040 standard PLC template in the TcHmi.

BaTemplateDescription

TcHmiBa templates are available for almost all TF8040 standard PLC templates. The scope of functions includes a validation of the linked BaObject [▶ 60] 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 [▶ 1229], due to project specific peculiarities, the new name can be communicated to the TcHmiBa template via the attribute *BaTemplateDescription* from the BA category.

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.



Further information can be found in the BaTemplate handling [▶ 1227] documentation.

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
```

3.8.6 BaInterface

TcHmiBa controls [▶ 1122] that support the BaInterface can also be used without the BaSite and TF8040 PLC templates. Functionality and behavior may vary.

For more information, see the documentation for BaInterfaceHandler [▶ 1245].

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

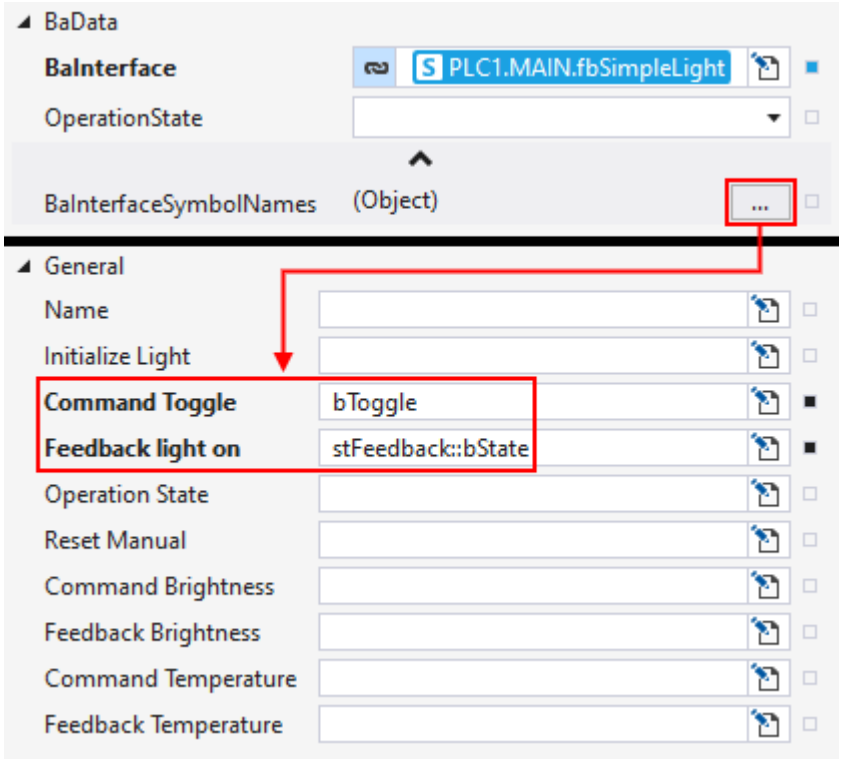
Explanation using the sample of a self-created PLC function block *FB_SimpleLight* for toggling a light. The state-variable *bState* is stored in a structure.

```
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 [▶ 1203].

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
```

4 Tutorials

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

Getting started

PLC

Creating a TwinCAT PLC project [▶ 64].

HMI

Creating a TcHmiBa project [▶ 71].

Video tutorials

Getting started with TF8040 is made even easier with a series of video tutorials. The videos are available on the Beckhoff website.

Filter: Industry -> Building automation

4.1 PLC

Chapter with instructions for programming PLC applications.

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 [▶ 693] 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

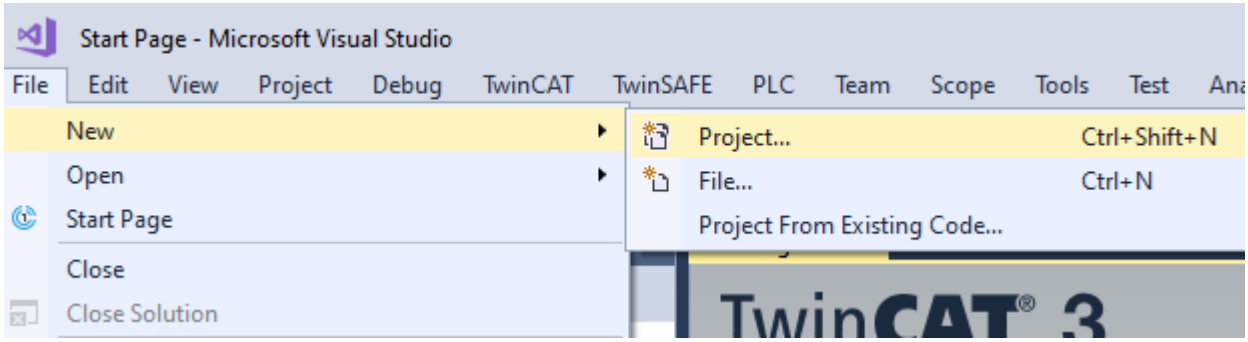
Install current XAR on full Windows systems.

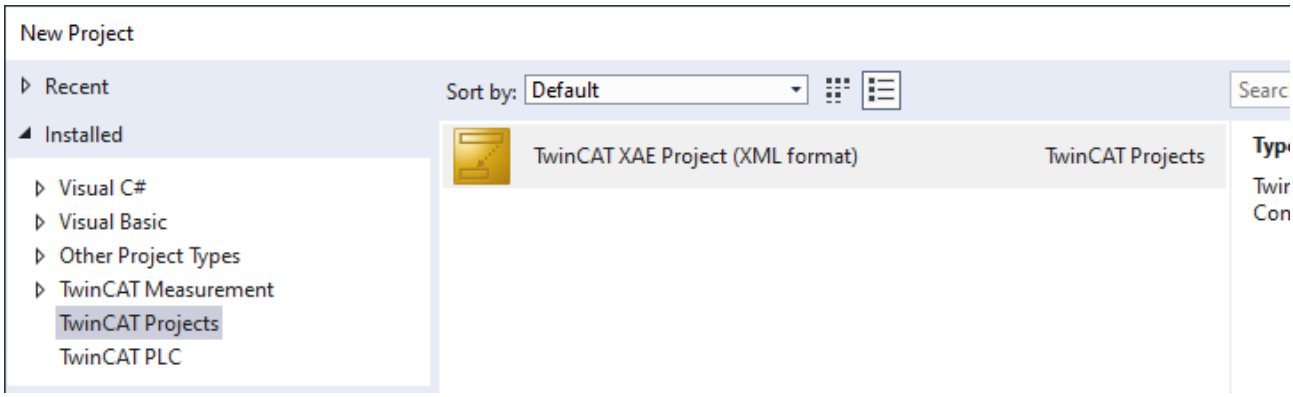
Image

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

Creating a TwinCAT project

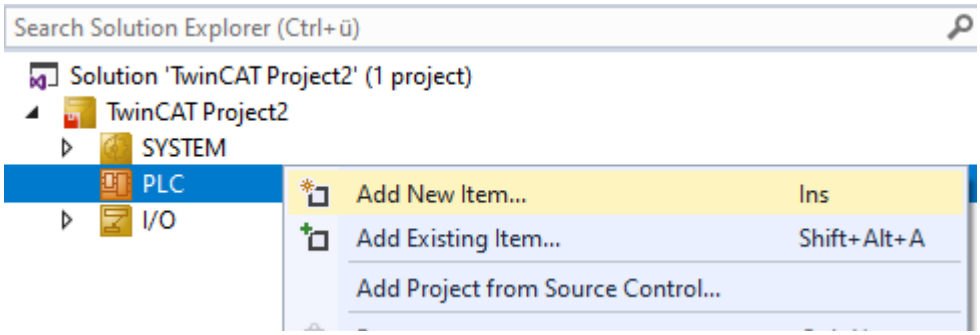
- Create new Solution.

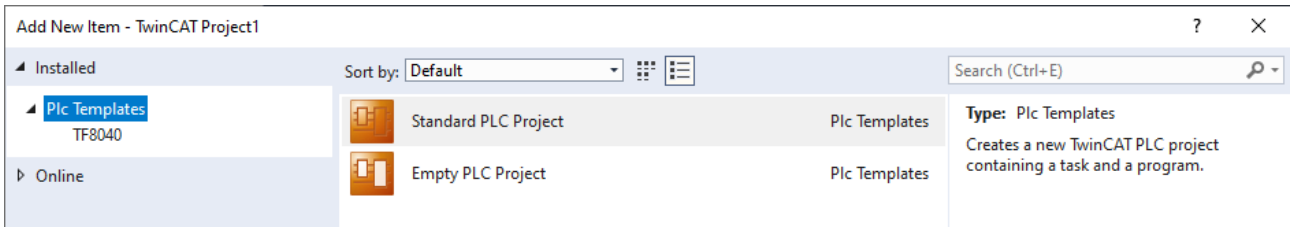




Adding a PLC project

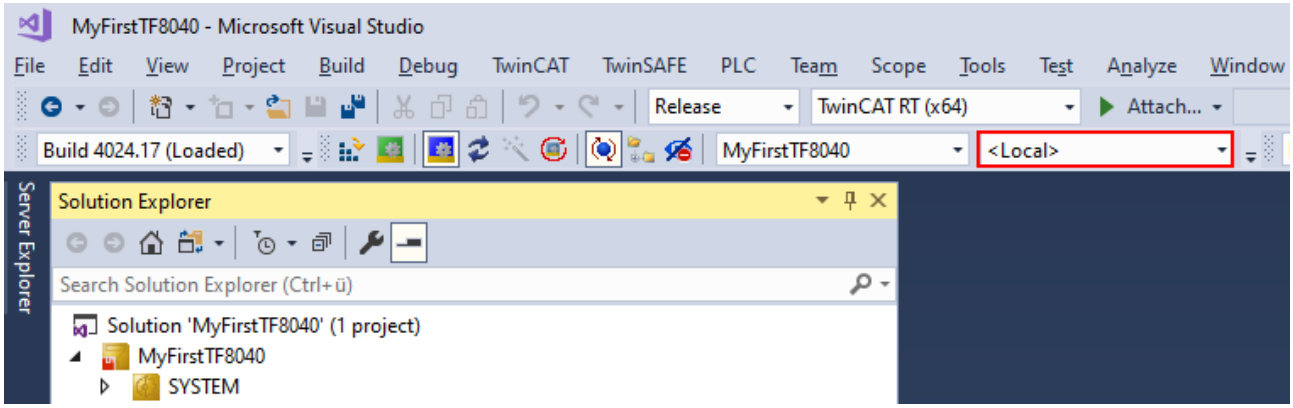
- Add matching PLC project template [▶ 693]:

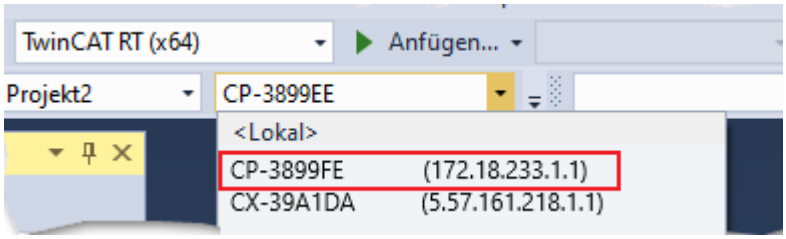




Choose Target System

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





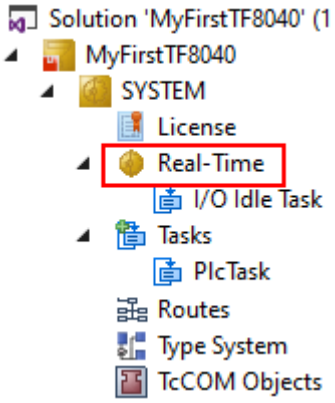
For more information refer to the chapter Choose Target System.

Project settings

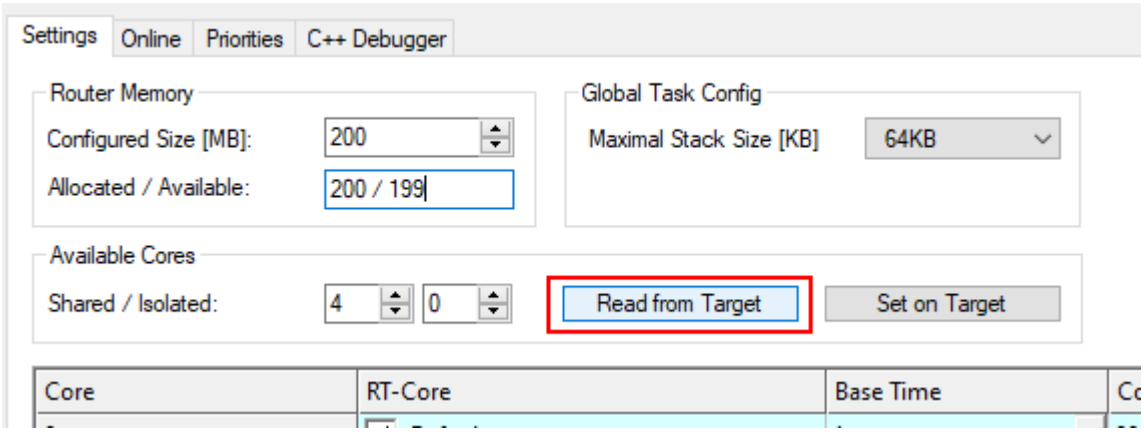
After the target system is selected, you can proceed with the project settings.

System

Real-time

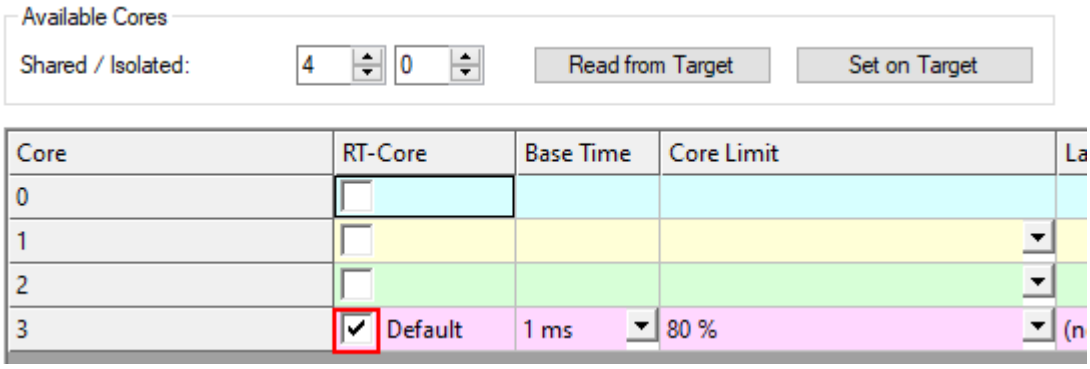


- Reading the existing hardware configuration:



- 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.

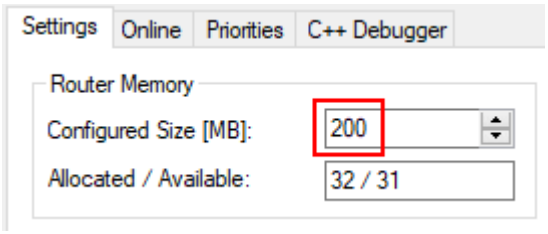


NOTICE

Do not use isolated cores.

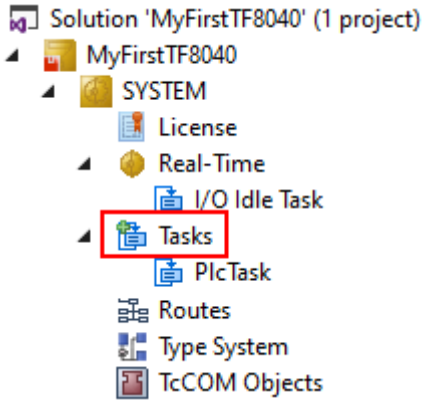
- Configuring the router memory

The memory should generally be set to 200 MB:



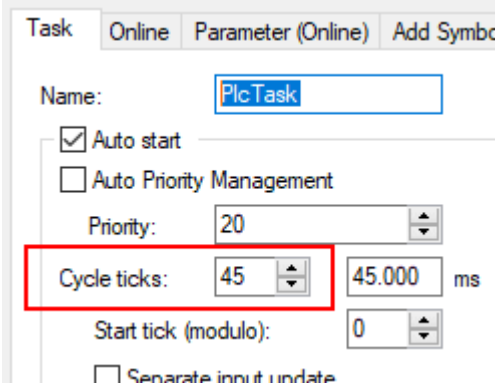
A restart of the operating system (on the target device) is required to apply the setting.

Tasks



Settings for the PLC task:

- Recommended cycle time: 45 ms



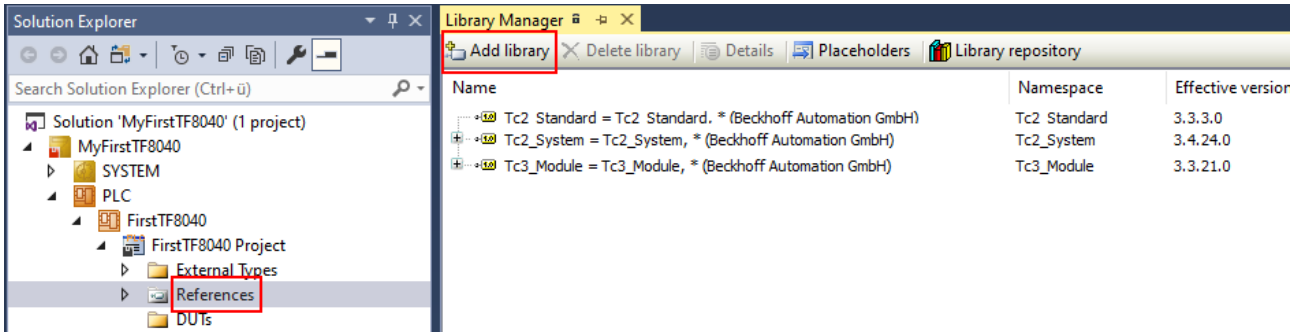
PLC

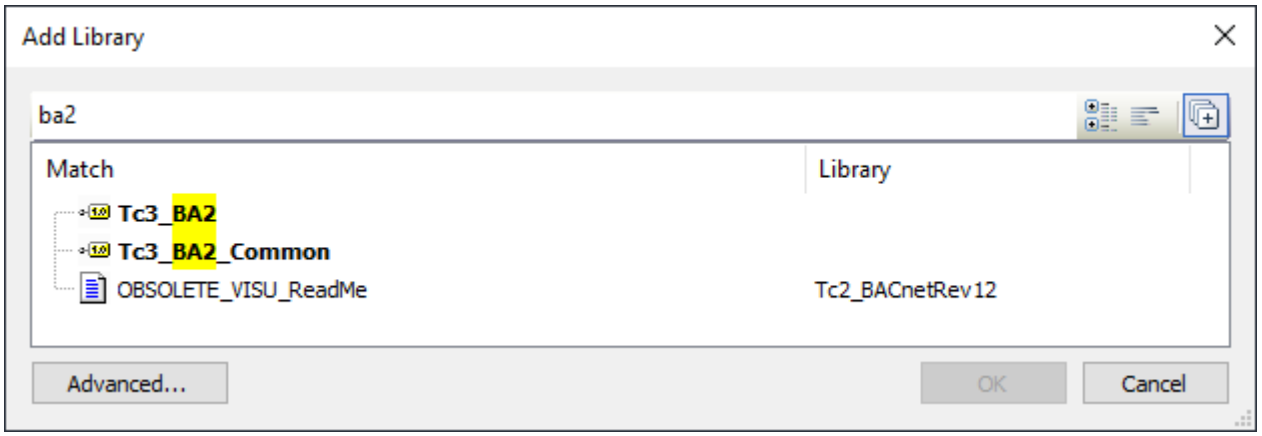
The settings described below are not necessary if a PLC project template [▶ 693] is used.

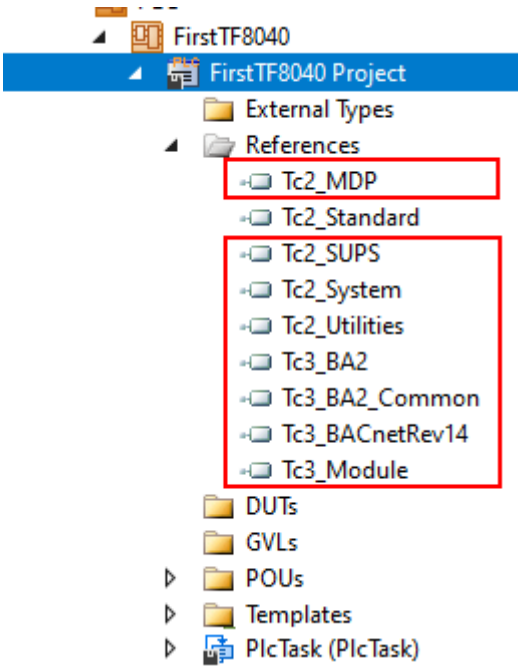
Libraries

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

- Tc3_BA2_Common
- Tc3_BACnetRev14
- Tc3_BA2







In the standard PLC-BA template [▶ 693] all necessary libraries are already loaded automatically.

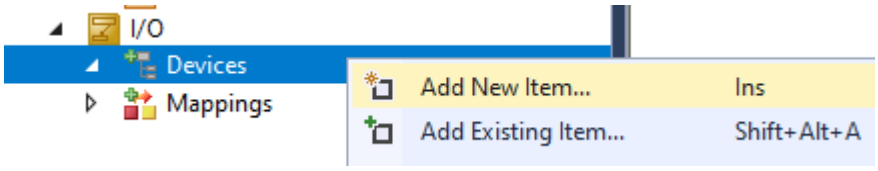
I/O

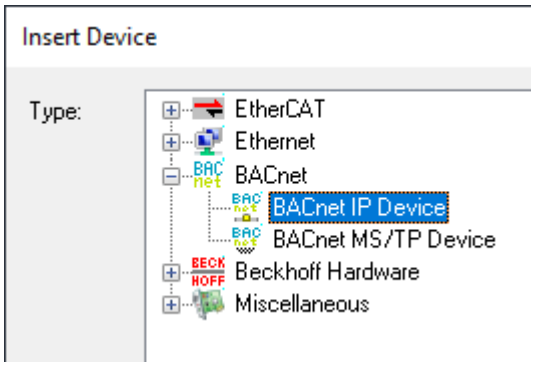
The procedure proposed here refers to the steps required to make the TF8040 function operational in combination with BACnet on the desired hardware.

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

BACNet

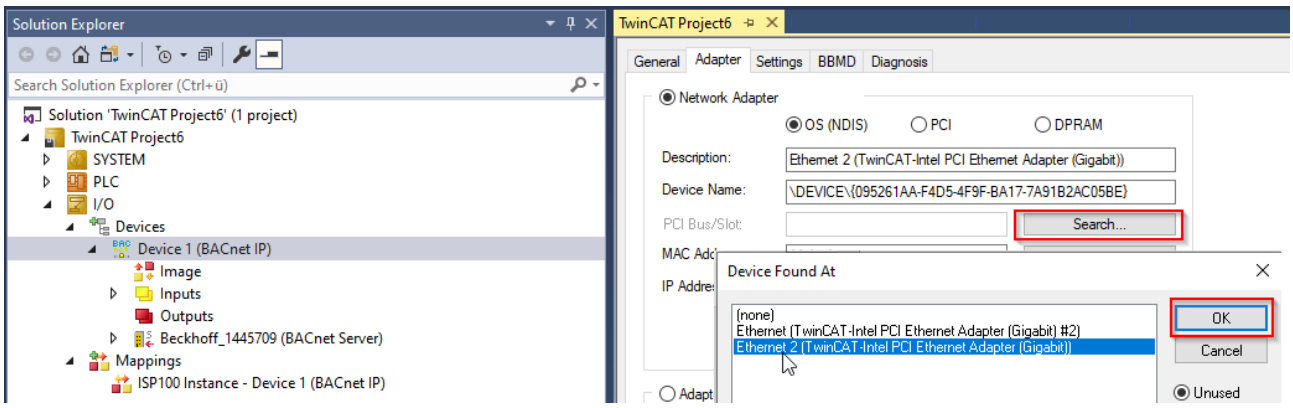
- Add BACnet device:





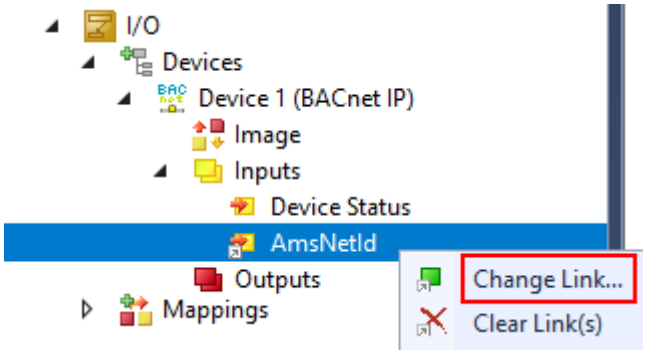
- Select the appropriate network adapter:

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

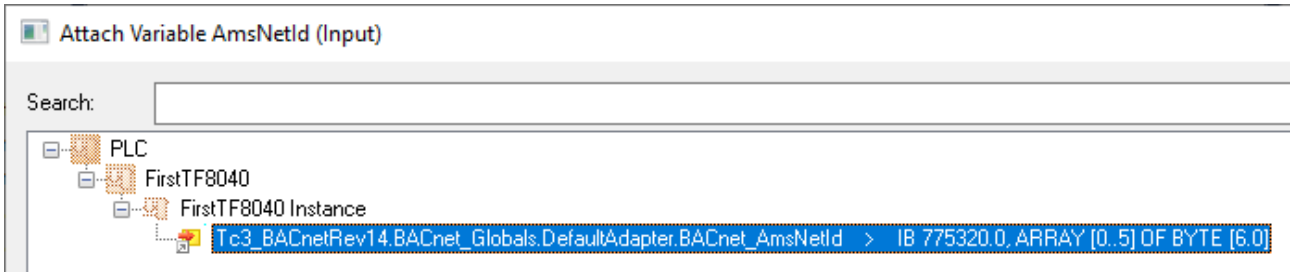


- Linking the BACnet adapter with the AMS NetID:

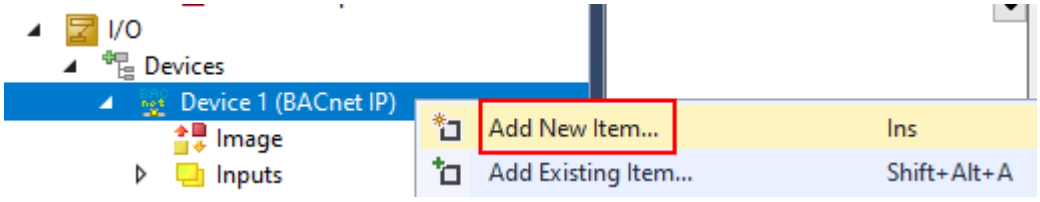
This step is only possible once the project has been compiled without errors.

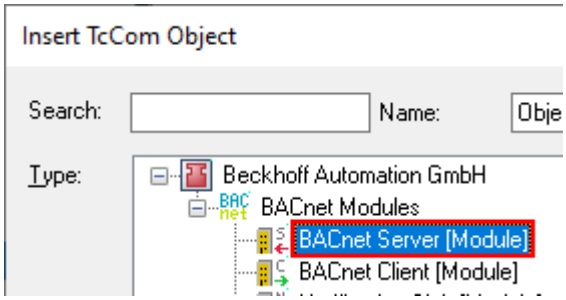


Selection of the BACnet adapter to be used:



- 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 [▶ 693] according to the recommended procedure [▶ 693].

Continue

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

4.2 HMI

Chapter with instructions for creating an HMI project.

4.2.1 Starting a project

Description

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

> By using TcHmiBa project templates [▶ 1275] individual steps can already be prepared accordingly.

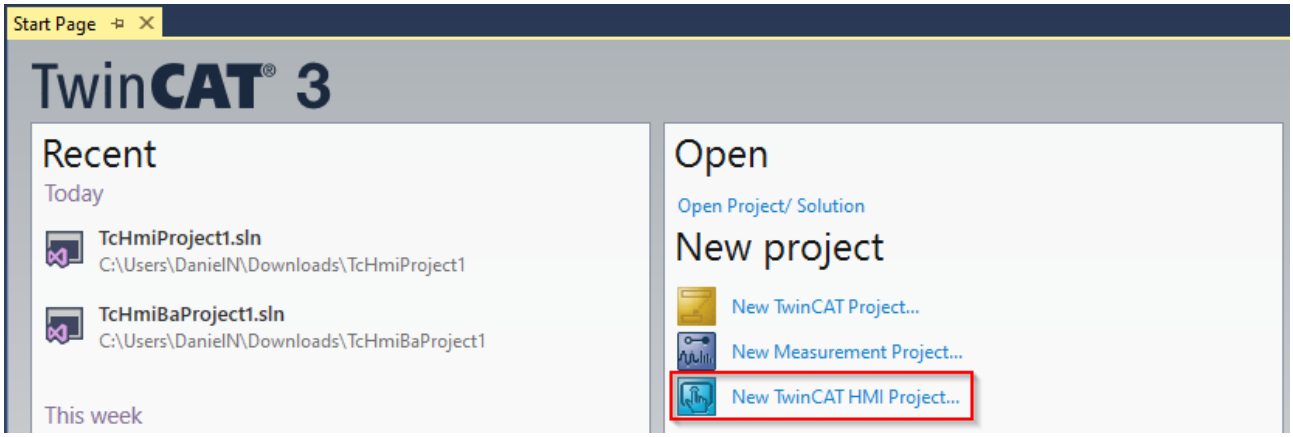
Procedure

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.).



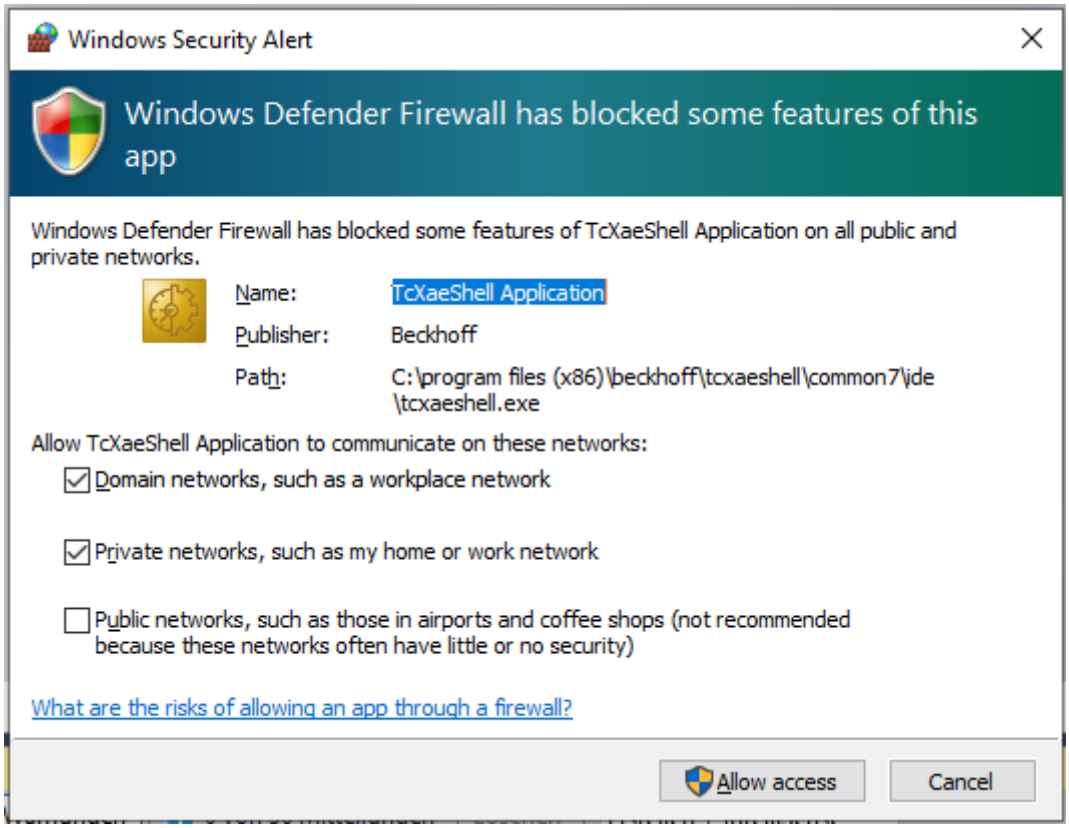
Installing TF8040

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.

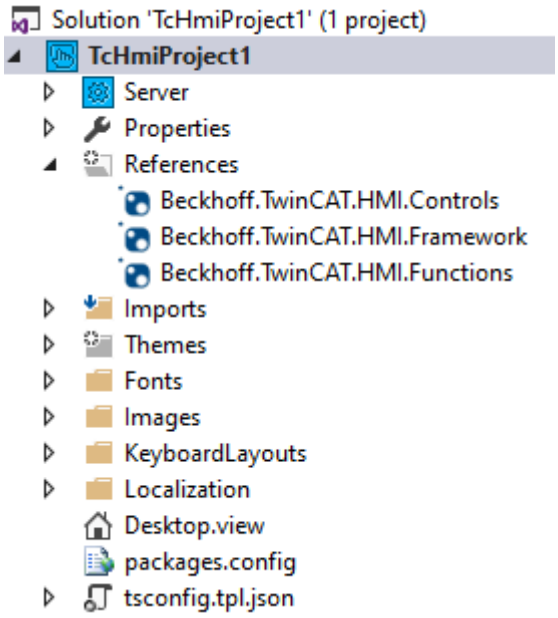
Create a new TcHmi project

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:

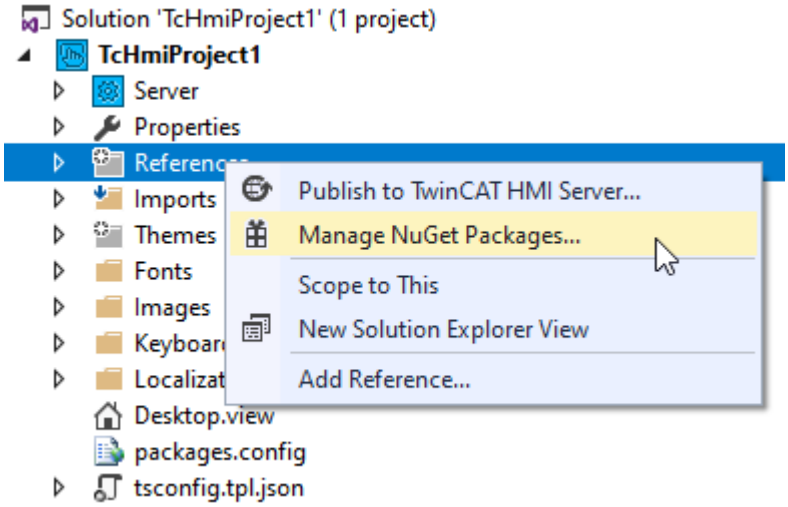


Installing NuGet packages

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

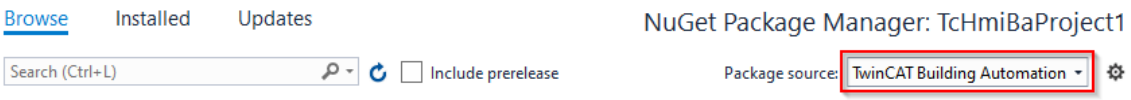
To be able to use them in the created project, the NuGet package **Beckhoff.TwinCAT.HMI.Controls** must be installed. The procedure for doing this is as follows:

- Open the NuGet package management

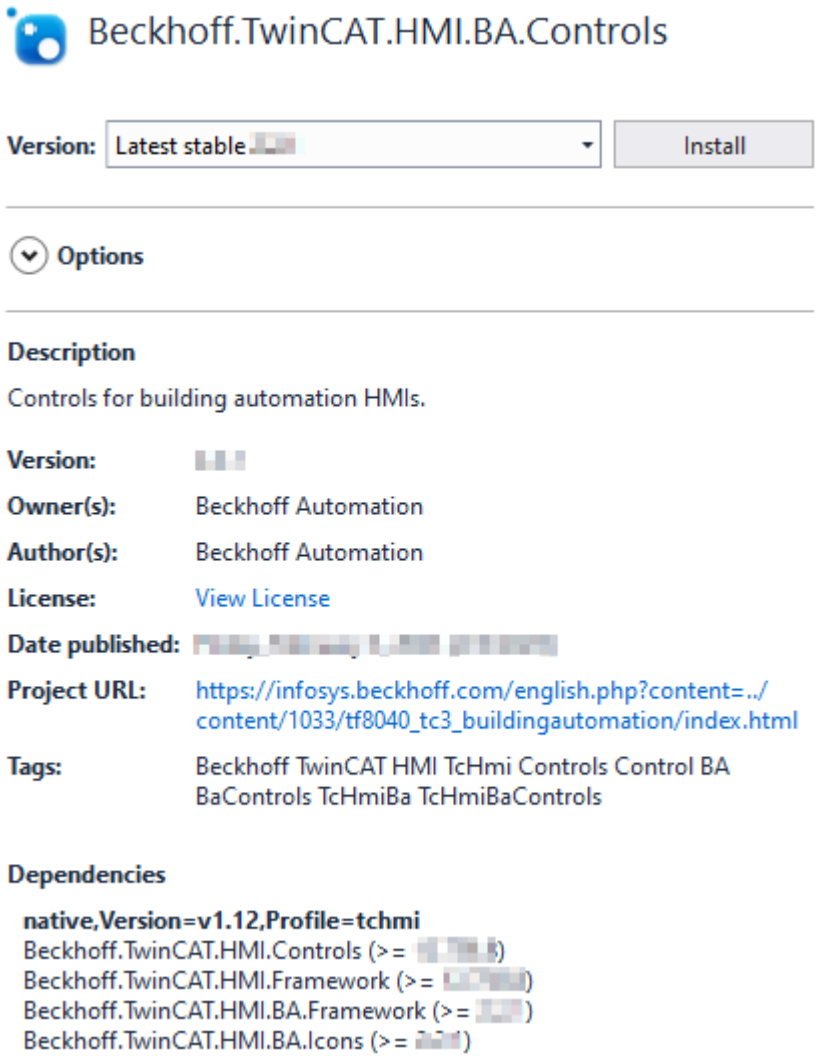


The correct package source **TwinCAT Building Automation** must be selected in the top right corner of the Package Manager.

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

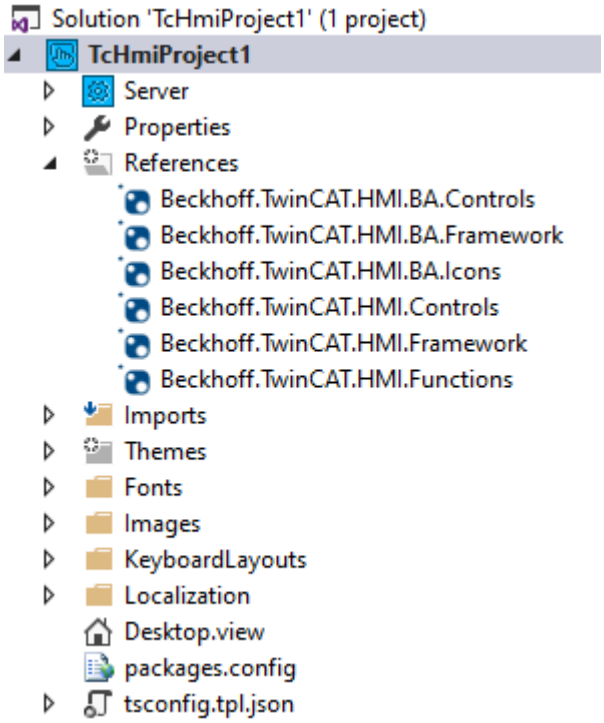


- Install **Beckhoff.TwinCAT.HMI.BA.Controls**

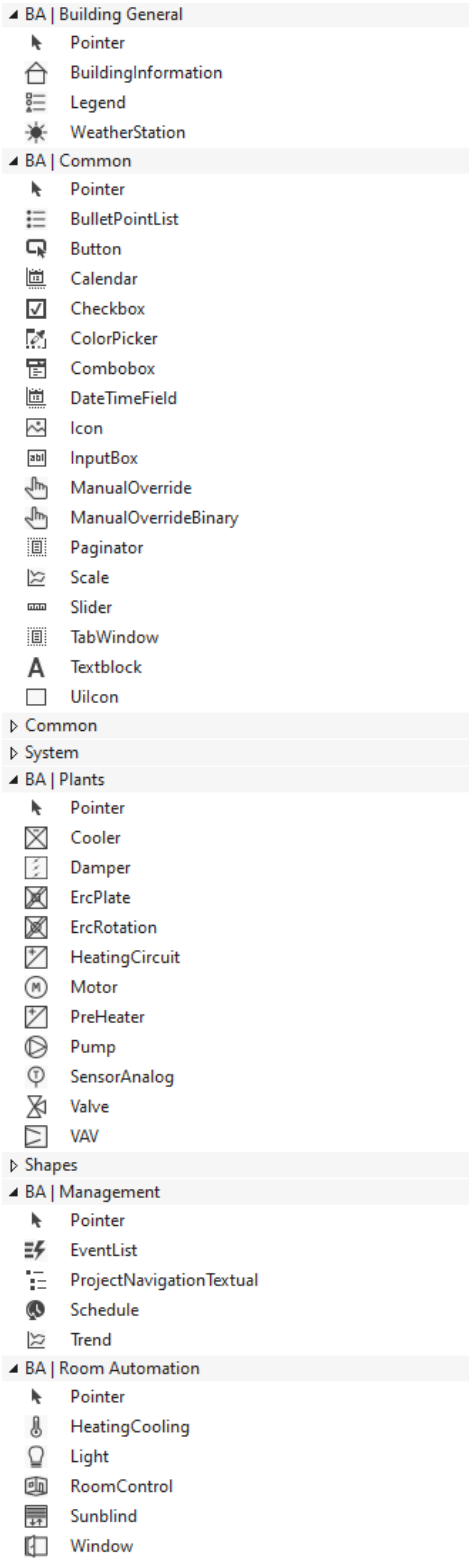


The packages Beckhoff.TwinCAT.HMI.BA.Framework [▶ 1242] and Beckhoff.TwinCAT.HMI.BA.Icons [▶ 1231] are also installed during the installation.

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:



Further information can be found in the respective documentation [▶ 1122] 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 [▶ 77].

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 [▶ 71].

Procedure

Please note the system requirements [▶ 1121] of the *BaSite-Extension*.

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

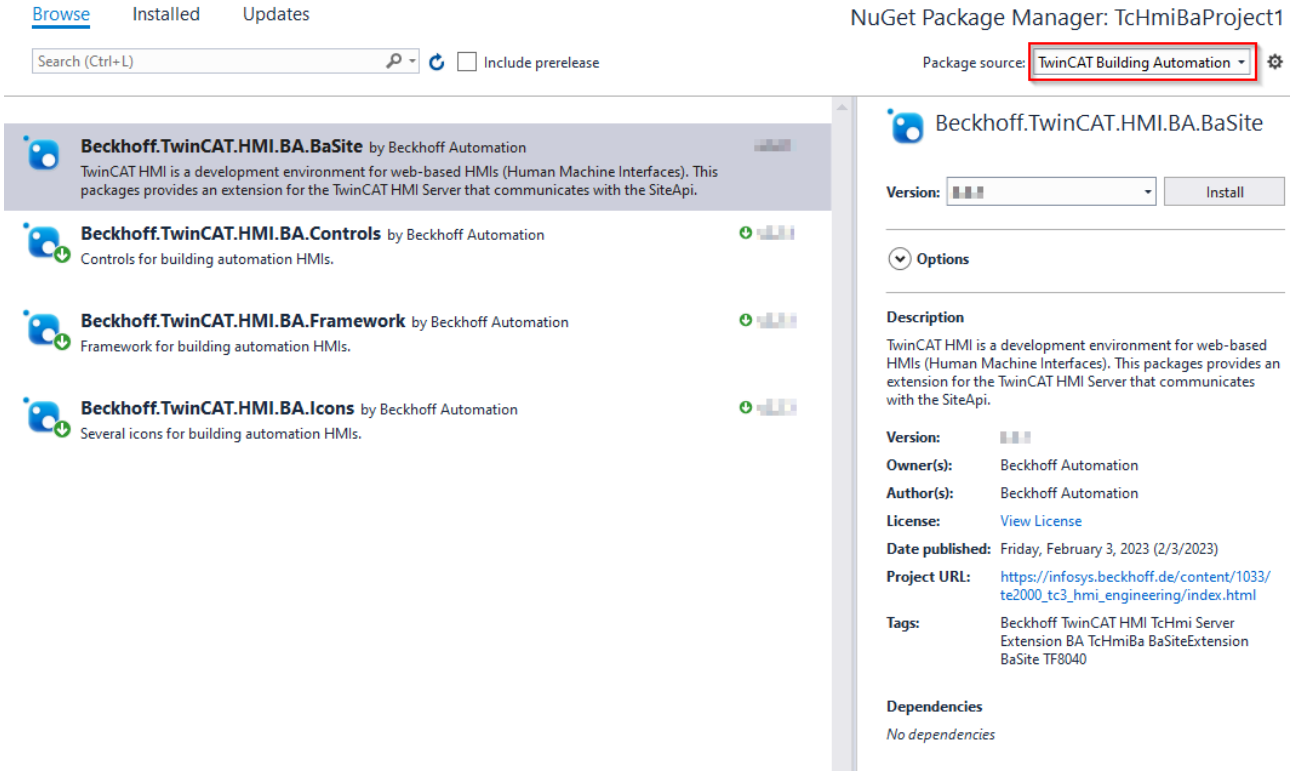
Description of required terms [▶ 32].

Preparing the server

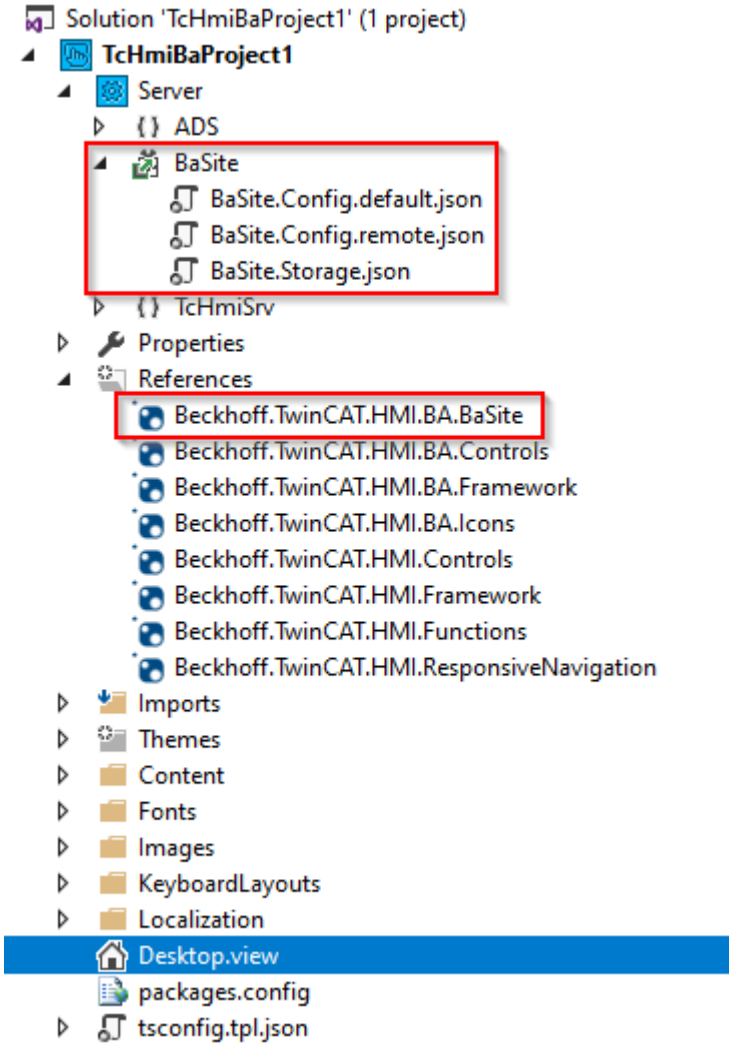
Before starting to create the visualization with generic functions, the server must be prepared.

Installation BaSite-Extension

To support the generic functionalities it is necessary to install the NuGet package Beckhoff.TwinCAT.HMI.BA.BaSite [▶ 1262].

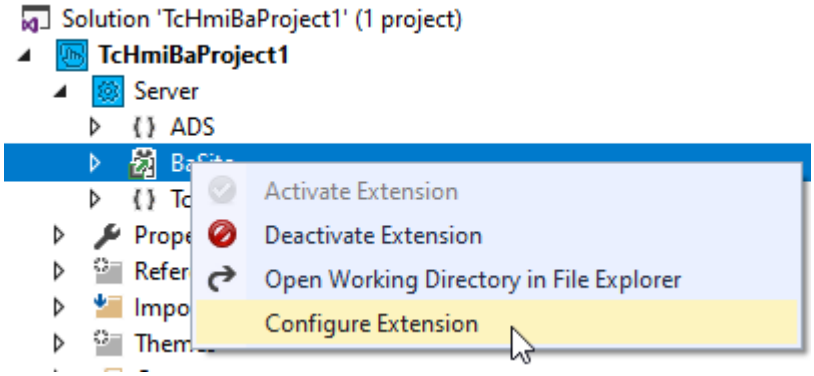


After installation the project tree should look like this:

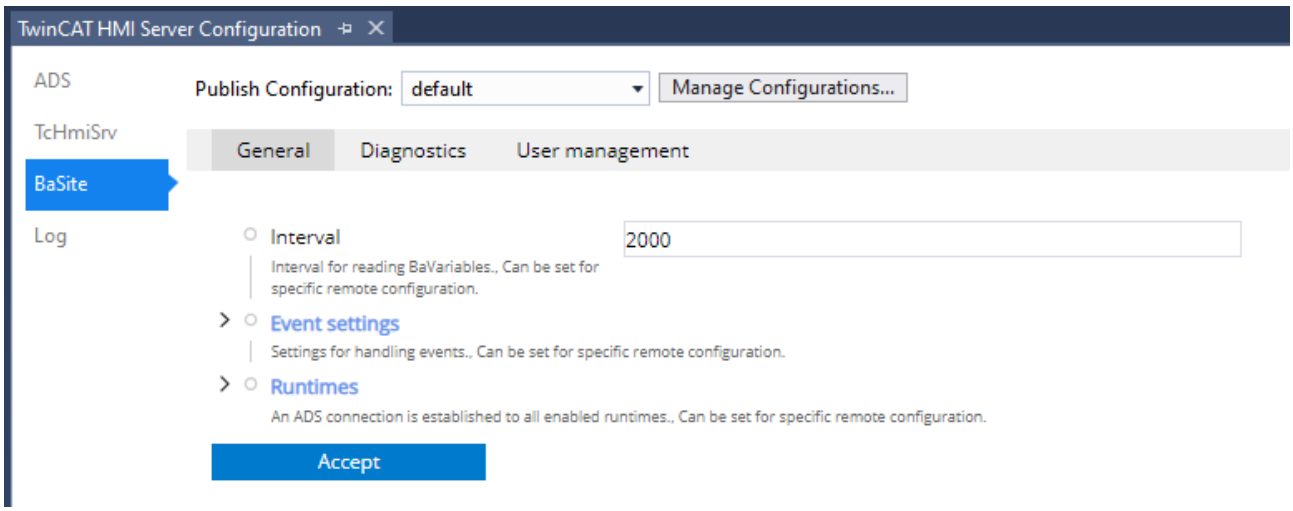


Configuration BaSite-Extension

Open the configuration page of the **BaSite-Extension**.



In the opened window all settings for the **BaSite-Extension** can be made.



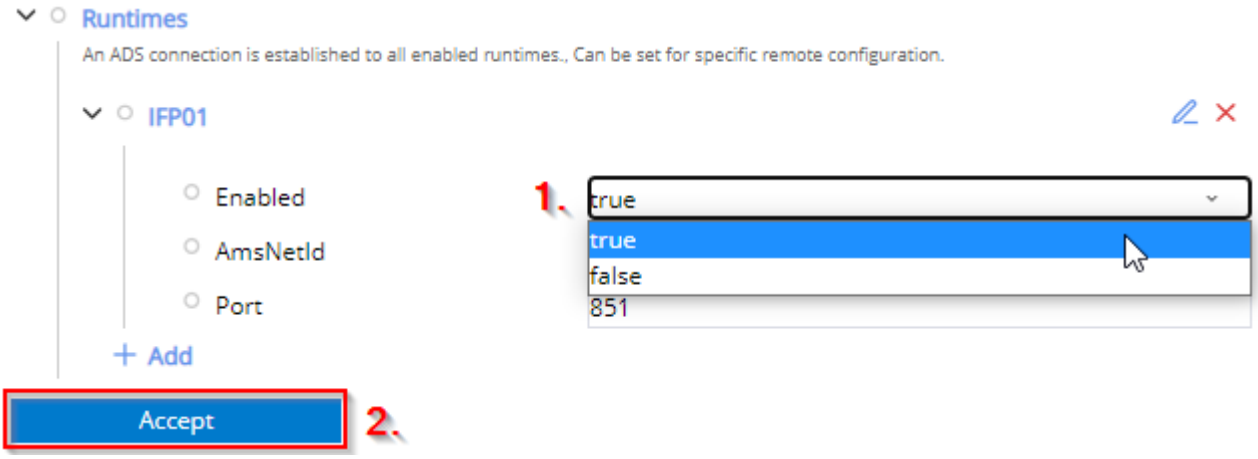
For more information about configuration and functions, see the server extension documentation [▶ 1262].

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

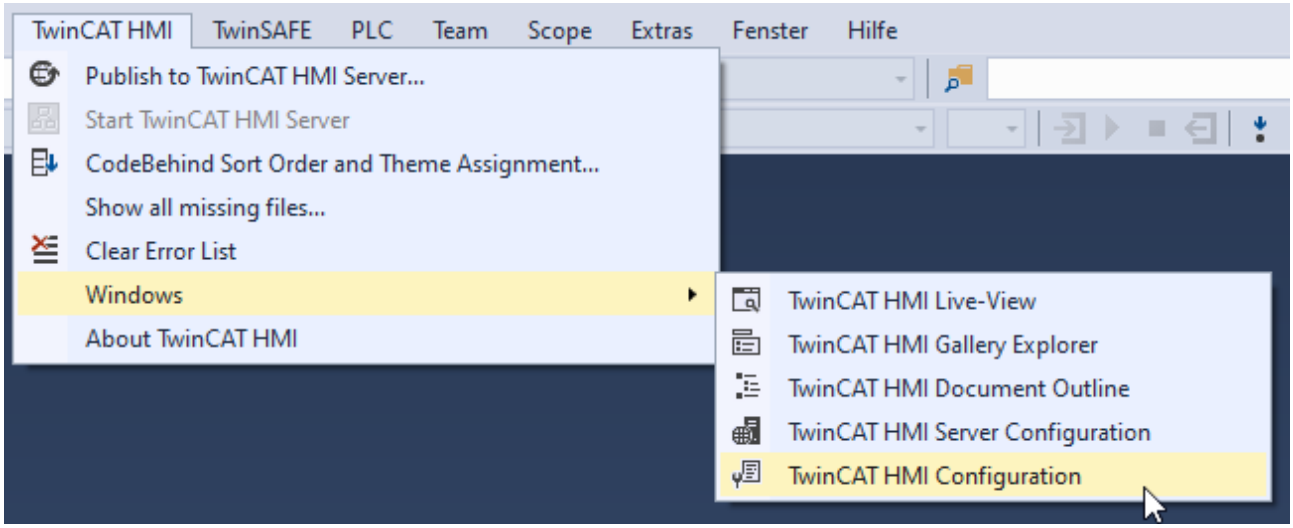
Enter the respective data for the active PLC at AmsNetId and Port.
Enter the respective details for the active PLC at **AmsNetId** and **Port**.

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

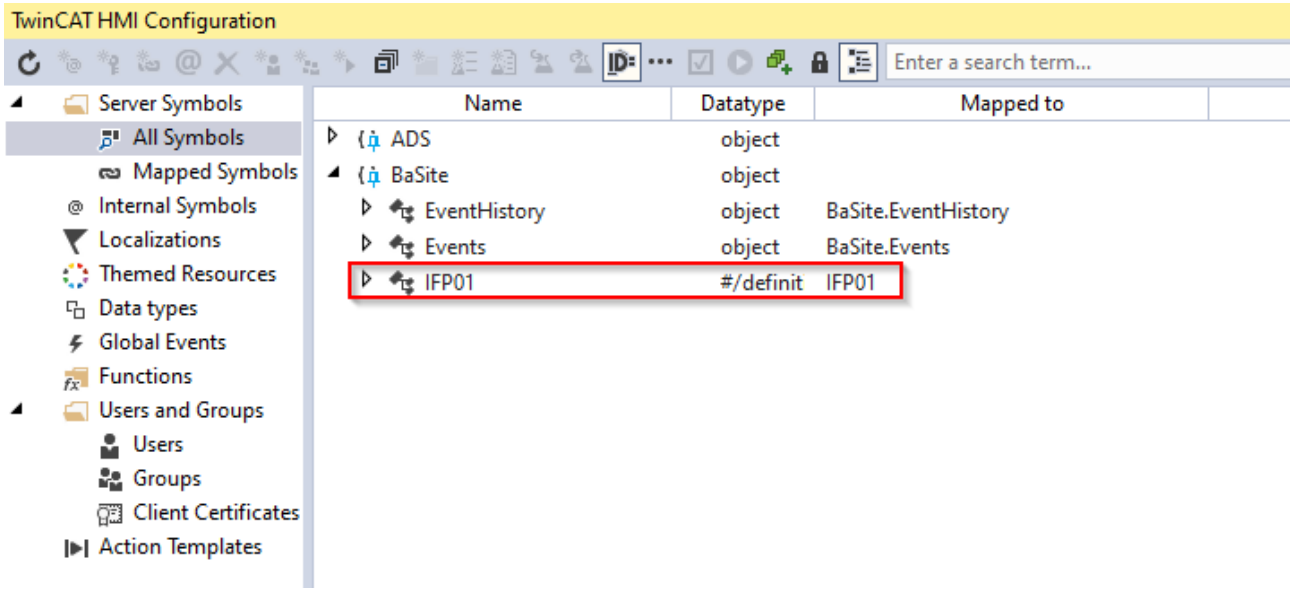
Afterwards the runtime has to be activated and the dialog has to be confirmed.



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

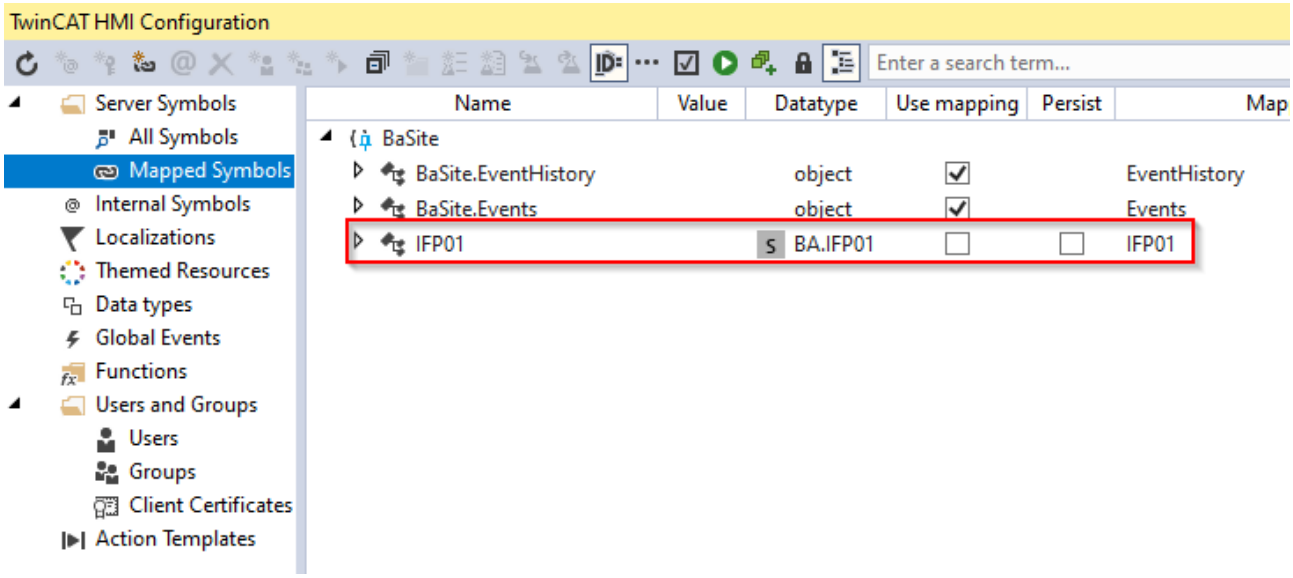


The runtime is then listed at **All Symbols** in the opened window.



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**.



Note the requirements for the necessary mappings [▶ 1264] for the generic functionalities.

The configuration of the server is now complete.

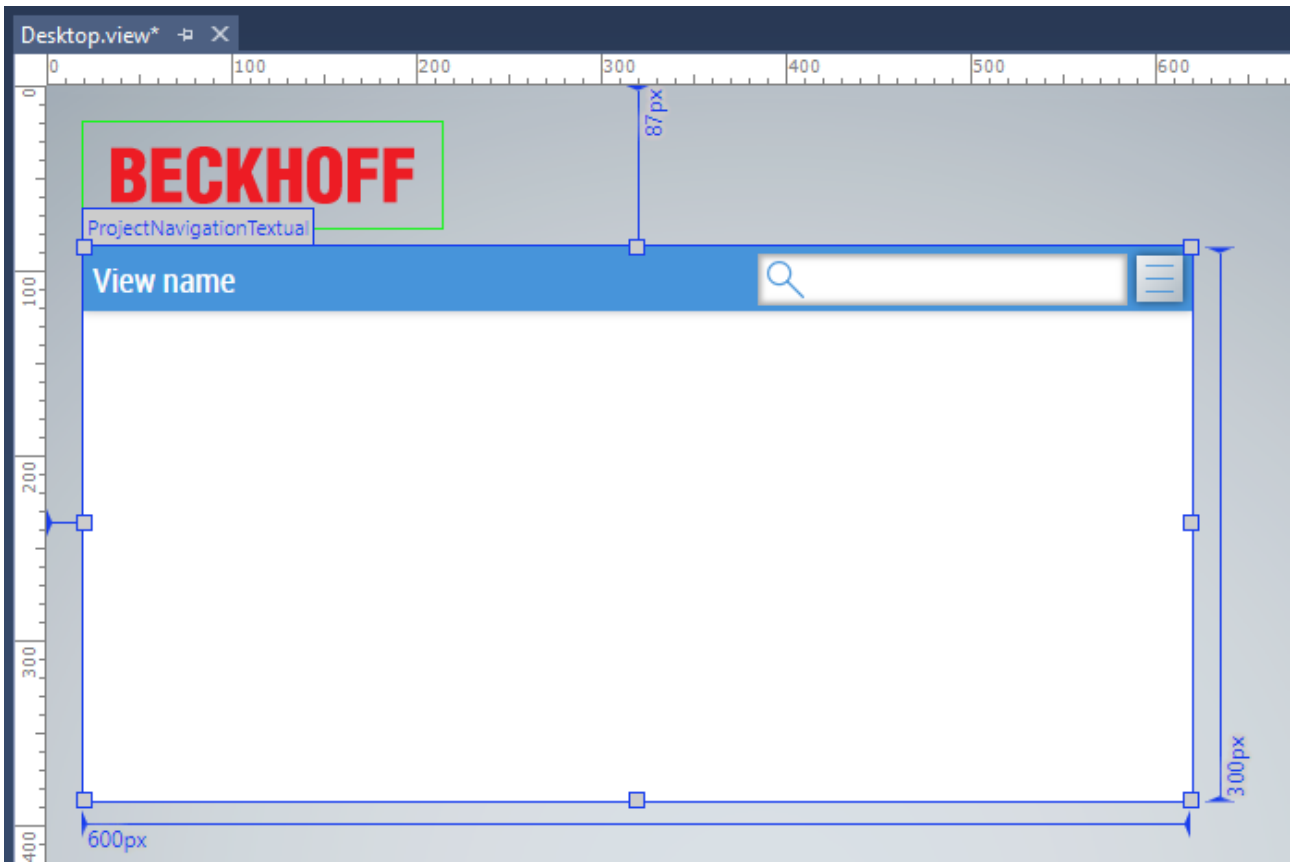
Using generic controls

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

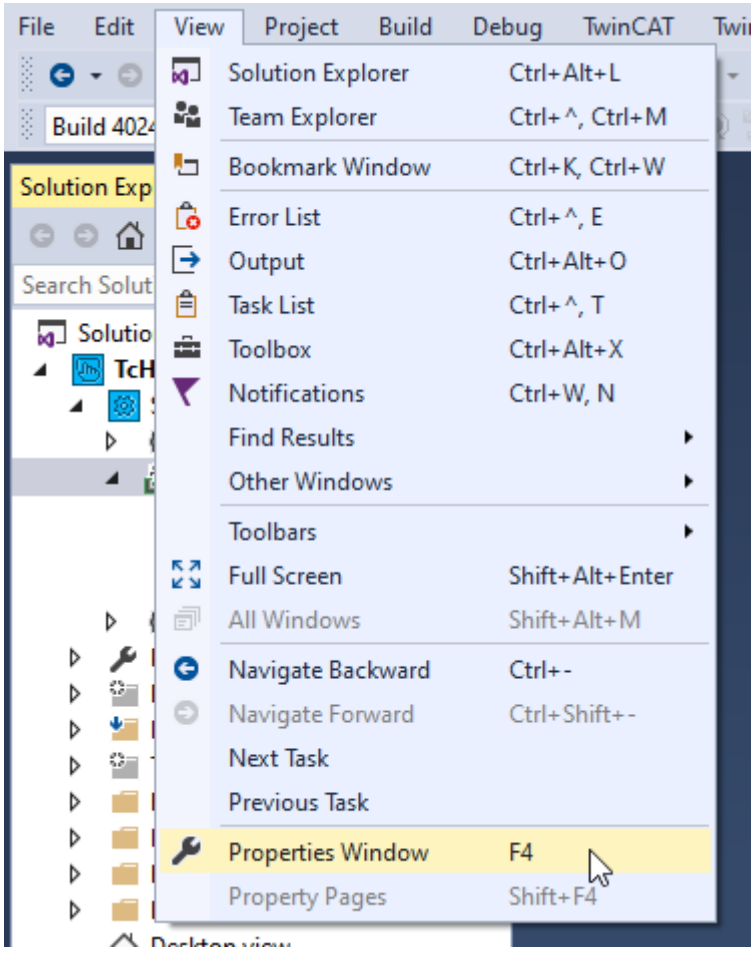
Further information can be found in the respective documentation of the individual controls.

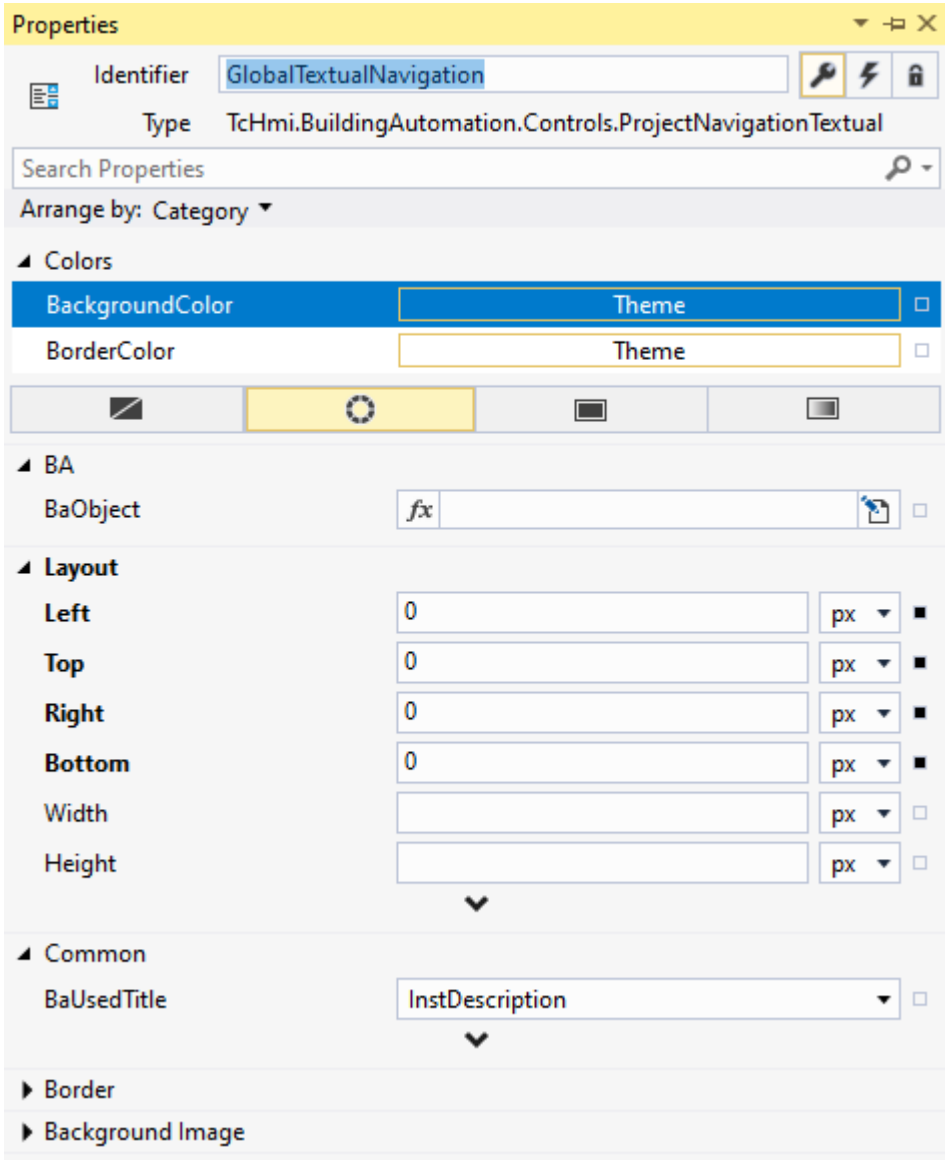
Project navigation

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:



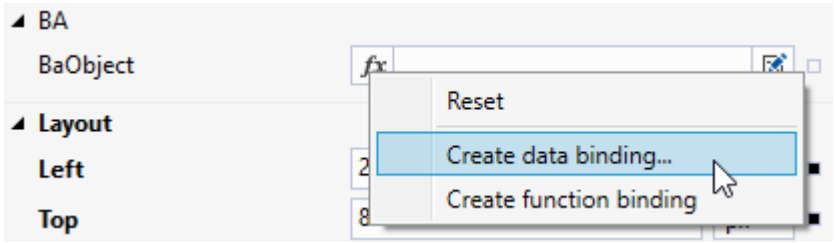
Open the Properties window.

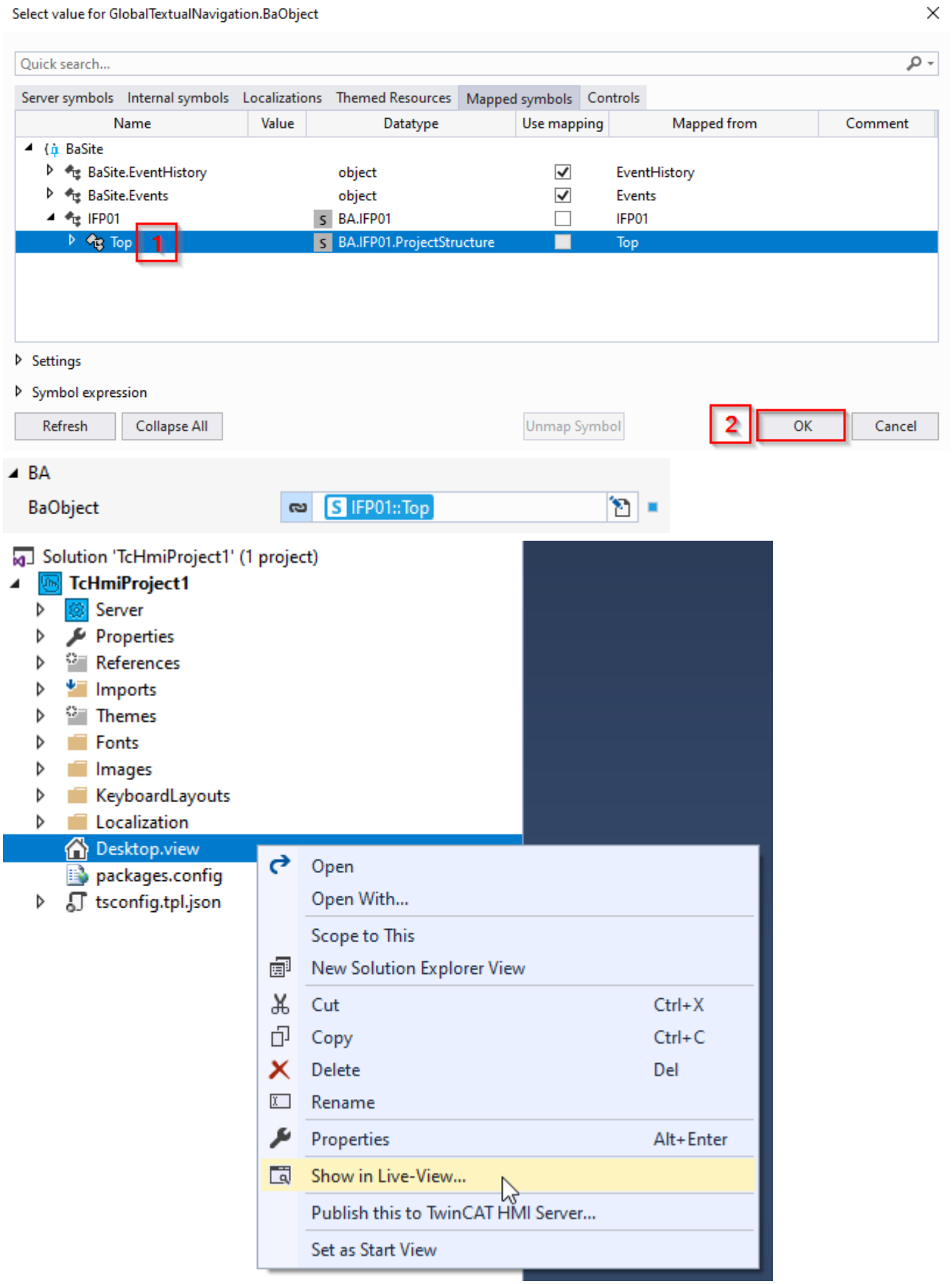




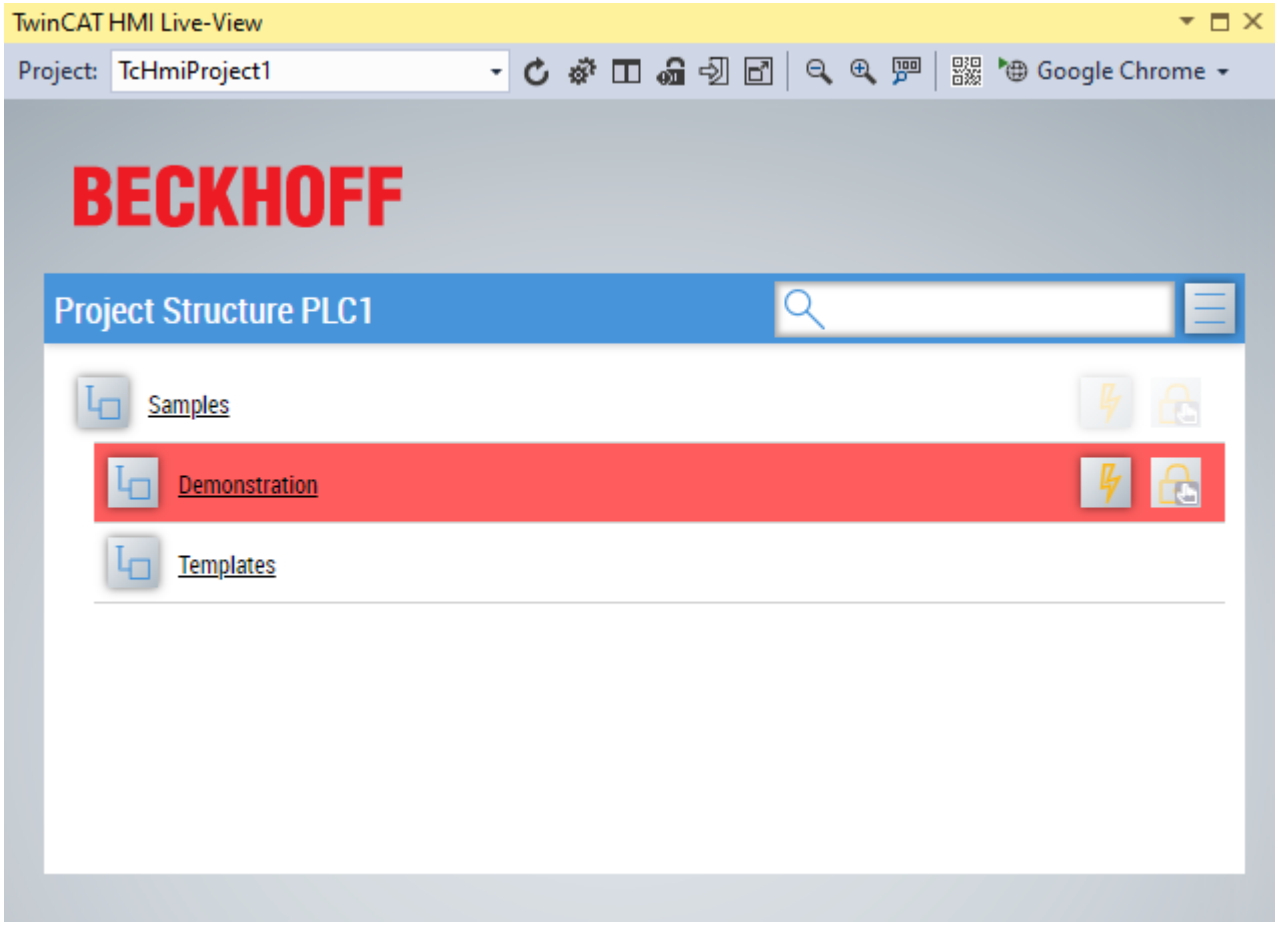
The content of the Properties window always depends on the current selection. To see the properties of the project navigation, the control must be selected on the *Desktop.view*.

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.



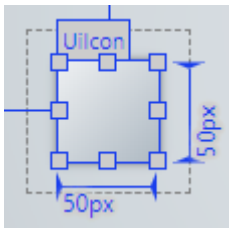


The Live-View should now look like this:



UiIcon

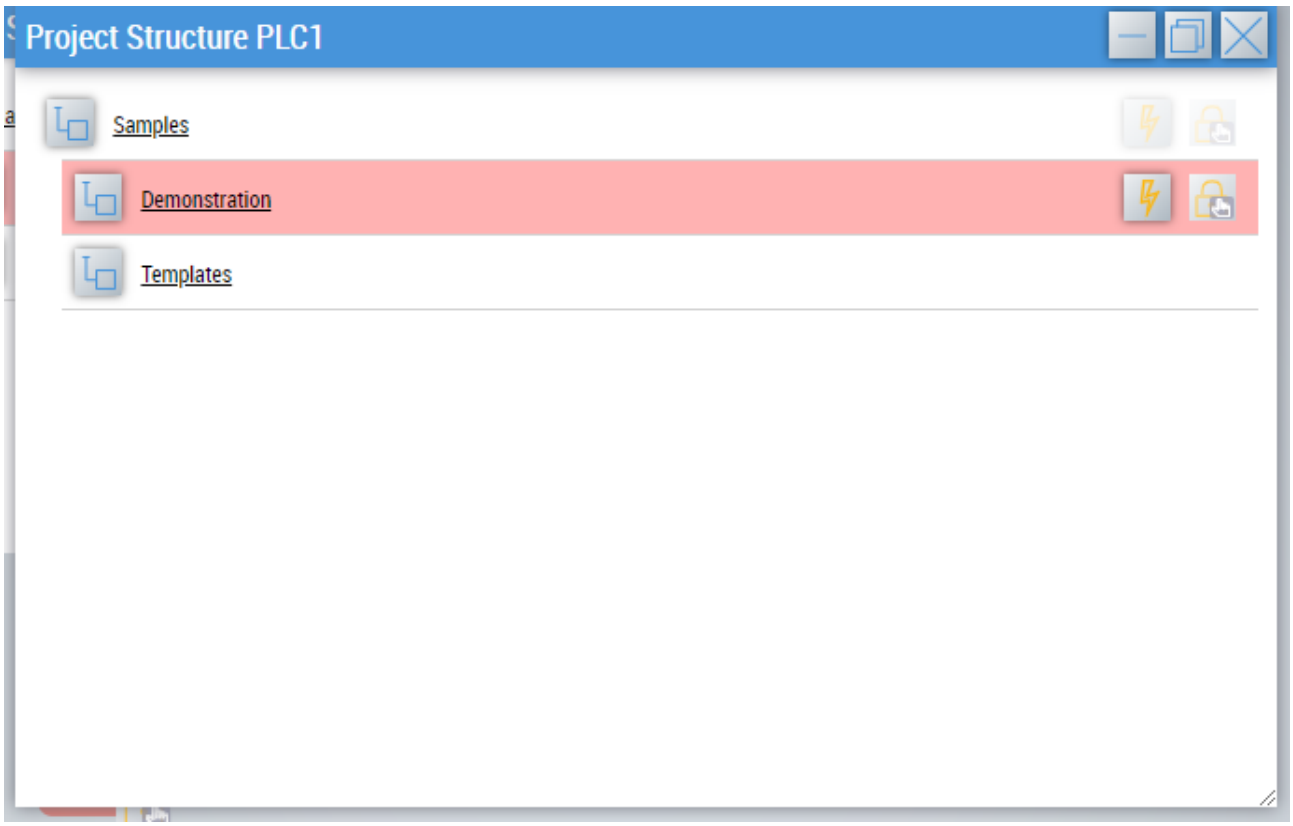
The UiIcon [▶ 1159] can be used for various applications. It is also located in the **BA | General** category of the toolbox.



Like all controls from TcHmiBa, the *UiIcon* 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 **UiIcon** 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/18733784075.zip
  Contains
  - Samples of PLC (templates [▶ 96])
  - Samples of HMI (concept [▶ 88], templates [▶ 94])

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 [▶ 64].

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 [▶ 77].

Please note the general comments.

Further information

- PLC programming [▶ 98]

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 [▶ 87].

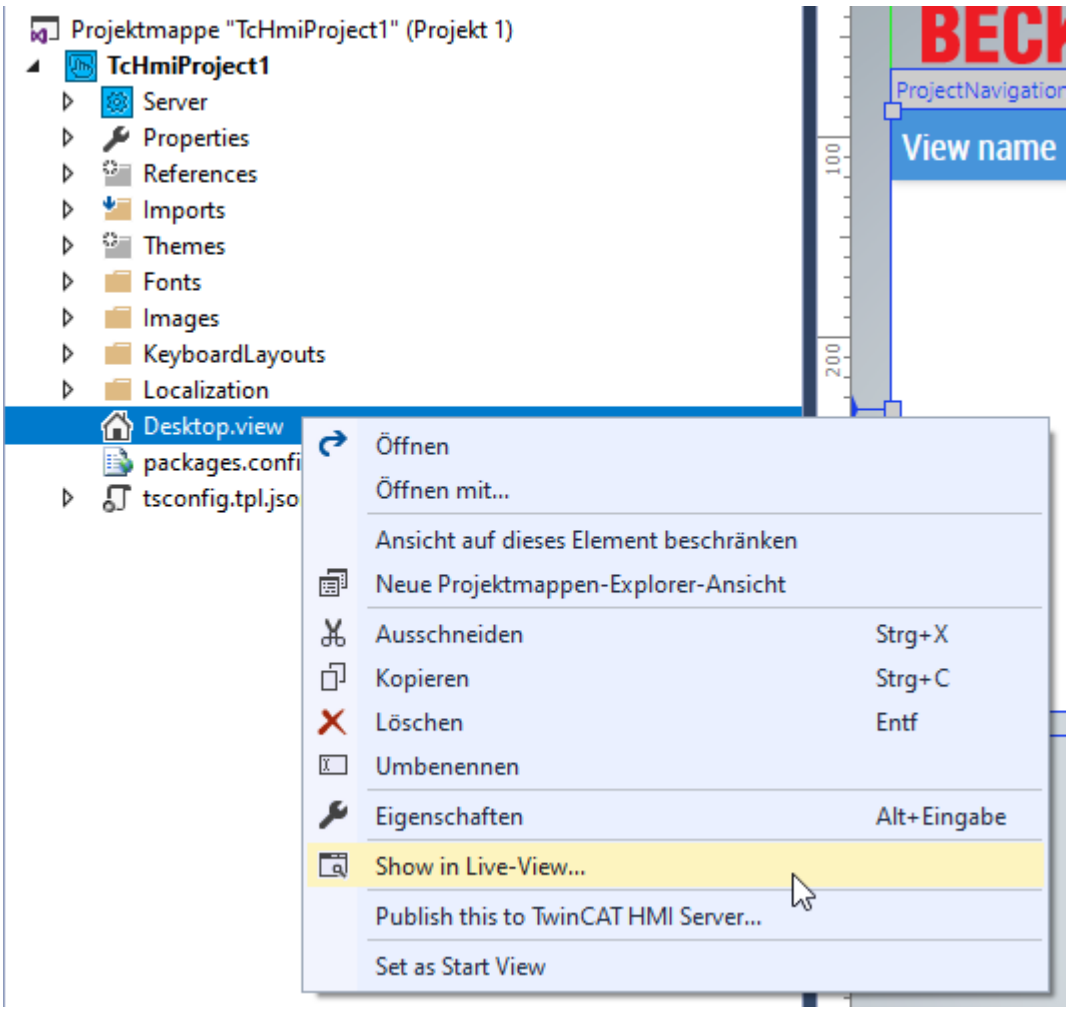
5.1.1 HMI

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

For more information on the required steps, refer to the sample documentation in the HMI [▶ 87] 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

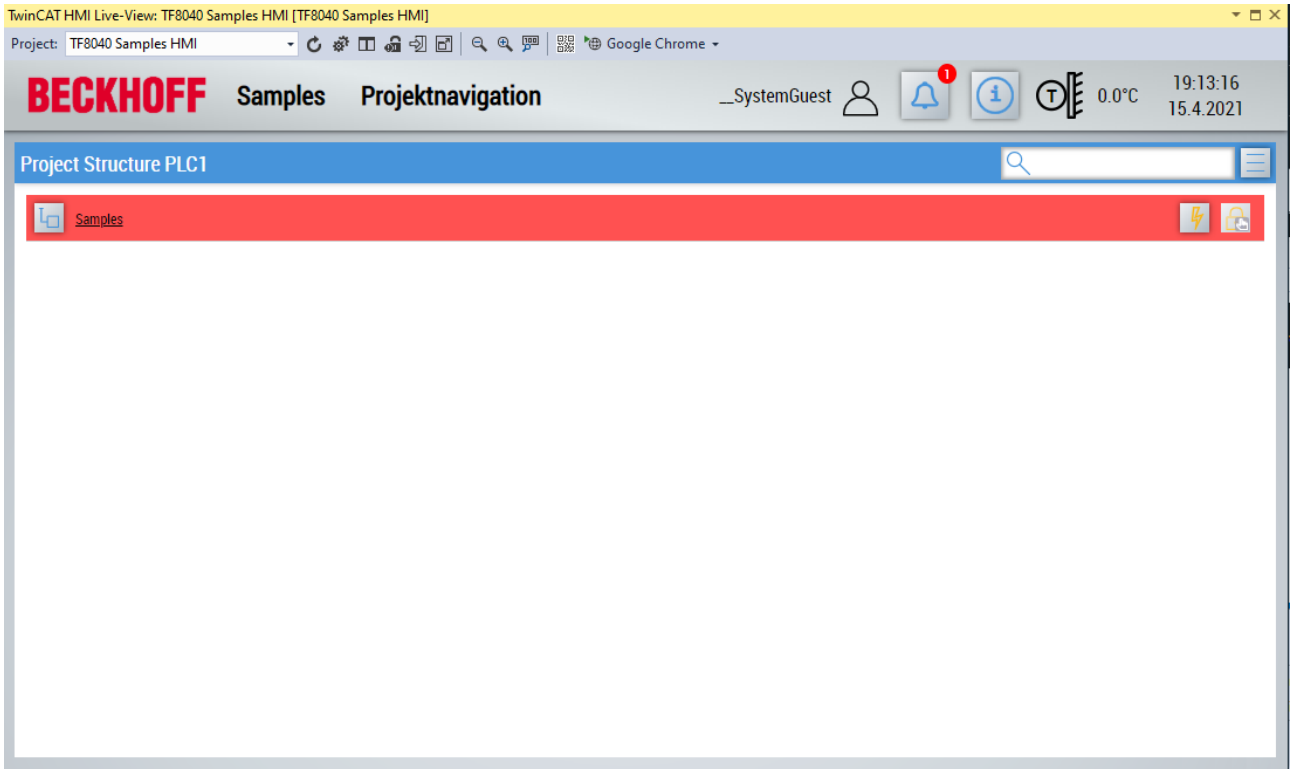
The header is provided with various functions (from left to right).

- 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 [▶ 1281].

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 [▶ 1167].

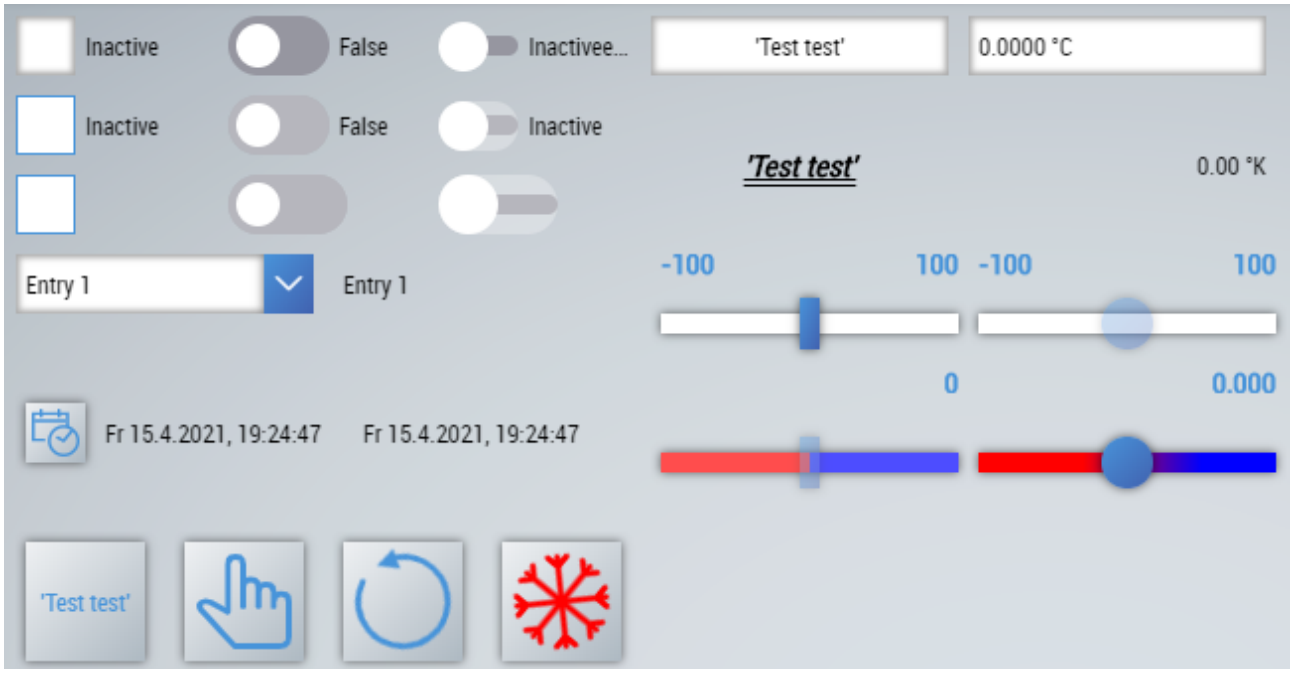
The following pages are located under the entry *Content\Samples*.

BasicComponents

This page displays all controls that are stored in the *BA | Common* toolbox category.

The controls are not linked to variables from the PLC.

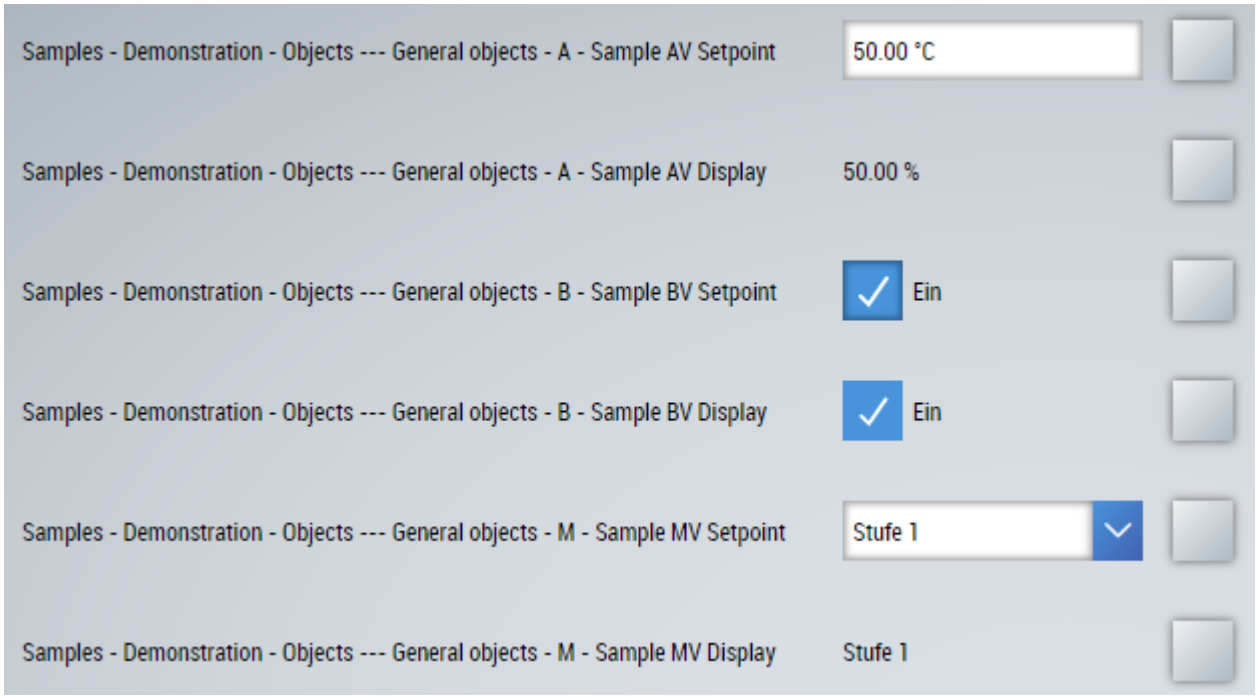
Further information can be found in the documentation for controls [▶ 1122].



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.



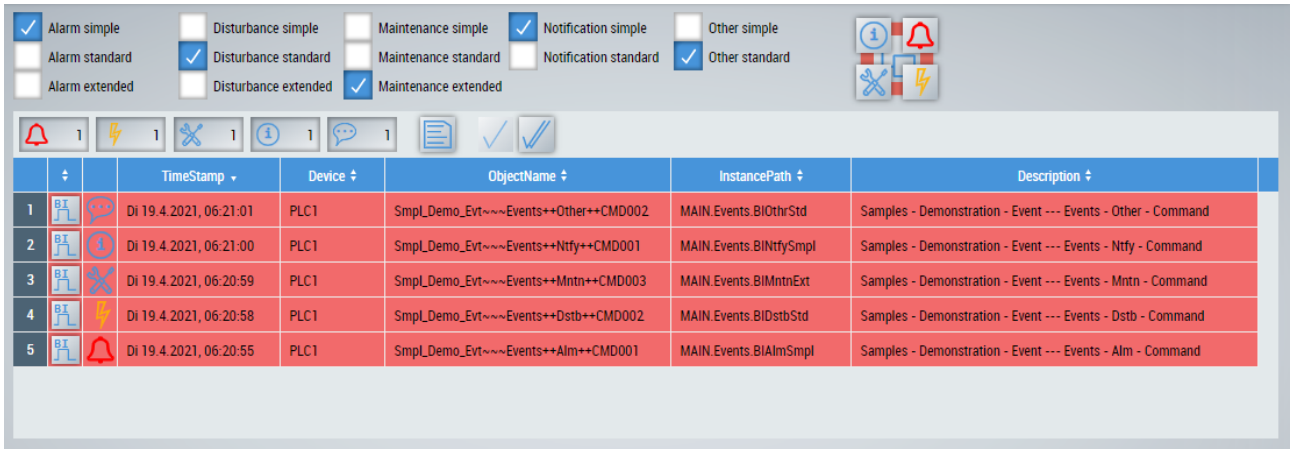
The following information is displayed per line:

- 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 *UiIcon* 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 *UiIcon*. Therefore, no events outside the view are displayed on this page.

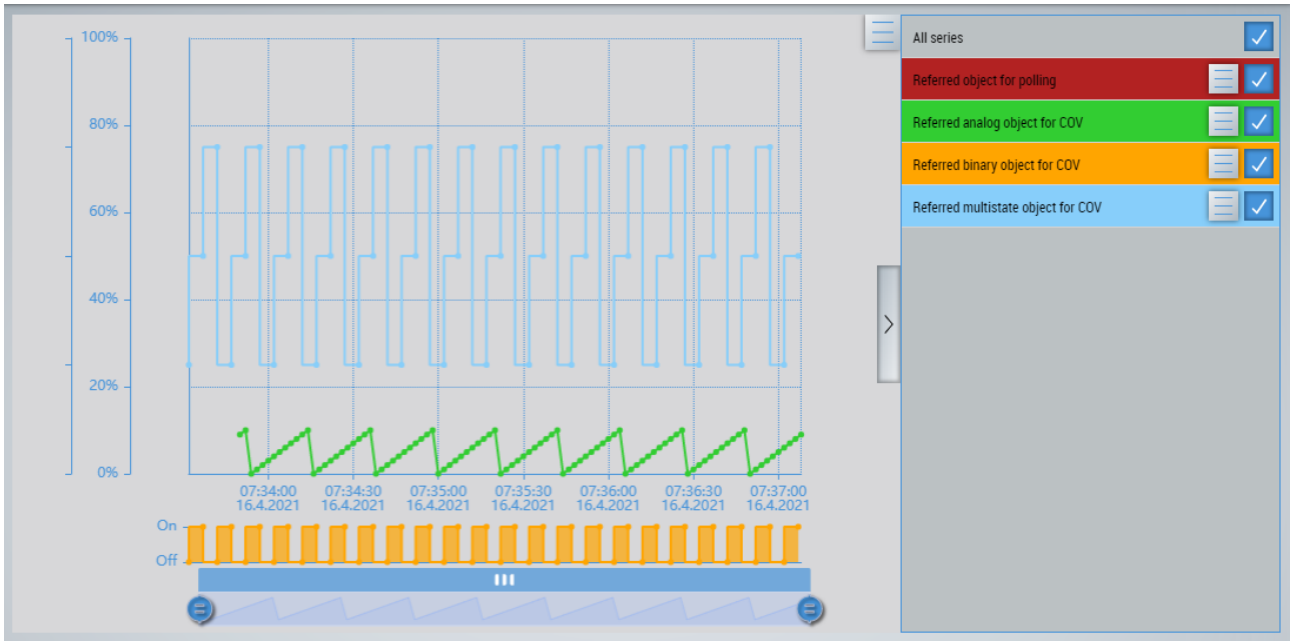


Trend

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 [▶ 1175].



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.

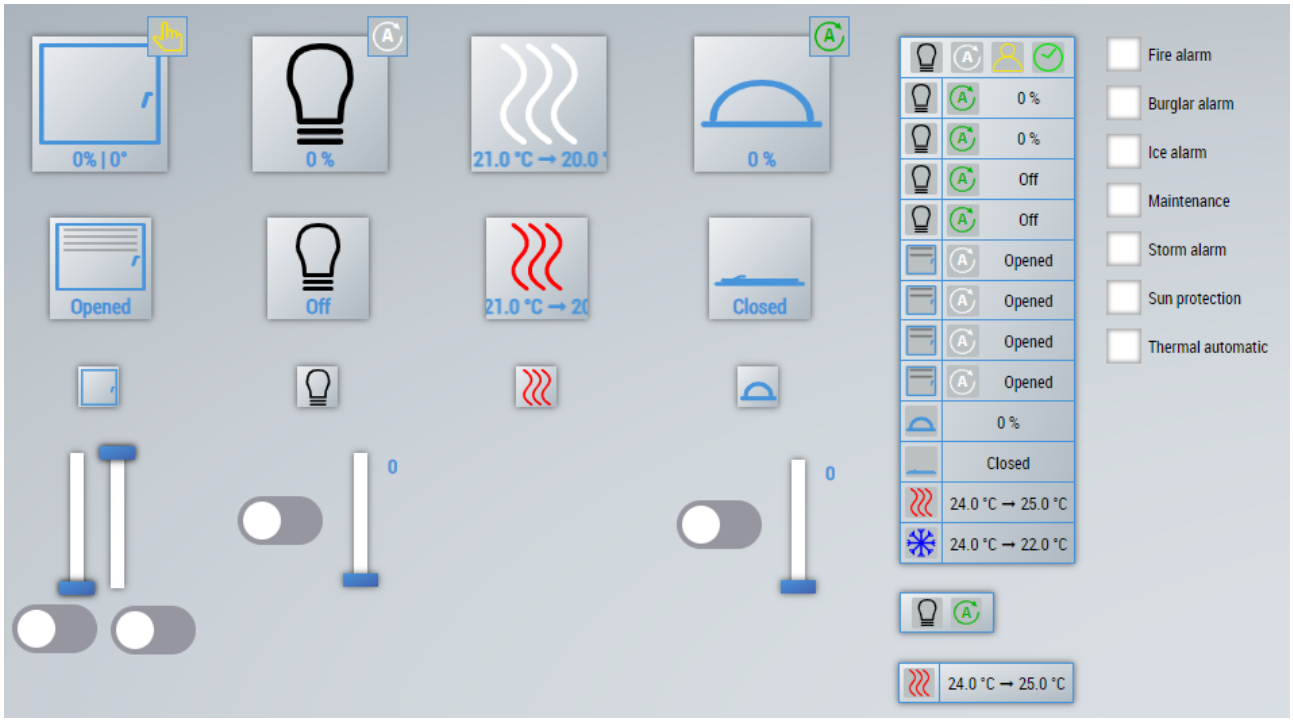
For more information, see the documentation on the Schedule [▶ 1171].

RoomAutomation

The following list shows the available controls for room automation:

Control	Description
Sunblind [▶ 1210]	Displays and controls the position and angle of a sunblind.
Light [▶ 1203]	Displays and controls the brightness value of a lamp (dimmable or on / off).
HeatingCooling [▶ 1200]	Displays and controls the air conditioning of a room.
Window [▶ 1214]	Shows and controls the position of a window (percentage or open/close).
RoomControl [▶ 1206]	This control can combine all of the above controls.

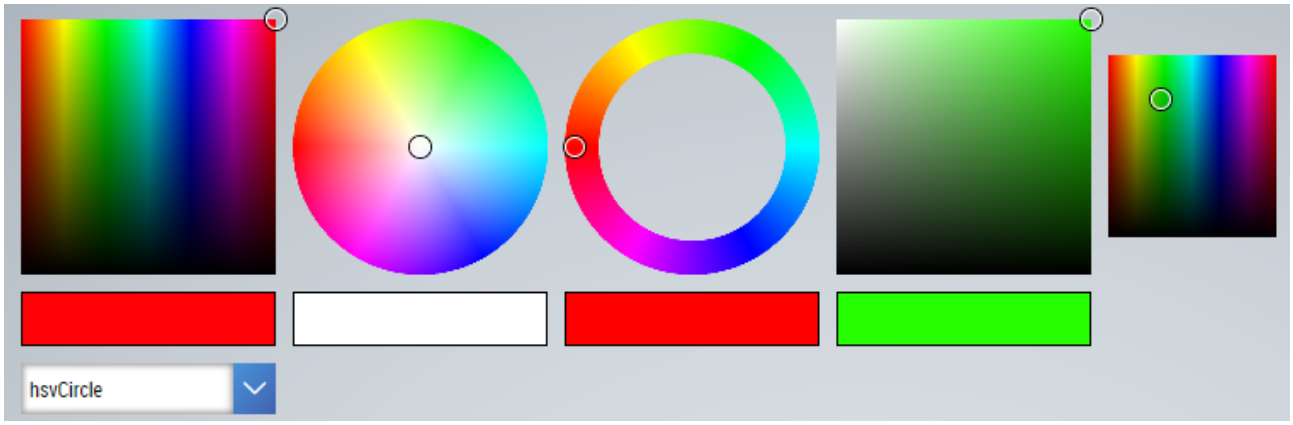
The controls are for demonstration purposes only and are therefore not linked to variables from the PLC.



ColorPicker

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

Further information can be found in the documentation for the ColorPicker [▶ 1137]



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 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 [▶ 87] 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.

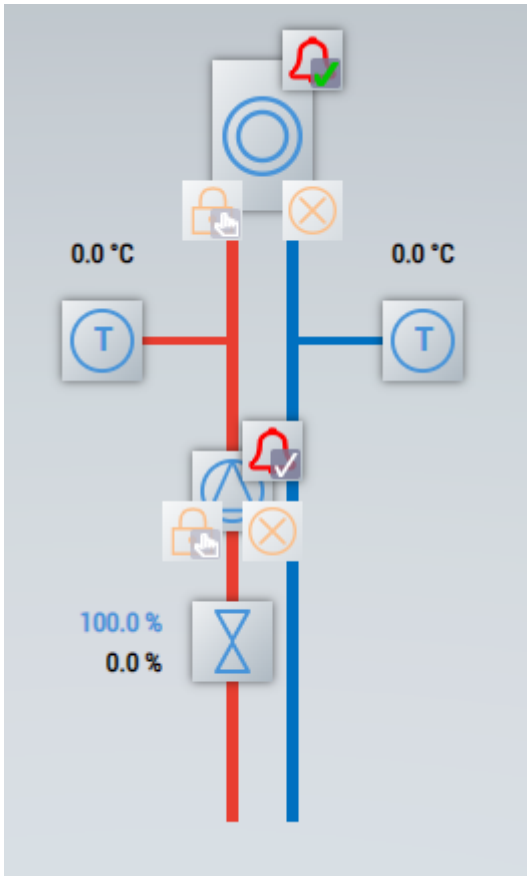
Project navigation

The generic navigation [▶ 1167] is displayed on the content of this page.

Systems

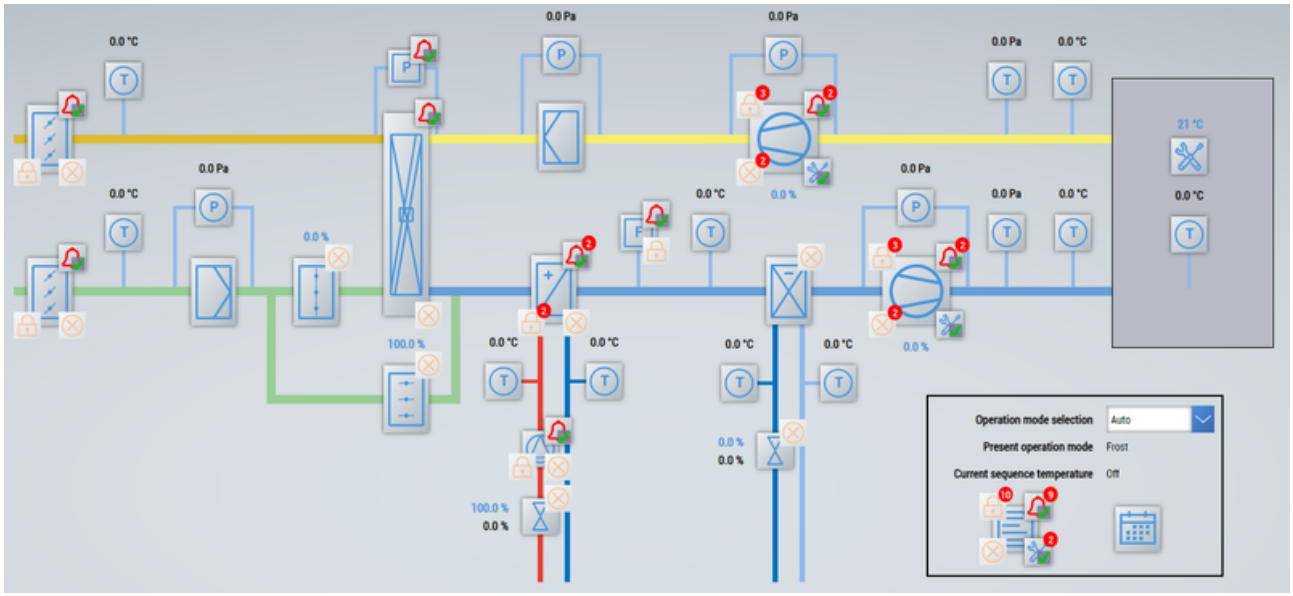
Heating circuit

Displays a heating circuit.



Ventilation and air-conditioning technology

Displays a ventilation system.



Room

Displays a sample room.

5.2.2 PLC

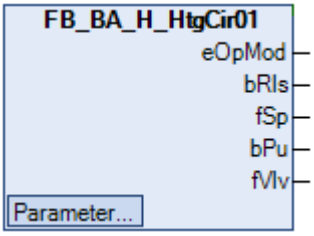
Explanation of the procedure for starting the sample project TF8040 Template Samples PLC and its contents.

Contents

The individual templates used in the project are described below.

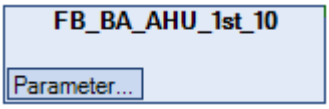
Heating circuit

Use of the template FB_BA_HtgCir01 [▶ 975].



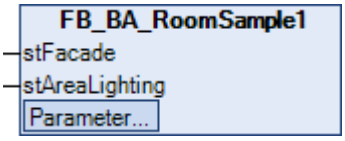
Ventilation system

Use of the template FB_BA_AHU_1st_10 [▶ 810].



Room sample

Use of the template FB_BA_RoomSample1 [▶ 931].



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 [▶ 264]. 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 *Fixed Parameters* region. However, variables are only provided for the integrator **within** the TwinCAT environment! They are not accessible from clients (e.g. Site Explorer [▶ 1285] or TcHmiBa [▶ 1120]).
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

TwinCAT Building Automation consists of several PLC libraries.

These are already available in the system installation and are also supplied with TwinCAT Building Automation:

- Tc3_BACnetRev14
  - BACnet function blocks
- Tc3_BA2_Common
  - Basic functions required by TwinCAT Building Automation as well as TF8020 BACnet.

Further libraries are exclusively included in the installation of TwinCAT Building Automation:

- Tc3_BA2 [▶ 267]
  - Application function blocks of the building automation for systems and room automation
  - basic mathematical functions
  - basic logical functions
- Tc3_XBA [▶ 99]
  - Base framework
  - Site Server for communication with the Site Explorer and the TwinCAT HMI
  - Objects

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 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

6.1.2.1.1.1.2 Communication

6.1.2.1.1.1.2.1 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.

6.1.2.1.1.1.3 Control

6.1.2.1.1.1.3.1 E_BA_StateLoopSync

E_BA_StateLoopSync shows the state of the synchronization of FB_BA_LoopSync [▶ 165].

```
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
eOk	Synchronization is error-free.
eErrAction	Synchronization of *eActionRm* faulty.
eErrDampConstant	Synchronization of *nDampConstant* faulty.
eErrDerivativeConstant	Synchronization of *fDerivativeConstant* faulty.
eErrIntegralConstant	Synchronization of *fIntegralConstant* faulty.
eErrMaxOutputRm	Synchronization of *fMaxOutputRm* faulty.
eErrMinOutputRm	Synchronization of *fMinOutputRm* faulty.
eErrNeutralZone	Synchronization of *fErrNeutralZone* faulty.
eErrOpMode	Synchronization of *eOpMode* faulty.
eErrProportionalConstant	Synchronization of *fProportionalConstant* faulty.

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 Events

6.1.2.1.1.1.5.1 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.

6.1.2.1.1.1.5.2 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 | x | x EVENT_ENABLED x | x | x According to BACnet, any combination of the AcknowledgeRequired bits (0 or 1) can be used. The EventNotification can be either an event or an alarm Acknowledgement: for each state transition (TO-OFFNORMAL, TO-NORMAL and TO-FAULT) it can be defined whether an acknowledgement is necessary or not.
eSimple	Neither incoming nor outgoing alarms need to be acknowledged.
eStandard	Only incoming, but not outgoing alarms need to be acknowledged. Acknowledgement but no reset of the alarm is required.
eExtended	Both acknowledgement and reset of the alarm are requested. Depending on the acknowledgement mode, a simple acknowledgement may be sufficient to trigger the reset.

6.1.2.1.1.1.5.3 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
```

6.1.2.1.1.1.5.4 E_BA_EventIconState

```
TYPE E_BA_EventIconState :
(
  Invalid          := 0,
  eNone            := 1,
  eIndicated       := 2,
  eGoneAcked       := 3,
  eGone            := 4,
  ePresentAcked    := 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*.
eGoneAcked	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*.
ePesentAcked	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!

6.1.2.1.1.1.5.5 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

6.1.2.1.1.1.5.6 E_BA_ObjectStateFlags

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

6.1.2.1.1.1.6 Groups

6.1.2.1.1.1.6.1 E_BA_AValCalcMode

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

6.1.2.1.1.1.6.2 E_BA_BValCalcMode

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

6.1.2.1.1.1.6.3 E_BA_MValCalcMode

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

6.1.2.1.1.1.7 Objects

6.1.2.1.1.1.7.1 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, ...)

6.1.2.1.1.1.7.2 Loop

6.1.2.1.1.1.7.2.1 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.

6.1.2.1.1.1.7.2.2 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.

6.1.2.1.1.1.7.2.3 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.

6.1.2.1.1.1.7.3 PlantCtrl

6.1.2.1.1.1.7.3.1 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.

6.1.2.1.1.1.8 Parameters

6.1.2.1.1.1.8.1 E_BA_Attribute

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

6.1.2.1.1.1.8.2 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

6.1.2.1.1.1.8.3 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.

6.1.2.1.1.1.8.4 E_BA_Parameter

Possible properties of a BACnet object.

```
TYPE E_BA_Parameter :
(
  Invalid                   := 0,
  eConfigurate              := 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
```

6.1.2.1.1.1.8.5 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

6.1.2.1.1.1.8.6 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.

6.1.2.1.1.1.9 Priority

6.1.2.1.1.1.9.1 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 [▶ 135], 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 [▶ 135], 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 [▶ 135] 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 [▶ 135] 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.

6.1.2.1.1.1.9.2 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	

6.1.2.1.1.1.10 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 Types

6.1.2.1.1.1.12.1 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.

6.1.2.1.1.1.12.2 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
```

6.1.2.1.1.1.13 Functional

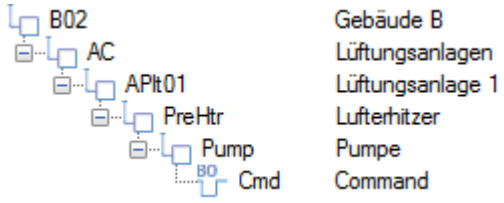
6.1.2.1.1.1.13.1 E_BA_NodeTypeTarget

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.

6.1.2.1.1.1.13.2 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
```

6.1.2.1.1.1.13.3 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
```

6.1.2.1.1.1.13.4 E_BA_ProcessSignalSource

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

6.1.2.1.1.1.13.5 E_BA_SubscriberState

```
TYPE E_BA_SubscriberState :
(
  Invalid              := 0,

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

```
) BYTE;
END_TYPE
```

6.1.2.1.1.13.6 E_BA_InitState

The enumeration describes the state of the initialization.

Syntax

```
{attribute 'qualified_only'}
TYPE E_BA_InitState :
(
  Invalid                    := 0,

  ePreInitInstance           := 1,
  eInitInstanceParam         := 2,
  eInitInstanceDependency    := 3,
  ePostInitInstance          := 4,

  eInitComDevice             := 5,
  ePreInitSupplement         := 6,
  ePostInitSupplementInst    := 7,
  ePostInitSupplement        := 8,

  ePreFirstExecute           := 9,
  ePostFirstExecute          := 10

) BYTE;
END_TYPE
```

Name	Description
ePreInitInstance	Pre-initialization of the instance.
eInitInstanceParam	Initialization of the instance parameters.
eInitInstanceDependency	Initialization of instance dependencies.
ePostInitInstance	Post-initialization of the instance.
eInitComDevice	Initialization of the ComDevice.
ePreInitSupplement	Pre-initialization of supplements.
ePostInitSupplementInst	Completion of the initialization of an instance part of a supplement.
ePostInitSupplement	Initialization of supplements is complete.
ePreFirstExecute	Before the first execution.
ePostFirstExecute	After the first execution.

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 [▶ 102]	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 S
TRING(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;
  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

The ST_BA_ObjectIdentifier structure contains the instance ID of the object types.

```
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

The ST_BA_SchedCalendar structure describes a calendar reference from an exception (ExceptionSchedule).

```
TYPE ST_BA_SchedCalendar :
STRUCT
  iRefCalendar          : I_BA_Object;

  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

The ST_BA_SchedException structure describes a calendar set of exceptions (ExceptionSchedule).

```
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

The type declaration combines individual exceptions into an array for the exception scheduler.

```
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

The type declaration combines individual exceptions into an array for the exception scheduler.

```
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

The type declaration combines individual exceptions into an array for the exception scheduler.

```
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

The type declaration combines individual scheduler entries into an array for a weekly scheduler.

```
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
```

6.1.2.1.1.2.10 Universal

6.1.2.1.1.2.10.1 Sequence

6.1.2.1.1.2.10.1.1 ST_BA_SeqLink

Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [▶ 168] and the function block FB_BA_SequenceLinkBase [▶ 171].

```
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 [▶ 124]	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 *bEn* of the FB_BA_SequenceLinkBase

6.1.2.1.1.2.10.1.2 ST_BA_SeqLinkData

This structure contains the data and commands of the individual FB_BA_SeqCtrl [▶ 168] sequence controllers.

```
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.
bIsActvSeqCtrl	BOOL	Active sequence controller

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'}
      nInstId_Auto                  : UDINT := BACnet_Globals.nBACnetInstId_Auto;
      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#FFFFFFFF;
    {endregion}
    {region 'Text generation'}
      sPlaceholderSign_Open         : STRING(1) := '{';
      sPlaceholderSign_Close        : STRING(1) := '}';
      sPlaceholderSign_Delimiter    : STRING(1) := ',';
      sPlaceholderSign_DenyConcat   : STRING(1) := '!';
      sPlaceholder_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 diagnosis 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 [▶ 32] have changed, which usually change only rarely.
nIncObjActivePriority	UINT	Indicates that the active priority of an object [▶ 32] has changed.
nIncObjStatus	UINT	Indicates that the status of an object [▶ 32] has changed.
nIncEvent	UINT	Indicates the change of an event [▶ 32].
nIncEventConfig	UINT	Indicates that the event configuration [▶ 32] of an object [▶ 32] has changed.
nInstId_Auto	UDINT	Indicates that a valid instance ID must be generated automatically.
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 for placeholder attributes.
sPlaceholderSign_DenyConcat	STRING(1)	Characters for placeholders not to be concatenated.
sPlaceholder_Empty	STRING(2)	Empty placeholders.
stUndefinedIdentifier	ST_BA_ObjectIdentifier [▶ 118]	Value representing an undefined object [▶ 32] reference.
aAckFlags_None	T_BA_EventTransitions [▶ 115]	Value, which represents no active AcknowledgeRequired flags.
aIndFilter_None	T_BA_EventConditionFlags [▶ 115]	Value, after which no event conditions [▶ 102] are filtered.
aIndFilter_EvtAll	T_BA_EventConditionFlags [▶ 115]	Value, to filter all event conditions [▶ 102].
aIndFilter_Evt	T_BA_EventConditionFlags [▶ 115]	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 'Sequence'}
      nMaxSeqCtrl                         : USINT := 8
      {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 *Site Server* 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 [▶ 42].
nDPAD_DefIndexDigits	UINT	Sets the number of digits for indexing levels in DPAD [▶ 42].
sDPAD_ObjectName_DefSeparator	T_BA_ShortString [▶ 120]	Sets the separator for separating object names.
eDPAD_ObjectName_ManOvrConcatMode	E_BA_ConcatDPADMode [▶ 100]	Default to concatenate strings when an object name has been manually overwritten.
sDPAD_Description_DefSeparator	T_BA_ShortString [▶ 120]	Sets the separator for separating descriptions.
eDPAD_Description_ManOvrConcatMode	E_BA_ConcatDPADMode [▶ 100]	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 *Description*. • FALSE displays the index automatically if > 1. • TRUE displays the index only if it is explicitly defined (placeholder).
eEvtMgmt_AckMode	E_BA_AcknowledgeMode [▶ 101]	Behavior when acknowledging events [▶ 32].
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 [▶ 115]	Defines the transitions [▶ 32] to be confirmed for the "Simple" alarm mode.
aAckFlags_Standard	T_BA_EventTransitions [▶ 115]	Defines the transitions [▶ 32] to be confirmed for the "Standard" alarm mode.
aAckFlags_Extended	T_BA_EventTransitions [▶ 115]	Defines the transitions [▶ 32] to be confirmed for the "Advanced" alarm mode.
aEventEn_Simple	T_BA_EventTransitions [▶ 115]	Defines transition states [▶ 32] to be taken into account for the "Simple" alarm mode.
aEventEn_Standard	T_BA_EventTransitions [▶ 115]	Defines transition states [▶ 32] to be taken into account for the "Standard" alarm mode.
aEventEn_Extended	T_BA_EventTransitions [▶ 115]	Defines transition states [▶ 32] 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 [▶ 115]	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 [▶ 108]	Possibility to customize default access rights for read access to parameters.
aDefWriteAccess	E_BA_Parameter [▶ 108]	Possibility to customize default access rights for write access to parameters.
nMultistate_StateCount	DINT	Sets the number of states for multi-state values.
fInput_DefResolution	REAL	Default value for the *Resolution* parameter of inputs.
fInput_DefScaleOffset	REAL	Default value for the *Offset* parameter of inputs.
fOutput_DefResolution	REAL	Default value for the *Resolution* parameter of outputs.
fOutput_DefScaleOffset	REAL	Default value for the *Offset* parameter of outputs.
eInput_DefSensor	E_BA_MeasuringElement	Selection of the sensor with the special input type FB_BA_AI_IOEx [▶ 193].
fDefLimitDeadband	REAL	Default value for the Dead band limit parameter.
nDefTimeDelay_ToAbnormal	UDINT	Default value for the parameter Time delay of transitions to abnormal states.
nDefTimeDelayAO_ToAbnormal	UDINT	Default value for the *Time Delay* parameter of transitions to abnormal states of analog outputs.
nDefTimeDelayBO_ToAbnormal	UDINT	Default value for the *Time Delay* parameter of transitions to abnormal states of binary outputs.
nDefTimeDelayMO_ToAbnormal	UDINT	Default value for the *Time Delay* parameter of transitions to abnormal states of multi-stage outputs.
fDefCOVIncrement	REAL	Default value for the COV increment parameter.
nPlantCtrl_OpModeCount	DINT	Sets the maximum number of plant operation modes of FB_BA_PlantCtrl function blocks.
nPlantCtrl_AggregateCount	DINT	Sets the maximum number of aggregate [▶ 42] references of FB_BA_PlantCtrl function blocks.
nSeqLink_RefCount	DINT	Maximum number of controller references in a sequence linker.
nMaxSeqCtrl	USINT	The global parameter specifies the number of sequence controllers. It limits the data and command structure *arrSeqLinkData* within the structure ST_BA_SeqLink [▶ 123]. The value must not be less than 1.
nCollect_RefCount	DINT	Sets the maximum number of references of FB_BA_Collector function blocks.
eLoop_DefOpMode	E_BA_PIDMode	Default value for the *Operation mode* parameter.
nTrend_BufferSize	UDINT	Number of entries in a trend buffer.

Name	Type	Description
stTrend_DefStartTime	ST_BA_DateTime	Default value for the Start time parameter.
stTrend_DefStopTime	ST_BA_DateTime	Default value for the Stop time parameter.
bTrend_DefStopOnFull	BOOL	FALSE, Ring buffer; TRUE, Fixed memory that does not store anything when the buffer is full.
nTrend_DefLogInterval	UDINT	Default value for the Logging interval parameter.
nTrend_DefNotificationThreshold	UDINT	Default value for the Notification threshold parameter.
nCal_EntryCount	DINT	Number of entries in a calendar object.
nSched_EntryCount	DINT	Number of entries on a weekday ('T_BA_SchedWeek' [▶ 122]) or an exception ('ST_BA_SchedCalendar' [▶ 121] / 'ST_BA_SchedException' [▶ 121]).
nSched_CalendarCount	DINT	Number of calendar references ('T_BA_SchedCalendar [▶ 122]').
nSched_ExceptionCount	DINT	Number of exceptions ('T_BA_SchedExceptionList' [▶ 122]).
nSim_AISen_DefDampConstant	UDINT	Default value for the parameter *Damping constant*.
tSub_ReadTolerance	TIME	Default value for the *Read tolerance* parameter.
bSub_ClearOnReadError	BOOL	Default value for the *Reset on read errors* parameter.
tSub_DefReadInterval	TIME	Default value for the *Read interval* parameter.
nGroupCmd_RefCount	DINT	Number of references.
nGroupDsp_RefCount	DINT	Number of references.
nGroupVal_RefCount	DINT	Number of references.

6.1.2.1.2.3 BACNet

6.1.2.1.2.3.1 XBA_BACnetParam

The list contains globally valid parameters with which basic settings relating to BACnet can be made in TF8040:

```
VAR_GLOBAL CONSTANT
  {region 'Objects'}
    {region 'Local'}
      {region 'Event Config'}
        sEventMessageTextFormat           : STRING := '{Descr} - {EvtTrans}';
      {endregion}
    {endregion}
    {region 'Remote'}
      {region 'Analog Output'}
        fRM_AO_WriteIncrement             : REAL := 0.0;
      {endregion}
      {region 'Structured View'}
        eView_SubordinateAnnotationMode   : E_BACnet_AnnotationTitle := E_BACnet_AnnotationTitle.eSy
mbolName;
      {endregion}
    {endregion}
  {endregion}
  {region 'Priorities'}
    aPriority                             : ARRAY[E_BA_Priority.First .. E_BA_Priority.Last] OF E_BA
Cnet_Priority := [
```

```
                (*  eProgram            *) E_BACnet_Priority.eP15,
                (*  eManualRemote       *) E_BACnet_Priority.eP8,
                (*  eManualLocal        *) E_BACnet_Priority.eP7,
                (*  eCritical           *) E_BACnet_Priority.eP3,
                (*  eLifeSafety         *) E_BACnet_Priority.eP1
              ];
   {endregion}
   {region 'Translation'}
    aNodeType                                 : ARRAY[E_BA_NodeType.First .. E_BA_NodeType.Last] OF E_B
ACnet_NodeType := [
                (*  eUnknown             *) E_BACnet_NodeType.eUnknown,
                (*  eOther               *) E_BACnet_NodeType.eOther,
                (*  eGeneral             *) E_BACnet_NodeType.eOrganizational,
                (*  eLocation            *) E_BACnet_NodeType.eOrganizational,
                (*  eBuilding            *) E_BACnet_NodeType.eOrganizational,
                (*  eBuildingElement     *) E_BACnet_NodeType.eOrganizational,
                (*  eInformationFocus    *) E_BACnet_NodeType.eOrganizational,
                (*  eControlCabinet      *) E_BACnet_NodeType.eOrganizational,
                (*  eTrade               *) E_BACnet_NodeType.eOrganizational,
                (*  eFloor               *) E_BACnet_NodeType.eOrganizational,
                (*  eRoom                *) E_BACnet_NodeType.eOrganizational,
                (*  ePlant               *) E_BACnet_NodeType.eOrganizational,
                (*  eAggregate           *) E_BACnet_NodeType.eEquipment,
                (*  eFunction            *) E_BACnet_NodeType.eFunctional
              ];
    aNotifyType                                 : ARRAY[E_BA_EventType.First .. E_BA_EventType.Last] OF
E_BACnet_NotifyType := [
                (*  eAlarm               *) E_BACnet_NotifyType.eAlarm,
                (*  eDisturb             *) E_BACnet_NotifyType.eAlarm,
                (*  eMaintenance         *) E_BACnet_NotifyType.eNotifyEvent,
                (*  eNotification        *) E_BACnet_NotifyType.eNotifyEvent,
                (*  eOther               *) E_BACnet_NotifyType.eNotifyEvent
              ];
    aEventState                                 : ARRAY[E_BA_EventState.First .. E_BA_EventState.Last] O
F E_BACnet_EventState := [
                (*  eNormal             *) E_BACnet_EventState.eNormal,
                (*  eFault              *) E_BACnet_EventState.eFault,
                (*  eOffnormal          *) E_BACnet_EventState.eOffnormal,
                (*  eLowLimit           *) E_BACnet_EventState.eLowLimit,
                (*  eHighLimit          *) E_BACnet_EventState.eHighLimit
              ];
   {endregion}
END_VAR
```

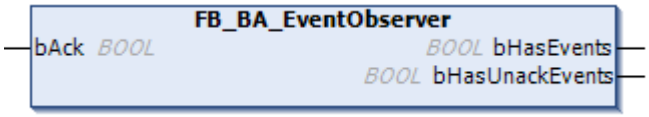
Name	Type	Description
sEventMessageTextFormat	STRING	A message text for the TwinCAT HMI BA as well as for a BACnet client is made up of different strings. The last part of the string is the *EventTrasitionText*. The texts for the various event transitions are in a list called TxtEvent_EN or TxtEvent_DE. These lists are created when a TF8040 Sample PLC is created. The front part of the message can consist of various texts. For example, from the *ObjectDescription*. The sEventMessageTextFormat string can be used to determine the format or composition of the EventMessageText. It is made up of texts and placeholders. Placeholders and free text such as a separator can be arranged one after the other. **Placeholder**: {ObjName} = Objectname {Descr} = Description {InstID} = InstanceID {EvtTrans} = EventTaransition In the following sample, the *Objectdescription* is placed in front of the *EventTransitionText*. The hyphen is used to place a separator between the two placeholders. sEventMessageTextFormat :='{Descr} - {EvtTrans};
fRM_AO_WriteIncrement	REAL	Threshold value that must be exceeded to output a value change at the analog output.
eView_SubordinateAnnotationMode	E_BACnet_AnnotationTitle	Property which is used to display a child object in the SubordinateList of the corresponding parent view object.
aPriority	ARRAY[E_BA_Priority [▶ 110].First .. E_BA_Priority.Last] OF E_BACnet_Priority	Defines the assignment of BACnet priorities from the Tc3_BACnetRev14 library in the priority array.
aNodeType	ARRAY[E_BA_NodeType [▶ 111].First .. E_BA_NodeType.Last] OF E_BACnet_NodeType	Conversion table for the *NodeType* property between the Tc2_XBA library and the Tc3_BACnetRev14 library.
aNotifyType	ARRAY[E_BA_EventType [▶ 103].First .. E_BA_EventType.Last] OF E_BACnet_NotifyType	Conversion table for the *NotifyType* property between the Tc2_XBA library and the Tc3_BACnetRev14.
aEventState	ARRAY[E_BA_EventState.First .. E_BA_EventState.Last] OF E_BACnet_EventState	Conversion table for the *EventState* property between the Tc2_XBA library and the Tc3_BACnetRev14.

6.1.2.1.3 POUs

6.1.2.1.3.1 FunctionBlocks

6.1.2.1.3.1.1 Events

6.1.2.1.3.1.1.1 FB_BA_EventObserver



The function block is used to evaluate alarms/events in the project structure and acknowledge them.

Access to the project structure with a parent/child relationship takes place via an assignment to the property *Parent*.

Syntax

```
FUNCTION_BLOCK FB_BA_EventObserver
VAR_INPUT
  bAck               : BOOL;
END_VAR
VAR_OUTPUT
  bHasEvents         : BOOL;
  bHasUnackEvents    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bAck	BOOL	The signal for acknowledging alarms / events is applied to the input.

Outputs

Name	Type	Description
bHasEvents	BOOL	Display that events are pending.
bHasUnackEvents	BOOL	Display that unacknowledged events are pending.

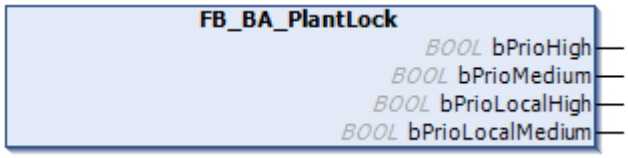
Properties

Name	Type	Access	Description
Parent	I_BA_View	Get, Set	Access to the project structure (base framework) is realized via an assignment to the property. If no assignment is made, the basis of the base framework is accessed. This means that all events / alarms are evaluated and their acknowledgement can be triggered accordingly. In addition, a warning is output to the error list of the TC3 development environment.
Valid	BOOL	Get	Indicates that a valid assignment is present at the interface *iParent* (parent).

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.2.1.0

6.1.2.1.3.1.1.2 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 [▶ 38] 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 of medium priority.
- **Local high**
  Enables a local shutdown of of higher priority.
- **Medium**
  Enables a higher-level shutdown of of medium priority.
- **High**
  Enables a higher-level shutdown of of higher priority.

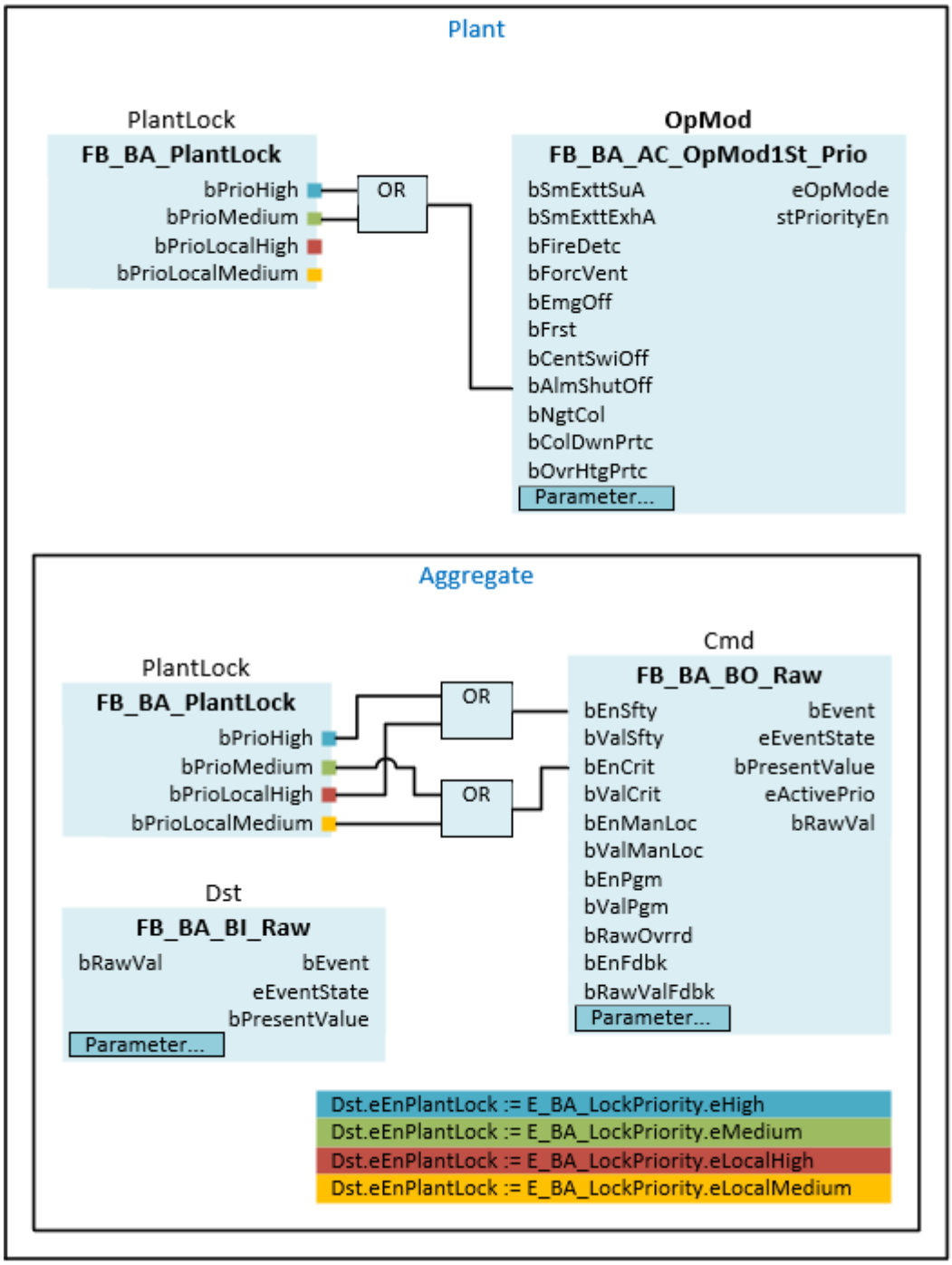
Medium priority is used for technically relevant faults that affect the safety of plants. High priority can be used for faults where the safety of people is at risk.

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 DST are as follows:

Dst.eEnPlantLock	Cmd.bPresentVal	Cmd.eActivePrio	OpMode.AlmShutOff
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

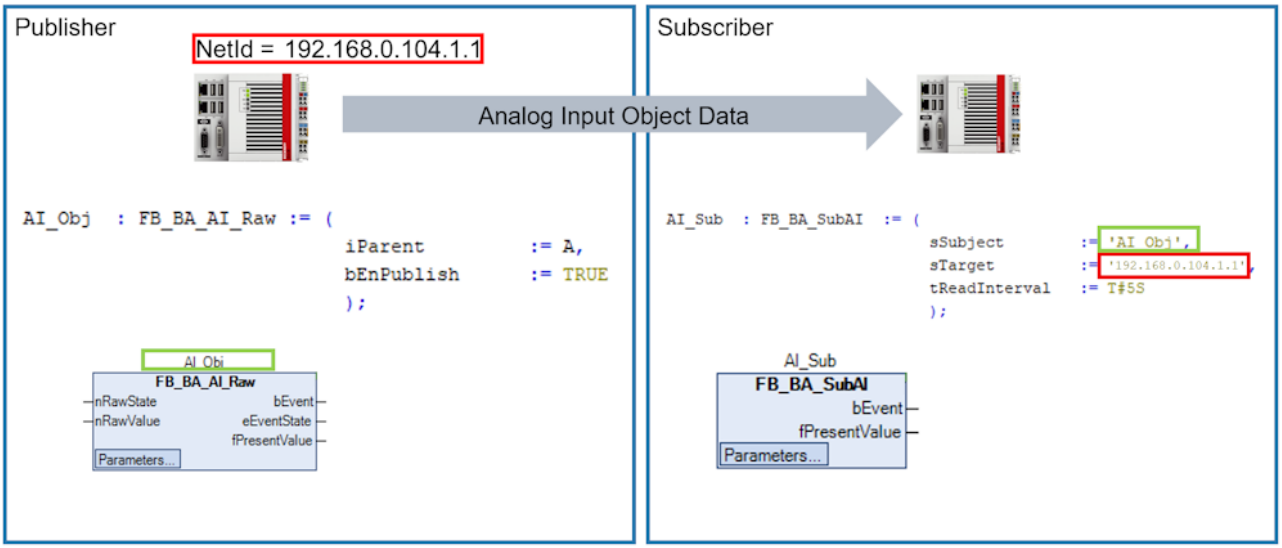
6.1.2.1.3.1.2 Reference

6.1.2.1.3.1.2.1 Publish and Subscribe

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 Publishers and Subscribers are implemented (e.g. on different PLCs), information is exchanged in a uniform manner.

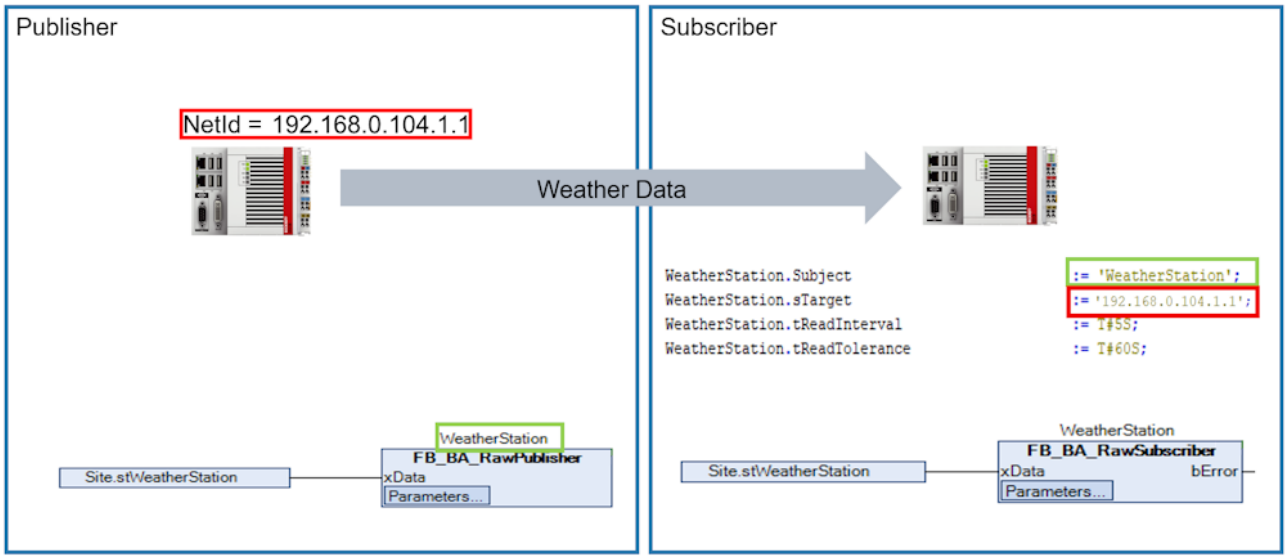
Sample 1



Application of Publisher and Subscriber to specific functions and objects.

Each function block can be used as Publisher. Function blocks that can subscribe to the desired data types serve as Subscribers. To do this, the unique object name of the Publisher and its AMS NetID must be specified on the Subscriber.

Sample 2



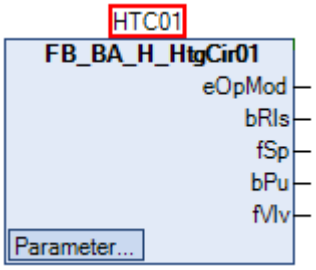
Application of Publisher and Subscriber for any data types.

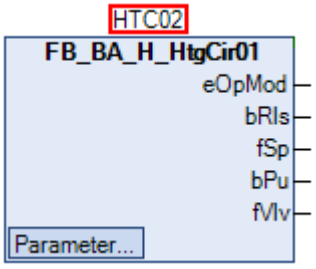
The FB_BA_RawPublisher [▶ 139] can be used to make information of different sizes available to the FB_BA_RawSubscriber [▶ 158]. As in the previous sample, the object name and the AMS NetID must be specified on the Subscriber. As indicated in the sample, this configuration takes place once in FB_init of FB_BA_RawSubscriber [▶ 158].

If there are several instances of an object with the same symbol name, it is necessary to specify the complete symbol name of the object in the parameter *sSubject*.

Sample 3

In the sample, the faults of two heating circuit pumps are to be published and subscribed to separately.

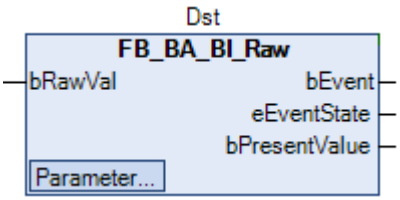




Within the heating circuits, the pumps *Pu* and the published binary objects of the fault *Dst* have the same name.

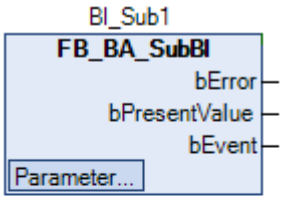
The parameter is therefore published from both templates in the same way.

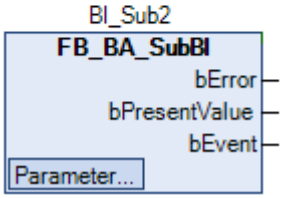
```
Dst                    : FB_BA_BI_Raw  := (
                                           bEnPublish     := TRUE
                                         );
```



Subscribing is done separately, you only need to ensure that the symbol path for the event to be subscribed to is specified correctly.

```
BI_Sub1     :   FB_BA_SubBI := (
                                sSubject       :='HTG.HTC01.Pu.Dst',
                                sTarget        :='127.0.0.1.1.1',
                                tReadInterval  := T#5S
                               );
BI_Sub2     :   FB_BA_SubBI:= (
                                sSubject       :='HTG.HTC02.Pu.Dst',
                                sTarget        :='127.0.0.1.1.1',
                                tReadInterval  := T#5S
                               );
```





6.1.2.1.3.1.2.1.1 Publisher

Function blocks for local or cross-control publishing of information.

6.1.2.1.3.1.2.1.1.1 FB_BA_RawPublisher

Function block for providing any information.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Publisher

Syntax

```
VAR_INPUT
  bEnPublish    : BOOL;
  xData         : ANY;
END_VAR
```

Inputs

Name	Type	Description
bEnPublish	BOOL	Variable for enabling variables or data sets via ADS.
xData	ANY	Data to be published.

Requirements

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

6.1.2.1.3.1.2.1.2 Subscriber

Function blocks for subscribing to local or cross-control information.

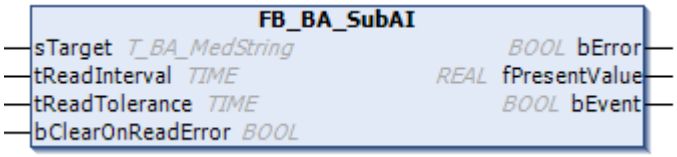
6.1.2.1.3.1.2.1.2.1 Objects

This folder offers objects that make it possible to subscribe to published object information either within the controller or across controllers.

6.1.Analog
2.1.
3.1.
2.1.
2.1.
1

Function blocks for subscribing to analog objects.

6.1.FB_BA_SubAI
2.1.
3.1.
2.1.
2.1.
1.1



The function block allows to subscribe to an analog input object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtA

Syntax

```
FUNCTION_BLOCK FB_BA_SubAI EXTENDS FB_BA_SubEvtA
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
```

```
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  fPresentValue             : REAL;
  bEvent                    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

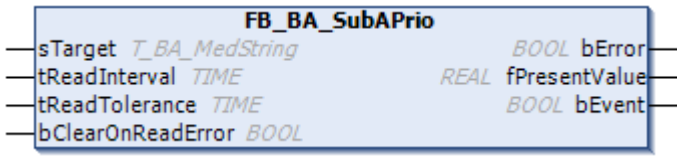
Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
fPresentValue	REAL	Current analog output value of the object.
bEvent	BOOL	The output is TRUE if an event or error has occurred.

Requirements

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

6.1.FB_BA_SubAPrio
2.1.
3.1.
2.1.
2.1.
1.2



The function block enables subscribing to an analog input object with a priority array.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtA

Syntax

```
FUNCTION_BLOCK FB_BA_SubAPrio EXTENDS FB_BA_SubEvtA
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
```

```
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  fPresentValue             : REAL;
  bEvent                    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

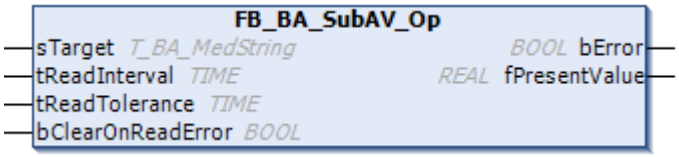
Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
fPresentValue	REAL	Current analog output value of the object.
bEvent	BOOL	The output is TRUE if an event or error has occurred.

Requirements

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

6.1.FB_BA_SubAV_Op
2.1.
3.1.
2.1.
2.1.
1.3



The function block allows to subscribe to an analog value object [▶ 202] to display an analog value.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubA

Syntax

```
FUNCTION_BLOCK FB_BA_SubAV_Op EXTENDS FB_BA_SubA
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR
```

```
VAR_OUTPUT
  bError                    : BOOL;
  fPresentValue             : REAL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
fPresentValue	REAL	Current analog output value of the object.

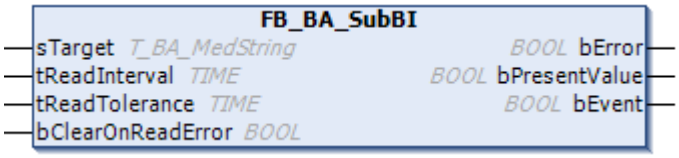
Requirements

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

6.1.Binary
2.1.
3.1.
2.1.
2.1.
2

Function blocks for subscribing to binary objects.

6.1.FB_BA_SubBI
2.1.
3.1.
2.1.
2.1.
2.1



The function block allows to subscribe to a binary input object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtB

Syntax

```
FUNCTION_BLOCK FB_BA_SubBI EXTENDS FB_BA_SubEvtB
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  bPresentValue             : BOOL;
  bEvent                    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

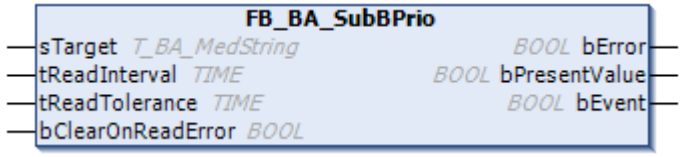
Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
bPresentValue	BOOL	Current binary output value of the object.
bEvent	BOOL	The output is TRUE if an event or error has occurred.

Requirements

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

6.1.FB_BA_SubBPrio
2.1.
3.1.
2.1.
2.1.
2.2



The function block allows to subscribe to a binary input object with a priority array.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtB

Syntax

```
FUNCTION_BLOCK FB_BA_SubBPrio EXTENDS FB_BA_SubEvtB
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  bPresentValue             : BOOL;
  bEvent                    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

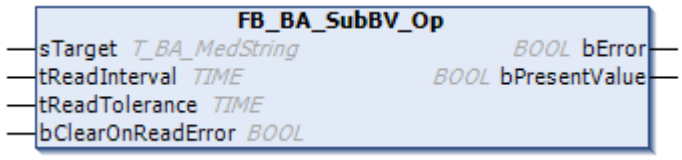
Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
bPresentValue	BOOL	Current binary output value of the object.
bEvent	BOOL	The output is TRUE if an event or error has occurred.

Requirements

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

6.1.FB_BA_SubBV_Op
2.1.
3.1.
2.1.
2.1.
2.3



The function block allows to subscribe to a binary value object [▶ 216] to display a binary value.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubB

Syntax

```
FUNCTION_BLOCK FB_BA_SubBV_Op EXTENDS FB_BA_SubB
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  bPresentValue             : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
bPresentValue	BOOL	Current binary output value of the object.

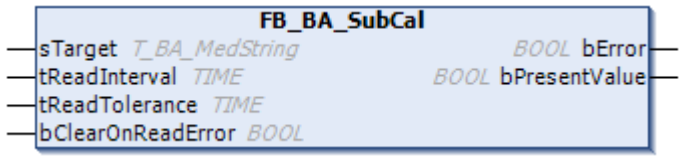
Requirements

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

6.1.Misc
2.1.
3.1.
2.1.
2.1.
3

Function blocks for subscribing to sophisticated objects.

6.1.FB_BA_SubCal
2.1.
3.1.
2.1.
2.1.
3.1



The function block allows to subscribe to a calendar object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubB

Syntax

```
FUNCTION_BLOCK FB_BA_SubCal EXTENDS FB_BA_SubB
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  bPresentValue             : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

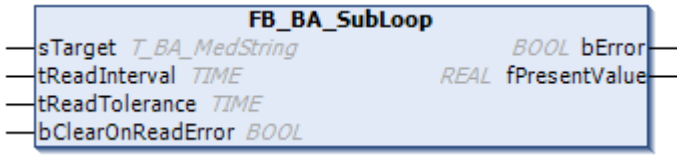
Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
bPresentValue	BOOL	Current binary output value of the object.

Requirements

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

6.1.FB_BA_SubLoop
2.1.
3.1.
2.1.
2.1.
3.2



The function block allows to subscribe to of a loop object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubA

Syntax

```
FUNCTION_BLOCK FB_BA_SubLoop EXTENDS FB_BA_SubA
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  fPresentValue             : REAL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
fPresentValue	REAL	Current analog output value of the object.

Requirements

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

6.1.FB_BA_SubSchedA
2.1.
3.1.
2.1.
2.1.
3.3

The function block allows to subscribe to an analog scheduler object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

Syntax

```
FUNCTION_BLOCK FB_BA_SubSchedA EXTENDS FB_BA_Subscriber
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
```

```
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  fPresentValue             : REAL;
  fPredictedValue           : REAL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

```
VAR_OUTPUT
  bError                    : BOOL;
  fPresentValue             : REAL;
  fPredictedValue           : REAL;
END_VAR
```

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
fPresentValue	REAL	Current analog output value of the object.
fPredictedValue	REAL	Value that is assumed after the next switching.

Properties

Name	Type	Access	Description
Unit	E_BA_Unit	Get	Unit.

Methods

Name	Description
GetData [▶ 149]	Contains the subscribed data.

Requirements

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

6.1.GetData
2.1.
3.1.
2.1.
2.1.
3.3.
1

The method saves subscribed data of a Publisher.

Syntax

```
METHOD GetData
VAR_OUTPUT
  pData    : PVOID;
  nSize    : DINT;
END_VAR
```

Outputs

Name	Type	Description
pData	pData	Pointer to the subscribed data.
nSize	DINT	Size of the subscribed data.

6.1.FB_BA_SubSchedB
2.1.
3.1.
2.1.
2.1.
3.4

The function block allows to subscribe to a binary scheduler object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

Syntax

```
FUNCTION_BLOCK FB_BA_SubSchedB EXTENDS FB_BA_Subscriber
VAR_INPUT
  sTarget                  : T_BA_MedString;
  tReadInterval            : TIME;
  tReadTolerance           : TIME;
  bClearOnReadError        : BOOL;
END_VAR

VAR_OUTPUT
  bError                   : BOOL;
  bPresentValue            : BOOL;
  bPredictedValue          : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
bPresentValue	BOOL	Current binary output value of the object.
bPredictedValue	BOOL	Value that is assumed after the next switching.

Properties

Name	Type	Access	Description
ActiveText	STRING	Get	Active text
InactiveText	STRING	Get	Inactive text

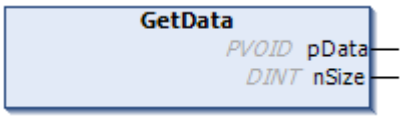
Methods

Name	Description
GetData [▶ 151]	Contains the subscribed data.

Requirements

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

6.1.GetData
2.1.
3.1.
2.1.
2.1.
3.4.
1



The method saves subscribed data of a Publisher.

Syntax

```
METHOD GetData
VAR_OUTPUT
  pData    : PVOID;
  nSize    : DINT;
END_VAR
```

Outputs

Name	Type	Description
pData	pData	Pointer to the subscribed data.
nSize	DINT	Size of the subscribed data.

6.1.FB_BA_SubSchedM
2.1.
3.1.
2.1.
2.1.
3.5

The function block allows to subscribe to a multi-state scheduler object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

Syntax

```
FUNCTION_BLOCK FB_BA_SubSchedM EXTENDS FB_BA_Subscriber
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  nPresentValue             : UDINT;
  nPredictedValue           : UDINT;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
nPresentValue	UDINT	Analog output value.
nPredictedValue	UDINT	Value that is assumed after the next switching.

Properties

Name	Type	Access	Description
StateCount	UDINT	Get	Number of possible states.
StateText	T_BA_StateTextArray [▶ 121]	Get	Output of the state text.

Methods

Name	Description
GetData [▶ 153]	Contains the subscribed data.

Requirements

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

6.1.GetData
2.1.
3.1.
2.1.
2.1.
3.5.
1

The method saves subscribed data of a Publisher.

Syntax

```
METHOD GetData
VAR_OUTPUT
  pData    : PVOID;
  nSize    : DINT;
END_VAR
```

Outputs

Name	Type	Description
pData	pData	Pointer to the subscribed data.
nSize	DINT	Size of the subscribed data.

6.1.FB_BA_SubTrend
2.1.
3.1.
2.1.
2.1.
3.6

The function block allows to subscribe to a trend object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

Syntax

```
FUNCTION_BLOCK FB_BA_SubTrend EXTENDS FB_BA_Subscriber
VAR_INPUT
  sTarget                  : T_BA_MedString;
```

```
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  bEvent                    : BOOL;
  nRecordCount              : UDINT;
  nTotalRecordCount         : UDINT;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
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.
nRecordCount	UDINT	Number of records.
nTotalRecordCount	UDINT	Absolute number of records.

Properties

Name	Type	Access	Description
Enable	BOOL	Get, Set	Enable of the subscription.

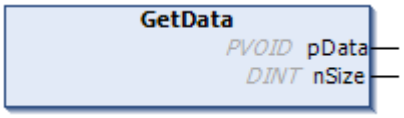
Methods

Name	Description
GetData [▶ 154]	Contains the subscribed data.

Requirements

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

6.1.GetData
2.1.
3.1.
2.1.
2.1.
3.6.
1



The method saves subscribed data of a Publisher.

Syntax

```
METHOD GetData
VAR_OUTPUT
  pData    : PVOID;
  nSize    : DINT;
END_VAR
```

Outputs

Name	Type	Description
pData	pData	Pointer to the subscribed data.
nSize	DINT	Size of the subscribed data.

6.1.Multistate
2.1.
3.1.
2.1.
2.1.
4

Function blocks for subscribing to multi-state objects.

6.1.FB_BA_SubMI
2.1.
3.1.
2.1.
2.1.
4.1

The function block allows to subscribe to a multi-state input object.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtM

Syntax

```
FUNCTION_BLOCK FB_BA_SubBV_Op EXTENDS FB_BA_SubEvtM
VAR_INPUT
  sTarget                  : T_BA_MedString;
  tReadInterval            : TIME;
  tReadTolerance           : TIME;
  bClearOnReadError        : BOOL;
END_VAR

VAR_OUTPUT
  bError                   : BOOL;
  nPresentValue            : UDINT;
  bEvent                   : BOOL
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

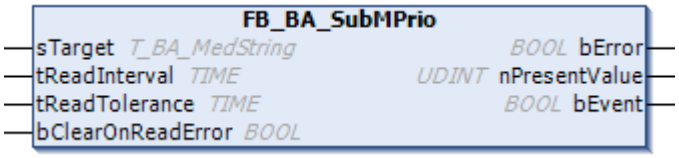
Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
nPresentValue	UDINT	Analog output value.
bEvent	BOOL	The output is TRUE if an event or error has occurred.

Requirements

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

6.1.FB_BA_SubMPrio
2.1.
3.1.
2.1.
2.1.
4.2



The function block allows to subscribe to a multi-state value object with a priority array.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubM

Syntax

```
FUNCTION_BLOCK FB_BA_SubBV_Op EXTENDS FB_BA_SubM
VAR_INPUT
  sTarget                   : T_BA_MedString;
  tReadInterval             : TIME;
  tReadTolerance            : TIME;
  bClearOnReadError         : BOOL;
END_VAR

VAR_OUTPUT
  bError                    : BOOL;
  nPresentValue             : UDINT;
  bEvent                    : BOOL
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

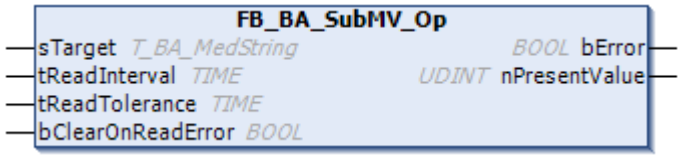
Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
nPresentValue	UDINT	Analog output value.
bEvent	BOOL	The output is TRUE if an event or error has occurred.

Requirements

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

6.1.FB_BA_SubMV_Op
2.1.
3.1.
2.1.
2.1.
4.3



The function block allows to subscribe to a multi-state value object [▶ 239] to display a multi-state value.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubM

Syntax

```
FUNCTION_BLOCK FB_BA_SubBV_Op EXTENDS FB_BA_SubM
VAR_INPUT
  sTarget                  : T_BA_MedString;
  tReadInterval            : TIME;
  tReadTolerance           : TIME;
  bClearOnReadError        : BOOL;
END_VAR

VAR_OUTPUT
  bError                   : BOOL;
  nPresentValue            : UDINT;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.

Outputs

Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.
nPresentValue	UDINT	Analog output value.

Requirements

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

6.1.2.1.3.1.2.1.2.2 FB_BA_RawSubscriber

Function block for subscribing to any information.

Inheritance hierarchy

FB_BA_Base

FB_BA_Subscriber

Syntax

```
FUNCTION_BLOCK FB_BA_RawSubscriber EXTENDS FB_BA_Subscriber
VAR_INPUT
  sTarget               : T_BA_MedString;
  tReadInterval         : TIME;
  tReadTolerance        : TIME;
  bClearOnReadError     : BOOL;
  xData                 : _System.AnyType;
END_VAR

VAR_OUTPUT
  bError                : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sTarget	T_BA_MedString [▶ 120]	AMS NetId of the Publisher.
tReadInterval	TIME	Read interval [s].
tReadTolerance	TIME	Waiting time until a pending error is output [ms].
bClearOnReadError	BOOL	If TRUE, the data is deleted if an error occurs.
xData	ANY	Read data.

Outputs

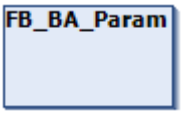
Name	Type	Description
bError	BOOL	Displays the current error state of the subscription. Details can be found in the corresponding error message in the event of an error.

Requirements

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

6.1.2.1.3.1.3 Accessories

6.1.2.1.3.1.3.1 FB_BA_Param



The function block enables configuration of the properties contained in the global variable list XBA_Param [▶ 126].

Existing limitations that apply to the writing of parameters are observed (see also Regions [▶ 98]).
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

6.1.2.1.3.1.3.2 EventSubscriber

The Event Subscribers enable generic subscription to events.

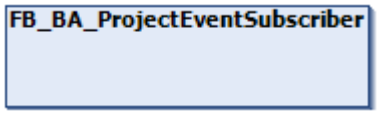
This allows you to respond to various events (such as in the project or on local objects) at a central location by overwriting the respective events within the methods provided.

The possible events are offered as a method and the call is made as required.

Notes on using the Event Subscriber

It should be noted that the use of these function blocks can affect system performance.
Depending on the programmed logic, the *OnObjectStateChange* method, for example, may run too slowly. This can lead to cycle overruns.

6.1.2.1.3.1.3.2.1 FB_BA_ProjectEventSubscriber



The Subscriber function block offers the possibility to respond to project-wide events.

Within the method, the user can describe and output information about the project state.

Method

Name	Description
OnProjectStateChanged [▶ 160]	Change of the project state.

Requirements

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

6.1.2.1.3.1.3.2.1.1 OnProjectStateChanged

The method is called when the project state has changed.

Syntax

```
METHOD PROTECTED OnProjectStateChanged
VAR_INPUT
  eState    : E_BA_ProjectState;
END_VAR
```

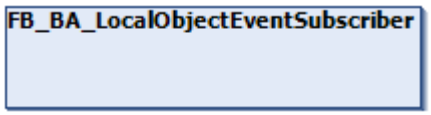
Inputs

Name	Type	Description
eState	E_BA_ProjectState [▶ 113]	State to which the project has changed.

Requirements

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

6.1.2.1.3.1.3.2.2 FB_BA_LocalObjectEventSubscriber



The Subscriber function block offers the possibility to respond to events in local objects.

Methods

Name	Description
OnBACnetObjectStateChanged [▶ 161]	Change the state of a BACnet object.
OnInitStateChange [▶ 161]	Changing the initialization state of an object.
OnInitInstanceId [▶ 162]	Change the instance ID of an object.
OnEventChange [▶ 162]	Change of an event within an object.

Requirements

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

6.1.2.1.3.1.3.2.2.1 OnBACnetObjectStateChanged

The method is called when the initialization state of a BACnet object changes.

Syntax

```
METHOD PROTECTED OnBACnetObjectStateChanged
VAR_INPUT
  iObject           : I_BA_Object;
  fbBACnetObject    : POINTER TO FB_BACnet_BaseObject;
  tState            : TCOM_STATE;
END_VAR
```

Inputs

Name	Type	Description
iObject	I_BA_Object	Interface to the context object.
fbBACnetObject	POINTER TO FB_BACnet_BaseObject	Supplement specific BACnet object of the context object.
tState	TCOM_STATE	New state of the BACnet object.

Requirements

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

6.1.2.1.3.1.3.2.2.2 OnInitStateChange

The method is called when the initialization state of an object changes.

Syntax

```
METHOD PROTECTED OnInitStateChange
VAR_INPUT
  iObject    : I_BA_Object;
  eState     : E_BA_InitState;
END_VAR
```

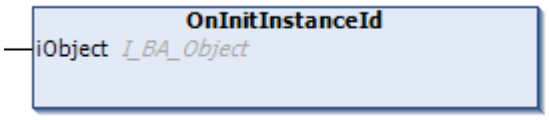
Inputs

Name	Type	Description
iObject	I_BA_Object	Interface to the context object. [▶ 264]
eState	E_BA_InitState [▶ 114]	New state after initialization.

Requirements

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

6.1.2.1.3.1.3.2.2.3 OnInitInstanceId



The method can be used to assign a specific value to the instance ID of an object.

Otherwise, the instance IDs are generated continuously.

Each instance ID may only be assigned once.

Syntax

```
METHOD PROTECTED OnInitInstanceId
VAR_INPUT
  iObject    : I_BA_Object;
END_VAR
```

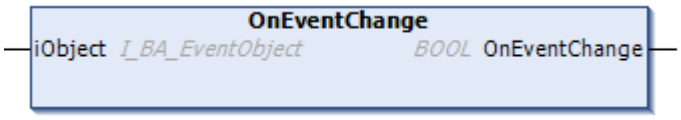
Inputs

Name	Type	Description
iObject	I_BA_Object	Interface to the context object. [▶ 264]

Requirements

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

6.1.2.1.3.1.3.2.2.4 OnEventChange



The method is called when an object generates an event, or when a generated object changes or is deleted.

Syntax

```
METHOD PROTECTED OnEventChange
VAR_INPUT
  iObject    : I_BA_Object;
END_VAR
```

Inputs

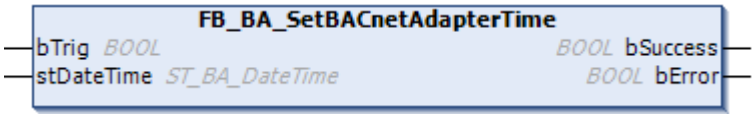
Name	Type	Description
iObject	I_BA_Object	Interface to the context object. [▶ 264]

Requirements

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

6.1.2.1.3.1.4 System

6.1.2.1.3.1.4.1 FB_BA_SetBACnetAdapterTime



The function block FB_BA_SetBACnetAdapterTime can be used to set the local BACnet system time of a TwinCAT system.

Syntax

```
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.

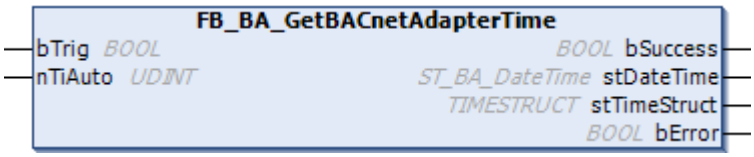
Outputs

Name	Type	Description
bSuccess	BOOL	The time specification *stDateTime* 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 v3.1.4024.56	Tc3_XBA from v5.6.1.0

6.1.2.1.3.1.4.2 FB_BA_GetBACnetAdapterTime



The function block FB_BA_GetBACnetAdapterTime 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.

Syntax

```
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

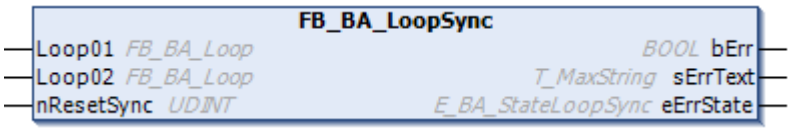
Name	Type	Description
bSuccess	BOOL	The system time was read successfully from the target system. The variable is TRUE for one cycle.
stDateTime	ST_BA_DateTime	Current BACnet system time of a TwinCAT system.
stTimeStruct	TIMESTRUCT	Current BACnet system time of a TwinCAT system.
bError	BOOL	Error while reading the BACnet system time.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.56	Tc3_XBA from v5.6.1.0

6.1.2.1.3.1.5 Control

6.1.2.1.3.1.5.1 FB_BA_LoopSync



The function block FB_BA_LoopSync is used for parameter synchronization for two controllers of type FB_BA_Loop [▶ 220]. 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 [▶ 220] 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 *eActionRm* fehlerhaft'	'Synchronization *eActionRm* faulty'	Possibly the wrong synchronization can be triggered by the input *eActionPgm*. This input overwrites the VAR_INPUT CONSTANT variable *eActionRm* to be monitored when occupied.
'Synchronisation *nDampConstant* fehlerhaft'	'Synchronization *nDampConstant* faulty'	It is possible that the wrong synchronization was triggered by the input *eActionPgm*. This input overwrites the VAR_INPUT CONSTANT variable *eActionRm* to be monitored when occupied.
'Synchronisation *fDerivativeConstant* fehlerhaft'	'Synchronization *fDerivativeConstant* faulty'	
'Synchronisation *fIntegralConstant* fehlerhaft'	'Synchronization *fIntegralConstant* faulty'	
'Synchronisation *fMaxOutputRm* fehlerhaft'	'Synchronization *fMaxOutputRm* faulty'	Possibly the wrong synchronization can be triggered by the input *fMaxOutputPgm*. This input overwrites the VAR_INPUT CONSTANT variable *fMaxOutputRm* to be monitored when occupied.
'Synchronisation *fMinOutputRm* fehlerhaft'	'Synchronization *fMinOutputRm* faulty'	Possibly the wrong synchronization can be triggered by the input *fMinOutputPgm*. This input overwrites the VAR_INPUT CONSTANT variable *fMinOutputRm* to be monitored when occupied.
'Synchronisation *fNeutralZone* fehlerhaft'	'Synchronization *fNeutralZone* faulty'	
'Synchronisation *eOpMode* fehlerhaft'	'Synchronization *eOpMode* faulty'	
'Synchronisation *fProportionalConstant* fehlerhaft'	'Synchronization *fProportionalConstant* faulty'	

Syntax

```
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 text form [▶ 166].
eErrState	E_BA_StateLoopSync [▶ 100]	The enumeration shows the state of the synchronization.

Inputs Outputs

Name	Type	Description
Loop1	FB_BA_Loop [▶ 220]	Reference to controller no. 1 of the parameter adjustment.
Loop2	FB_BA_Loop [▶ 220]	Reference to controller no. 2 of the parameter adjustment.

Properties

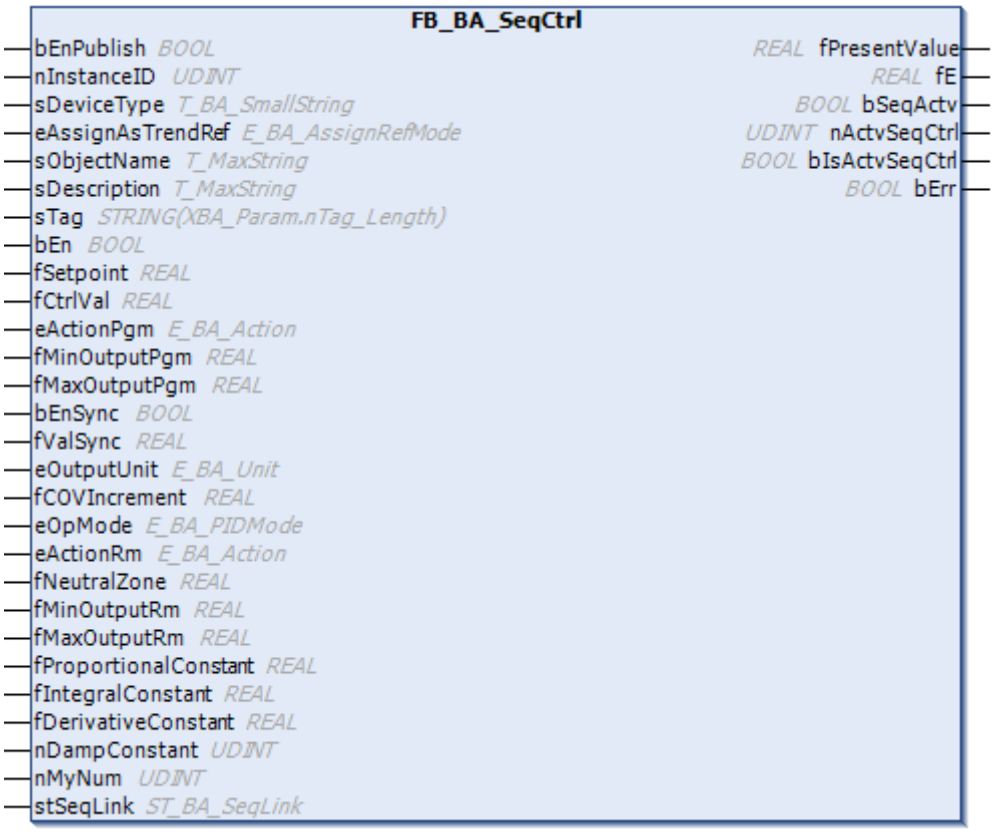
Name	Type	Access	Description
ErrText	T_MaxString	Get	The property *ErrText* displays the error texts from *sErrText*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.56	Tc3_XBA from v5.6.1.0

6.1.2.1.3.1.6 Sequence

6.1.2.1.3.1.6.1 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 [▶ 171] 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 [▶ 171] takes place via the structure stSeqLink [▶ 123].

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.eReverse		fMaxOutputPgm / fMaxOutputRm
TRUE	FALSE	E_BA_Action.eDirect		fMinOutputPgm / fMinOutputRm
FALSE	FALSE			0
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl > nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl < nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl < nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl > nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE		nActvSeqCtrl = nMyNum	FB_BA_Loop.fPresentValue

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 [▶ 264]

FB_BA_Loop [▶ 220]

Syntax

```
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	My number in the sequence. This number may only be present once in a sequence control. An internal limitation only allows values from 1 - XBA_Param.nMaxSeqCtrl [▶ 126].

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 *IsActvSeqCtrl* also displays this message.
bErr	BOOL	Indicates that an error has been detected. More detailed information is shown in the properties *ErrorParamMaxSeqCtrl and SeqNumMultiple*. Further explanations on error analysis can be found at Error detection [▶ 169].

/ Inputs Outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [▶ 168] and the function block FB_BA_SequenceLinkBase [▶ 171].

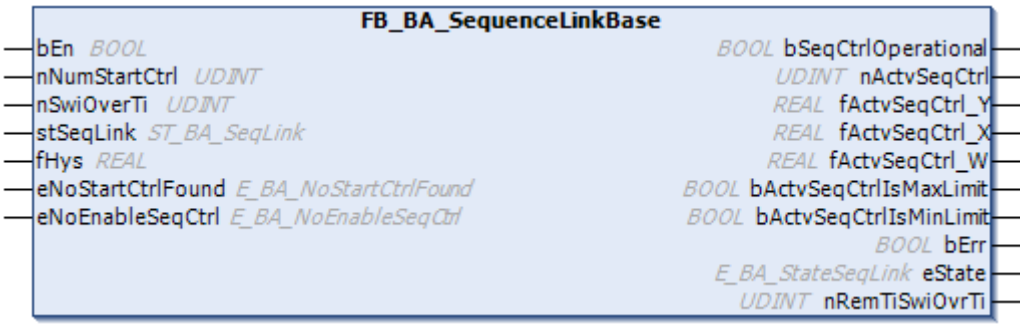
Properties

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 [▶ 284]. 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 v3.1.4024.56	Tc3_XBA from v5.6.1.0

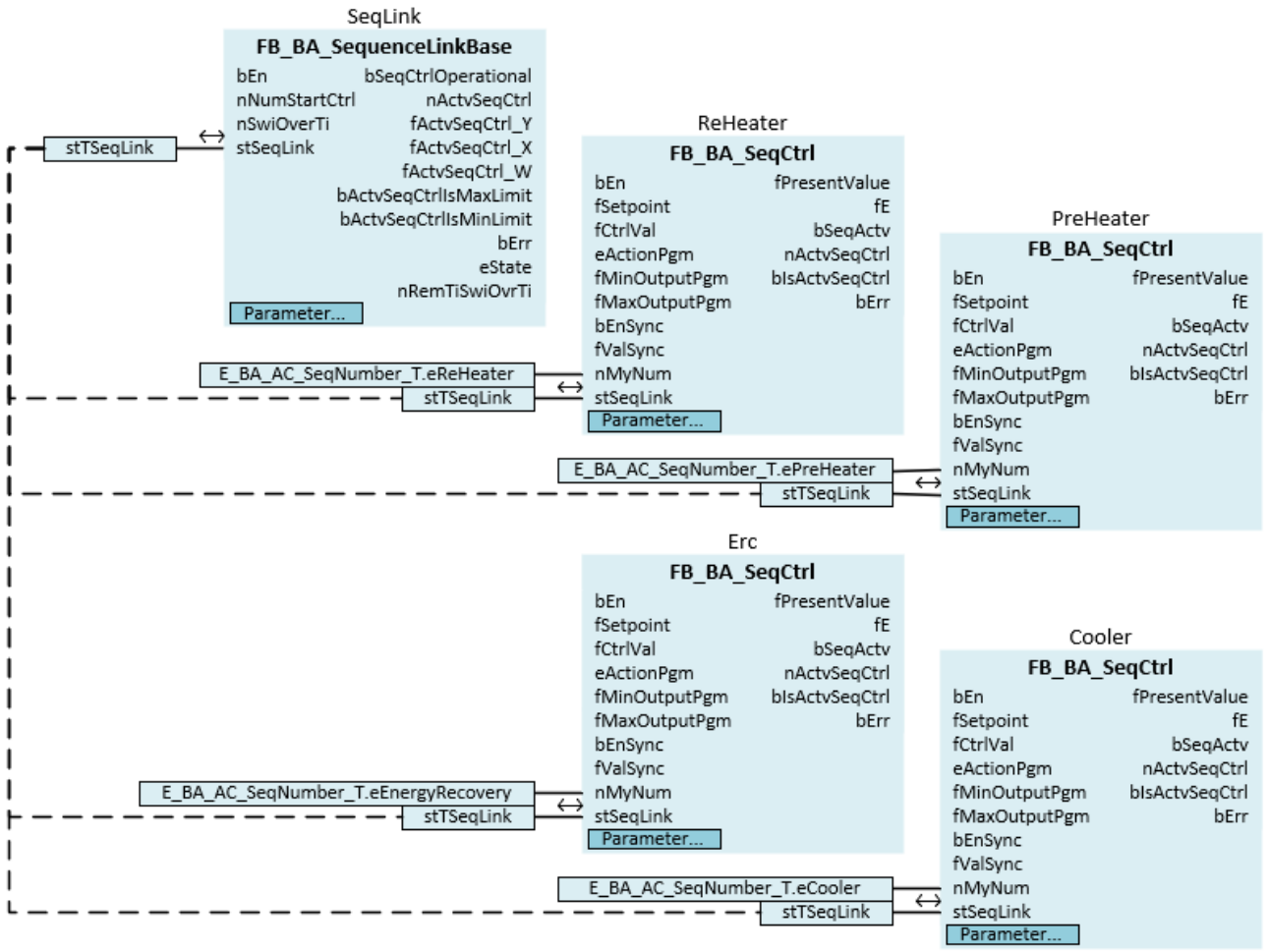
6.1.2.1.3.1.6.2 FB_BA_SequenceLinkBase



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 [▶ 168] and the *FB_BA_SequenceLinkBase* takes place via the structure stSeqLink [▶ 123].



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 [▶ 168] on the ranking of the active controller.

If the sequence controller specified at *nNumStartCtrl* is not ready for operation, the enum eNoStartCtrlFound 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 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 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

Syntax

```
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 [▶ 284].
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, *bActvSeqCtrlIsMaxLimit* is not set to FALSE until *fActvSeqCtrl_Y* has fallen below (*fActvSeqCtrl_Y* - *fHys*).
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, *bActvSeqCtrlIsMinLimit* is not set to FALSE until *fActvSeqCtrl_Y* has risen above (*fActvSeqCtrl_Y* + *fHys*).
bErr	BOOL	Indicates that an error has been detected. More detailed information is provided by the property *StateText*. In addition, the enum eState displays further information. Further explanations on error analysis can be found at Error detection [▶ 173].
eState	E_BA_StateSeqLink	The enumeration shows the state of the *FB_BA_SequenceLinkBase* and the sequence control.
nRemTiSwiOvrTi	UDINT	Countdown of switching to the next sequence controller [s].

Inputs Outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [▶ 168] and the function block FB_BA_SequenceLinkBase [▶ 171].

Inputs CONSTANT

Name	Type	Description
fHys	REAL	This hysteresis value sets a hysteresis around the upper and lower output limit *bActvSeqCtrlIsMaxLimit* / *bActvSeqCtrlIsMinLimit* of the active sequence controller. An internal limitation only allows values from 0 - 2.
eNoStartCtrlFound	E_BA_NoStartCtrlFound	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	If the start controller at the input *nNumStartCtrl* 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 *E_BA_Action.eDirect*.
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 *E_BA_Action.eReverse*.
NextActionActvSeqCtrl	UDINT	Get	Depending on the control direction of the active sequence controller *nActvSeqCtrl* 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 *E_BA_NoEnableSeqCtrl.eLastSeqNum* 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 *E_BA_NoEnableSeqCtrl.eFirstSeqNum* 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 *nActvSeqCtrl*.
ActvSeqCtrl_ControlValue	REAL	Get	The property shows the control value of the active sequence controller, see *fActvSeqCtrl_Y*.
ActvSeqCtrl_IsMaxLimit	BOOL	Get	The property indicates that the upper output limit of the active sequence controller has been reached, see *bActvSeqCtrlIsMaxLimit*.
ActvSeqCtrl_IsMinLimit	BOOL	Get	The property indicates that the lower output limit of the active sequence controller has been reached, see *bActvSeqCtrlIsMinLimit*.
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 [▶ 284]. 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 *nNumStartCtrl*, 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.
NextSeqSpIsSmaller	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 an 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 v3.1.4024.56	Tc3_XBA from v5.6.1.0

6.1.2.1.3.2 Functions

6.1.2.1.3.2.1 Parameter

6.1.2.1.3.2.1.1 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 [▶ 120]:

```
F_BA_CalendarBuilder()
  .AppendDate(2020, E_BA_Month.eMarch, 10)
  .AppendDate(2020, E_BA_Month.eSeptember, 15)
  .Build()
```

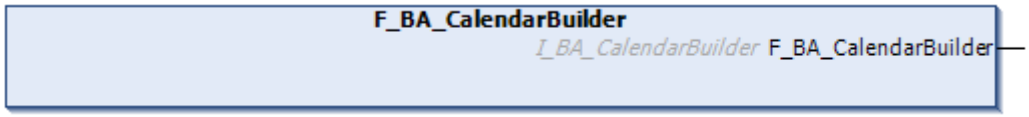
The benefits are:

- Self-explanatory notation
- Highly flexible
- Equally applicable in declaration and implementation code

See also:

Examples

6.1.2.1.3.2.1.1.1 F_BA_CalendarBuilder



The function provides methods that simplify the initialization of calendars [▶ 120]:

```
Cal1    : 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 [▶ 180]	Adds a date.
AppendDateRange [▶ 179]	Adds a date range.
AppendWeekNDay [▶ 181]	Adds a specific day of the week within a month.

See also:

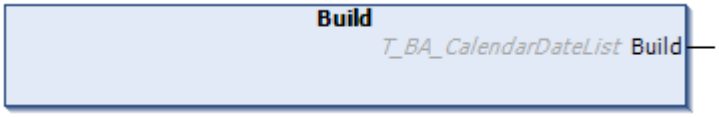
Samples

Requirements

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

6.1.2.1.3.2.1.1.1.1 Management

6.1.Build
2.1.
3.2.
1.1.
1.1.
1



The method outputs the contents of a date list [▶ 120].

6.1.2.1.3.2.1.1.1.2 Value

6.1.AppendDateRange
2.1.
3.2.
1.1.
1.2.
1

The method adds a defined period to the initialized calendar.

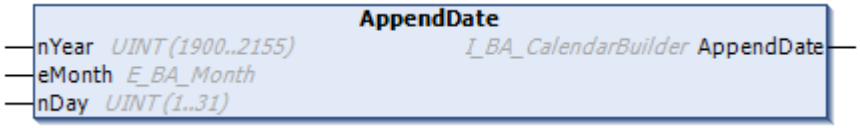
Illustration

```
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.
3.2.
1.1.
1.2.
2



The method appends a date value to the initialized calendar.

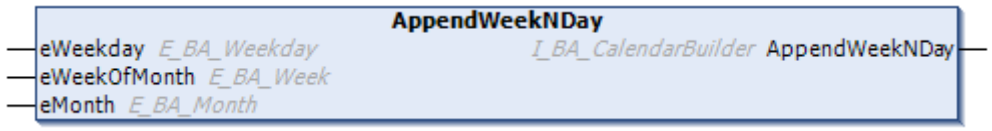
Illustration

```
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.
3.2.
1.1.
1.2.
3



The method appends a defined day to the initialized calendar, in a given week of a month, in a given month.

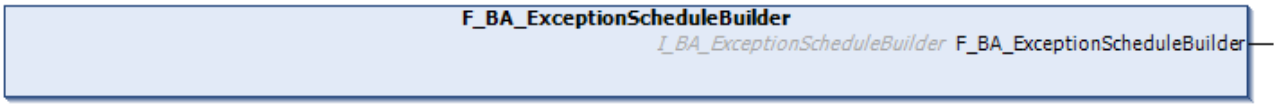
Illustration

```
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

6.1.2.1.3.2.1.1.2 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 [▶ 182]	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

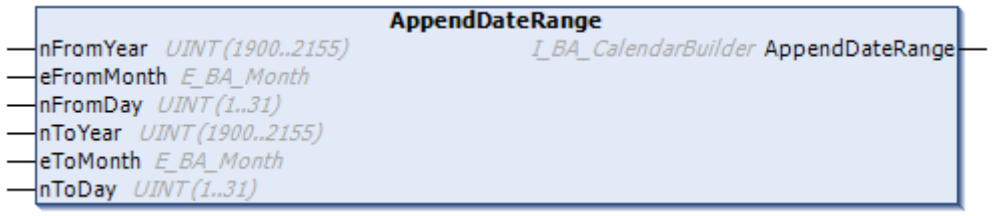
6.1.2.1.3.2.1.1.2.1 Management

6.1.Build
2.1.
3.2.
1.1.
2.1.
1

The method outputs the content Exception calendar [▶ 122].

6.1.2.1.3.2.1.1.2.2 Value

6.1.AppendDateRange
2.1.
3.2.
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.
3.2.
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.
3.2.
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.

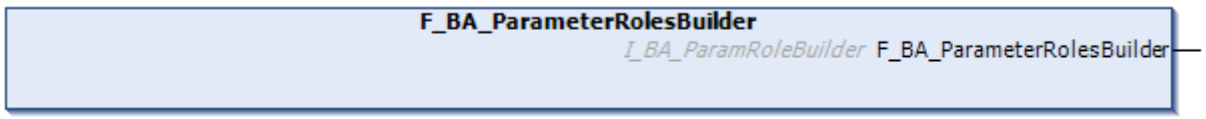
Illustration

```
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

6.1.2.1.3.2.1.1.3 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:

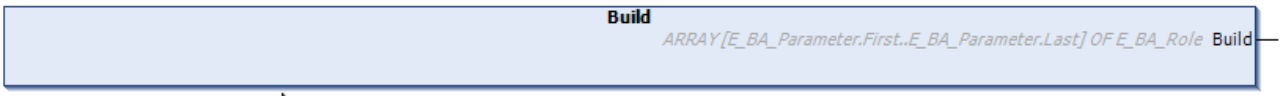
FB_BA_Param [▶ 159]

Requirements

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

6.1.2.1.3.2.1.1.3.1 Management

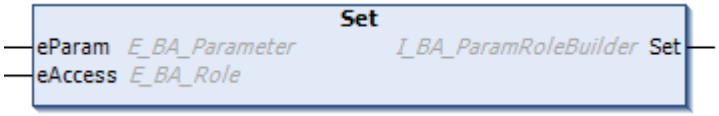
6.1.Build
2.1.
3.2.
1.1.
3.1.
1



The method outputs the content of the possible BACnet Properties [▶ 108] according to the user role [▶ 99].

6.1.2.1.3.2.1.1.3.2 Value

6.1.Set
2.1.
3.2.
1.1.
3.2.
1



The method provided a BACnet property depending on the user role.

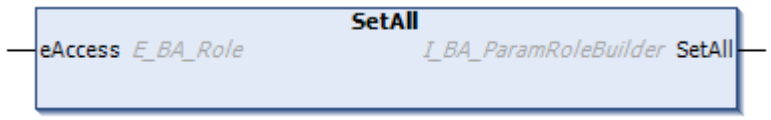
Illustration

```
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 [▶ 108]	BACnet property.
eAccess	E_BA_Role [▶ 99]	Definition of access rights.

6.1.SetAll
2.1.
3.2.
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 [▶ 99]	Definition of access rights.

6.1.SetRange
2.1.
3.2.
1.1.
3.2.
3

The method provided several BACnet properties depending on the user role.

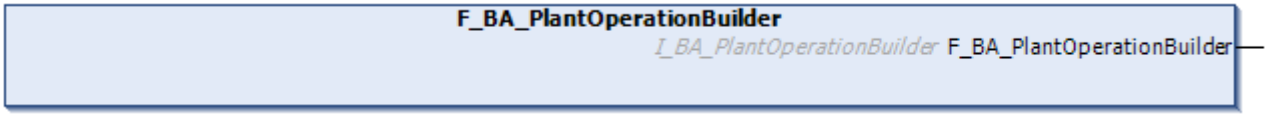
Illustration

```
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 [▶ 108]	First BACnet Property.
eParamTo	E_BA_Parameter [▶ 108]	Last BACnet Property.
eAccess	E_BA_Role [▶ 99]	Definition of access rights.

6.1.2.1.3.2.1.1.4 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 [▶ 186]	Adds an aggregate.
AddOperationMode [▶ 187]	Adds an operation mode.

See also:

FB_BA_PlantCtrl [▶ 480]

Requirements

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

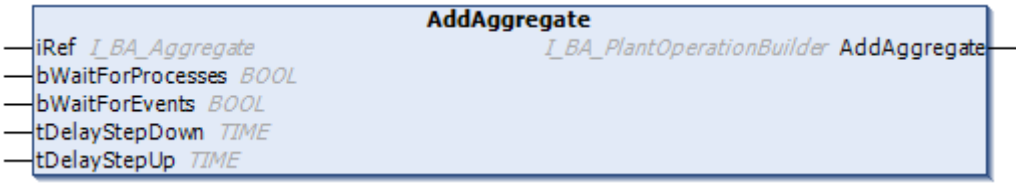
6.1.2.1.3.2.1.1.4.1 Management

6.1.Build
2.1.
3.2.
1.1.
4.1.
1

The method outputs the operating state [▶ 123] of the plant.

6.1.2.1.3.2.1.1.4.2 Value

6.1.AddAggregate
2.1.
3.2.
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.
3.2.
1.1.
4.2.
2

The method defines an operation mode for the plant operation.

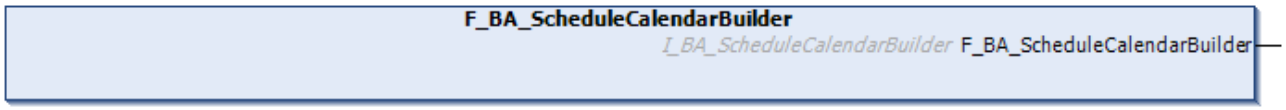
Illustration

```
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 [▶ 110]	Priority
eIgnore	E_BA_AggregateIgnoreFlags [▶ 106]	Defines the switching conditions.

6.1.2.1.3.2.1.1.5 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(Cal1)
                                        .AppendA(T#0H, 1.0).AppendA(T#6H, 2.0).AppendNull(T#12H)
                                        .Close()
                                    .Build(),
        );
```

Methods

Name	Description
AppendReference [▶ 188]	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

6.1.2.1.3.2.1.1.5.1 Management

6.1.Build
2.1.
3.2.
1.1.
5.1.
1

The method outputs the content of a schedule [▶ 122].

6.1.2.1.3.2.1.1.5.2 Value

6.1.AppendReference
2.1.
3.2.
1.1.
5.2.
1

The method passes a calendar reference.

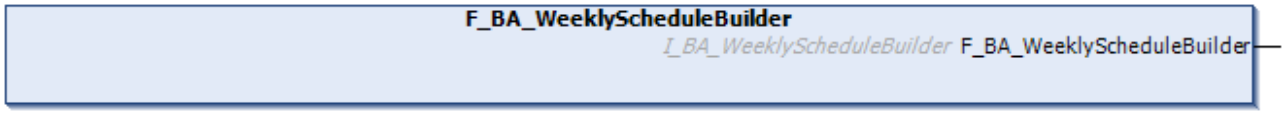
Illustration

```
METHOD AppendReference : I_BA_ScheduleCalendarEntryBuilder
VAR_INPUT
  iCalendar    : I_BA_Object;
END_VAR
```

Inputs

Name	Type	Description
iCalendar	I_BA_Object	Calendar reference.

6.1.2.1.3.2.1.1.6 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 [▶ 190]	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

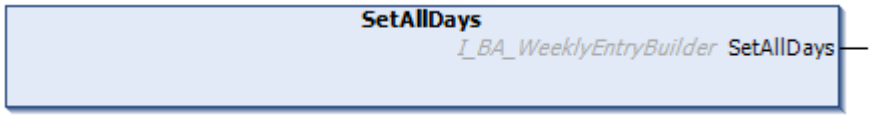
6.1.2.1.3.2.1.1.6.1 Management

6.1.Build
2.1.
3.2.
1.1.
6.1.
1

The method outputs the content of a weekly schedule [▶ 122].

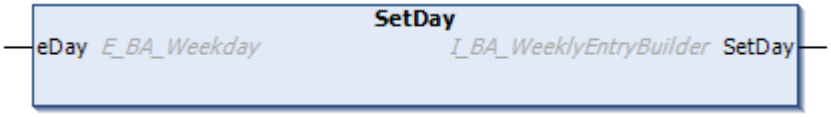
6.1.2.1.3.2.1.1.6.2 Value

6.1.SetAllDays
2.1.
3.2.
1.1.
6.2.
1



The method defines all days of the week for the weekly schedule.

6.1.SetDay
2.1.
3.2.
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.
3.2.
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.

6.1.2.1.3.3 Objects

6.1.2.1.3.3.1 Local

The following chapter provides information about the local server objects.

6.1.2.1.3.3.1.1 Analog

Description of the analog function blocks.

6.1.2.1.3.3.1.1.1 FB_BA_AI

The function block 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseAI [▶ 243]

Syntax

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

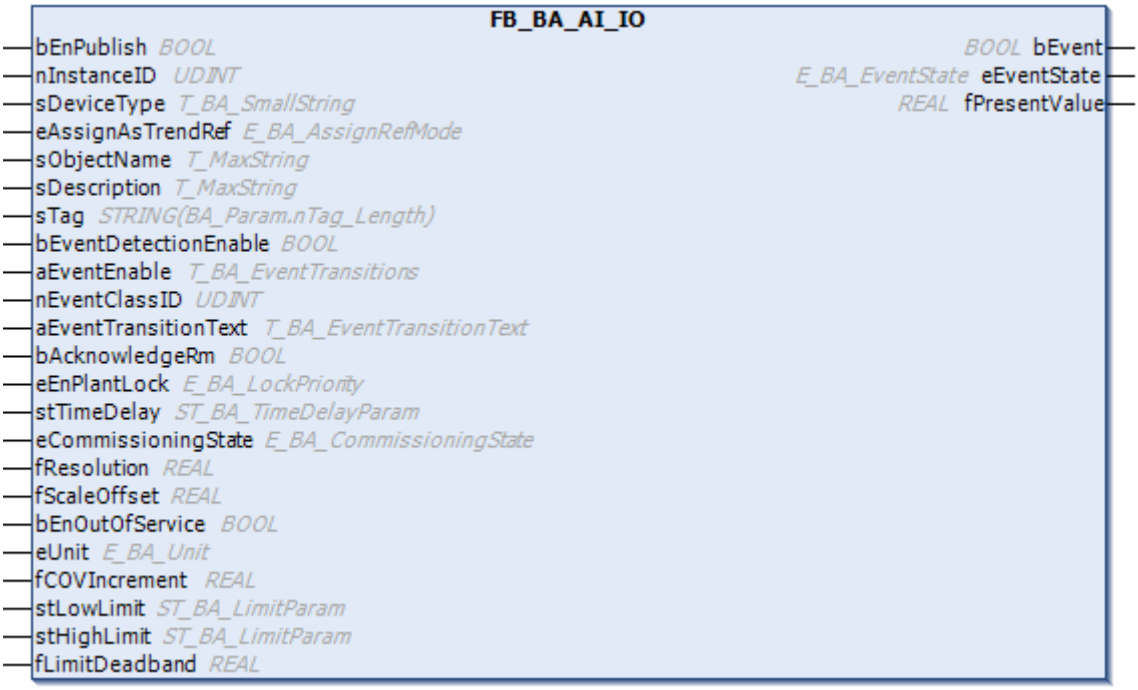
Inputs

Name	Type	Description
fVal	REAL	Scaled, analog input 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

6.1.2.1.3.3.1.1.2 FB_BA_AI_IO



The function block 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseAI [▶ 243]

Syntax

```
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

6.1.2.1.3.3.1.1.3 FB_BA_AI_IOEx

The function block 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseAI [▶ 243]

Syntax

```
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'}
    bConfigurate          : 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
bConfigurate	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

6.1.2.1.3.3.1.1.4 FB_BA_AI_Raw

The function block 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseAI [▶ 243]

Syntax

```
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

6.1.2.1.3.3.1.1.5 FB_BA_AO

The function block 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseAO [▶ 245]

Syntax

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

Inputs

Name	Type	Description
eRlbty	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

6.1.2.1.3.3.1.1.6 FB_BA_AO_IO

The function block 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseAO [▶ 245]

Syntax

```
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
eOverridenPolarity	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

6.1.2.1.3.3.1.1.7 FB_BA_AO_Raw

The function block 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 [▶ 34].

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseAO [▶ 245]

Syntax

```
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.

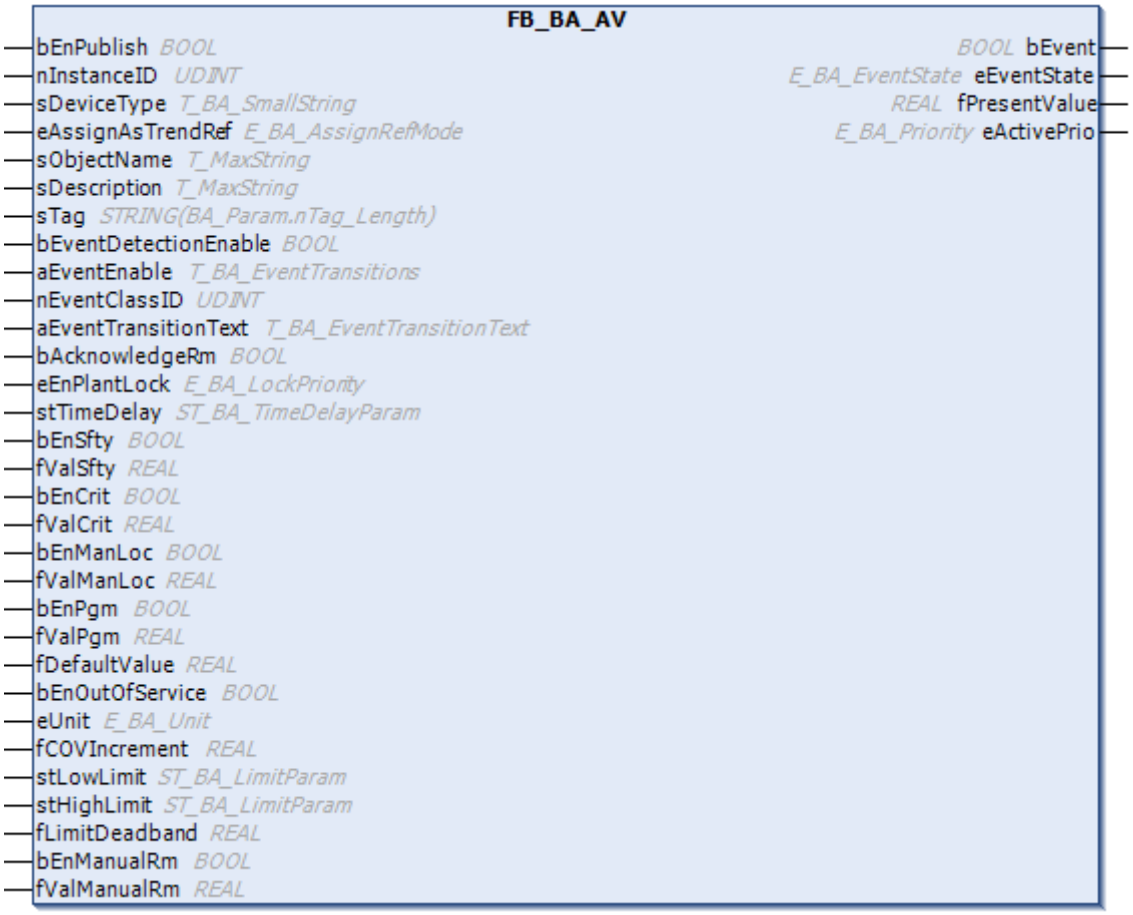
Outputs

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

6.1.2.1.3.3.1.1.8 FB_BA_AV



The function block represents an analog Value object.

It has a priority array and can be commanded [▶ 34].

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

Syntax

```
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 [▶ 110]	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 [▶ 120]	Parameterization of the lower limit value monitoring of an analog object. The variable *bEnable* must be TRUE to enable limit value monitoring. The variable *fValue* is used to parameterize the lower limit value.
stHighLimit	ST_BA_LimitParam [▶ 120]	Parameterization of the upper limit value monitoring of an analog object. The variable *bEnable* must be TRUE to enable limit value monitoring. The variable *fValue* is used to parameterize the upper limit value.
fLimitDeadband	REAL	Dead band or hysteresis for the values *HighLimit* and *LowLimit* 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

6.1.2.1.3.3.1.1.9 FB_BA_AV_Op

The function block 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 [▶ 264]

Syntax

```
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 [▶ 113]	The variable indicates whether an object of the type FB_BA_..._OP serves as a display or input object. *eVarInput* = 1 The object is used to display a value. The value is passed to the object at an input within the PLC. *eParameter* = 2 The object is used to enter a parameter that can be changed by a BACnet client or the TwinCAT HMI.

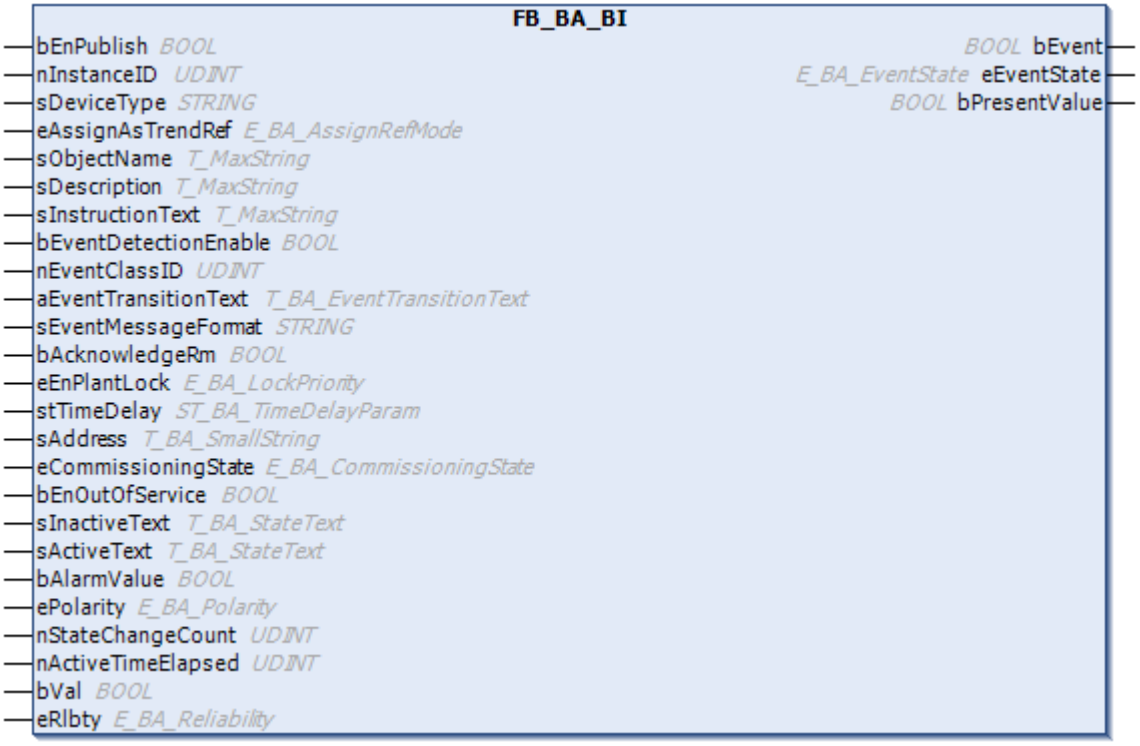
Requirements

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

6.1.2.1.3.3.1.2 Binary

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

6.1.2.1.3.3.1.2.1 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseBI [▶ 248]

Illustration

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

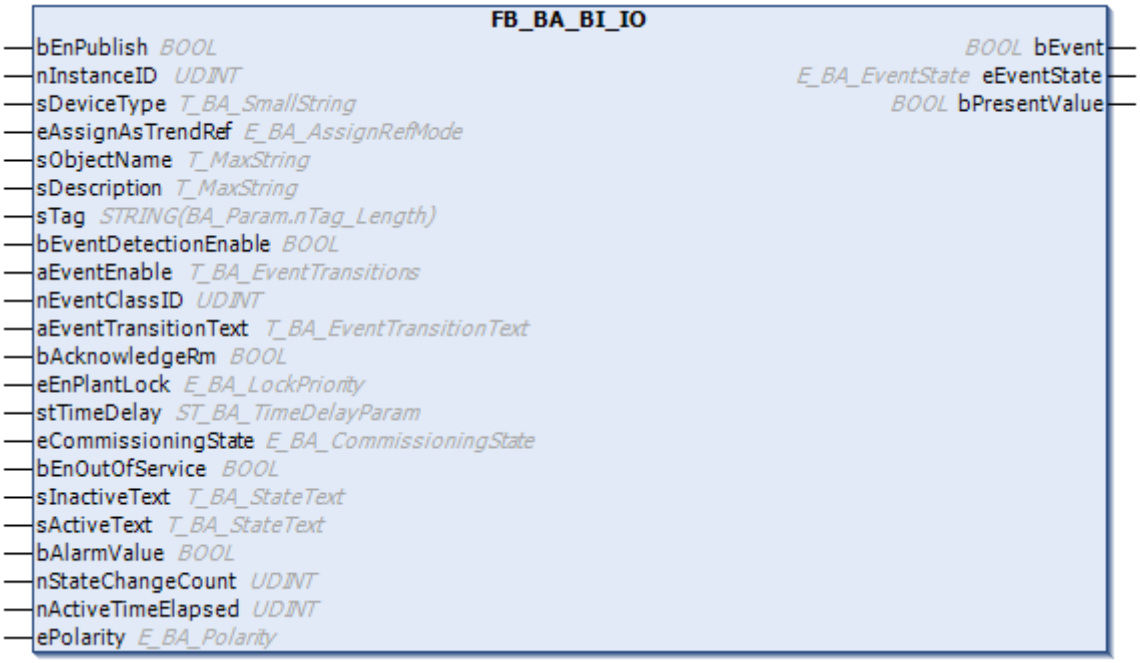
Inputs

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

6.1.2.1.3.3.1.2.2 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseBI [▶ 248]

FB_BA_BI [▶ 204]

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

6.1.2.1.3.3.1.2.3 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseBI [▶ 248]

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

6.1.2.1.3.3.1.2.4 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseBO [▶ 250]

Illustration

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

Inputs

Name	Type	Description
eRlbty	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

6.1.2.1.3.3.1.2.5 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseBO [▶ 250]

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.
eOverridenPolarity	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

6.1.2.1.3.3.1.2.6 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 [▶ 34].

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseBO [▶ 250]

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

6.1.2.1.3.3.1.2.7 FB_BA_BV

The function block FB_BA_BV represents a binary value object.

It has a priority array and can therefore be commanded [▶ 34].

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

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 [▶ 110]	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 [▶ 121]	Text output when the object is inactive.
sActiveText	T_BA_StateText [▶ 121]	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 *Present_Value* has changed since the date and the last reset.
nActiveTimeElapsed	UDINT	Time [s] in which the *Present_Value* of the object had the value ACTIVE. The time is valid from the last reset by the property *Time_Of_Active_Time_Reset*.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.
bValManualRm	BOOL	Binary value for the "Manual Remote" priority.

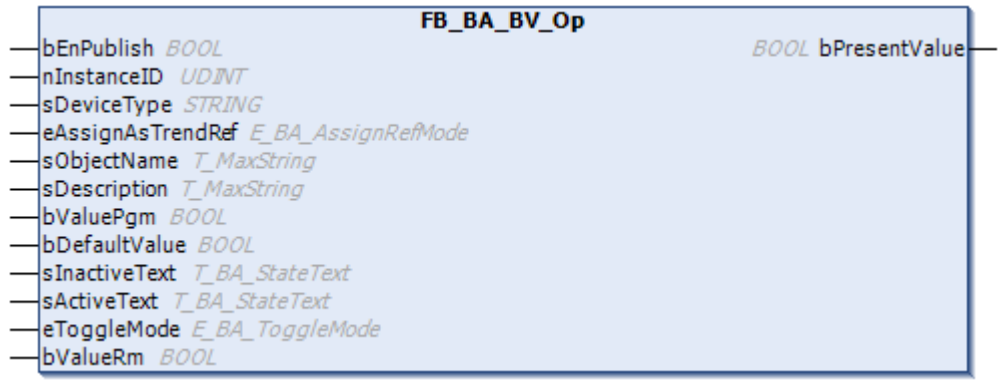
VAR

Name	Type	Description
stStateChangeTime	ST_BA_DateTime	This property shows the time of the last state change. The state change refers to the *Present_Value* 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

6.1.2.1.3.3.1.2.8 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 [▶ 264]

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 [▶ 113]	The variable indicates whether an object of the type FB_BA_..._OP serves as a display or input object. *eVarInput* = 1 The object is used to display a value. The value is passed to the object at an input within the PLC. *eParameter* = 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

6.1.2.1.3.3.1.3 Misc

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

6.1.2.1.3.3.1.3.1 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 [▶ 264]

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 [▶ 120]	Date list.

Inputs CONSTANT

Name	Type	Description
bWriteDateList	BOOL	Enable to write to a date list.

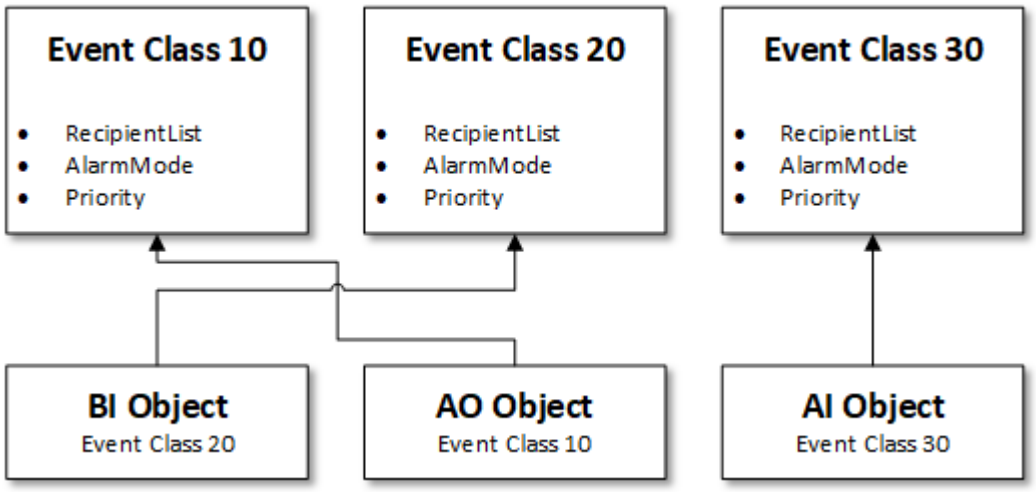
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.17	Tc3_BA2 from v4.8.9.0

6.1.2.1.3.3.1.3.2 FB_BA_EC

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 [▶ 264]

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 UDIN
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 [▶ 103]	This parameter is used to describe an event in more detail. The type of the event also describes the representation in the TwinCAT HMI [▶ 1120].
eAlarmMode	E_BA_AlarmMode [▶ 101]	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 [▶ 126].
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 [▶ 115]	Acknowledgement required.

Requirements

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

6.1.2.1.3.3.1.3.3 FB_BA_Loop

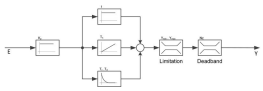
The function block FB_BA_Loop represents a PID controller within the project structure of TF8040.

Functional diagram

The controller can be operated either in parallel structure or with upstream P part. This is specified by the input *eOpMode*.

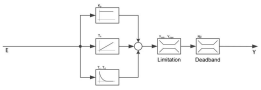
Upstream P part:

eOpMode := E_BA_PIDMode. eP1ID



Parallel structure:

eOpMode := E_BA_PIDMode. ePID



Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

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
  {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 *bEnSync* this value is written to *fPresentValue*.

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.

VAR

Name	Type	Description
fCtrlDeviation	REAL	Control deviation
eActionSource	E_BA_ProcessSignalSource [▶ 113]	Definition of whether the control direction is to be treated as a variable or as a parameter.
eMinMaxOutputSource	E_BA_ProcessSignalSource [▶ 113]	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

6.1.2.1.3.3.1.3.4 FB_BA_SchedA

The function block FB_BA_SchedA represents an analog scheduler within the project structure of TF8040.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_BaseSched [▶ 256]

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.

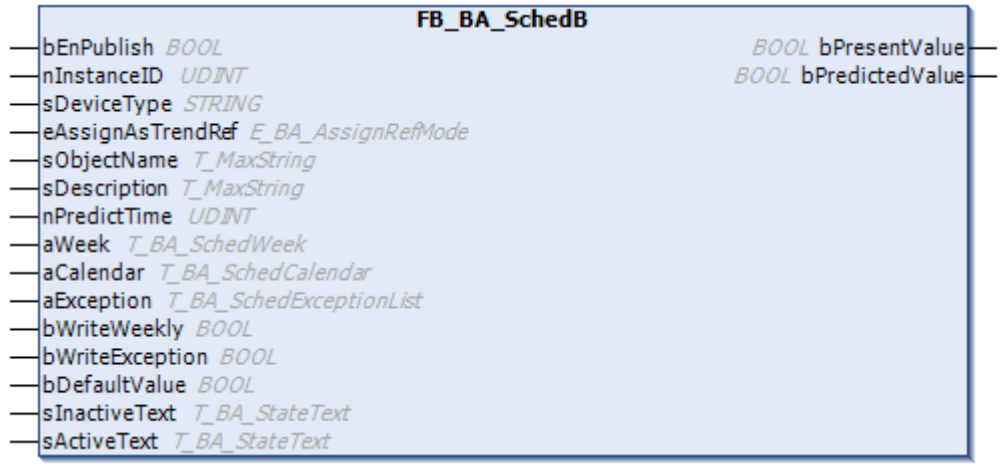
Inputs CONSTANT PERSISTENT

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

6.1.2.1.3.3.1.3.5 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 [▶ 264]

FB_BA_BaseSched [▶ 256]

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	Value that is assumed after the next switching.

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 [▶ 121]	Text output when the object is inactive.
sActiveText	T_BA_StateText [▶ 121]	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

6.1.2.1.3.3.1.3.6 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 [▶ 264]

FB_BA_BaseSched [▶ 256]

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 [▶ 121]	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

6.1.2.1.3.3.1.3.7 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 [▶ 264]

FB_BA_EventObject [▶ 261]

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.

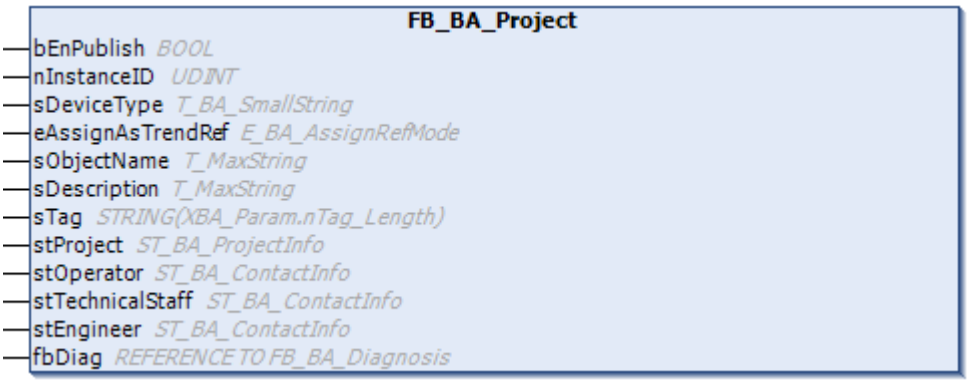
VAR_INPUT CONSTANT PERSISTENT

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 [▶ 120]	Parameter object.

Requirements

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

6.1.2.1.3.3.1.3.8 FB_BA_Project



The function block provides the ability to describe the project in more detail using optional information.

The information provided can be accessed by clients (e.g. HMI or Site Explorer) and processed accordingly (e.g. as a project-related information overview of contact persons, etc.).

The general diagnostic functions (see XBA_Globals.Diag [▶ 124]) are also referenced.

This function block may only be instantiated once in the project.

The instance ID always corresponds to that of the BACnet device, even if a different ID has been pre-initialized.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

Syntax

```
FUNCTION_BLOCK FB_BA_Project EXTENDS FB_BA_Object
VAR_INPUT CONSTANT
  {region 'Project'}
    stProject           : ST_BA_ProjectInfo;
  {endregion}
  {region 'Contact'}
    stOperator          : ST_BA_ContactInfo;
    stTechnicalStaff    : ST_BA_ContactInfo;
    stEngineer          : ST_BA_ContactInfo;
  {endregion}
END_VAR
```

Inputs

Name	Type	Description
stProject	ST_BA_ProjectInfo [▶ 117]	Further information on the project (name, description, location).
stOperator	ST_BA_ContactInfo [▶ 116]	Contact information of the operator (name, telephone number, e-mail address, web pages).
stTechnicalStaff	ST_BA_ContactInfo [▶ 116]	Contact information of the technical staff (name, telephone number, e-mail address, web pages).
stEngineer	ST_BA_ContactInfo [▶ 116]	Contact information of the integrator (name, telephone number, e-mail address, web pages).

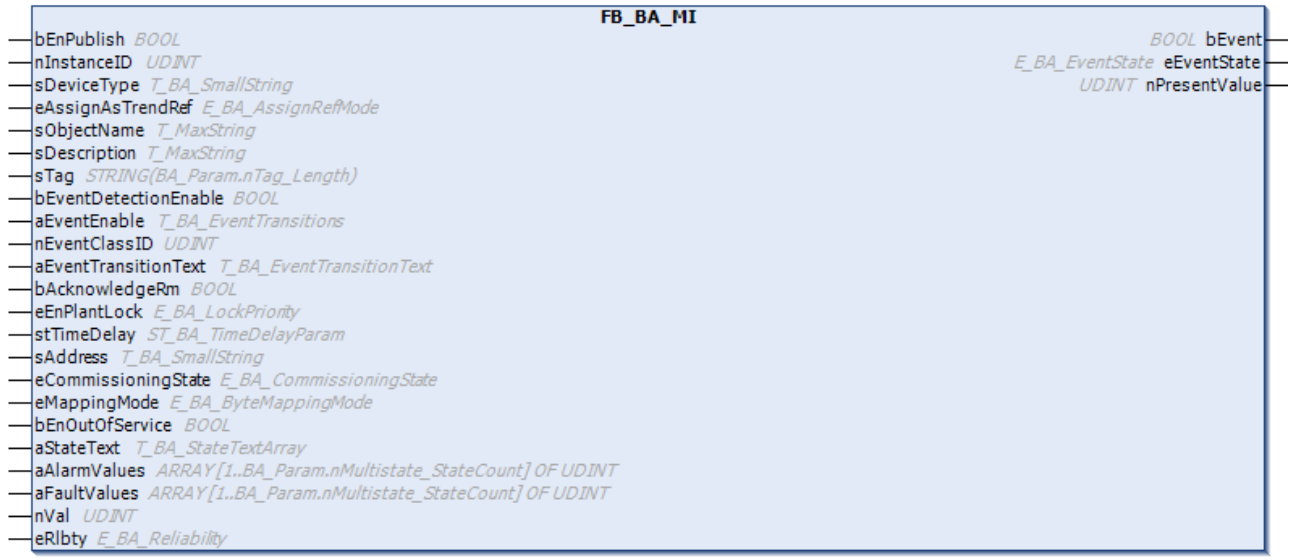
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.2.1.3.3.1.4 Multistate

Reporting, switching and processing of multi-state values.

6.1.2.1.3.3.1.4.1 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseMI [▶ 253]

Illustration

```
FUNCTION_BLOCK FB_BA_MI EXTENDS FB_BA_BaseMI
VAR_INPUT
  nVal      : UDINT := 1;
  eRlbt     : E_BA_Reliability;
END_VAR
```

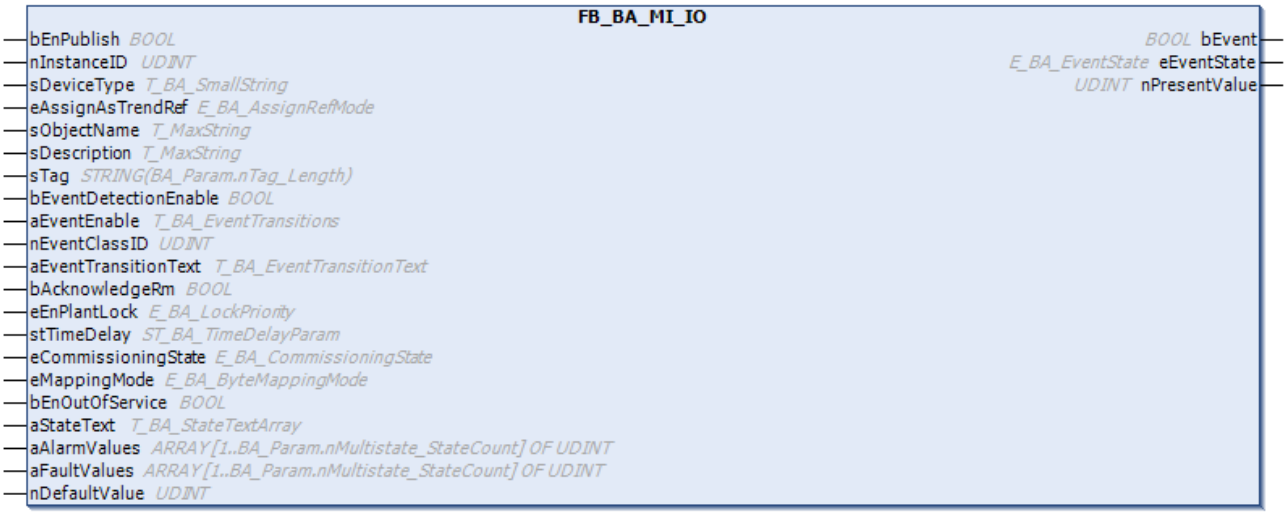
Inputs

Name	Type	Description
nVal	UDINT	Scaled, analog input value.
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

6.1.2.1.3.3.1.4.2 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseMI [▶ 253]

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.

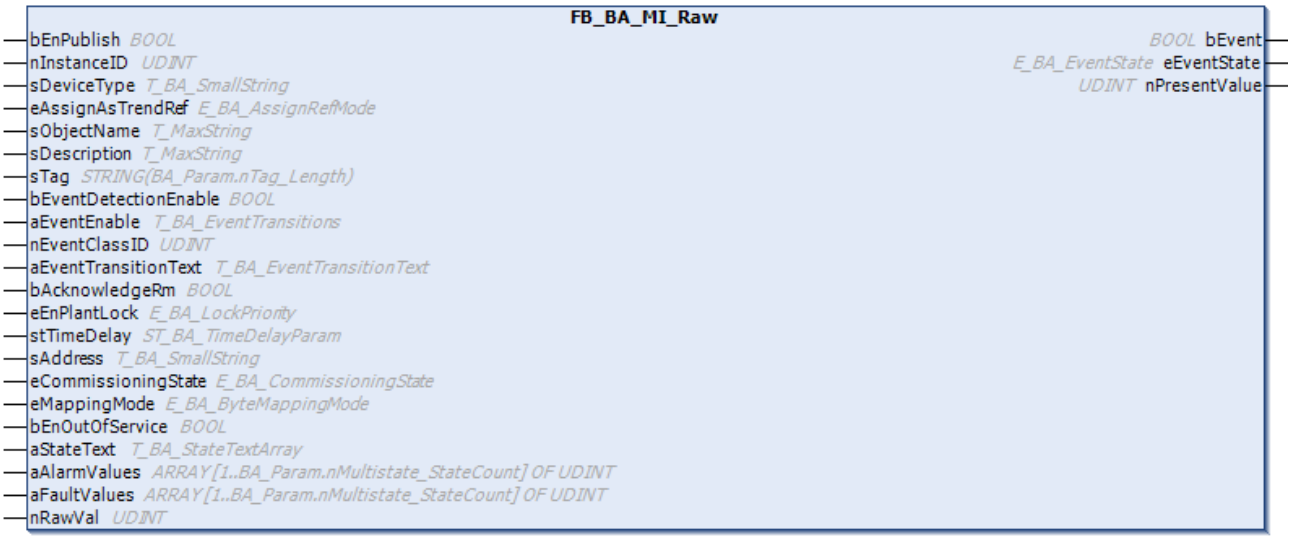
VAR

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

6.1.2.1.3.3.1.4.3 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseMI [▶ 253]

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

6.1.2.1.3.3.1.4.4 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseMO [▶ 254]

Illustration

```
FUNCTION_BLOCK FB_BA_MO EXTENDS FB_BA_BaseMO
VAR_INPUT
  eRlbty          : E_BA_Reliability;
  bRawOvrrd       : BOOL;
  bEnFdbck        : BOOL;
  nRawValFdbck    : UDINT;
END_VAR
```

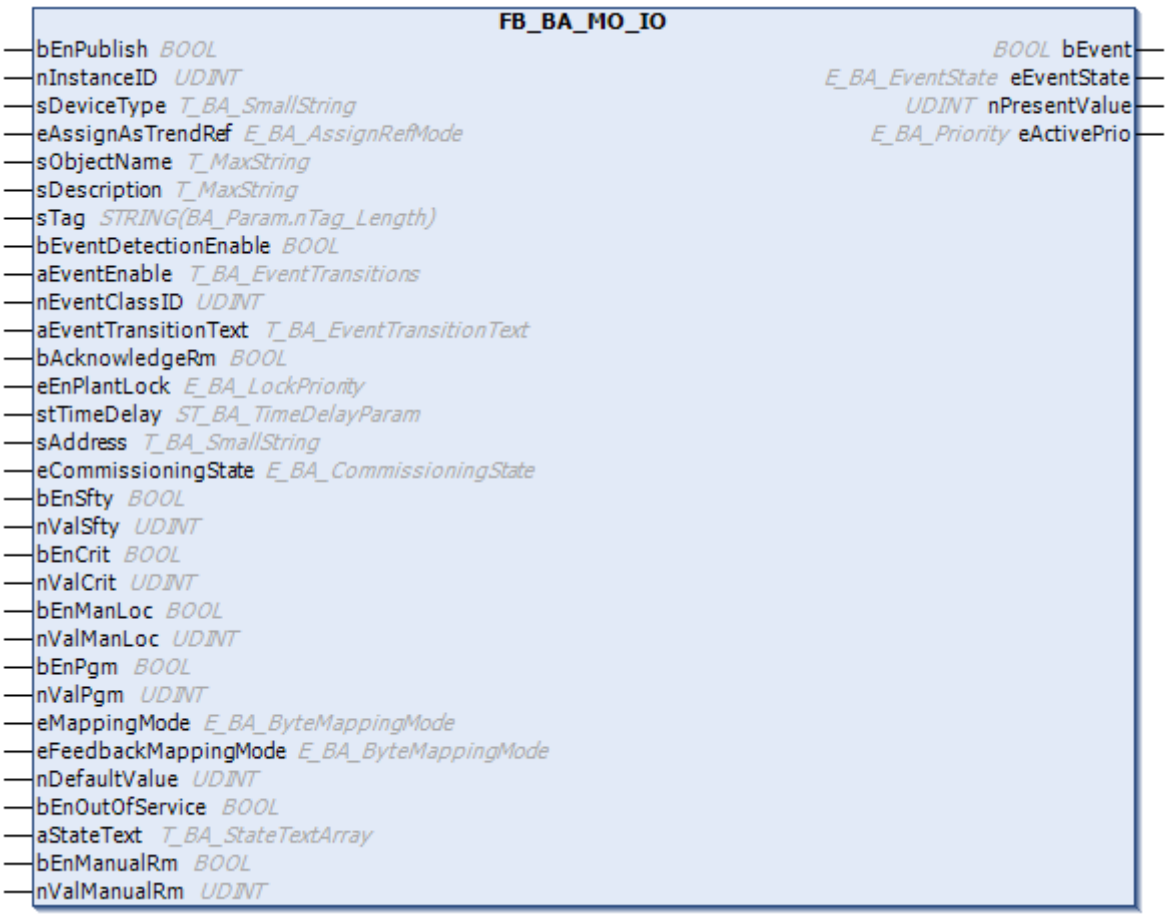
Inputs

Name	Type	Description
eRlbty	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

6.1.2.1.3.3.1.4.5 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseMO [▶ 254]

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

6.1.2.1.3.3.1.4.6 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

FB_BA_BaseMO [▶ 254]

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.

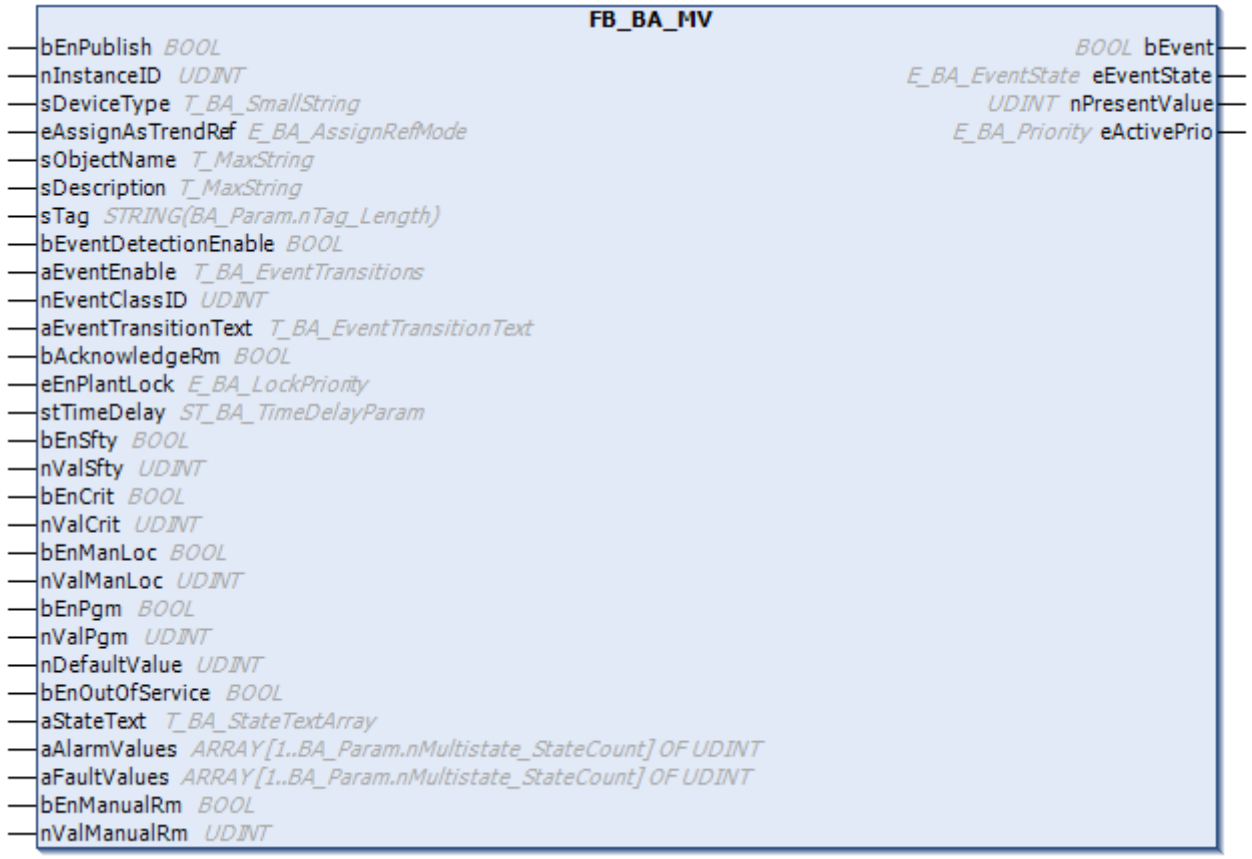
Outputs

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

6.1.2.1.3.3.1.4.7 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

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 [▶ 110]	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 [▶ 121]	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.

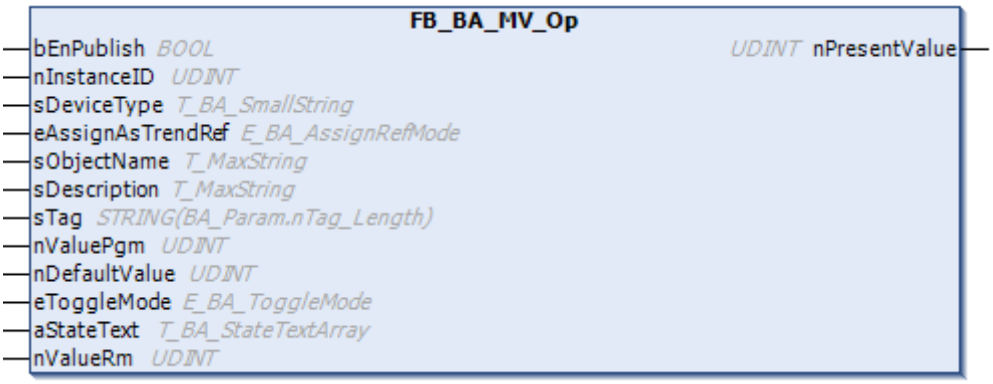
VAR

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

6.1.2.1.3.3.1.4.8 FB_BA_MV_Op



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 [▶ 264]

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 *bPresentValue* of an object is generated depending on the input *bValuePgm*.
aStateText	T_BA_StateTextArray [▶ 121]	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

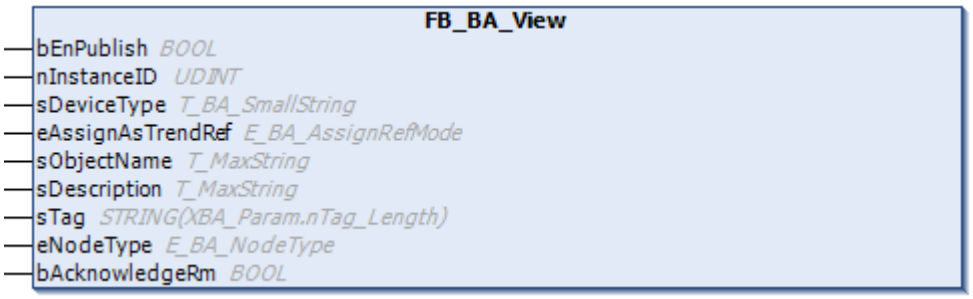
Name	Type	Description
nStateCount	UDINT	Number of states of a multi-state object.
eValueSource	E_BA_ProcessSignalSource [▶ 113]	The variable indicates whether an object of the type FB_BA_..._OP serves as a display or input object. *eVarInput* = 1 The object is used to display a value. The value is passed to the object at an input within the PLC. *eParameter* = 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

6.1.2.1.3.3.1.5 Structurize

6.1.2.1.3.3.1.5.1 FB_BA_View



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 [▶ 264]

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_NodeType.Automatic;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Operational Parameters'}
    bAcknowledgeRm      : BOOL;
  {endregion}
END_VAR
VAR
  {region 'SubInit'}
    eDPADMode           : E_BA_DPADMode := E_BA_DPADMode.Undefined;
  {endregion}
  {region 'Events'}
    fbActiveEvents      : FB_BA_EventIndicator;
  {endregion}
END_VAR
```

Inputs CONSTANT PERSISTENT

Name	Type	Description
eNodeType	E_BA_NodeType [▶ 111]	The variable *NodeType* 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 [▶ 101]	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 *FB_BA_View* 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 *eDPADMode*.
fbActiveEvents	FB_BA_EventIndicator	Gives information about events of all objects of a View object.

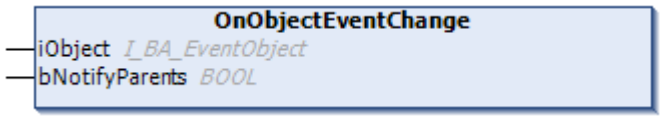
Methods

Name	Description
OnObjectEventChange [▶ 242]	Change of an event in a sub-object.

Requirements

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

6.1.2.1.3.3.1.5.1.1 OnObjectEventChange



The method is called when the event of a subordinate object changes.

The user can respond to the changes in this method.

Syntax

```
METHOD PROTECTED OnObjectEventChange
VAR_INPUT
  iObject           : I_BA_EventObject;
  bNotifyParents    : BOOL := TRUE;
END_VAR
```

Inputs

Name	Type	Description
iObject	I_BA_EventObject	Interface to the context object. [▶ 264]
bNotifyParents	BOOL	Enable for forwarding the event to the next higher-level object. This enable must be transferred to the basic function block. Sample: `SUPER^.OnObjectEventChange(iObject, TRUE);`

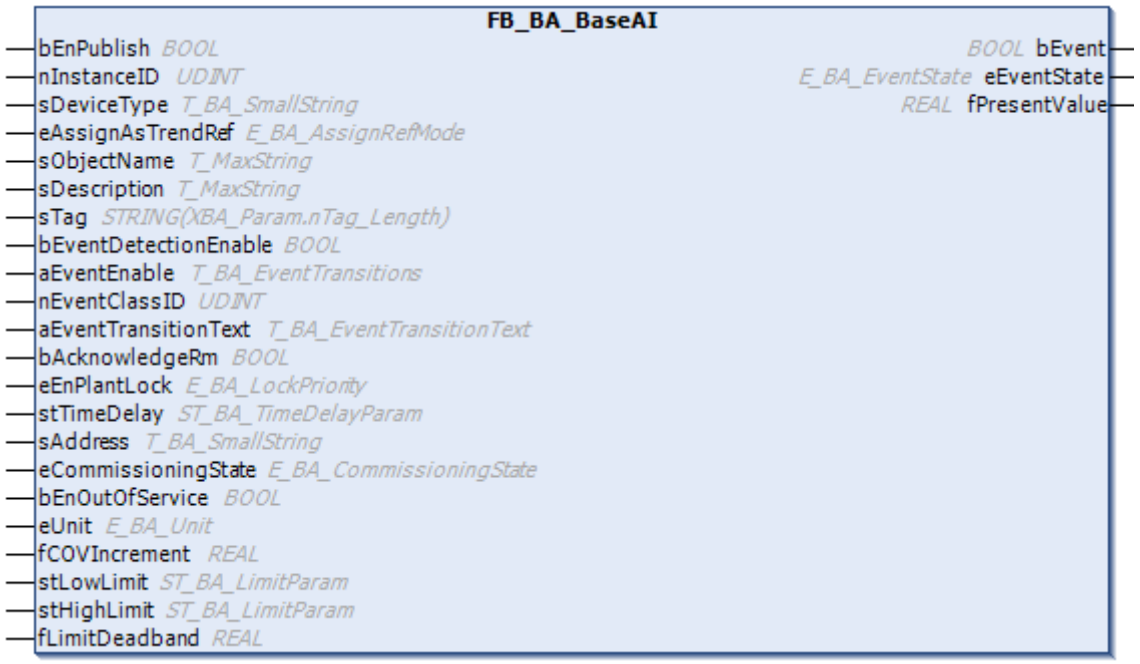
Requirements

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

6.1.2.1.3.3.1.6 Base

These objects form the basis of the previously described function blocks. They inherit the local function blocks.

6.1.2.1.3.3.1.6.1 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

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 [▶ 126]	REAL	Resolution of an analog signal for scaling a measured value.
fScaleOffset [▶ 126]	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 [▶ 120]	Parameterization of the lower limit value monitoring of an analog object. The variable *bEnable* must be TRUE to enable limit value monitoring. The variable *fValue* is used to parameterize the lower limit value.
stHighLimit	ST_BA_LimitParam [▶ 120]	Parameterization of the upper limit value monitoring of an analog object. The variable *bEnable* must be TRUE to enable limit value monitoring. The variable *fValue* is used to parameterize the upper limit value.
fLimitDeadband	REAL	Dead band or hysteresis for the values *HighLimit* and *LowLimit* 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

6.1.2.1.3.3.1.6.2 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

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 [▶ 110]	Active priority

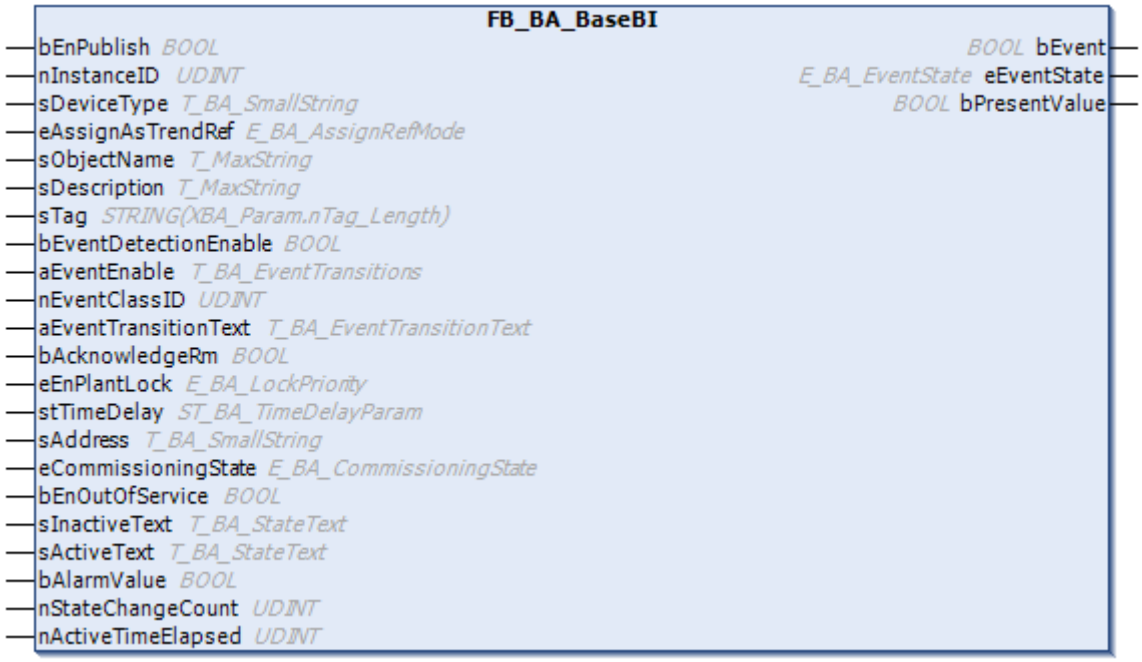
Inputs CONSTANT PERSISTENT

Name	Type	Description
fResolution [▶ 126]	REAL	Resolution of an analog signal for scaling a measured value.
fScaleOffset [▶ 126]	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 [▶ 120]	Parameterization of the lower limit value monitoring of an analog object. The variable *bEnable* must be TRUE to enable limit value monitoring. The variable *fValue* is used to parameterize the lower limit value.
stHighLimit	ST_BA_LimitParam [▶ 120]	Parameterization of the upper limit value monitoring of an analog object. The variable *bEnable* must be TRUE to enable limit value monitoring. The variable *fValue* is used to parameterize the upper limit value.
fLimitDeadband	REAL	Dead band or hysteresis for the values *HighLimit* and *LowLimit* 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

6.1.2.1.3.3.1.6.3 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

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 [▶ 121]	Text output when the object is inactive.
sActiveText	T_BA_StateText [▶ 121]	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 *Present_Value* has changed since the date and the last reset.
nActiveTimeElapsed	UDINT	Time [s] in which the *Present_Value* of the object had the value ACTIVE. The time is valid from the last reset by the property *Time_Of_Active_Time_Reset*.

VAR

Name	Type	Description
stStateChangeTime	ST_BA_DateTime	This property shows the time of the last state change. The state change refers to the *Present_Value* 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

6.1.2.1.3.3.1.6.4 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

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 [▶ 110]	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 [▶ 121]	Text output when the object is inactive.
sActiveText	T_BA_StateText [▶ 121]	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 *Present_Value* has changed since the date and the last reset.
nActiveTimeElapsed	UDINT	Time [s] in which the *Present_Value* of the object had the value ACTIVE. The time is valid from the last reset by the property *Time_Of_Active_Time_Reset*.
bValManualRm	BOOL	Binary value for the "Manual Remote" priority.
bEnManualRm	BOOL	Enabling the "Manual Remote" priority.

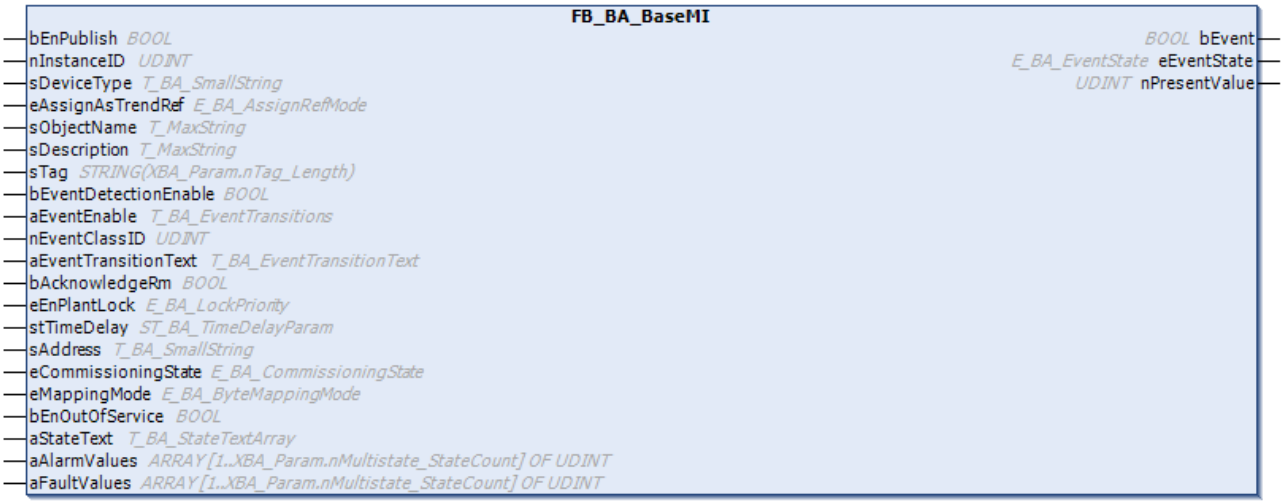
VAR

Name	Type	Description
stStateChangeTime	ST_BA_DateTime	This property shows the time of the last state change. The state change refers to the *Present_Value* 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

6.1.2.1.3.3.1.6.5 FB_BA_BaseMI



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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

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 [▶ 121]	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

6.1.2.1.3.3.1.6.6 FB_BA_BaseMO

The function block FB_BA_BaseMO represents a multi-level output.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

FB_BA_ComEventObject [▶ 260]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseMO EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_MultistateOutObjec
t, 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 [▶ 110]	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 [▶ 121]	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.

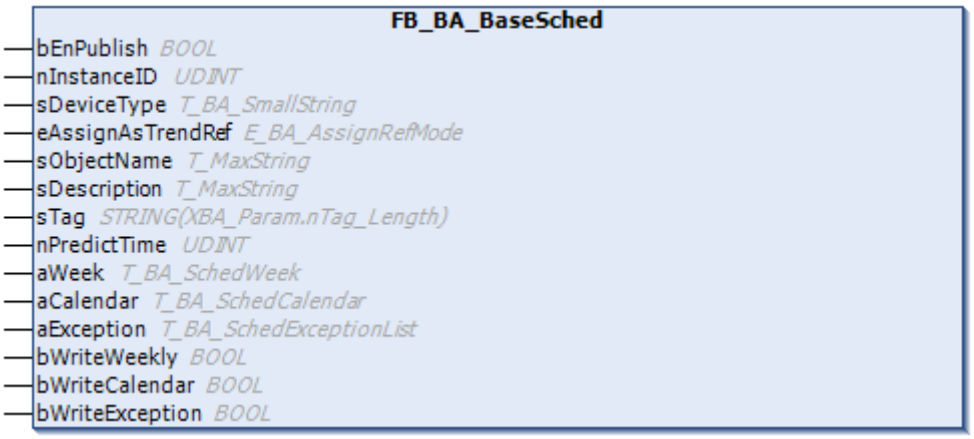
VAR

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

6.1.2.1.3.3.1.6.7 FB_BA_BaseSched



The function block FB_BA_BaseSched forms the base for scheduler function blocks.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

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 [▶ 122]	Weekly scheduler.
aCalendar	T_BA_SchedCalendar [▶ 122]	Calendar.
aException	T_BA_SchedExceptionList [▶ 122]	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

6.1.2.1.3.3.1.6.8 FB_BA_BaseStateMV

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 [▶ 264]

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 [▶ 112]	Defines the reference to the object whose description is displayed to represent the state.
aStateText	T_BA_StateTextArray [▶ 121]	The array is used to declare the state texts of a multi-state object.

VAR

Name	Type	Description
stActiveInfo	ST_BA_ActiveInfo [▶ 122]	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

6.1.2.1.3.3.1.6.9 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 [▶ 264]

FB_BA_BaseStateMV [▶ 257]

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

6.1.2.1.3.3.1.6.10 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 [▶ 264]

FB_BA_EventObject [▶ 261]

FB_BA_EventObjectEx [▶ 263]

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
```

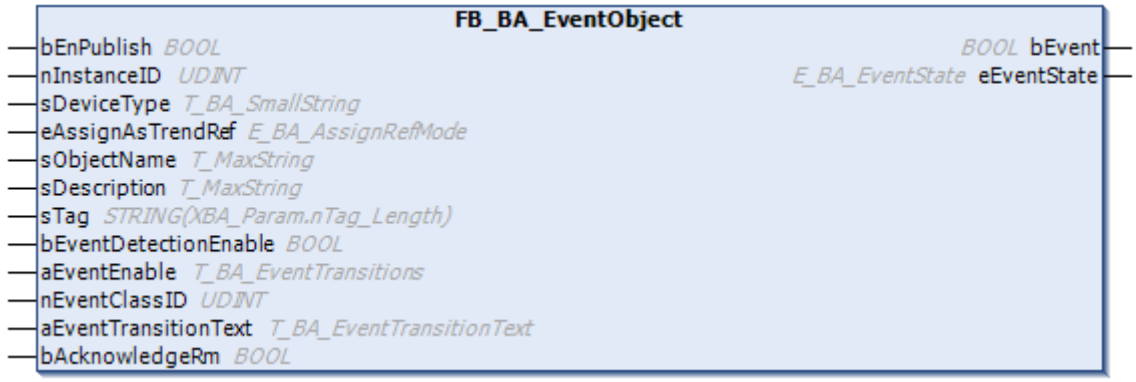
Inputs CONSTANT PERSISTENT

Name	Type	Description
eCommissioningState	E_BA_CommissioningState [▶ 107]	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

6.1.2.1.3.3.1.6.11 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

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_EventObject EXTENDS FB_BA_Object IMPLEMENTS I_BA_EventObject, I_BA_Eve
ntValue
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.
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 [▶ 32])
aEventTransitionText	T_BA_EventTransitionText [▶ 115]	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 *TxtEvent_EN* or *TxtEvent_DE*. 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.

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 [▶ 103]	Event type.
eAlarmMode	E_BA_AlarmMode [▶ 101]	Definition of how to deal with an alarm.
eReliability	E_BA_Reliability	This variable indicates whether the value of the *Present_Value* 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 *BACnetEventTransitionBits*.
stStateFlags	ST_BA_StatusFlags	Mapping of the BACnet property *Status_Flags*.
stAckedTransitions	ST_BA_EventTransitions	Mapping of the BACnet Properties *Acked_Transitions*.
aEventEnable	T_BA_EventTransitions [▶ 115]	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

6.1.2.1.3.3.1.6.12 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 [▶ 264]

FB_BA_EventObject [▶ 261]

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 [▶ 110]	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).
stTimeDelay	ST_BA_TimeDelayParam [▶ 120]	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 *PLC BA-Template* 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 *nDefTimeDelay_ToAbnormal* and *nDefTimeDelay_ToNormal*.

Requirements

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

6.1.2.1.3.3.1.6.13 FB_BA_Object

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 [▶ 264]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_Object EXTENDS FB_BA_BasePublisher IMPLEMENTS I_BA_Object, I_BACnet_Ob
jectOwner
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
t level 0)
    eState               : E_BA_ObjectState := E_BA_ObjectState.First;
    sFmtPresentValue     : STRING;// Present value in a formated 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 [▶ 120]	This property is a text description of the physical device connected to the analog output.
eAssignAsTrendRef	E_BA_AssignRefMode [▶ 111]	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 [▶ 42]).
sTag	STRING	This property represents an additional description text for the object. The string *sTag* 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 [▶ 112]	Selection of the object type.
stAttributes	ST_BA_ObjectAttributes [▶ 114]	Attribute definition.
nLevel	UINT	Current hierarchical level according to the base object (level 0).
eState	E_BA_SubscriberState [▶ 113]	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

6.1.2.2 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
);
END_TYPE
```

Name	Description
eNoMedium	No medium.
eHeatMedium	Heating medium.
eCoolMedium	Cooling medium.

6.1.2.2.1.1.1.1.1.2 E_BA_PipeSys

```
TYPE E_BA_PipeSys :
(
  Invalid                 := 0,
  e2Pipe                  := 1,
  e4Pipe                  := 2
);
END_TYPE
```

Name	Description
e2Pipe	Two-pipe system means that heating and cooling media are used alternatively in one pipe system.
e4Pipe	In the four-pipe system, the heating and cooling media have a separate pipe system.

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 [▶ 396] via the value *fFixPos* [%].
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 [▶ 340]).
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_ShdObjType

Enumerator for selecting the shading object type.

```
TYPE E_BA_ShdObjType :
(
  Invalid    := 0,
  eTetragon  := 1,
  eGlobe     := 2
) BYTE;
END_TYPE
```

Name	Description
eTetragon	Object type is a rectangle.
eGlobe	Object type is a ball.

6.1.2.2.1.1.1.3.3 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_StatePIDControlSync

E_BA_StatePIDControlSync shows the state of the synchronization of FB_BA_PIDControlSync [▶ 424].

```
TYPE E_BA_StatePIDControlSync :
(
  eOk                          := 1,
  eErrAction                   := 2,
  eErrDampConstant             := 3,
  eErrDerivativeConstant       := 4,
  eErrIntegralConstant         := 5,
  eErrMaxOutput                := 6,
  eErrMinOutput                := 7,
  eErrNeutralZone              := 8,
  eErrOpMode                   := 9,
  eErrProportionalConstant     := 10
) UDINT;
END_TYPE
```

Name	Description
eOK	Synchronization is error-free.
eErrAction	Incorrect synchronization of the VAR_INPUT CONSTANT variable *eAction* at FB_BA_PIDControl [▶ 420].
eErrDampConstant	Incorrect synchronization of the VAR_INPUT CONSTANT variable *nDampConstant* at FB_BA_PIDControl [▶ 420].
eErrDerivativeConstant	Incorrect synchronization of the VAR_INPUT CONSTANT variable *fDerivativeConstant* at FB_BA_PIDControl [▶ 420].
eErrIntegralConstant	Incorrect synchronization of the VAR_INPUT CONSTANT variable *fIntegralConstant* on FB_BA_PIDControl [▶ 420].
eErrMaxOutput	Incorrect synchronization of the VAR_INPUT CONSTANT variable *fMaxOutput* at FB_BA_PIDControl [▶ 420].
eErrMinOutput	Incorrect synchronization of the VAR_INPUT CONSTANT variable *fMinOutput* at FB_BA_PIDControl [▶ 420].
eErrNeutralZone	Incorrect synchronization of the VAR_INPUT CONSTANT variable *fNeutralZone* at FB_BA_PIDControl [▶ 420].
eErrOpMode	Incorrect synchronization of the VAR_INPUT CONSTANT variable *eOpMode* at FB_BA_PIDControl [▶ 420].
eErrProportionalConstant	Incorrect synchronization of the VAR_INPUT CONSTANT variable *fProportionalConstant* on FB_BA_PIDControl [▶ 420].

6.1.2.2.1.1.2.4 Schedule

6.1.2.2.1.1.2.4.1 E_BA_CtrlFct

The FB_BA_OptimizedOn and FB_BA_OptimizedOff optimization functions perform different internal calculations depending on whether the system to be controlled is a heating or cooling system.

```
{attribute 'qualified_only'}
TYPE E_BA_CtrlFct :
(
  eCtrlFct_Off       := 0,
  eCtrlFct_Heating   := 1,
```

```
  eCtrlFct_Cooling   := 2
);
END_TYPE
```

Name	Description
eCtrlFct_Off	No selection made. The optimization function will not start.
eCtrlFct_Heating	The optimization function considers the heating mode.
eCtrlFct_Cooling	The optimization function considers the cooling mode.

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 Functions

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	Transferred light value [%] in absolute value control mode.
fLgtT	REAL	Transferred light temperature [K].
bActv	BOOL	The sender of the telegram is active. This bit is only evaluated by the priority control.
ePrio	E_BA_LightingPrio [▶ 268]	Priority of the active telegram. This enumeration is only evaluated by the priority control.
nEvtInc	ULINT	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 Sun Protection

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 [m].
fWdwHght	REAL	Height of the window [m].
aCnr	ST_BA_Cnr [▶ 273]	Coordinates of the window corners and information as to whether this corner point is in the shade.
nGrp	DINT	Specification of the group to which the window belongs.
bVld	BOOL	Plausibility of the entered data: *bVld* = 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: *bShdd* = 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 [▶ 383], FB_BA_SunBldPrioSwi8 [▶ 384], FB_BA_SunBldTgmSel4 [▶ 390], FB_BA_SunBldTgmSel8 [▶ 391] evaluated. The sun protection actuators FB_BA_SunBldActr [▶ 374] and FB_BA_RolBldActr [▶ 365] ignore it.
ePrio	E_BA_SunBldPrio [▶ 269]	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_ShdObj

List entry for a shading object.

```
TYPE ST_BA_ShdObj :
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_ShdObjType;
  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_ShdObjType [▶ 269]	Object type
bVld	BOOL	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).

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 Aggregates

6.1.2.2.1.2.2.1.1 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.

6.1.2.2.1.2.2.1.2 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.

6.1.2.2.1.2.2.1.3 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 [▶ 270]	Enum of the "Safety" priority to be written.
bEnCrit	BOOL	Enable for writing the "Critical" priority.
eValCrit	E_BA_Mdlt [▶ 270]	Enum of the priority "Critical" to be written.
bEnPgm	BOOL	Enable for writing the "Program" priority.
eValPgm	E_BA_Mdlt [▶ 270]	Enum of priority "Program" to be written.

6.1.2.2.1.2.2.1.4 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	Type	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.

6.1.2.2.1.2.2.1.5 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 [▶ 473] / FB_BA_StepMdlt [▶ 475].
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 [▶ 473] / FB_BA_StepMdlt [▶ 475].
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 *nRemSecOff* has elapsed, the step is switched off as well as *bEnDown* and the next step in the falling switch-off sequence becomes the active one.
bRdyUp	BOOL	After expiration of the start-up delay *nRemSecOn* and a TRUE at the feedback *bFdb* of the step sequence function block, *bRdyUp* is set and the next step is activated.
bRdyDown	BOOL	After the switch-off delay *nRemSecOff* has expired, *bRdyDown* 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.

6.1.2.2.1.2.2.1.6 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.

6.1.2.2.1.2.2.2 PlantControl

6.1.2.2.1.2.2.2.1 ST_BA_Aggregate

The aggregate structure is used for bidirectional communication between the connected aggregate FB_BA_Aggregate [▶ 478] and the plant control FB_BA_PlantControl [▶ 480].

The structure contains the switching values, the plant profile and the current step of the step sequence control.

The plant control receives feedback from the aggregate for the step sequence control.

Syntax

```
TYPE ST_BA_Aggregate:
STRUCT
  nStep               : UDINT;
  nPrio               : UDINT;
  nProfile            : UDINT;
  fValue              : REAL;
  nValue              : UDINT;
  bValue              : BOOL;
  bUp                 : BOOL;
  bDown               : BOOL;
  bWatchdog           : BOOL;
  bFdbUp              : BOOL;
  bFdbDown            : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
nStep	UDINT	Indicates the current step of the step sequence control.
nPrio	UDINT	Shows the active priority of the profile.
nProfile	UDINT	Indicates which profile is active.
fValue	REAL	Transmits the analog switching value to the aggregate.
nValue	UDINT	Transmits the numerical switching value to the aggregate.
bValue	BOOL	Transmits the Boolean switching value to the aggregate.
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.
bWatchdog	BOOL	Bidirectional monitoring of communication between the aggregate and the plant control. If an aggregate is not present within the step sequence control, the step sequence control cannot run into this step.
bFdbUp	BOOL	Feedback "Up" from the respective aggregate to the plant control.
bFdbDown	BOOL	Feedback "Down" of the respective aggregate to the plant control.

6.1.2.2.1.2.2.2.2 ST_BA_ProfileDescription

The parameterization structure is used to describe a profile within FB_BA_PlantControl [▶ 480].

Syntax

```
TYPE ST_BA_ProfileDescription:
STRUCT
  sDescription    : T_MaxString;
  nPrio           : UDINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
sDescription	T_MaxString	The plant profile in question in plain text. For display purposes only.
nPrio	UDINT	The priority is used to command all aggregates of a plant. The priority is transferred from the plant control program FB_BA_PlantControl [▶ 480] within the ST_BA_Aggregate [▶ 278] to the aggregates and output at the function block FB_BA_Aggregate [▶ 478]. Within the aggregate templates, this priority is used to command the objects.

6.1.2.2.1.2.2.2.3 ST_BA_ProfileParameter

The parameterization structure is used within the plant control FB_BA_PlantControl [▶ 480] for the parameterization of the parameters of a step or an aggregate. The structure contains the binary, numerical and analog switching value for the aggregate, the delay times and the enable for the use of the feedback signals from the aggregates.

Syntax

```
TYPE ST_BA_PlantControl_ParamSet:
STRUCT
  sStep          : T_MaxString;
  sProfile       : T_MaxString;

  fDelayUp       : REAL;
  fDelayDown     : REAL;

  fValue         : REAL;
  nValue         : UDINT;
  bValue         : BOOL;

  bEnFdbUp       : BOOL;
  bEnFdbDown     : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
sStep	T_MaxString	The plant step in question in plain text. The content of the text is specified via the *arrStepDescription* parameter structure in the function block FB_BA_PlantControl [▶ 480].
sProfile	T_MaxString	The profile in question in plain text. The content of the text is specified via the *arrProfileDescription* structure in the function block FB_BA_PlantControl [▶ 480].
fDelayUp	REAL	Time specification of the start-up delay [s,ms]. The time specification is only taken into account within the plant control for upshifting if it is > 0.
fDelayDown	REAL	Time specification of switch-off delay [s,ms]. The time specification is only taken into account within the plant control for downshifting if it is > 0.
fValue	REAL	Analog switching value for the aggregate.
nValue	UDINT	Numerical switching value for the aggregate.
bValue	BOOL	Binary switching value for the aggregate.
bEnFdbUp	BOOL	A TRUE means that the feedback from the aggregate is used in the step "Up".
bEnFdbDown	BOOL	A TRUE means that the feedback from the aggregate is used in the step "Down".

6.1.2.2.1.2.2.3 Sequence

6.1.2.2.1.2.2.3.1 ST_BA_PIDControlSequence

Data and command structure between the individual sequence controllers FB_BA_PIDControlSequence and the control block FB_BA_PIDControlSequenceLink [▶ 489].

Syntax

```
TYPE ST_BA_PIDControlSequence :
STRUCT
  arrSeqLinkData    : ARRAY[1..MAX(1,BA_Param.nMaxSeqCtrl)] OF ST_BA_PIDControlSequenceData;
  fE                : REAL;
  nActvSeqCtrl      : UDINT;
  bSync             : BOOL;
  bEnSeqLink        : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
arrSeqLinkData	ARRAY[1..MAX(1,BA_Param.nMaxSeqCtrl [▶ 284])] OF ST_BA_PIDControlSequenceData [▶ 281]	This field of the *ST_BA_PIDControlSequenceData* structure displays the data and commands of the individual sequence controllers. This field is limited by the global parameter *BA_Param.nMaxSeqCtrl*.
fE	REAL	Displays the control deviation of the active sequence controller. This depends on the control direction of the respective sequence controller. *E_BA_Action.eDirect -> fE = fX-fW* *E_BA_Action.eReverse -> fE = fW-fX*
nActvSeqCtrl	UDINT	Number of the active sequence controller.
bSync	BOOL	A synchronization pulse has been triggered on one of the sequence controllers. This pulse makes the sequence controller the active one in the sequence.
bEnSeqLink	BOOL	This variable shows *bEn* of the function block FB_BA_PIDControlSequenceLink [▶ 489].

6.1.2.2.1.2.2.3.2 ST_BA_PIDControlSequenceData

This structure contains the data and commands of the individual sequence controllers FB_BA_PIDControlSequence [▶ 486].

Syntax

```
TYPE ST_BA_PIDControlSequenceData :
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	Process value
fE	REAL	Displays the control deviation of the sequence controller. This depends on the control direction of the respective sequence controller. *E_BA_Action.eDirect -> fE = fX-fW* E_BA_Action.eReverse -> *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	The sequence controller is ready to operate.
bWatchdog	BOOL	The watchdog is set at each PLC cycle in each sequence controller and reset after evaluation in the sequence link.
bSeqNumMultiple	BOOL	Indicates that the own sequence number *nMyNum* has been assigned multiple times to sequence controllers.
bEn	BOOL	Input Enable sequence controller
bIsActvSeqCtrl	BOOL	Indicates that the sequence controller is the active one in the sequence.

6.1.2.2.1.2.2.4 Schedule

6.1.2.2.1.2.2.4.1 ST_BA_TempChangeFunction

For an outside temperature *fOutsideTemp*, this structure contains the expected temperature change *fRoomTempChange* when switching on(FB_BA_OptimizedOn) or switching off(FB_BA_OptimizedOff) a heating or cooling system.

```
TYPE ST_BA_TempChangeFunction :
STRUCT
  fOutsideTemp     : ARRAY[1..FB_BA_OptimizationBase.nOptOnOff_MaxNodes] OF REAL;
  fRoomTempChange  : ARRAY[1..FB_BA_OptimizationBase.nOptOnOff_MaxNodes] OF REAL;
END_STRUCT
END_TYPE
```

Name	Type	Description
fOutsideTemp	REAL	The considered outside temperature interpolation points of the optimization function in °C.
fRoomTempChange	REAL	The expected room temperature change values in K/min associated with the outside temperatures.

6.1.2.2.1.2.2.4.2 T_BA_ScheduleWeek

The type declaration includes a weekly schedule. The first part of the two-dimensional array specifies the days of the week, the second part the schedule entries, see ST_BA_SchedEntry.

The number of schedule entries is limited by the global parameter *BA_Param.nScheduleEntryCount*, see BA_Param [▶ 284].

```
TYPE
  T_BA_ScheduleWeek : ARRAY[E_BA_Weekday.First..E_BA_Weekday.Last, 1..MAX(1,BA_Param.nScheduleEntryC
ount)] OF ST_BA_SchedEntry;
END_TYPE
```

6.1.2.2.1.2.2.4.3 T_BA_ScheduleExceptionList

The type declaration contains a field of 24 schedule exception profiles, see ST_BA_ScheduleException [▶ 283].

This field is limited by the global parameter *BA_Param.nScheduleExceptionCount*, see BA_Param [▶ 284].

```
TYPE
  T_BA_ScheduleExceptionList : ARRAY[1 .. BA_Param.nScheduleExceptionCount] OF ST_BA_ScheduleExcepti
on;
END_TYPE
```

6.1.2.2.1.2.2.4.4 T_BA_ScheduleExceptionEntryList

The type declaration contains 6 schedule entries for an exception profile, see ST_BA_SchedEntry.

This field is limited by the global parameter *BA_Param.nScheduleEntryCount*, see BA_Param [▶ 284].

```
TYPE
  T_BA_ScheduleExceptionEntryList    : ARRAY[1 .. BA_Param.nScheduleEntryCount] OF ST_BA_SchedEntry;
END_TYPE
```

6.1.2.2.1.2.2.4.5 ST_BA_ScheduleException

The structure represents a schedule exception profile.

```
TYPE ST_BA_ScheduleException :
STRUCT
  eType     : E_BA_DateValChoice := E_BA_DateValChoice.Invalid;
  uDate     : U_BA_DateVal;
  aEntry    : T_BA_ScheduleExceptionEntryList;
END_STRUCT
END_YPE
```

Name	Type	Description
eType	E_BA_DateValChoice	Specification of the time period type for the exception profile entry. You can choose from 3 options (Simple date, Date range, Week and day).
uDate	U_BA_DateVal	Specification of the date value for the specified period type *eType*.
aEntry	T_BA_ScheduleExceptionEntryList [▶ 283]	List of schedule entries for an exception profile entry.

6.1.2.2.2 GVLs

6.1.2.2.2.1 BA_Globals

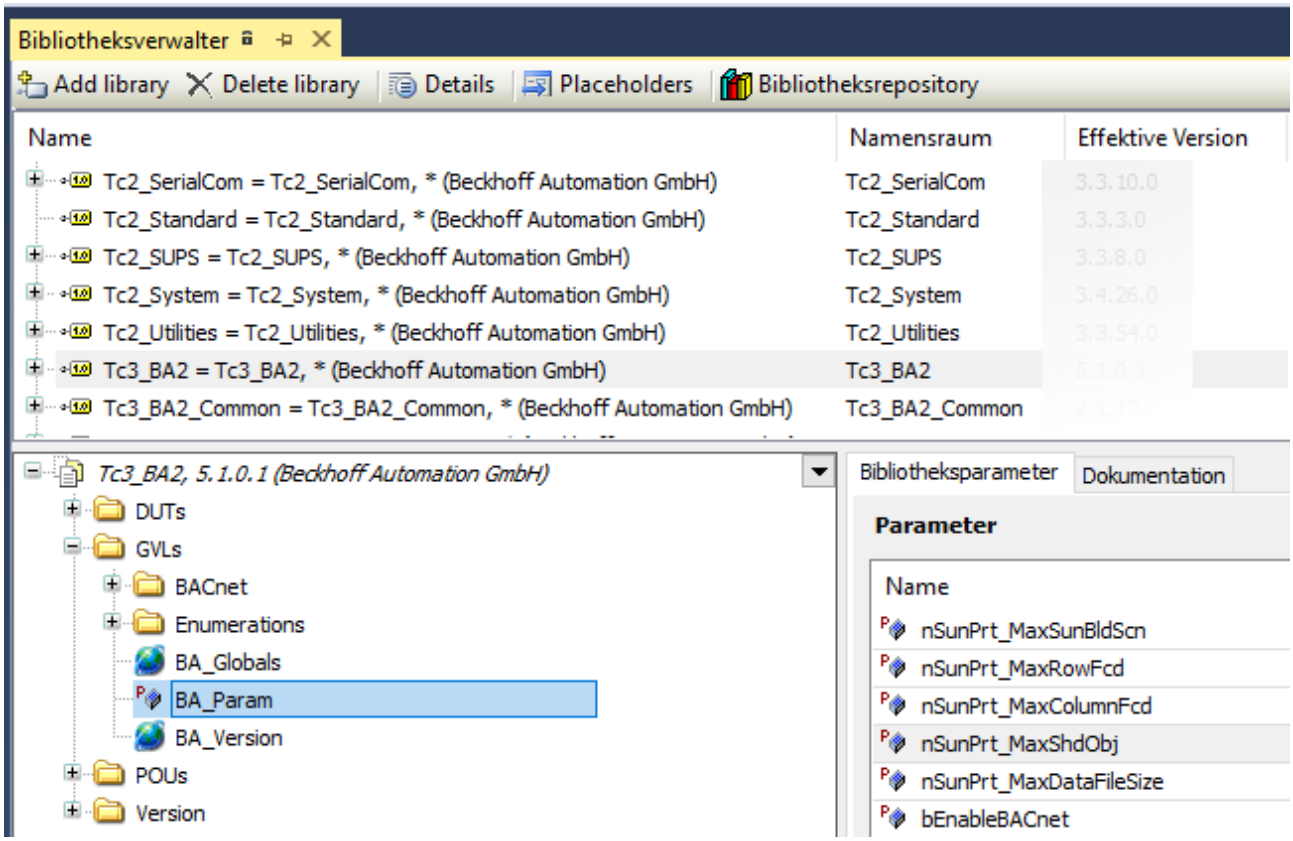
The global variable list contains value flags.

```
VAR_GLOBAL
{region 'Room-Automation'}
  nEvtIncLight     : ULINT;
  nEvtIncSunBld    : UDINT;
{endregion}
{region 'Constants'}
 {region 'General'}
  nNoActivePrio  : UDINT := 16#FFFFFFFF; // Value of the constant indicates that no priority is acti
ve
 {endregion}
{endregion}
END_VAR
```

Name	Type	Description
nEvtIncLight	ULINT	Counter light telegrams. Telegrams of certain function blocks, e.g. FB_BA_LightingEvt [▶ 309] are instantiated more often in a building. If they send telegrams of the same priority, it is desirable that the last one sent is valid. To do this, these function blocks increment the global counter *nEvtIncLight*. Telegram selectors of type FB_BA_LightingTgmSel4/ FB_BA_LightingTgmSel8 [▶ 310] evaluate this counter and accept the last telegram sent.
nEvtIncSunBld	UDINT	Same functionality as *nEvtIncLight*, only for sun protection. The telegram selectors are here: FB_BA_SunBldTgmSel4/ FB_BA_SunBldTgmSel8 [▶ 310].
nNoActivePrio	UDINT	The value of the constant indicates that no priority is active.

6.1.2.2.2.2 BA_Param

The global parameter list contains general parameters to initialize objects.




```
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}
    {region 'Plant Control'}
      nMaxPriority             : UDINT := 16;
      nMaxAggregate            : UDINT := 16;
      nNumberOfOpModes         : UDINT := 24;
```

```
    {endregion}
  {endregion}
  {region 'Objects'}
    {region 'Schedule'}
      nScheduleEntryCount      : UDINT := 6;
      nScheduleCalendarCount   : UDINT := 3;
      nScheduleExceptionCount  : UDINT := 24;
    {endregion}
  {endregion}
END_VAR
```

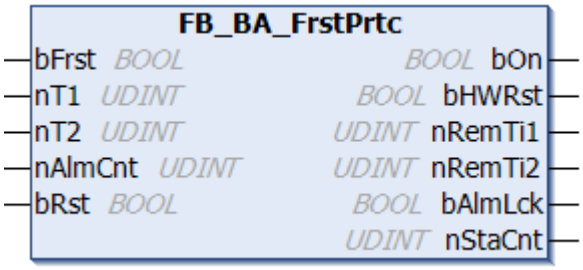
Name	Type	Description
nSunPrt_MaxSunBldScn	USINT	Maximum set of scenes that can be administered by a FB_BA_SunBldScn.
nSunPrt_MaxRowFcd	UINT	Maximum number of floors that can be monitored by the shading correction (horizontal orientation of windows).
nSunPrt_MaxColumnFcd	UINT	Maximum number of axes that 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	The global parameter specifies the number of sequence controllers. It limits the data and command structure *arrSeqLinkData* within the structure ST_BA_SeqLink [▶ 123]. The value must not be less than 1.
nMaxPriority	UDINT	The global parameter specifies the maximum number of priorities within the plant control FB_BA_PlantControl [▶ 480]. The value must not be less than 1.
nMaxAggregates	UDINT	The global parameter specifies the maximum number of aggregates within the plant control FB_BA_PlantControl [▶ 480]. The value must not be less than 1.
nNumberOfOpModes	UDINT	The global parameter specifies the maximum number of operation modes within the plant controller FB_BA_PlantControl [▶ 480]. The value must not be less than 1.
nScheduleEntryCount	UDINT	The global parameter specifies the maximum number of schedule entries, e.g. for a weekday T_BA_ScheduleWeek [▶ 282] or exception entry T_BA_ScheduleExceptionEntryList [▶ 283]. The value must not be less than 1.
nScheduleExceptionCount	UDINT	The global parameter specifies the maximum number of exception entries, e.g. T_BA_ScheduleExceptionList [▶ 283]. The value must not be less than 1.

6.1.2.2.3 POUs

6.1.2.2.3.1 FunctionBlocks

6.1.2.2.3.1.1 Air Conditioning

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*.

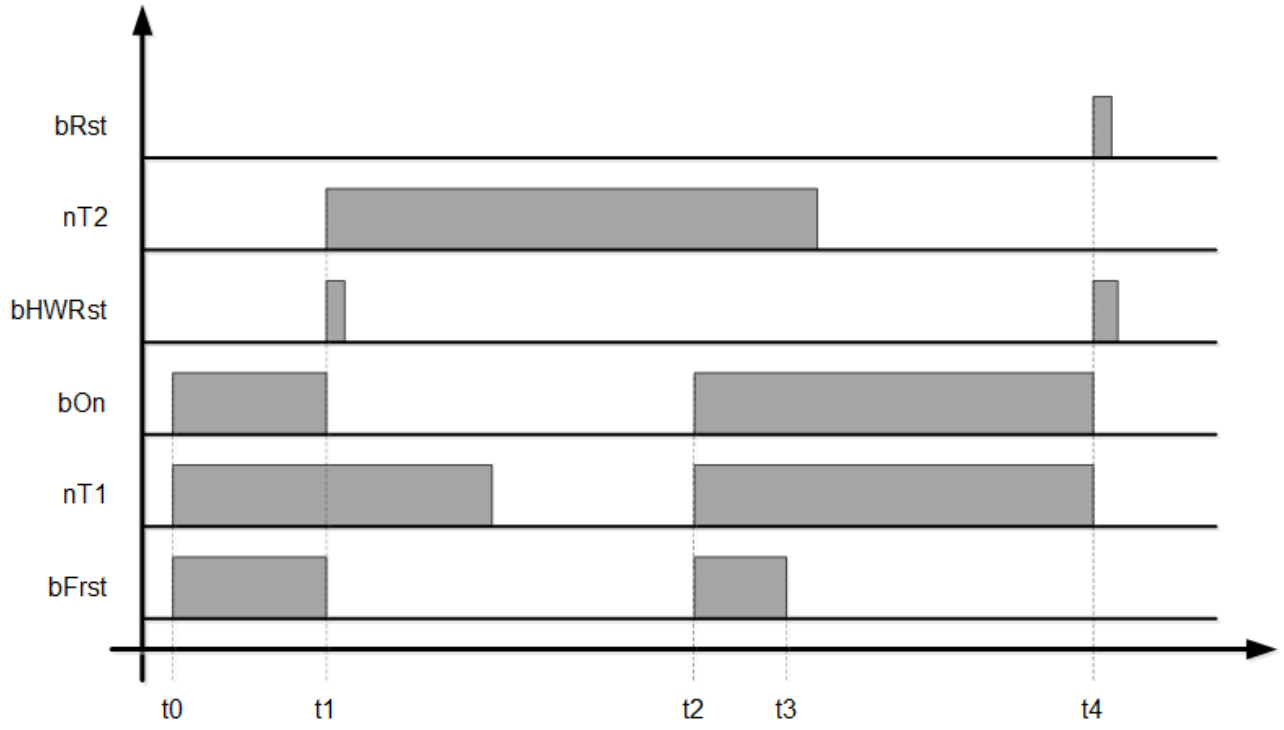
If the frost alarm ceases due to activation of the heating coil within the time *nT1* (*bFrst* = FALSE), the plant automatically restarts. For the plant restart *bOn* becomes FALSE, and at output *bHWRst* a pulse for acknowledgement of a latching circuit in the control cabinet is issued. With the restart 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

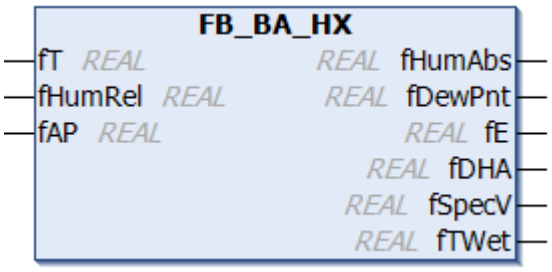
```
VAR_OUTPUT
  bOn        : BOOL;
  bHWRst     : BOOL;
  nRemTi1    : UDINT;
  nRemTi2    : UDINT;
  bAlmLck    : BOOL;
  nStaCnt    : UDINT;
END_VAR
```

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

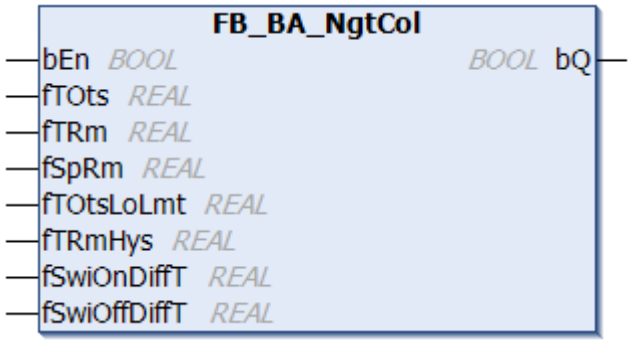
```
VAR_OUTPUT
  fHumAbs  : REAL;
  fDewPnt  : REAL;
  fE       : REAL;
  fDHA     : REAL;
  fSpecV   : REAL;
  fTWet    : REAL;
END_VAR
```

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/m³]
fSpecV	REAL	Specific volume [m³/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* = TRUE).
- The outside temperature is not too low (f*TOts* > f*TOtsLoLmt*).
- The outside temperature is sufficiently low compared to the room temperature (f*TRm* - *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* = FALSE).
- The outside temperature is too low *fTOts* < (*fTOtsLoLmt* - 0.5).
- The outside temperature is too high compared to the room temperature (*fTRm* - *fTOts*) < *fSwiOffDiffT.*
- The room temperature is lower than the setpoint. *fTRm* ≤ *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]. An internal check is carried out to ensure that the difference between (*fSwiOnDiffT* - *fSwiOffDiffT*) > 0.5.

Outputs

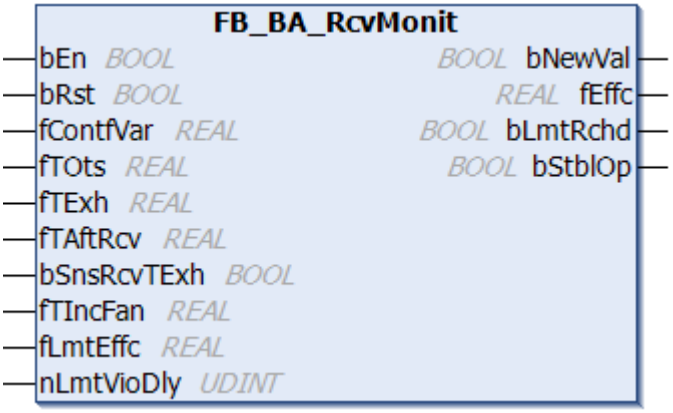
```
VAR_OUTPUT
  bQ  : BOOL;
END_VAR
```

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 extract 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 *bStblOp*, and the calculated efficiency is output at *fEffc*. If the state is not "stable", a FALSE signal appears at output *bStblOp*, 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 (*bStblOp*=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.

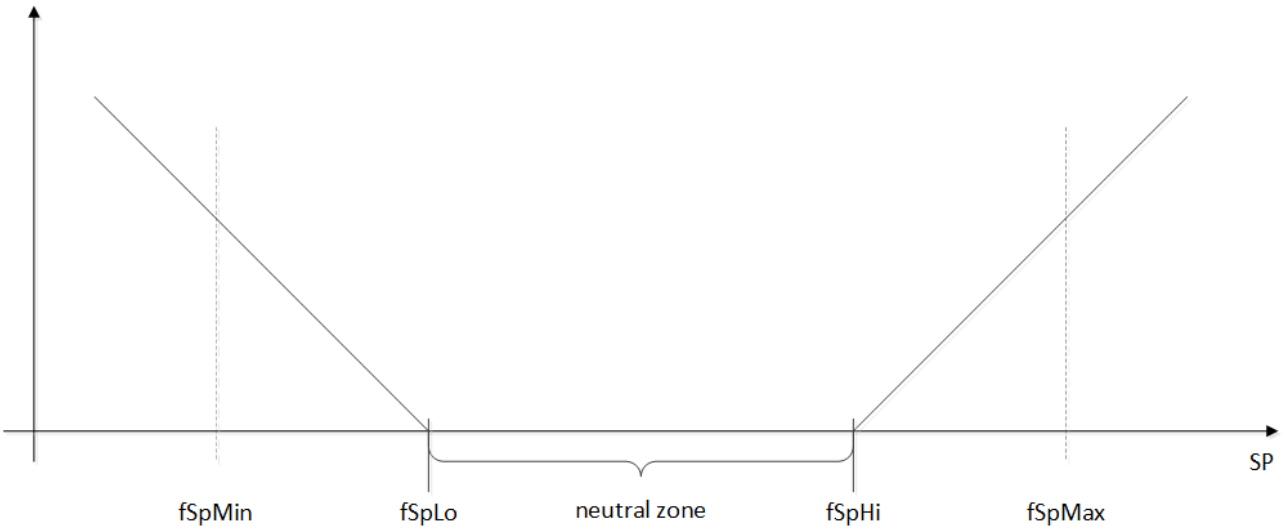
Checks and limits for the setpoints

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 >= fSpMax	Last valid values of fSpMin and fSpMax are used.
fSpHi > fSpMax	fPrSpHi = fSpMax
fSpLo < fSpMin	fPrSpLo = fSpMin

Checking	bErr	Action
fSpMin >= fSpMax	TRUE	fSpErr = ((fSpMin + fSpMax) / 2) fPrSpHi = fPrSpLo = fPrRcv = fSpErr
fSpHi < fSpMin		
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 f*PrSpHi* und *fPrSpLo* (Present Setpoint).

Setpoint for heat recovery

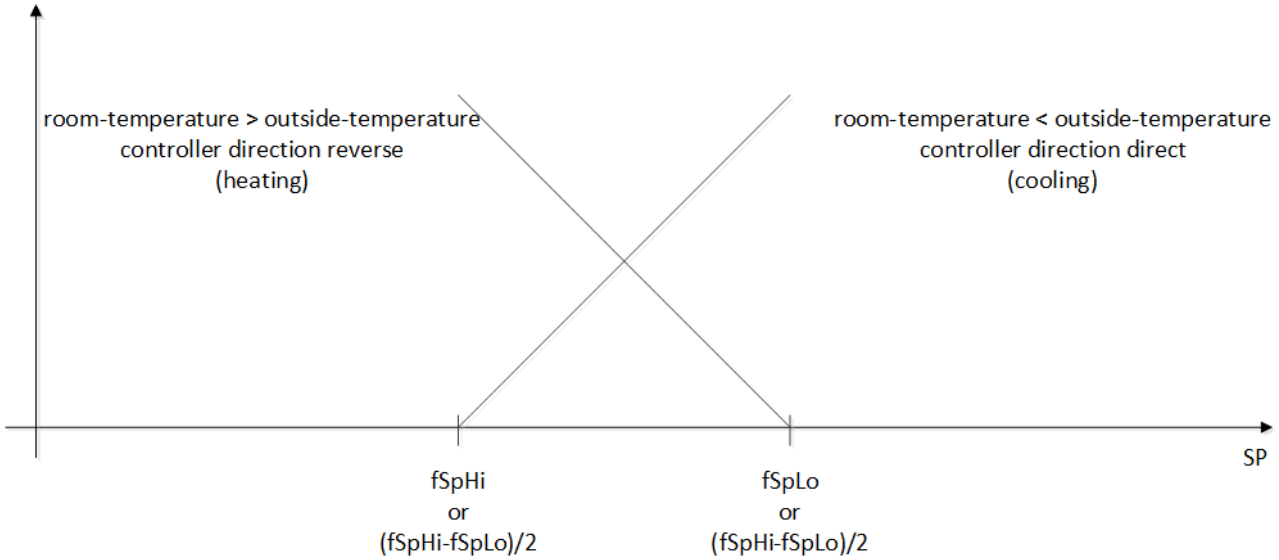
For a heat recovery system, the setpoint *fSpRcv* is calculated either from the average of the upper and lower setpoints, *fSpHi* and *fSpLo* or as a function of the control direction of the heat recovery system.
The method is defined by the input variable *bSlcnSpRcv*:

b SlcnSpRcv	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

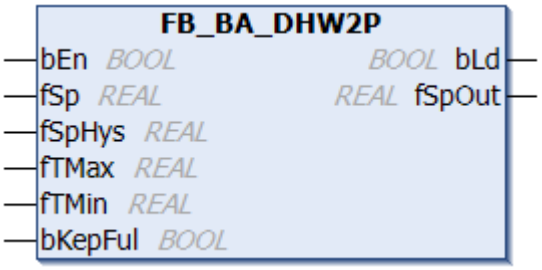
Error description
01: Warning: The setpoints are not in a logical order: Either (*fSpMin* >= *fSpMax*) OR (*fSpHi* < *fSpMin*) OR (*fSpLo* > *fSpMax*)

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 Domestic Hot Water

6.1.2.2.3.1.2.1 FB_BA_DHW2P

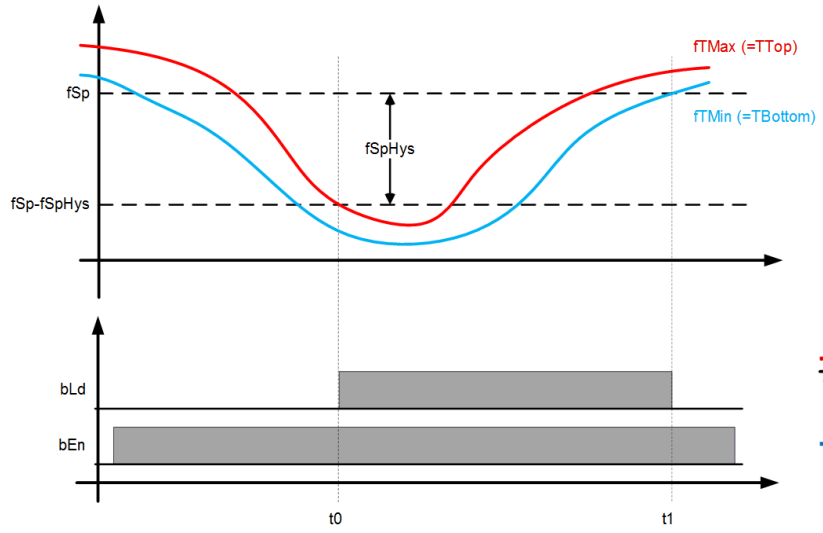


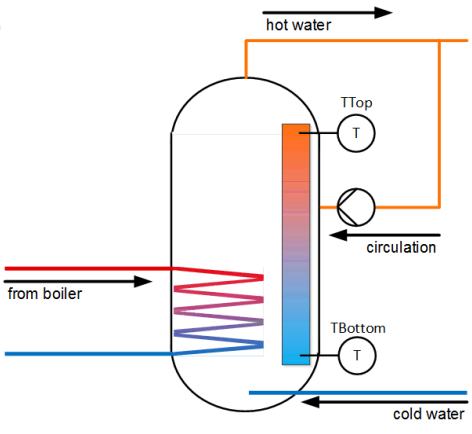
This function block FB_BA_DHW2P controls the charging (heating) of a hot water tank via an on-off controller. Tank heating is activated at input *bEn*. If tank heating is active the output *bLd* is TRUE.
The variable *fSp* is used to transfer the setpoint for the service water temperature to the function 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

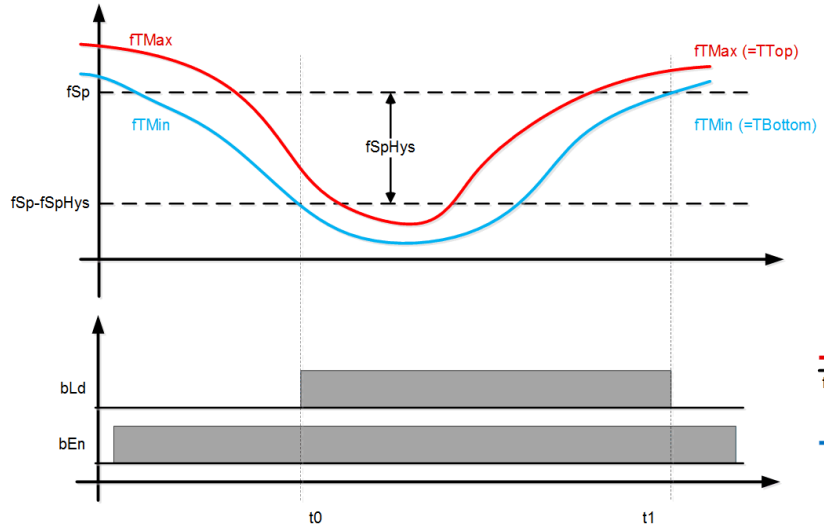
Charging is requested if *fTMax* falls below the value of *fSp-fSpHys*. The charge request is disabled if *fTMin* is above the setpoint for *fSp*.
Due to the fact that the sensor at the top generally measures the highest temperature, the heating is not switched on until the hot water tank has been discharged.

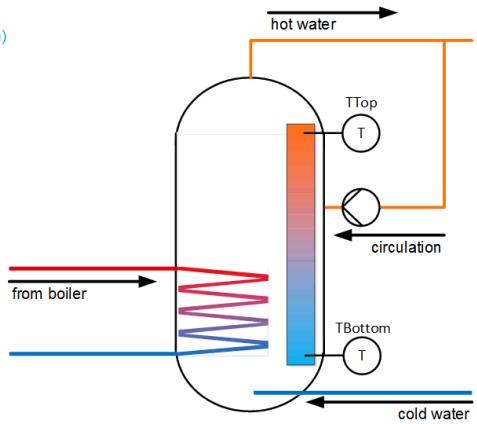




bKepFul = TRUE

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

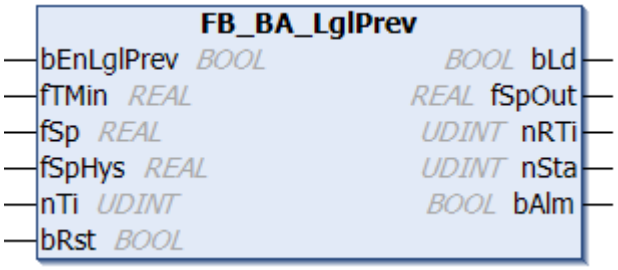
```
VAR_OUTPUT
  bLd     : BOOL;
  fSpOut  : REAL;
END_VAR
```

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.

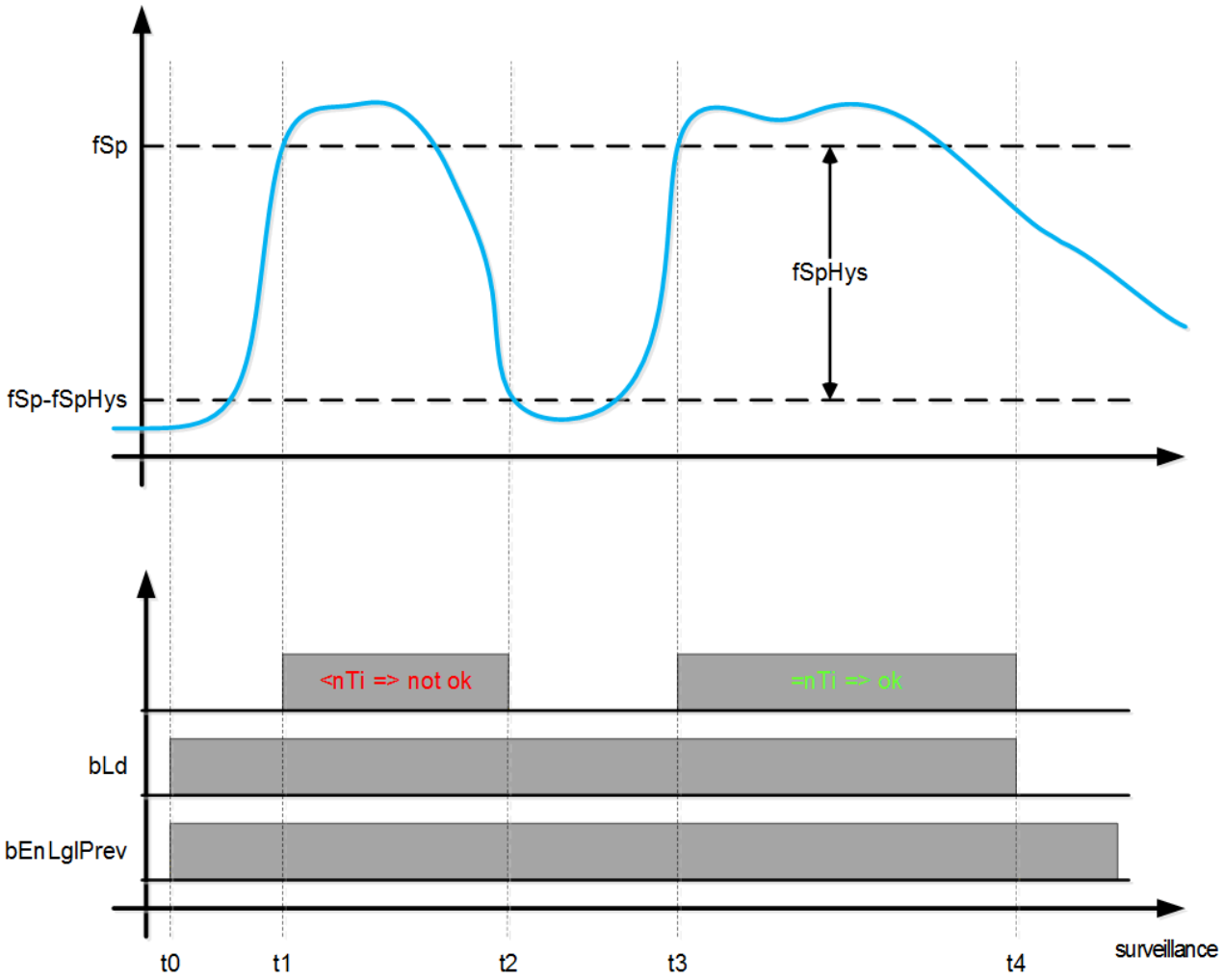
Requirements

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 f*Sp -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.

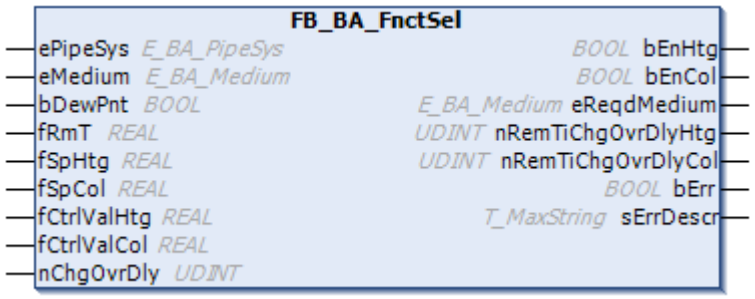
Requirements

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 Functions

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 aveilable (*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.

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
ePipeSys	E_BA_PipeSys [▶ 267]	Pipe system (2Pipe, 4Pipe) of the plant.
eMedium	E_BA_Medium [▶ 267]	Selection of the medium for the entire two-pipe network (NoMedium, HeatMedium, CoolMedium).
bDewPnt	BOOL	Dew point sensor: if *bDewPnt* = FALSE, then the cooling controller is locked.
fRmT	REAL	Room temperature
fSpHtg	REAL	Calculated setpoint of the supply air temperature for the heater.
fSpCol	REAL	Calculated setpoint of the supply air temperature for the cooler.
fCtrlValHtg	REAL	Current output value of the heating controller. Used internally as switching criterion from heating to cooling: *fCtrlValHtg* must be 0.
fCtrlValCol	REAL	Current output value of the cooling controller. Used internally as switching criterion from cooling to heating: *fCtrlValCol* 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
```

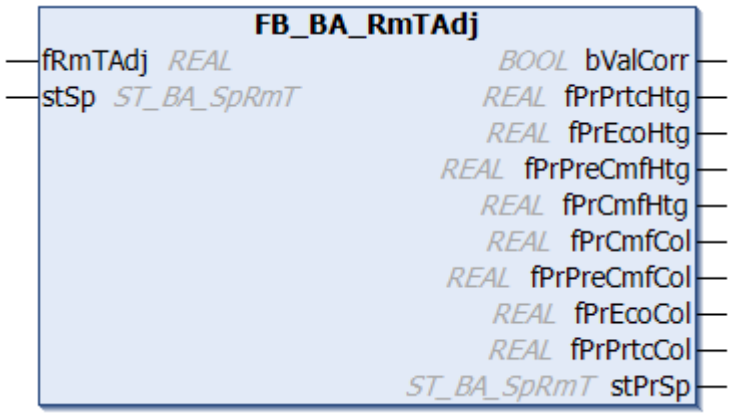
Name	Type	Description
bEnHtg	BOOL	Heating controller enable.
bEnCol	BOOL	Cooling controller enable.
eReqdMedium	E_BA_Medium [▶ 267]	Requested medium (see Determination of needs).
nRemTiChgOvrDlyHtg	UDINT	Countdown [s] for switchover delay from cooling to heating.
nRemTiChgOvrDlyCol	UDINT	Countdown [s] 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

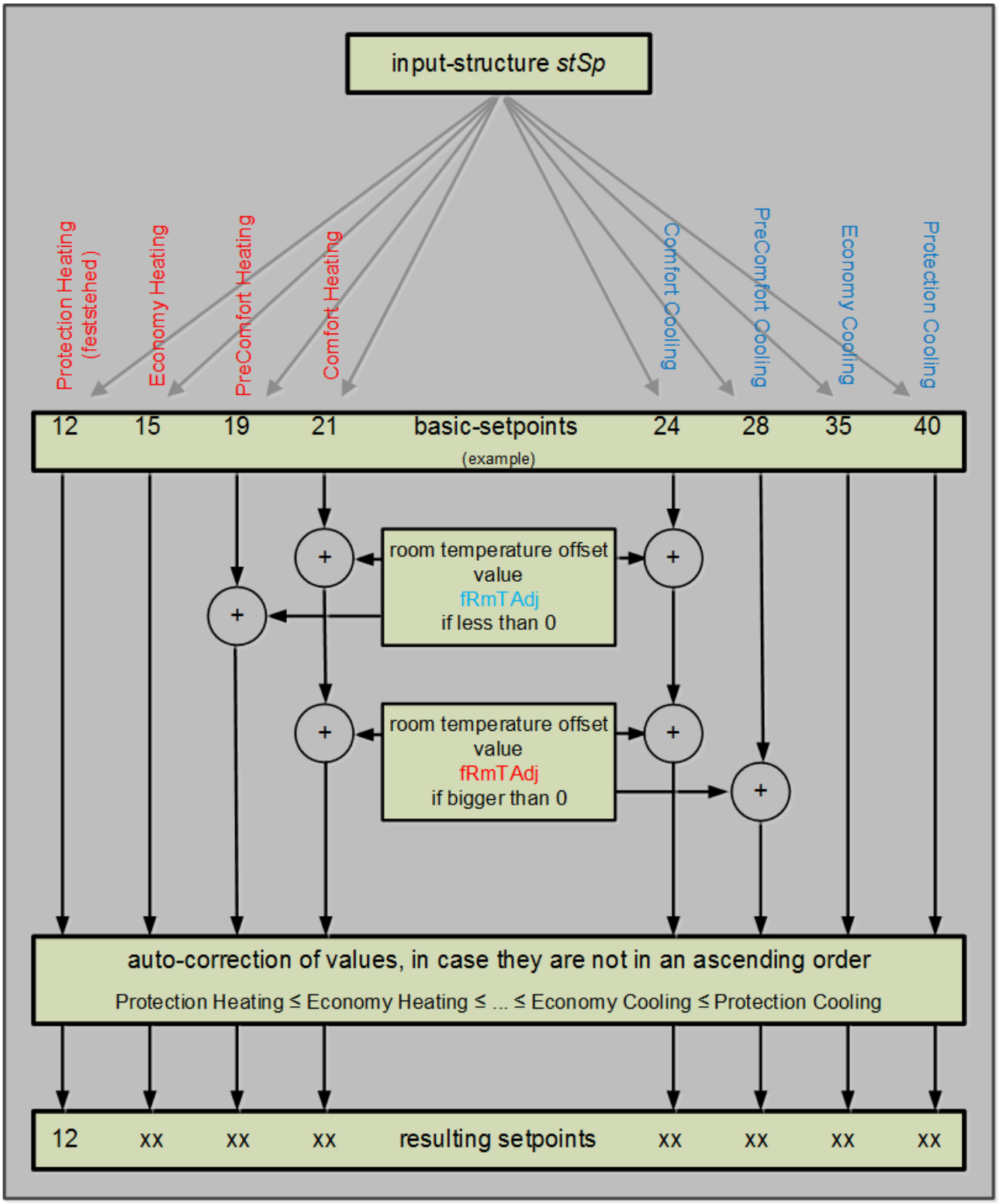
Requirements

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 [▶ 272]	Input structure for the setpoints.

Outputs

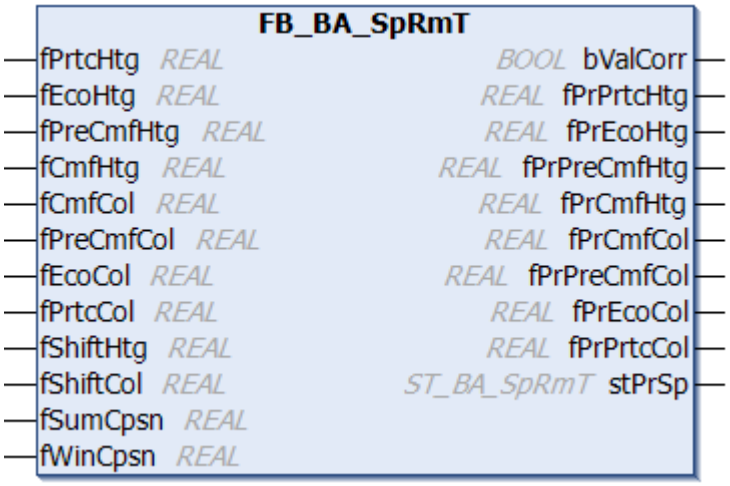
```
VAR_OUTPUT
  bValCorr       : BOOL;
  fPrPrtcHtg     : REAL;
  fPrEcoHtg      : REAL;
  fPrPreCmfHtg   : REAL;
  fPrCmfHtg      : REAL;
  fPrPrtcCol     : REAL;
  fPrEcoCol      : REAL;
  fPrPreCmfCol   : REAL;
  fPrCmfCol      : REAL;
  stPrSp         : ST_BA_SpRmT;
END_VAR
```

Name	Type	Description
bValCorr	BOOL	Autocorrection for the values was performed, see above.
fPrPrtcHtg	REAL	Resulting "Protection Heating" setpoint.
fPrEcoHtg	REAL	Resulting setpoint "Economy Heating".
fPrPreCmfHtg	REAL	Resulting setpoint "PreComfort Heating".
fPrCmfCol	REAL	Resulting setpoint "Comfort Cooling".
fPrPreCmfCol	REAL	Resulting setpoint "PreComfort Cooling".
fPrEcoCol	REAL	Resulting setpoint Economy Cooling.
fPrPrtcCol	REAL	Resulting setpoint "Protection Cooling".
stPrSp	ST_BA_SpRmT [▶ 272]	Consolidated output of the resulting values in a structure.

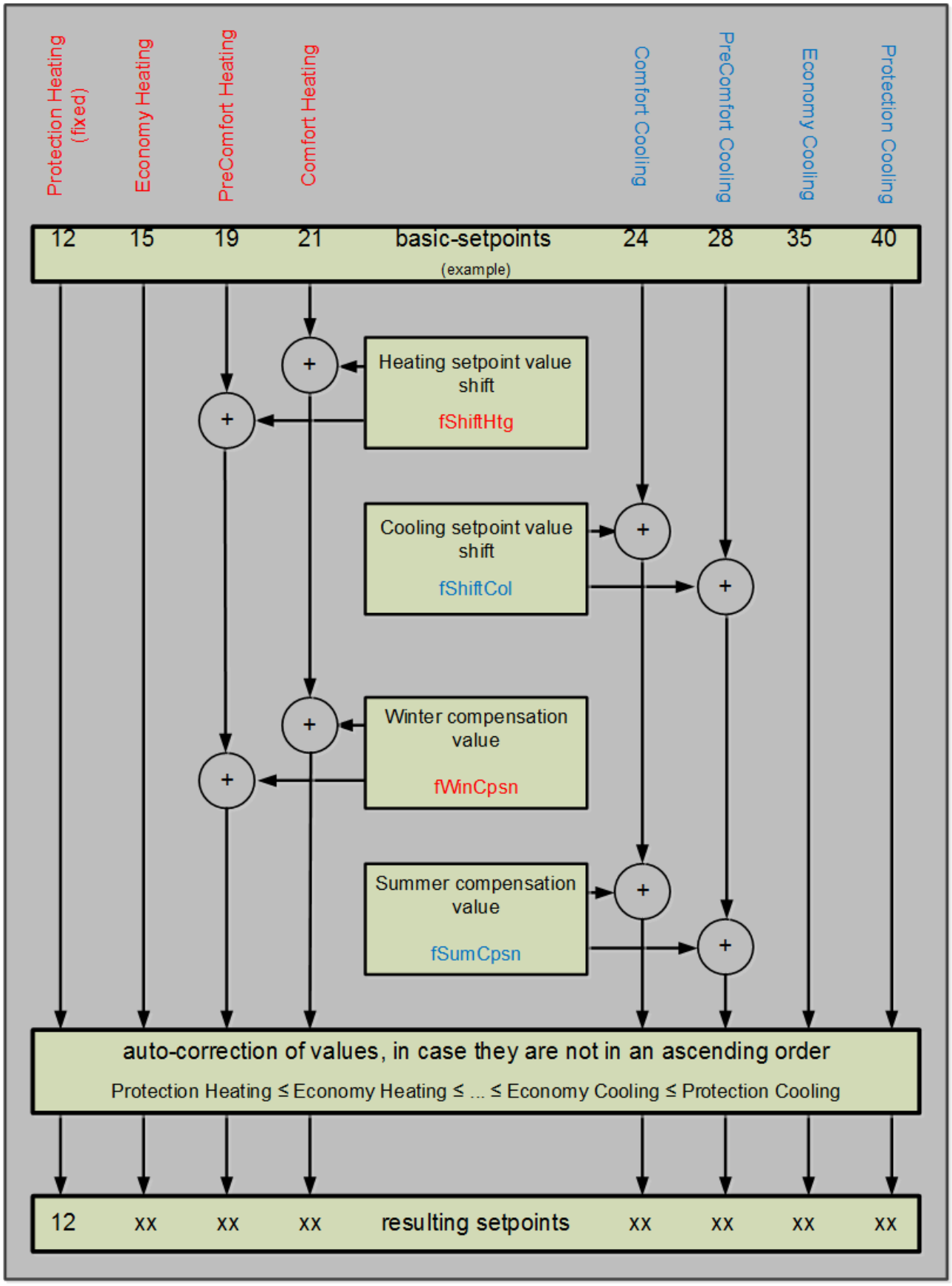
Requirements

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 setpoint "Protection Heating".
fEcoHtg	REAL	Basic setpoint "Economy Heating".
fPreCmfHtg	REAL	Basic setpoint "PreComfort Heating".
fCmfHtg	REAL	Basic setpoint "Comfort Heating".
fCmfCol	REAL	Basic setpoint "Comfort Cooling".
fPreCmfCol	REAL	Basic setpoint "PreComfort Cooling".
fEcoCo	REAL	Basic setpoint "Economy Cooling".
fPrtcCol	REAL	Basic setpoint "Protection Cooling".
fShiftHtg	REAL	Setpoint value shift "Heating".
fShiftCol	REAL	Setpoint value shift "Cooling".
fSumCpsn	REAL	Value Summer compensation.
fWinCpsn	REAL	Value Winter compensation.

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
```

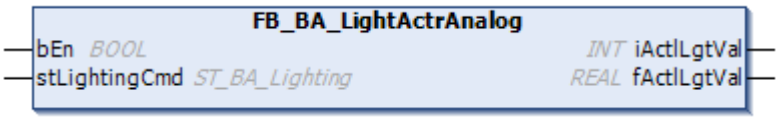
Name	Type	Description
bValCorr	BOOL	Autocorrection for the values was performed, see above.
fPrPrtcHtg	REAL	Resulting "Protection Heating" setpoint.
fPrEcoHtg	REAL	Resulting setpoint "Economy Heating".
fPrPreCmfHtg	REAL	Resulting setpoint "PreComfort Heating".
fPrCmfCol	REAL	Resulting setpoint "Comfort Cooling".
fPrPreCmfCol	REAL	Resulting setpoint "PreComfort Cooling".
fPrEcoCol	REAL	Resulting setpoint Economy Cooling.
fPrPrtcCol	REAL	Resulting setpoint "Protection Cooling".
stPrSp	ST_BA_SpRmT [▶ 272]	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_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 [▶ 272]	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

6.1.2.2.3.1.3.2.2 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 [▶ 272]. 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 [▶ 268]	Priority E_BA_LightingPrio of the telegram, preset to *eScene1*.

Outputs

```
VAR_OUTPUT
  stLighting    : ST_BA_Lighting;
  bActv         : BOOL;
END_VAR
```

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	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

6.1.2.2.3.1.3.2.3 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 [▶ 272]	Telegram inputs

Outputs

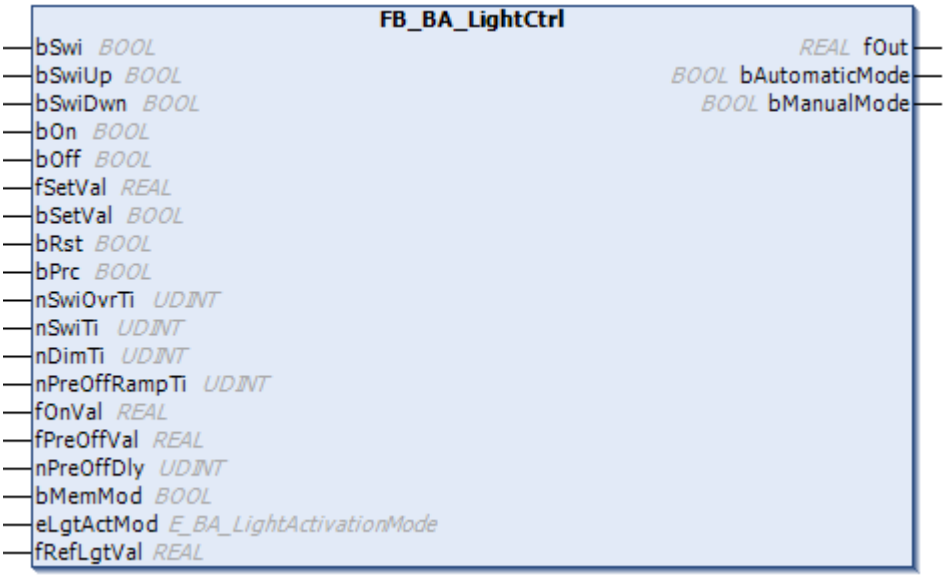
```
VAR_OUTPUT
  stLighting      : ST_BA_Lighting;
  nNumActvTgm     : UINT;
  ePrioActvTgm    : E_BA_LightingPrio;
END_VAR
```

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	Resulting telegram
nNumActvTgm	UINT	Indicates which input results, e.g. if *stLightingTgm_3* is passed, *nNumActvTgm* = 3. If *nNumActvTgm* = 0, no telegram is active.
ePrioActvTgm	E_BA_LightingPrio [▶ 268]	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

6.1.2.2.3.1.3.2.4 FB_BA_LightCtrl



The function block is a universal block for the implementation of various lighting functions.

It is used to control one or more lights.

It enables the illumination to be controlled manually via a switch or room control unit or the automatic control of a lighting group by means of presence detection. By connecting the function block accordingly, it can be used for the following functions:

- Automatic light presence-dependent
- Manual switching of illumination with dimming function
- Combination of automatic light via presence and manual control.

Automatic illumination via presence can be combined with manual operation of the illumination via the inputs *bSwi*, *bSwiUp*, *bSwiDwn*, *bOn*, *bOff*. The last state change of one of the inputs mentioned is then valid.

If the illumination is switched on via presence detection at the input *bPrc*, the illumination can be switched off or changed via the inputs *bSwi*, *bSwiUp* or *bSwiDwn*. The manual intervention is valid until a falling edge at *bPrc*.

If there is another rising edge at the input *bPrc*, the illumination is switched on again in automatic mode.

However, the illumination is only switched on automatically via *bPrc* if the automatic lighting system is in fully automatic mode. However, the illumination is only switched on automatically via *bPrc* if the automatic lighting system is in fully automatic mode.

If the current control value of the illumination is the result of manual operation, the function block is in manual mode. This is displayed at the *bManualMode* output. If the current control value of the illumination is the result of a positive or negative edge at input *bPrc*, the illumination is in automatic mode *bAutomaticMode* = TRUE. The current control value for the lighting group is output at output *fOut* from 0% to 100%.

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;
  bMemMod           : BOOL;
  eLgtActMod        : E_BA_LightActivationMode;
  fRefLgtVal        : REAL  := -1;
END_VAR
```

Name	Type	Description
bSwi	BOOL	A short press of the button at the input deactivates the illumination if it is currently switched on. If the illumination is off, it is switched on by a short press of the button. A long press of the button causes the lights in the light group to brighten or dim.
bSwiUp	BOOL	A short press of the button switches the illumination on. A long press of the button will cause the lights to become brighter.
bSwiDwn	BOOL	A short press of the button switches the illumination off. A long press of the button dims the lights.
bOn	BOOL	Regardless of the current state of the illumination, the light is switched to 100% by an edge at the input.
bOff	BOOL	Regardless of the current state of the illumination, the light is switched to 0% by an edge at the input.
fSetVal	REAL	Manual pre-set value. The value at the input *fSetVal* is taken over as light output value by a rising edge at *bSetVal*.
bSetVal	BOOL	Takeover of the manual preset value.
bRst	BOOL	This input resets the manual and automatic operation of the function block. This switches off the illumination. The output *fOut* follows a switch-off ramp. See *fPreOffVal*, *nPreOffDly* and *nPreOffRampT*.
bPrc	BOOL	Presence signal input. If the function block is in fully automatic mode, a rising edge can enable the function via this input and a falling edge can disable it. The light is switched on or off. In semi-automatic mode, only a falling edge at this input disables the function block.
nSwiOvrTi	UDINT	Time [ms] for differentiating between short and long button presses.
nSwiTi	UDINT	Ramp for the switching functions in seconds, related to a dimming from 0 to 100%.
nDimTi	UDINT	Ramp for the dimming functions in seconds, related to a dimming from 0 to 100%.
nPreOffRampTi	UDINT	Ramp used to drive to a base value *fPreOffVal* before switching off.
fOnVal	REAL	Switch-on value if the "Memory mode" option is not selected via the *bMemMod* input.
fPreOffVal / nPreOffDly	REAL / 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.
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 *fOnValMan*.
eLgtActMod	E_BA_LightActivationMode [▶ 267]	Activation mode of the light control function. 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. This input is pre-initialized with "-1". If it is not assigned, this is recognized and it is assumed that this function is the only one for controlling the lights. Thus, the output value *fOut* is used as the light value to assess the next switching actions.

Outputs

```
VAR_OUTPUT
  fOut              : REAL;
  bAutomaticMode    : BOOL;
  bManualMode       : BOOL;
END_VAR
```

Name	Type	Description
fOut	REAL	Light output value, 0...100%.
bAutomaticMode	BOOL	The function block was activated via a rising edge at the presence input, which is only possible in "Fully automatic" activation mode.
bManualMode	BOOL	The function block was activated or overridden by a button (*bOn*, *bSwi*, *bSwiUp*, *bSwiDwn*) or by setting to a value (*fSetValMan*/*bSetValMan*).

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_LightCtrlValue

This function block is used to determine a switch-on value (0..100%) for constant light regulation so that it starts with a brightness close to the setpoint and the light does not have to be adjusted for a long time.

Function

The functions of this function block are only available if it is enabled with a TRUE signal at *bEn*. Otherwise, the output *fOnValue* assumes the default value at the input *fOnValDefault*. If *fOnValDefault* is not connected, it is set to 50 by initialization.

Teach-in

In the enabled state (*bEn* = TRUE), the function block can be taught-in. A step sequence is started with a rising edge at *bTeach*.

The light is first switched off via the light control telegram *stTeachLighting*. Then wait for the time *nBrtnsAdjTi* in seconds so that the light sensor can correctly detect the lighting conditions. The brightness value that is read in via the input *fBrtns* is saved. The light is now set to 100% and *nBrtnsAdjTi* is waited for in seconds

again. The difference between the recorded brightness value and the stored value is the possible gain due to artificial light and is stored as a persistent output variable *fBrtnsGain*. The light value is set to 0 again at the end of the routine.

The light control telegram *stTeachLighting* is only active during this teach-in phase and has a high priority with *ePrio* := *E_BA_LightingPrio.eMaintenance* over the manual or automatic commands that do not represent an alarm. This function block therefore fulfills the condition of being parallel to the constant light regulation - the respective telegrams can ultimately be routed via a priority switch.

In addition, the *bTeaching* output is set during the teaching phase. This can be used to query light sensors that do not output a continuous signal more quickly, thereby shortening the teaching process.

Principle of operation

If the function block is enabled (*bEn* = TRUE) and has been taught at least once so that the output value *fBrtnsGain* has assumed a value greater than zero, an input value *fOnValue* is calculated continuously:

The difference between the setpoint and actual value, *fSP* to *fBrtns*, is compared to the gain from the artificial light *fBrtnsGain* using the rule of three. This results in the necessary percentage brightness gain. This is added to the current light value present at the *fRefLgtVal* input.

A function block that has not been taught in is recognized by the fact that the output *fBrtnsGain* is still at 0. In this case, the default value *fOnValDefault* is output even if the function block is enabled (*bEn* = TRUE).

fRefLgtVal

Light value of a single light actuator representing the light zone. With the actual brightness value and setpoint as well as the determined brightness gain *fBrtnsGain*, only the percentage increase required to start the constant light regulation close to the setpoint can be determined. If the current light value is known, the absolute switch-on value results from the sum.

In most applications, the light value will be 0% before switching on. However, with lighting controllers using priorities, it is possible for a lighting group to be controlled by a lower priority before constant light regulation is switched on and is therefore already at a certain level.

It is therefore important to know the current light control value.

Inputs

```
VAR_INPUT
  bEn                : BOOL;
  bTeach             : BOOL;
  fBrtns             : REAL;
  fSP                : REAL;
  nBrtnsAdjTi        : UDINT := 5;
  fOnValDefault      : REAL = 50;
  fRefLgtVal         : REAL;
END_VAR
```

Name	Type	Description
bEn	BOOL	A TRUE signal at this input enables the function block. If the function block has not yet been taught in, i.e. the brightness gain determined at output *fBrtnsGain* is still 0, the default value *fOnValue* is permanently assumed at output *fOnValDefault*. This value is also set if the function block is not enabled (*bEn* = FALSE).
bTeach	BOOL	A rising edge at this input starts the teaching process.
fBrtns	REAL	Input for the measured brightness value.
fSP	REAL	Input for the brightness setpoint.
nBrtnsAdjTi	UDINT	Waiting time in seconds that is waited during the teaching process after a light change in order to receive a correct value from the light sensor.
fOnValDefault	REAL	Standard output value [0..100%] that is passed through at output *fOnVal* if the function block is either not enabled (*bEn* = FALSE) or not taught-in (*fBrtnsGain* = 0).
fRefLgtVal	REAL	Light value [%] of a single light actuator representing the light zone.

Outputs

```
VAR_OUTPUT
  fOnValue           : REAL;
  bTeaching          : BOOL;
  stTeachLighting    : ST_BA_Lighting;
END_VAR
```

Name	Type	Description
fOnValue	REAL	Determined switch-on value. If the function block is not enabled or not taught-in, the default value *fOnValDefault* is output here.
bTeaching	BOOL	A TRUE signal at this output indicates that the teaching process is active.
stTeachLighting	ST_BA_Lighting [▶ 272]	Light control telegram with the pre-set priority *E_BA_LightingPrio.eMaintenance*.

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio              : BYTE := E_BA_LightingPrio.eMaintenance;
END_VAR
```

Name	Type	Description
ePrio	E_BA_LightingPrio [▶ 268]	Priority of the telegram, pre-set to *E_BA_LightingPrio.eMaintenance*.

PERSISTENT outputs

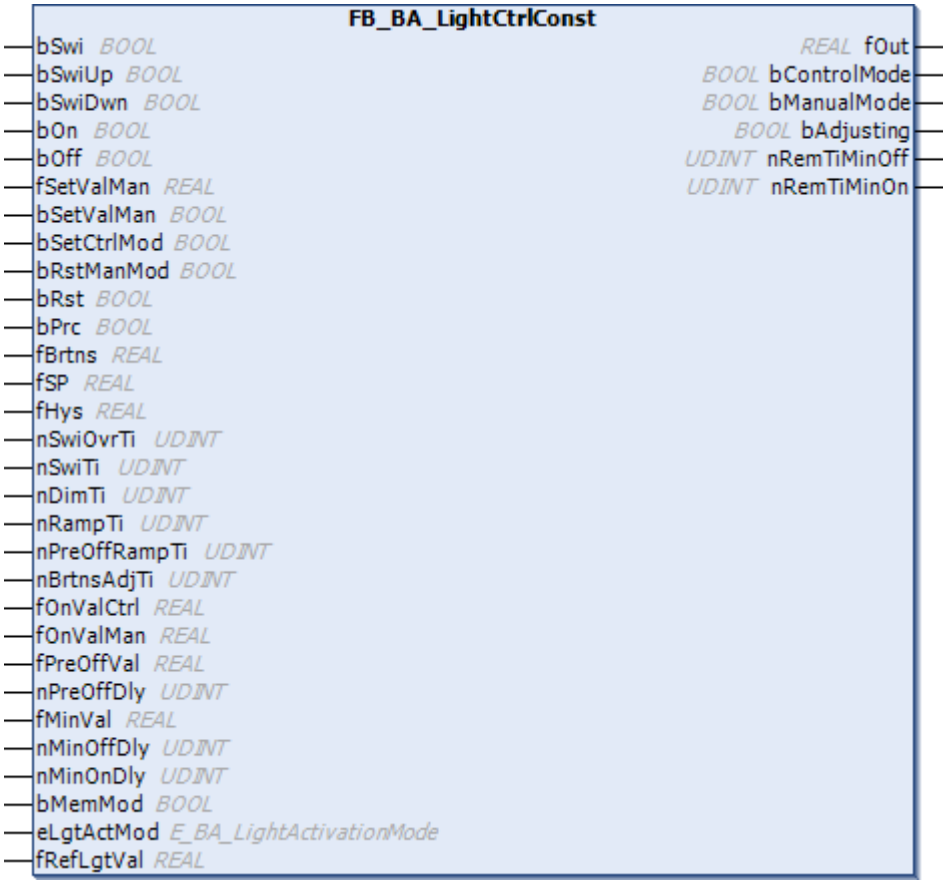
```
VAR_OUTPUT PERSISTENT
  fBrtnsGain         : REAL;
END_VAR
```

Name	Type	Description
fBrtnsGain	REAL	Determined brightness gain. This value is related to the difference from a brightness (*fBrtns*) at a light value of 0% to a brightness at a light value of 100%.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.55	Tc3_BA2 from v5.3.19.0

6.1.2.2.3.1.3.2.6 FB_BA_LightCtrlConst



This function block is a constant light regulation 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.

If it is ensured that the function is the only one that controls the light, the input 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*, *bSetCtrlMod* 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 [▶ 267]).

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.

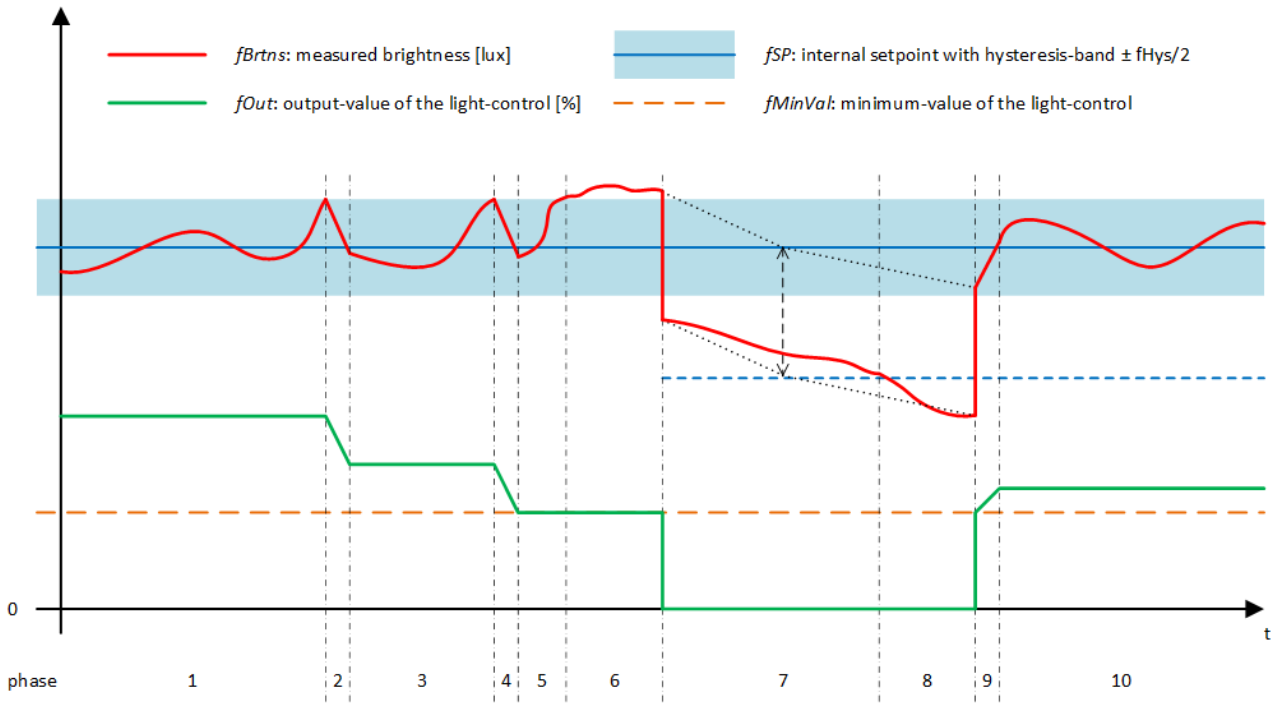
It can therefore be useful to limit the light control to a minimum control value and then switch off the light completely when the outdoor brightness is very high.

If the control has reached its minimum control value *fMinVal* and is to regulate down 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 resulting from this is the brightness gain by switching on to the minimum value.

The new threshold for switching on again is now the control setpoint minus the brightness gain. 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 choice of this threshold value is illustrated in the following diagram:



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).

1. The threshold value is undershot for the time *nOnDly* [s].
2. The light is again first switched to the minimum value and then dimmed up until the measured brightness corresponds to the setpoint *fSP*.
3. 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%.
- A change to the input *fSetValMan*. This value is then adopted as the new light output value.

"Off" in manual mode means that the overall function is still active!

A positive edge at *bRstManMod* or *bSetCtrlMod* switches from manual override back to control mode.

Memory mode

If memory mode is enabled at input *bMemMod*, 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 disabled, switching on in manual mode always takes place with the value *fOnValMan*.

Disabling the function

From control mode or manual mode, two events cause the function to change to the disabled 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* [s] at the output before switching off. The function block is then deactivated.

During the switch-off phase, the function block can be enabled again to return to normal operation.

Enabling takes place via short button presses on *bSwi*, *bSwiUp* or *bSwiDown* and rising edges on *bOn* or *bPrc*. However, the presence signal input *bPrc* is only active in full automatic mode.

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 mode 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 mode 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 pre-set value. The values at input *fSetValMan* are adopted as the light output value by a rising edge at *bSetValMan*. When the function block is set, it switches directly to manual mode, regardless of whether it was previously in control mode or in the disabled state.
bSetValMan	BOOL	Adoption of the manual pre-set value, see *fSetValMan*.
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 allows the constant light control function block to switch back to automatic mode if 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: *fPreOffVal*, *nPreOffDly* and *nPreOffRampT*.
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 via this input if it was not previously activated and a falling edge can disable it. In semi-automatic mode, only a falling edge at this input disables the constant light function block. "Activating" here means that the function is switched on in manual mode (outputs *bControlMode* = TRUE and *bManualMode* = FALSE). Deactivating means: the function is not active in manual or control mode (outputs *bControlMode* = FALSE and *bManualMode* = FALSE).
fBrtns	REAL	Current brightness for constant light automatic: Range and unit depend on the light sensor.
fSP	REAL	Brightness setpoint for the constant light automatic: An adjustment with *fBrtnsSen* is aimed at. The range and unit therefore depend on the light sensor.

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 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 depend on the used light sensor, see above: *fBrtnsSP* and *fBrtnsSen*.
nSwiOvrTi	UDINT	Time [ms] for differentiating between short and long button presses.
nSwiTi	UDINT	Ramp for the switching functions in seconds, related to a dimming from 0 to 100%.
nDimTi	UDINT	Ramp for the dimming functions in seconds, related to a dimming from 0 to 100%.
nRampTi	UDINT	Control ramp of the constant light automatic in seconds, related to a dimming from 0 to 100%.
nPreOffRampTi	UDINT	Ramp used to drive to a base value *fPreOffVal* before switching off.
nBrtnsAdjTi	UDINT	Waiting time [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 *eOperationalMode*.
fPreOffVal / nPreOffDly	REAL / 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 regulation switches the light off after *nOffDly* [s] has elapsed. If the light is switched off and the controller detects a light demand above the minimum value again, the controller switches the light on again after *nOnDly* [s] to initially *fMinVal*.
nMinOffDly	UDINT	Constant light automatic: Switch-off waiting time in seconds, see *fMinVal*.
nMinOnDly	UDINT	Constant light automatic: Switch-on waiting time in seconds, see *fMinVal*.
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 *fOnValMan*.
eLgtActMod	E_BA_LightActivationMode [▶ 267]	Activation mode of the light control function. 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. This input is pre-initialized with "-1". If it is not assigned, this is recognized and it is assumed that this function is the only one for controlling the lights. Thus, the output value *fOut* 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
```

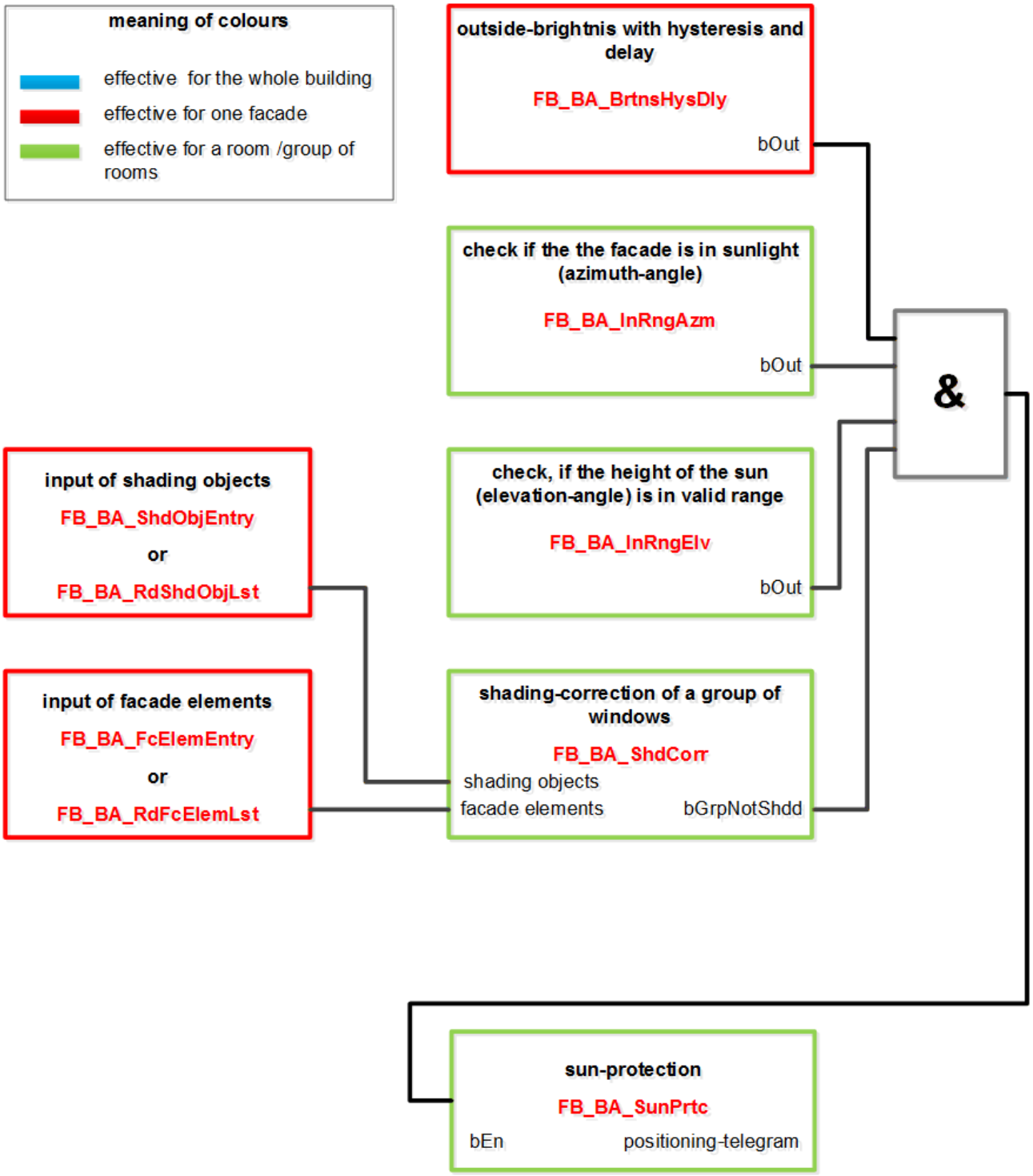
Name	Type	Description
fOut	REAL	Light output value, 0...100%.
bControlMode	BOOL	Indicates whether the function block is operating in control mode.
bManualMode	BOOL	Indicates whether or not the function block is working.
bAdjusting	BOOL	The constant light control function block is in regulation mode. 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	Control output: The control output is at the minimum value and the timer runs until switching to "0%". This output shows the remaining seconds.
nRemTiMinOn	UDINT	Control output: The constant light regulation has switched off the light due to sufficient ambient light, but now light is required again and the timer runs down until it is switched on again. This output shows the remaining seconds.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.55	Tc3_BA2 from v5.3.19.0

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	β=0°
East	β=90°
South	β=180°
West	β=270°

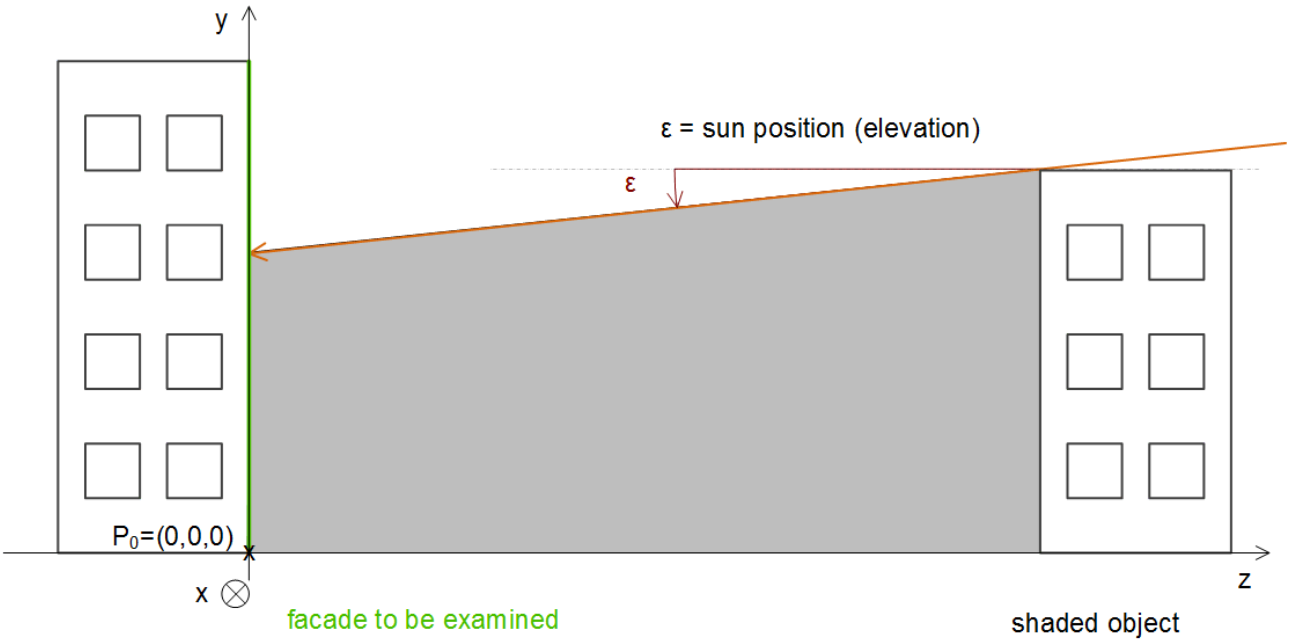
In the southern hemisphere is the sun path is reversed: Although it also rises in the east, ad midday it is in the north. The facade orientation is adjusted to this path:

Line of sight	Facade orientation
South	β=0°
East	β=90°
North	β=180°
West	β=270°

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 [▶ 585] (shading correction) is parameterized they are activated through a boolean input, *bSouth*

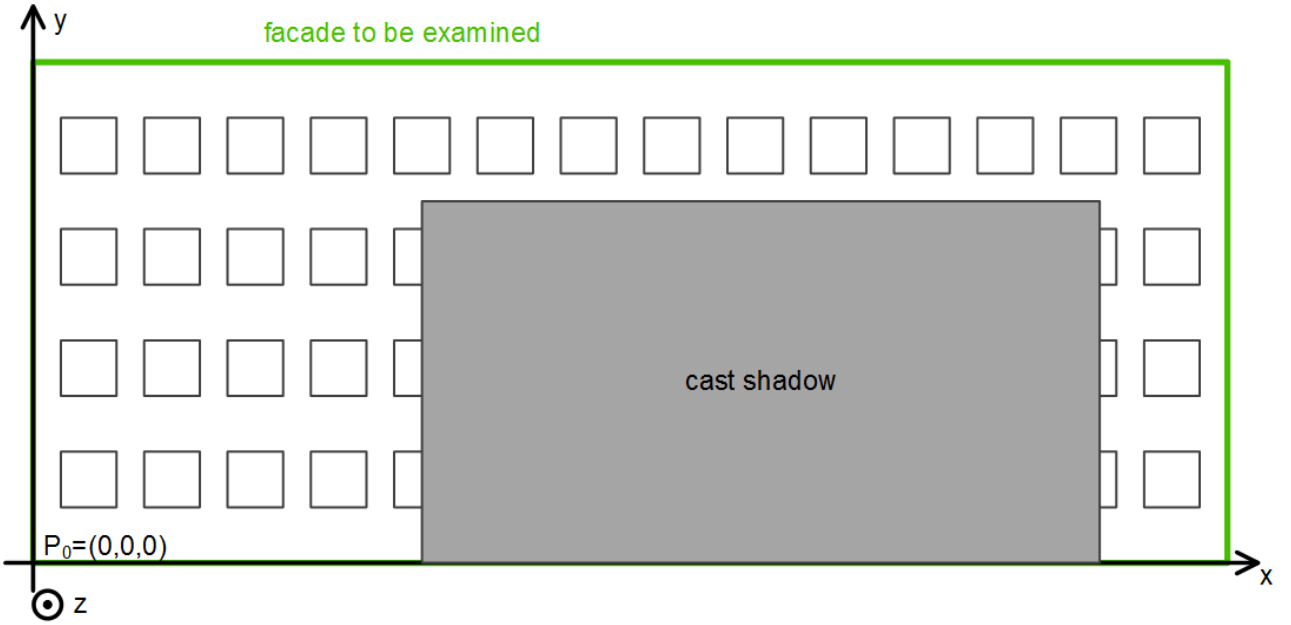
The following two illustrations are intended to further clarify the position of the point of origin P_0 as well as the orientation of the coordinate system:

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.

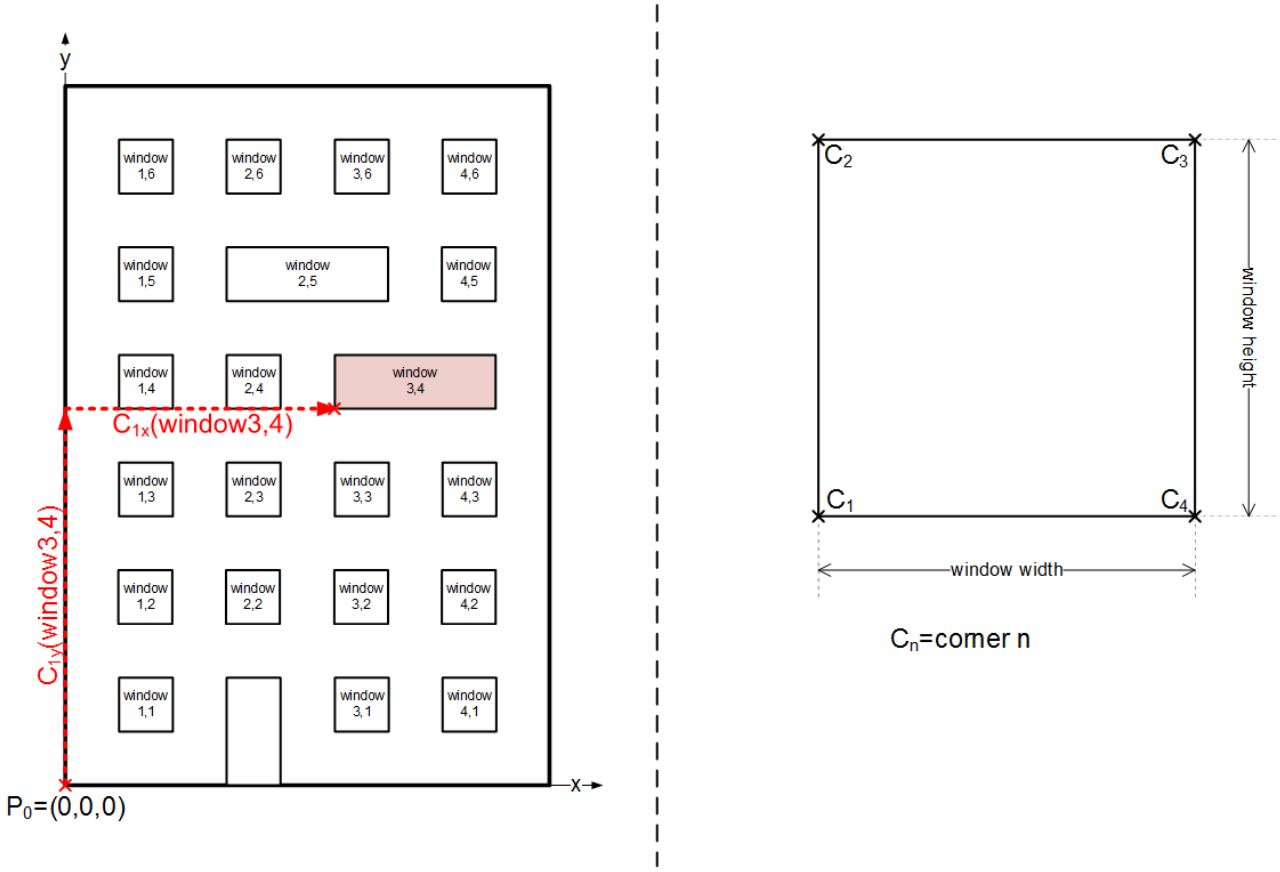
Observation from the front



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.



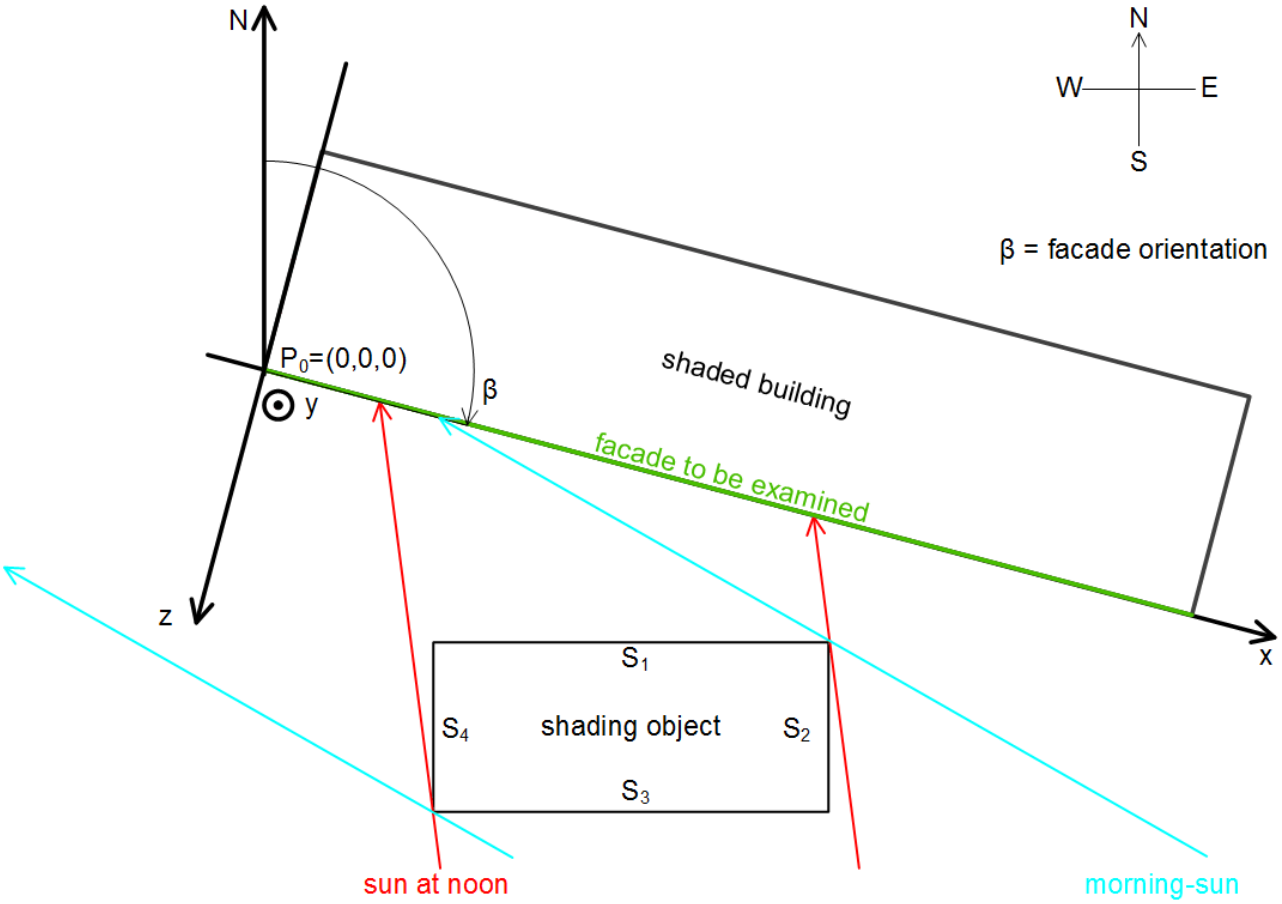
In addition, the window width and the window height have to be specified.

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

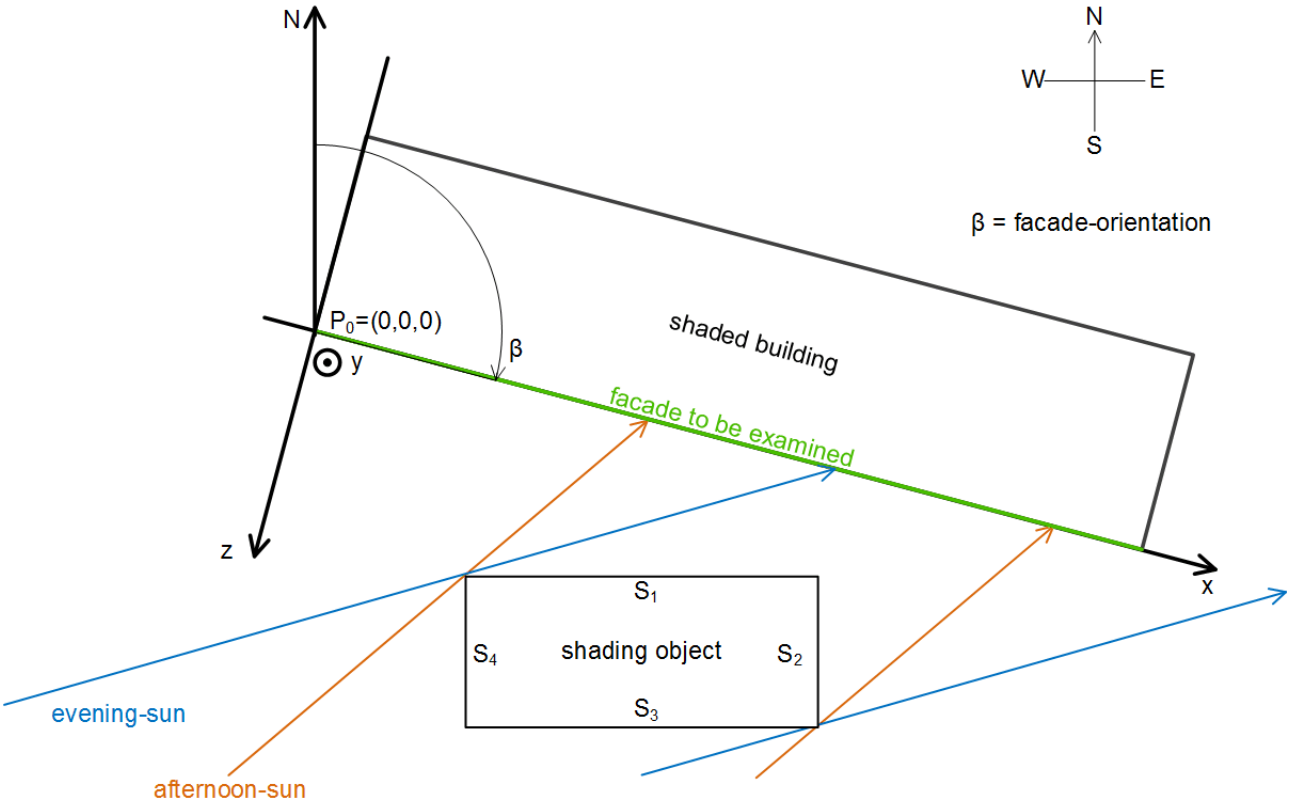
When describing the shading objects, distinction is made between angular objects (building, column) and objects that are approximately spherical (e.g. trees). Angular objects can be subdivided into square shadow-casting facades according to their shadows, noting which cast the main shadow throughout the day:

Morning/noon

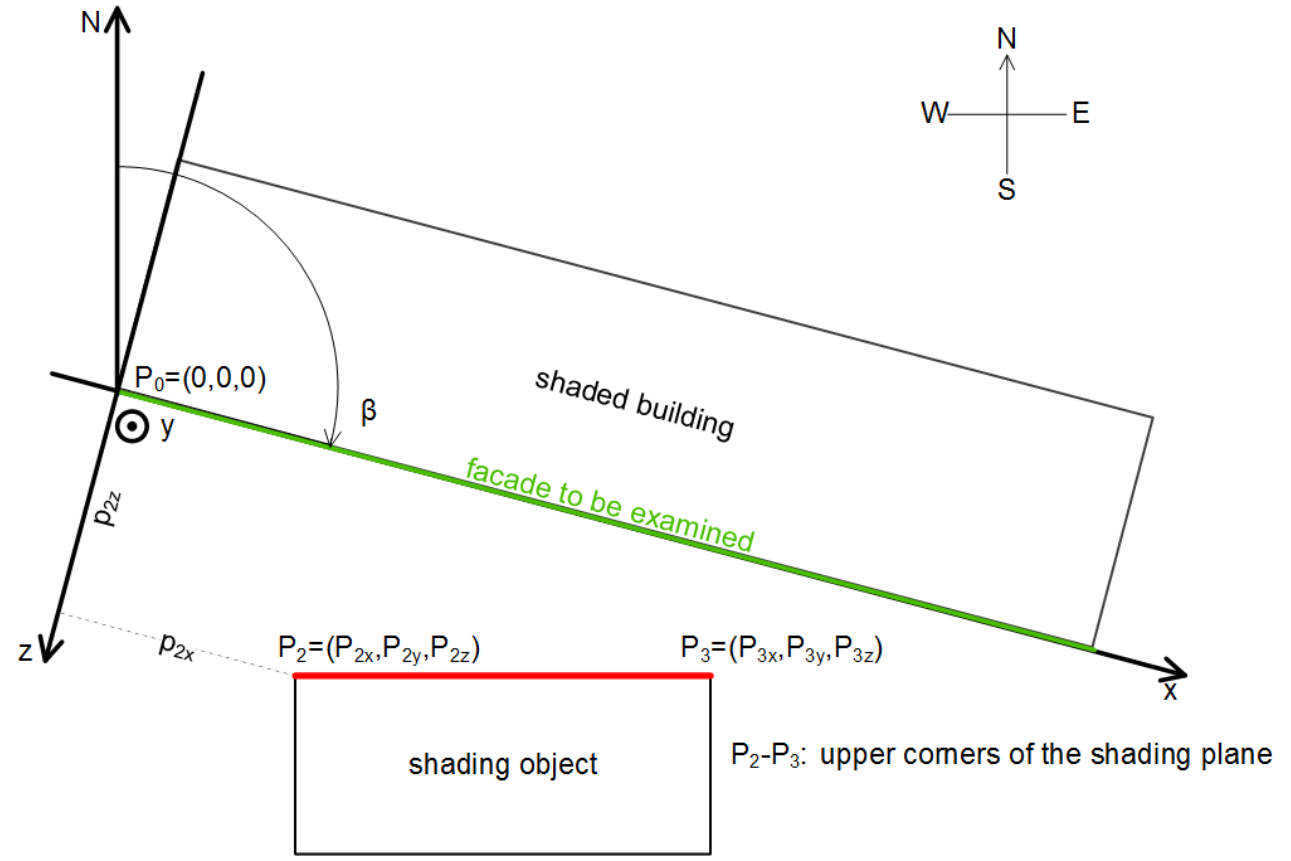


In the morning and around noon, the shadow is mainly cast by the sides S_1 and S_4. S_2 and S_3 do not have to be considered, unless they are higher.

Afternoon/evening



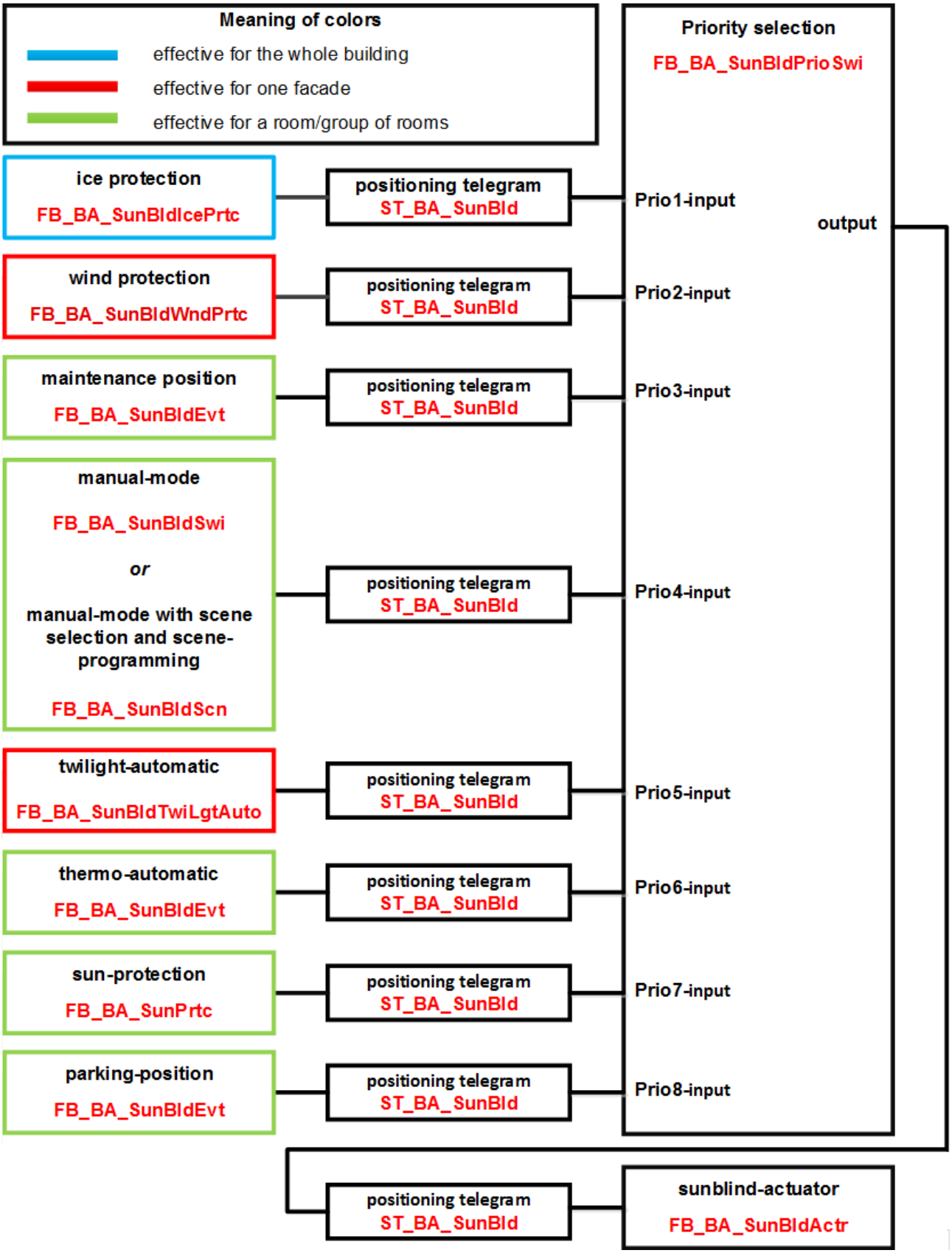
In the afternoon and evening, the total shade can be determined solely through S_1 and S_2. In this case it is therefore sufficient to specify S_1, S_2 and S_4 as shadow casters. The entry is made on the basis of the four corners or their coordinates in relation to the zero point of the facade:



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.

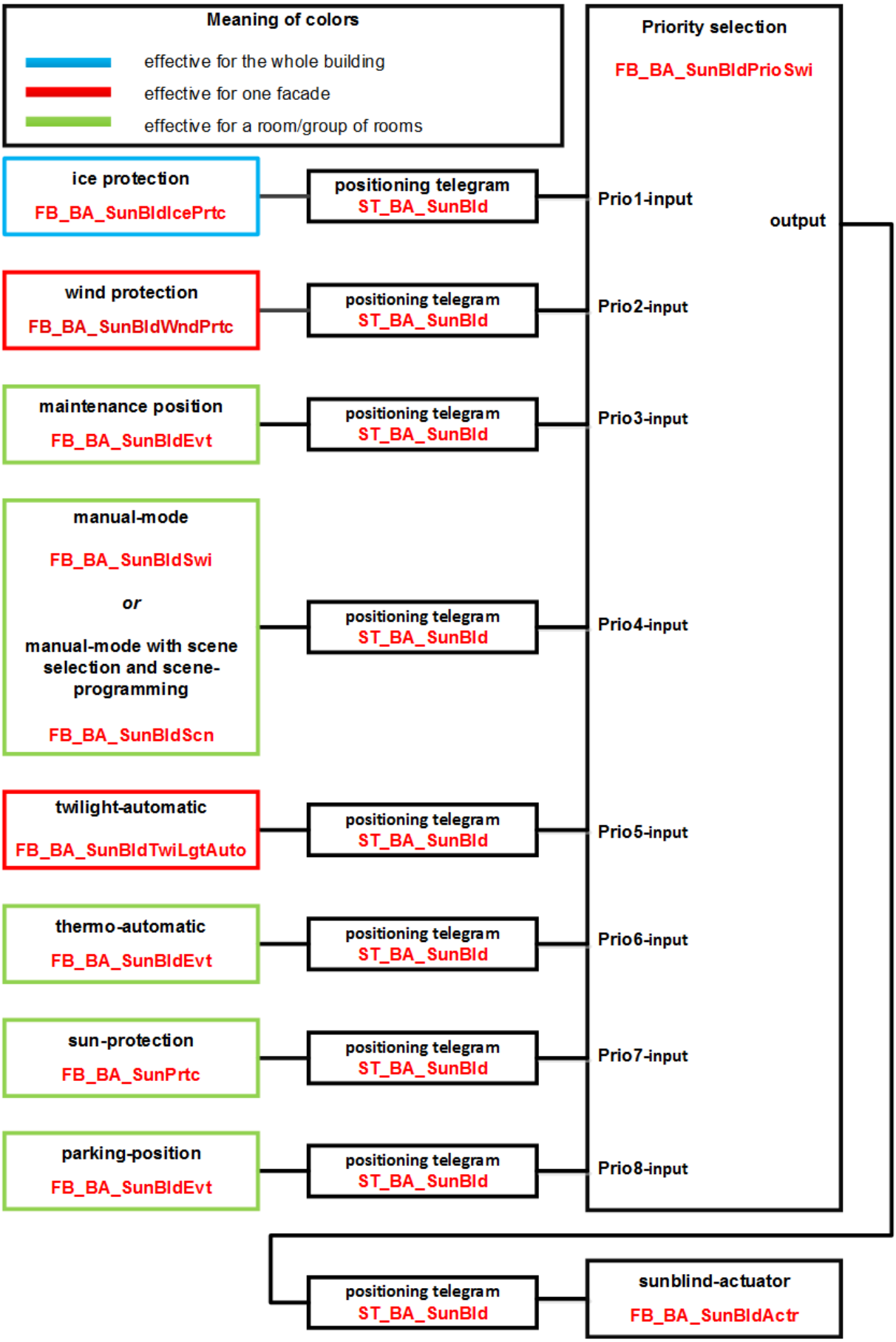
The input of shadow-casting ball elements is done by entering the center of the ball and its radius:

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



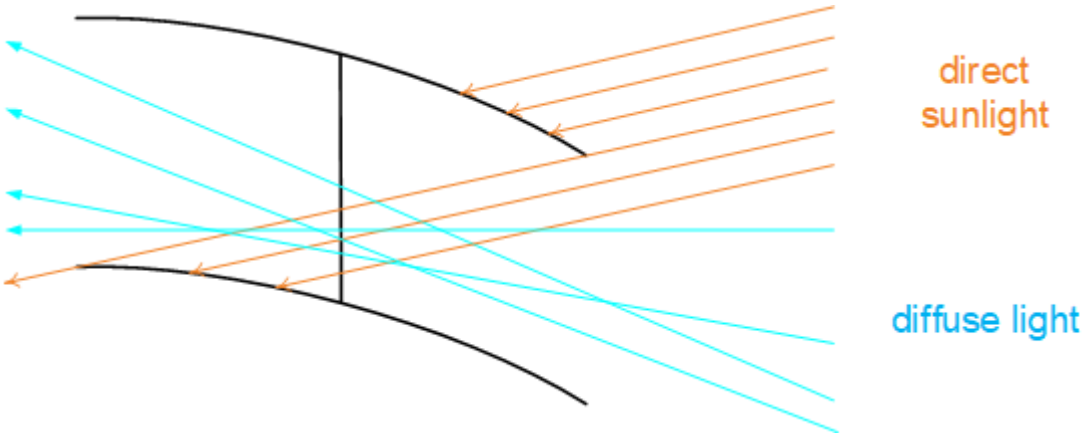
6.1.2.2.3.1.3.3.4 Sun protection: Basic principles and definitions

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

Lamella setpoint tracing

A blind with lamellas that can be adjusted offers the option of intelligent sun protection here. The position of the lamellas 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 lamella angle:

- the current sun elevation (elevation angle)
- the sun position, i.e. the azimuth angle
- the facade orientation
- the lamella width
- the lamella spacing

Effective elevation angle

If the blind is viewed in section as above, the angle of incidence does not depend solely on the sun 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 elevation angle.

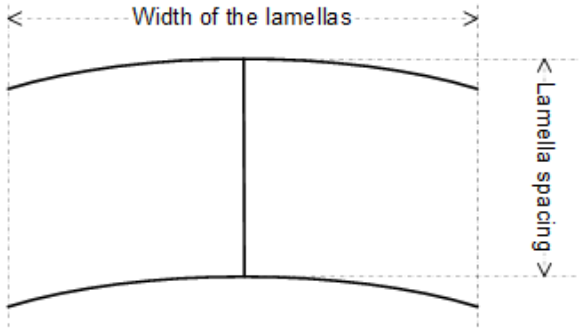
This relationship can easily be illustrated with a set square positioned upright on the table: Viewed directly from the side you can see a triangle with two 45° angles and one 90° angle. If the triangle is rotated, the side on the table appears to become shorter and the two original 45° angles change. 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.

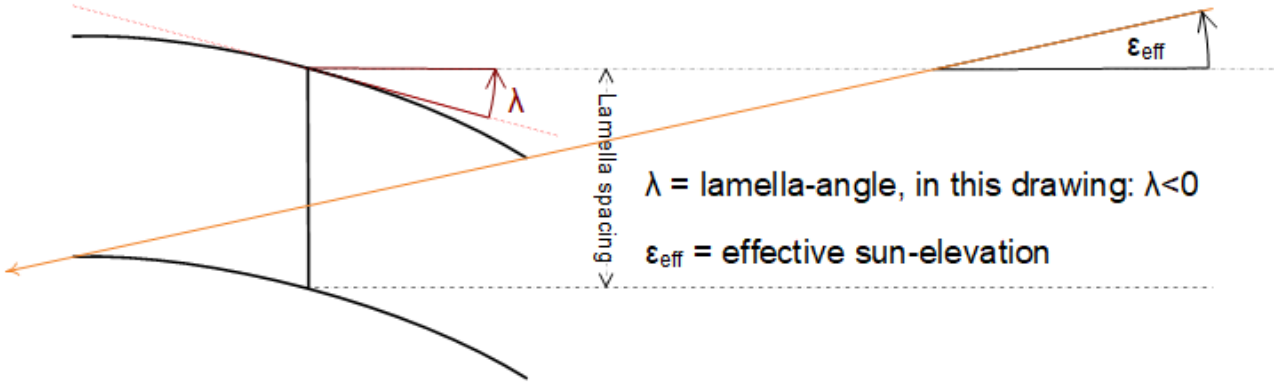
The following three images illustrate the relationship between the effective elevation angle and the blind dimensions, and how the resulting lamella angle λ changes during the day:

Lamella-angle

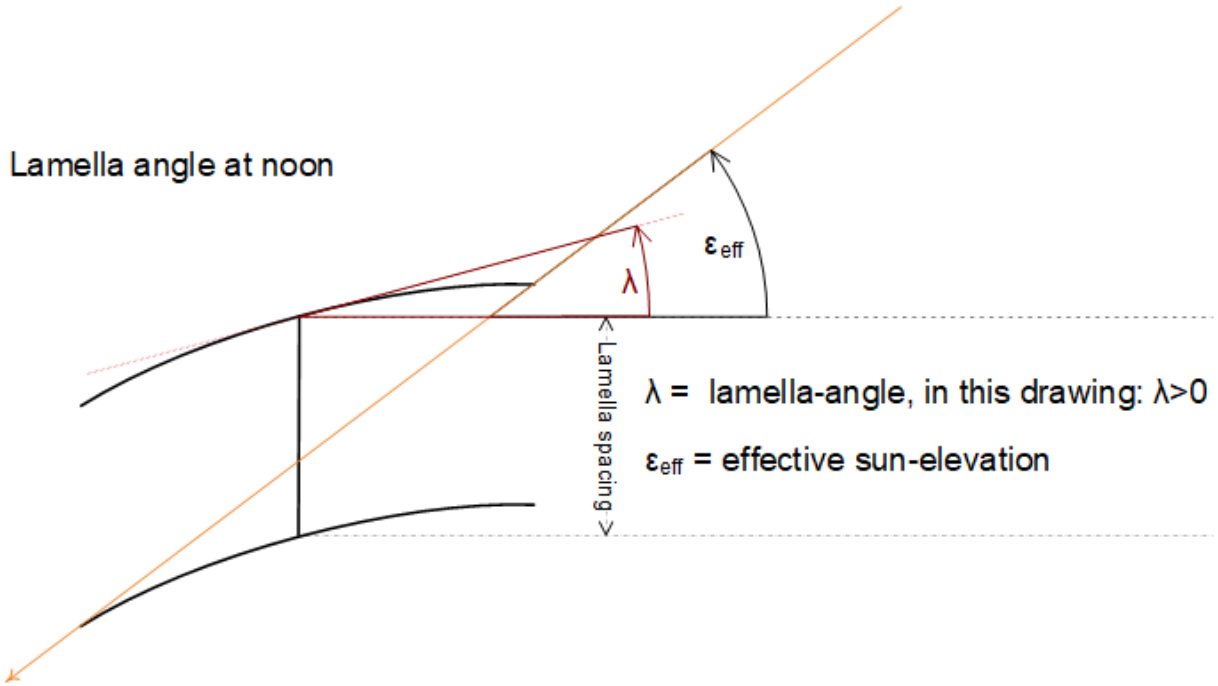
Lamella at an angle of λ=0



Lamella-angle in the morning and in the evening



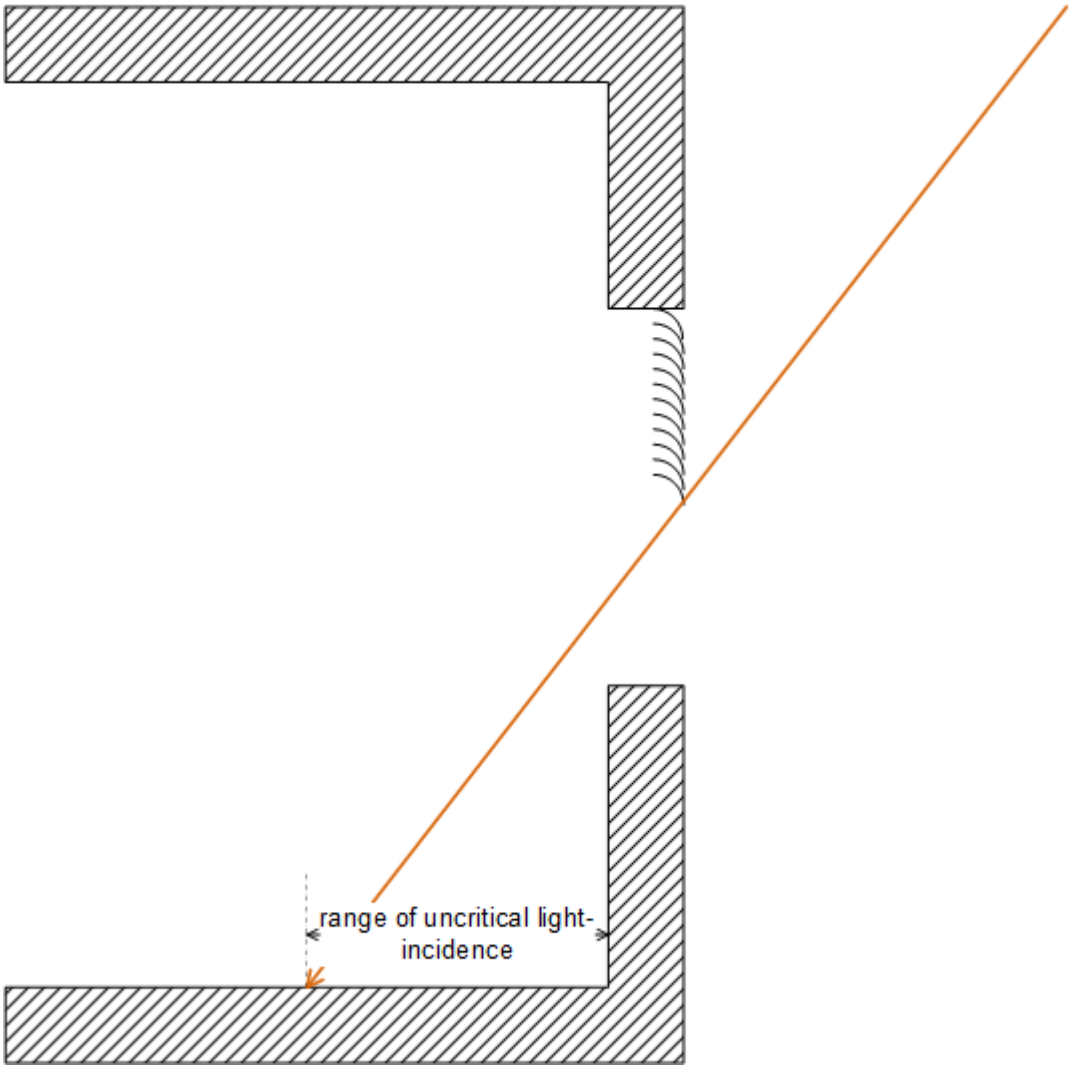
Lamella angle at noon



More detailed information on this topic can be found in the chapter Effective elevation angle [▶ 338].

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.

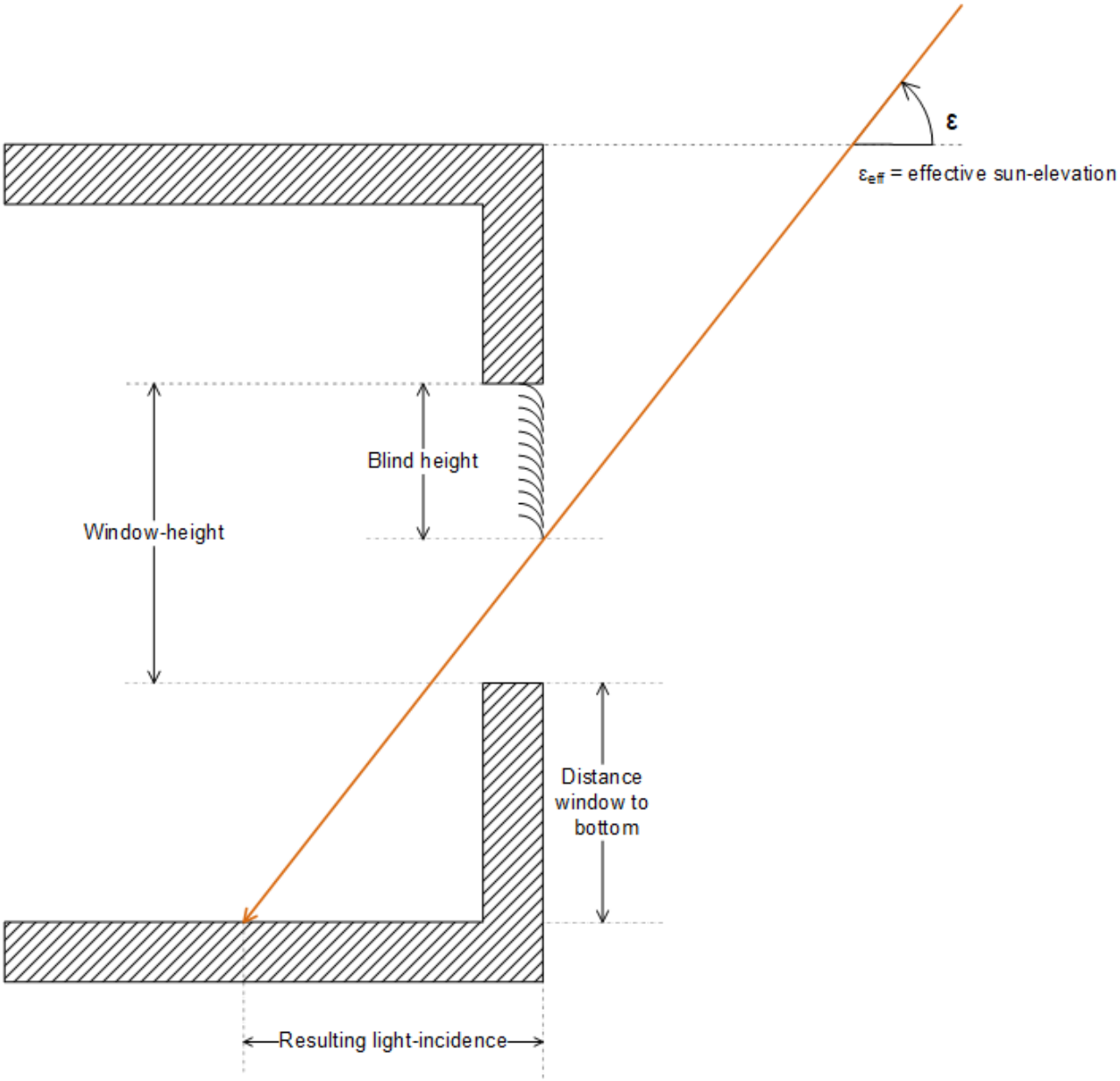


In order to be able to calculate at any time the appropriate blind height that guarantees that the incidence of sunlight does not exceed a certain value, the following values are necessary.

Required for the calculation of the respective blind height:

- Height of the sun (elevation)
- Window height
- Distance between the window and the floor

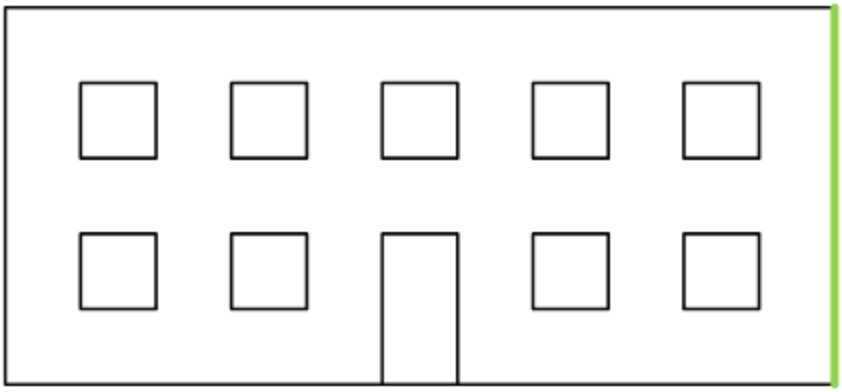
The following illustration shows where these parameters are to be classified:



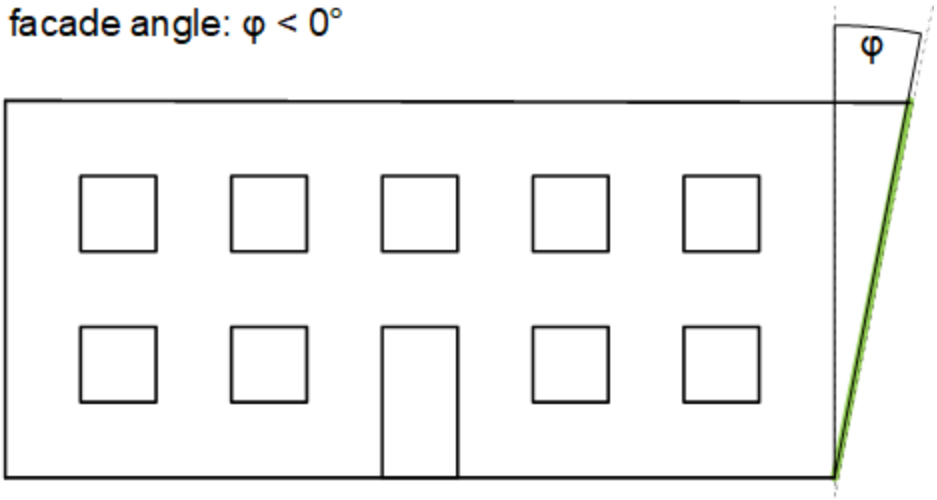
Influence of the facade inclination

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:

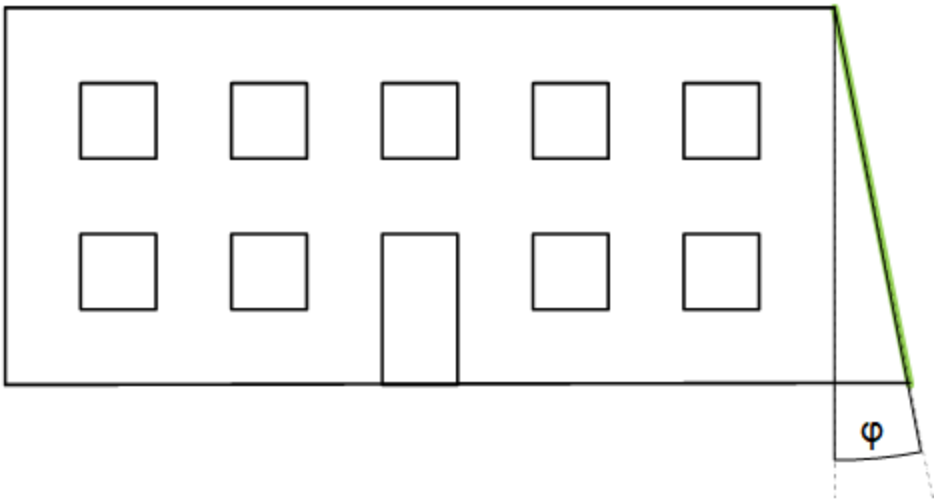
facade angle: φ = 0°



facade angle: φ < 0°

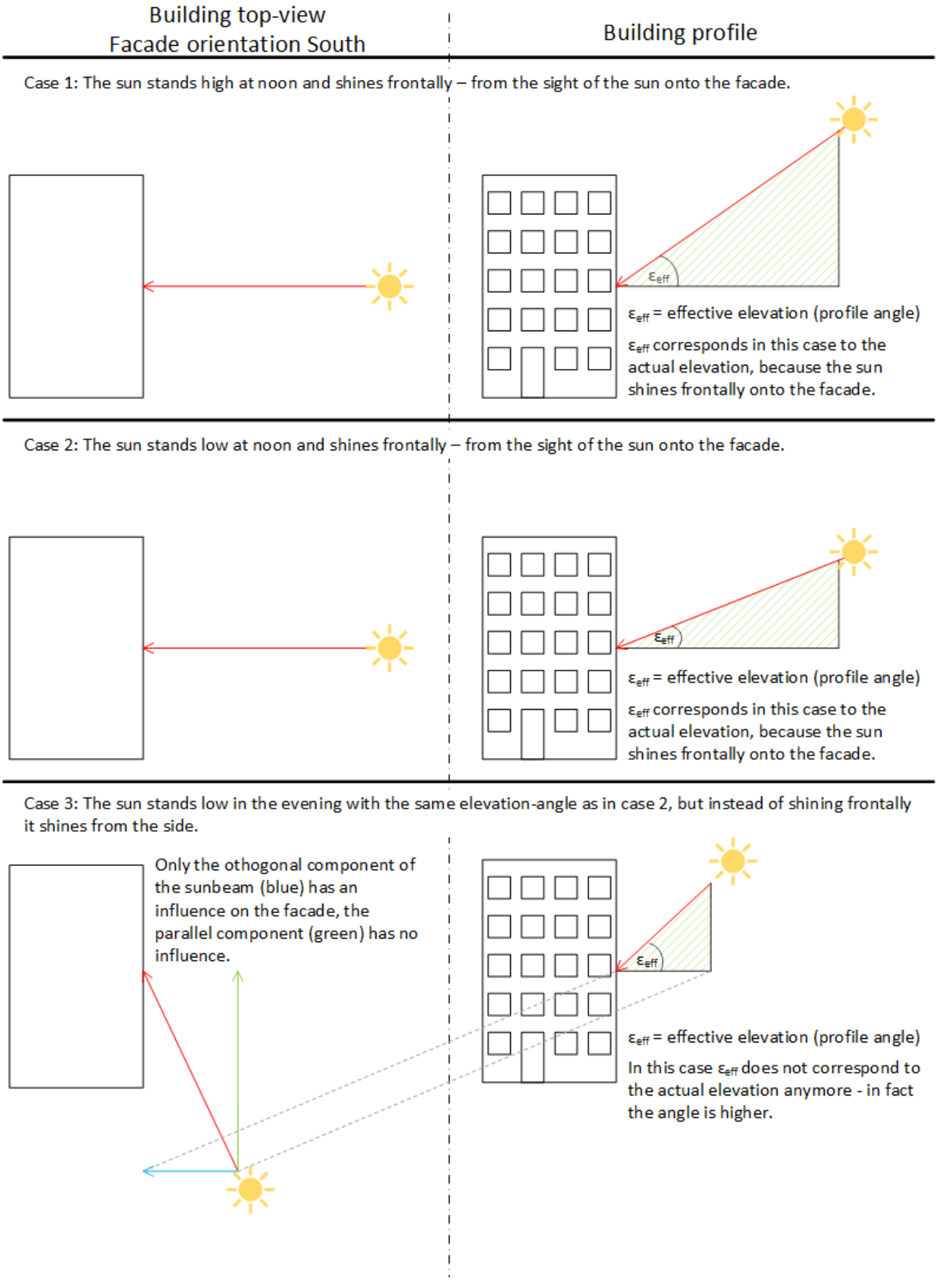


facade angle: φ > 0°



6.1.2.2.3.1.3.3.4.1 Effective elevation angle

This chapter describes the relationship between the incidence of sunlight on the facade and the elevation angle.



Case 1 shows a typical incidence of sunlight on a south-facing facade at midday.

The sun is high compared to the morning and evening.

The green shaded triangle represents the elevation angle ε.

Seen from the direction of the sun (azimuth), only the orthogonal component has a direct effect on the facade!

In this case, the elevation angle ε is therefore equal to the effective elevation angle ε_{eff} or the profile angle for the calculation.

Case 2 shows a typical incidence of sunlight at midday on a south-facing facade, but lower.

The green shaded triangle represents the elevation angle ε, which is smaller compared to case 1.

Here too, the elevation angle ε is equal to the effective elevation angle ε_{eff} or the profile angle for the calculation.

Case 3 shows a typical incidence of sunlight on a south-facing facade in the evening. The elevation angle is the same as in case 2, so case 3 is the continuation of case 1 on the same day.

The green shaded triangle from case 2 (building in profile) is now tilted forwards.

However, it represents the orthogonal lighting component. The elevation angle for this component is visibly larger than in case 2, where the sun shines orthogonally onto the facade.

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 [▶ 274] within the program.

The shading correction FB_BA_ShdCorr [▶ 366] reads the information from this list. The management function block FB_BA_ShdObjEntry [▶ 370] 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 [▶ 273].

The management function block FB_BA_FcdElemEntry [▶ 345] and the shading correction FB_BA_ShdCorr [▶ 366] 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_FcdE
lem;
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
```

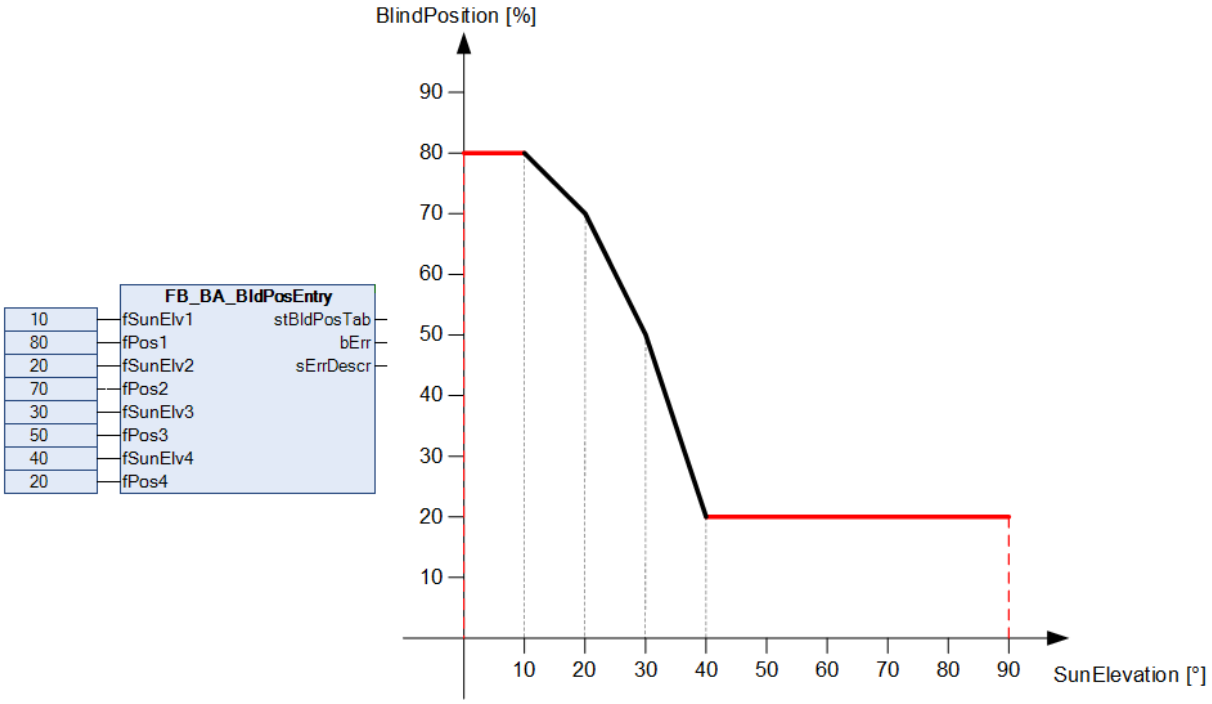
6.1.2.2.3.1.3.3.7 FB_BA_BldPosEntry

The function block FB_BA_BldPosEntry serves for the input of interpolation points for the function block FB_BARSunProtectionEx [▶ 396], 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 [▶ 396] 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 [▶ 396], 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 ≤ *fSunElv* ≤ 90 ([°] - scope source values)
- 0 ≤ *fPos* ≤ 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 [▶ 273] 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 ≤*fSunElv* ≤ 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 [▶ 396] thus increases to 6, see ST_BA_BldPosTab [▶ 273].

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
```

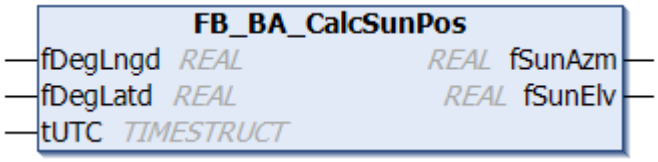
Name	Type	Description
stBldPosTab	ST_BA_BldPosTab [▶ 273]	Transfer structure of the interpolation points
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description

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

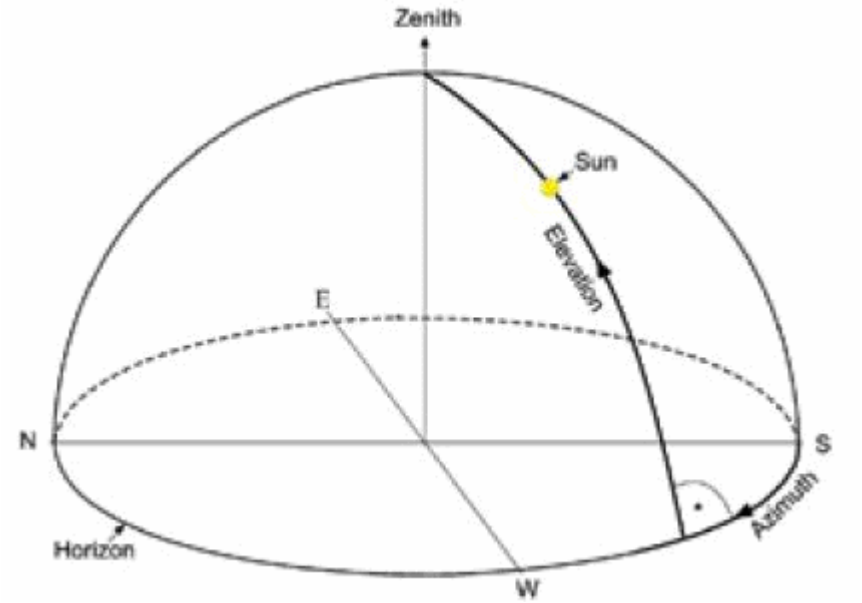
6.1.2.2.3.1.3.3.8 FB_BA_CalcSunPos



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 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 is, 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 [°]. The latitude can assume values between 0° (at the equator) and ±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°
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 [▶ 402] can be used to read this time from a target system.

Outputs

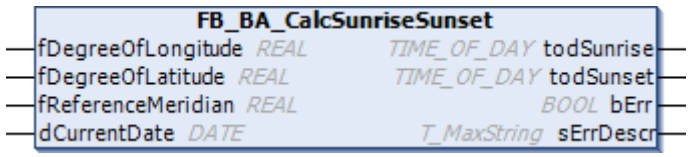
```
VAR_OUTPUT
  fSunAzm     : REAL;
  fSunElv     : REAL;
END_VAR
```

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_CalcSunriseSunset



The FB_BA_CalcSunriseSunset function block is used to calculate the sunrise and sunset by specifying the longitude, latitude, reference meridian and time.

The earth is divided into several time zones. Each time zone is associated with a reference meridian. Reference meridian for some of the time zones:

Time zone	Reference meridian	Time
GMT (Greenwich Mean Time)	$\lambda_{GMT} = 0°$	GMT + 0h
CET (Central European Time)	$\lambda_{CET} = 15°$	GMT + 1h
CEST (Central European Summer Time)	$\lambda_{CEST} = 30°$	GMT + 2h

Inputs

```
VAR_INPUT
  fDegreeOfLongitude       : REAL := 8.5;
  fDegreeOfLatitude        : REAL := 51.9;
  fReferenceMeridian       : REAL;
  dCurrentDate             : DATE;
END_VAR
```

Name	Type	Description
fDegreeOfLongitude	REAL	Longitude in degrees. Eastern longitudes are positive, western longitudes are negative. This input is internally limited to -180°...180°. Nevertheless, an error is displayed at the *bErr* output if an entry is made outside these limits.
fDegreeofLatitude	REAL	Latitude in degrees. Northern latitudes are positive, southern latitudes are negative. This input is internally limited to -66.55°...66.55°. Nevertheless, an error is displayed at output *bErr* if an entry is made outside these limits. **Info:** -66.55° and 66.55° are the latitudes of the Artic and Antarctic Circle. Beyond these limits, the sun may never rise or set on certain days. A statement about sunrise and sunset is then not possible.
fReferenceMeridian	TIMESTRUCT	Reference meridian of the time zone. Eastern meridians are positive, western meridians are negative. As the reference meridian refers to longitude, this input is also limited to -180°...180°. Nevertheless, an error is displayed at the *bErr* output if an entry is made outside these limits.
dCurrentDate	DATE	Current date.

Outputs

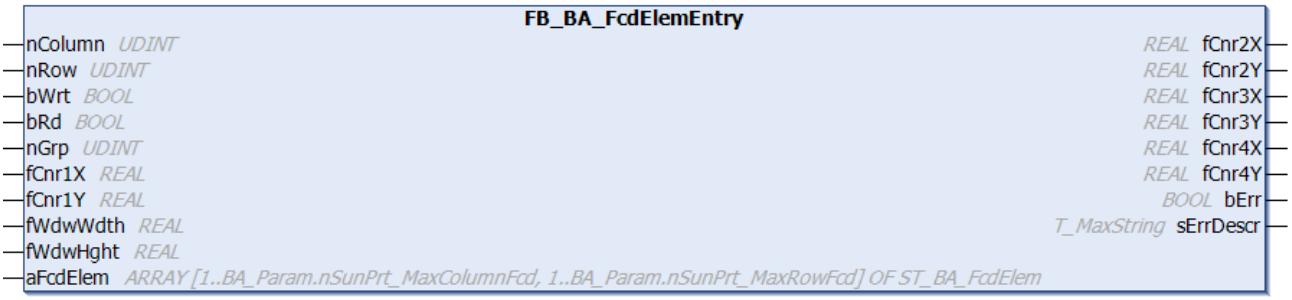
```
VAR_OUTPUT
  todSunrise     : TOD;
  todSunset      : TOD;
  bErr           : BOOL;
  sErrDescr      : T_MaxString;
END_VAR
```

Name	Type	Description
todSunrise	TOD	Sunrise. Output of hour and minute.
todSunset	TOD	Sunset. Output of hour and minute.
bErr	BOOL	Result verification for the entered values.
sErrDescr	T_MaxString	Contains the description of the error that has the highest internal priority.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.1.3.3.10 FB_BA_FcdElemEntry



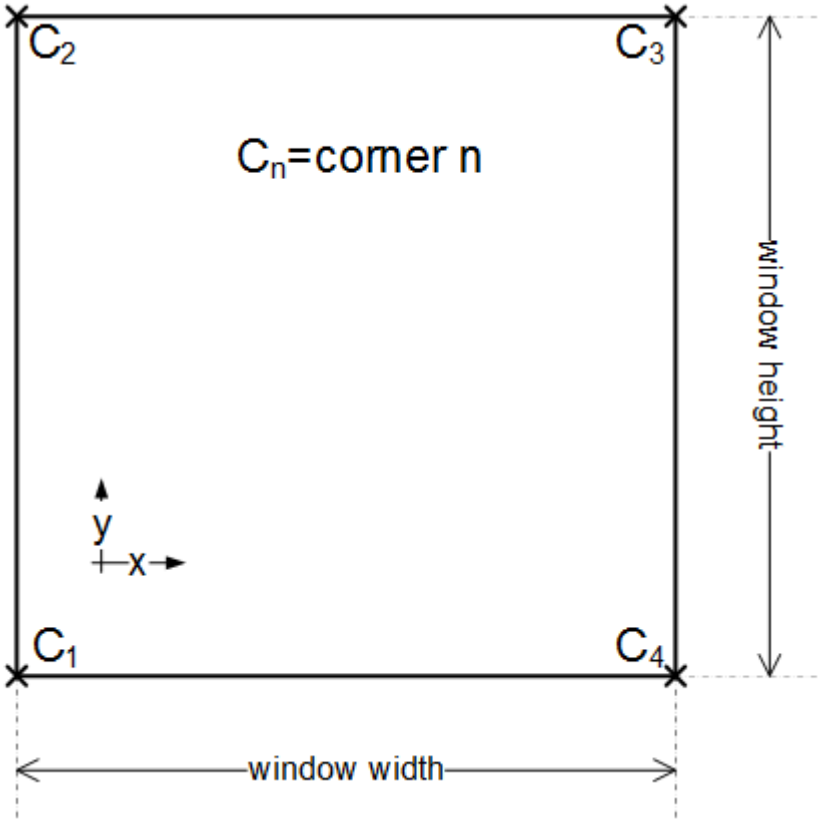
The function block FB_BA_FcdElemEntry is used to manage all facade elements (windows) of a facade, which is globally stored in a List of facade elements [▶ 339]. It is intended to facilitate inputting element information - not least with regard to using the TC3 PLC HMI. A schematic representation of the objects with description of the coordinates is shown in Shading correction: principles and definitions [▶ 325].

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 [▶ 273]). 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 C2-C4 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 [▶ 273] therefore also contains a plausibility bit *bVld*, which forwards precisely this information to the function block FB_BA_ShdCorr [▶ 366] to prevent miscalculations.

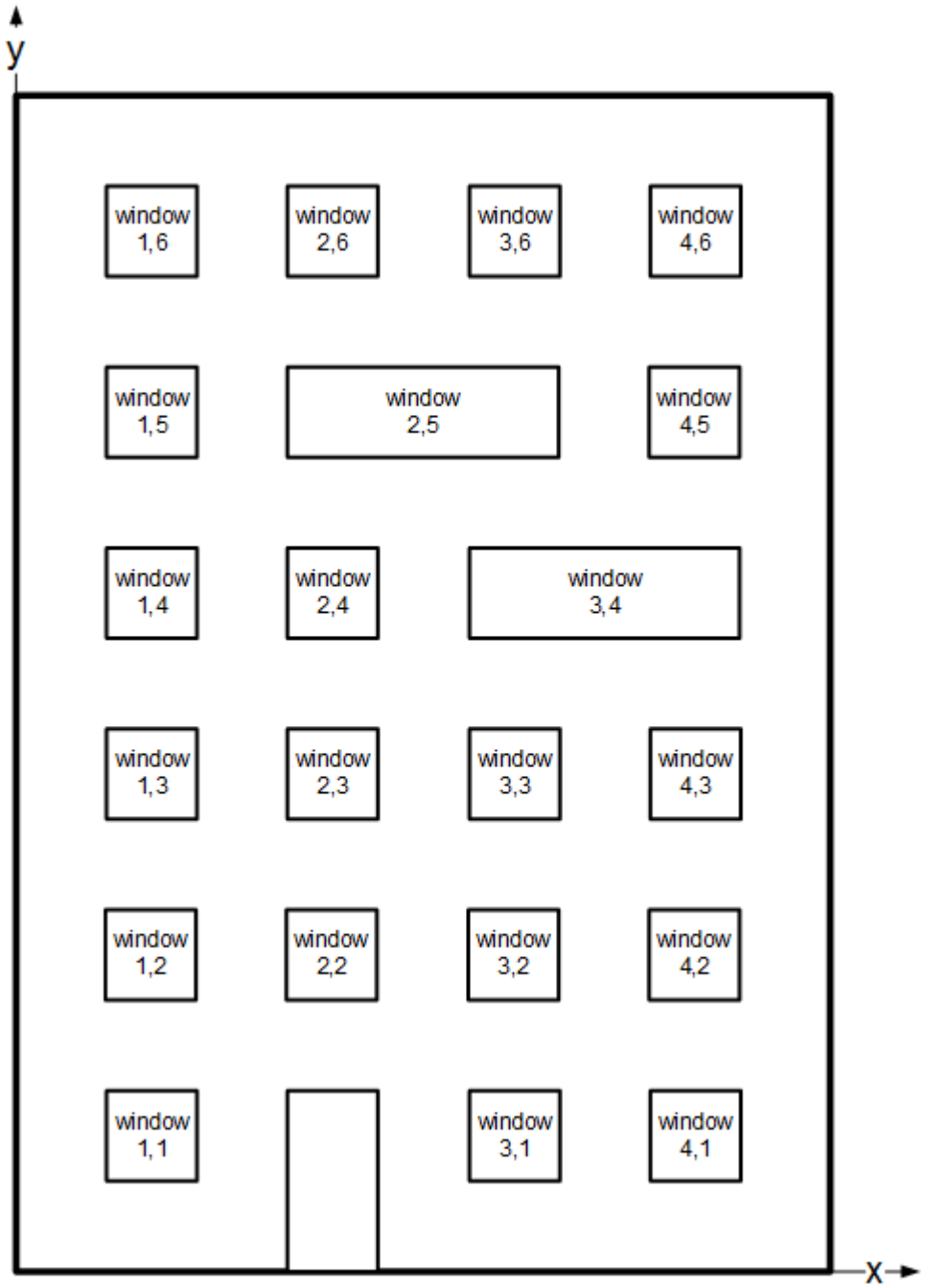
This approach is to be regarded only as a proposal. It is naturally also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on *bWrt*.

Error (bErr=TRUE)

The function block FB_BA_ShdCorr [▶ 366], 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
- *fWdwWdth* 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 [▶ 273] 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 aFcdElem.
nRow	UDINT	ditto row index. *nRow* **and** *nColumn* **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, *aFcdElem[nColumn, nRow]* is read into the function block and assigned to the input variables *nGrp* to *fWdwHght*. The resulting output variables are *fCnr2X* to *fCnr4Y*. If data are already present on the inputs *nGrp* to *fWdwHght* 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 *aFcdElem[nColumn, nRow]*.
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 [▶ 346])
fCnr2Y	REAL	Y-coordinate determined for corner point 2 of the window [m] (see function description [▶ 346])
fCnr3X	REAL	X-coordinate determined for corner point 3 of the window [m] (see function description [▶ 346])
fCnr2Y	REAL	Y-coordinate determined for corner point 3 of the window [m] (see function description [▶ 346])
fCnr4X	REAL	X-coordinate determined for corner point 4 of the window [m] (see function description [▶ 346])
fCnr2Y	REAL	Y-coordinate determined for corner point 4 of the window [m] (see function description [▶ 346])
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

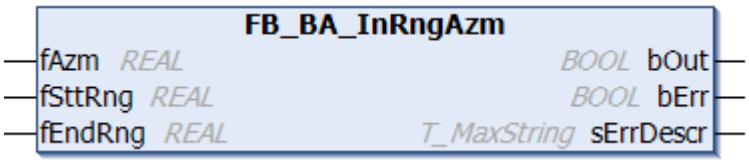
```
VAR_IN_OUT
  aFcdElem : ARRAY[1..Param.nSunPrt_MaxColumnFcd, 1..Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Name	Type	Description
aFcdElem	ARRAY OF ST_BA_FcdElem [▶ 273]	List of facade elements [▶ 339]

Requirements

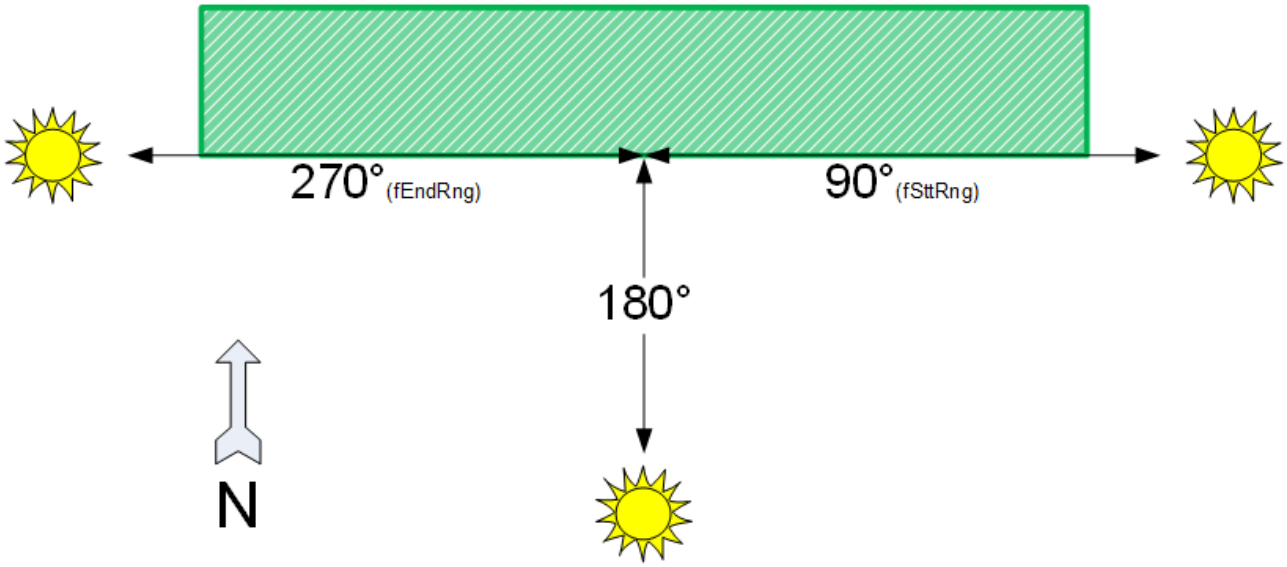
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.11 FB_BA_InRngAzm

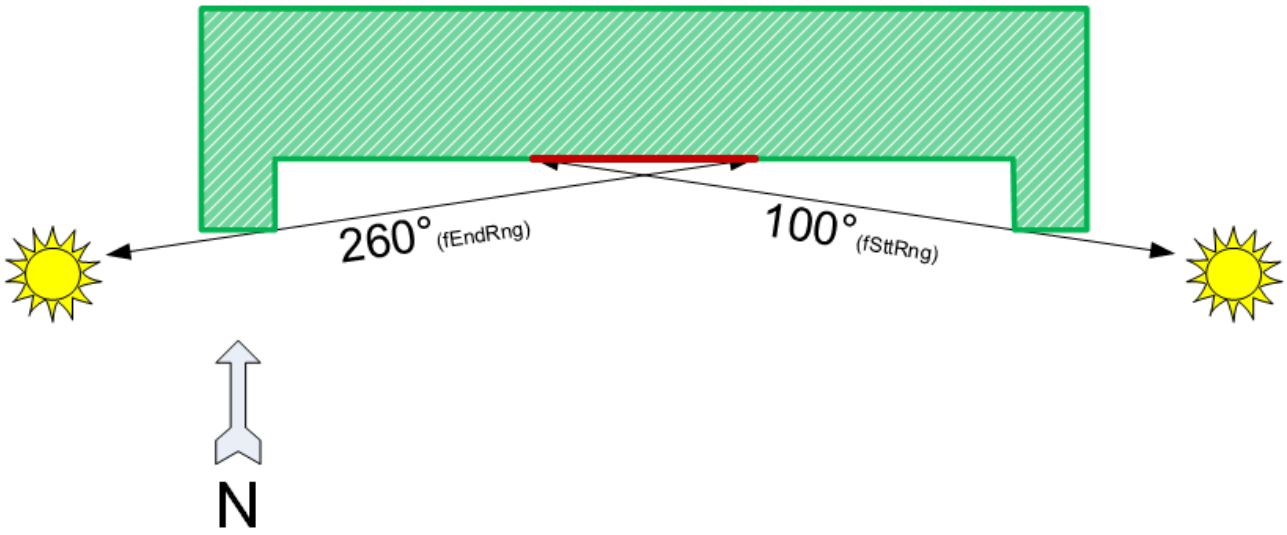


This function block FB_BA_InRngAzm checks whether the current azimuth angle (horizontal position of the sun) lies within the limits entered. As can be seen in the overview [▶ 332], the function block provides an additionally evaluation as to whether the sun protection of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

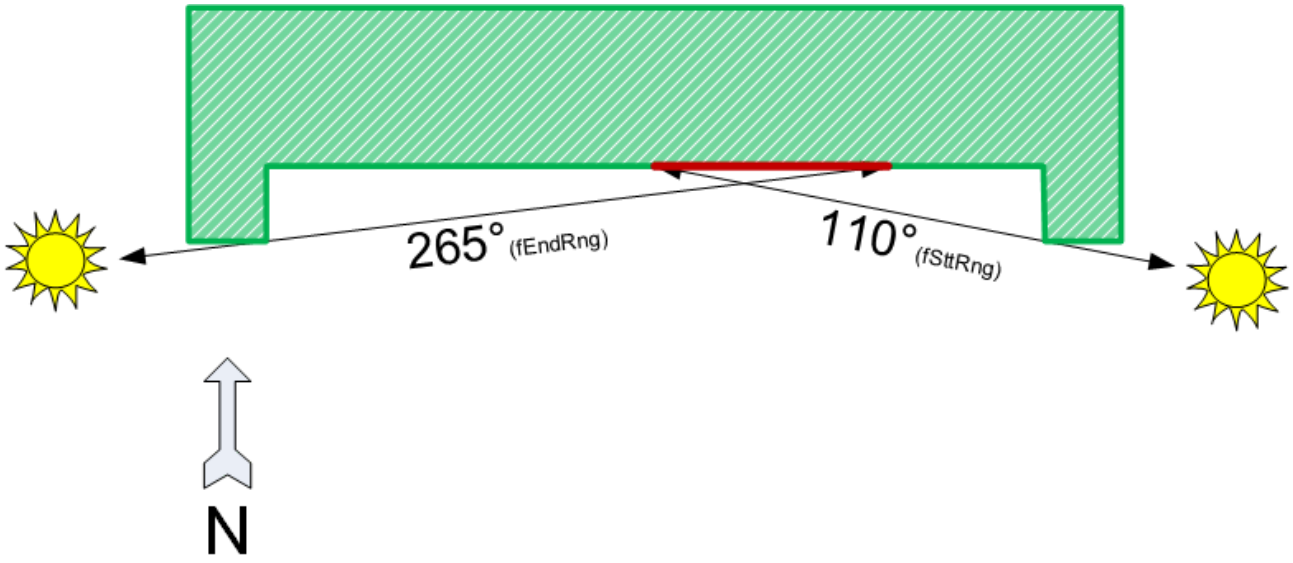
The sun incidence azimuth angle on a smooth facade will always be *facade orientation-90°... facade orientation+90°*.



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
```

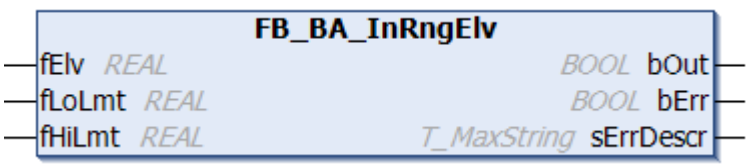
Name	Type	Description
bOut	BOOL	The facade element is in the sun if the output is TRUE.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description

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

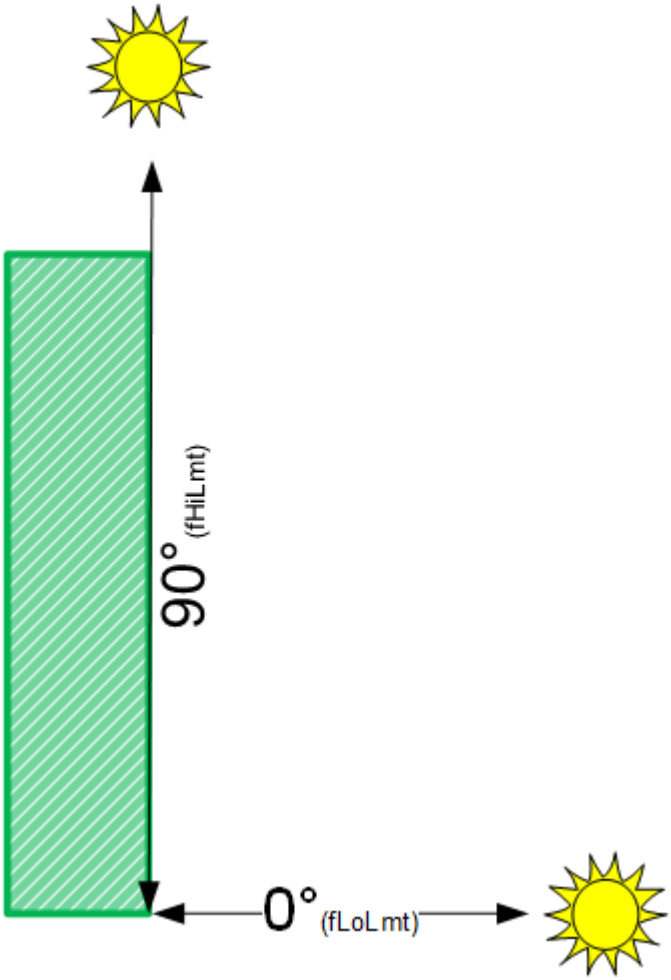
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.3.3.12 FB_BA_InRngElv

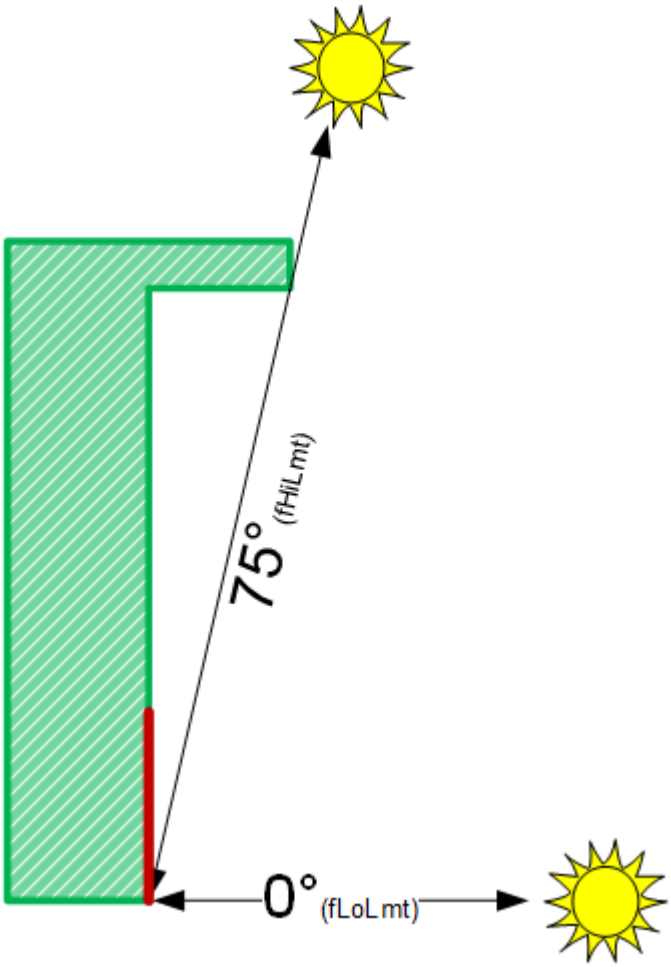


This function block FB_BA_InRngElv checks whether the current elevation angle (vertical position of the sun) lies within the limits entered. As can be seen in the overview [▶ 332], the function block provides an additionally evaluation as to whether the sun protection of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

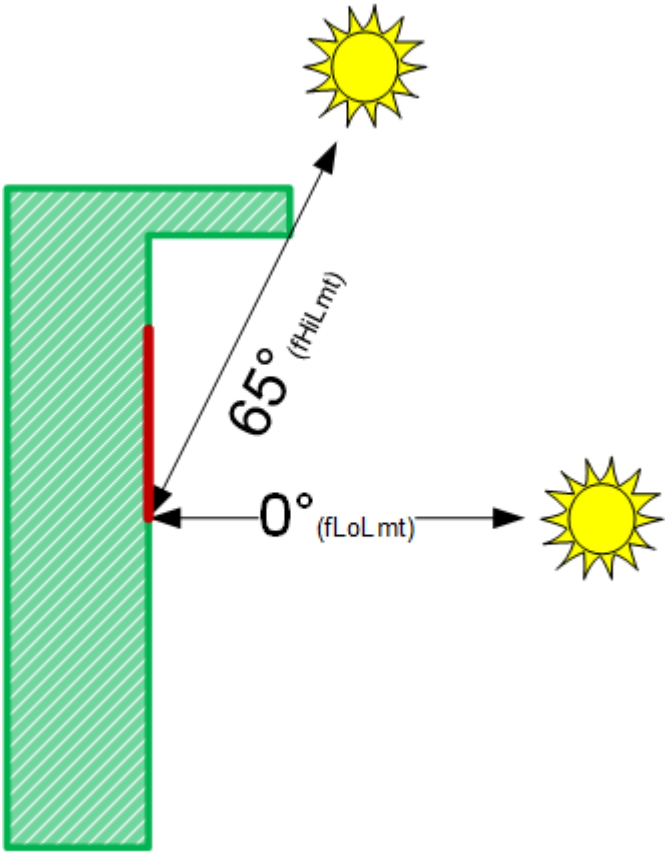
A normal vertical facade is irradiated by the sun at an angle of elevation of 0° to maximally 90°.



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
```

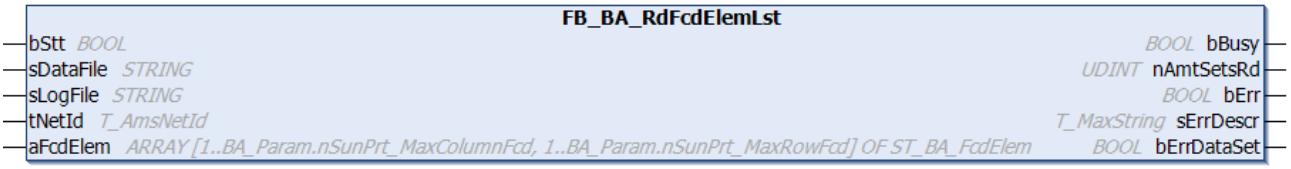
Name	Type	Description
bOut	BOOL	The facade element is in the sun if the output is TRUE.
bErr	BOOL	In case of a fault, e.g. if warning stages are active, this output is set to TRUE.
sErrDescr	T_MAXSTRING	Contains the error description

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

6.1.2.2.3.1.3.3.13 FB_BA_RdFcdElemLst



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 [▶ 339]. 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 [▶ 339]. 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 [▶ 366], does not accept elements with the group entry '0'.

EN_FacadeElements.xls

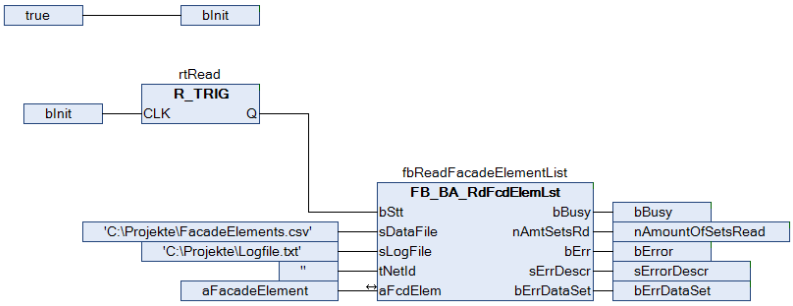
	A	B	C	D	E	F	G	H	I	J
1	Number	Description		IndexColumn	IndexRow	Window-Width	Window-Height	P1x	P1y	Group
2				(Axis)	(Floor)	[m]	[m]	[m]	[m]	
3		Text								
4	1	Description	@	1	1	1,2	1,3	1,5	1	2
5	2	Description	@	0	1	1,2	1,3	2,7	1	2
6	3	Description	@	3	1	1,2	1,3	4,4	1	2
7	4	Description	@	4	1	1,2	1,3	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.

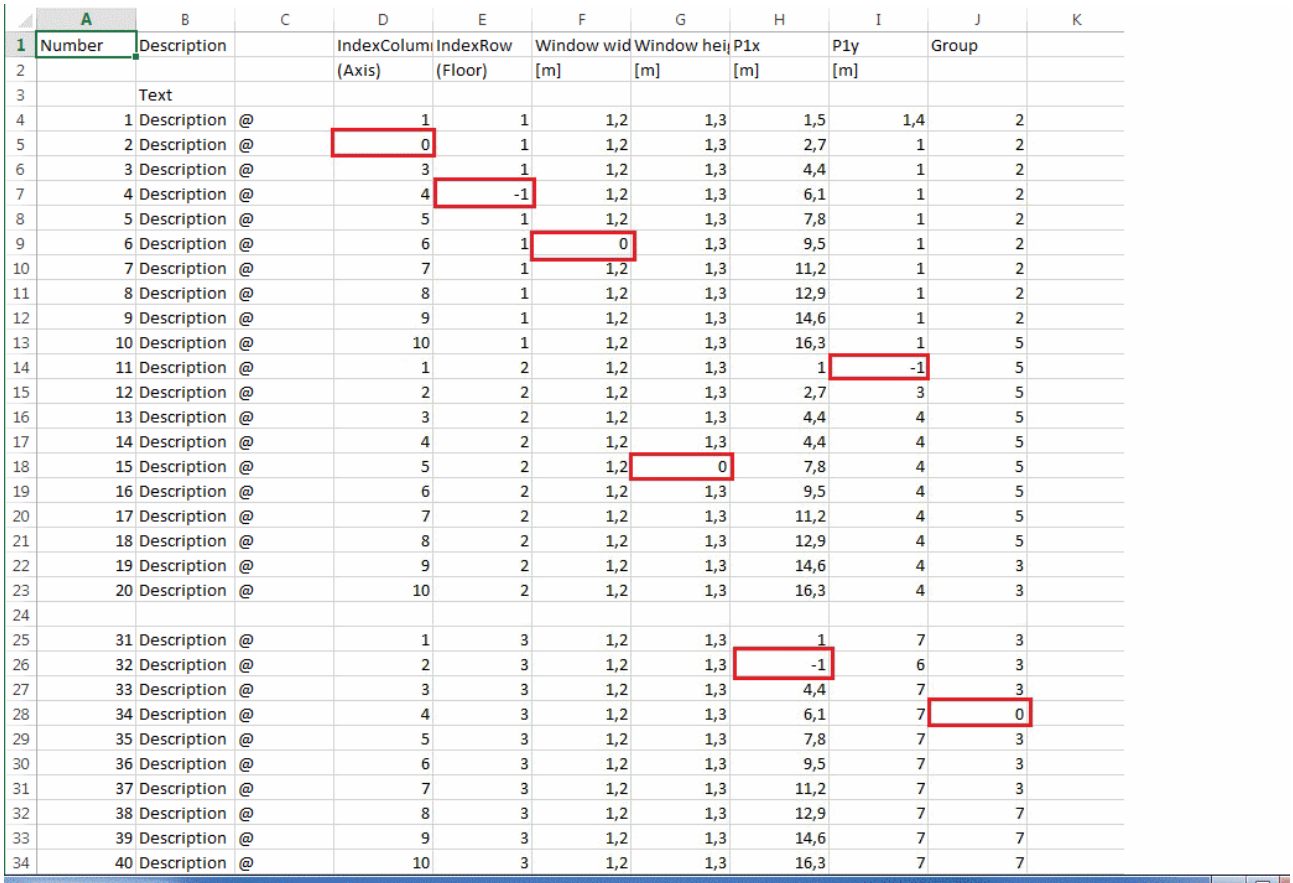
Program sample

```
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
```

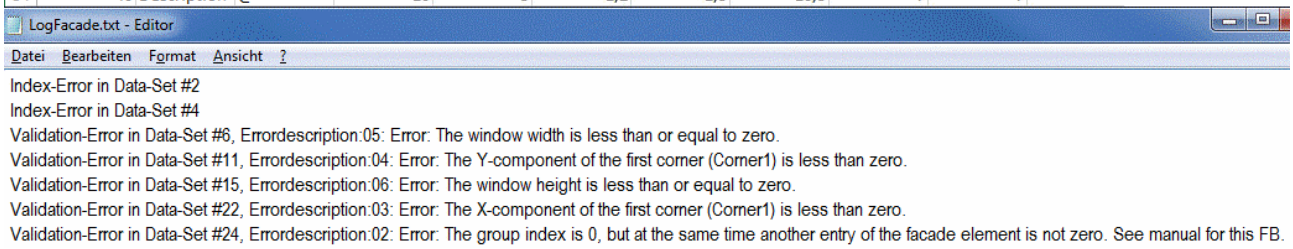


In this sample the variable *bInit* is initially set to TRUE when the PLC starts. Hence, the input *bStt* on the function block *fbReadFacadeElementList* receives a once-only rising edge that triggers the reading process. The file *FacadeElements.csv* is read, which is located in the folder *C:\Projekte*. 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	IndexRow	Window wid	Window hei	P1x	P1y	Group	
2				(Axis)	(Floor)	[m]	[m]	[m]	[m]		
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	



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 [▶ 370] 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	STRING	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	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. For the local computer an empty string may be specified.

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	ARRAY OF ST_BA_FcdElem [▶ 273]	List of facade elements [▶ 339]

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

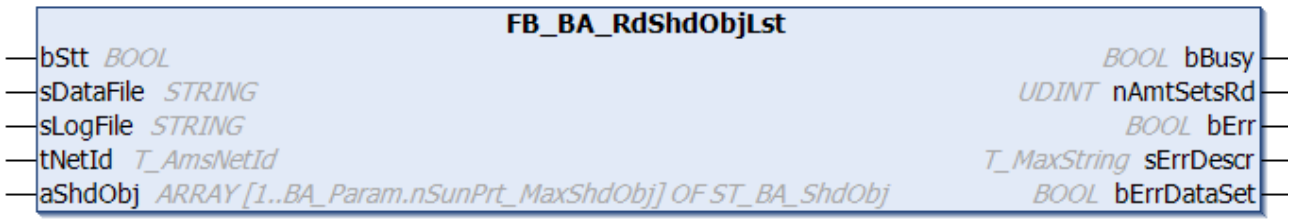
Error description
01: File handling error: Opening the log file - the ADS error number is stated.
02: File handling error: Opening the data file - the ADS error number is stated.
03: File handling error: Reading the data file - the ADS error number is stated.
04: Error: During reading of the data file 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_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 [▶ 339]. 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_ShdObjEntry [▶ 370]).
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).
- 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.
- The table 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

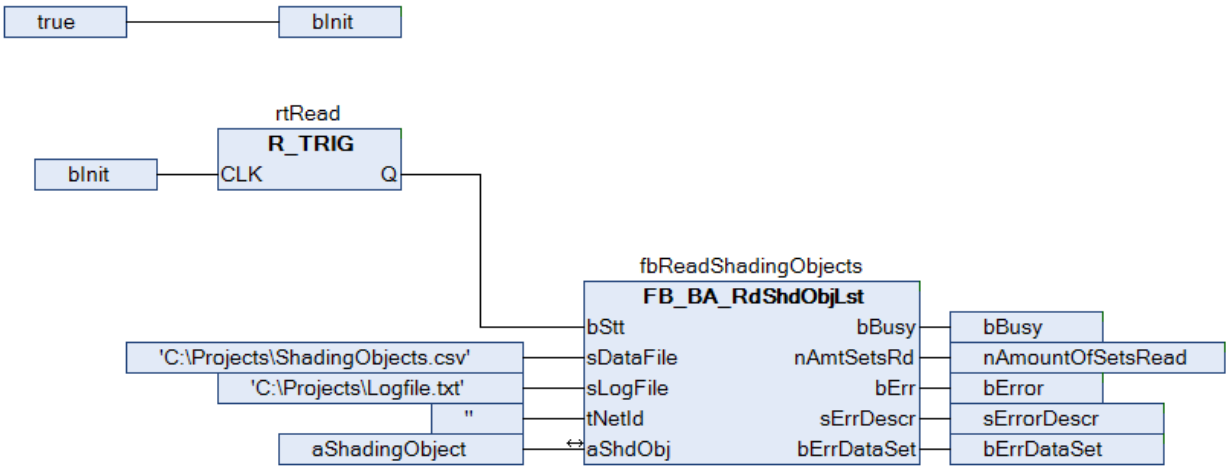
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Number	Description		Type	Begin	End	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Tetragon	(Month)	(Month)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Globe									
4		Text											
5	1	Description	@	0	1	2	-94,75	0	36,06	11	-70,71	11	68,59
6	2	Description	@	0	1	2	-23,33	0	9,9	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

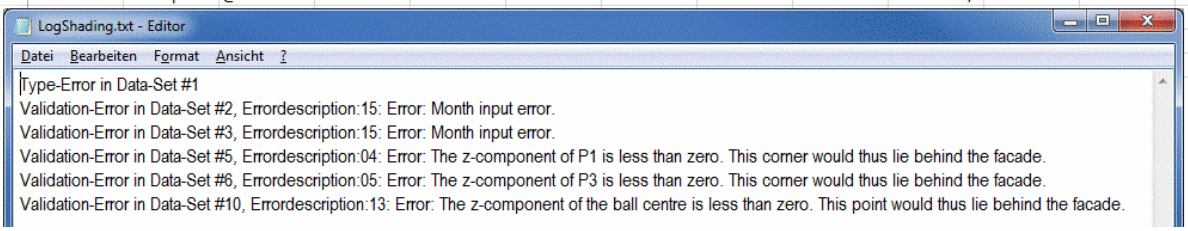
```
PROGRAM ReadShadingObjects
VAR
    bInit                       :   BOOL;
    rtRead                      :   R_TRIG;
    fbReadShadingObjects        :   FB_BA_RdShdObjLst;
    aShadingObject              :   ARRAY [1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;

    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			



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 [▶ 370] and allocated an error description. The third error likewise occurred after the internal validation.

Please note that the missing coordinates are filled in as follows:

fP2x = fP1x; fP2z = fP1z; fP4x = fP3x; fP4y = fP1y; fP4z = fP3z;

Error messages complaining about the equality of two points must be checked against these equations.

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	STRING	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	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. For the local computer an empty string may be specified.

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_ShdObj;
END_VAR
```

Name	Type	Description
aShdObj	ARRAY OF ST_BA_ShdObj [▶ 274]	List of shading objects [▶ 339]

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

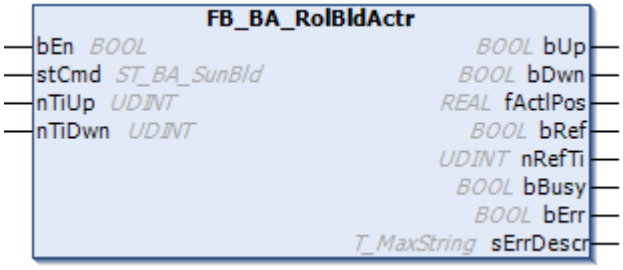
Error description
01: File handling error: Opening the log file - the ADS error number is stated.
02: File handling error: Opening the data file - the ADS error number is stated.
03: File handling error: Reading the data file - the ADS error number is stated.
04: Error: During reading of the data file 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

6.1.2.2.3.1.3.3.15 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 [▶ 274]. In addition, the positioning telegram stSunBld [▶ 274] 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 [▶ 388].

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 [▶ 274]. 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 + 5 s.

The referencing process can be interrupted by a manual "down" command or by repositioning.

After a plant restart, the function block executes a reference run.

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 *bUp* and *bDwn* and the function block remains in a state of rest.
stSunBld	ST_BA_SunBld [▶ 274]	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;
  bBusy           : BOOL;
  bErr            : BOOL;
  sErrDescr       : T_MAXSTRING
END_VAR
```

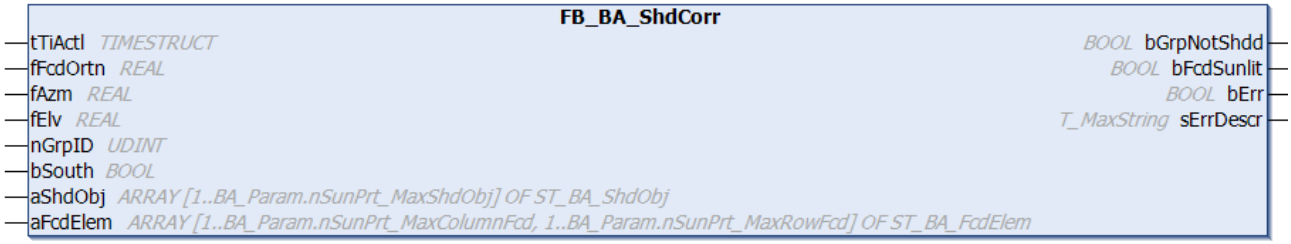
Name	Type	Description
bUp	BOOL	Roller blind control output up
bDwn	BOOL	Roller blind control output down
fActlPos	READ	Current position in percent.
bRef	BOOL	The roller blind is in referencing mode, i.e. the output *bUp* is set for the complete travel-up time + 5s. Only a manual "down" command can move the roller blind in the opposite direction and terminate this mode.
nRefTi	UDINT	Referencing countdown display [s]
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

6.1.2.2.3.1.3.3.16 FB_BA_ShdCorr



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 [▶ 339] 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 [▶ 339] 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.

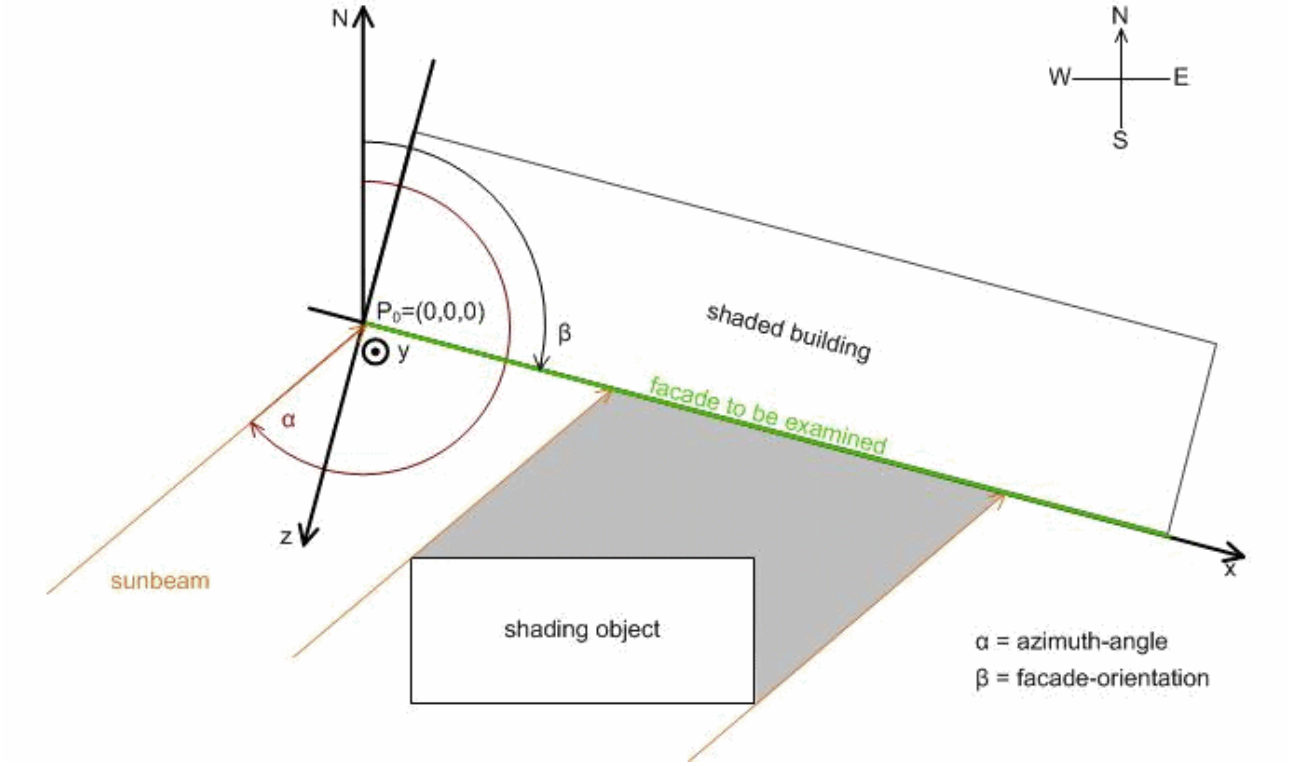
On the basis of the facade orientation (*fFcdOrtn*), the direction of the sun (*fAzm*) and the sun elevation (*fElv*), a calculation can be performed for each corner of a window to check whether this lies in a shaded area. A window group is considered to be completely shaded if all corners are shaded.

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):

Viewing direction	Facade orientation
North	β=0°
East	β=90°
South	β=180°
West	β=270°

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°

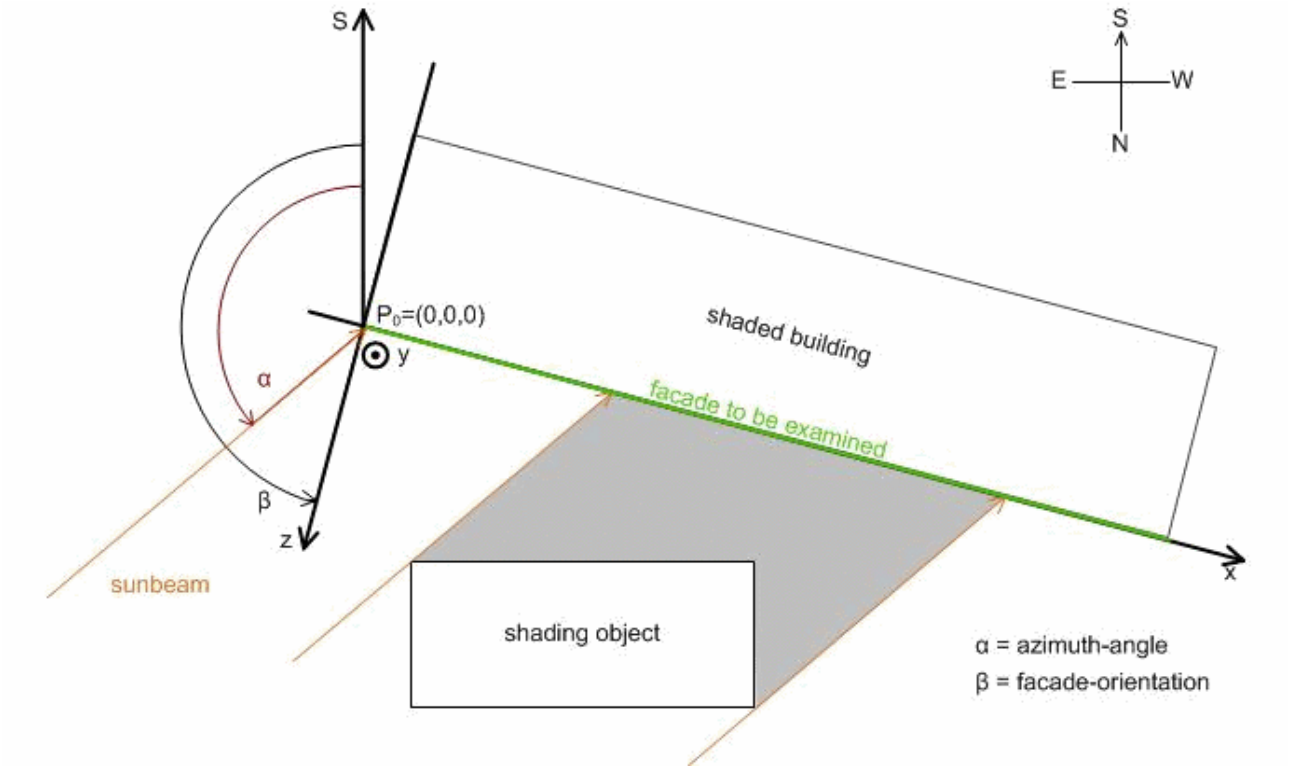


In addition, a calculation is also not required, if the sun has not yet risen, i.e. the sun elevation is below 0°. In both cases the output *bFcdSunlit* is set to FALSE.

The situation is different for the southern hemisphere. The following applies to the facade orientation (looking out the window):

Viewing direction	Facade orientation
South	β=0°
East	β=90°
North	β=180°
West	β=270°

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
  tTiActl    : TIMESTRUCT;
  fFcdOrtn   : REAL;
  fAzm       : REAL;
  fElv       : REAL;
  nGrpID     : DINT;
  bSouth     : BOOL;
  aShdObj    : ARRAY[1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Name	Type	Description
tTiActl	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 [▶ 345]). 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	BOOL	FALSE: Calculations refer to conditions in the northern hemisphere - TRUE: in the southern hemisphere
aShdObj	ARRAY OF ST_BA_ShdObj [▶ 274]	List of shading objects [▶ 339].

/ 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	ARRAY OF ST_BA_FcdElem [▶ 273]	List of facade elements [▶ 339]

Outputs

```
VAR_OUTPUT
  bGrpNotShdd    : BOOL;
  bFcdSunlit     : BOOL;
  bErr           : BOOL;
  sErrDescr      : T_MAXSTRING;
END_VAR
```

Name	Type	Description
bGrpNotShdd	BOOL	Is TRUE as long as the window group is not calculated as shaded.
bFcdSunlit	BOOL	This output is set to TRUE if the sun is shining on the facade. See description above.
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 window group (group of facade elements): Index is less than or equal to 0.
02: Error: A facade element is invalid.

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_ShdObjEntry

The function block FB_BA_ShdObjEntry serves for the administration of all shading elements in a facade, which are globally saved in a list of shading elements [▶ 339]. 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 [▶ 325].

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 [▶ 274]). 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 f*P4z* 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*, f*P1y*, f*P1z*, f*P2y*, f*P3x*, f*P3y* and f*P3z* 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 *nIId* upon a positive edge on *bWrt* , regardless of whether they represent valid values or not. The element structure ST_BA_ShdObj [▶ 274] therefore contains a plausibility bit *bVId*, which forwards precisely this information to the function block FB_BA_ShdCorr [▶ 366] 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;
  fP1x         : REAL;
  fP1y         : REAL;
  fP1z         : 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 *aShdObj*. The variable *nId* **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	BOOL	The information of the selected element, *aShdObj[nId]*, is read into the function block with a positive edge at this input and assigned to the input variables *fP1x* to eType and the output variables *fP2x* to *fP4z*. If at this time data have already been applied to the inputs *fP1x* to eType, the previously read data are immediately overwritten with these.
bWrt	BOOL	A positive edge writes the values applied to inputs *fP1x* to *eType* and the values determined and assigned to outputs *fP2x* to *fP4z* to the selected field element *aShdObj[nId]*.
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_ShdObjType [▶ 269]	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_ShdObj;
END_VAR
```

Name	Type	Description
aShdObj	ARRAY OF ST_BA_ShdObj [▶ 274]	List of shading objects [▶ 339]

Outputs

```
VAR_OUTPUT
  fP2x       : REAL;
  fP2z       : REAL;
  fP4x       : REAL;
  fP4y       : REAL;
  fP4z       : REAL;
  bErr       : BOOL;
  sErrDescr  : T_MAXSTRING;
END_VAR
```

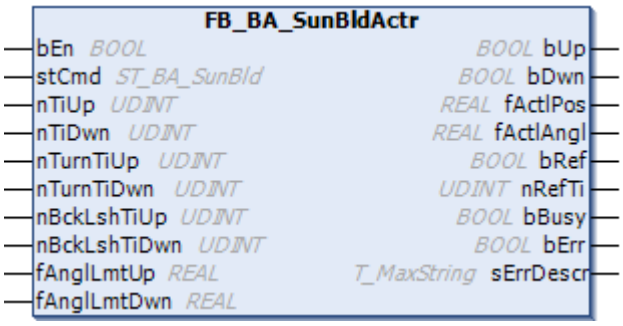
Name	Type	Description
fP2x	REAL	Determined X-coordinate of point 2 of the shading element (rectangle) [m] (see "Note [▶ 370]" above).
fP2Z	REAL	Determined Z-coordinate of point 2 of the shading element (rectangle) [m] (see "Note [▶ 370]" above)
fP4x	REAL	Determined X-coordinate of point 4 of the shading element (rectangle) [m] (see "Note [▶ 370]" above).
fP4y	REAL	Determined Y-coordinate of point 4 of the shading element (rectangle) [m] (see "Note [▶ 370]" above).
fP4z	REAL	Determined Z-coordinate of point 4 of the shading element (rectangle) [m] (see "Note [▶ 370]" above).
bErr	BOOL	Contains the error description.
sErrDescr	T_MAXSTRING	Contains the error description

Error description
01: Error: The input *nId* is outside the permissible limits 1...*nMaxShdObj*.
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 *eType* - 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

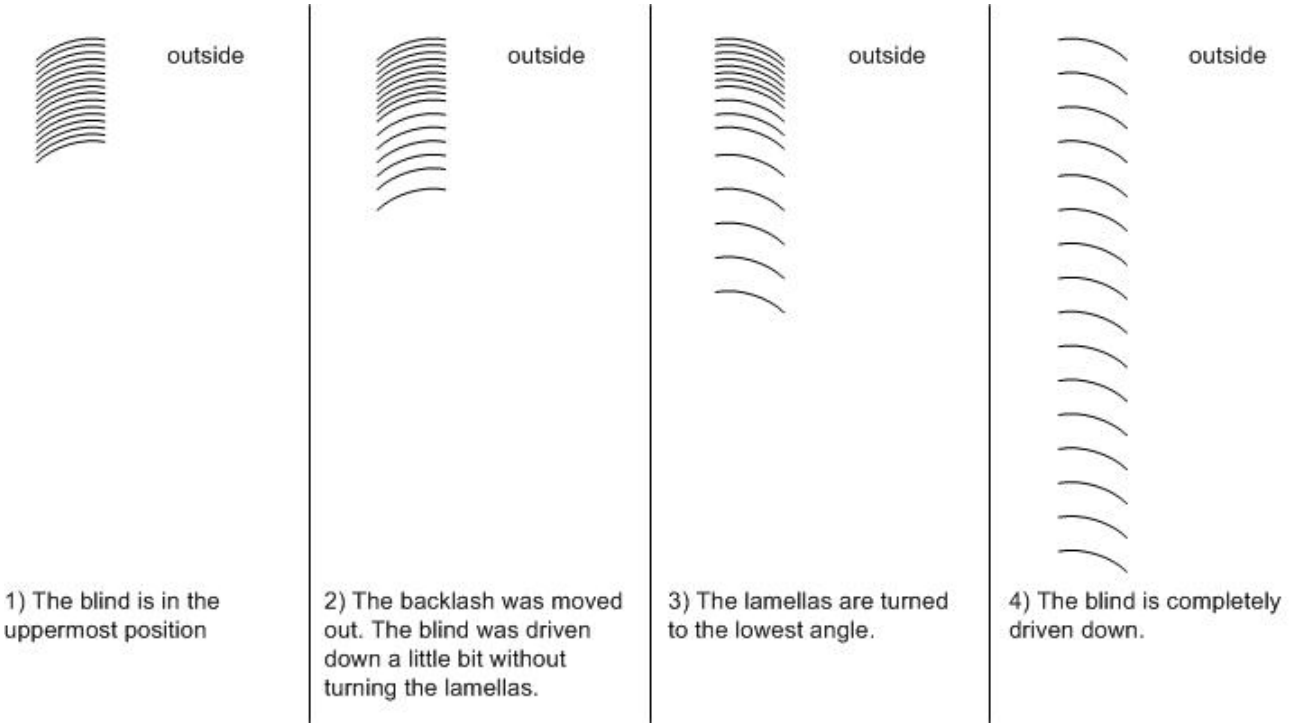
6.1.2.2.3.1.3.3.18 FB_BA_SunBldActr



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 [▶ 274]. In addition, the positioning telegram stSunBld [▶ 274] 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 [▶ 388].

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):

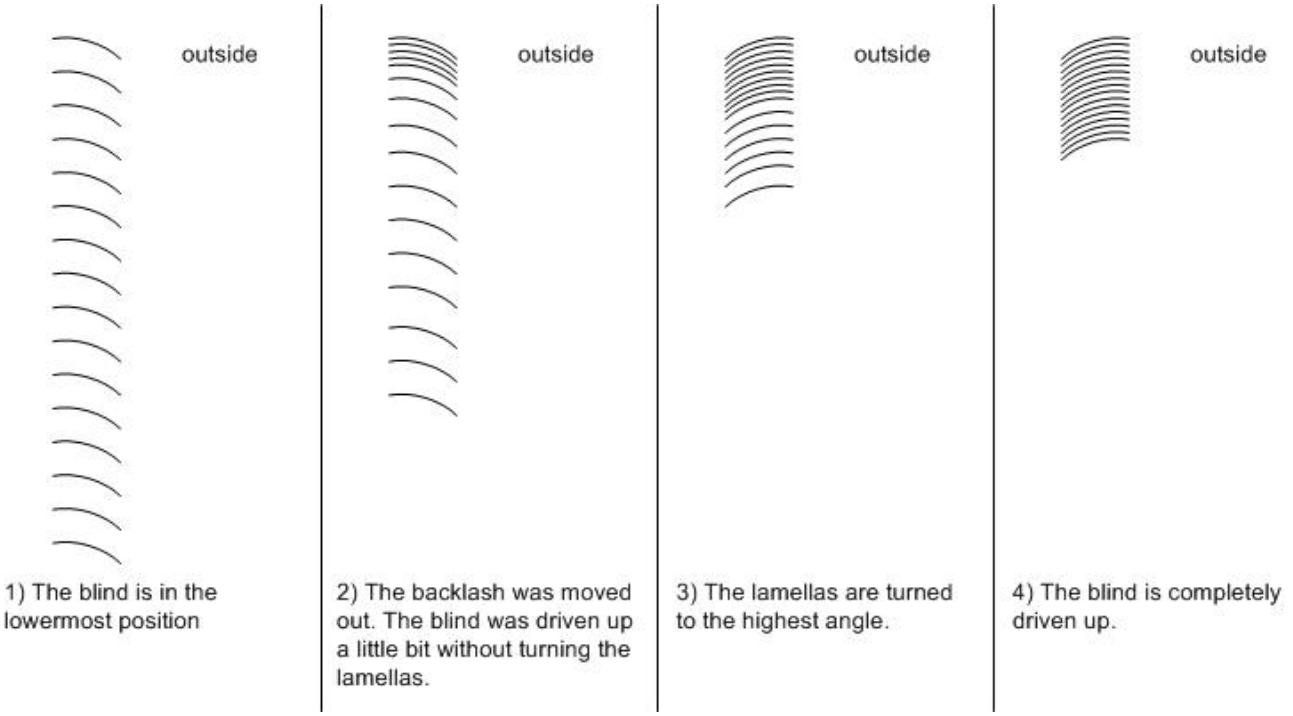
Downward travel profile:



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

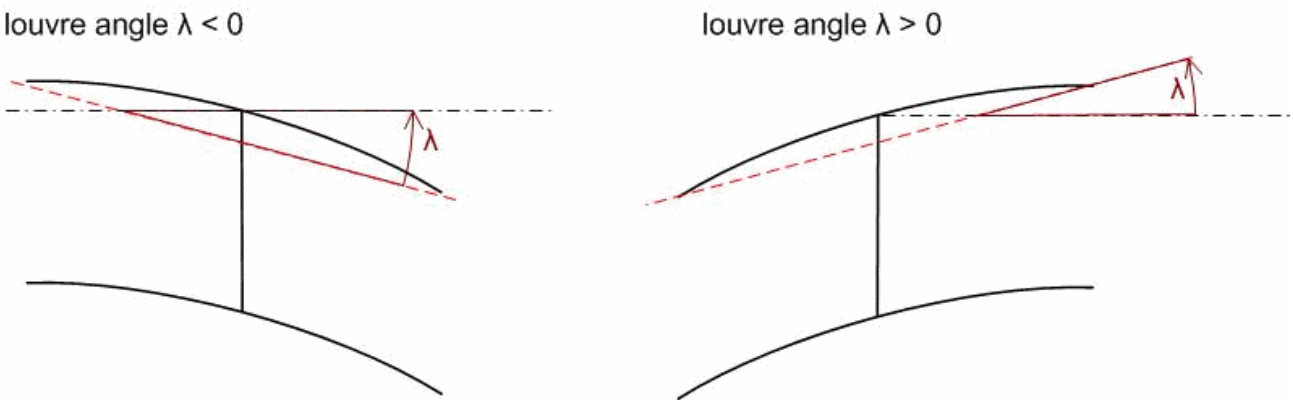
For the calculation of the (height) position and the slat angle, the following times now have to be determined for both the upward and downward movement:

- 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.



Functioning

As a rule, the function block controls the blind based on the information from the positioning telegram stSunBld [▶ 689]. 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 travel-up time. The position is then in any case "0" and the lamella angle is at its maximum. Since a blind positioning without encoder is naturally always error-prone, it is important to reference automatically as often as possible: every time the position "0" is to be approached (the angle does not matter), the 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 up time + 5 s.

The referencing process can be interrupted by a manual "down" command or by repositioning.

After a plant restart, the function block executes a reference run.

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 +/-4°.

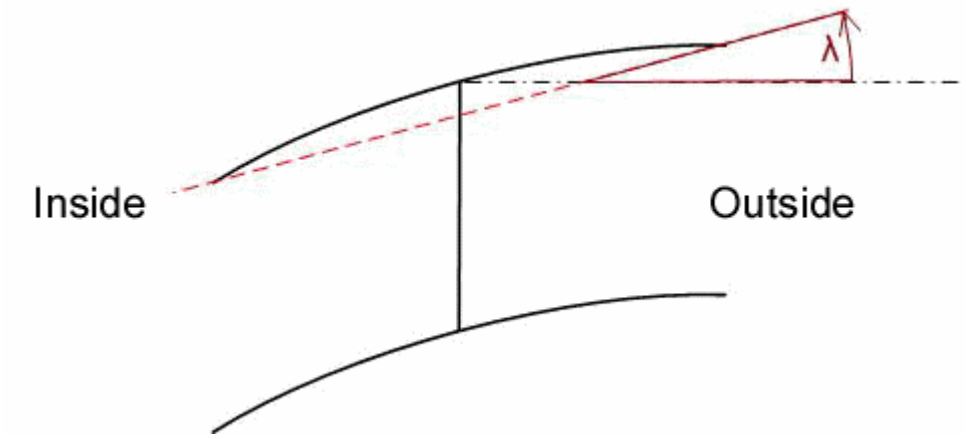
Inputs

```
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 *bUp* and *bDwn* and the function block remains in a state of rest.
stCmd	ST_BA_SunBld [▶ 274]	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.

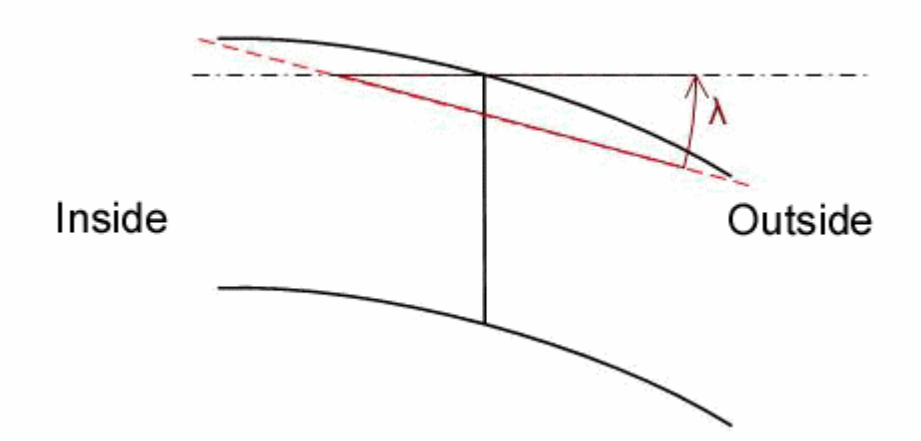
The slat angle λ, as defined above, is then typically greater than zero.



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;
  bBusy         : BOOL;
  bErr          : BOOL;
  sErrDesc      : T_MAXSTRING;
END_VAR
```

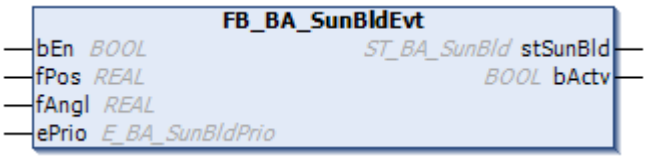
Name	Type	Description
bUp	BOOL	Control output for blind up
bDwn	BOOL	Control output for blind down
fActlPos	REAL	Current position in percent
fActlAngl	REAL	Current lamella angle [°]
bRef	BOOL	The blind is in referencing mode, i.e. the output *bUp* is set for the complete travel-up time + 5s. Only a manual "down" command can move the blind in the opposite direction and terminate this mode.
nRefTi	UDINT	Referencing countdown display [s]
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: Lamella 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

6.1.2.2.3.1.3.3.19 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 *bActv* is set in the positioning telegram stSunBld [▶ 274] at the output and the values *fPos* entered at the In-Out variables for the blind height [%] and *fAngl* for the lamella angle [°] are forwarded 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 [▶ 274] is reset and the positions for height and angle are set to "0". The priority function block (e.g. FB_BA_SunBldPrioSwi4 [▶ 599]) enables a function with lower priority to take over the control by resetting.

Inputs

```
VAR_INPUT
  bEn       : BOOL;
  fPos      : REAL;
  fAngl     : 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 ST_BA_SunBld [▶ 274]. 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	Lamella angle of the blind [°] in case of activation.

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio     : E_BA_SunBldPrio := E_BA_SunBldPrio.eScene1;
END_VAR
```

Name	Type	Description
ePrio	E_BA_SunBldPrio [▶ 269]	Priority of the active telegram.

Outputs

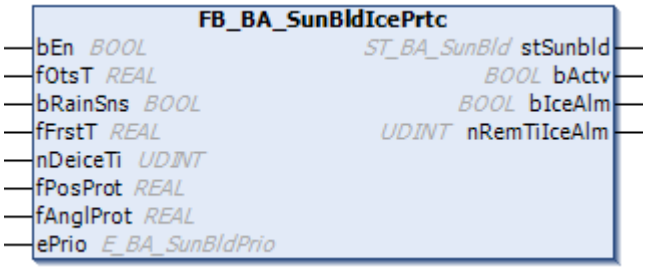
```
VAR_OUTPUT
  stSunBld    : ST_BA_SunBld;
  bActv       : BOOL;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output structure of the blind positions.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] 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

6.1.2.2.3.1.3.3.20 FB_BA_SunBldIcePrtc



The function block FB_BA_SunBldIcePrtc deals with direction-independent anti-icing.

Weather protection has the highest priority for blind control (see Overview [▶ 332]) and should ensure that the blind is not damaged by ice.

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* (lamella 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 ST_BA_SunBld [▶ 274]. A FALSE signal resets the active flag again and sets position and angle to zero. *bActv* is FALSE. This means that another function takes over control of the blind via the priority controller.
fOtsT	REAL	Outside temperature [°C]
bRainSns	BOOL	Input for a rain sensor.
fFrstT	REAL	Icing up temperature limit value [°] Celsius. This value may not be greater than 0. Otherwise an error is output.
nDeiceTi	UDINT	Time until the deicing of the blind after icing up [s]. After that the ice alarm is reset.
fPosProt	REAL	Height position of the blind [%] in the case of protection.
fAnglProt	REAL	Lamella 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 [▶ 269]	Priority of the active telegram.

Outputs

```
VAR_OUTPUT
  stSunBld          : ST_BA_SunBld;
  bActv             : BOOL;
  bIceAlm           : BOOL;
  nRemTiIceAlm      : UDINT;
END_VAR
```

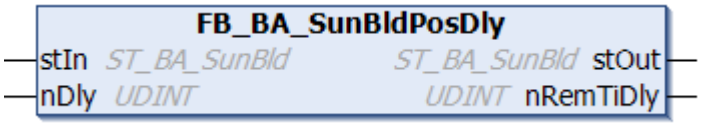
Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output structure of the blind positions.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] and is solely used to indicate whether the function block sends an active telegram.
bIceAlm	BOOL	Indicates the icing up alarm.
nRemTiIceAlm	UDINT	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 (*fFrstT*), the remaining number of seconds until the 'all-clear' signal is given (*bIceAlm* = 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) 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

6.1.2.2.3.1.3.3.21 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 overall current values.

This function block relays automatic commands from the input telegram stIn [▶ 274] to the output telegram stOut [▶ 274] with a delay. A distinction is made between three cases

1. the blind position *fPos* has changed in automatic mode (*bManMode* = FALSE in telegram *stIn*)
2. the lamella angle *fAngl* 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*)

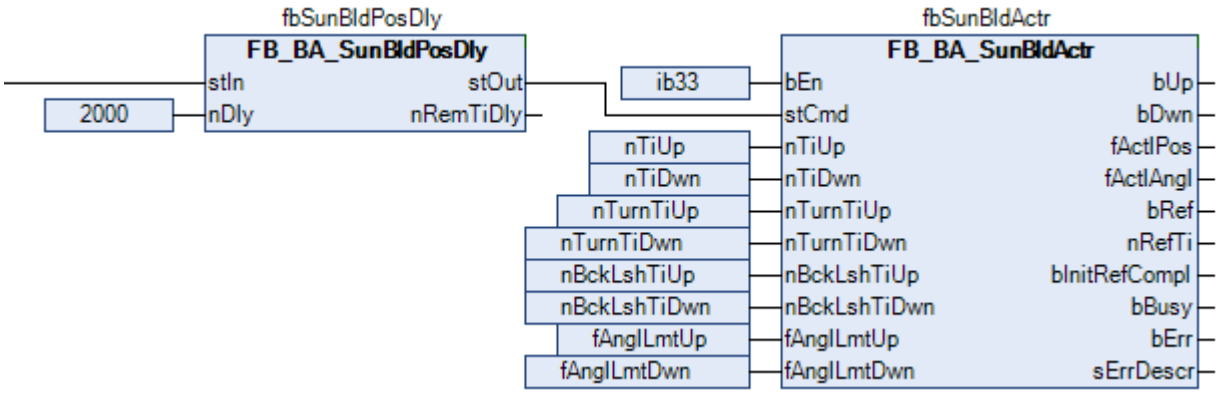
The output telegram *stOut* is always a direct copy of the input telegram *stIn*. In these three cases, however, the output telegram *stOut* is fixed for the time of *nDly* [ms].

This ensures that the blind controlled via the function block FB_BA_SunBldActr [▶ 374] 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 [▶ 274]	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 [▶ 274]	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

6.1.2.2.3.1.3.3.22 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 *fPos* and the slat angle *fAngl*.
bRstManFnct	BOOL	A TRUE at this input deletes the telegram at the output (*stSunBld.bActv* = FALSE, *bActv* = FALSE, *stSunBld.fPos* = 0, *stSunBld.fAngl* = 0) and no new values are passed through. The output *bActv* 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 [▶ 269]	Priority of the output telegram.

Outputs

```
VAR_OUTPUT
  stSunBld   : ST_BA_SunBld;
  bActv      : BOOL;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] 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

6.1.2.2.3.1.3.3.23 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 [▶ 274] 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. *fPos* = 0, *fAngl* = 0, *bManUp* = FALSE, *bManDwn* = FALSE, *bManMod* = FALSE, *bActv* = FALSE. Since the blind function block FB_BA_SunBldActr [▶ 374] or the roller blind function block FB_BA_RolBldActr [▶ 365] 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 [▶ 274]	Positioning telegrams available for selection. *stSunBld_Prio1* has the highest priority and *stSunBld_Prio4* the lowest.

Outputs

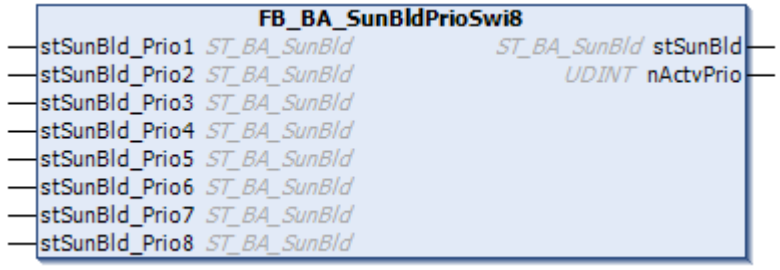
```
VAR_OUTPUT
  stSunBld     : ST_BA_SunBld;
  nActvPrio    : UDINT;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
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_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 [▶ 274] 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. *fPos* = 0, *fAngl* = 0, *bManUp* = FALSE, *bManDwn* = FALSE, *bManMod* = FALSE, *bActv* = FALSE. Since the blind function block FB_BA_SunBldActr [▶ 374] or the

roller blind function block FB_BA_RolBldActr [▶ 365] 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 [▶ 274]	Positioning telegrams available for selection. *stSunBld_Prio1* has the highest priority and *stSunBld_Prio8* the lowest.

Outputs

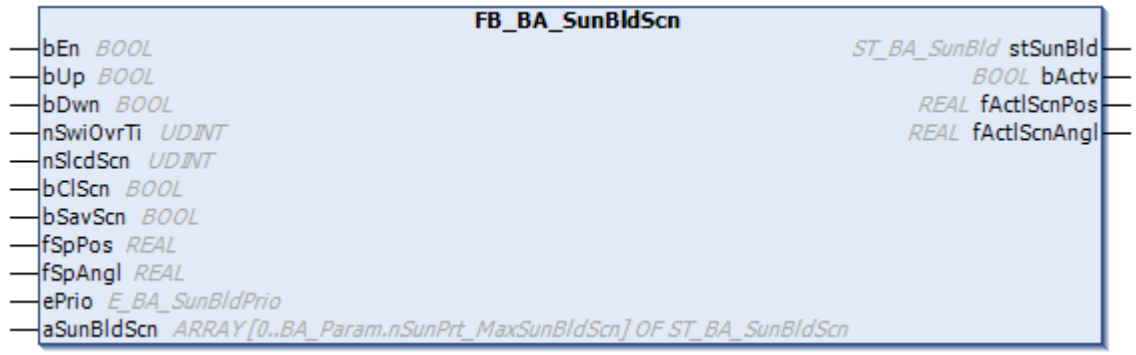
```
VAR_OUTPUT
  stSunBld     : ST_BA_SunBld;
  nActvPrio    : UDINT;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
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

6.1.2.2.3.1.3.3.25 FB_BA_SunBldScn



The function block FB_BA_SunBldScn represents an extension of the manual operation FB_BA_SunBldSw [▶ 388] by a scene memory and a call function. The blind actuator FB_BA_SunBldActr [▶ 374] or the roller blind actuator FB_BA_RolBldActr [▶ 365] 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 [▶ 374] or the roller blind function block FB_BA_RolBldActr [▶ 365] 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 [▶ 274]. If a command input is activated for longer than the entered time *nSwiOvrTi* [ms], the corresponding control command latches. Activating a command input again releases this latch.

A rising edge on *bSavScn* saves the current position and lamella angle in the scene selected in *nSlcdScn*. This procedure is possible at any time, even during active positioning.

A rising edge at *bClScn* calls the scene that is present at input *nSlcdScn*. This scene call can be interrupted by manual commands to *bUp* and *bDwn*.

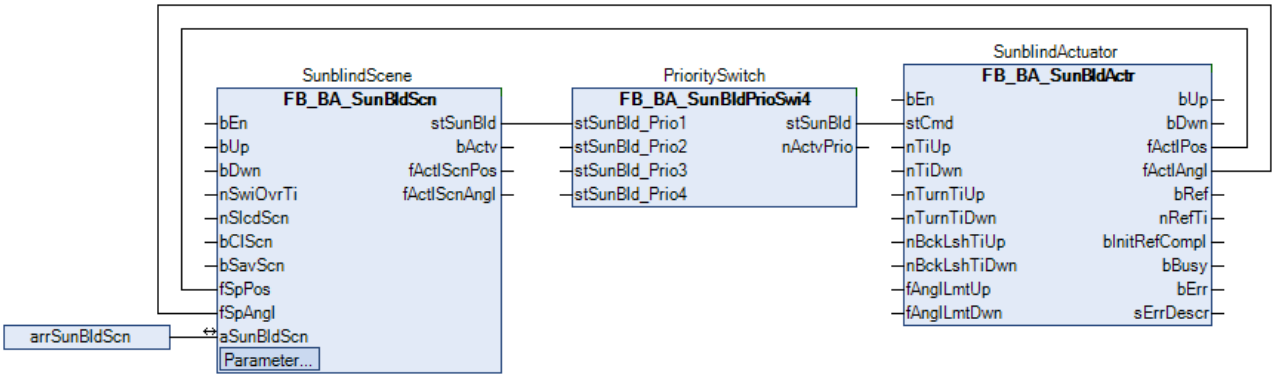
Manual control and scene positioning are each carried out with the priority that is preselected under *ePrio*.

If the function is no longer active due to the resetting of *bEn*, then the active flag in the positioning telegram stSunBld [▶ 274] is reset and the positions for height and angle are set to "0". The priority function block (e.g. *FB_BA_SunBldPrioSwi4*) enables a function with lower priority to take over the control by resetting.

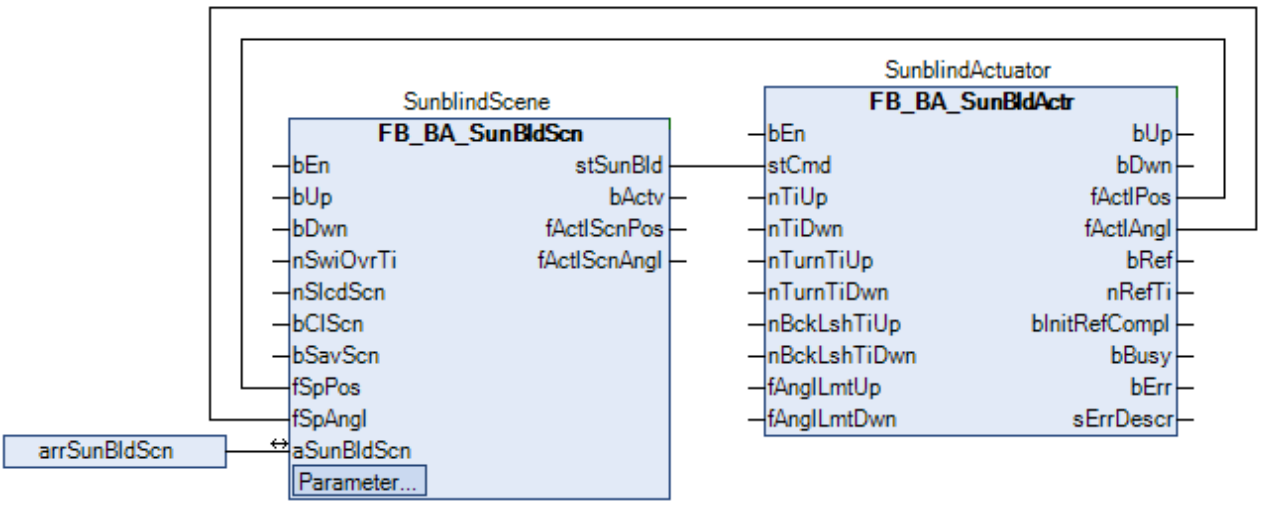
Linking to the blind function block

Like the "normal" manual mode function block FB_BA_SunBldSwi [▶ 388], 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 [▶ 274]. 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 ST_BA_Sunbld [▶ 274] - *bManMod* and *bActv* 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 *bActv* 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 [▶ 274] 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 (*bSavScn*) or called (*bClScn*). Internally limited to a minimum value from 0 to BA_Param.nSunPrt_MaxSunBldScn [▶ 284]
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 FB_BA_SunBldActr or FB_BA_RolBldActr 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 [▶ 269]	Priority of the active telegram.

/ Inputs/Outputs

```
VAR_IN_OUT
  aSunBldScn : ARRAY[0..BA_Param.nSunPrt_MaxSunBldScn] OF ST_BA_SunBldScn;
END_VAR
```

Name	Type	Description
aSunBldScn	ARRAY OF ST_BA_SunBldScn [▶ 275]	Table with the scene entries

Outputs

```
VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  bActv         : BOOL;
  fActlScnPos   : REAL;
  fActlScnAngl  : REAL;
END_VAR
```

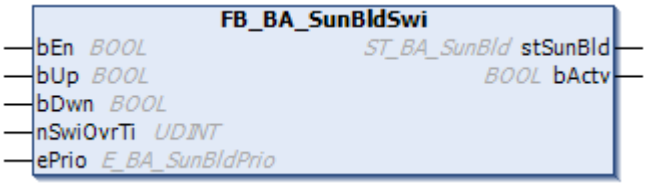
Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] and is solely used to indicate whether the function block sends an active telegram.
fActlScnPos	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) 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

6.1.2.2.3.1.3.3.26 FB_BA_SunBldSwi



With the help of the function block FB_BA_SunBldSwi the blind actuator FB_BA_SunBldActr [▶ 374] or the roller blind actuator FB_BA_RolBldActr [▶ 365] can be controlled in manual operation mode. The connection takes place via the positioning telegram ST_BA_Sunbld [▶ 274] either directly or with an additional priority control.

Operation

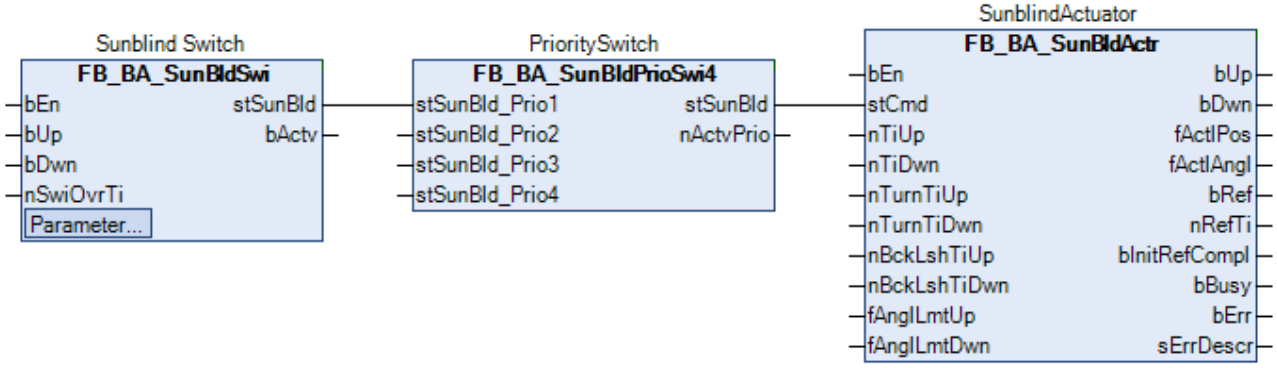
In manual mode, the function block controls the blind function block FB_BA_SunBldActr [▶ 374] or the roller shutter function block FB_BA_RolBldActr [▶ 365] via the command inputs *bUp* and *bDwn*; *bUp* has priority. The commands are passed on to the respective commands *bManUp* and *bManDwn* of the 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.

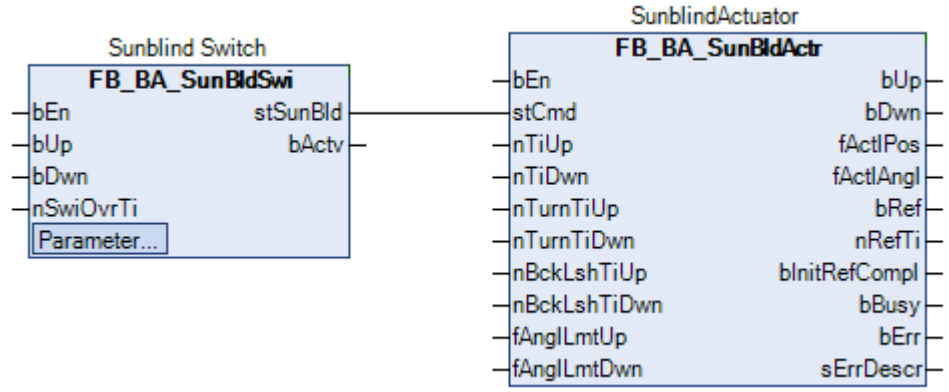
Linking to the blind function block

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 [▶ 274].

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 [▶ 274] - *bManMod* and *bActv* 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 *bActv* 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 [▶ 274] 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 [▶ 269]	Priority of the active telegram.

Outputs

```
VAR_OUTPUT
  stSunBld  : ST_BA_SunBld;
  bActv     : BOOL;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] 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

6.1.2.2.3.1.3.3.27 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 [▶ 274] 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 *nEvtIncSunBld* in the global variable list BA_Globals [▶ 283].

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 [▶ 374] or the roller blind function block FB_BA_RolBldActr [▶ 365] 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 [▶ 274]	Telegram inputs

Outputs

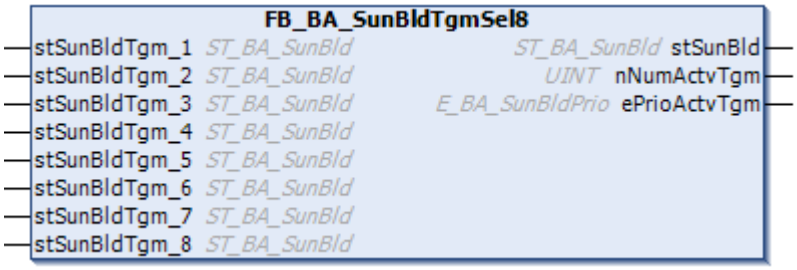
```
VAR_OUTPUT
  stSunBld       : ST_BA_SunBld;
  nNumActvTgm    : UINT;
  ePrioActvTgm   : E_BA_SunBldPrio;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
nNumActvTgm	UINT	Indicates which input is valid, e.g. if *stSunBldTgm_3* is passed, *nNumActvTgm* = 3. If *nNumActvTgm* = 0, no telegram is active.
ePrioActvTgm	E_BA_SunBldPrio [▶ 269]	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_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 [▶ 274] 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 telegram (last writer wins) is valid, determined by the counter variable *nEvtIncLight* in the global variable list BA_Globals [▶ 283].

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 [▶ 374] or the roller blind function block FB_BA_RolBldActr [▶ 365] 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 [▶ 274]	Telegram inputs

Outputs

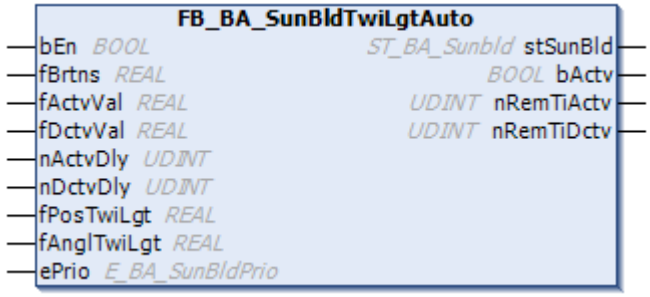
```
VAR_OUTPUT
  stSunBld       : ST_BA_SunBld;
  nNumActvTgm    : UINT;
  ePrioActvTgm   : E_BA_SunBldPrio;
END_VAR
```

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
nNumActvTgm	UINT	Indicates which input is valid, e.g. if *stSunBldTgm_3* is passed, *nNumActvTgm* = 3. If *nNumActvTgm* = 0, no telegram is active.
ePrioActvTgm	E_BA_SunBldPrio [▶ 269]	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

6.1.2.2.3.1.3.3.29 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 [▶ 274]. 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 [▶ 274] 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 ST_BA_SunBld [▶ 274] - *bManMod* and *bActv* 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 *bActv* itself.
fBrtns	REAL	Outdoor brightness [lx].
fActvVal	REAL	Activation limit value [lx]. The value *fActvVal* is internally limited to values from 0 to *fDctvVal*.
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	Lamella 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 [▶ 269]	Priority of the active telegram.

Outputs

```
VAR_OUTPUT
  stSunBld          : ST_BA_SunBld;
  bActv             : BOOL;
  nRemTiActv        : UDINT;
  nRemTiDctv        : UDINT;
END_VAR
```

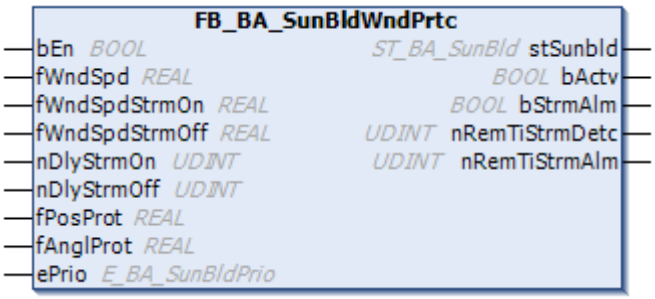
Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] and is solely used to indicate whether the function block sends an active telegram.
nRemTiActv	UDINT	Shows the time remaining [s] after falling below the switch value *fActvVal* 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 *fDctvVal* 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 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) 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

6.1.2.2.3.1.3.3.30 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 [▶ 332]) 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 [▶ 274] - *bManMod* and *bActv* 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 *fWndSpd*. A value greater than this limit value triggers the alarm after the entered time *nDlyStrmOn*.
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 *fWndSpd*. A value smaller than or equal to this limit value resets the alarm after the entered time *nDlyStrmOff*.
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 [▶ 269]	Priority of the active telegram.

Outputs

```
VAR_OUTPUT
  stSunBld             : ST_BA_SunBld;
  bActv                : BOOL;
  bStrmAlm             : BOOL;
  nRemTiStrmDetc       : UDINT;
  nRemTiStrmAlm        : UDINT;
END_VAR
```

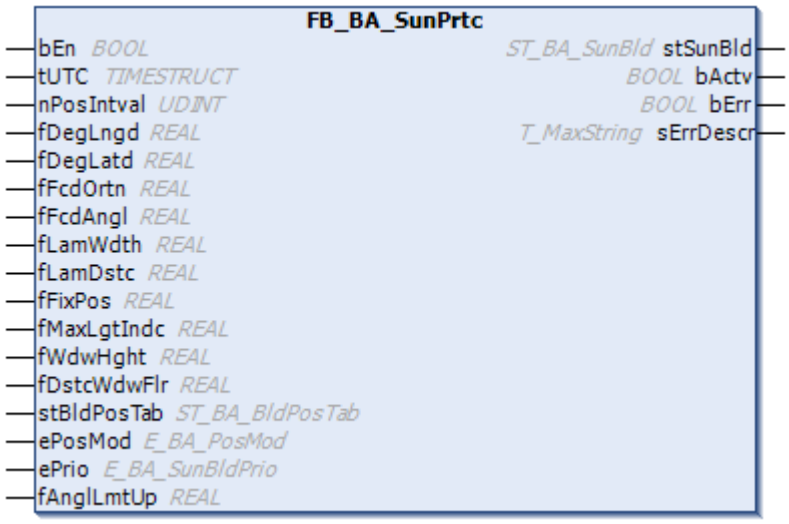
Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Output telegram, for the position and angle of the lamella.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] and is solely used to indicate whether the function block sends an active telegram.
bStrmAlm	BOOL	Indicates the storm alarm
nRemTiStrmDetc	UDINT	In an uncritical case this second counter constantly indicates the alarm delay time *nDlyStrmOn*. If the measured wind speed *fWndSpd* is above the activation limit value *fWndSpdStrmOn*, 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 *nDlyStrmOff*. If the measured wind speed *fWndSpd* falls below the deactivation limit value *fWndSpdStrmOff*, 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 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) 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

6.1.2.2.3.1.3.3.31 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 [▶ 273]). See also description of FB_BA_BldPosEntry [▶ 561].
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 (*bActv*) is reset in the positioning structure *stSunBld* of the type ST_BA_SunBld [▶ 274] 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 (*fPos*, *fAngl*) 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 FB_BA_GetTime [▶ 618] 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	β=0°
East	β=90°
South	β=180°
West	β=270°

The following applies for the southern hemisphere:

Viewing direction	Facade orientation
South	β=0°
East	β=90°
North	β=180°
West	β=270°

Name	Type	Description
fFcdAngl	REAL	Facade inclination [°] (see Facade inclination [▶ 336]).
fLamWdth	REAL	Width of the lamellas in mm (see sketch [▶ 333]).
fLamDstc	REAL	Lamella spacing in mm (see sketch [▶ 333]).
fFixPos	REAL	Fixed (constant) blind height [0...100%]. Applies if *ePosMod* = *ePosModFix* (see E_BA_PosMod [▶ 685]).
fMaxLgtIndc	REAL	Maximum desired light incidence in mm measured from the outside of the wall (see height adjustment [▶ 558]). The parameters *fWdwHght* and *fDstcWdwFlr* 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 *fMaxLgtIndc*. Applies if *ePosMod* = *ePosModeMaxIncidence* (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 [▶ 273]	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 E_BA_PosMod). For a more detailed description please refer to FB_BA_BldPosEntry [▶ 561].
ePosMod	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 [▶ 269]	Priority of the active telegram.
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 [▶ 274]	Output telegram, for the position and angle of the lamella.
bActv	BOOL	Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 274] 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 *fFcdAngl* is outside the valid range of -90°..90°.
05: Error: The value for the slat spacing (*fLamDstc*) is greater than or equal to the value for the slat width (*fLamWdth*). 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 *fLamWdth* is zero.
07: Error: The value entered for the slat spacing *fLamDstc* is zero.
08: Error: The value entered for the fixed blind height (*fFixPos*) is greater than 100 or less than 0. At the same time, positioning "fixed blind height" is selected - *ePosMod* = ePosModFix.
09: Error: The "Values valid" bit (*bVld*) in the *stBldPosTab* positioning table is not set - invalid values, see FB_BA_BldPosEntry. At the same time, "Table" positioning is selected – *ePosMod* = ePosModTab.
10: Error: The value entered for the maximum required light incidence *fMaxLgtIndc* is less than or equal to zero. At the same time, "maximum light incidence" is selected – *ePosMod* = ePosModMaxIndc.
11: Error: The value entered for the window height *fWdwHght* is less than or equal to zero. At the same time, "maximum light incidence" is selected – *ePosMod* = ePosModMaxIndc.
12: Error: The distance between lower window edge and floor *fDstcWdwFlr* that was entered is less than zero. At the same time, "maximum light incidence" is selected – *ePosMod* = ePosModMaxIndc.
13: Error: An invalid positioning mode is entered at input *ePosMod*.

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) 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 minute (0...59).
nSecond	WORD	The second (0...59).
nMilliseconds	WORD	The millisecond (0...999).

Outputs

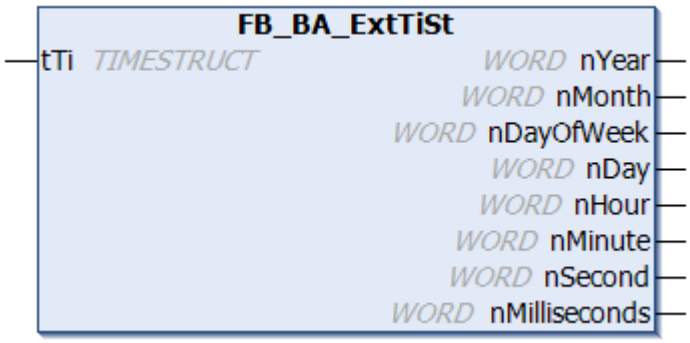
```
VAR_OUTPUT
  bNewData  : BOOL;
  tTi       : TIMESTRUCT;
END_VAR
```

Name	Type	Description
bNewData	BOOL	The output is TRUE in the cycle in which the input variables have changed.
tTi	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

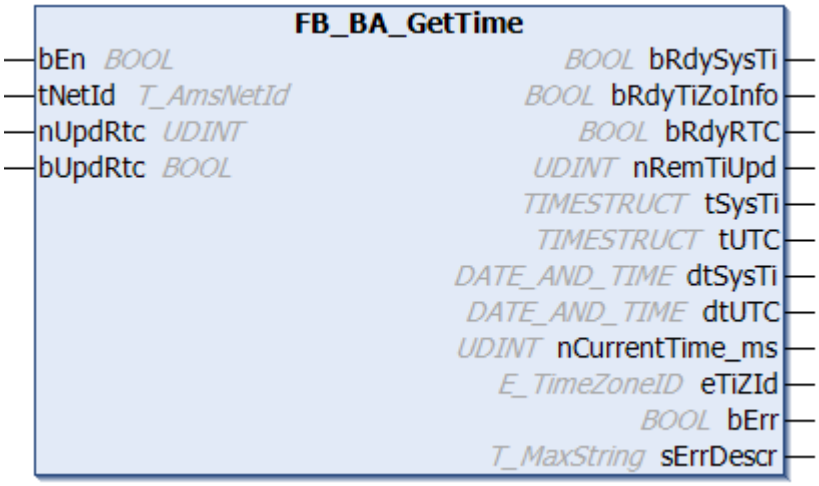
```
VAR_OUTPUT
  nYear           : WORD;
  nMonth          : WORD;
  nDayOfWeek      : WORD;
  nDay            : WORD;
  nHour           : WORD;
  nMinute         : WORD;
  nSecond         : WORD;
  nMilliseconds   : WORD;
END_VAR
```

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 minute (0...59).
nSecond	WORD	The second (0...59).
nMilliseconds	WORD	The millisecond (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 *bEn* = 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. For the local computer an empty string may be specified.
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 *nUpdRtc*, 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.
tSysTi	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 (*bEn* = FALSE), the output *stSysTi* 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 (*bEn*=FALSE), the output *tUTC* and its subelements (day month, etc.) show 0.
dtSysTi / dtUTC	DT	Same as *stSysTi*/ *stUTC*, but in DATE-AND-TIME format: year-month-day-hour-minute-seconds. **Info**: If the function block is not enabled (*bEn* = FALSE), the outputs *dtSysTi* and *dtUTC* 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 *dtSysTi* / *dtUTC*.
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.

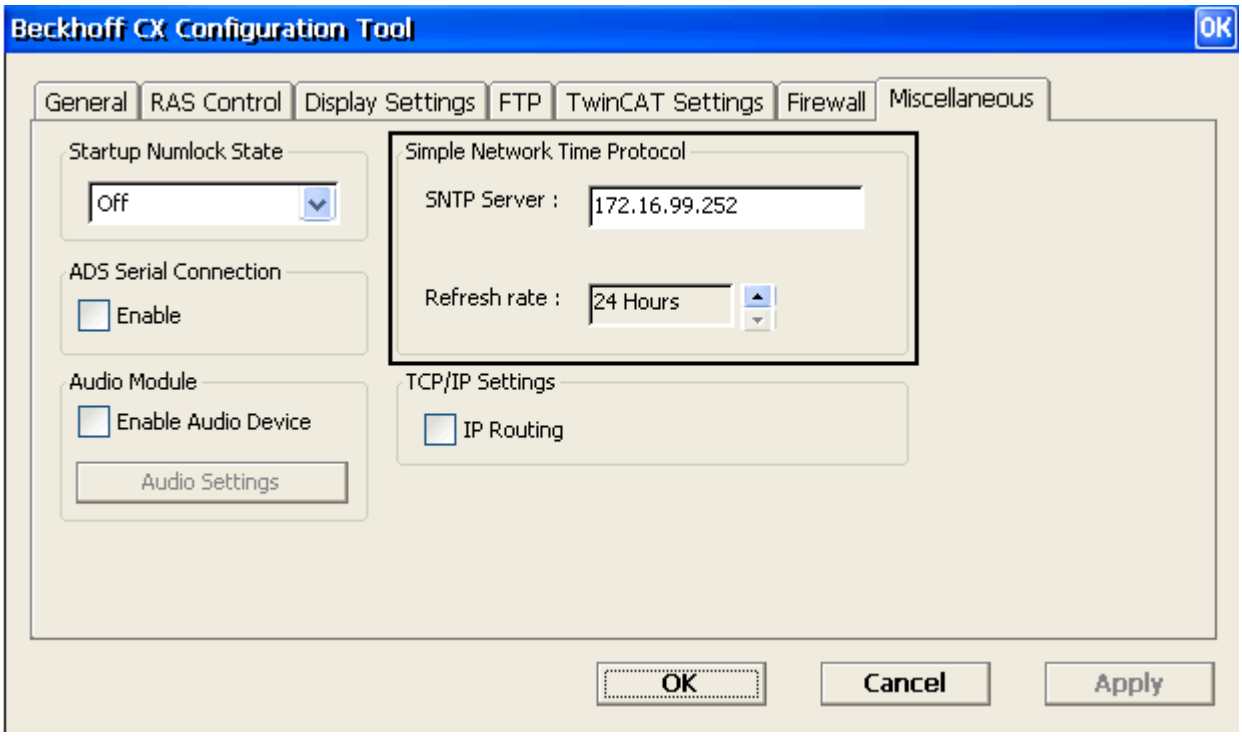
Requirements

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 local NT system time and the date of a TwinCAT system can be set with the function block FB_BA_SetTime (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. More information can be found in 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 *sNetId* := ''; 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 [▶ 400], 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 *bSet*, 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 created time structure *stSysTi* is internally checked for limits (see TIMESTRUCT).

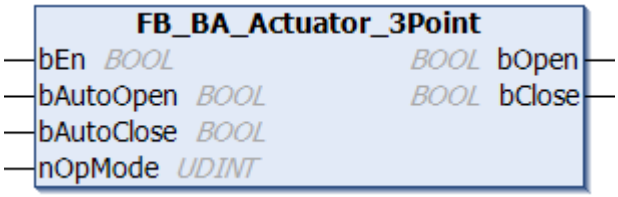
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5 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

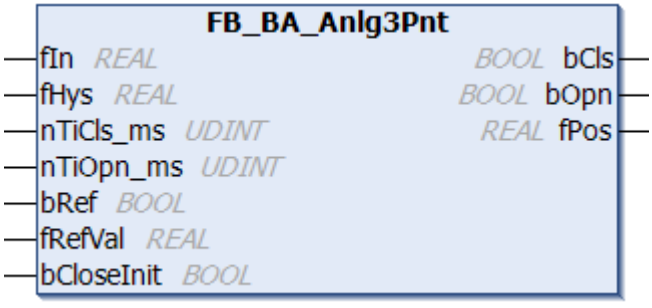
```
VAR_OUTPUT
  bOpen    : BOOL;
  bClose   : BOOL;
END_VAR
```

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 *fRefVal* [0…100 %].
fRefVal	REAL	Value for referencing the actuator with *bRef* [0…100 %]. Internally limited to values between 0 and 100.
bCloseInit	BOOL	If this input is TRUE, output *bCls* is TRUE for the time udiTiOpn_ms.

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	*bEn* is the general enable of the function block. If *bEn* is FALSE, the message output *bQ* 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 *E_BA_AntBlkgMode.eOffTime*.
bExtReq	BOOL	Active in the E_BA_AntBlkgMode.eExternalRequest 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 *E_BA_AntBlkgMode.eExternalRequest* or *E_BA_AntBlkgMode.eOffTime*. 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 *bLock* is FALSE again. If *bLock* becomes TRUE during the output of a blocking protection pulse, then the blocking protection pulse is interrupted. After *bLock* 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 *bQ*.

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

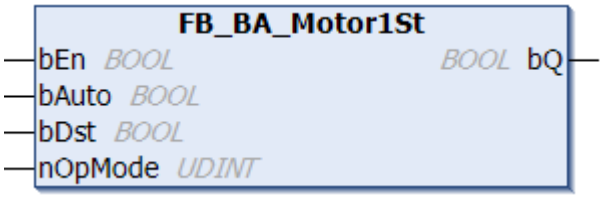
```
VAR_OUTPUT
  bQ                : BOOL;
  nRemOffMin        : UDINT;
  nRemImplLngt      : UDINT;
END_VAR
```

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 *bQ*.

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 (*nOpMode* = 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

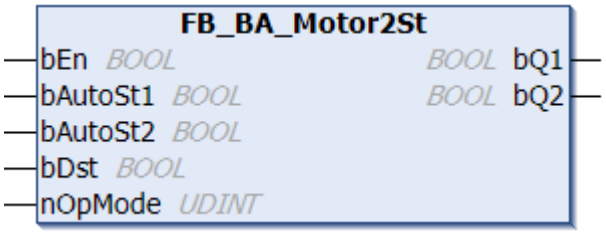
```
VAR_OUTPUT
  bQ  : BOOL;
END_VAR
```

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 (*nOpMode* = 0).
bAutoSt2	BOOL	Request of the actuator to step 2 in automatic mode (*nOpMode* = 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

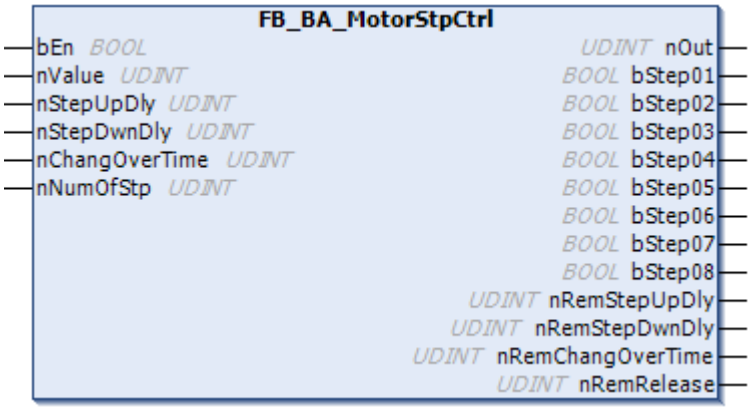
```
VAR_OUTPUT
  bQ1   : BOOL;
  bQ2   : BOOL;
END_VAR
```

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

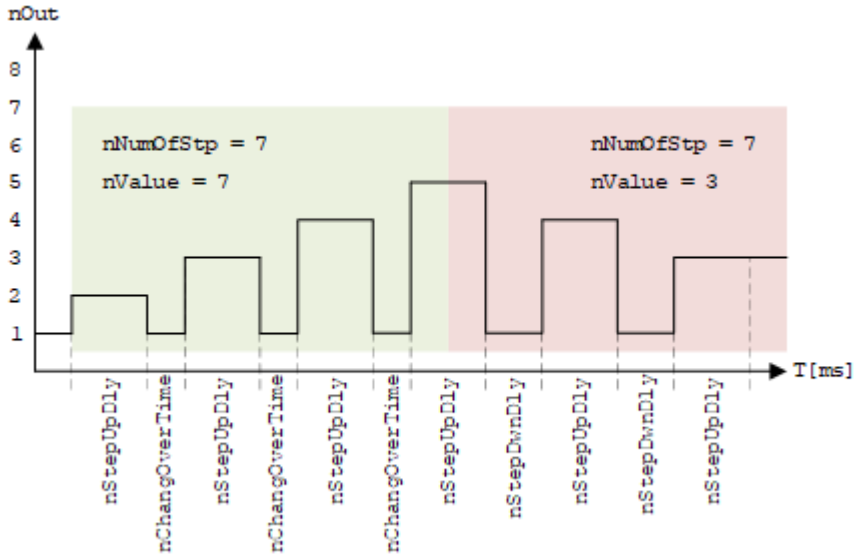
6.1.2.2.3.1.5.1.6 FB_BA_MotorStpCtrl



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 *bEn* 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].

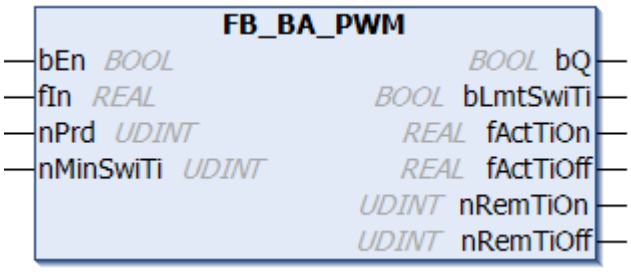
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [▶ 413].
NumberOfSteps	UDINT	Get	Outputs the current or corrected value of the number of steps. In the error case of nNumOfStp there is a detailed description of the error at error handling [▶ 413].
Step	STRING	Get	Output of the current step depending on *nOut*.

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 *rIn* (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> *fIn* >100!

If *fIn* = 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 *fIn* is set to 0, an immediate jump occurs to the Off step (*bQ*=FALSE), or to the On step if *fIn*=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* / n*RemTiOff* 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 *nPrd*.

Outputs

```
VAR_OUTPUT
  bQ              : BOOL;
  bLmtSwiTi       : BOOL;
  fActTiOn        : REAL;
  fActTiOff       : REAL;
  nRemTiOn        : UDINT;
  nRemTiOff       : UDINT;
END_VAR
```

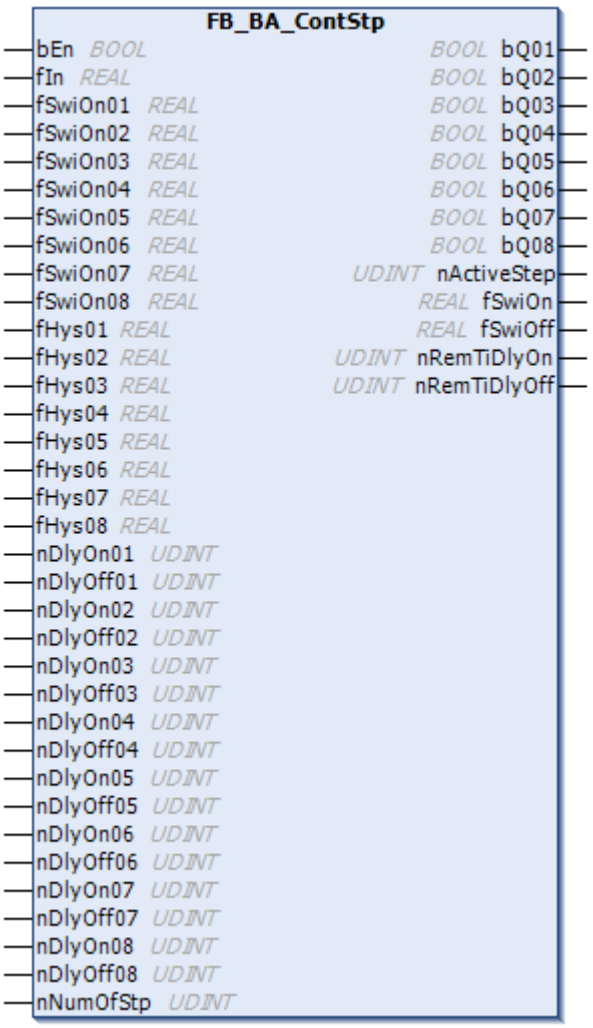
Name	Type	Description
bQ	BOOL	PWM output.
bLmtSwiTi	BOOL	Information output to indicate that the input signal is so low that the minimum switch-on time is used as limit.
fActTiOn	REAL	Information output: calculated switch-on time [s].
fActTiOff	REAL	Information output: calculated switch-off time [s].
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

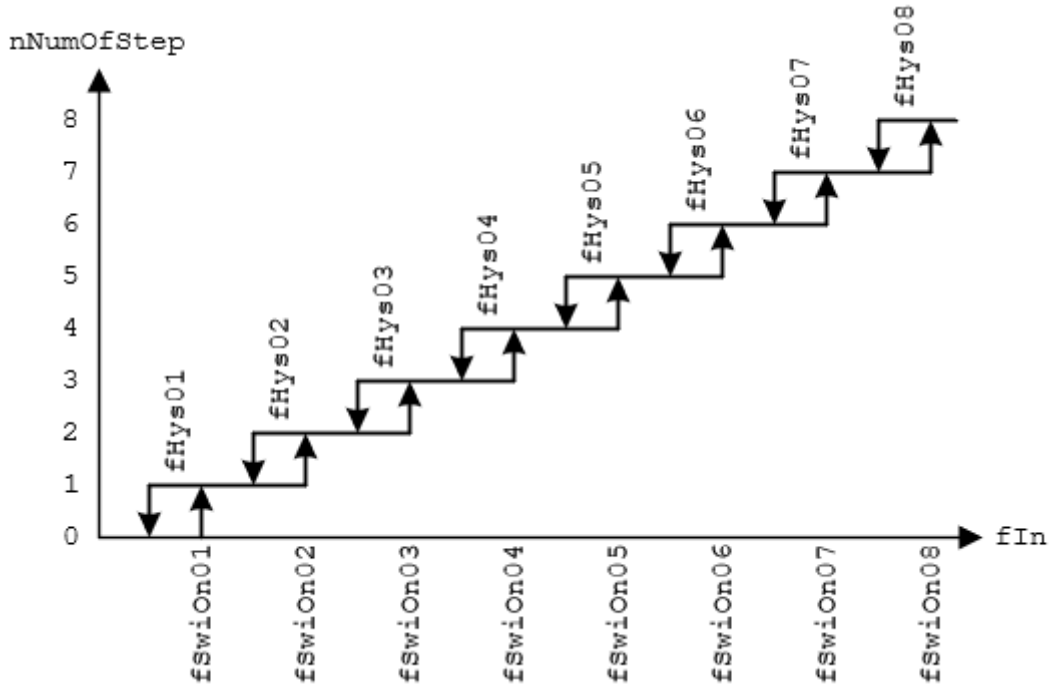
6.1.2.2.3.1.5.2 Control

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*.

nAc-tiveSt ep	nNu-mOf-Step	fSwiO n	fSwiO ff	nRem TiD-lyOn	nRem TiDly-Off	bQ01	bQ02	bQ03	bQ04	bQ05	bQ06	bQ07	bQ08
0	0	fSwiO n01	fSwiO n01 - fHys01	nDlyO n01		FALS E	FALS E	FALS E	FALS E	FALS E	FALS E	FALS E	FALS E
1	> = 1	fSwiO n02	fSwiO n01 - fHys01	nDlyO n02	nDlyOf f01	**TRUE**	FALS E	FALS E	FALS E	FALS E	FALS E	FALS E	FALS E
2	> = 2	fSwiO n03	fSwiO n02 - fHys02	nDlyO n03	nDlyOf f02	**TRUE**	**TRUE**	FALS E	FALS E	FALS E	FALS E	FALS E	FALS E
3	> = 3	fSwiO n04	fSwiO n03 - fHys03	nDlyO n04	nDlyOf f03	**TRUE**	**TRUE**	**TRUE**	FALS E	FALS E	FALS E	FALS E	FALS E
4	> = 4	fSwiO n05	fSwiO n04 - fHys04	nDlyO n05	nDlyOf f04	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALS E	FALS E	FALS E	FALS E
5	> = 5	fSwiO n06	fSwiO n05 - fHys05	nDlyO n06	nDlyOf f05	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALS E	FALS E	FALS E
6	> = 6	fSwiO n07	fSwiO n06 - fHys06	nDlyO n07	nDlyOf f06	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALS E	FALS E
7	> = 7	fSwiO n08	fSwiO n07 - fHys07	nDlyO n08	nDlyOf f07	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALS E
8	8	fSwiO n08	fSwiO n08 - fHys08		nDlyOf f08	**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 *bEn* is FALSE, all message outputs *bQ0N* 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

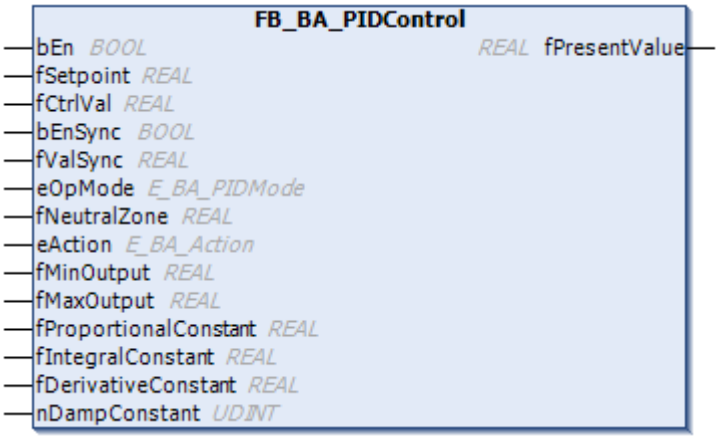
```
VAR_OUTPUT
  bQ01           : BOOL;
  bQ02           : BOOL;
  bQ03           : BOOL;
  bQ04           : BOOL;
  bQ05           : BOOL;
  bQ06           : BOOL;
  bQ07           : BOOL;
  bQ08           : BOOL;
  nActiveStep    : UDINT;
  fSwiOn         : REAL;
  fSwiOff        : REAL;
  nRemTiDlyOn    : 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_PIDControl



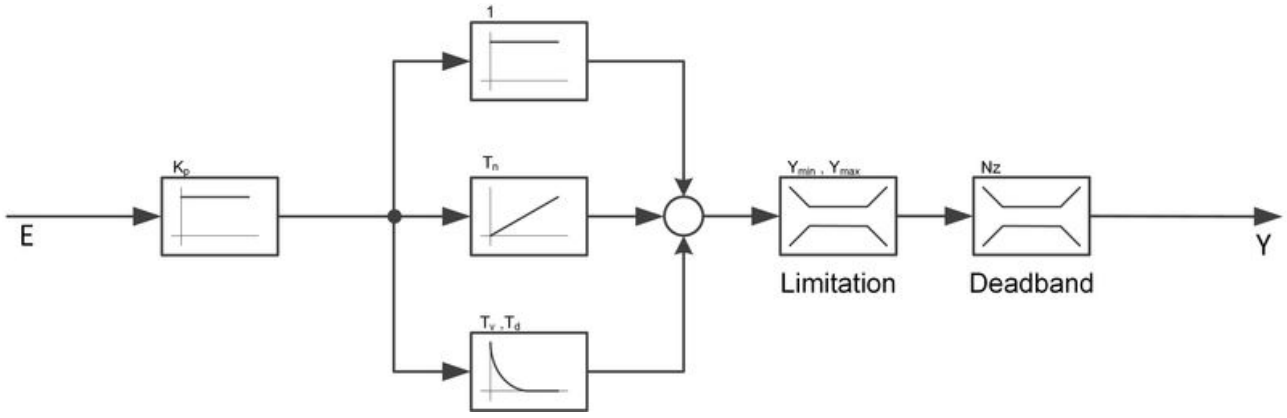
The function block FB_BA_PIDCtrl is a universal PID controller.

The controller is divided internally into two consecutive parts:

- the controller itself, illustrated in the functional diagrams below as P, I and D part with an output limitation.
- a deadband element (neutral zone) that applies a hysteresis to the output changes of the controller.

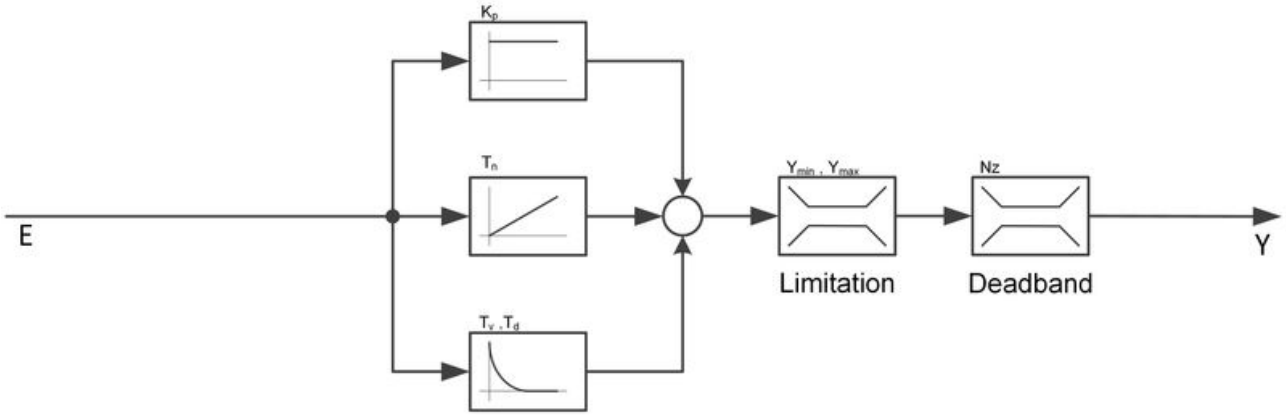
Operation mode "Upstream P part":

(*eOpMode* = E_BA_PIDMode.eP1ID)



"Parallel structure" operation mode:

(*eOpMode* = E_BA_PIDMode.ePID)



Control direction

With *eAction* = E_BA_Action.eReverse, the control direction of the controller is reversed so that a control deviation of less than 0 causes a positive change in the control value. This is achieved by a negative calculation of the control deviation:

eAction	*fE* (control deviation)	Control direction
E_BA_Action.eDirect	*fCtrlVal-fSetpoint* (actual value-setpoint)	direct (cooling)
E_BA_Action.eReverse	*fSetpoint-fCtrlVal* (setpoint - actual value)	indirect (heating)

Passive behavior (bEn = FALSE)

The outputs are set as follows:

fPresentValue	0.0

The internal values for the P, I, and D parts are set to 0, also the values for the I and D parts of the preceding cycle. In case of a restart the control value is thus calculated in the first cycle without past values.

Active behavior (bEn = TRUE)

In the first cycle, the I and D parts are calculated without historical values, as already mentioned.

Anti-Reset-Windup

If the I part is active, the controller ensures that it is retained, if the controller output *rY* is about to move beyond the limits *fMinOutput* oder *fMaxOutput*. A preliminary calculation of the controller output takes place inside the controller in every cycle.

Anti-reset windup at min limit

If the precalculation is smaller than the lower output limit *fMinOutput*, the I part is prevented from falling further and is limited to the value of the last PLC cycle. However, an increase in the I part remains possible.

Anti-reset windup at max limit

On the other hand, if the precalculation is greater than the upper limit *fMaxOutput*, the I part is prevented from increasing further and is also limited to the value of the last PLC cycle. In this case, a drop in the I part remains possible.

Synchronizations

There are several cases where controller output must not only be limited, but also synchronized to a new value by manipulating the I part (if not active, then the D part or the P part). These cases are prioritized because of the potential for simultaneity:

Prio	Description	Conditions	Comments
1	Synchronization via *bEnSync*/*fValSync*	Controller enabled - *bEn* = TRUE	A positive signal on *bEnSync* sets the I part so that the control value assumes the value *fValSync*. If *bEn* and *bEnSync* are set at the same time, this method can be used to set an initial value from which the controller "sets off". If the I part is not active, the D part is set accordingly. Note that only the rising edge of *bEnSync* is evaluated internally as this is a Set action. A TRUE signal must be applied again to the input *bEnSync* for renewed synchronization, for instance with a transfer value.
2	Range synchronization *fMinOutput*	Controller enabled – *bEn* = TRUE *fMinOutput* <> fMinOutput_1 (lastCycle) fY_Test < *fMinOutput*	If the lower range limit has changed and the precalculated controller output is now smaller than *fMinOutput* then synchronization is performed to *fMinOutput*.
3	Range synchronization *fMaxOutput*	Controller enabled – *bEn* = TRUE *fMaxOutput* <> fMaxOutput_1 (lastCycle) fY_Test > *fMaxOutput*	If the upper range limit has changed and the precalculated controller output is now greater than *fMaxOutput*, then synchronization is performed to *fMaxOutput*
4	Synchronization with reversal of the control direction	Controller enabled – *bEn* = TRUE *eAction* <> eAction_1 (last cycle)	It is synchronized so that the output holds the value BEFORE the reversal: *fPresentValue* = fPresentValue_1 (last cycle)
5	Anti-Reset-Windup	Controller enabled – *bEn* = TRUE	see Anti-Reset-Windup

Neutral zone

A value of *fNeutralZone* > 0.0 enables the function of the neutral zone (deadband). A value equal to zero deactivates the deadband element and the values at the input are passed directly through.

If, for the active controller, the change at the input of the element in a PLC cycle is smaller than *fNeutralZone* / 2 in comparison with the previous PLC cycle, then the output is held at the value of the previous cycle until the change is larger than or equal to *fNeutralZone* / 2.

This function is intended to avoid an unnecessarily large number of actuating pulses.

Syntax

```
VAR_INPUT
  bEn                          : BOOL;
  fSetpoint                    : REAL;
  fCtrlVal                     : REAL;
  bEnSync                      : BOOL;
  fValSync                     : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory':='Config'}
  eOpMode                      : E_BA_PIDMode := E_BA_PIDMode.ePID;
  {attribute 'parameterCategory':='Deadband'}
  fNeutralZone                 : REAL := 0.0;
  {attribute 'parameterCategory':='Config'}
  eAction                      : E_BA_Action := E_BA_Action.eReverse;
```

```
  {attribute 'parameterCategory':='Config'}
  fMinOutput                    : REAL := 0;
  {attribute 'parameterCategory':='Config'}
  fMaxOutput                    : REAL := 100;
  {attribute 'parameterCategory':='Config'}
  fProportionalConstant         : REAL := 3;
  {attribute 'parameterCategory':='Config'}
  {attribute 'parameterUnit':= 's,ms'}
  fIntegralConstant             : REAL;
  {attribute 'parameterCategory':='Config'}
  {attribute 'parameterUnit':= 's,ms'}
  fDerivativeConstant           : REAL;
  {attribute 'parameterCategory':='Config'}
  {attribute 'parameterUnit':= 's'}
  nDampConstant                 : UDINT;
END_VAR
VAR_OUTPUT
  fPresentValue                 : REAL;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enabling the controller
fSetpoint	REAL	Setpoint of the controlled system
fCtrlVal	REAL	Actual value of the controlled system
bEnSync	BOOL	Synchronizes the controller to the value of *fValSync*.
fValSync	REAL	Synchronization value. The value *fValSync* is internally limited to values from *fMinOutput* to *fMaxOutput*.

Inputs CONSTANT PERSISTENT

Name	Type	Description
eOpMode	E_BA_PIDMode	Mode of operation of the controller: PID mode or P-ID mode.
fNeutralZone	REAL	Dead zone
eAction	E_BA_Action	Control direction of the controller
fMinOutput	TIME	Lower controller output limit [%]. The value *fMinOutput* is limited at the top by *fMaxOutput*.
fMaxOutput	TIME	Upper controller output limit [%].
fProportionalConstant	REAL	Controller gain. Only affects the P part. Internally limited to a minimum value of 0.
fIntegralConstant	REAL	Integral action time of the I part [s,ms]. A null value at this parameter disables the I part.
fDerivativeConstant	REAL	Rate time of the D part [s,ms]. A null value at this parameter disables the D part.
nDampConstant	UDINT	Damping time of the D part [s].

Outputs

Name	Type	Description
fPresentValue	REAL	Control value. Range limited by *fYMin* and *fYMax*.

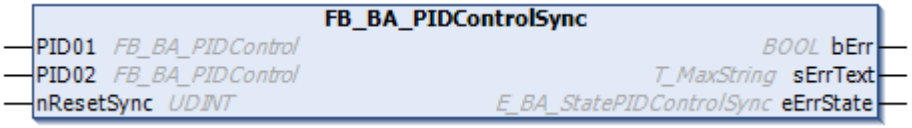
Properties

Name	Type	Access	Description
AntiResetWindup	BOOL	Get	The controller is in anti-reset windup status.
ControlDeviation	REAL	Get	Control deviation
IsMaxLimit	BOOL	Get	The controller output *fPresentValue* is at the upper output limit *fMaxOutput*.
IsMinLimit	BOOL	Get	The controller output *fPresentValue* is at the lower output limit *fMinOutput*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.56	Tc3_BA2 from v5.4.2.0

6.1.2.2.3.1.5.2.3 FB_BA_PIDControlSync



The function block FB_BA_PIDControlSync is used for parameter synchronization for 2 controllers of type FB_BA_PIDControl [▶ 420]. 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_PIDControl [▶ 420] are synchronized:

- eOpMode
- eActionRm
- fProportionalConstant
- fIntegralConstant
- fDerivativeConstant
- fMaxOutputRm
- fMinOutputRm
- nDampConstant
- fNeutralZone

Error detection

The error messages listed below are detected by the FB_BA_PIDControlSync.

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
'Synchronisation *eAction* fehlerhaft'	'Synchronization *eAction* faulty'
'Synchronisation *nDampConstant* fehlerhaft'	'Synchronization *nDampConstant* faulty'
'Synchronisation *fDerivativeConstant* fehlerhaft'	'Synchronization *fDerivativeConstant* faulty'
'Synchronisation *fIntegralConstant* fehlerhaft'	'Synchronization *fIntegralConstant* faulty'
'Synchronisation *fMaxOutput* fehlerhaft'	'Synchronization *fMaxOutput* faulty'
'Synchronisation *fMinOutput* fehlerhaft'	'Synchronization *fMinOutput* faulty'
'Synchronisation *fNeutralZone* fehlerhaft'	'Synchronization *fNeutralZone* faulty'
'Synchronisation *eOpMode* fehlerhaft'	'Synchronization *eOpMode* faulty'
'Synchronisation *fProportionalConstant* fehlerhaft'	'Synchronization *fProportionalConstant* faulty'

Syntax

```
FUNCTION_BLOCK FB_BA_LoopSync
VAR_OUTPUT
  bErr          : BOOL;
  sErrText      : T_MaxString;
  eErrState     : E_BA_StatePIDControlSync;
END_VAR
VAR_IN_OUT
  PID01         : FB_BA_PIDControl;
  PID02         : FB_BA_PIDControl;
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 text form [▶ 425].
eErrState	E_BA_StatePIDControlSync [▶ 271]	The enumeration shows the state of the synchronization.

Inputs Outputs

Name	Type	Description
Loop1	FB_BA_PIDControl [▶ 420]	Reference to controller no. 1 of the parameter adjustment.
Loop2	FB_BA_PIDControl [▶ 420]	Reference to controller no. 2 of the parameter adjustment.

Properties

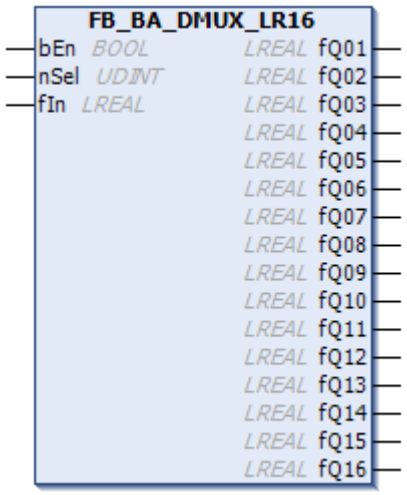
Name	Type	Access	Description
ErrText	T_MaxString	Get	The property *ErrText* displays the error texts from *sErrText*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.56	Tc3_BA2 from v5.4.2.0

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 *fQ01...fQ16*, which is to assume the value of input *fIn*.
fIn	LREAL	Value to be output.

Outputs

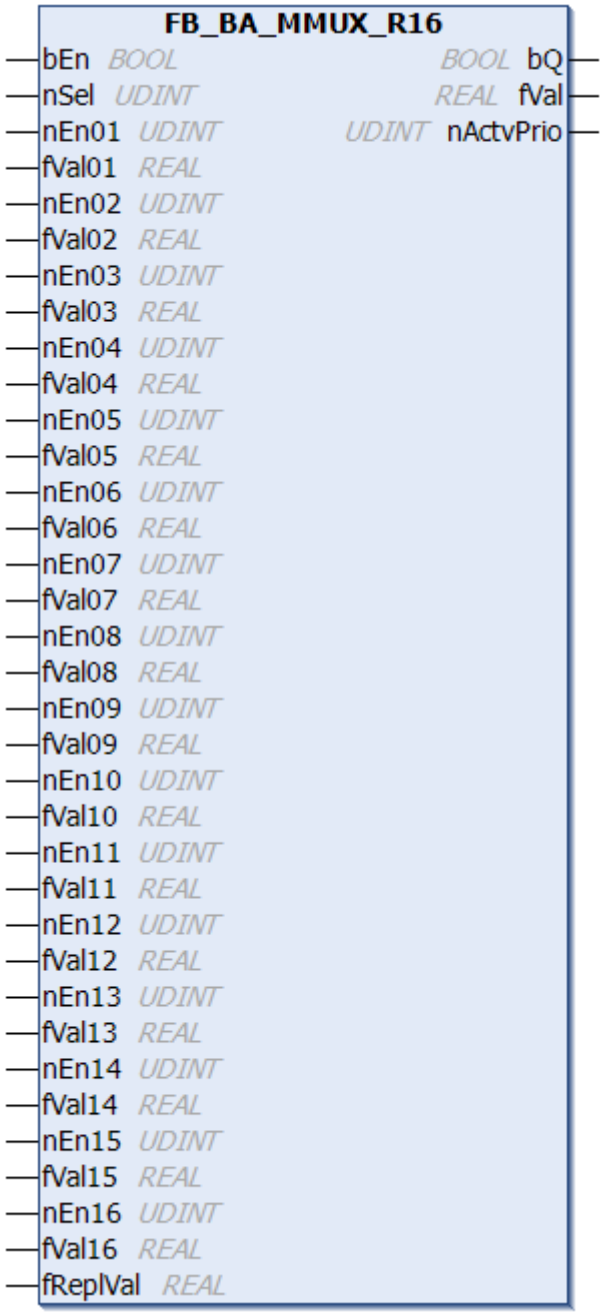
```
VAR_OUTPUT
  fQ01  : LREAL;
  fQ02  : LREAL;
  fQ03  : LREAL;
  fQ04  : LREAL;
  fQ05  : LREAL;
  fQ06  : LREAL;
  fQ07  : LREAL;
  fQ08  : LREAL;
  fQ09  : LREAL;
  fQ10  : LREAL;
  fQ11  : LREAL;
  fQ12  : LREAL;
  fQ13  : LREAL;
  fQ14  : LREAL;
  fQ15  : LREAL;
  fQ16  : LREAL;
END_VAR
```

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.

Sample:

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 [▶ 283] 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

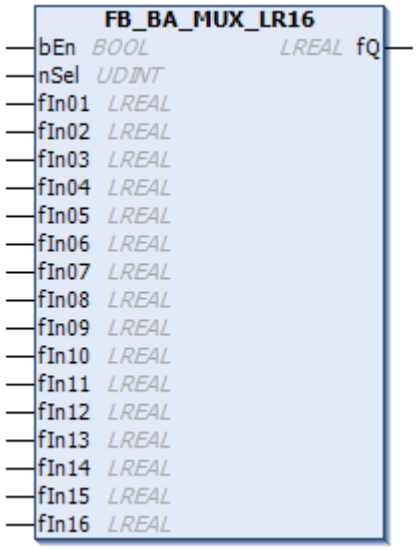
```
VAR_OUTPUT
  bQ          : BOOL;
  fVal        : REAL;
  nActvPrio   : UDINT;
END_VAR
```

Name	Type	Description
bQ	BOOL	TRUE if the selector *nSel* matches an input selector condition *nEnxx*.
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 *nActvPrio*.

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 *fQ*.
f01...f16	LREAL	Input values to select from.

Outputs

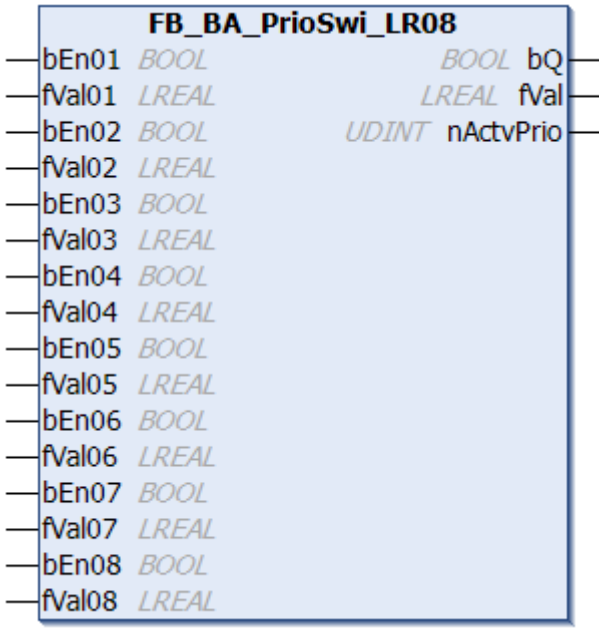
```
VAR_OUTPUT
  fQ   : LREAL;
END_VAR
```

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 [▶ 124] 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
```

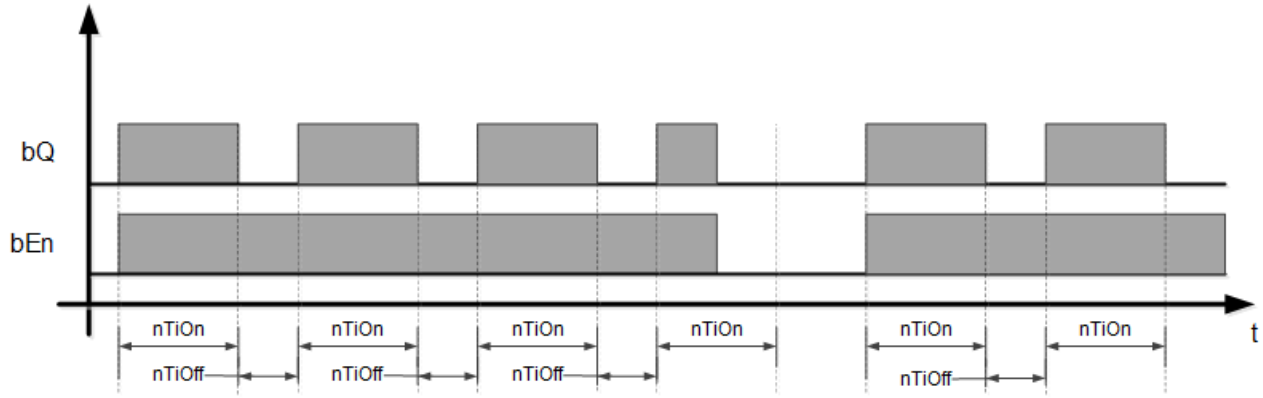
Name	Type	Description
bQ	BOOL	Output to indicate whether a priority is enabled.
fVal	LREAL	Output of the value of the current (highest) priority that is enabled.
nActvPrio	UDINT	Current (highest) priority that is enabled.

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 *bQ* [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 [▶ 437] 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

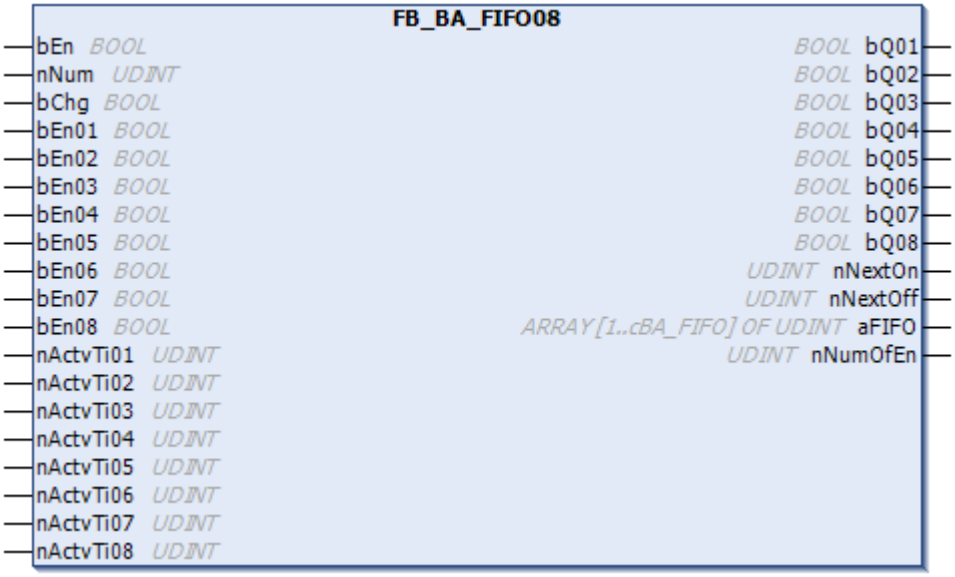
```
VAR_OUTPUT
  bQ01          : BOOL;
  bQ02          : BOOL;
  bQ03          : BOOL;
  bQ04          : BOOL;
  nNextOn       : UDINT;
  nNextOff      : UDINT;
  aFIFO         : ARRAY [1..4] OF UDINT;
  nNumOfEn      : UDINT;
END_VAR
```

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	ARRAY OF 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 [▶ 436] 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;
EN_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

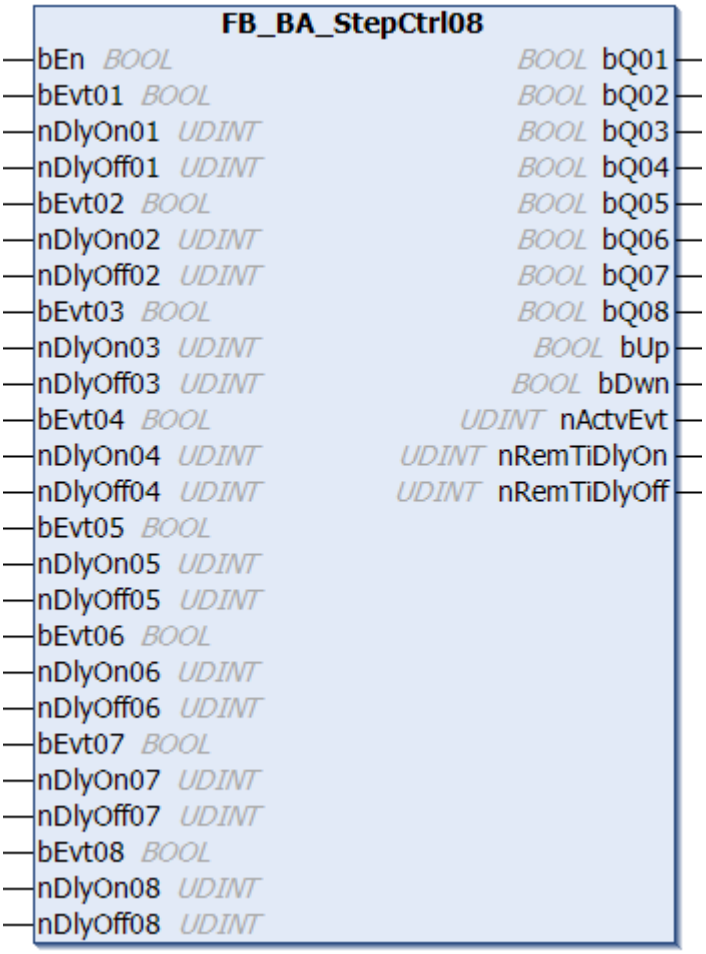
```
VAR_OUTPUT
  bQ01          : BOOL;
  bQ02          : BOOL;
  bQ03          : BOOL;
  bQ04          : BOOL;
  bQ05          : BOOL;
  bQ06          : BOOL;
  bQ07          : BOOL;
  bQ08          : BOOL;
  nNextOn       : UDINT;
  nNextOff      : UDINT;
  aFIFO         : ARRAY [1..8] OF UDINT;
  nNumOfEn      : 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	ARRAY OF 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

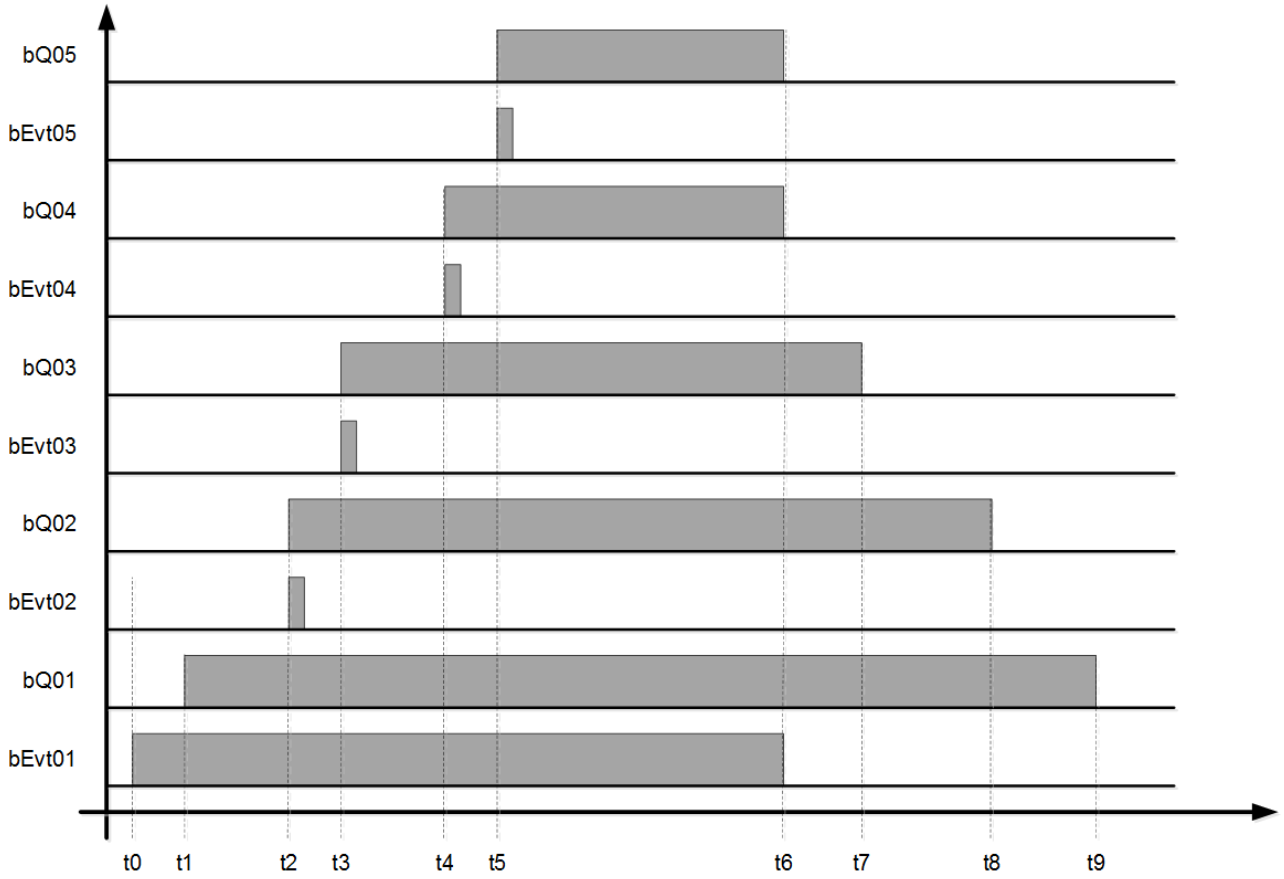


The function block FB_BA_StepCtrl08 is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs from *bQ01* to *bQ08* are set to FALSE. The 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 *bQ01…08* [s]
nDlyOff01…08	UDINT	Switch-off delay for output *bQ01…08* [s]

Outputs

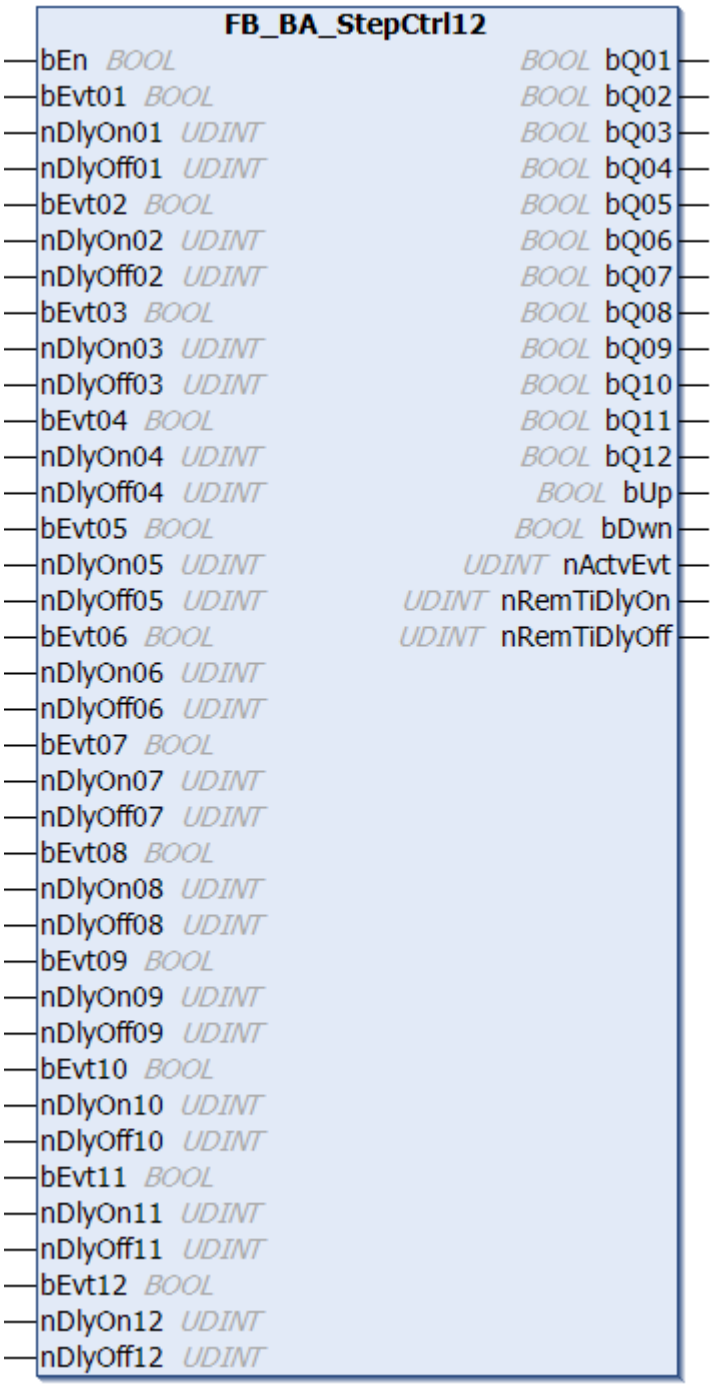
```
VAR_OUTPUT
  bQ01                : BOOL;
  bQ02                : BOOL;
  bQ03                : BOOL;
  bQ04                : BOOL;
  bQ05                : BOOL;
  bQ06                : BOOL;
  bQ07                : BOOL;
  bQ08                : BOOL;
  bUp                 : BOOL;
  bDwn                : 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

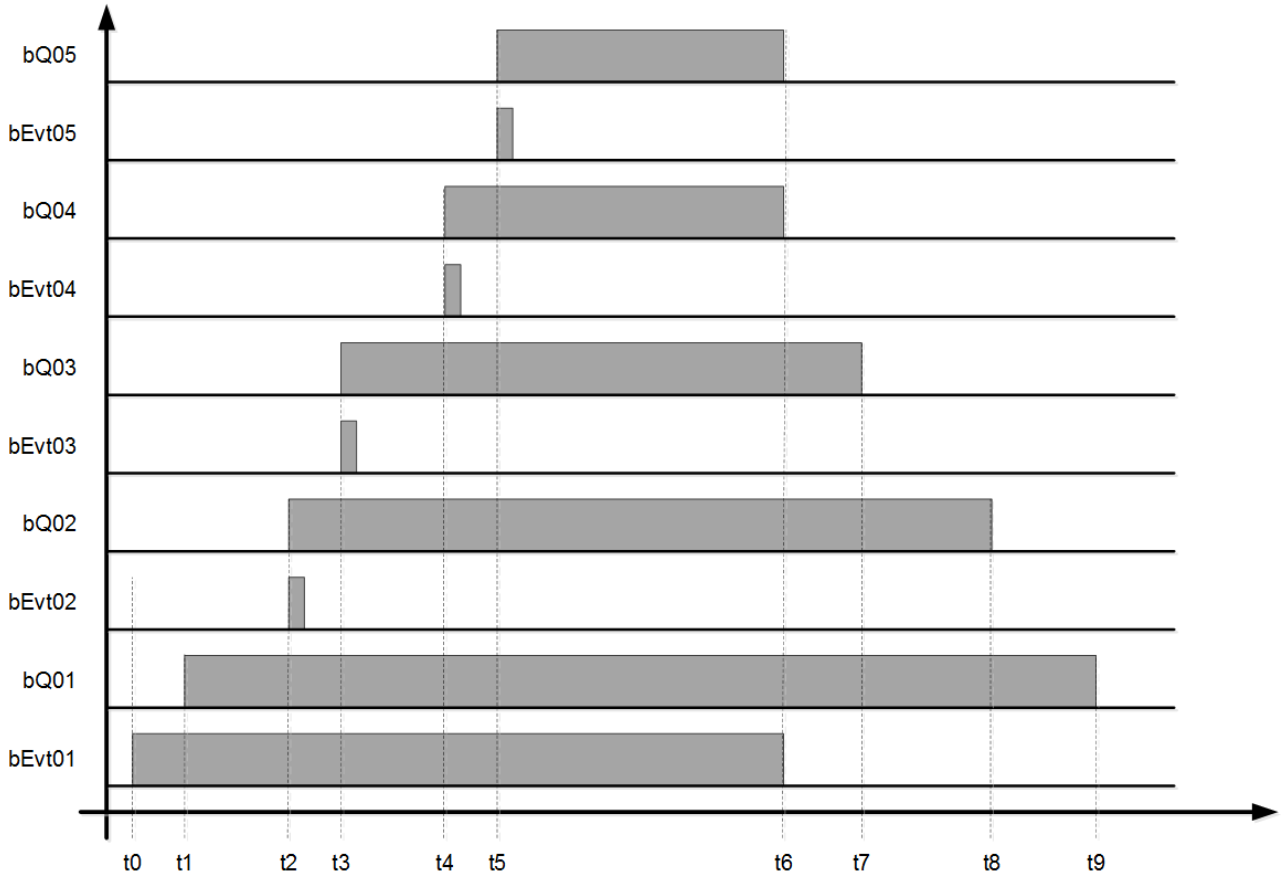


The function block FB_BA_StepCtrl12 is used for output sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs from *bQ01* to *bQ12* are set to FALSE. The 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 *bQ01…12* [s]
nDlyOff01…12	UDINT	Switch-off delay for output *bQ01…12* [s]

Outputs

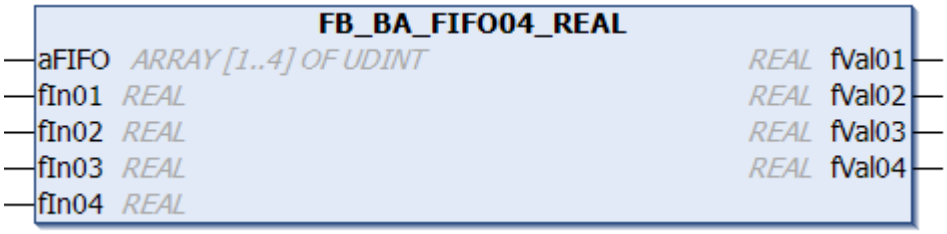
```
VAR_OUTPUT
  bQ01             : BOOL;
  bQ02             : BOOL;
  bQ03             : BOOL;
  bQ04             : BOOL;
  bQ05             : BOOL;
  bQ06             : BOOL;
  bQ07             : BOOL;
  bQ08             : BOOL;
  bQ09             : BOOL;
  bQ10             : BOOL;
  bQ11             : 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 [▶ 436]. 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 f*Val03*

fIn03 on output f*Val01*

fIn04 on output f*Val02*

Inputs

```
VAR_INPUT
  aFIFO        : Array [1..4] OF UDINT;
  fIn01…fIn04  : REAL;
END_VAR
```

Name	Type	Description
aFIFO	ARRAY OF 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 [▶ 437]. 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 f*Val04*

fIn02 on output f*Val03*

fIn03 on output f*Val01*

fIn04 on output f*Val02*

Inputs

```
VAR_INPUT
  aFIFO          : Array [1..8] OF UDINT;
  fIn01…fIn08    : REAL;
END_VAR
```

Name	Type	Description
aFIFO	ARRAY OF 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

6.1.2.2.3.1.5.3.12 LastWriterWins

6.1.2.2.3.1.5.3.12.1 FB_BA_LastWriterWins_R04

The function block switches the input value *fInxx* at output fQ that changed last.

The output *nActInput* shows which input is currently being output.

As long as nothing changes on the function block (the PLC has just started) and all inputs are set to "0", "0" is also output at the outputs *fQ* and *nActInput*.

If several inputs change simultaneously in a PLC cycle, the input with the lower ordinal number has priority, e.g.: *fIn01* before *fIn02*.

Inputs

```
VAR_INPUT
  aFIFO          : Array [1..8] OF UDINT;
  fIn01…fIn08    : REAL;
END_VAR
```

Name	Type	Description
fIn01…fIn04	REAL	Input values.

Outputs

```
VAR_OUTPUT
  fQ           : REAL;
  nActInput    : USINT;
END_VAR
```

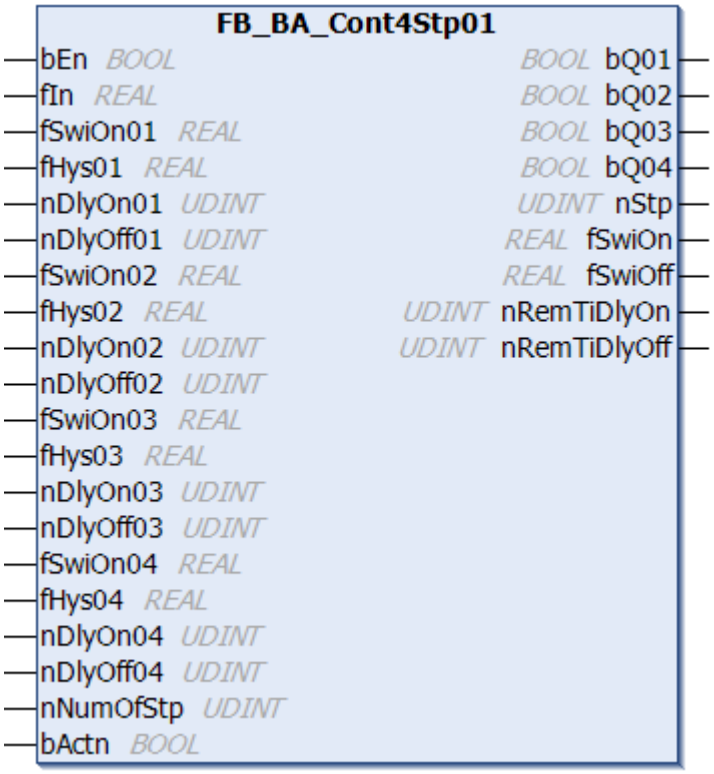
Name	Type	Description
fQ	REAL	Output value.
nActInput	USINT	Indicates which input value is currently being output.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.55	Tc3_BA2 from v5.3.19.0

6.1.2.2.3.1.5.4 Hysteresis 2-Point-Control

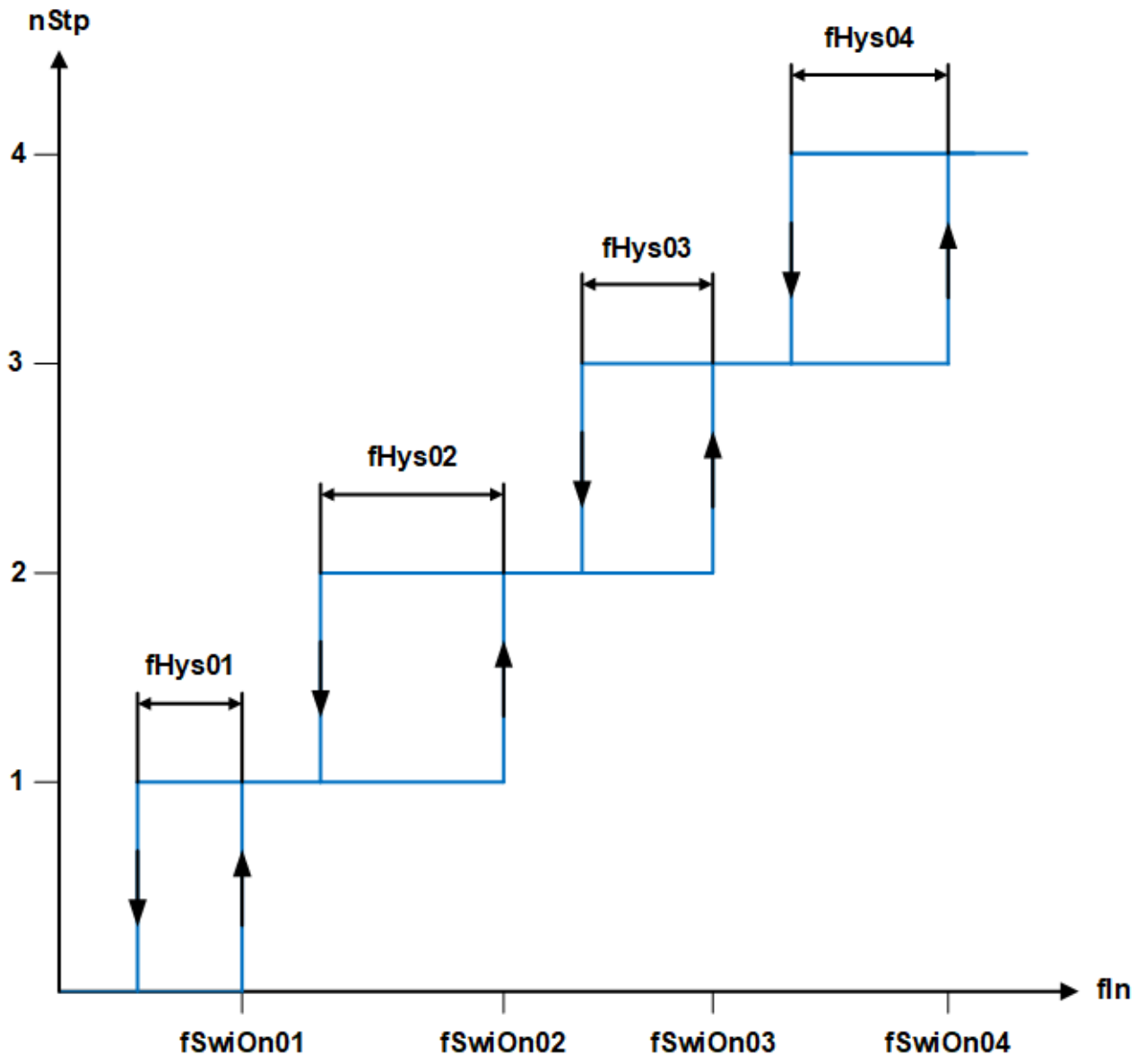
6.1.2.2.3.1.5.4.1 FB_BA_Cont4Stp01



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

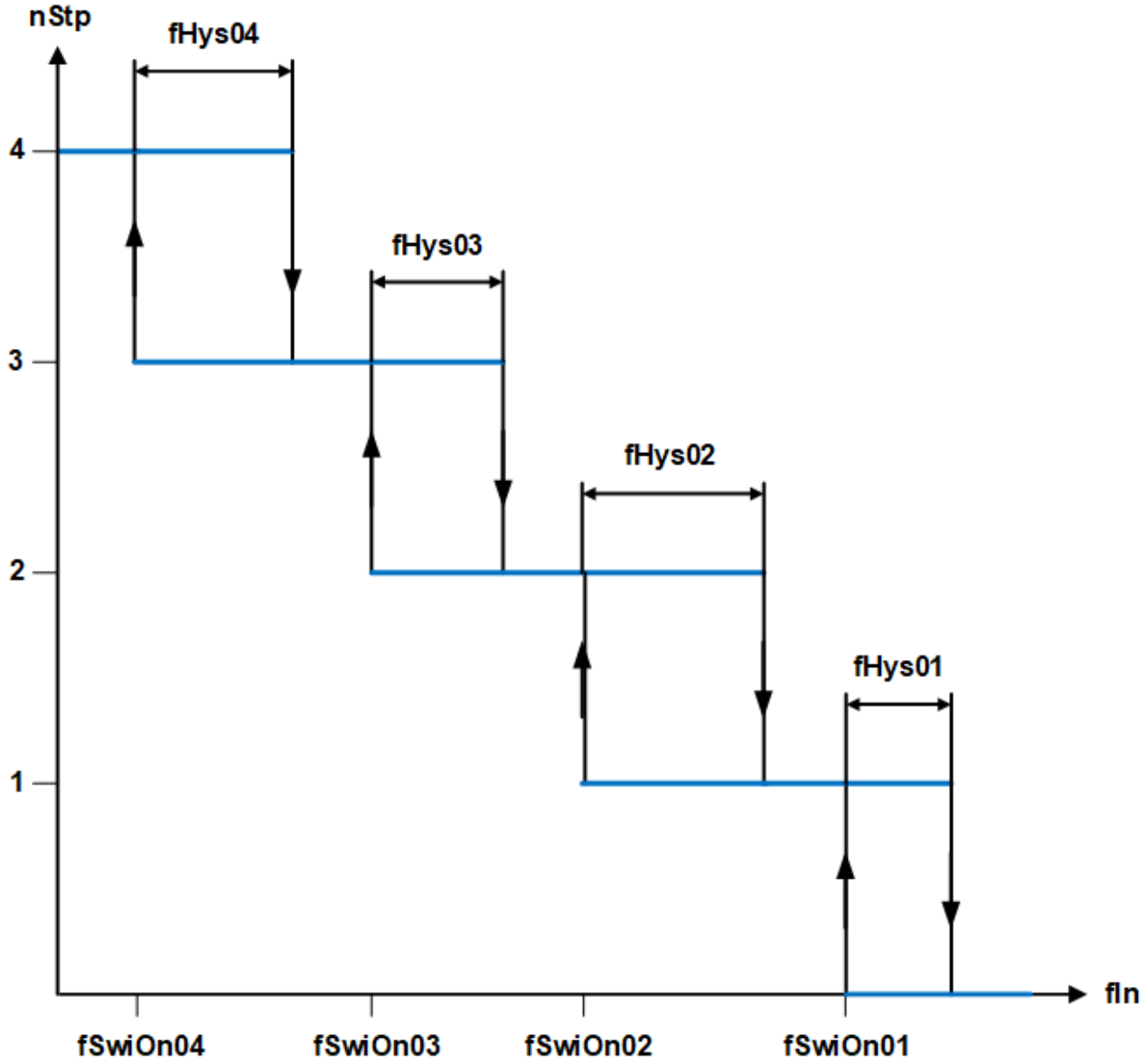
Control direction of parameter *bActn* = FALSE = Reverse = Heating



nStp	nNumOf-Stp	fSwiOn	fSwiOff	nRemTiDlyOn	nRemTiDlyOff	bQ01	bQ02	bQ03	bQ04
0	0	fSwiOn01	fSwiOn01 - fHys01	nDlyOn01	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	fSwiOn02	fSwiOn01 - fHys01	nDlyOn02	nDlyOff01	TRUE	FALSE	FALSE	FALSE
2	>= 2	fSwiOn03	fSwiOn02 - fHys02	nDlyOn03	nDlyOff02	TRUE	TRUE	FALSE	FALSE
3	>= 3	fSwiOn04	fSwiOn03 - fHys03	nDlyOn04	nDlyOff03	TRUE	TRUE	TRUE	FALSE
4	>= 4	fSwiOn04	fSwiOn04 - fHys04	0	nDlyOff04	TRUE	TRUE	TRUE	TRUE

Diagram 02

Control direction parameter *bActn* = TRUE = Direct = Cooling



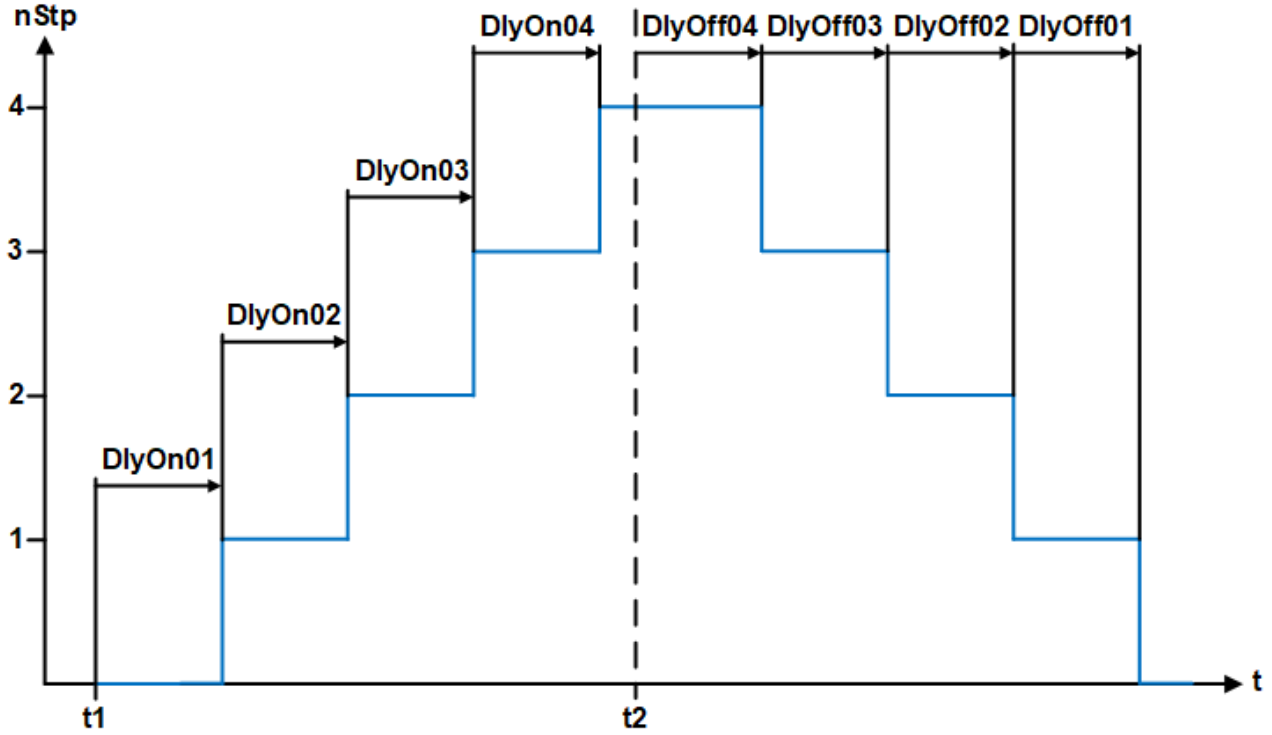
nStp	nNumOf-Stp	fSwiOn	fSwiOff	nRemTiDlyOn	nRemTiDlyOff	bQ01	bQ02	bQ03	bQ04
0	0	fSwiOn01	fSwiOn01 + fHys01	nDlyOn01	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	fSwiOn02	fSwiOn01 + fHys01	nDlyOn02	nDlyOff01	TRUE	FALSE	FALSE	FALSE
2	>= 2	fSwiOn03	fSwiOn02 + fHys02	nDlyOn03	nDlyOff02	TRUE	TRUE	FALSE	FALSE
3	>= 3	fSwiOn04	fSwiOn03 + fHys03	nDlyOn04	nDlyOff03	TRUE	TRUE	TRUE	FALSE
4	4	fSwiOn04	fSwiOn04 + fHys04	0	nDlyOff04	TRUE	TRUE	TRUE	TRUE

Diagram 03

Timing of the switch-on and switch-off delays

At time t1 *fIn* jumps from *fSwiOn01* to *fSwiOn04*

At time t2 *fIn* 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 *bEn* 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
```

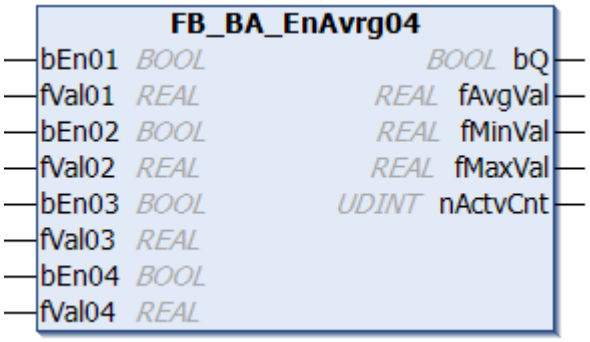
Name	Type	Description
bQ01	BOOL	Display of status step 01 TRUE = ON; FALSE = OFF nStp >= 1
bQ02	BOOL	Display of status step 02 TRUE = ON; FALSE = OFF nStp >= 2
bQ03	BOOL	Display of status step 03 TRUE = ON; FALSE = OFF nStp >= 3
bQ04	BOOL	Display of status step 04 TRUE = ON; FALSE = OFF nStp >= 4
nStp	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

6.1.2.2.3.1.5.5 Mathematics

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 *bEn0x* = 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 *fVal0* to *fVal0x*.

Outputs

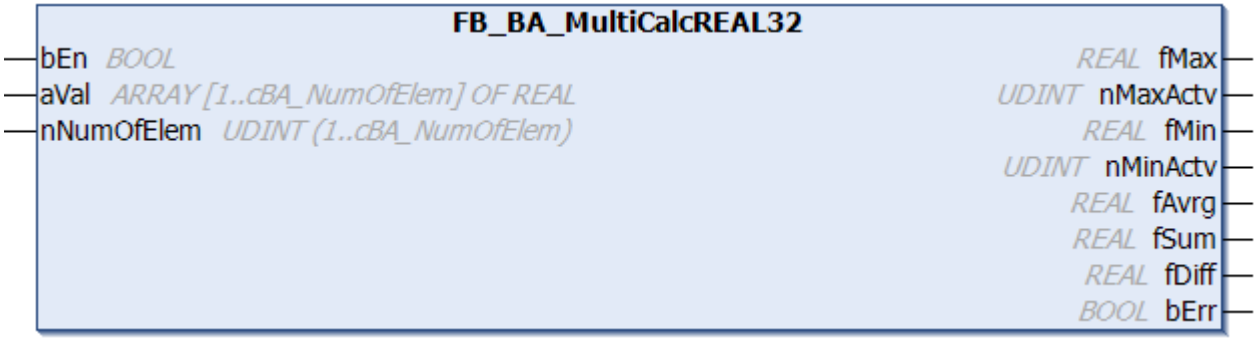
```
VAR_OUTPUT
  fAvgVal      : REAL;
  fMinVal      : REAL;
  fMaxVal      : REAL;
  nActvCnt     : UDINT;
END_VAR
```

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 *fAvrg*
- 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	fAvrg = 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	ARRAY OF REAL	Field with the values to be calculated.
nNumOfElem	UDINT	Number of input values to be used for the calculation.

Outputs

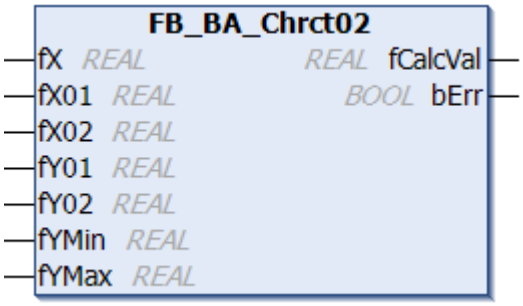
```
VAR_OUTPUT
  fMax         : REAL;
  nMaxActv     : UDINT;
  fMin         : REAL;
  nMinActv     : UDINT;
  fAvrg        : REAL;
  fSum         : REAL;
  fDiff        : REAL;
  bErr         : BOOL;
END_VAR
```

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.
fAvrg	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

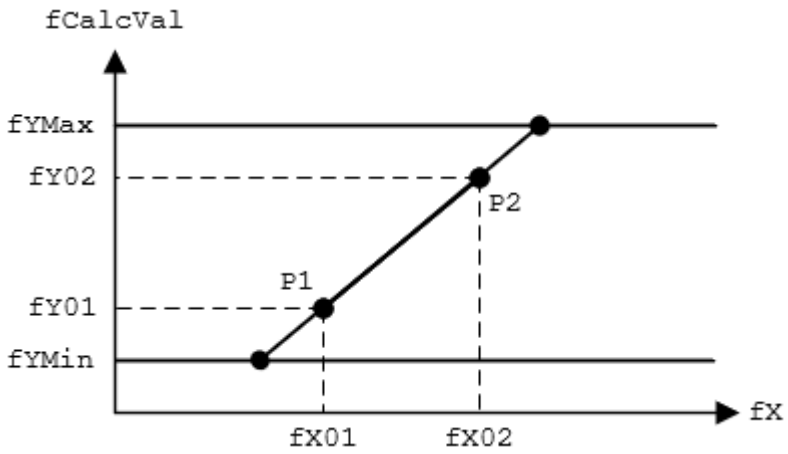
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.3 FB_BA_Chrct02



The function block FB_BA_Chrct02 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 *fCalcVal*. The input for the lower limit is limited to the global parameter BAComn_Global.fMinReal from the Tc3_BA2_Common library.
fYMax	REAL	Upper limit of *fCalcVal*. The input for the upper limit is limited to the global parameter BAComn_Global.fMaxReal from the Tc3_BA2_Common library.

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.

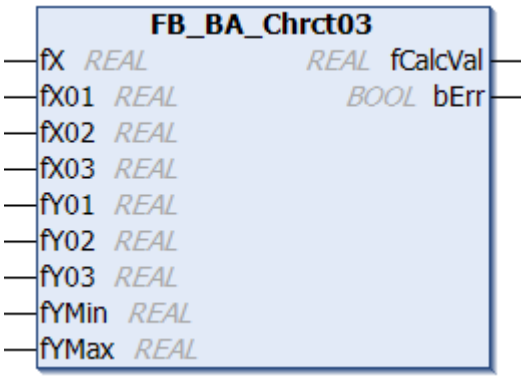
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [▶ 456].

Requirements

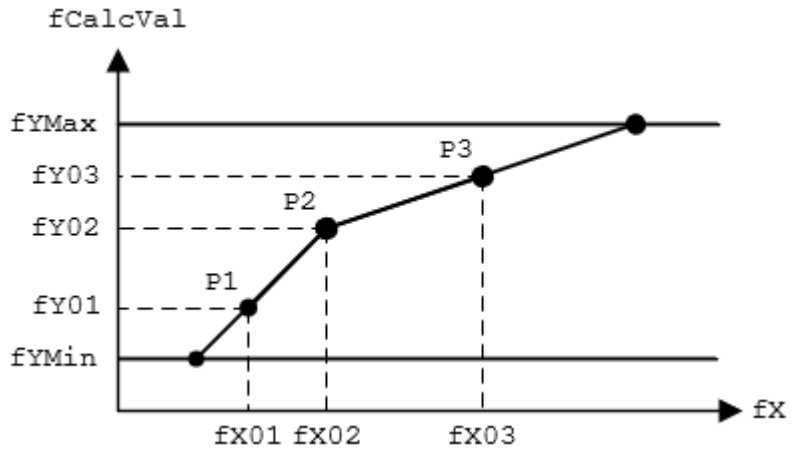
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.4 FB_BA_Chrtc03



The function block FB_BA_Chrct03 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 *fCalcVal*. The input for the lower limit is limited to the global parameter BAComn_Global.fMinReal from the Tc3_BA2_Common library.
fYMax	REAL	Upper limit of *fCalcVal*. The input for the upper limit is limited to the global parameter BAComn_Global.fMaxReal from the Tc3_BA2_Common library.

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.

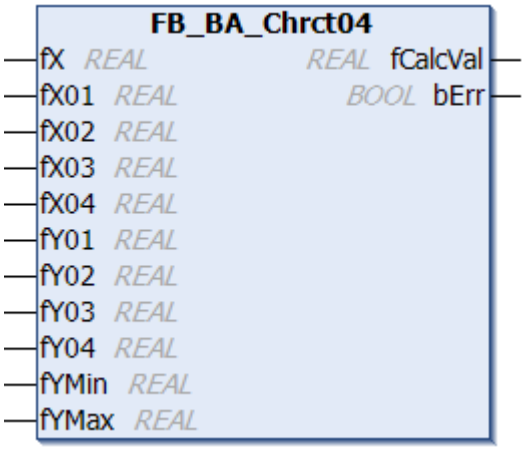
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [▶ 458].

Requirements

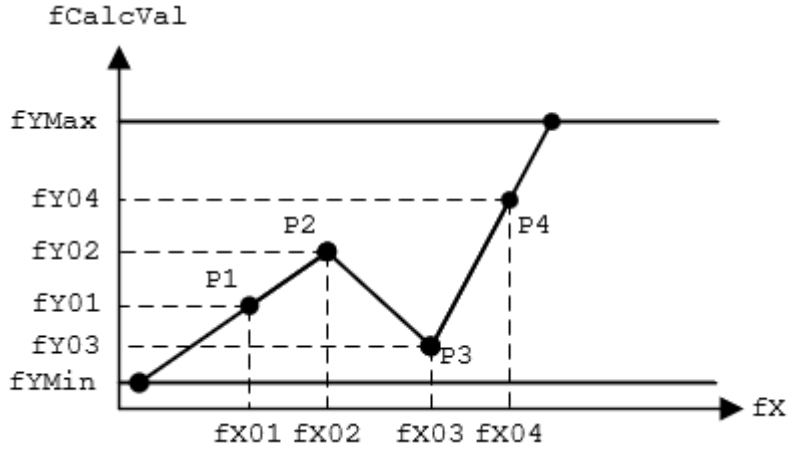
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.5 FB_BA_Chrct04



The function block FB_BA_Chrct04 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 *fCalcVal*. The input for the lower limit is limited to the global parameter BAComn_Global.fMinReal from the Tc3_BA2_Common library.
fYMax	REAL	Upper limit of *fCalcVal*. The input for the upper limit is limited to the global parameter BAComn_Global.fMaxReal from the Tc3_BA2_Common library.

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.

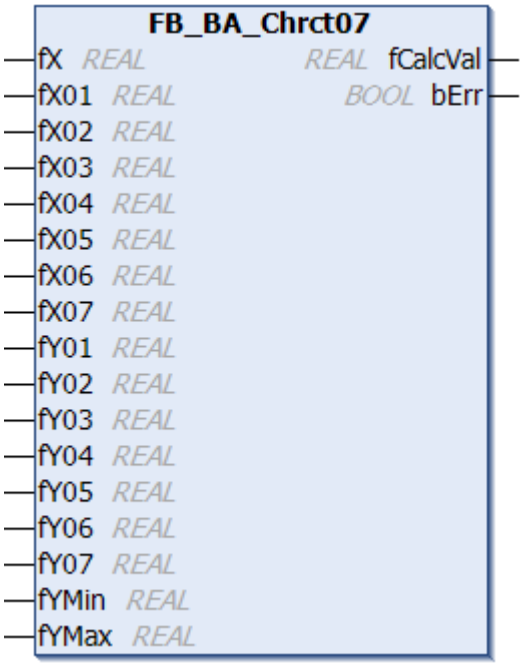
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [▶ 459].

Requirements

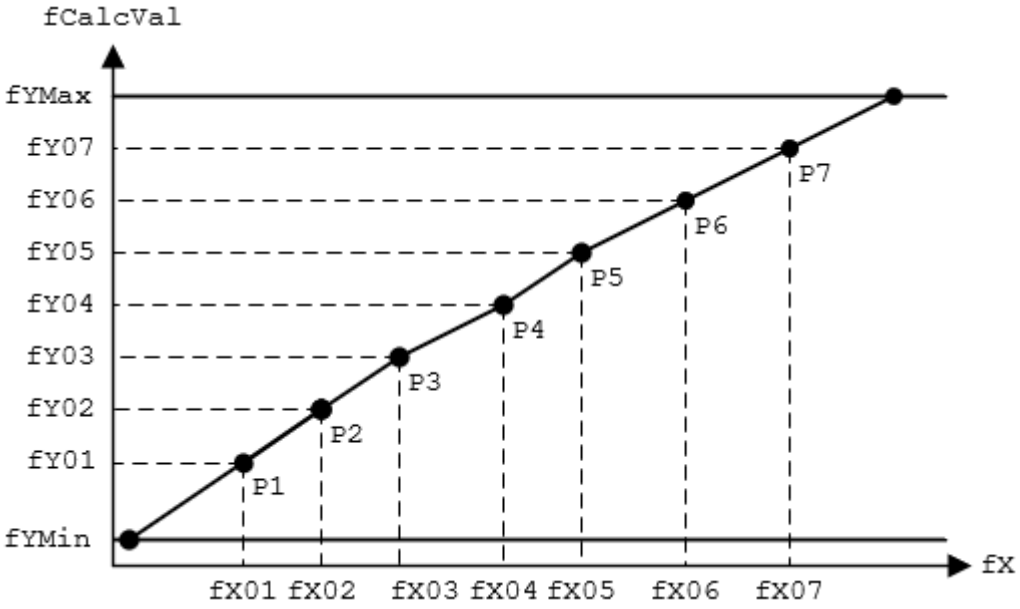
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.5.6 FB_BA_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 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 *fCalcVal*. The input for the lower limit is limited to the global parameter BAComn_Global.fMinReal from the Tc3_BA2_Common library.
fYMax	REAL	Upper limit of *fCalcVal*. The input for the upper limit is limited to the global parameter BAComn_Global.fMaxReal from the Tc3_BA2_Common library.

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.

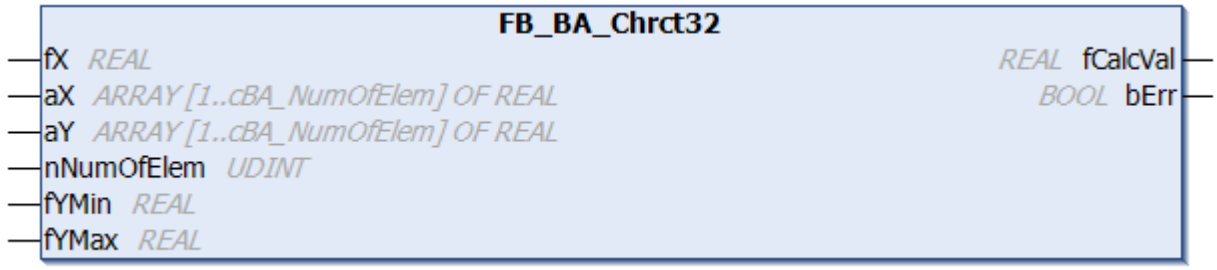
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [▶ 461].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

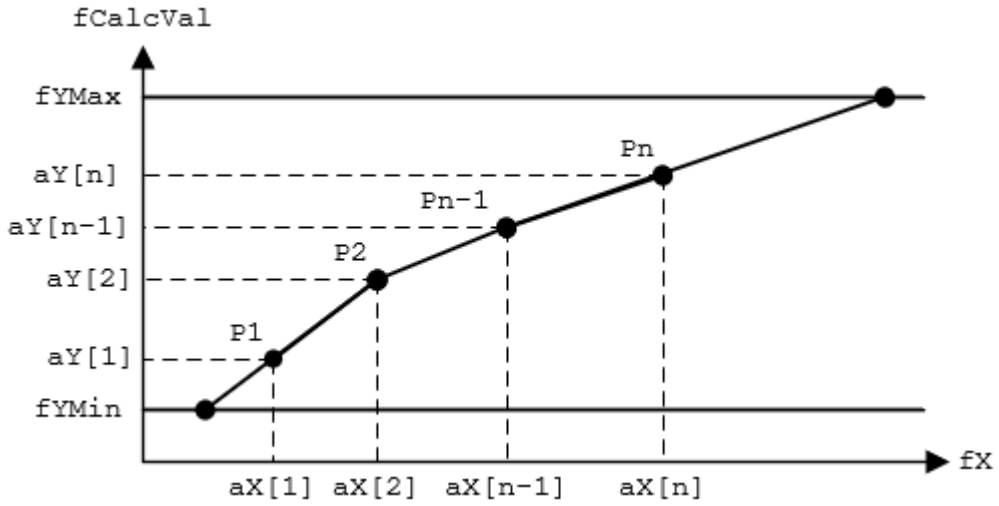
6.1.2.2.3.1.5.5.7 FB_BA_Chrct32



The function block FB_BA_Chrct32 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_Chrct02 [▶ 456], FB_BA_Chrct04 [▶ 459] and FB_BA_Chrct07 [▶ 461], 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..FB_BA_Chrct32.cBA_NumOfElem] OF REAL;
  aY           : ARRAY [1..FB_BA_Chrct32.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	ARRAY OF REAL	X-value for the interpolation points.
aY	ARRAY OF 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 *fCalcVal*. The input for the lower limit is limited to the global parameter BAComn_Global.fMinReal from the Tc3_BA2_Common library.
fYMax	REAL	Upper limit of *fCalcVal*. The input for the upper limit is limited to the global parameter BAComn_Global.fMaxReal from the Tc3_BA2_Common library.

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.

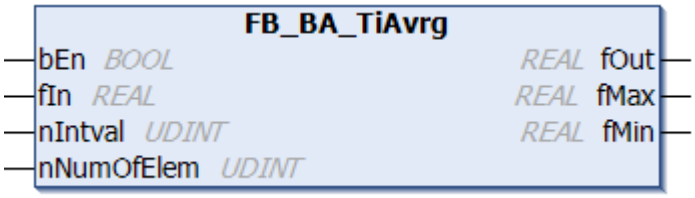
Properties

Name	Type	Access	Description
ErrorDescription	T_MaxString	Get	Issues a detailed description of errors, see error handling [▶ 461].

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_TiAvrg



The function block FB_BA_TiAvrg 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	nNu-mOfElem	fIn	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.
fIn	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

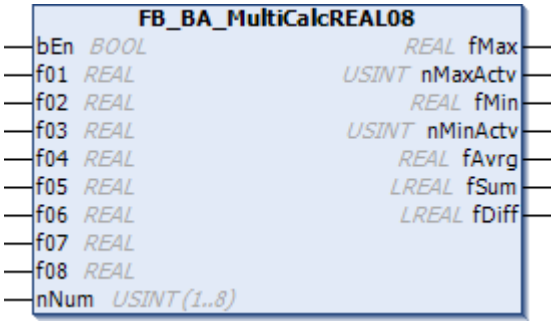
```
Ausgänge
  fOut        : REAL;
  fMax        : REAL;
  fMin        : REAL;
END_VAR
```

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

6.1.2.2.3.1.5.5.9 FB_BA_MultiCalcREAL08



The function block FB_BA_MultiCalcREAL08 calculates the following values from the 8 input values *f01...f08* in the enabled state(*bEn* = TRUE):

1. the maximum value of all inputs *fMax*
2. the input at which this maximum value is applied *nMaxActv*
3. the minimum value of all inputs *fMin*
4. the input at which this minimum value is applied *nMinActv*
5. the average of all inputs *fAvrg*
6. the sum of all inputs *fSum*
7. the difference between the maximum and minimum value *fDiff*

If not all inputs are used for the calculation, the number can be limited via an entry at *nNum*: *nNum* = 6, for example, can be used to limit the calculations to inputs f01...f06.

Any entry greater than 8 is automatically limited to 8, any entry less than 1 is automatically set to 1.

Sample:

Inputs	Output
bEn = TRUE	fMax = 32
f01 = 32	nMaxActv = 1
f02 = 17	fMin = 5
f03 = 5	nMinActv = 3
f04 = 9	fAvrg = 18.5
f05 = 16	fSum = 111
f06 = 32	fDiff = 27
f07 = 25	
f08 = 44	
nNum = 6	

Inputs

```
VAR_INPUT
  bEn          : BOOL;
  f01…f08      : REAL;
  nNum         : USINT(1..8);
END_VAR
```

Name	Type	Description
bEn	BOOL	Activation of the block function.
f01...f08	REAL	Input values to be used for the calculation.
nNum	USINT	Number of input values to be used for the calculation.

Outputs

```
VAR_OUTPUT
  fMax        : REAL;
  nMaxActv    : USINT;
  fMin        : REAL;
  nMinActv    : USINT;
  fAvrg       : REAL;
  fSum        : REAL;
  fDiff       : REAL;
END_VAR
```

Name	Type	Description
fMax	REAL	Maximum value of all inputs.
nMaxActv	USINT	Input at which the maximum value is present.
fMin	REAL	Minimum value of all inputs.
nMinActv	UDSNT	Input at which the minimum value is present.
fAvrg	REAL	Average value of all inputs
fSum	REAL	Sum of all inputs
fDiff	REAL	Difference between maximum and minimum value.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

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 *bEn* is FALSE, the message output *bQ* 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 *BAComn_Global.udiMaxSecInMilli* 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 *BAComn_Global.udiMaxSecInMilli* from the Tc3_BA2_Common library, see (BAComn_Global).

Outputs

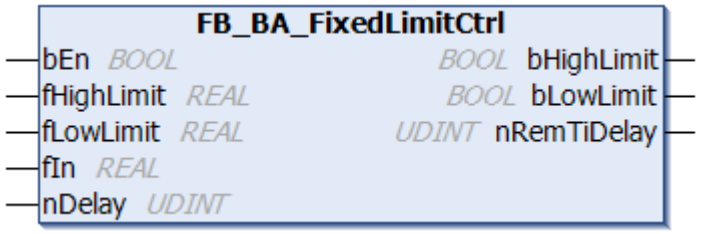
```
VAR_OUTPUT
  bQ                      : BOOL;
  nRemTiFdbDelay          : UDINT;
  nRemTiInterruptionDelay : UDINT;
END_VAR
```

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 *bQ* is set. The default comes from *nFdbDelay*.
nRemTiInterruptionDelay	UDINT	Remaining time [s] until output *bQ* is set. The default comes from *nInterruptionDelay*.

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 *bEn* is FALSE, the message outputs *bHighLimit* and *bLowLimit* 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 *bHighLimit/bLowLimit*. The input of the time is limited to the global parameter *BAComn_Global.udiMaxSecInMilli* from the Tc3_BA2_Common library (see BAComn_Global).

Outputs

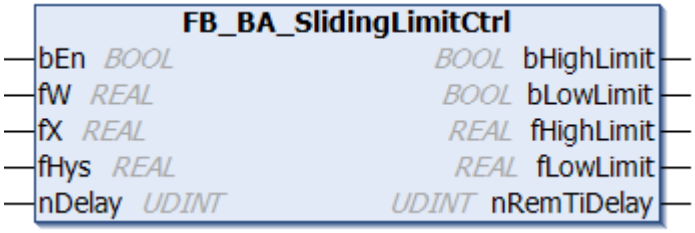
```
VAR_OUTPUT
  bHighLimit          : BOOL;
  bLowLimit           : BOOL;
  nRemTiDelay         : UDINT;
END_VAR
```

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 *bHighLimit* or *bLowLimit* is set.

Requirements

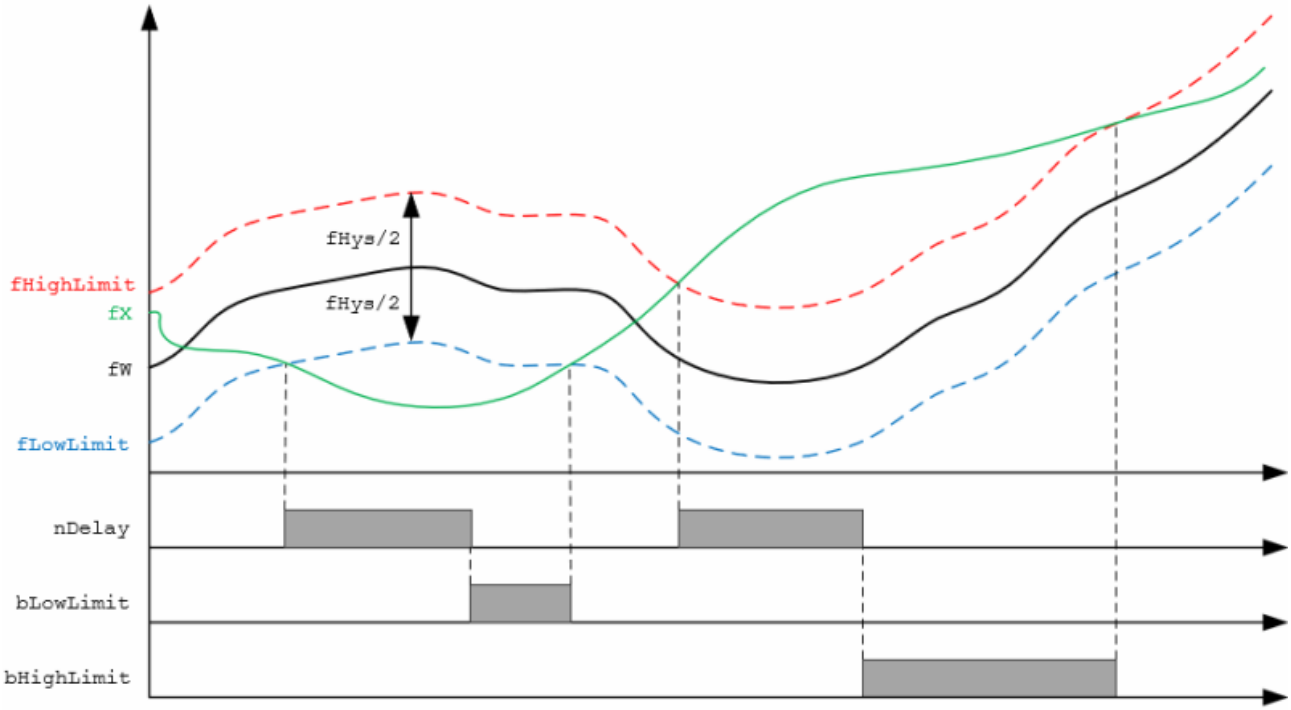
Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

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 *bEn* is FALSE, the message outputs *bHighLimit* and *bLowLimit* are also FALSE.
fW	REAL	Setpoint
fX	REAL	Actual value
fHys	REAL	Hysteresis
nDelay	UDINT	Response delay [s] of the outputs *bHighLimit/bLowLimit*. The input of the time is limited to the global parameter *BAComn_Global.udiMaxSecInMilli* from the Tc3_BA2_Common library (see BAComn_Global).

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 *bHighLimit* or *bLowLimit* is set. The default comes from *nDelay*.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

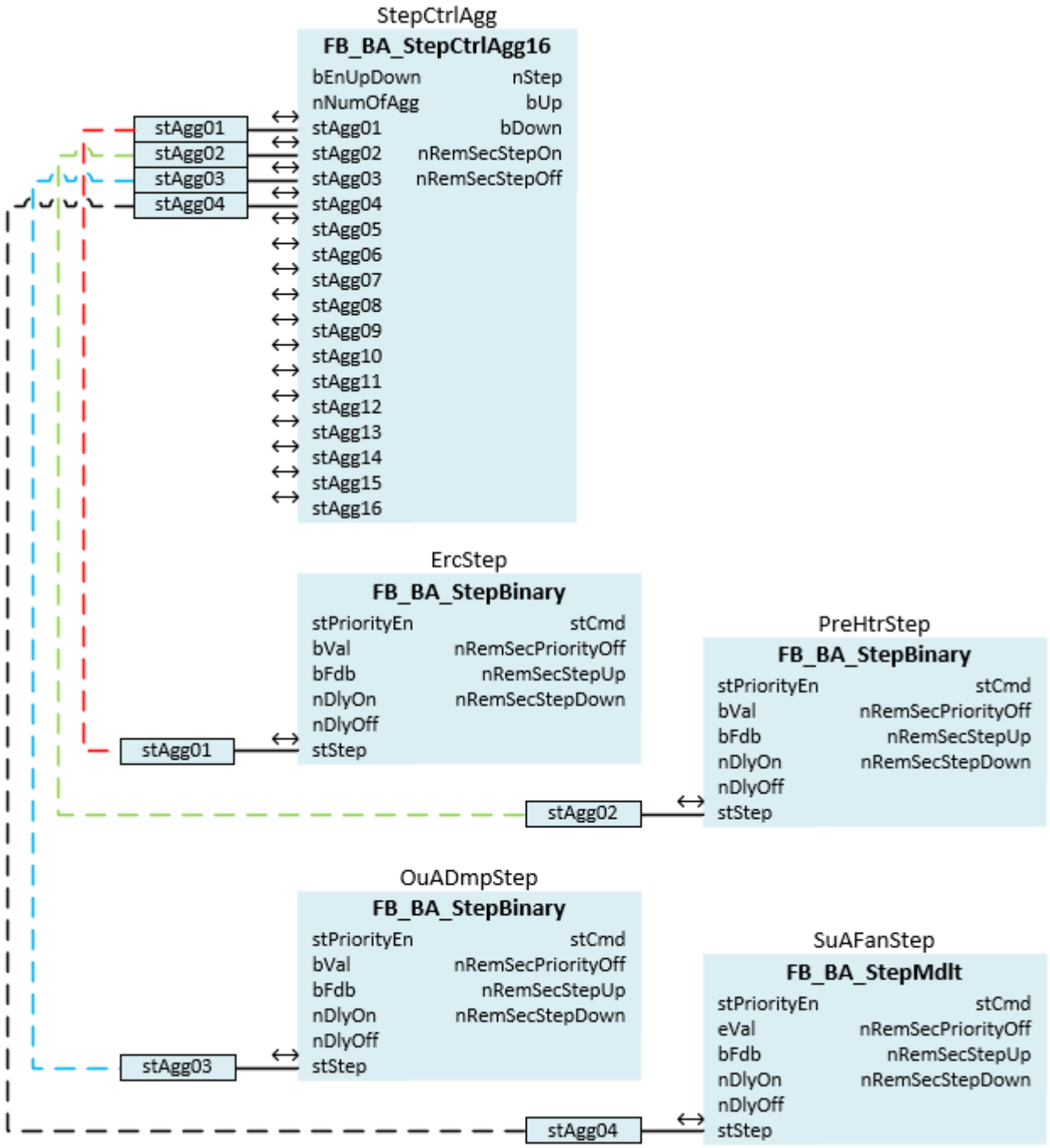
6.1.2.2.3.1.5.7 StepControl

6.1.2.2.3.1.5.7.1 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 [▶ 473], FB_BA_StepMdlt [▶ 475]) 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].

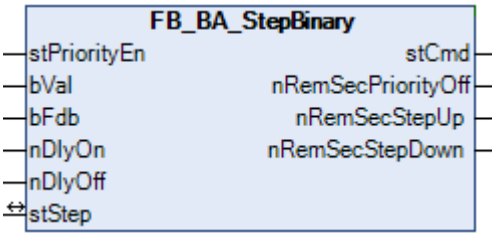
Inputs Outputs

Name	Type	Description
stAgg01-16	ST_BA_Step [▶ 277]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [▶ 168] 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

6.1.2.2.3.1.5.7.2 FB_BA_StepBinary



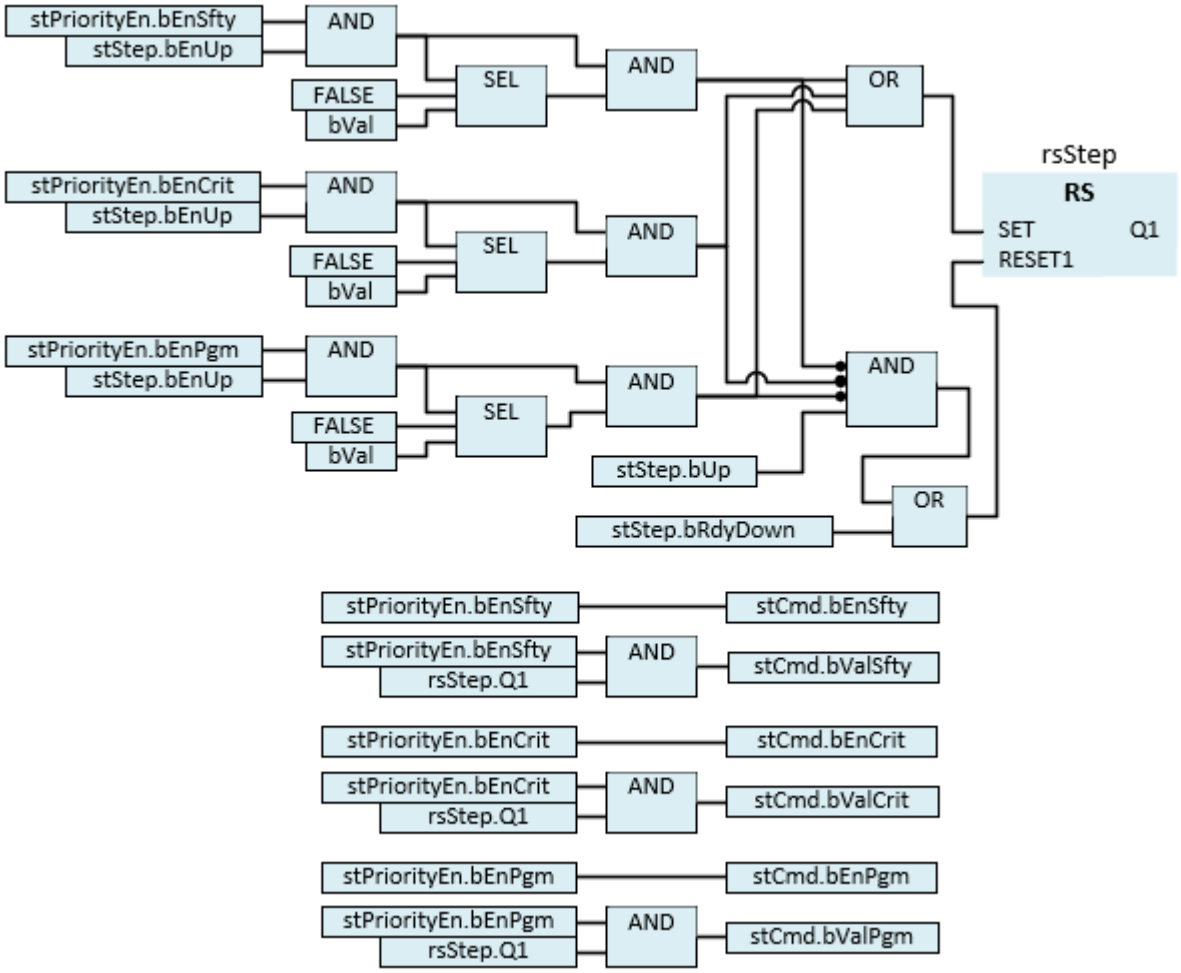
The function block is part of a step sequence control.

It communicates via the data and command structure *stStep* with the higher-level control unit of the step sequence control FB_BA_StepCtrlAgg16 [▶ 471].

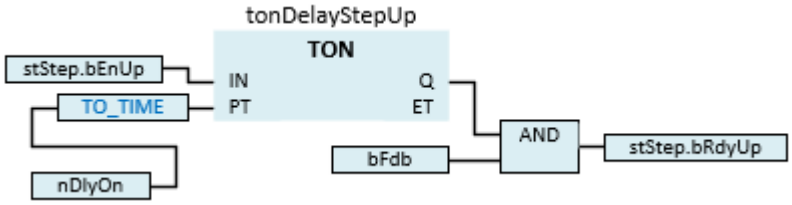
The command structure *stPriorityEn* contains the plant enable and the priorities.

The command structure *stCmd* controls the connected aggregate.

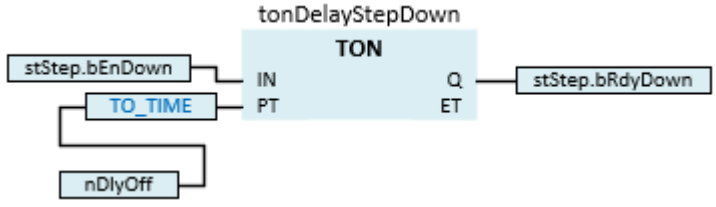
Switch-on conditions of the connected aggregate



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 [▶ 278]	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. *bFdb* 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 [▶ 277]	The command structure *stCmd* 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].

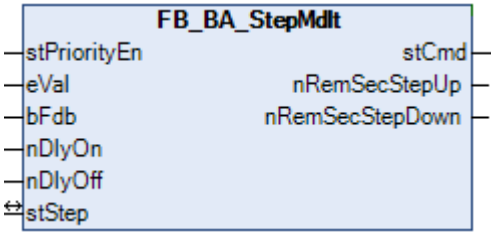
Inputs Outputs

Name	Type	Description
stStep	ST_BA_Step [▶ 277]	Data and command structure between the step sequence function block *FB_BA_StepBinary* and the control block of the step sequence FB_BA_StepCtrlAgg16 [▶ 471].

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.1.5.7.3 FB_BA_StepMdlt



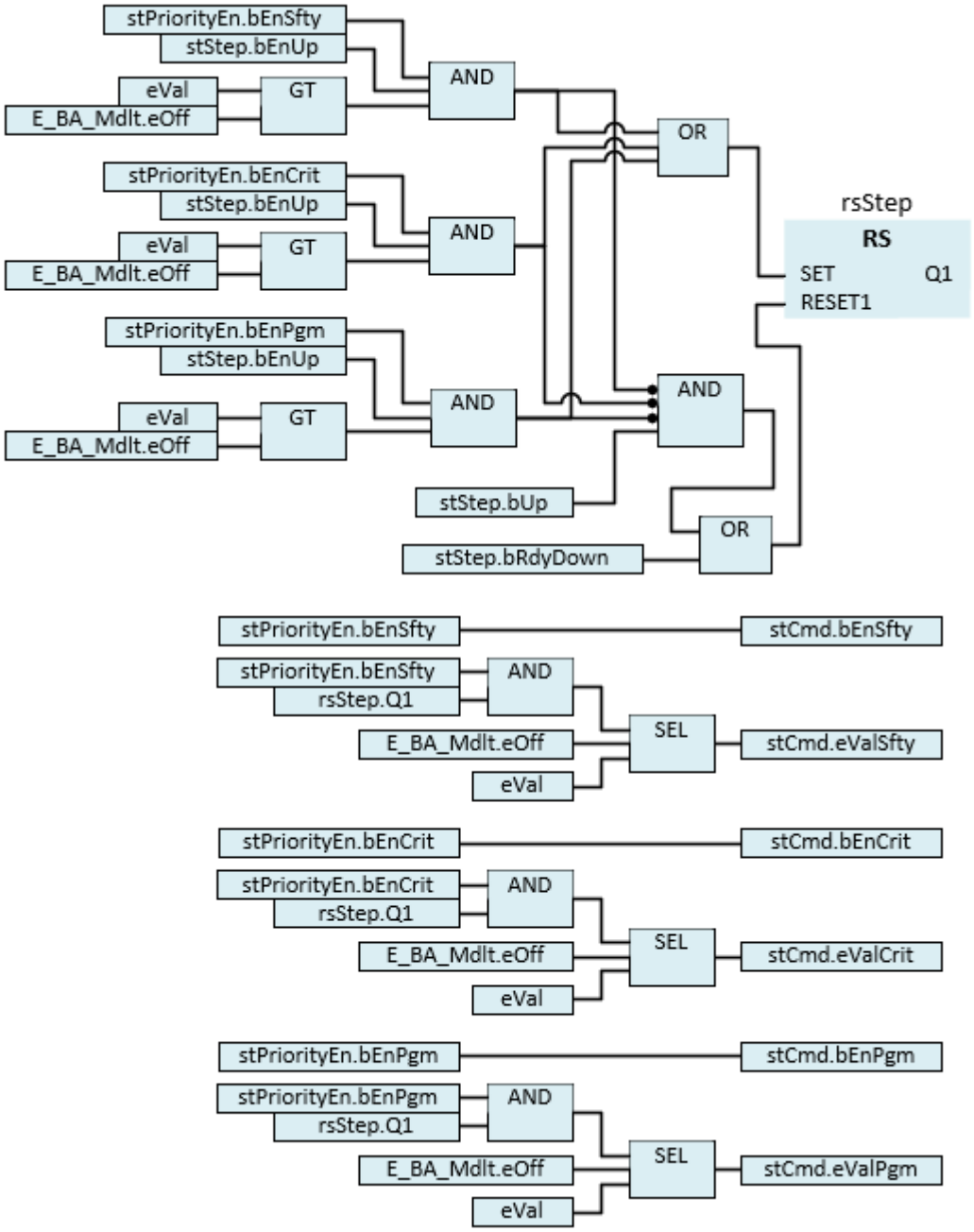
The function block is part of a step sequence control.

It communicates via the data and command structure *stStep* with the higher-level control unit of the step sequence control FB_BA_StepCtrlAgg16 [▶ 471].

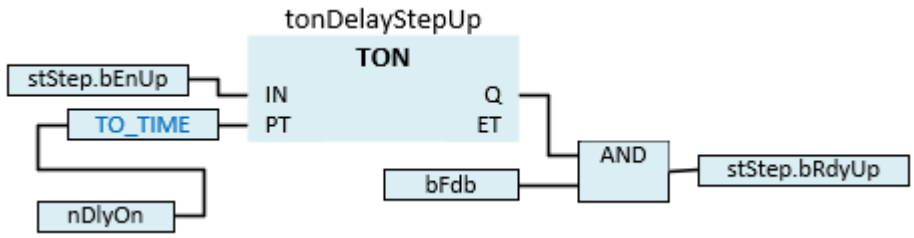
The command structure *stPriorityEn* contains the plant enable and the priorities.

The command structure *stCmd* controls the connected aggregate.

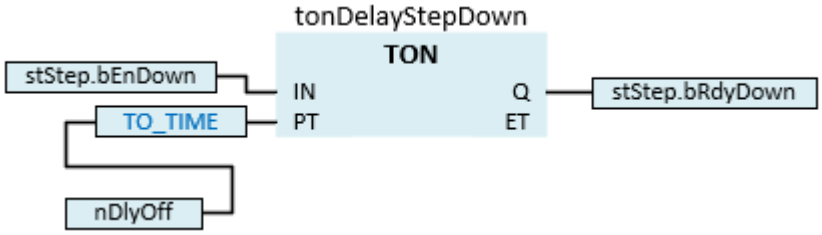
Switch-on condition of the connected aggregate



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 [▶ 278]	The command structure includes the plant enable and the priorities "Safety", "Critical" and "Program".
eVal	E_BA_Mdlt [▶ 270]	Switch-on value for the connected aggregate.
bFdb	BOOL	Feedback of the connected aggregate. The feedback is required for switching to the next step. *bFdb* 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 [▶ 276]	The command structure *stCmd* 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

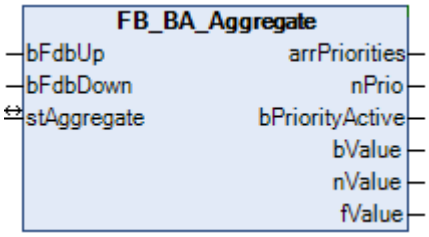
Name	Type	Description
stStep	ST_BA_Step [▶ 277]	Data and command structure between the step sequence function block *FB_BA_StepBinary* and the control block of the step sequence FB_BA_StepCtrlAgg16 [▶ 471].

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 PlantControl

6.1.2.2.3.1.5.8.1 FB_BA_Aggregate



The aggregate function block represents a receive block of a step sequence controller.

It communicates via the data and command structure *stAggregate* with the higher-level plant control unit of the step sequence control FB_BA_PlantControl [▶ 480].

The function block is used to evaluate the structure *stAggregate*. The result is output at the inputs and outputs of the function block.

Syntax

```
FUNCTION_BLOCK FB_BA_Aggregate
VAR_INPUT
  bFdbUp              : BOOL;
  bFdbDown            : BOOL;
END_VAR
VAR_OUTPUT
  arrPriorities       : ARRAY [0..BA_Param.nMaxPriority] OF BOOL;
  nPrio               : UDINT;
  bPriorityActive     : BOOL;
  bValue              : BOOL;
  nValue              : UDINT;
  fValue              : REAL;
END_VAR
VAR_IN_OUT
  stAggregate         : ST_BA_Aggregate;
END_VAR
```

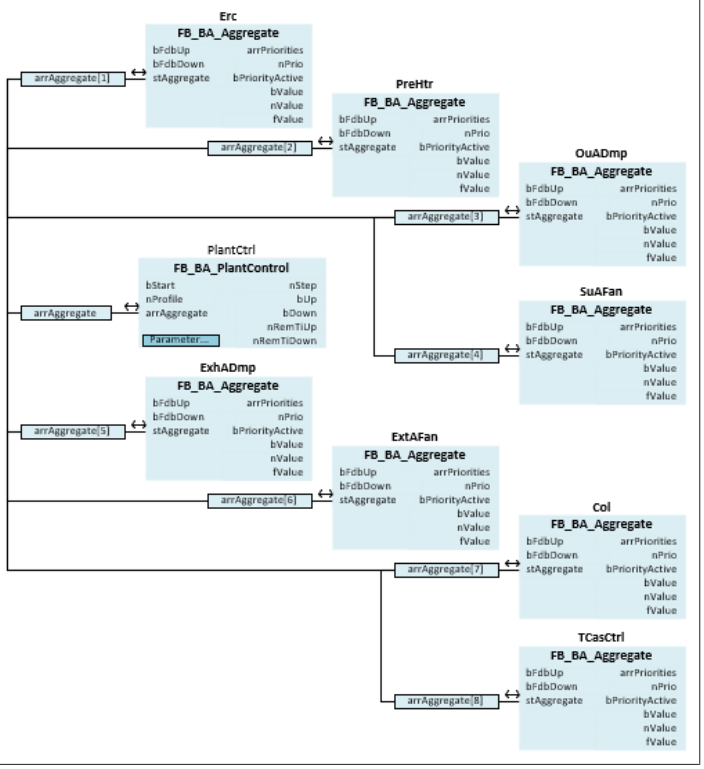
Inputs

Name	Type	Description
bFdbUp	BOOL	A feedback from the aggregate is applied to the input. It is used within the step sequence control to switch up to the next higher step. The feedback is transmitted to the plant control unit via the data and command structure *stAggregate*.
bFdbDown	BOOL	A feedback from the aggregate is applied to the input. It is used to switch down to the next lower step. The feedback is transmitted to the plant control unit via the data and command structure *stAggregate*.

Outputs

Name	Type	Description
arrPriorities	BOOL	The binary array outputs the enables of the priorities. Only one priority is active at a time. *nPrio* indicates which one it is.
nPrio	UDINT	Indicates which priority is active.
bPriorityActive	BOOL	One priority is enabled.
bValue	BOOL	Binary switching value for the aggregate.
nValue	UDINT	Multi-stage switching value for the aggregate.
fValue	REAL	Analog control value for the aggregate.

/ Inputs Outputs

Name	Type	Description
stAggregate	ST_BA_Aggregate [▶ 278]	The aggregate structure is used for bidirectional communication between the connected aggregate FB_BA_Aggregate and the plant control FB_BA_PlantControl [▶ 480]. 

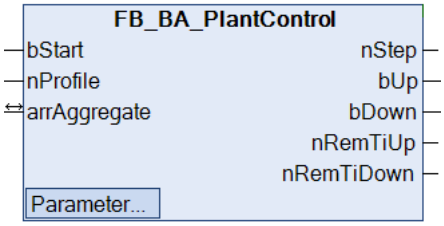
Properties

Name	Type	Access	Description
Down	BOOL	Get	Indicates that the sequence of the step sequence control is in the falling state.
OpMode	UDINT	Get	Displays the current operation mode of the plant control.
Prio	UDINT	Get	Indicates which priority is the active one in the plant control.
Step	UDINT	Get	Indicates the current step of the plant control.
Up	BOOL	Get	Indicates that the sequence of the step sequence control is in the rising state.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.55	Tc3_BA2 from v5.3.19.0

6.1.2.2.3.1.5.8.2 FB_BA_PlantControl



The function block realizes the sequential start-up of a plant.

When starting, the aggregates are switched on one after the other in a specific order.

When switching off the plant, the aggregates are switched off in reverse order.

The steps of the start sequence and the profiles with the specific releases of individual aggregates result in a switching matrix.

Switching matrix

arrStepDescription		arrProfileDescription	[1]	[2]	[3]	[4]	[5]	[6]
		sDescription	Auto	NightCooling	CoolDownProtection	OverHeatingProtection	ForcedVentilation	CriticalOff
		nPrio	Priority.nProgra	Priority.nProgram	Priority.nProgram	Priority.nProgram	Priority.nCritical	Priority.nCritical
		arrProfileParameter						
		[Profile,Step]						
[1]	Erc	bValue	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[2]	PreHeater PreRinse	bValue	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[3]	DamperOuA	bValue	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[4]	FanSupplyAir	bValue	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		nValue	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eMax	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eOf
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[5]	DamperExhA	bValue	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[6]	FanExtractAir	bValue	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		nValue	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eMax	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eOf
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[7]	Cooler	bValue	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[8]	Enabling TemperatureControl	bValue	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		nValue	E_BA_Mdlt.eOn	E_BA_Mdlt.eOff	E_BA_Mdlt.eMax	E_BA_Mdlt.eMin	E_BA_Mdlt.eOn	E_BA_Mdlt.eOf
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

The plant steps are shown in the column framed vertically in red.

The start sequence of the plant begins with the element [1] ERC, energy recovery. The element [2] PreHeater PreRinse, etc. follows.

The starting sequence of the aggregates is the same for all plant profiles.

The horizontal green framed area contains the descriptions of the plant profiles. The profiles can be used to command aggregates with regard to project-specific requirements. Each plant profile has its own step sequence within which the aggregates are commanded. The input variable *nProfile* determines the plant profile or the column in the control matrix.

The area highlighted in yellow contains the command for the ERC (energy recovery) aggregate for the Auto profile (normal operation). As this is the binary release of the aggregate, *bValue* = TRUE. Multi-stage aggregates can be commanded with the variable *nValue* and continuous aggregates with the variable *fValue*.

The blue framed field contains variables for parameterization of the start sequence. These parameters can be used to define different timers and conditions for switching the start sequence up and down for each profile.

In this sample, *bEnFdbUp* = TRUE. This means that feedback from the aggregate PreHeater is expected before the start sequence switches up and the next aggregate with the index [3] DmpOuA (outside air damper) switches on.

The start of the sequence is activated depending on the input variable *bStart*. A rising edge at *bStart* begins commanding the aggregate with the index [1] and continues with the start process until the highest level of switching sequences is reached.

In the picture, the start of the sequence in the Auto profile is illustrated with the green arrow and contains 8 switching sequences.

A falling edge at *bStart* triggers the shutdown of the start sequence in reverse order. This is shown in the picture by the red arrow.

If *bStart* becomes FALSE (falling edge) during the start process, the sequence runs down again from the currently active step to the complete shutdown of all aggregates.

This is shown in the picture by the red arrow.

NOTICE

Locking the aggregates

If a changeover is made from one profile to another during upward or downward travel, the commands for the new profiles are issued directly to the aggregates until the plant step *nStep*. The aggregates above the current sequence (> *nStep*) receive the following commands: *bValue* = FALSE, *nValue* = 0 and *fValue* = 0.0

Necessary interlocks regarding the switch-on and switch-off sequence are therefore not always guaranteed.

External interlocking of the aggregates must therefore be used to ensure that no critical switching states occur. Alternatively, the active operation mode can first be completely shut down in its sequence before switching to a new profile using *nProfile*.

Syntax

```
FUNCTION_BLOCK FB_BA_PlantCtrl
VAR_INPUT
  bStart                  : BOOL;
  nProfile                : UDINT;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  arrStepDescription      : ARRAY [1..MAX(1,BA_Param.nMaxAggregate)] OF T_MaxString;
  arrProfileDescription   : ARRAY [1..MAX(1,BA_Param.nMaxProfile)] OF ST_BA_ProfielDescription;
  arrProfileParameter     : ARRAY [1..MAX(1,BA_Param.nMaxProfile),1..MAX(1,BA_Param.nMaxAggregate)]
 OF ST_BA_ProfileParameter;
END_VAR
VAR_OUTPUT
  nStep                : UDINT;
  bUp                  : BOOL;
  bDown                : BOOL;
  nRemTiUp             : UDINT;
  nRemTiDown           : UDINT;
END_VAR
```

```
VAR_IN_OUT
  arrAggregate          : ARRAY [1..MAX(1,BA_Param.nMaxAggregate)] OF ST_BA_Aggregate;
END_VAR
```

Inputs

Name	Type	Description
bStart	BOOL	A rising edge starts the sequence. On a falling edge, the sequence is run down in reverse order.
nProfile	UDINT	The current, valid profile is transferred to the control matrix via the input.

Inputs CONSTANT PERSISTENT

Name	Type	Description
arrStepDescription	ARRAY [1..MAX(1,BA_Param.nMaxAggregate [▶ 284])] OF T_MaxString	The plain text of the plant steps is specified in the array. This information is purely for display purposes. The red framed area in the switching matrix indicates this. Initialization plain text plant steps, see sample 1.
arrProfileDescription	ARRAY [1..MAX(1,BA_Param.nMaxProfile [▶ 284])] OF ST_BA_ProfileDescription [▶ 279]	The profiles of the control matrix are defined in the array. The green framed area in the switching matrix indicates this. Initialization of the profile description, see sample 2.
arrProfileParameter	ARRAY [1..MAX(1,BA_Param.nMaxProfile [▶ 284]),1..MAX(1,BA_Param.nMaxAggregate [▶ 284])] OF ST_BA_ProfileParameter [▶ 280]	The two-dimensional array contains all the parameters for commanding the aggregates and parameterization of the step sequences. Each element of the array contains the profile parameter structure. In the switching matrix, the orange and blue framed area indicates a part of the array. Initialization of the parameters for the Auto profile, see sample 3.

Samples

Sample 1: Initialization plain text plant steps

```
PlantControl.arrStepDescription[1]  := txtEvent_EnergyRecovery;
PlantControl.arrStepDescription[2]  := txtEvent_PreRinse;
PlantControl.arrStepDescription[3]  := txtEvent_DamperOutsideAir;
PlantControl.arrStepDescription[4]  := txtEvent_SupplyAirFan;
PlantControl.arrStepDescription[5]  := txtEvent_DamperExhaustAir;
PlantControl.arrStepDescription[6]  := txtEvent_ExtractAirFan;
PlantControl.arrStepDescription[7]  := txtEvent_Cooler;
PlantControl.arrStepDescription[8]  := txtEvent_EnableTemperatureControl;
```

Sample 2: Initialization of the profile description

```
PlantControl.arrProfileDescription[1].sDescription  := txtEvent_Auto;
PlantControl.arrProfileDescription[1].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[2].sDescription  := txtEvent_NightCooling;
PlantControl.arrProfileDescription[2].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[3].sDescription  := txtEvent_CoolDownProtection;
PlantControl.arrProfileDescription[3].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[4].sDescription  := txtEvent_OverHeatingProtection;
PlantControl.arrProfileDescription[4].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[5].sDescription  := txtEvent_ForcedVentilation;
PlantControl.arrProfileDescription[5].nPrio         := Priority.nCritical;

PlantControl.arrProfileDescription[6].sDescription  := txtEvent_CriticalOff;
PlantControl.arrProfileDescription[6].nPrio         := Priority.nCritical;

PlantControl.arrProfileDescription[7].sDescription  := txtEvent_SmokeExtraction_P;
PlantControl.arrProfileDescription[7].nPrio         := Priority.nLifeSafety;

PlantControl.arrProfileDescription[8].sDescription  := txtEvent_LifeSafetyOff;
PlantControl.arrProfileDescription[8].nPrio         := Priority.nLifeSafety;
```

Sample 3: Initializing the parameters for the Auto profile

The first index of the array stands for the profile *nProfile*, the second for the step in the start sequence or for the aggregate.

```
// Profile Auto
// PlantControl.arrProfileParameter[Profile,Step]
// Step 1 = Erc,
// Step 2 = PreHtr
// Step 3 = DamperOuA
// Step 4 = FanSupplyAir
// Step 5 = DamperExhA
// Step 6 = FanExtractAir
// Step 7 = Cooler
// Step 8 = EnablingTemperatureControl

PlantControl.arrProfileParameter[1,1].bValue            := TRUE;
PlantControl.arrProfileParameter[1,1].nValue            := 0;
PlantControl.arrProfileParameter[1,1].fValue            := 0;
PlantControl.arrProfileParameter[1,1].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,1].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,1].bEnFdbUp          := FALSE;
PlantControl.arrProfileParameter[1,1].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,2].bValue            := TRUE;
PlantControl.arrProfileParameter[1,2].nValue            := 0;
PlantControl.arrProfileParameter[1,2].fValue            := 0;
PlantControl.arrProfileParameter[1,2].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,2].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,2].bEnFdbUp          := TRUE;
PlantControl.arrProfileParameter[1,2].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,3].bValue            := TRUE;
PlantControl.arrProfileParameter[1,3].nValue            := 0;
PlantControl.arrProfileParameter[1,3].fValue            := 0;
PlantControl.arrProfileParameter[1,3].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,3].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,3].bEnFdbUp          := TRUE;
PlantControl.arrProfileParameter[1,3].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,4].bValue            := FALSE;
PlantControl.arrProfileParameter[1,4].nValue            := E_BA_Mdlt.eOn;
PlantControl.arrProfileParameter[1,4].fValue            := 0;
PlantControl.arrProfileParameter[1,4].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,4].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,4].bEnFdbUp          := FALSE;
PlantControl.arrProfileParameter[1,4].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,5].bValue            := TRUE;
PlantControl.arrProfileParameter[1,5].nValue            := 0;
PlantControl.arrProfileParameter[1,5].fValue            := 0;
PlantControl.arrProfileParameter[1,5].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,5].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,5].bEnFdbUp          := TRUE;
PlantControl.arrProfileParameter[1,5].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,6].bValue            := FALSE;
PlantControl.arrProfileParameter[1,6].nValue            := E_BA_Mdlt.eOn;
PlantControl.arrProfileParameter[1,6].fValue            := 0;
PlantControl.arrProfileParameter[1,6].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,6].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,6].bEnFdbUp          := FALSE;
PlantControl.arrProfileParameter[1,6].bEnFdbDown        := FALSE;
```

Outputs

Name	Type	Description
nStep	UDINT	The variable indicates which step the sequence is in. *nStep* = 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.
nRemTiUp	UDINT	Countdown switching to the next level [s].
nRemTiDown	UDINT	Countdown switching off the active level [s].

Inputs Outputs

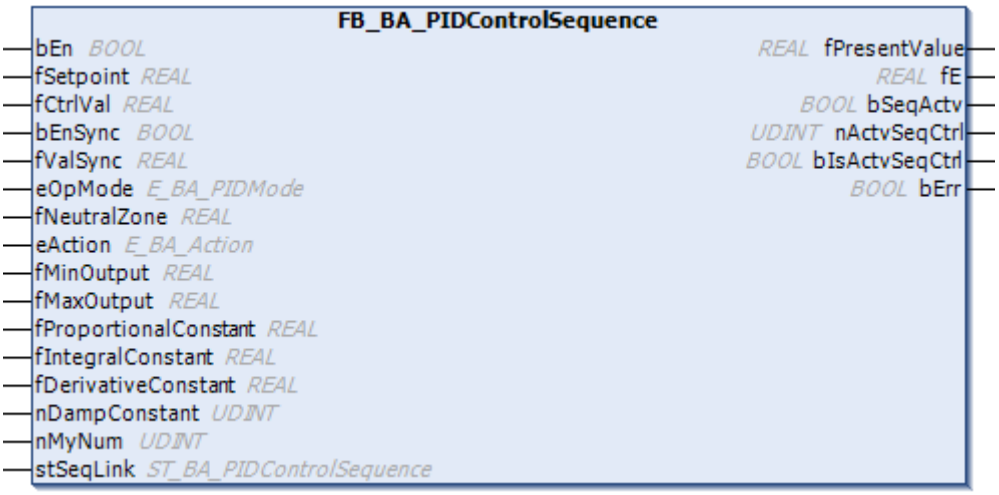
Name	Type	Description
arrAggregates	ST_BA_Aggregate [▶ 278]	The array contains a bidirectional data and command structure for each aggregate. The order of the aggregates in the switching matrix is determined by the connection order of the structures.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.55	Tc3_BA2 from v5.3.19.0

6.1.2.2.3.1.5.9 Sequence

6.1.2.2.3.1.5.9.1 FB_BA_PIDControlSequence



PID sequence controller as part of a sequence.

On receipt of the enable *bEn* = TRUE, the higher-level control block FB_BA_SequenceLinkBase [▶ 171] 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 [▶ 171] takes place via the structure stSeqLink [▶ 123].

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	stSequenceLink.bEnSeqLink	eActionPgm/eActionRm	nActvSeqCtrl/ nMyNum	fPresentValue
TRUE	FALSE	E_BA_Action.eReverse		fMaxOutputPgm / fMaxOutputRm
TRUE	FALSE	E_BA_Action.eDirect		fMinOutputPgm / fMinOutputRm
FALSE	FALSE			0
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl > nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl < nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl < nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl > nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE		nActvSeqCtrl = nMyNum	FB_BA_Loop.fPresentValue

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 [▶ 264]

FB_BA_Loop [▶ 220]

Syntax

```
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	My number in the sequence. This number may only be present once in a sequence control. An internal limitation only allows values from 1 - XBA_Param.nMaxSeqCtrl [▶ 126].

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 *IsActvSeqCtrl* also displays this message.
bErr	BOOL	Indicates that an error has been detected. More detailed information is shown in the properties *ErrorParamMaxSeqCtrl and SeqNumMultiple*. Further explanations on error analysis can be found at Error detection [▶ 487].

/ Inputs Outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [▶ 168] and the function block FB_BA_SequenceLinkBase [▶ 171].

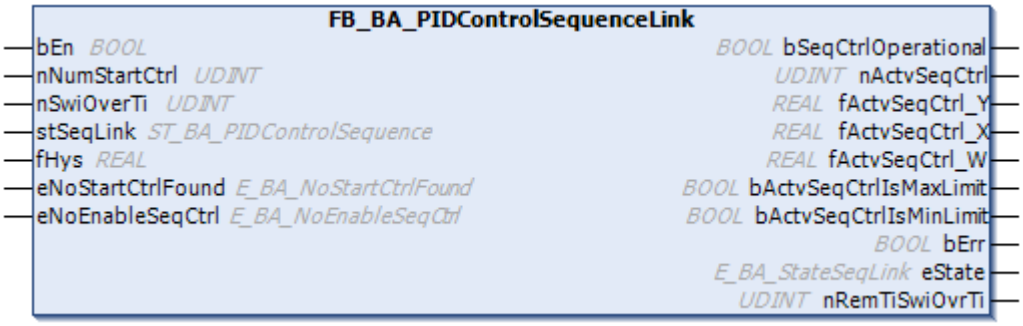
Properties

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 [▶ 284]. 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 v3.1.4024.56	Tc3_BA2 from v5.4.2.0

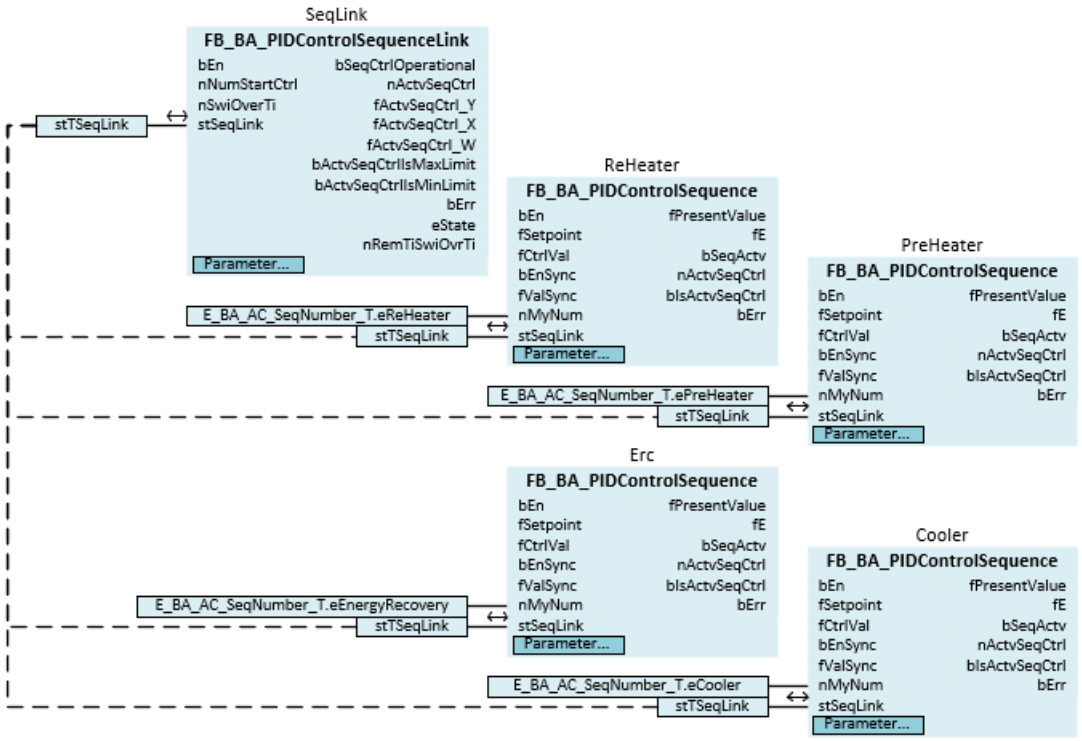
6.1.2.2.3.1.5.9.2 FB_BA_PIDControlSequenceLink



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 [▶ 168] and the *FB_BA_SequenceLinkBase* takes place via the structure stSeqLink [▶ 123].



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 [▶ 168] on the ranking of the active controller.

If the sequence controller specified at *nNumStartCtrl* is not ready for operation, the enum eNoStartCtrlFound 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 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 additionally displays the messages.

Global 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

Syntax

```
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 [▶ 284].
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, *bActvSeqCtrlIsMaxLimit* is not set to FALSE until *fActvSeqCtrl_Y* has fallen below (*fActvSeqCtrl_Y* - *fHys*).
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, *bActvSeqCtrlIsMinLimit* is not set to FALSE until *fActvSeqCtrl_Y* has risen above (*fActvSeqCtrl_Y* + *fHys*).
bErr	BOOL	Indicates that an error has been detected. More detailed information is provided by the property *StateText*. In addition, the enum eState displays further information. Further explanations on error analysis can be found at Error detection [▶ 491].
eState	E_BA_StateSeqLink	The enumeration shows the state of the *FB_BA_SequenceLinkBase* and the sequence control.
nRemTiSwiOvrTi	UDINT	Countdown of switching to the next sequence controller [s].

/ Inputs Outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl [▶ 168] and the function block FB_BA_SequenceLinkBase [▶ 489].

Inputs CONSTANT

Name	Type	Description
fHys	REAL	This hysteresis value sets a hysteresis around the upper and lower output limit *bActvSeqCtrlIsMaxLimit* / *bActvSeqCtrlIsMinLimit* of the active sequence controller. An internal limitation only allows values from 0 - 2.
eNoStartCtrlFound	E_BA_NoStartCtrlFound	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	If the start controller at the input *nNumStartCtrl* 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 *E_BA_Action.eDirect*.
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 *E_BA_Action.eReverse*.
NextActionActvSeqCtrl	UDINT	Get	Depending on the control direction of the active sequence controller *nActvSeqCtrl* 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 *E_BA_NoEnableSeqCtrl.eLastSeqNum* 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 *E_BA_NoEnableSeqCtrl.eFirstSeqNum* 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 *nActvSeqCtrl*.
ActvSeqCtrl_ControlValue	REAL	Get	The property shows the control value of the active sequence controller, see *fActvSeqCtrl_Y*.
ActvSeqCtrl_IsMaxLimit	BOOL	Get	The property indicates that the upper output limit of the active sequence controller has been reached, see *bActvSeqCtrlIsMaxLimit*.
ActvSeqCtrl_IsMinLimit	BOOL	Get	The property indicates that the lower output limit of the active sequence controller has been reached, see *bActvSeqCtrlIsMinLimit*.
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 [▶ 284]. 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 *nNumStartCtrl*, 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.
NextSeqSpIsSmaller	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 an 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.

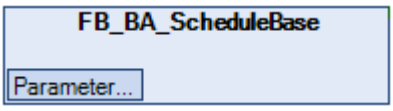
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.56	Tc3_BA2 from v5.4.2.0

6.1.2.2.3.1.5.10 Schedule

6.1.2.2.3.1.5.10.1 Base

6.1.2.2.3.1.5.10.1.1 FB_BA_ScheduleBase



The basic schedule represents the parameterization and evaluation of the derived schedule function blocks FB_BA_ScheduleAnalog [▶ 512], FB_BA_ScheduleBinary [▶ 513] and FB_BA_ScheduleMultistate [▶ 512].

The function block contains the actual schedule entries for 24 exception days and 7 weekdays.

The schedule entries for the 24 exception days are parameterized in the *aException* list.

The schedule entries for the 7 days of the week are parameterized in the *aWeek* list.

The three derived schedule function blocks FB_BA_ScheduleAnalog [▶ 512], FB_BA_ScheduleBinary [▶ 513] and FB_BA_ScheduleMultistate [▶ 512] have identical functionality - they differ only in the data type of the switching values, see example initializations.

Different data types cannot be mixed within a schedule function block.

Determination of the resulting switching command

All schedule entries active at a specific time are summarized in an internal priority matrix. From these schedule entries, the one with the highest switching time determines the switching value at one of the derived function blocks FB_BA_ScheduleAnalog [▶ 512], FB_BA_ScheduleBinary [▶ 513] or FB_BA_ScheduleMultistate [▶ 512]. The schedule entries are evaluated cyclically.

A schedule command is only valid for one day at a time (until midnight). Either a new switching command is defined from midnight or the default value of the derived schedule function blocks is output at the *PresentValue*.

The basic schedule determines the resulting switching command entry *PresentValue* according to the following criteria:

An active exception profile with an active schedule entry has priority over a weekly timer; the weekly timer is then not taken into account.

If an active exception profile does not have a schedule entry, the weekly timer is automatically added. This entry is equivalent to an active exception profile with an active schedule entry.

The weekly timer only comes into play if no exception profile is active and determines the *PresentValue* based on active schedule entries.

If the weekly timer does not have an active schedule entry either, the default value = *PresentValue*.

Samples of configuration in connection with other Schedule function blocks [▶ 498].

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
  {attribute 'parameterCategory':='Value'}
    aWeek         : T_BA_ScheduleWeek;
  {attribute 'parameterCategory':='Value'}
    aException    : T_BA_ScheduleExceptionList;
  {endregion}
END_VAR
```

Name	Type	Description
aWeek	T_BA_ScheduleWeek [▶ 282]	The weekly program *aWeek* contains a daily profile for each day of the week (Monday to Sunday). A profile consists of the following properties: • List of schedule entries Sample declarations can be found attached [▶ 498].
aException	T_BA_ScheduleExceptionList [▶ 283]	The *aException* list contains 24 schedule exception profiles. A profile consists of the following properties: • Period types (Simple date, Date range, Week and day) • Date value • List of schedule entries Sample declarations can be found attached [▶ 498].

Properties

Name	Type	Access	Definition location	Description
Exception	T_BA_ScheduleExceptionList [▶ 283]	Get / Set	Public	The *Exception* property enables read and write access to the exception entries of the schedule.
Week	T_BA_ScheduleWeek [▶ 282]	Get / Set	Public	The *Week* property enables read and write access to the weekly program of the schedule.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.2.0

6.1.Samples for initializations
2.2.
3.1.
5.1
0.1.
1.1

Weekly schedule/week for FB_BA_ScheduleBinary [▶ 513]

From Monday to Sunday, a function is switched on at 6:00. It is switched off every day at 18:00. If this function was switched on externally after 18:00, it is switched off again at 0:00.

```
// Monday
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.bVal           := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour          := 6;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour          := 18;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond        := 0;
// Tuesday
Schedule.aWeek[E_BA_Weekday.eTuesday,1].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].uValue.bVal          := TRUE;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nHour         := 6;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].uValue.bVal          := FALSE;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nHour         := 18;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].uValue.bVal          := FALSE;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nHour         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nSecond       := 0;
// Wednesday
Schedule.aWeek[E_BA_Weekday.eWednesday,1].eState             := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].uValue.bVal        := TRUE;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nHour       := 6;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nMinute     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nSecond     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].eState             := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].uValue.bVal        := FALSE;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nHour       := 18;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nMinute     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nSecond     := 0;
```

```
Schedule.aWeek[E_BA_Weekday.eWednesday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nSecond        := 0;
// Thursday
Schedule.aWeek[E_BA_Weekday.eThursday,1].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,1].uValue.bVal            := TRUE;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nHour           := 6;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,2].uValue.bVal            := FALSE;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nHour           := 18;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,3].uValue.bVal            := FALSE;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nHour           := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nSecond         := 0;
// Friday
Schedule.aWeek[E_BA_Weekday.eFriday,1].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,1].uValue.bVal              := TRUE;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nHour             := 6;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nSecond           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,2].uValue.bVal              := FALSE;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nHour             := 18;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nSecond           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,3].uValue.bVal              := FALSE;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nHour             := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nSecond           := 0;
// Saturday
Schedule.aWeek[E_BA_Weekday.eSaturday,1].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].uValue.bVal            := TRUE;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nHour           := 6;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].uValue.bVal            := FALSE;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nHour           := 18;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].uValue.bVal            := FALSE;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nHour           := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nSecond         := 0;
// Sunday
Schedule.aWeek[E_BA_Weekday.eSunday,1].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,1].uValue.bVal              := TRUE;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nHour             := 6;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nSecond           := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,2].uValue.bVal              := FALSE;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nHour             := 18;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nSecond           := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,3].uValue.bVal              := FALSE;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nHour             := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nSecond           := 0;
```

Weekly schedule for FB_BA_ScheduleMultistate [▶ 512]

From Monday to Sunday, the value 2 is set at 23:00 and 0:00, but the value 1 is set at 5:00.

```
// Monday
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.udiVal            := 2;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour             := 23;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute           := 0;
```

```
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.udiVal           := 2;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.udiVal           := 1;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour            := 5;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond          := 0;
// Tuesday
Schedule.aWeek[E_BA_Weekday.eTuesday,1].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].uValue.udiVal          := 2;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nHour           := 23;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].uValue.udiVal          := 2;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nHour           := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].uValue.udiVal          := 1;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nHour           := 5;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nSecond         := 0;
// Wednesday
Schedule.aWeek[E_BA_Weekday.eWednesday,1].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].uValue.udiVal        := 2;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nHour         := 23;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].uValue.udiVal        := 2;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nHour         := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].uValue.udiVal        := 1;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nHour         := 5;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nSecond       := 0;
// Thursday
Schedule.aWeek[E_BA_Weekday.eThursday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,1].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nHour          := 23;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,2].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,3].uValue.udiVal         := 1;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nHour          := 5;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nSecond        := 0;
// Friday
Schedule.aWeek[E_BA_Weekday.eFriday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,1].uValue.udiVal           := 2;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nHour            := 23;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,2].uValue.udiVal           := 2;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,3].uValue.udiVal           := 1;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nHour            := 5;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nSecond          := 0;
// Saturday
Schedule.aWeek[E_BA_Weekday.eSaturday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nHour          := 23;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nMinute        := 0;
```

```
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].uValue.udiVal         := 1;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nHour          := 5;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nSecond        := 0;
// Sunday
Schedule.aWeek[E_BA_Weekday.eSunday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,1].uValue.udiVal           := 2;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nHour            := 23;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,2].uValue.udiVal           := 2;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,3].uValue.udiVal           := 1;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nHour            := 5;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nSecond          := 0;
```

Combination of the exception in the date range with a weekly schedule for FB_BA_ScheduleAnalog [▶ 512]

This sample shows a combination of a date range in connection with a weekly timer.

In the date range from March 1, 2026 to September 31, 2026, the analog value 14.0 is specified every day at 0:00.

Outside this date range, the weekly schedule is active. From Monday to Friday, the value 22.0 is issued at 6:00, the value 16.0 at 18:00 and the value 14.5 at 0:00.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay01;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 0;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.rVal                   := 14.0;

// Monday
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.rVal             := 22.0;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour            := 6;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.rVal             := 16.0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour            := 18;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.rVal             := 14.5;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond          := 0;
// Tuesday
Schedule.aWeek[E_BA_Weekday.eTuesday,1].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].uValue.rVal            := 22.0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nHour           := 6;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].uValue.rVal            := 16.0;
```

```
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nHour          := 18;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].uValue.rVal           := 14.5;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nSecond        := 0;
// Wednesday
Schedule.aWeek[E_BA_Weekday.eWednesday,1].eState              := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].uValue.rVal         := 22.0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nHour        := 6;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nMinute      := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nSecond      := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].eState              := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].uValue.rVal         := 16.0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nHour        := 18;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nMinute      := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nSecond      := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].eState              := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].uValue.rVal         := 14.5;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nHour        := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nMinute      := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nSecond      := 0;
// Thursday
Schedule.aWeek[E_BA_Weekday.eThursday,1].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,1].uValue.rVal          := 22.0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nHour         := 6;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,2].uValue.rVal          := 16.0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nHour         := 18;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,3].uValue.rVal          := 14.5;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nHour         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nSecond       := 0;
// Friday
Schedule.aWeek[E_BA_Weekday.eFriday,1].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,1].uValue.rVal            := 22.0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nHour           := 6;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,2].uValue.rVal            := 16.0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nHour           := 18;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,3].uValue.rVal            := 14.5;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nHour           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nSecond         := 0;
```

Exception Date

The following samples show possible variants for the *date exception* of a schedule with a binary schedule entry.

Variant 01

The *exception date* is always active. Switching on at 6:00 and switching off at 18:00 every day.

```
Schedule.aException[1].eType                                  := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                     := 255;
Schedule.aException[1].uDate.stDate.eMonth                    := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                      := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek                := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                 := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                  := TRUE;
Schedule.aException[1].aEntry[2].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                 := 18;
```

```
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variant 02

For the entire year 2025, switching on at 6:00 and switching off at 18:00 every day.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 125;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek                 := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                   := TRUE;
Schedule.aException[1].aEntry[2].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                  := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variant 03

In February, switching on at 6:00 and switching off at 18:00 every day.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 255;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek                 := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                   := TRUE;
Schedule.aException[1].aEntry[2].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                  := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variant 04

Switching on at 6:00 and switching off at 18:00 on the 29th of every month.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 255;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek                 := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                   := TRUE;
Schedule.aException[1].aEntry[2].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                  := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variant 05

Switching on at 6:00 and switching off at 18:00 every Monday.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 255;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek                 := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
```

```
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 06

On March 29, switching on at 6:00 and switching off at 18:00.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 255;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 07

In 2025, always switching on at 6:00 and switching off at 18:00 on the 29th of each month.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 125;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 08

The 29th of a month must be a Monday to switch on at 6:00 and to switch off at 18:00.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 255;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 09

In April 2025, switching on at 6:00 and switching off at 18:00 every day.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 125;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eApril;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
```

```
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 10

In April, switching on at 6:00 and switching off at 18:00 every Monday.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                    := 255;
Schedule.aException[1].uDate.stDate.eMonth                   := E_BA_Month.eApril;
Schedule.aException[1].uDate.stDate.nDay                     := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek               := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 11

In 2030, switching on at 6:00 and switching off at 18:00 every Monday.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                    := 130;
Schedule.aException[1].uDate.stDate.eMonth                   := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                     := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek               := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 12

On May 29, 2030, switching on at 6:00 and switching off at 18:00.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                    := 130;
Schedule.aException[1].uDate.stDate.eMonth                   := E_BA_Month.eMay;
Schedule.aException[1].uDate.stDate.nDay                     := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek               := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 13

In 2030, switching on at 6:00 and switching off at 18:00 every Monday in May.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                    := 130;
Schedule.aException[1].uDate.stDate.eMonth                   := E_BA_Month.eMay;
```

```
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 14

In 2030, the 29th of the month must be a Monday to switch on at 6:00 and to switch off at 18:00.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 130;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 15

April 29 must be a Monday to switch on at 6:00 and switch off at 18:00.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 255;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eApril;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 16

February 28, 2025 would have to be a Monday to switch on at 6:00 and to switch off at 18:00. The function block checks whether the switch on condition is correct. Since February 28, 2025 is on a Friday, the weekday *eDayOfWeek* will be corrected to Friday, so that on Friday, February 28, 2025 it will be switched on at 6:00 and switched off at 18:00.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 125;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay28;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Exception Week and Day/WeekNDay

The following samples show possible variants for the *exception week* and *exception day* with a binary schedule entry.

Variant 01

The *WeekNDay* exception is always active. Switching on at 6:00 and switching off at 18:00 every day.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 02

In August, switching on at 6:00 and switching off at 18:00 every day.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.eAugust;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 03

On days 8-14 of each month, switching on at 6:00 and switching off at 18:00.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.eWeek2;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variant 04

Every Monday, switching on at 6:00 and switching off at 18:00.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
```

```
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 05

On days 29-31 in March, switching on at 6:00 and switching off at 18:00.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth               := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth         := E_BA_Week.eWeek5;
Schedule.aException[1].uDate.stWeekNDay.eWeekday             := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 06

On every Monday in March, switching on at 6:00 and switching off at 18:00.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth               := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth         := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday             := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 07

One of the days 29-31 of a month must be a Monday to switch on at 6:00 and to switch off at 18:00.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth               := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth         := E_BA_Week.eWeek5;
Schedule.aException[1].uDate.stWeekNDay.eWeekday             := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                 := FALSE;
```

Variant 08

One of the days 1-7 in March must be a Monday to switch on at 6:00 and to switch off at 18:00.

```
Schedule.aException[1].eType                                 := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth               := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth         := E_BA_Week.eWeek1;
Schedule.aException[1].uDate.stWeekNDay.eWeekday             := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute              := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond              := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                 := TRUE;
Schedule.aException[1].aEntry[2].eState                      := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                := 18;
```

```
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Autocorrection initializations

In the event of incorrect initialization, an autocorrection takes place within the function block. The following samples show possible initializations for autocorrection.

DateRange

Sample 01

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 255;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 255;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.Unspecified;
```

Automatic correction because no specific period is specified.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.Invalid;
```

Sample 02

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 127;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eMonday;
```

Automatic correction, the date ranges are swapped and the weekdays are corrected.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay30;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eWednesday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 127;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eWednesday;
```

Sample 03

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eTuesday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.Unspecified;
```

Automatic correction. The weekdays are adjusted if incorrect.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eMonday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eTuesday;
```

Sample 04

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 222;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := 123;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := 222;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 220;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth        := 121;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay          := 145;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek    := 127;
```

Automatic correction, the highest valid entries are used for year, month and day - 31.12.2100. The day of the week is adjusted to the resulting date.

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 200;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := E_BA_Month.eDecember;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eFriday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 200;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth        := E_BA_Month.eDecember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay          := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek    := E_BA_WeekDay.eFriday;
```

Sample 05

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 124;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth        := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay          := E_BA_Day.eDay30;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek    := E_BA_WeekDay.Unspecified;
```

Automatic correction, the days of the month are adjusted. The day of the week is adjusted to the resulting date.

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 124;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eThursday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth        := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay          := E_BA_Day.eDay28;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek    := E_BA_WeekDay.eSaturday;
```

Sample 06

```
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                   := E_BA_SchedEntryState.Invalid;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.bVal              := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour             := 30;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute           := 120;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond           := 124;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHundredths       := 124;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                   := E_BA_SchedEntryState.eUndefined;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.bVal              := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour             := 11;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute           := 12;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond           := 13;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHundredths       := 14;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                   := E_BA_SchedEntryState.eNull;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.bVal              := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour             := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute           := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond           := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHundredths       := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,4].uValue.bVal              := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHour             := 99;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nMinute           := 111;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nSecond           := 222;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHundredths       := 100;
```

Automatic correction, sorting in the weekly schedule according to the order shown in the entries: *E_BA_SchedEntryState.eValue* > *E_BA_SchedEntryState.eNull* > *E_BA_SchedEntryState.eUndefined*. E_BA_SchedEntryState.Invalid becomes *E_BA_SchedEntryState.eUndefined*.

The times will be adjusted.

```
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.bVal               := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour              := 23;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute            := 59;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond            := 59;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHundredths        := 99;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                    := E_BA_SchedEntryState.eNull;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.bVal               := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour              := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute            := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond            := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHundredths        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                    := E_BA_SchedEntryState.eUndefined;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.bVal               := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour              := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute            := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond            := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHundredths        := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].eState                    := E_BA_SchedEntryState.eUndefined;
Schedule.aWeek[E_BA_Weekday.eMonday,4].uValue.bVal               := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHour              := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nMinute            := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nSecond            := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHundredths        := 255;
```

Sample 07

```
Schedule.aException[1].eType                                     := E_BA_DateValChoice.Invalid;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear        := 127;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth       := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek   := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear          := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth         := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay           := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek     := E_BA_WeekDay.eMonday;
```

Automatic correction, the date range is swapped and the days of the week are corrected.

```
Schedule.aException[1].eType                                     := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay         := E_BA_Day.eDay30;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek   := E_BA_WeekDay.eWednesday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear          := 127;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth         := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay           := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek     := E_BA_WeekDay.eWednesday;
```

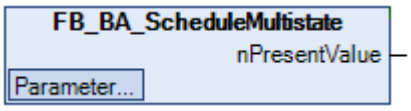
Sample 08

```
Schedule.aException[1].eType                                     := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear        := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth       := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek   := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear          := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth         := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay           := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek     := E_BA_WeekDay.Unspecified;
```

Automatic correction, the days of the week are adjusted

```
Schedule.aException[1].eType                                     := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear        := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth       := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek   := E_BA_WeekDay.eMonday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear          := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth         := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay           := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek     := E_BA_WeekDay.eTuesday;
```

6.1.2.2.3.1.5.10.2 FB_BA_ScheduleMultistate



The function block FB_BA_ScheduleMultistate represents a numerical schedule. It consists essentially of the base class FB_BA_ScheduleBase [▶ 496].

The function block uses the base class to determine the numerical output value. If the output value *nPresentValue* is not determined within the base class, *nDefaultValue* = *nPresentValue*.

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    {attribute 'parameterCategory':='Value'}
      nDefaultValue    : UDINT;
  {endregion}
END_VAR
```

Name	Type	Description
nDefaultValue	UDINT	The default value *nDefaultValue* is active in the following cases: *nPresentValue* = *nDefaultValue* if the output value *nPresentValue* is not determined within the base class FB_BA_ScheduleBase [▶ 496].

Outputs

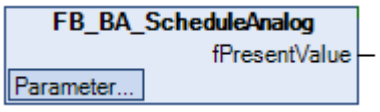
```
VAR_OUTPUT
  nPresentValue    : UDINT;
END_VAR
```

Name	Type	Description
nPresentValue	UDINT	Current value It consists essentially of the base class FB_BA_ScheduleBase [▶ 496]. The numerical output value for *nPresentValue* is determined within the base class. If the output value *nPresentValue* is not determined within the base class, *nDefaultValue* = *nPresentValue*.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.1.5.10.3 FB_BA_ScheduleAnalog



The function block FB_BA_ScheduleAnalog represents an analog schedule. It consists essentially of the base class FB_BA_ScheduleBase [▶ 496].

The function block uses the base class to determine the analog output value. If the output value *fPresentValue* is not determined within the base class, *fDefaultValue* = *fPresentValue*.

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    {attribute 'parameterCategory':='Value'}
      fDefaultValue    : REAL;
  {endregion}
END_VAR
```

Name	Type	Description
fDefaultValue	REAL	The default value *fDefaultValue* is active in the following cases: *fPresentValue* = *fDefaultValue* if the output value *fPresentValue* is not determined within the base class FB_BA_ScheduleBase [▶ 496].

Outputs

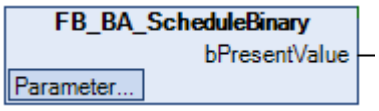
```
VAR_OUTPUT
  fPresentValue    : REAL;
END_VAR
```

Name	Type	Description
fPresentValue	REAL	Current value It consists essentially of the base class FB_BA_ScheduleBase [▶ 496]. The numerical output value for *fPresentValue* is determined within the base class. If the output value *fPresentValue* is not determined within the base class, *fDefaultValue* = *fPresentValue*.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.1.5.10.4 FB_BA_ScheduleBinary



The function block FB_BA_ScheduleBinary represents a binary schedule. It consists essentially of the base class FB_BA_ScheduleBase [▶ 496].

The function block uses the base class to determine the binary output value. If the output value *bPresentValue* is not determined within the base class, *bDefaultValue* = *bPresentValue*.

Inputs CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    {attribute 'parameterCategory':='Value'}
      bDefaultValue    : BOOL;
  {endregion}
END_VAR
```

Name	Type	Description
bDefaultValue	BOOL	The default value *bDefaultValue* is active in the following cases: *bPresentValue* = *bDefaultValue* if the output value *bPresentValue* is not determined within the base class FB_BA_ScheduleBase [▶ 496].

Outputs

```
VAR_OUTPUT
  bPresentValue    : BOOL;
END_VAR
```

Name	Type	Description
bPresentValue	BOOL	Current value It consists essentially of the base class FB_BA_ScheduleBase [▶ 496]. The numerical output value for *bPresentValue* is determined within the base class. If the output value *bPresentValue* is not determined within the base class, *bDefaultValue* = *bPresentValue*.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.2.0

6.1.2.2.3.2 Functions

6.1.2.2.3.2.1 Universal

6.1.2.2.3.2.1.1 Priority

6.1.2.2.3.2.1.1.1 F_BA_BinaryCrit

The function of return type ST_BA_Binary [▶ 277] enables writing to the priority "Critical" of the return type of the function.

In the function table you can see which states are output at the return type of the function.

bEnCrit	F_BA_BinarySfty-Crit.bEnSfty	F_BA_BinarySfty-Crit.bValSfty	F_BA_BinarySfty-Crit.bEnCrit	F_BA_BinarySfty-Crit.bValCrit	F_BA_BinarySfty-Crit.bEnPgm	F_BA_BinarySfty-Crit.bValPgm
FALSE	stCmd.bEnSfty	stCmd.bValSfty	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
TRUE	stCmd.bEnSfty	stCmd.bValSfty	**bEnCrit**	**bValCrit**	stCmd.bEnPgm	stCmd.bValPgm

Syntax

```
FUNCTION F_BA_BinaryCrit  : ST_BA_Binary
VAR_INPUT
  bEnCrit      : BOOL;
```

```
  bValCrit      : BOOL;
  stCmd         : ST_BA_Binary;
END_VAR
```

Inputs

Name	Type	Description
bEnCrit	BOOL	Enable for writing the priority "Critical" of the return type ST_BA_Binary [▶ 277].
bValCrit	BOOL	Value of the priority "Critical" of the return type ST_BA_Binary [▶ 277] to be written if *bEnCrit* has the value TRUE.
stCmd	ST_BA_Binary [▶ 277]	The command structure *stCmd* transmits the enables and switching values of the priorities to the connected aggregate.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.2.1.1.2 F_BA_BinarySfty

The function of return type ST_BA_Binary [▶ 277] enables writing to the priority "Safety" of the return type of the function.

In the function table you can see which states are output at the return type of the function.

bEnSfty	F_BA_BinarySfty-Crit.bEnSfty	F_BA_BinarySfty-Crit.bValSfty	F_BA_BinarySfty-Crit.bEnCrit	F_BA_BinarySfty-Crit.bValCrit	F_BA_BinarySfty-Crit.bEnPgm	F_BA_BinarySfty-Crit.bValPgm
FALSE	stCmd.bEnSfty	stCmd.bValSfty	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
TRUE	**bEnSfty**	**bValSfty**	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm

Syntax

```
FUNCTION F_BA_BinarySfty : ST_BA_Binary
VAR_INPUT
  bEnSfty       : BOOL;
  bValSfty      : BOOL;
  stCmd         : ST_BA_Binary;
END_VAR
```

Inputs

Name	Type	Description
bEnSfty	BOOL	Enable for writing the priority "Safety" of the return type ST_BA_Binary [▶ 277].
bValSfty	BOOL	Value of the priority "Safety" of the ST_BA_Binary [▶ 277] return type to be written if *bEnCrit* has the value TRUE.
stCmd	ST_BA_Binary [▶ 277]	The command structure *stCmd* transmits the enables and switching values of the priorities to the connected aggregate.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.2.1.1.3 F_BA_BinarySftyCrit

The function of return type ST_BA_Binary [▶ 277] enables writing to the priorities "Safety" and "Critical" of the return type of the function.

In the function table you can see which states are output at the return type of the function.

bEnSfty	bEnCrit	F_BA_BinarySfty-Crit.bEnSfty	F_BA_BinarySfty-Crit.bValSfty	F_BA_BinarySfty-Crit.bEnCrit	F_BA_BinarySfty-Crit.bValCrit	F_BA_BinarySfty-Crit.bEnPgm	F_BA_BinarySfty-Crit.bValPgm
FALSE	FALSE	stCmd.bEnSfty	stCmd.bValSfty	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
TRUE	FALSE	**bEnSfty**	**bValSfty**	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
FALSE	**TRUE**	stCmd.bEnSfty	stCmd.bValSfty	**bEnCrit**	**bValCrit**	stCmd.bEnPgm	stCmd.bValPgm
TRUE	**TRUE**	**bEnSfty**	**bValSfty**	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm

Syntax

```
FUNCTION F_BA_BinarySftyCrit : ST_BA_Binary
VAR_INPUT
  bEnSfty      : BOOL;
  bValSfty     : BOOL;
  bEnCrit      : BOOL;
  bValCrit     : BOOL;
  stCmd        : ST_BA_Binary;
END_VAR
```

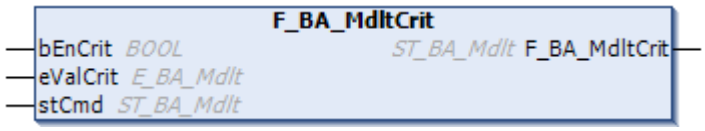
Inputs

Name	Type	Description
bEnSfty	BOOL	Enable for writing the priority "Safety" of the return type ST_BA_Binary [▶ 277].
bValSfty	BOOL	Value of the priority "Safety" of the ST_BA_Binary [▶ 277] return type to be written if *bEnCrit* has the value TRUE.
bEnCrit	BOOL	Enable for writing the priority "Critical" of the return type ST_BA_Binary [▶ 277].
bValCrit	BOOL	Value of the priority "Critical" of the return type ST_BA_Binary [▶ 277] to be written if *bEnCrit* has the value TRUE.
stCmd	ST_BA_Binary [▶ 277]	The command structure *stCmd* transmits the enables and switching values of the priorities to the connected aggregate.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.2.1.1.4 F_BA_MdltCrit



The function of return type ST_BA_Mdlt [▶ 276] enables writing to the priority "Critical" of the return type of the function.

In the function table you can see which states are output at the return type of the function.

bEnCrit	F_BA_MdltCrit.bEnSfty	F_BA_MdltCrit.eValSfty	F_BA_MdltCrit.bEnCrit	F_BA_MdltCrit.eValCrit	F_BA_MdltCrit.bEnPgm	F_BA_MdltCrit.eValPgm
FALSE	stCmd.bEnSfty	stCmd.eValSfty	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
TRUE	stCmd.bEnSfty	stCmd.eValSfty	**bEnCrit**	**eValCrit**	stCmd.bEnPgm	stCmd.eValPgm

Syntax

```
FUNCTION F_BA_MdltCrit : ST_BA_Mdlt
VAR_INPUT
  bEnCrit      : BOOL;
  eValCrit     : E_BA_Mdlt;
  stCmd        : ST_BA_Mdlt;
END_VAR
```

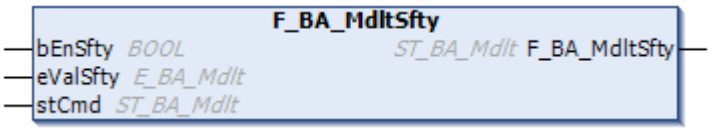
Inputs

Name	Type	Description
bEnCrit	BOOL	Enable for writing the "Critical" priority of the return type ST_BA_Mdlt [▶ 276].
eValCrit	E_BA_Mdlt [▶ 270]	Value of the priority "Critical" of the return type ST_BA_Mdlt [▶ 276] to be written if *bEnCrit* has the value TRUE.
stCmd	ST_BA_Mdlt [▶ 276]	The command structure *stCmd* is used to control modulating aggregates.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.2.1.1.5 F_BA_MdltSfty



The function of return type ST_BA_Mdlt [▶ 276] enables writing to the priority "Safety" of the return type of the function.

In the function table you can see which states are output at the return type of the function.

bEnSfty	F_BA_MdltSfty.bEnSfty	F_BA_MdltSfty.eValSfty	F_BA_MdltSfty.bEnCrit	F_BA_MdltSfty.eValCrit	F_BA_MdltSfty.bEnPgm	F_BA_MdltSfty.eValPgm
FALSE	stCmd.bEnSfty	stCmd.eValSfty	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
TRUE	**bEnSfty**	**eValSfty**	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm

Syntax

```
FUNCTION F_BA_MdltSfty : ST_BA_Mdlt
VAR_INPUT
  bEnSfty      : BOOL;
  eValSfty     : E_BA_Mdlt;
  stCmd        : ST_BA_Mdlt;
END_VAR
```

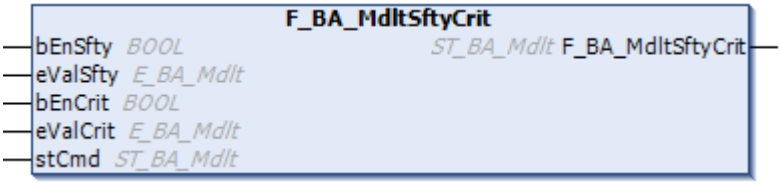
Inputs

Name	Type	Description
bEnSfty	BOOL	Enable for writing the priority "Safety" of the return type ST_BA_Mdlt [▶ 276].
eValSfty	E_BA_Mdlt [▶ 270]	Value of the priority "Safety" of the return type ST_BA_Mdlt [▶ 276] to be written if *bEnCrit* has the value TRUE.
stCmd	ST_BA_Mdlt [▶ 276]	The command structure *stCmd* is used to control modulating aggregates.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.2.1.1.6 F_BA_MdltSftyCrit



The function of return type ST_BA_Mdlt [▶ 276] enables writing to the priorities "Safety" and "Critical" of the return type of the function.

In the function table you can see which states are output at the return type of the function.

bEnSfty	bEnCrit	F_BA_MdltCrit.bEnSfty	F_BA_MdltCrit.eValSfty	F_BA_MdltCrit.bEnCrit	F_BA_MdltCrit.eValCrit	F_BA_MdltCrit.bEnPgm	F_BA_MdltCrit.eValPgm
FALSE	FALSE	stCmd.bEnSfty	stCmd.eValSfty	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
FALSE	**TRUE**	stCmd.bEnSfty	stCmd.eValSfty	**bEnCrit**	**eValCrit**	stCmd.bEnPgm	stCmd.eValPgm
TRUE	FALSE	**bEnSfty**	**eValSfty**	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
TRUE	**TRUE**	**bEnSfty**	**eValSfty**	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm

Syntax

```
FUNCTION F_BA_MdltSfty : ST_BA_Mdlt
VAR_INPUT
  bEnSfty      : BOOL;
  eValSfty     : E_BA_Mdlt;
  bEnCrit      : BOOL;
  eValCrit     : E_BA_Mdlt;
  stCmd        : ST_BA_Mdlt;
END_VAR
```

Inputs

Name	Type	Description
bEnSfty	BOOL	Enable for writing the priority "Safety" of the return type ST_BA_Mdlt [▶ 276].
eValSfty	E_BA_Mdlt [▶ 270]	Value of the priority "Safety" of the return type ST_BA_Mdlt [▶ 276] to be written if *bEnCrit* has the value TRUE.
bEnCrit	BOOL	Enable for writing the "Critical" priority of the return type ST_BA_Mdlt [▶ 276].
eValCrit	E_BA_Mdlt [▶ 270]	Value of the priority "Critical" of the return type ST_BA_Mdlt [▶ 276] to be written if *bEnCrit* has the value TRUE.
stCmd	ST_BA_Mdlt [▶ 276]	The command structure *stCmd* is used to control modulating aggregates.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.22	Tc3_BA2 from v5.2.5.0

6.1.2.2.3.2.2 Conversion

6.1.2.2.3.2.2.1 Temperature conversion function

Temperature conversion functions between Kelvin, Celsius, Réaumur and Fahrenheit.

Overview

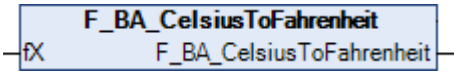
	Kelvin [K]	Degrees Celsius [°C]	Degrees Réaumur [°R]	Degrees Fahrenheit [°F]
Absolute zero	0	-273.15	-218.52	-459.67
Melting point	273.15	0	0	32
Boiling point	373.15	100	80	212

(The melting and boiling points refer to pure water)

Conversion rules

	Kelvin [K]	Degrees Celsius [°C]	Degrees Réaumur [°R]	Degrees Fahrenheit [°F]
X = Kelvin [K]	-	= X − 273.15 °C	= 4/5(X − 273.15) [°R]	-459.67
Degrees Celsius [°C]	= x + 273.15 K	-	0	32
Degrees Réaumur [°R]		100	-	212
Degrees Fahrenheit [°F]				-

6.1.2.2.3.2.2.2 F_BA_CelsiusToFahrenheit



The function F_BA_CelsiusToFahrenheit of return type LREAL is used to convert a temperature value in degrees Celsius to a temperature value in degrees Fahrenheit.

Syntax

```
FUNCTION F_BA_CelsiusToFahrenheit : LREAL
VAR_INPUT
  fX    : LREAL;
END_VAR
```

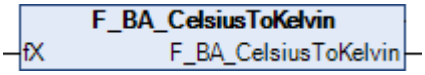
Inputs

Name	Type	Description
fX	LREAL	Input value in degrees Celsius.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.3 F_BA_CelsiusToKelvin



The function F_BA_CelsiusToKelvin of the return type REAL is used to convert a temperature value in degrees Celsius to a temperature value in Kelvin.

Syntax

```
FUNCTION F_BA_CelsiusToKelvin : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

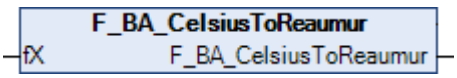
Inputs

Name	Type	Description
fX	REAL	Input value in degrees Celsius.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.4 F_BA_CelsiusToReaumur



The function F_BA_CelsiusToReaumur of return type REAL is used to convert a temperature value in degrees Celsius to a temperature value in degrees Réaumur.

Syntax

```
FUNCTION F_BA_CelsiusToReaumur : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

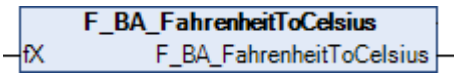
Inputs

Name	Type	Description
fX	REAL	Input value in degrees Celsius.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.5 F_BA_FahrenheitToCelsius



The function F_BA_FahrenheitToCelsius of return type REAL is used to convert a temperature value in degrees Fahrenheit to a temperature value in degrees Celsius.

Syntax

```
FUNCTION F_BA_FahrenheitToCelsius : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

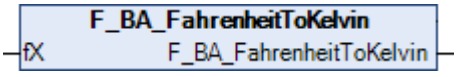
Inputs

Name	Type	Description
fX	REAL	Input value in degrees Fahrenheit.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.6 F_BA_FahrenheitToKelvin



The function F_BA_FahrenheitToKelvin of the return type REAL is used to convert a temperature value in degrees Fahrenheit to a temperature value in Kelvin.

Syntax

```
FUNCTION F_BA_FahrenheitToKelvin : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

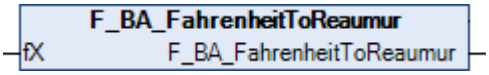
Inputs

Name	Type	Description
fX	REAL	Input value in degrees Fahrenheit.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.7 F_BA_FahrenheitToReaumur



The function F_BA_FahrenheitToReaumur of return type REAL is used to convert a temperature value in degrees Fahrenheit to a temperature value in degrees Réaumur.

Syntax

```
FUNCTION F_BA_FahrenheitToReaumur : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

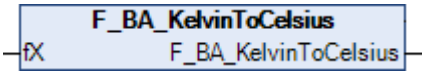
Inputs

Name	Type	Description
fX	REAL	Input value in degrees Fahrenheit.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.8 F_BA_KelvinToCelsius



The function F_BA_KelvinToCelsius of return type REAL is used to convert a temperature value in Kelvin to a temperature value in degrees Celsius.

Syntax

```
FUNCTION F_BA_KelvinToCelsius : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

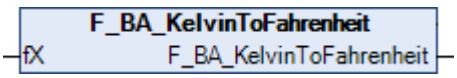
Inputs

Name	Type	Description
fX	REAL	Input value in Kelvin.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.9 F_BA_KelvinToFahrenheit



The function F_BA_KelvinToFahrenheit of the return type REAL is used to convert a temperature value in Kelvin to a temperature value in degrees Fahrenheit.

Syntax

```
FUNCTION F_BA_KelvinToFahrenheit : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

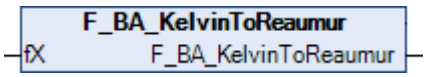
Inputs

Name	Type	Description
fX	REAL	Input value in Kelvin.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.10 F_BA_KelvinToReaumur



The function F_BA_KelvinToReaumur of return type REAL is used to convert a temperature value in Kelvin to a temperature variable in degrees Réaumur.

Syntax

```
FUNCTION F_BA_KelvinToReaumur : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

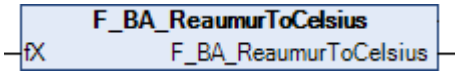
Inputs

Name	Type	Description
fX	REAL	Input value in Kelvin.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.11 F_BA_ReaumurToCelsius



The function F_BA_ReaumurToCelsius of return type REAL is used to convert a temperature value in degrees Réaumur to a temperature value in degrees Celsius.

Syntax

```
FUNCTION F_BA_ReaumurToCelsius : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

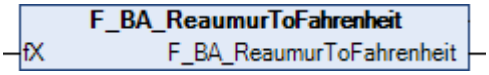
Inputs

Name	Type	Description
Fx	REAL	Input value in degrees Réaumur.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.12 F_BA_ReaumurToFahrenheit



The function F_BA_ReaumurToFahrenheit of the return type REAL is used to convert a temperature value in degrees Réaumur to a temperature value in degrees Fahrenheit.

Syntax

```
FUNCTION F_BA_ReaumurToFahrenheit : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

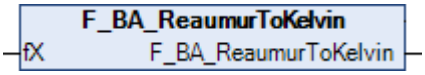
Inputs

Name	Type	Description
Fx	REAL	Input value in degrees Réaumur.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

6.1.2.2.3.2.2.13 F_BA_ReaumurToKelvin



The function F_BA_ReaumurToKelvin of the return type REAL is used to convert a temperature value in degrees Réaumur to a temperature value in Kelvin.

Syntax

```
FUNCTION F_BA_ReaumurToKelvin : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

Inputs

Name	Type	Description
fX	REAL	Input value in degrees Réaumur.

Prerequisites

Development environment	Required PLC library
TwinCAT from v3.1.4024.62	Tc3_BA2 from v5.5.14.0

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.

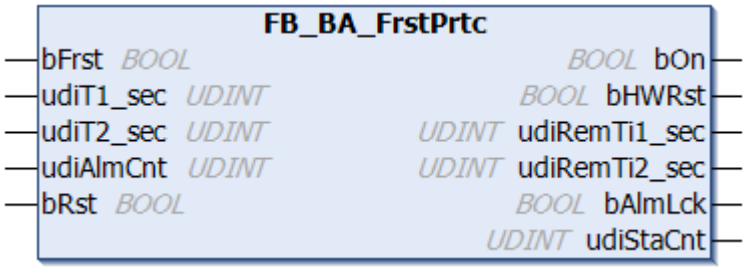
6.1.2.3.2.1 POUs

6.1.2.3.2.1.1 Air conditioning equipment

Function blocks

Name	Description
FB_BA_FrstPrtc [▶ 525]	Monitoring of frost alarm and emergency heating
FB_BA_HX [▶ 527]	Calculation of dew point temperature, specific enthalpy and absolute humidity
FB_BA_NgtCol [▶ 528]	Summer night cooling
FB_BA_RcvMonit [▶ 529]	Function block for calculating the efficiency of an energy recovery system
FB_BA_SPSupvis [▶ 532]	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*.

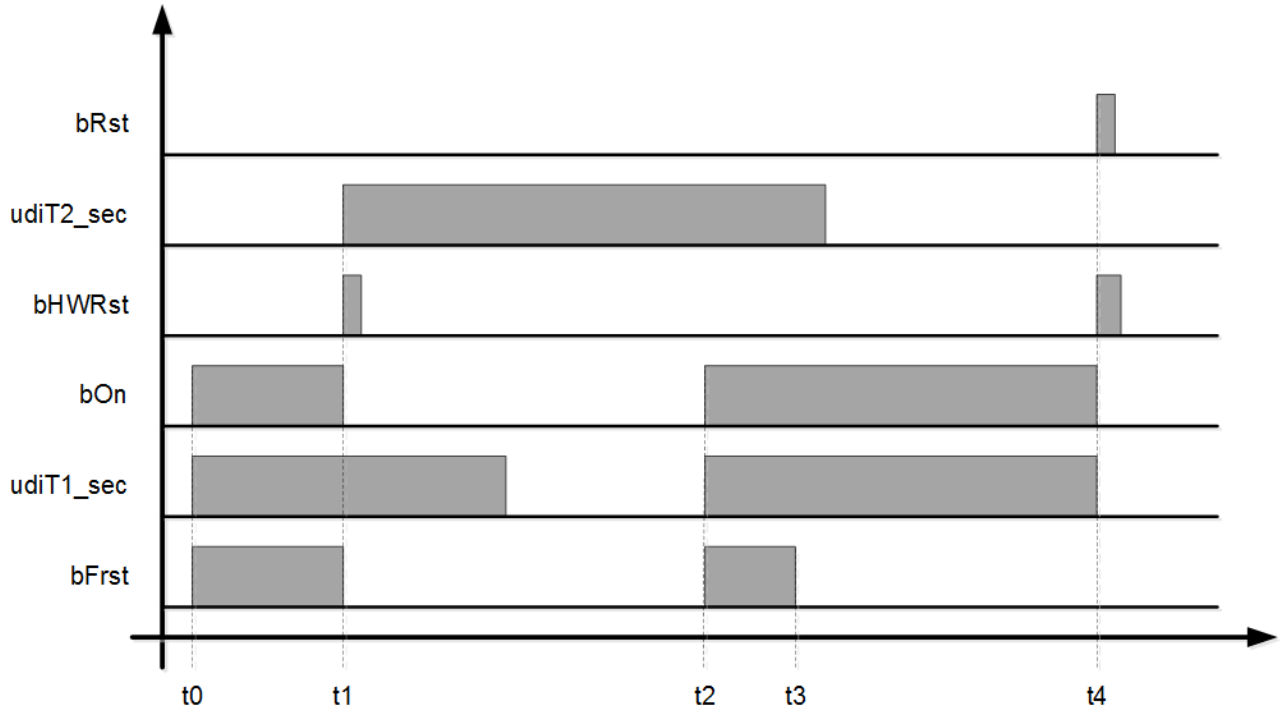
If the frost alarm ceases due to activation of the heating coil within the time *udiT1_sec* (*bFrst*=FALSE), the plant automatically restarts. For the plant restart *bOn* becomes FALSE, and at output *bHWRst* a pulse for acknowledgement of a latching circuit in the control cabinet is issued. With the 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.

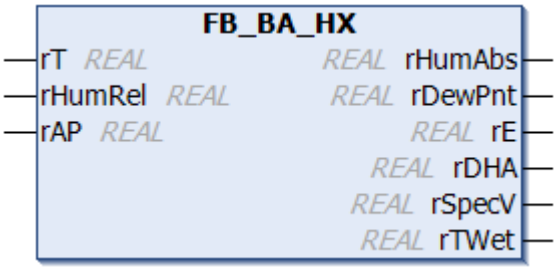
bAlmLck: Alarm lock - stored alarm.

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].

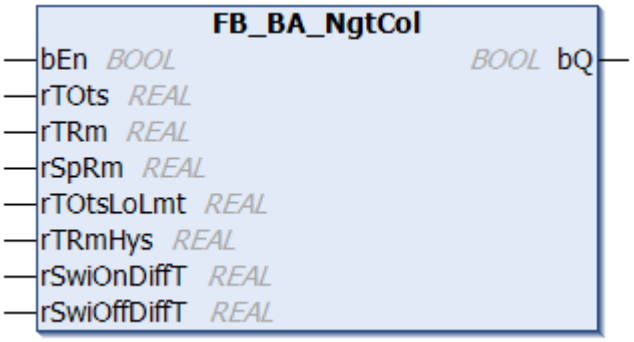
lrSpecV: Specific volume [m³/kg].

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≤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

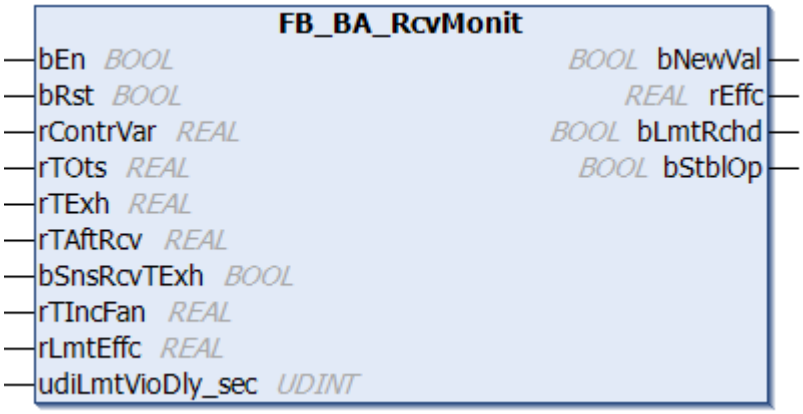
```
bQ  : BOOL;
```

bQ: Summer night cooling on.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

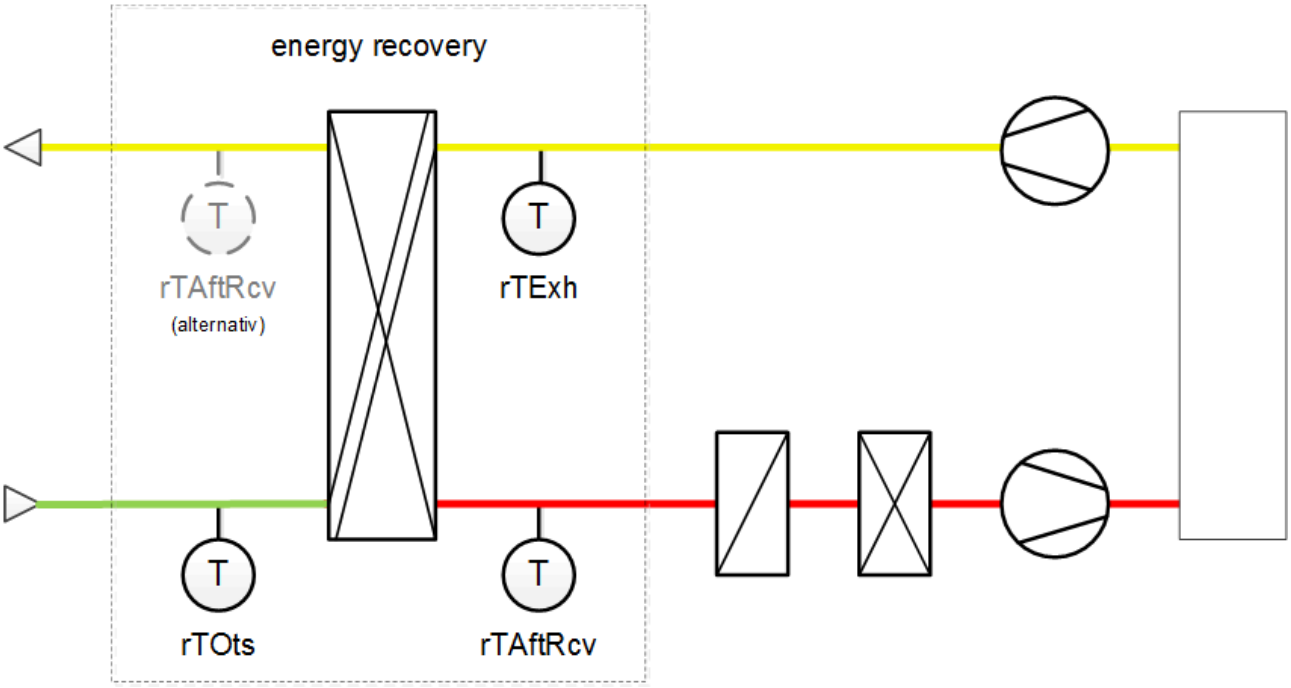
6.1.2.3.2.1.1.4 FB_BA_RcvMonit



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 *bStblOp*, and the calculated efficiency is output at *rEffc*. If the state is not "stable", a FALSE signal appears at output *bStblOp*, 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 (*bStblOp*=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.

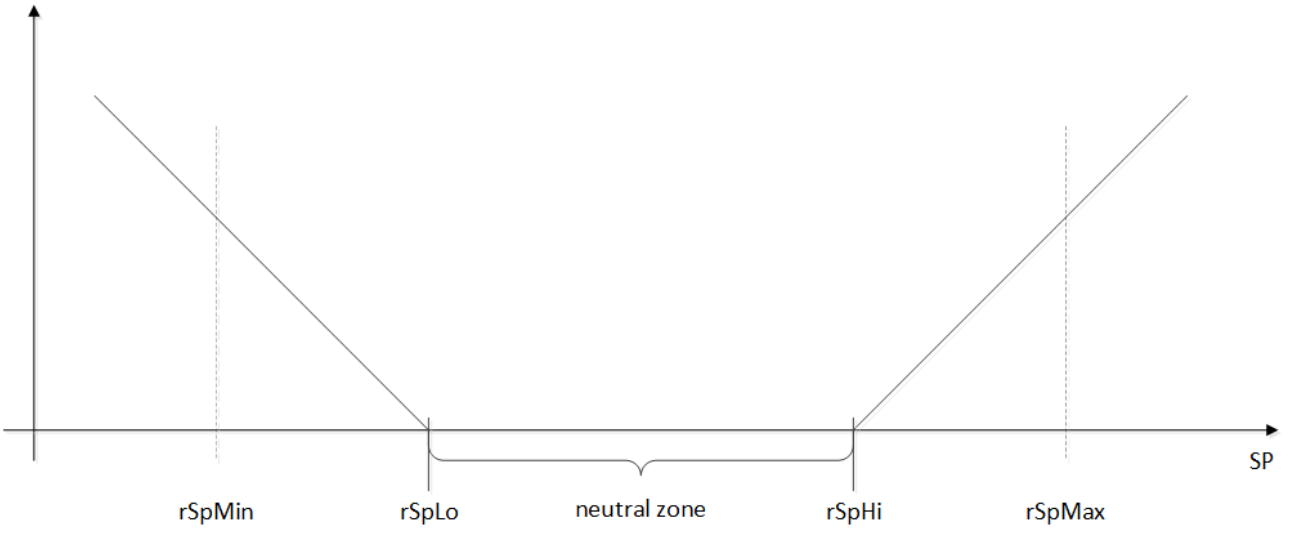
Checking and limitation of the setpoints

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 rSpLo and rSpHi are used
rSpMin >= rSpMax	last valid values of rSpMin and rSpMax are used
rSpHi > rSpMax	rPrSpHi = rSpMax
rSpLo < rSpMin	rPrSpLo = rSpMin

Checking	bErr	Action
rSpMin >= rSpMax	TRUE	rSpErr = ((rSpMin + rSpMax) / 2) rPrSpHi = rPrSpLo = rPrRcv = rSpErr
rSpHi < rSpMin		
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

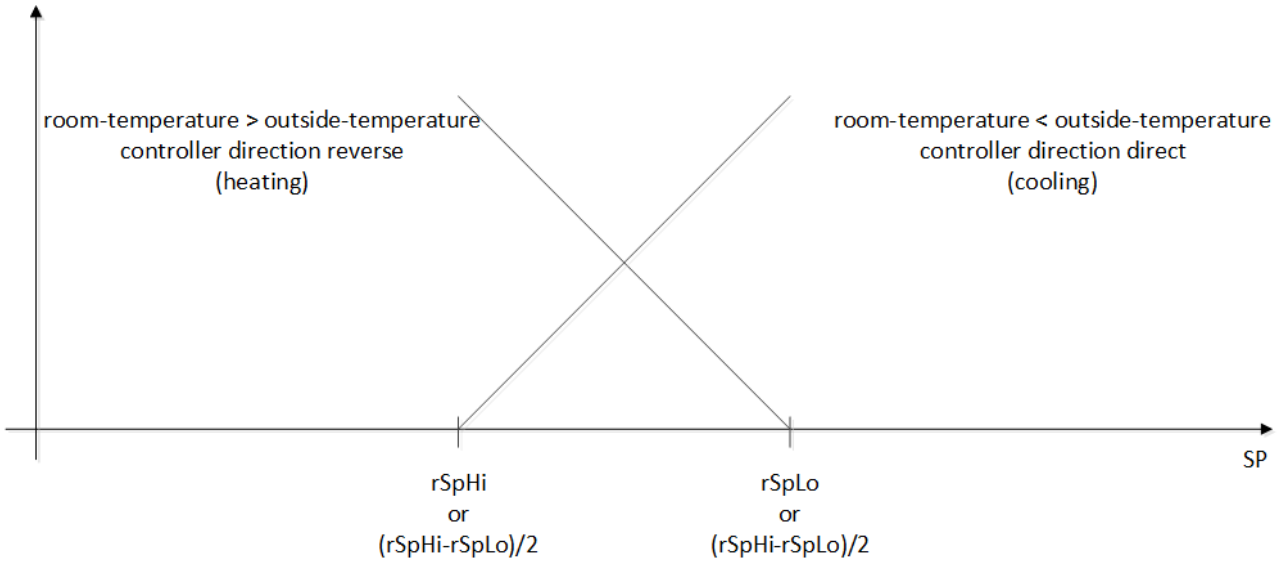
For heat recovery, the setpoint *rSpRcv* is optionally calculated from the mean value of the upper and lower setpoint, *rSpHi* and *rSpLo,* or depending on the control direction of the heat recovery system.
The method is defined through the input variable *bSlcnSpRcv*:

b SlcnSpRcv	rSpRcv
TRUE	Mean value of *rSpLo* and *rSpHi*
FALSE	Depends on direction of action, defined through input bActRcv

If the setpoint is defined depending on the direction of action, the following applies:

bActRcv	Control direction	rSpRcv
TRUE	direct (cooling)	*rSpHi*
FALSE	indirect (heating)	*rSpLo*

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

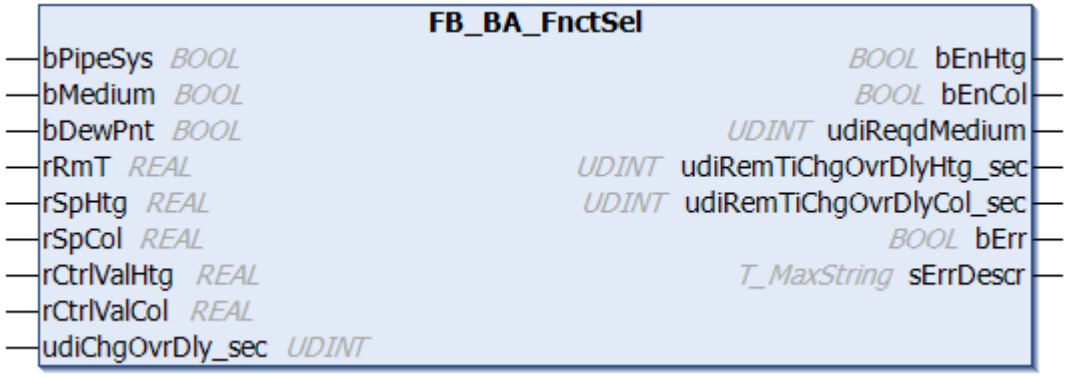
6.1.2.3.2.1.2 Room automation

6.1.2.3.2.1.2.1 Heating, cooling

Function blocks

Name	Description
FB_BA_FnctSel [▶ 534]	Function selection (heating and/or cooling) in two- or four-pipe network.
FB_BA_RmTAdj [▶ 537]	Adjustment of the room temperature setpoint.
FB_BA_SpRmT [▶ 540]	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 (*rCtrlValCol*)	Heating controller output = 0 (*rCtrlValHtg*)
Room temperature (*rRmT*) < heating setpoint (*rSpHtg*)	Room temperature (*rRmT*) > cooling setpoint (*rSpCol*)
Cooling controller enable (*bEnCol*) is FALSE over at least the changeover time *udiChgOvrDly_sec* [s]	Heating controller enable (*bEnHtg*) is FALSE over at least the changeover time *udiChgOvrDly_sec* [s]
Four-pipe system is selected (*bPipesys*=TRUE) OR two-pipe system is selected and heating medium is available (*bPipeSys*=FALSE AND *bMedium*=FALSE)	Four-pipe system is selected (*bPipesys*=TRUE) OR two-pipe system is selected and cooling medium is available (*bPipeSys*=FALSE AND *bMedium*=TRUE)
	The dew-point monitor does not respond (*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 (*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).

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*

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.

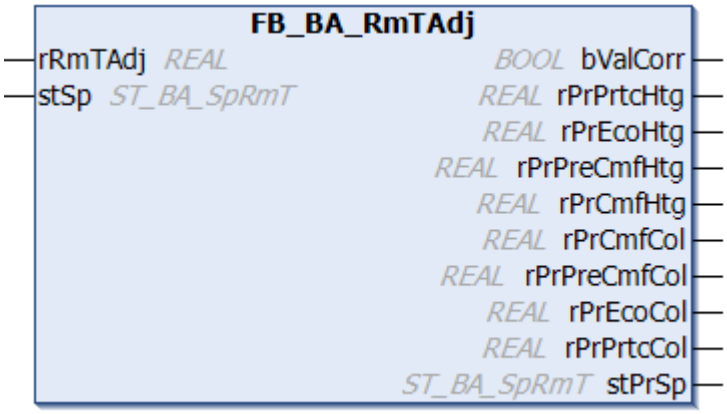
sErrDescr: Contains the error description.

Error description
01: Warning: The heating setpoint is higher than or equal to the cooling setpoint

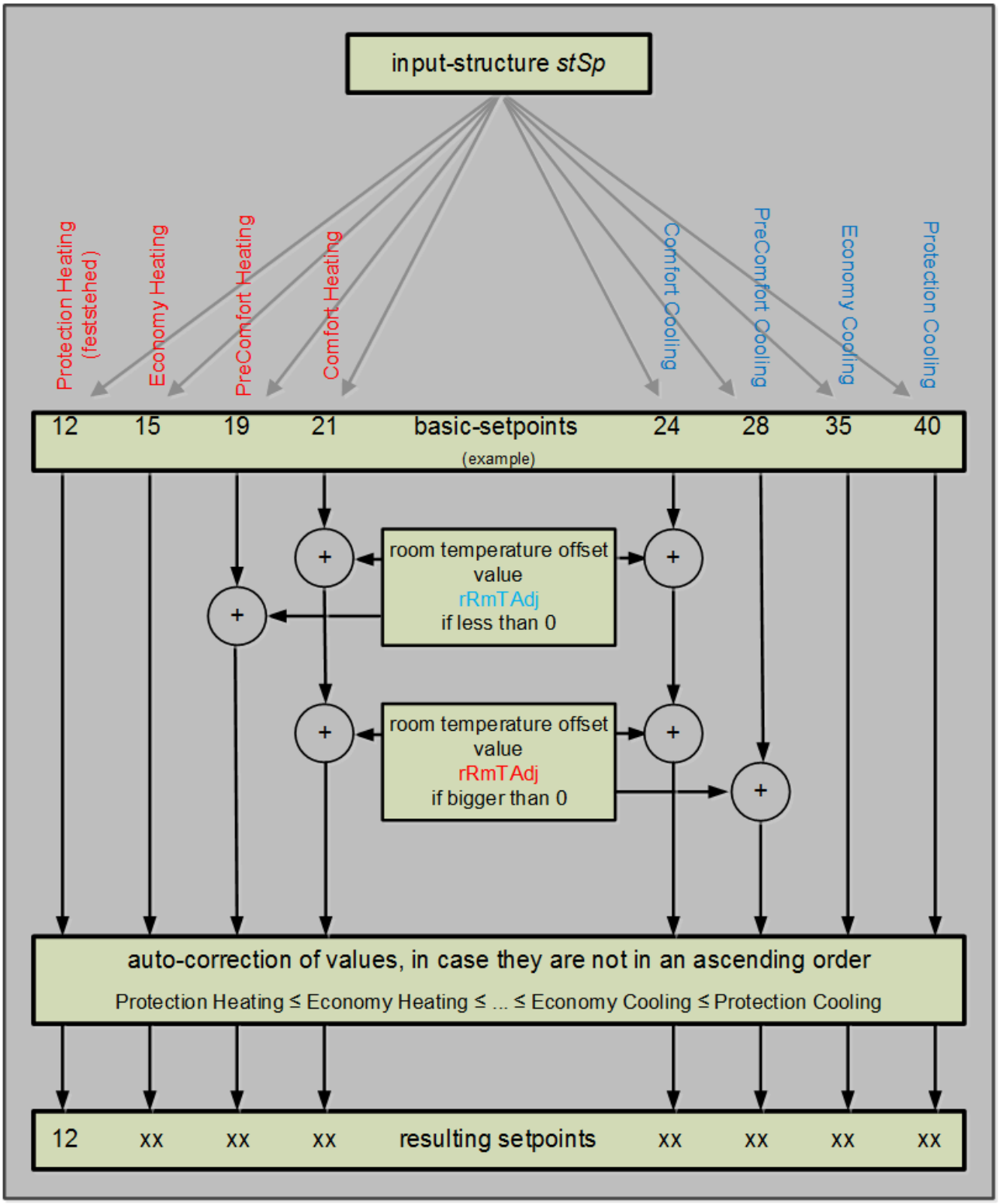
Requirements

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 [▶ 688]).

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.

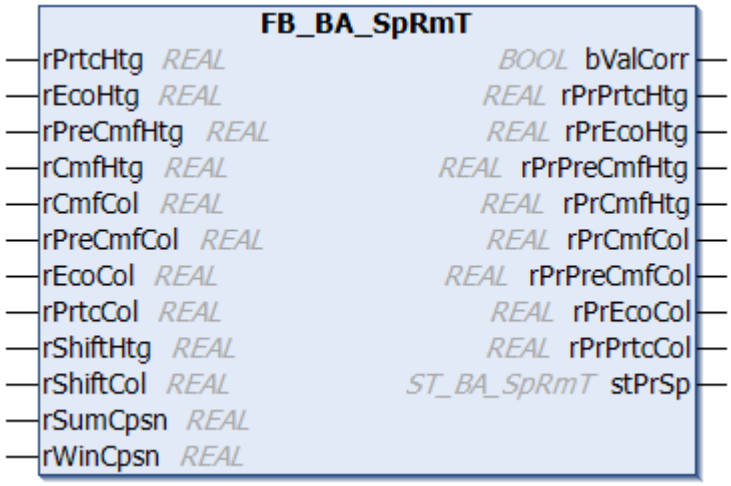
rPrPrtcCol: Resulting Protection Cooling setpoint.

stPrSp: Consolidated output of the resulting values in a structure (see ST_BA_SpRmT [▶ 688]).

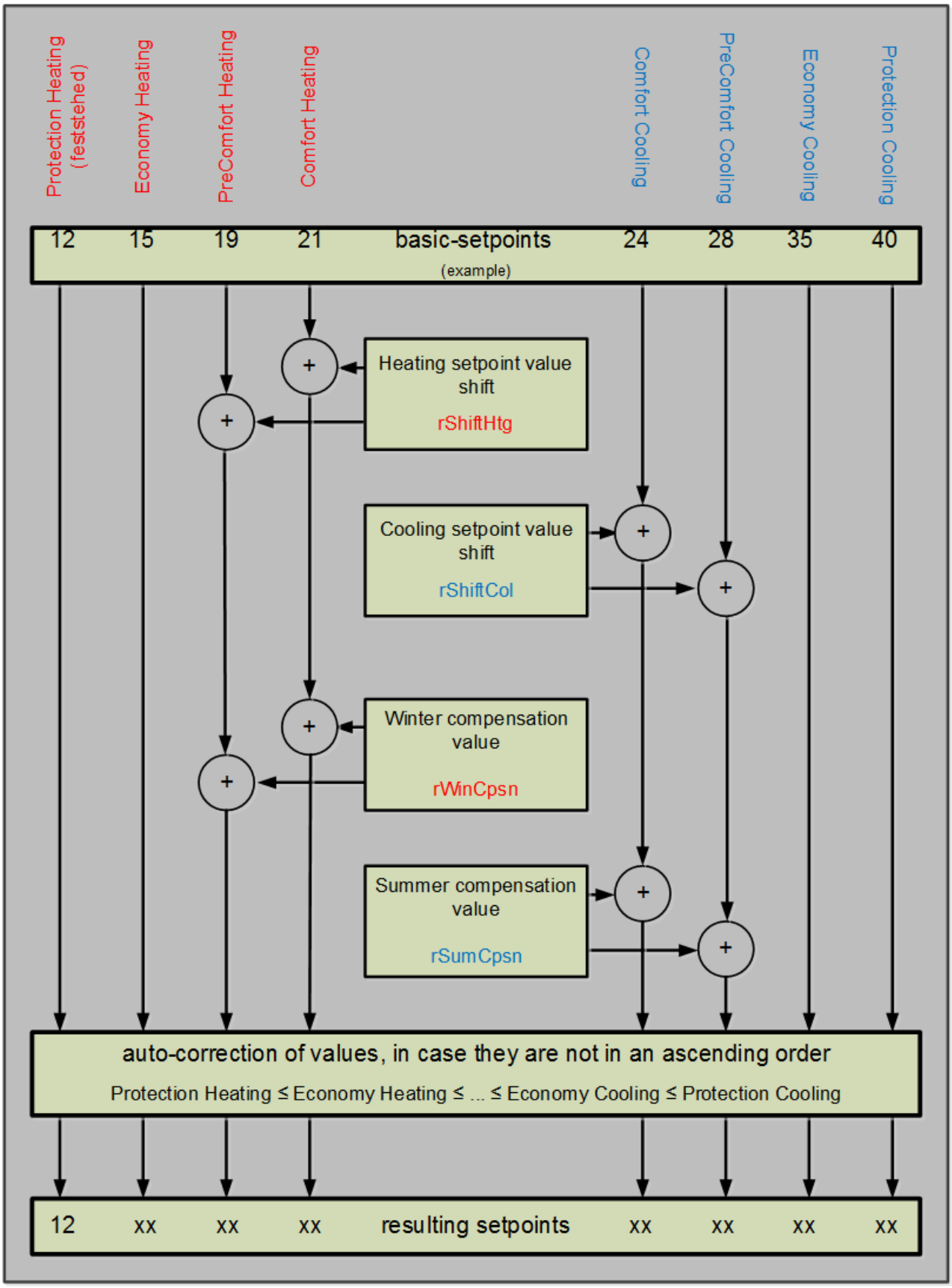
Requirements

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 [▶ 688]).

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

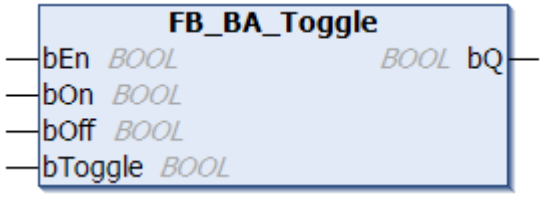
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.2 Lighting

Function blocks

Name	Description
FB_BA_Toggle [▶ 543]	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 [▶ 546]

Shading correction: Basic principles and definitions [▶ 546]

Overview of automatic sun protection [▶ 554]

Sun protection: Basic principles and definitions [▶ 556]

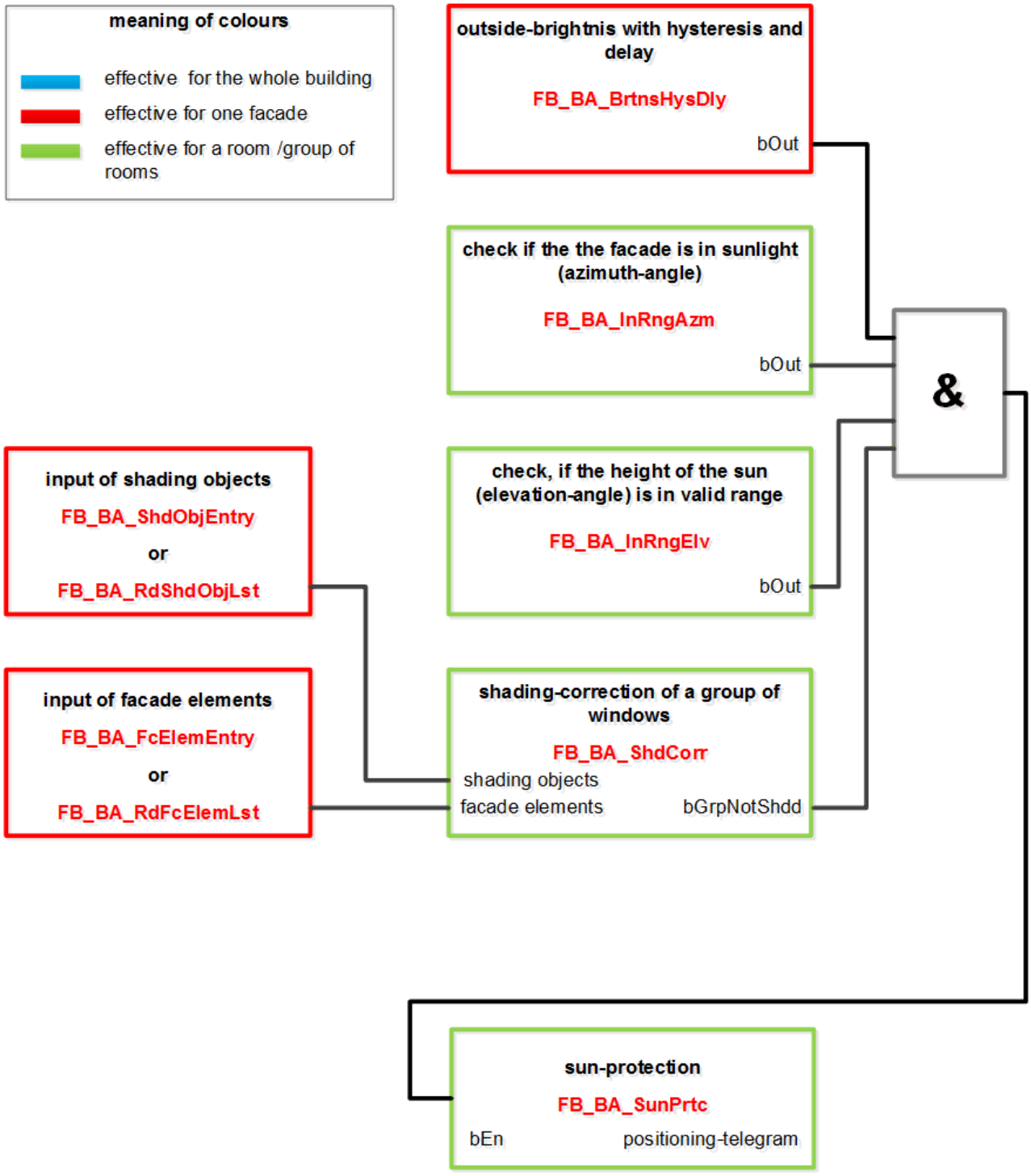
List of shading elements [▶ 561]

List of facade elements [▶ 561]

Function blocks

Name	Description
FB_BA_BldPosEntry [▶ 561]	Sun protection function: Input of blind positions.
FB_BA_CalcSunPos [▶ 563]	Calculation of sun position
FB_BA_FcdElemEntry [▶ 565]	Shading correction: Input of facade elements per function block.
FB_BA_InRngAzm [▶ 570]	Verification of valid sun position and sun direction range (azimuth angle)
FB_BA_InRngElv [▶ 572]	Verification of valid sun position and sun elevation range (elevation angle)
FB_BA_RdFcdElemLst [▶ 575]	Shading correction: Input of facade elements via data list (csv).
FB_BA_RdShdObjLst [▶ 579]	Shading correction: Input of shading objects via data list (csv).
FB_BA_RolBldActr [▶ 583]	Roller shutter actuator
FB_BA_ShdCorr [▶ 585]	Shading correction function block
FB_BA_ShdObjEntry [▶ 588]	Shading correction: Input of shading objects per function block.
FB_BA_SunBldActr [▶ 591]	Blind actuator
FB_BA_SunBldEvt [▶ 596]	Output of a specified blind position and angle in percent
FB_BA_SunBldIcePrtc [▶ 597]	Anti-icing
FB_BA_SunBldPosDly [▶ 598]	Switch-on delay for blinds/groups of blinds
FB_BA_SunBldPrioSwi4 [▶ 599]	Priority control, 4 inputs
FB_BA_SunBldPrioSwi8 [▶ 600]	Priority control, 8 inputs
FB_BA_SunBldScn [▶ 601]	Manual operation with scene selection and programming
FB_BA_SunBldSwi [▶ 604]	Manual operation
FB_BA_SunBldTwiLgtAuto [▶ 606]	Automatic twilight function
FB_BA_SunBldWndPrtc [▶ 607]	Protection against wind damage
FB_BA_SunPrtc [▶ 609]	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	β=0°
East	β=90°
South	β=180°
West	β=270°

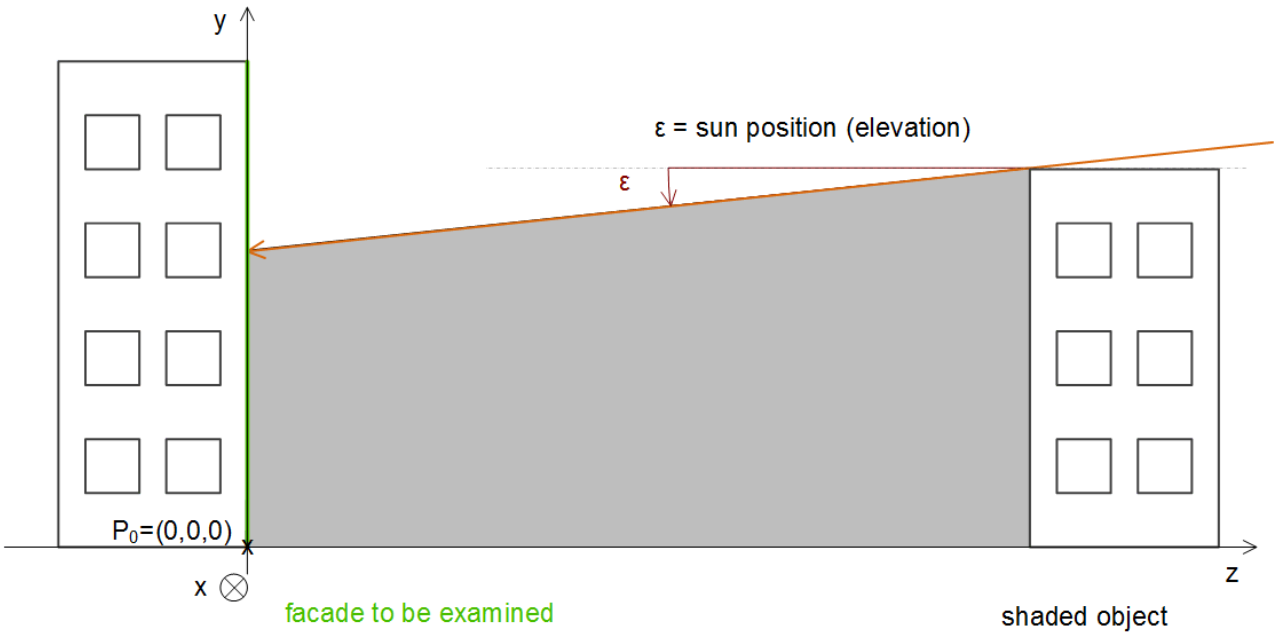
In the southern hemisphere is the sun path is the other way round: Although it also rises in the east, ad midday it is in the north. The facade orientation is adjusted to this path:

Line of sight	Facade orientation
South	β=0°
East	β=90°
North	β=180°
West	β=270°

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 [▶ 585] (shading correction) is parameterized they are activated through a boolean input, *bSouth*

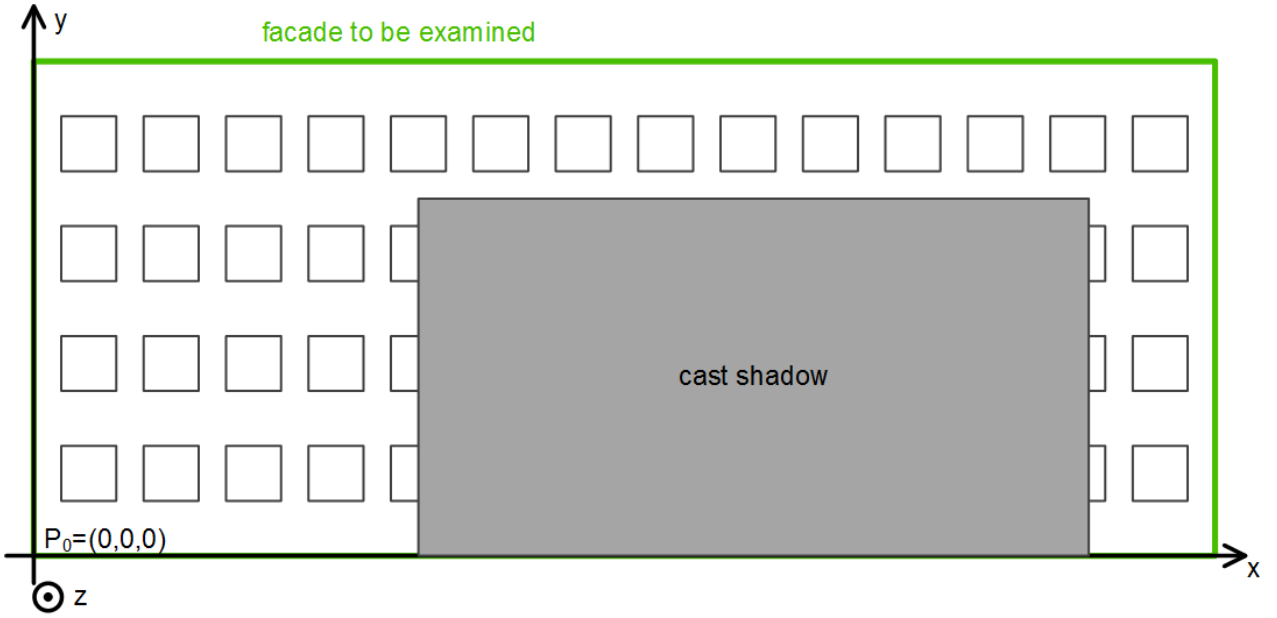
The following two illustrations are intended to further clarify the position of the point of origin P_0 as well as the orientation of the coordinate system:

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.

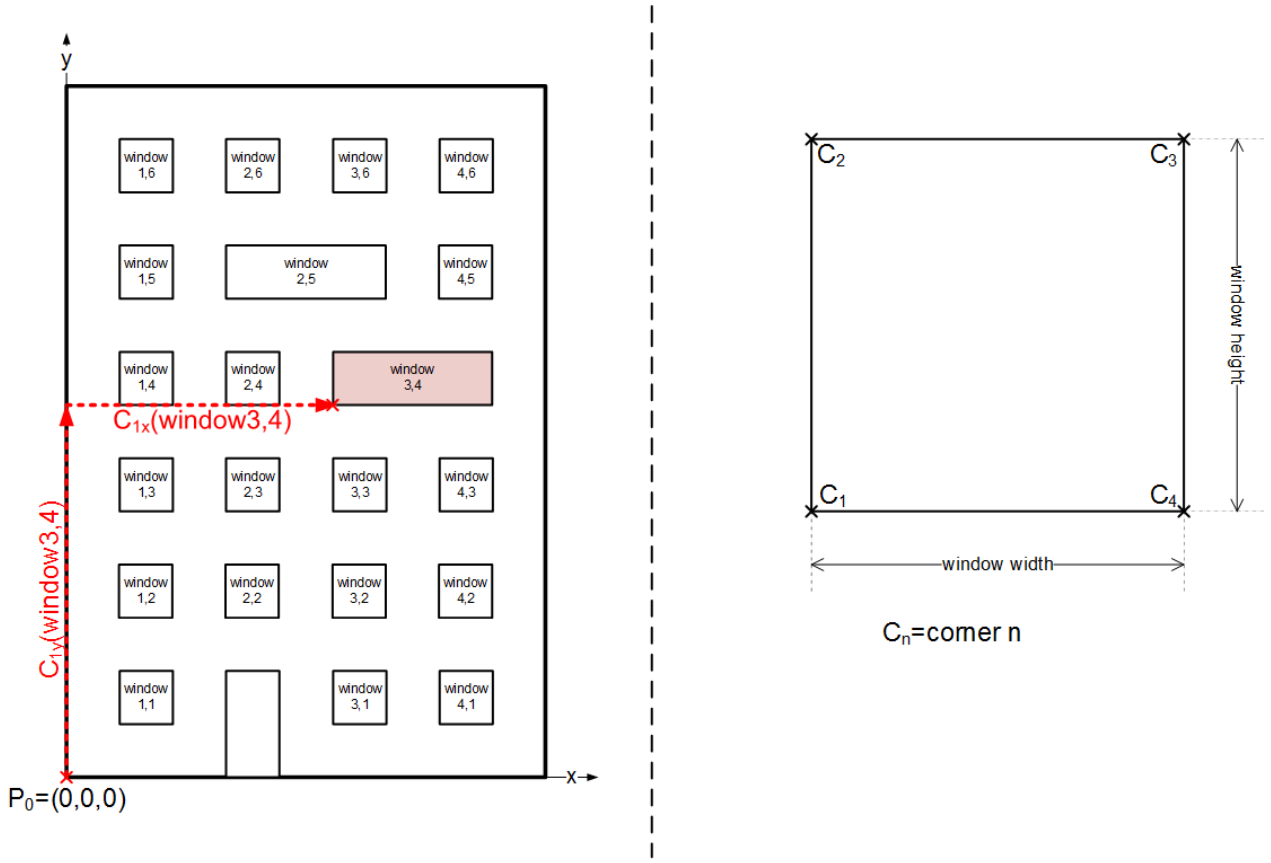
Observation from the front



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.



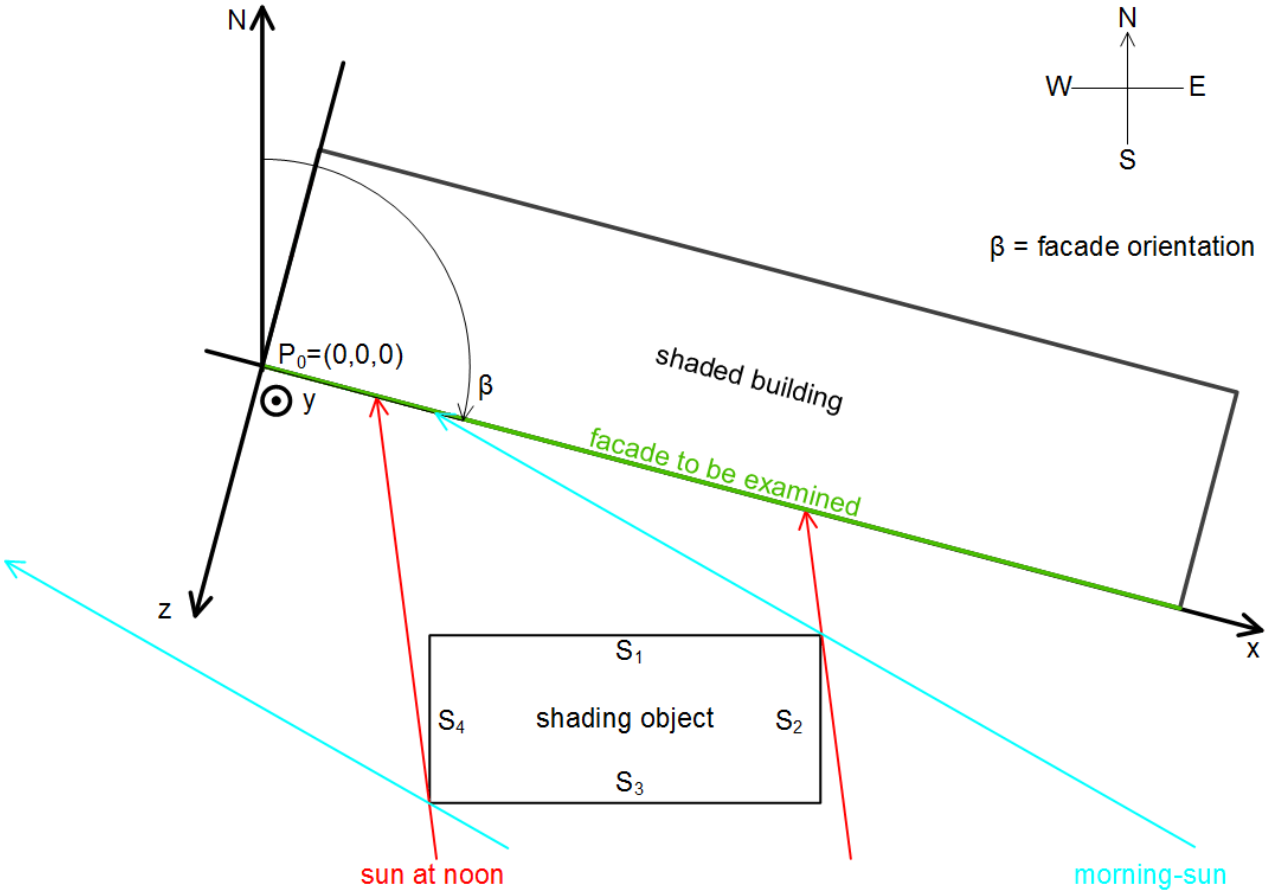
In addition, the window width and the window height have to be specified.

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

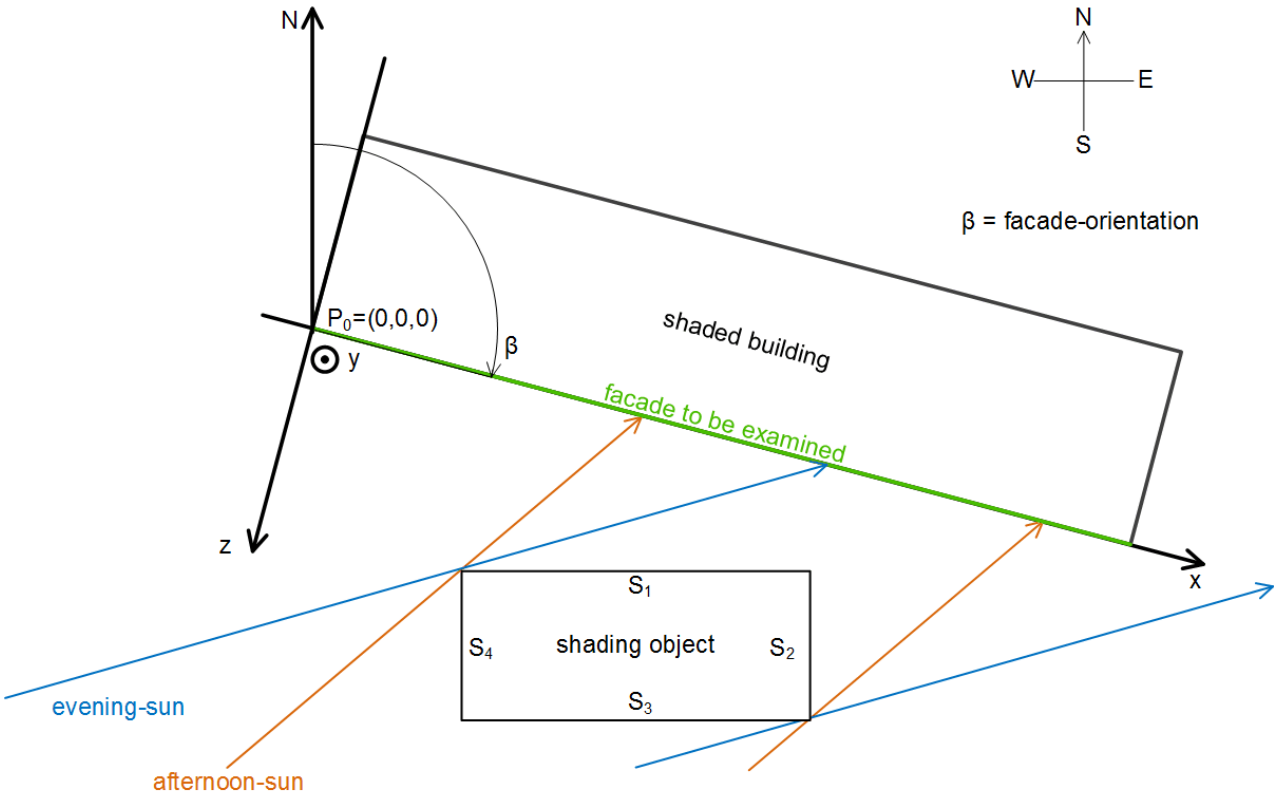
When describing the shading objects, distinction is made between angular objects (building, column) and objects that are approximately spherical (e.g. trees). Angular objects can be subdivided into rectangular shadow-casting facades depending on their shadow projection; you should consider which surfaces cast the main shadows over the day:

Morning/noon

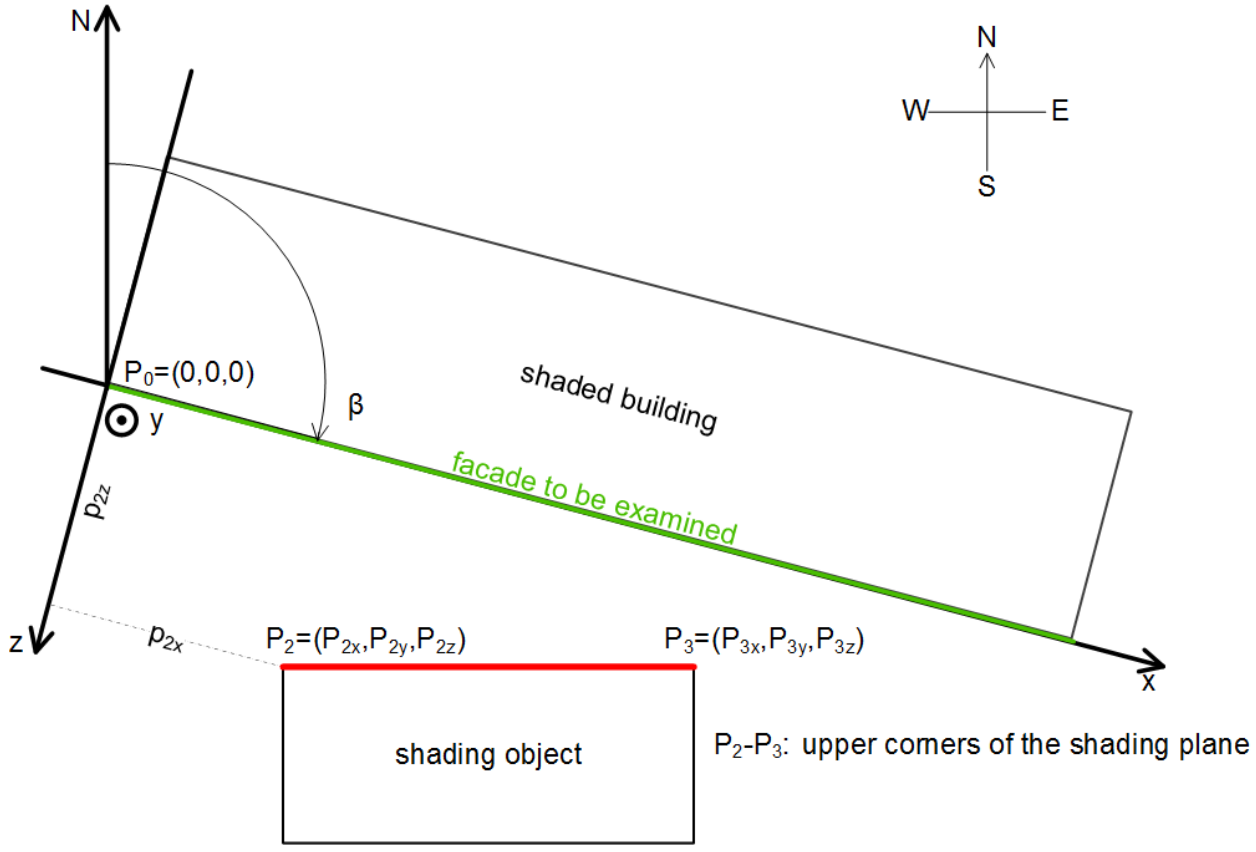


In the morning and around noon, the shadow is mainly cast by the sides S_1 and S_4. S_2 and S_3 do not have to be considered, unless they are higher.

Afternoon/evening



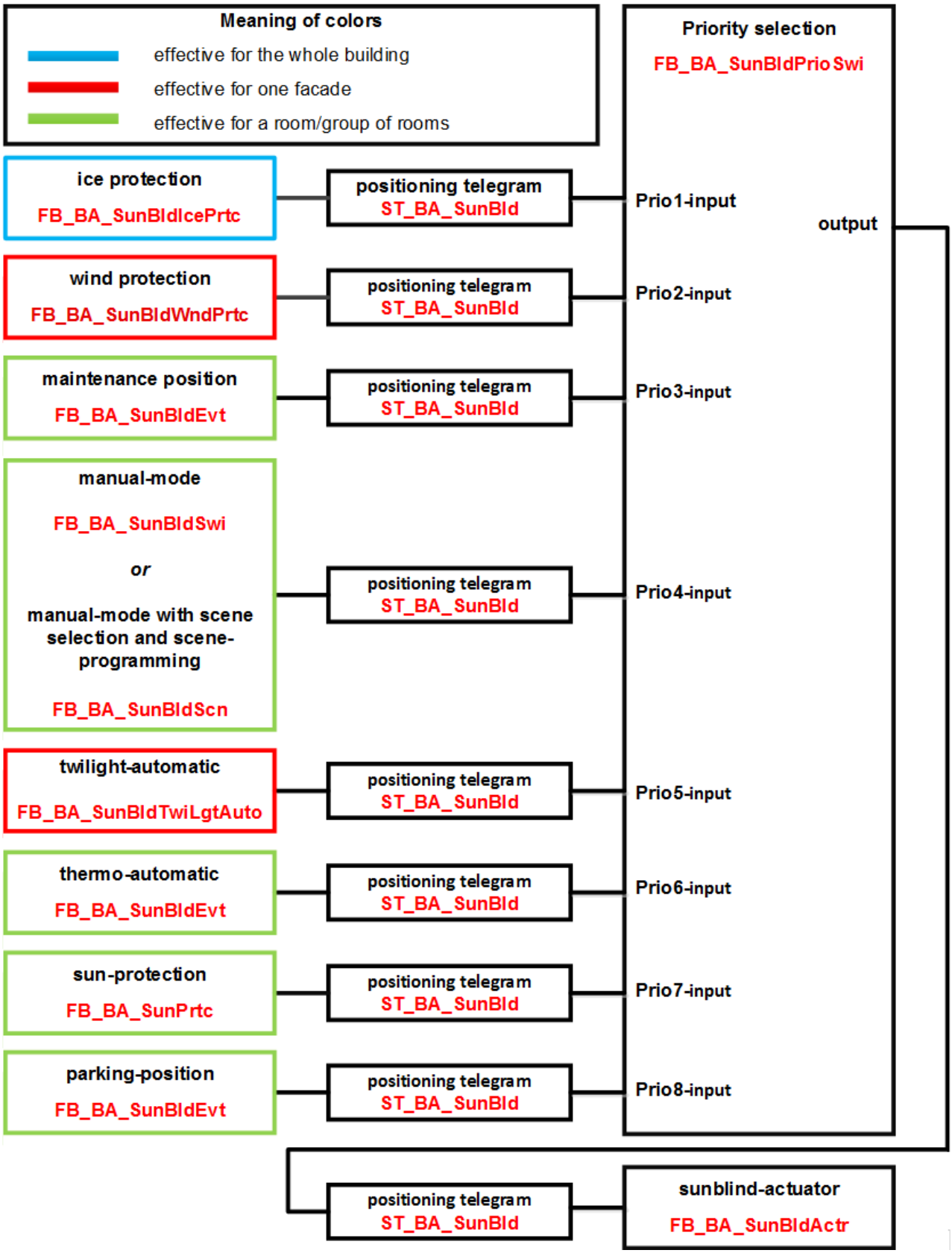
In the afternoon and evening, the total shade can be determined solely through S_1 and S_2. In this case it is therefore sufficient to specify S_1, S_2 and S_4 as shadow casters. The entry is made on the basis of the four corners or their coordinates in relation to the zero point of the facade:



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.

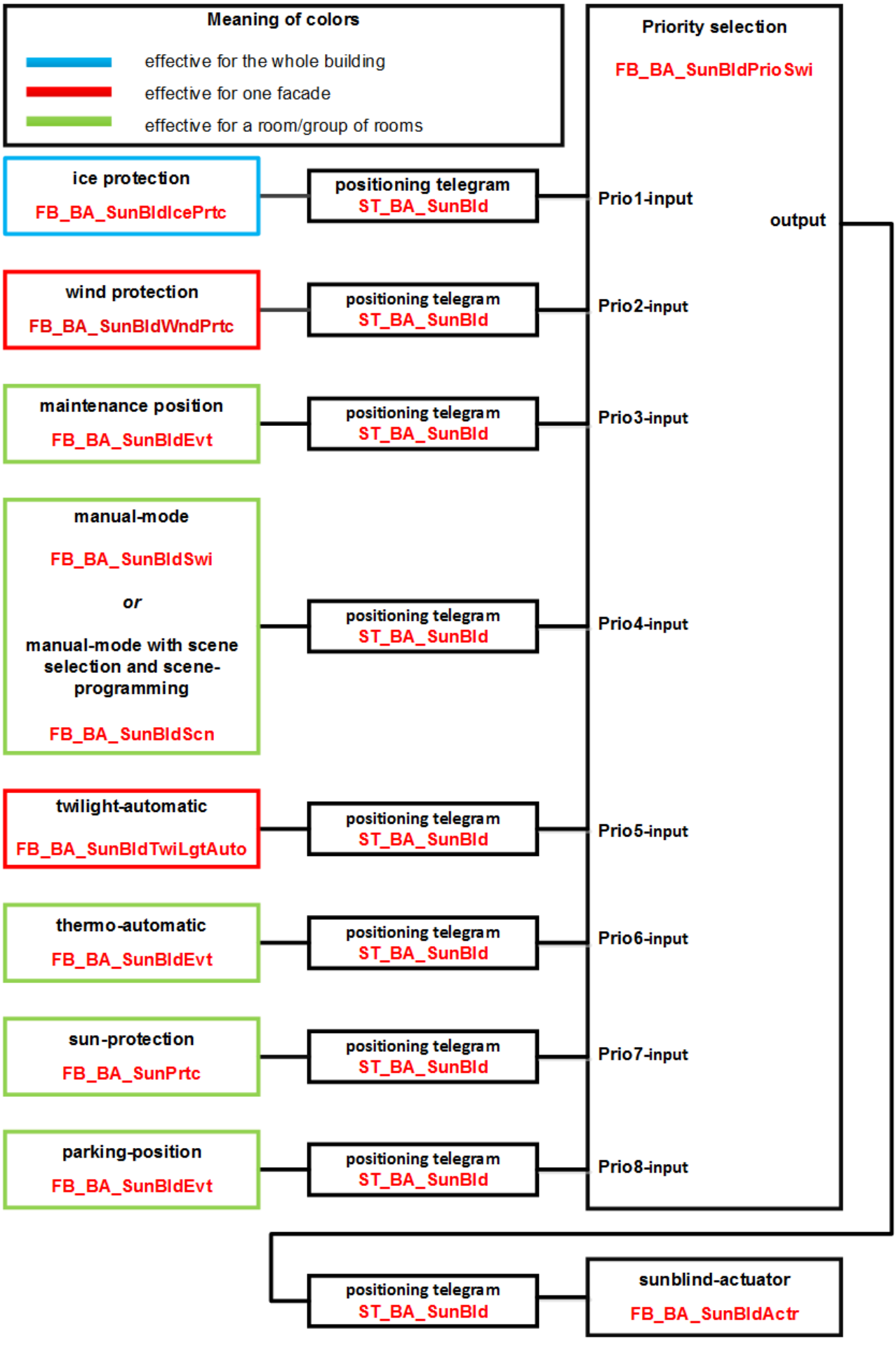
The input of shadow-casting ball elements is done by entering the center of the ball and its radius:

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


Meaning of colors
effective for the whole building
effective for one facade
effective for a room/group of rooms
Priority selection
FB_BA_SunBldPrioSwi
ice protection
FB_BA_SunBldIcePrtc
positioning telegram
ST_BA_SunBld
Prio1-input
output
wind protection
FB_BA_SunBldWndPrtc
positioning telegram
ST_BA_SunBld
Prio2-input
maintenance position
FB_BA_SunBldEvt
positioning telegram
ST_BA_SunBld
Prio3-input
manual-mode
FB_BA_SunBldSwi
or
manual-mode with scene selection and scene-programming
FB_BA_SunBldScn
positioning telegram
ST_BA_SunBld
Prio4-input
twilight-automatic
FB_BA_SunBldTwiLgtAuto
positioning telegram
ST_BA_SunBld
Prio5-input
thermo-automatic
FB_BA_SunBldEvt
positioning telegram
ST_BA_SunBld
Prio6-input
sun-protection
FB_BA_SunPrtc
positioning telegram
ST_BA_SunBld
Prio7-input
parking-position
FB_BA_SunBldEvt
positioning telegram
ST_BA_SunBld
Prio8-input
positioning telegram
ST_BA_SunBld
sunblind-actuator
FB_BA_SunBldActr

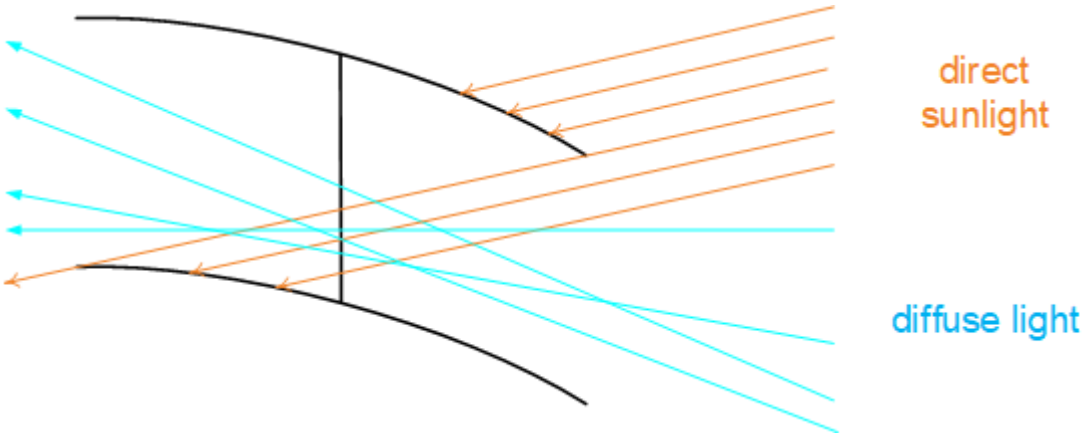
6.1.2.3.2.1.2.3.4 Sun protection: Basic principles and definitions

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

Lamella setpoint tracing

A blind with lamellas that can be adjusted offers the option of intelligent sun protection here. The position of the lamellas 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 lamella angle:

- the current sun elevation (elevation angle)
- the sun position, i.e. the azimuth angle
- the facade orientation
- the lamella width
- the lamella spacing

Effective elevation angle

If the blind is viewed in section as above, the angle of incidence does not depend solely on the sun 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 elevation angle.

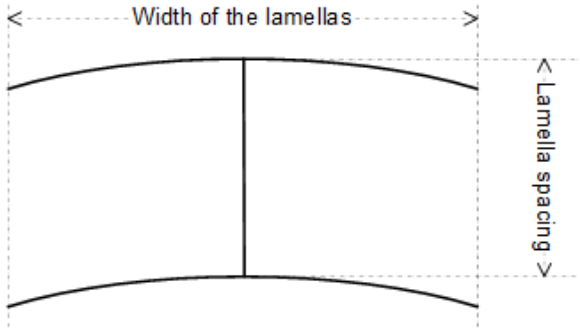
This relationship can easily be illustrated with a set square positioned upright on the table: Viewed directly from the side you can see a triangle with two 45° angles and one 90° angle. If the triangle is rotated, the side on the table appears to become shorter and the two original 45° angles change. 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.

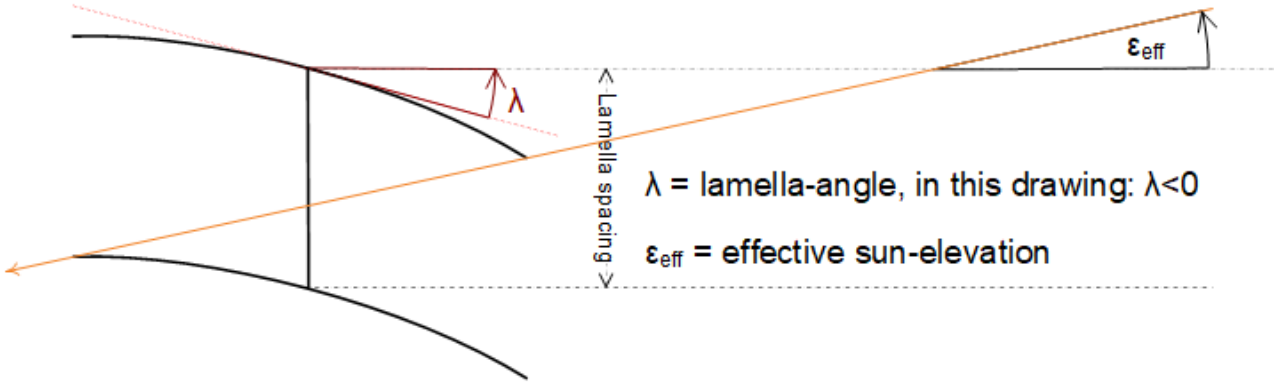
The following three images illustrate the relationship between the effective elevation angle and the blind dimensions, and how the resulting lamella angle λ changes during the day:

Lamella-angle

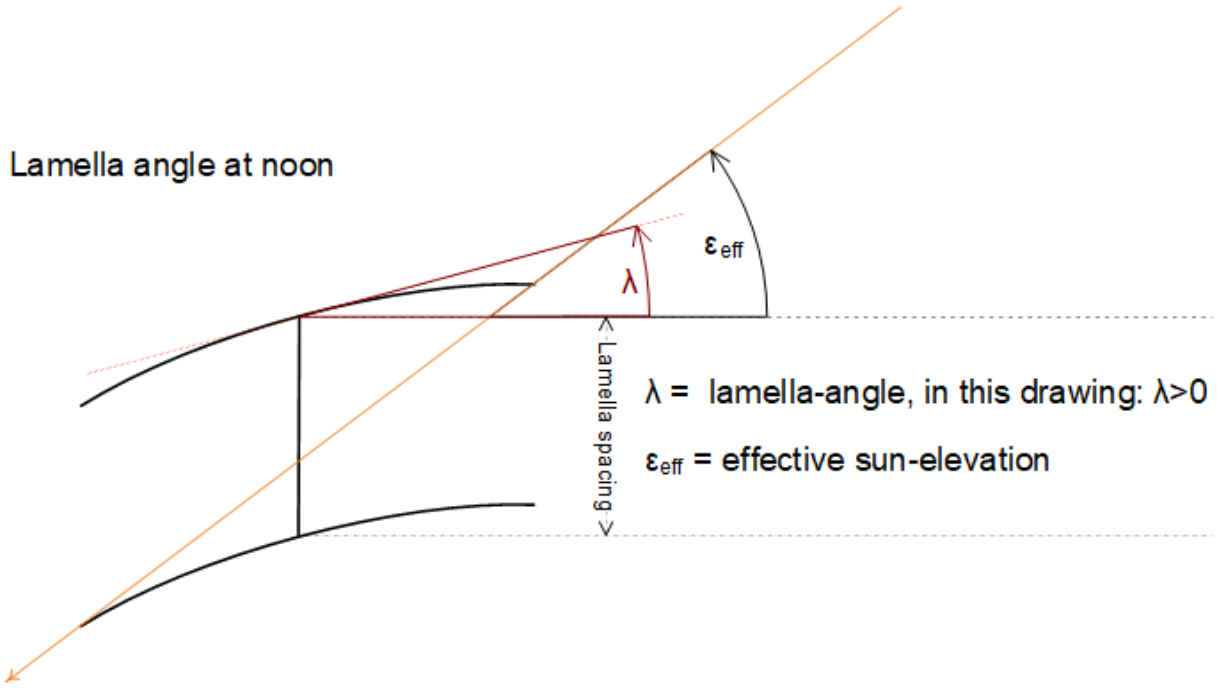
Lamella at an angle of λ=0



Lamella-angle in the morning and in the evening

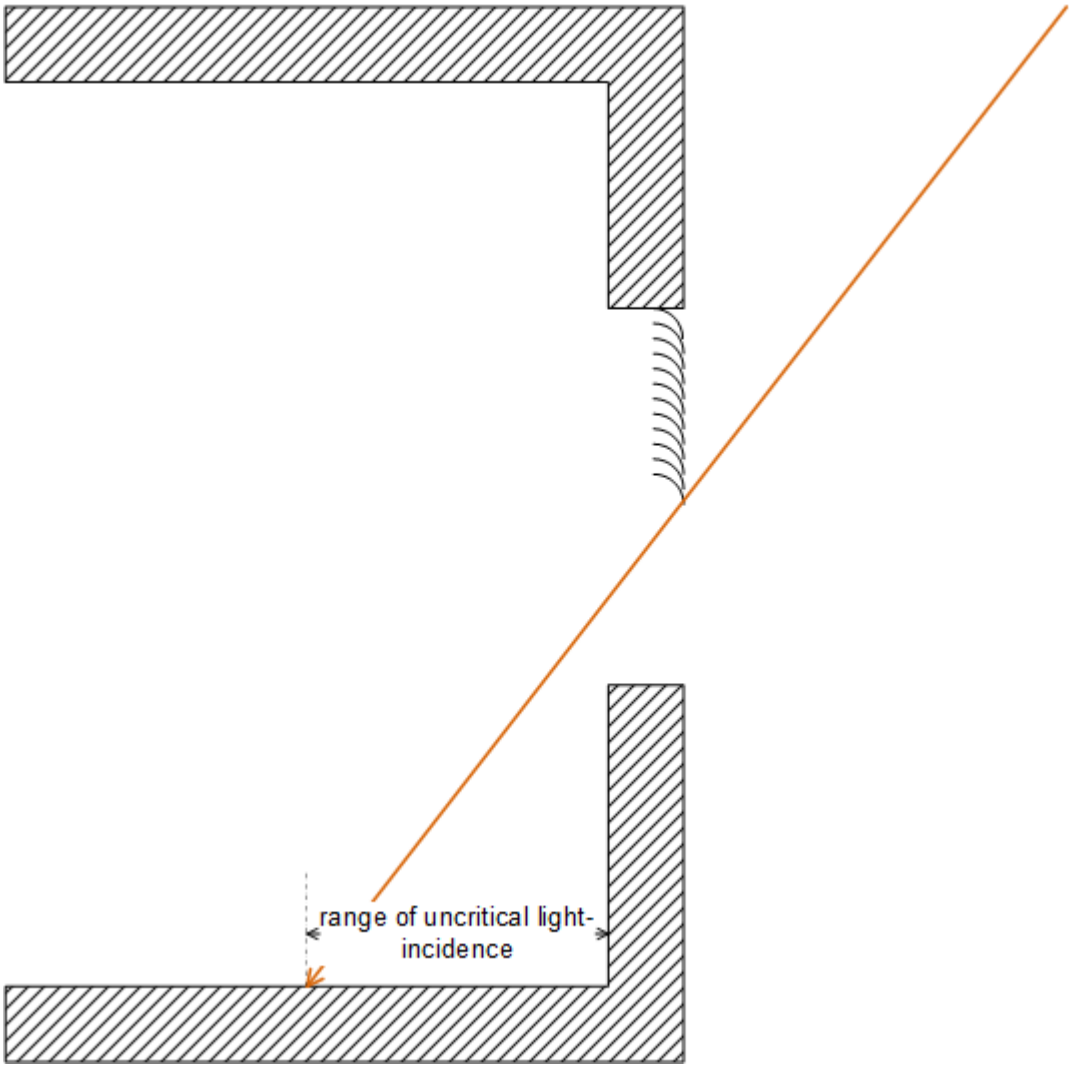


Lamella angle at noon



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.

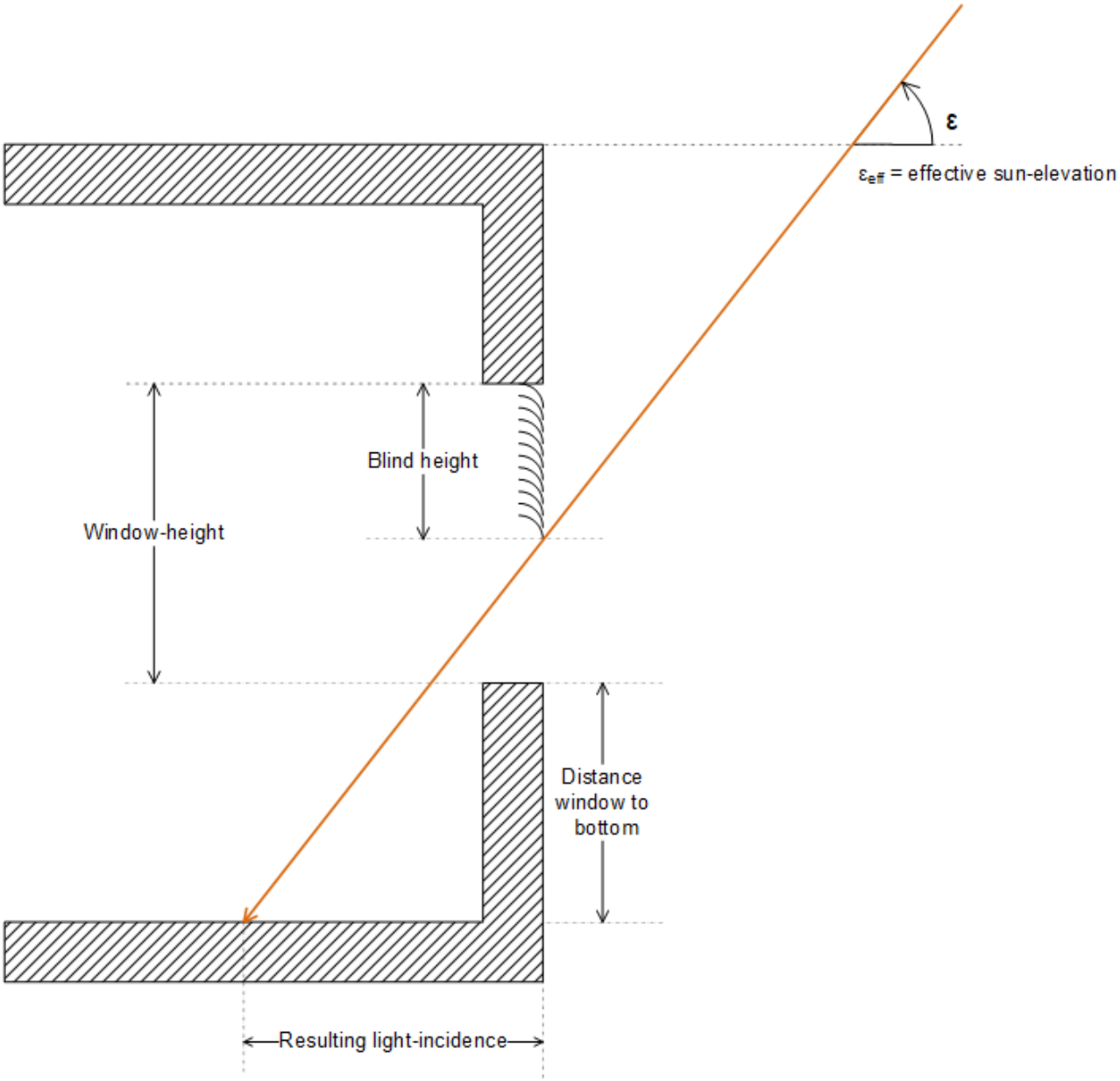


In order to be able to calculate at any time the appropriate blind height that guarantees that the incidence of sunlight does not exceed a certain value, the following values are necessary.

Required for the calculation of the respective blind height:

- Sun elevation
- Window height
- Distance between the window and the floor

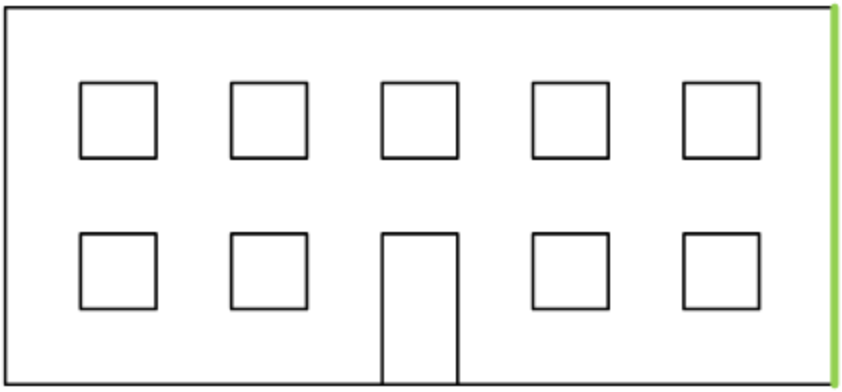
The following illustration shows where these parameters are to be classified:



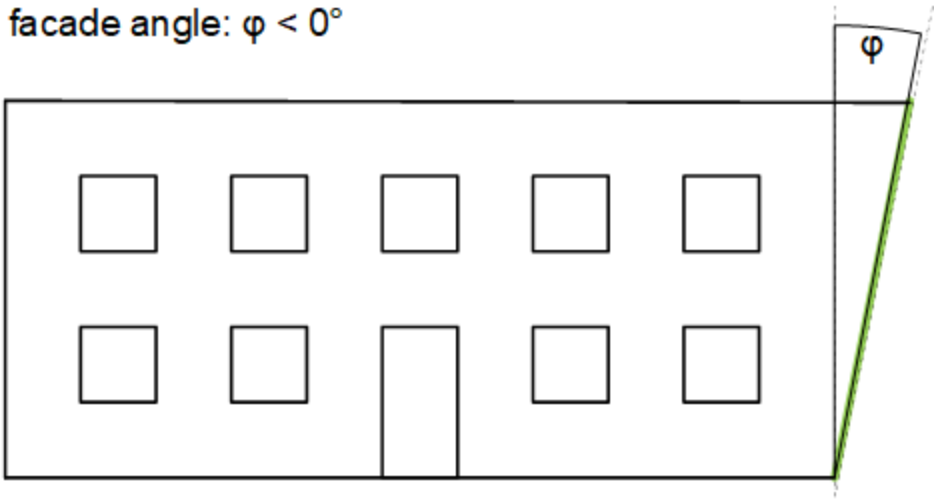
Influence of the facade inclination

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:

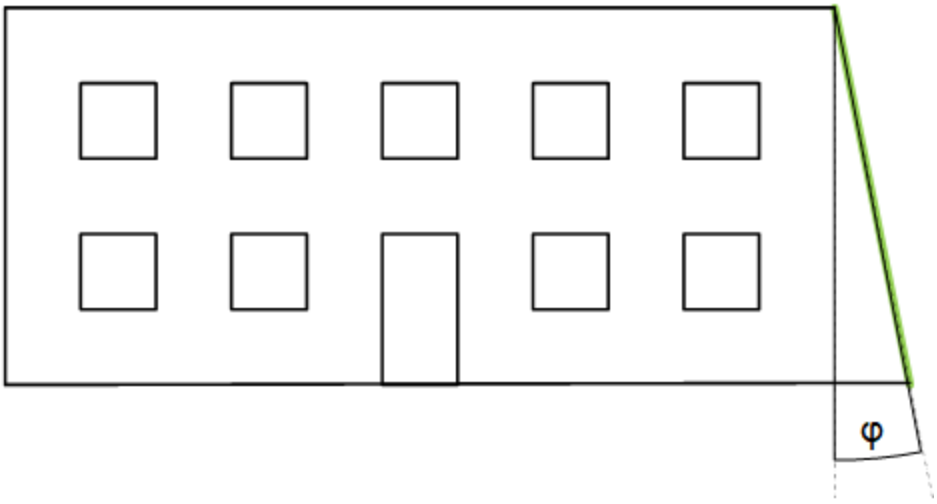
facade angle: φ = 0°



facade angle: φ < 0°



facade angle: φ > 0°



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_ShdObj [▶ 688] within the program.

The shading correction FB_BA_ShdCorr [▶ 585] reads the information from this list. The management function block FB_BA_ShdObjEntry [▶ 588] 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_ShdObj;
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 [▶ 687].

The management function block FB_BA_FcdElemEntry [▶ 565] and the shading correction FB_BA_ShdCorr [▶ 585] 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

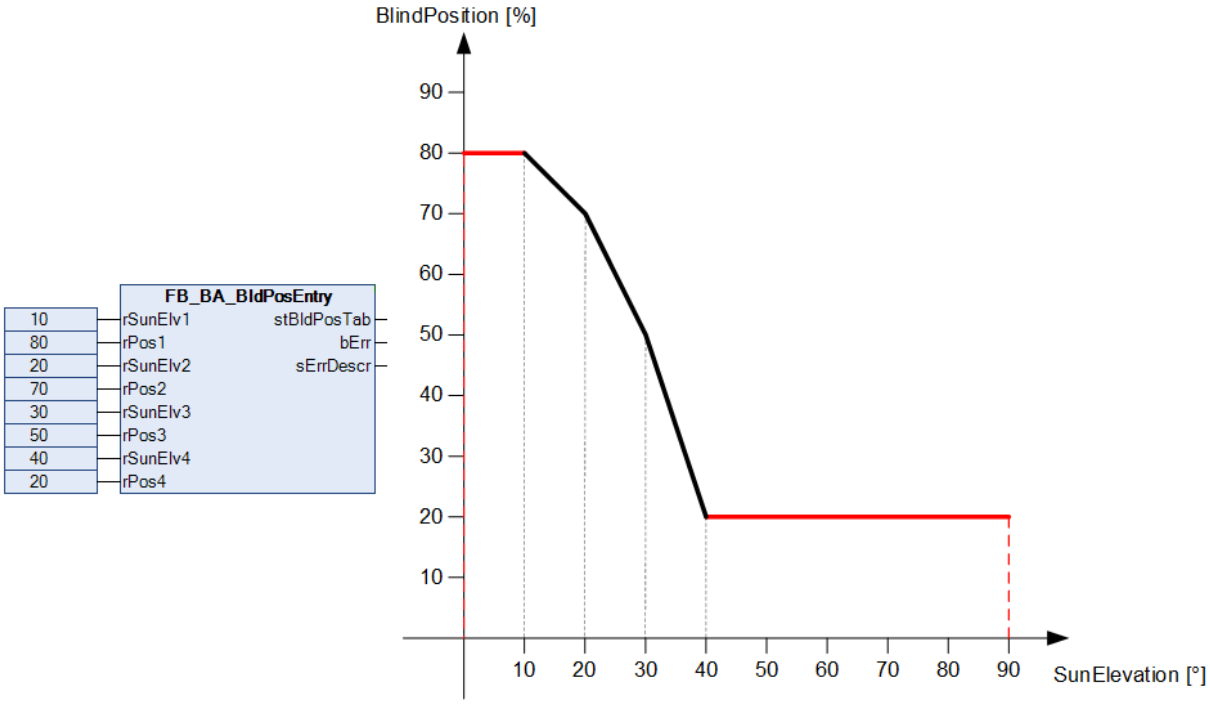
This function block is used for entering interpolation points for the function block FB_BA_SunPrtc [▶ 609], if this function block is operated in height positioning mode with the aid of a table, see E_BA_PosMod [▶ 685].

In addition to the operating modes "Fixed shutter height" and "Maximum incidence of light", the function block FB_BA_SunPrtc [▶ 609] 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 [▶ 609], 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 ≤ *rSunElv* ≤ 90 ([°] - scope source values)
- 0 ≤ *rPos* ≤ 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 [▶ 686] 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* ≤ *rSunElv* ≤ *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 [▶ 609] to 6; see ST_BA_BldPosTab [▶ 686].

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 [▶ 686].

bErr: This output is switched to TRUE if the parameters entered are erroneous.

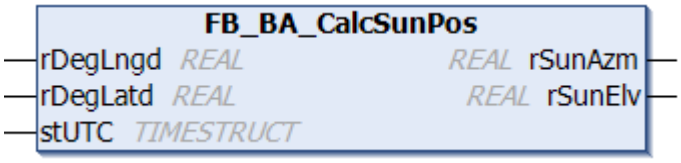
sErrDescr: Contains the error description.

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

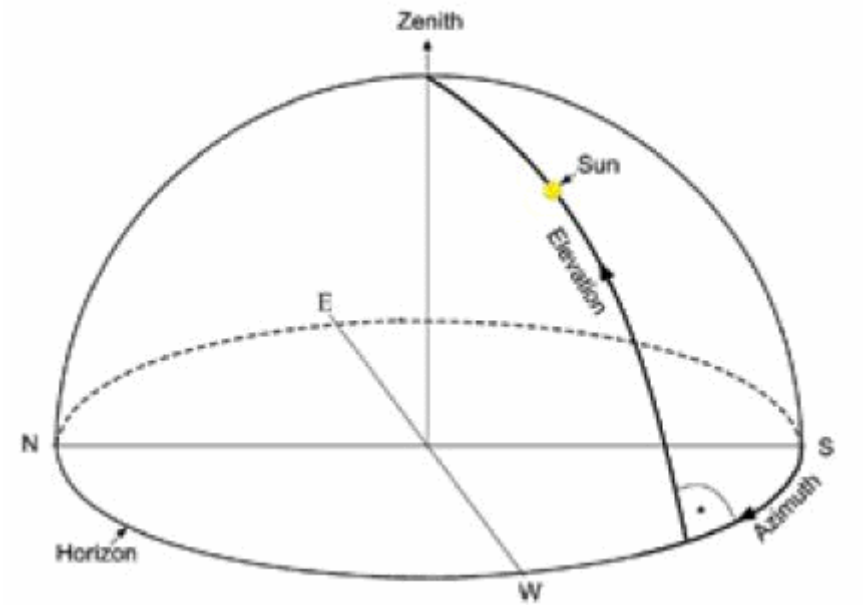
6.1.2.3.2.1.2.3.8 FB_BA_CalcSunPos



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 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 is, 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 [▶ 618] can be used to read this time from a target system.

VAR_OUTPUT

```
rSunAzm     : REAL;
rSunElv     : REAL;
```

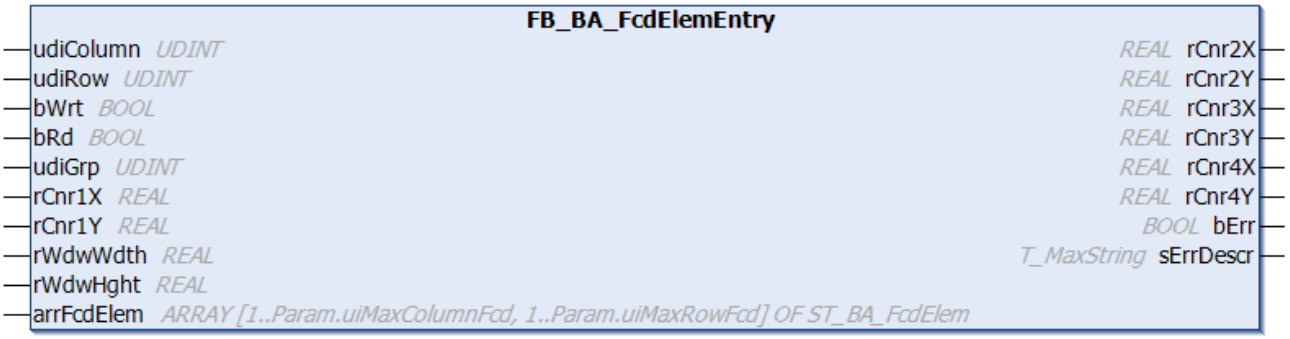
rSunAzm: Direction of the sun (northern hemisphere: 0° north ... 90° east ... 180° south ... 270° west ... / southern hemisphere: 0° south ... 90° east ... 180° north ... 270° west ...).

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



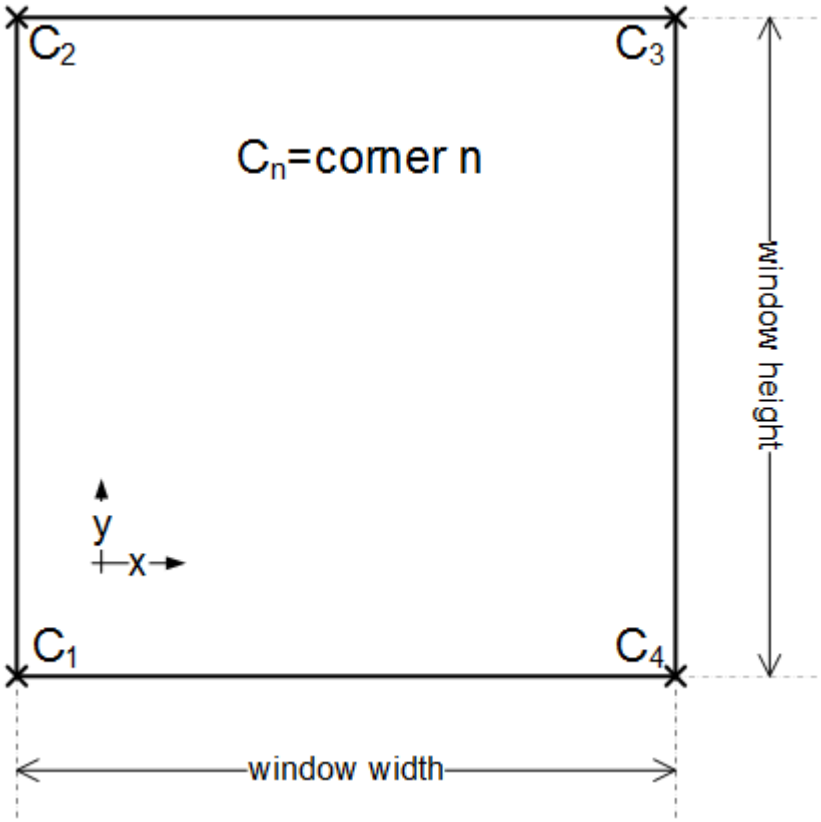
This function block serves the administration of all facade elements (windows) in a facade, which are saved globally in a list of facade elements [▶ 561]. It is intended to facilitate inputting element information - not least with regard to using the TC3 PLC HMI. A schematic illustration of the objects with description of the coordinates is given in Shading correction: principles and definitions [▶ 546].

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 [▶ 687]). 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 C2-C4 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 [▶ 687] therefore also contains a plausibility bit *bVld*, which forwards precisely this information to the function block FB_BA_ShdCorr [▶ 585] to prevent miscalculations.

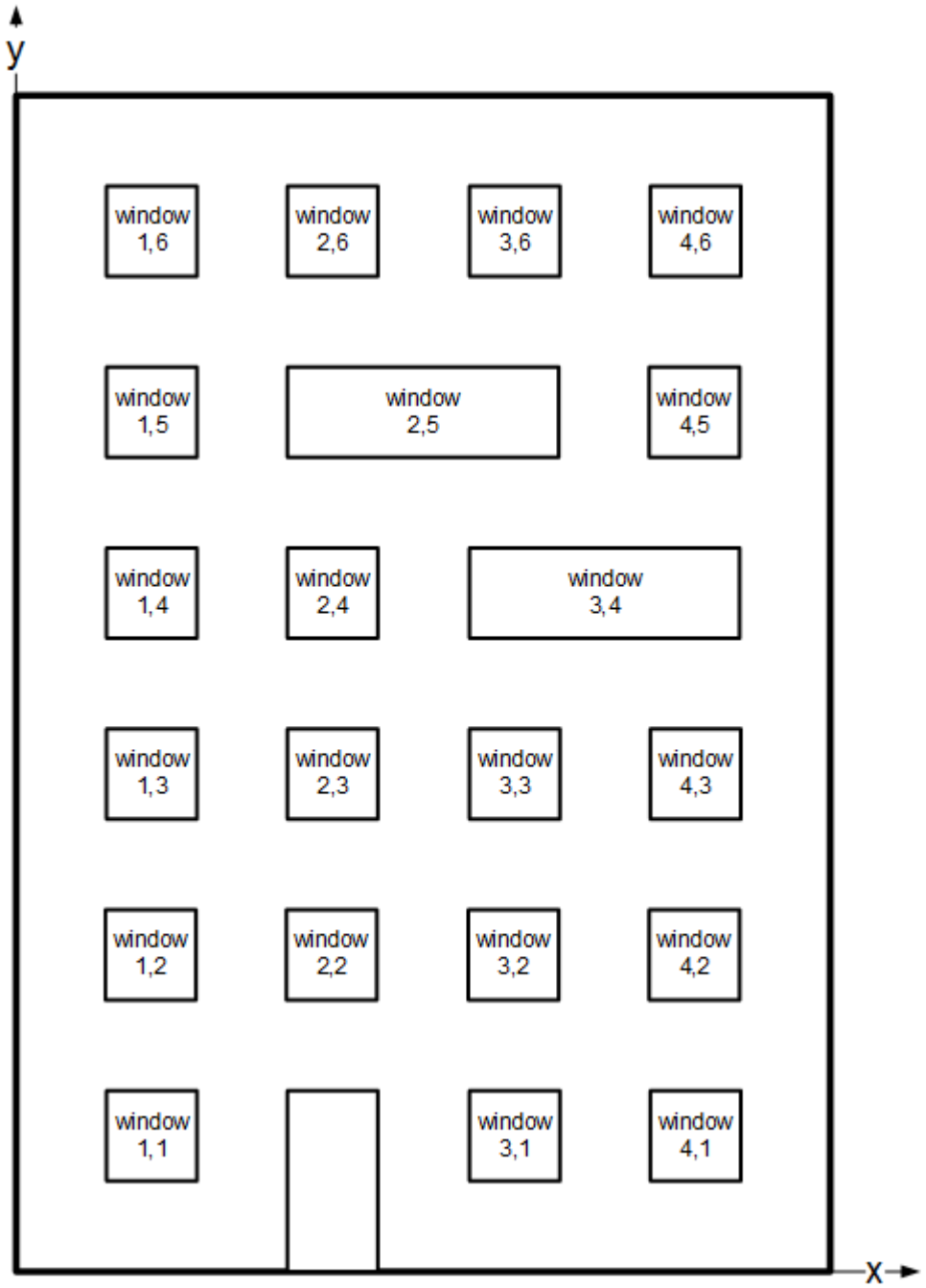
This approach is to be regarded only as a proposal. It is naturally also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on *bWrt*.

Error (bErr=TRUE)

The function block FB_BA_ShdCorr [▶ 585], 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 [▶ 687], 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;
rCnr1X    : REAL;
rCnr1Y    : 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 [▶ 565]" above.

rCnr2Y: Calculated Y-coordinate of corner point 2 of the window [m]. See "Note [▶ 565]" above.

rCnr3X: Calculated X-coordinate of corner point 3 of the window [m]. See "Note [▶ 565]" above.

rCnr3Y: Calculated Y-coordinate of corner point 3 of the window [m]. See "Note [▶ 565]" above.

rCnr4X: Calculated X-coordinate of corner point 4 of the window [m]. See "Note [▶ 565]" above.

rCnr4Y: Calculated Y-coordinate of corner point 4 of the window [m]. See "Note [▶ 565]" 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

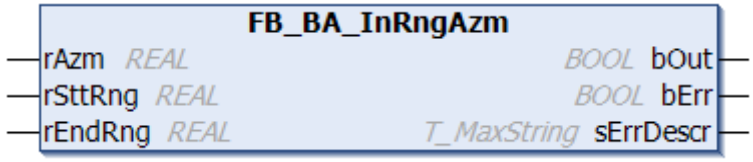
```
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: List of facade elements (see List of facade elements [▶ 561]).

Requirements

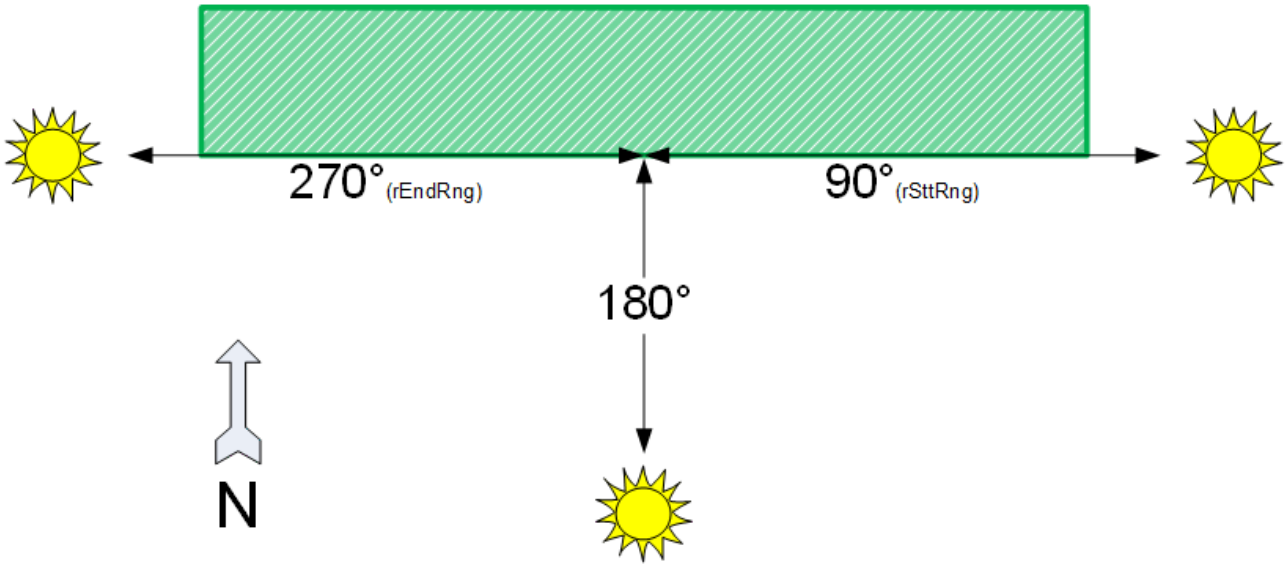
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.10 FB_BA_InRngAzm

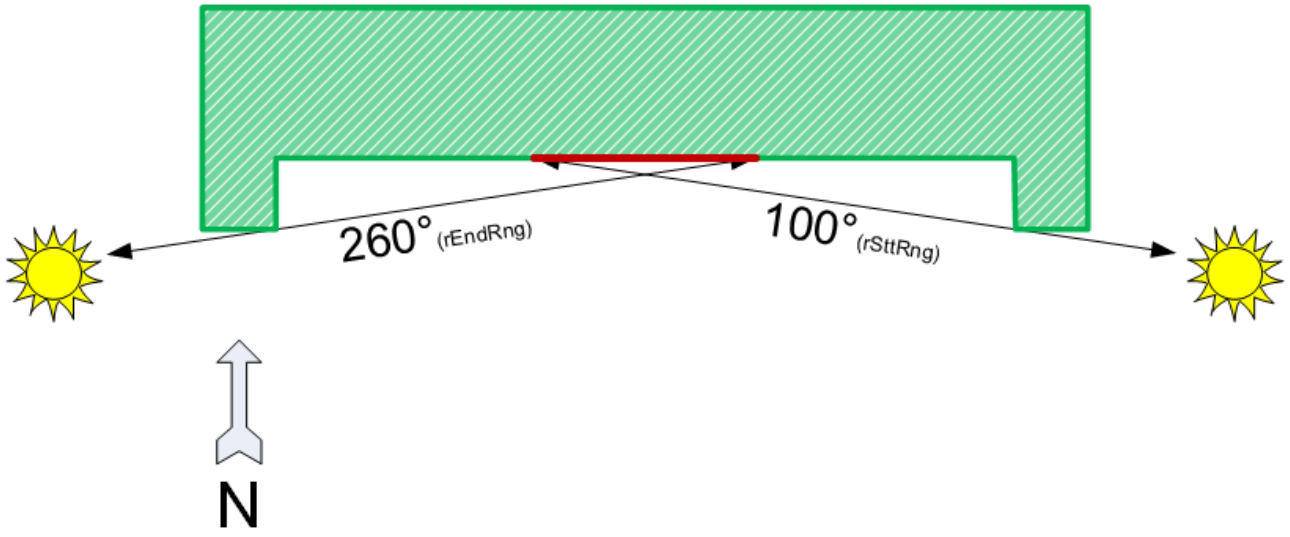


This function block checks whether the current azimuth angle (horizontal position of the sun) lies within the limits entered. As can be seen in the overview [▶ 554], the function block provides an additionally evaluation as to whether the sun shading of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

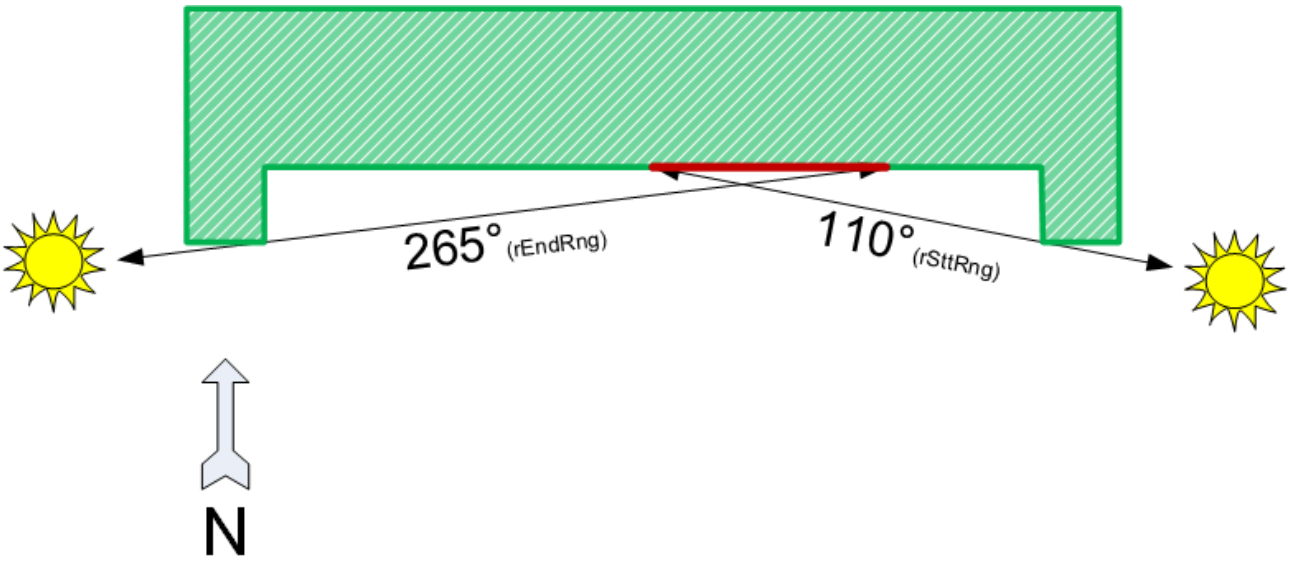
A smooth facade is always irradiated by the sun at an azimuth angle of *Facade orientation-90°* to *Facade orientation+90°*.



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.

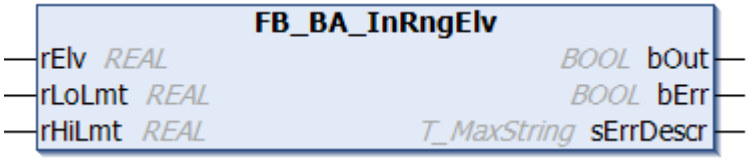
sErrDescr: Contains the error description.

Error description
01: Error: *rSttRng* or *rEndRng* less than 0° or greater than 360°.
02: Error: The difference between *rSttRng* and *rEndRng* is greater than 180°. This range is too large for analyzing the insolation on a facade.

Requirements

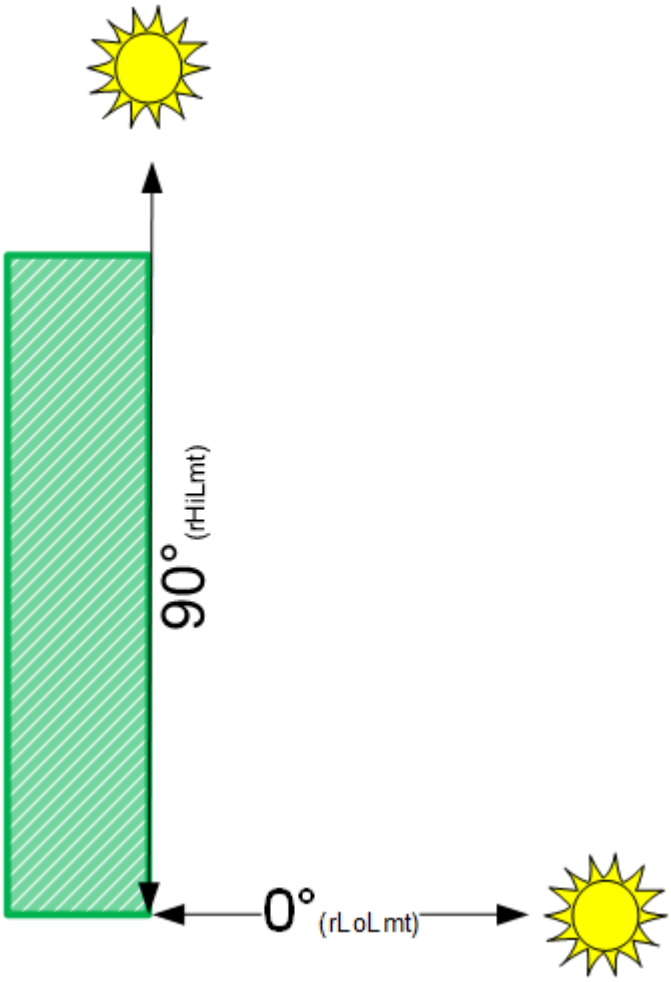
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.2.3.11 FB_BA_InRngElv

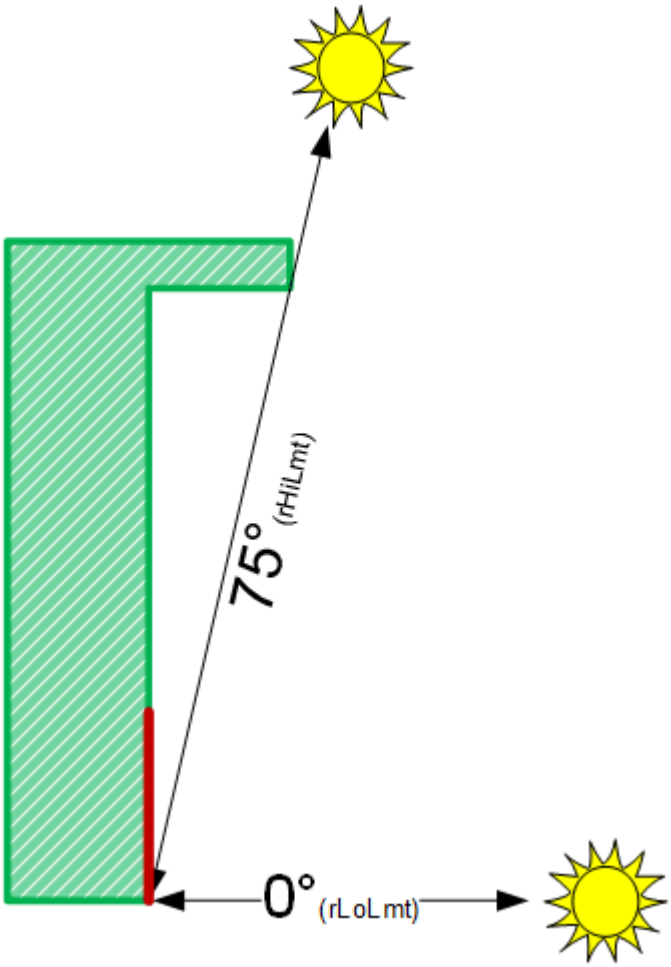


This function block checks whether the current angle of elevation (vertical position of the sun) lies within the limits entered. As can be seen in the overview [▶ 554], the function block provides an additionally evaluation as to whether the sun shading of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

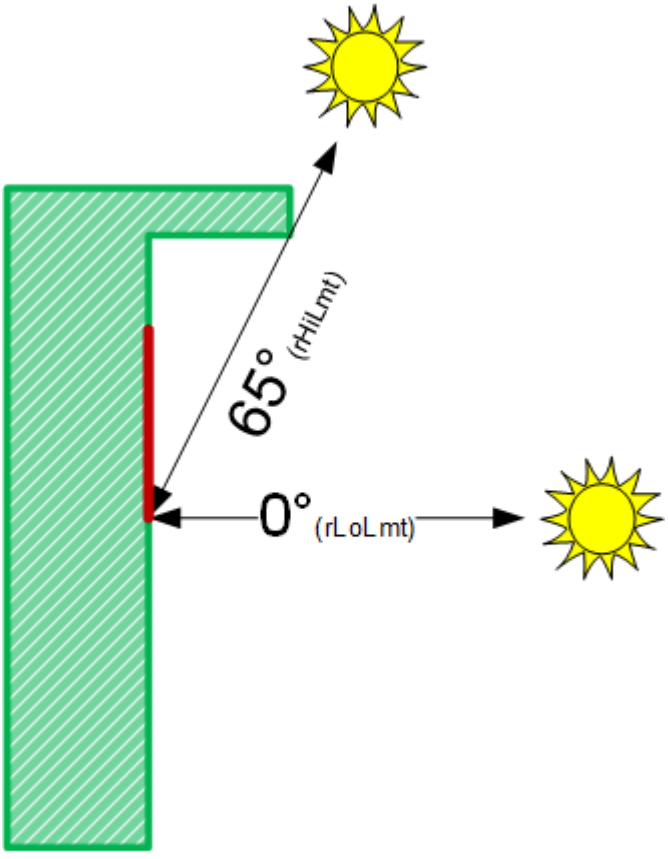
A normal vertical facade is irradiated by the sun at an angle of elevation of 0° to maximally 90°.



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: *rHiLmt* less than or equal to *rLoLmt*.
02: Error: *rLoLmt* is less than 0° or *rHiLmt* 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

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 [▶ 561]. 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 [▶ 561]. 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 [▶ 585], 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										

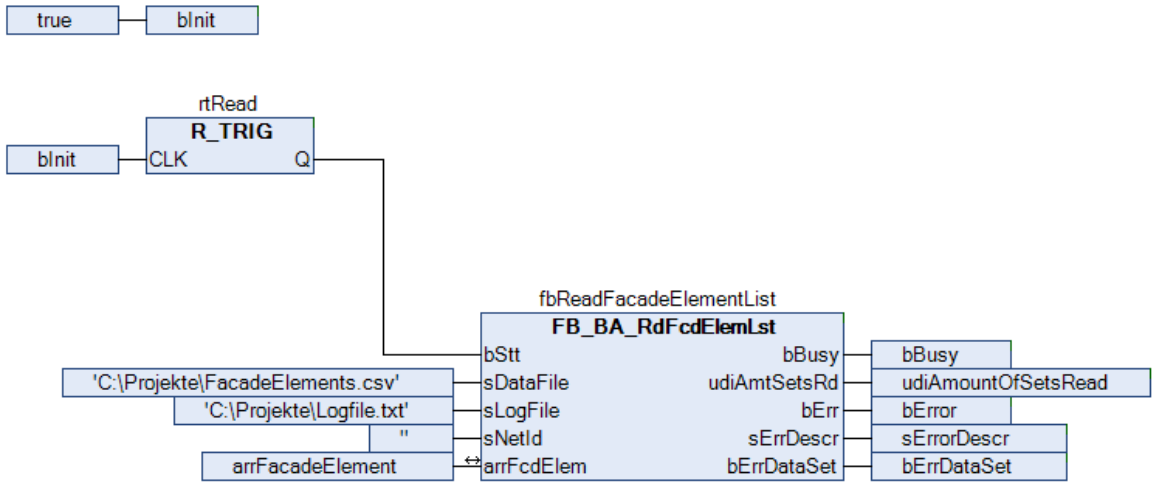
Log file

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		IndexColum	IndexRow	Window wid	Window hei	P1x	P1y	Group	
2				(Axis)	(Floor)	[m]	[m]	[m]	[m]		
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 [▶ 588] 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

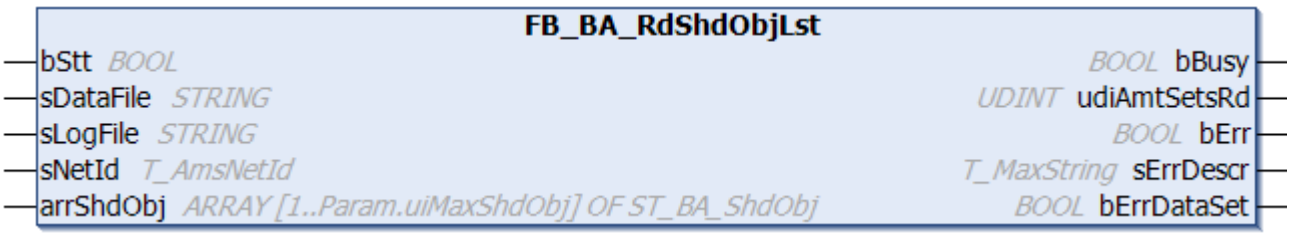
```
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: List of facade elements [▶ 561].

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 [▶ 561]. 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 [▶ 588].
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

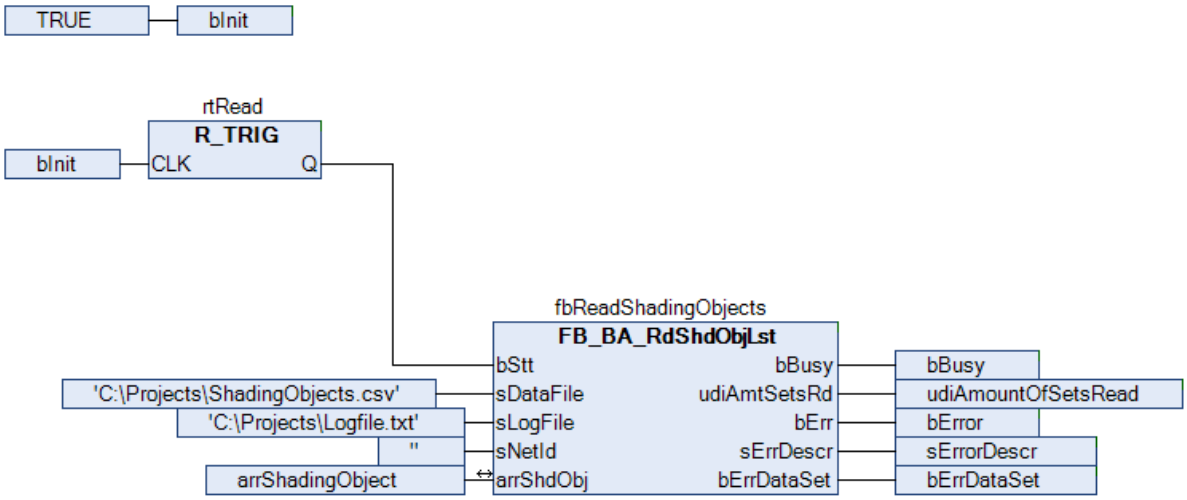
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Number	Description		Type	Begin	End	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Tetragon	(Month)	(Month)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Globe									
4		Text											
5	1	Description	@	0	1	2	-94,75	0	36,06	11	-70,71	11	68,59
6	2	Description	@	0	1	2	-23,33	0	9,9	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

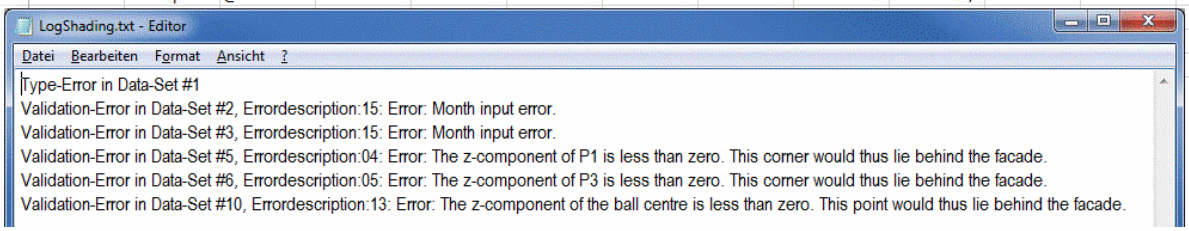
```
PROGRAM ReadShadingObjects
VAR
    bInit                   :   BOOL;
    rtRead                  :   R_TRIG;
    fbReadShadingObjects    :   FB_BA_RdShdObjLst;
    arrShadingObject        :   ARRAY [1..uiMaxShdObj] OF ST_BA_ShdObj;

    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 [▶ 588] 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

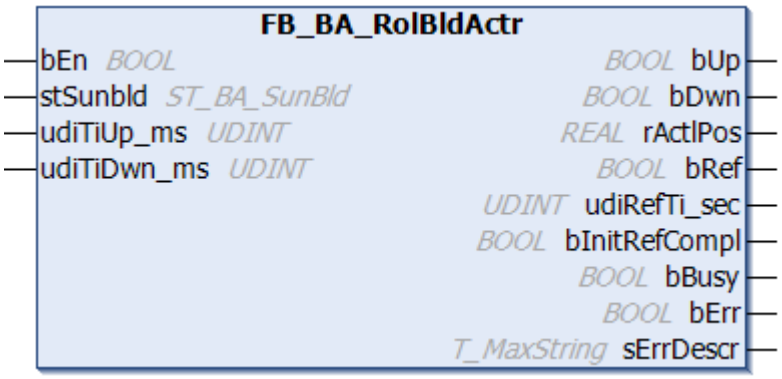
```
arrShdObj: ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;
```

arrShdObj: List of shading objects [▶ 561].

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 [▶ 689] can be used to move the roller shutter to any position. In addition, the positioning telegram stSunBld [▶ 689] 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 [▶ 604].

Structure of the blind positioning telegram stSunBld [▶ 689].

```
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 [▶ 689]. 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 [▶ 689].

udiTiUp_ms: Complete time for driving up [ms].

udiTiDwn_ms: Complete time for driving down in ms.

```
bUp            : BOOL;
bDwn           : BOOL;
rActlPos       : REAL;
bRef           : BOOL;
udiRefTi_sec   : UDINT;
bInitRefCompl  : BOOL;
bBusy          : BOOL;
bErr           : BOOL;
sErrDescr      : T_MAXSTRING;
```

bUp: Control output roller blind up.

bDwn: Control output roller blind down.

rActlPos: Current position in percent.

bRef: The roller blind is referencing, i.e. the output *bUp* is set for the complete travel-up time + 5 s. Only a manual "down" command can move the roller blind in the opposite direction and terminate this mode.

udiRefTi_sec: Referencing countdown display [s].

bInitRefCompl: 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.

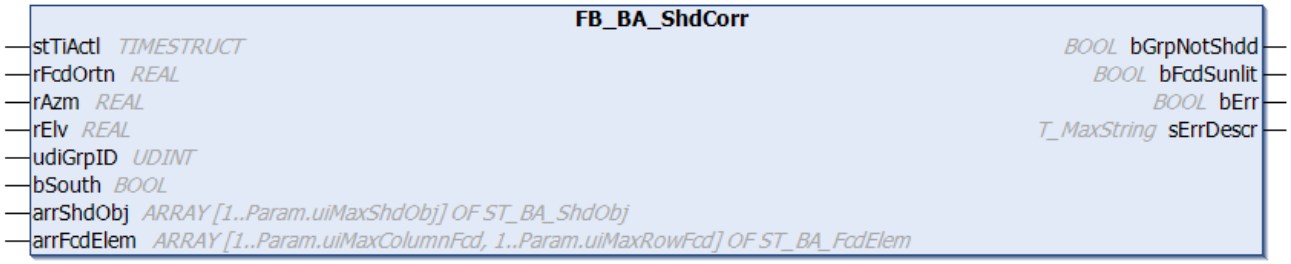
sErrDescr: Contains the error description.

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 [▶ 561] 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 [▶ 561] 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.

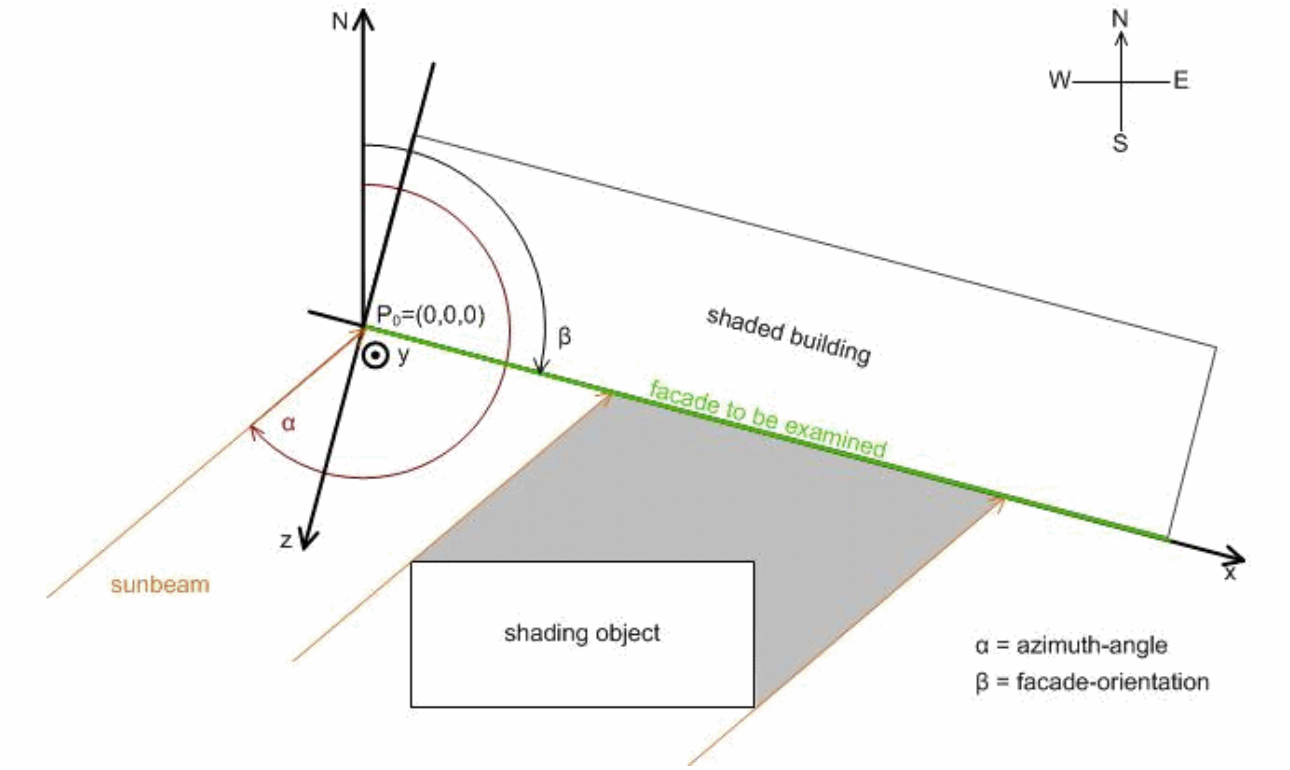
On the basis of the facade orientation (*rFcdOrtn*), the direction of the sun (*rAzm*) and the height of the sun (*rElv*), a calculation can be performed for each corner of a window to check whether this lies in a shaded area. A window group is considered to be completely shaded if all corners are shaded.

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):

Line of sight	Facade orientation
North	β=0°
East	β=90°
South	β=180°
West	β=270°

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°

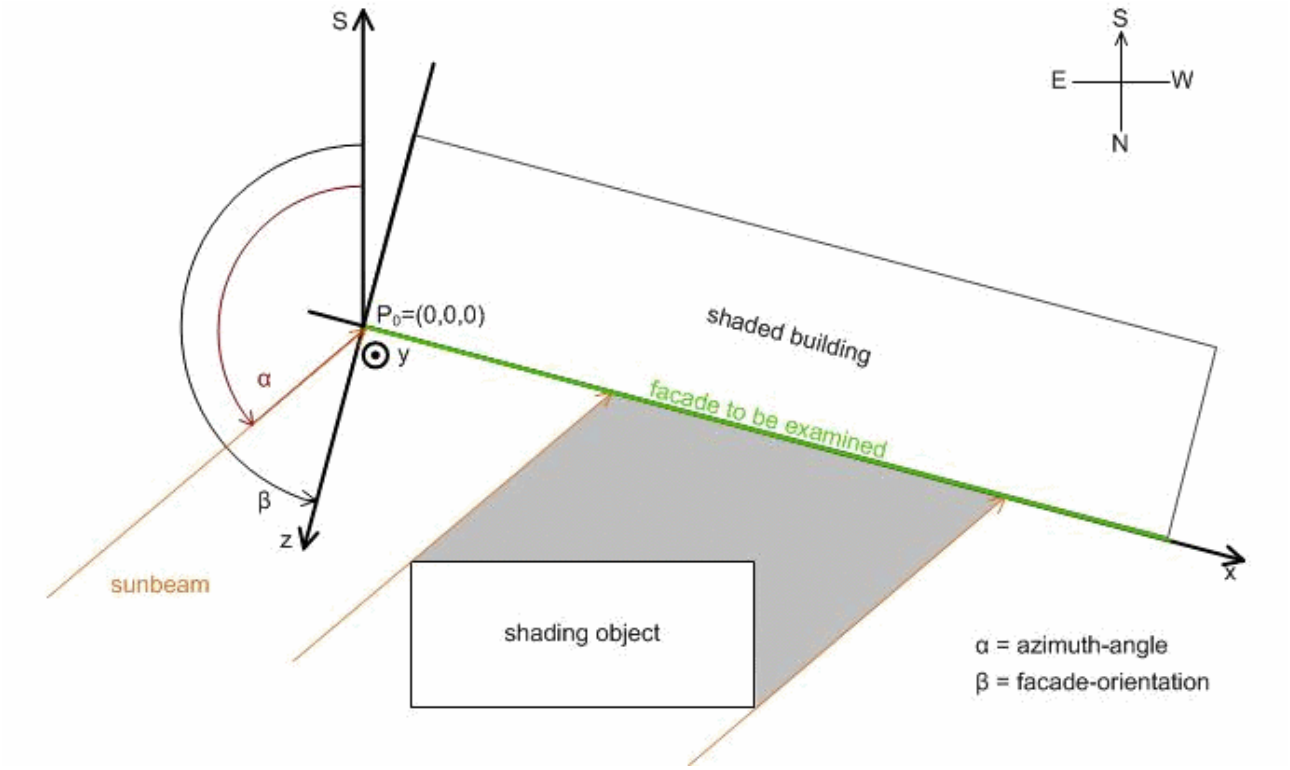


In addition, a calculation is also not required, if the sun has not yet risen, i.e. the solar elevation is below 0°. In both cases the output *bFcdSunlit* is set to FALSE.

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	β=0°
East	β=90°
North	β=180°
West	β=270°

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_ShdObj;
```

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 [▶ 565]. 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 [▶ 561].

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 usiGrpId under consideration is 0.
02: Error: An element of the facade list is invalid. This is specified in the error description sErrDescr as arrFcdElem[nColumn,nRow].

VAR_IN_OUT

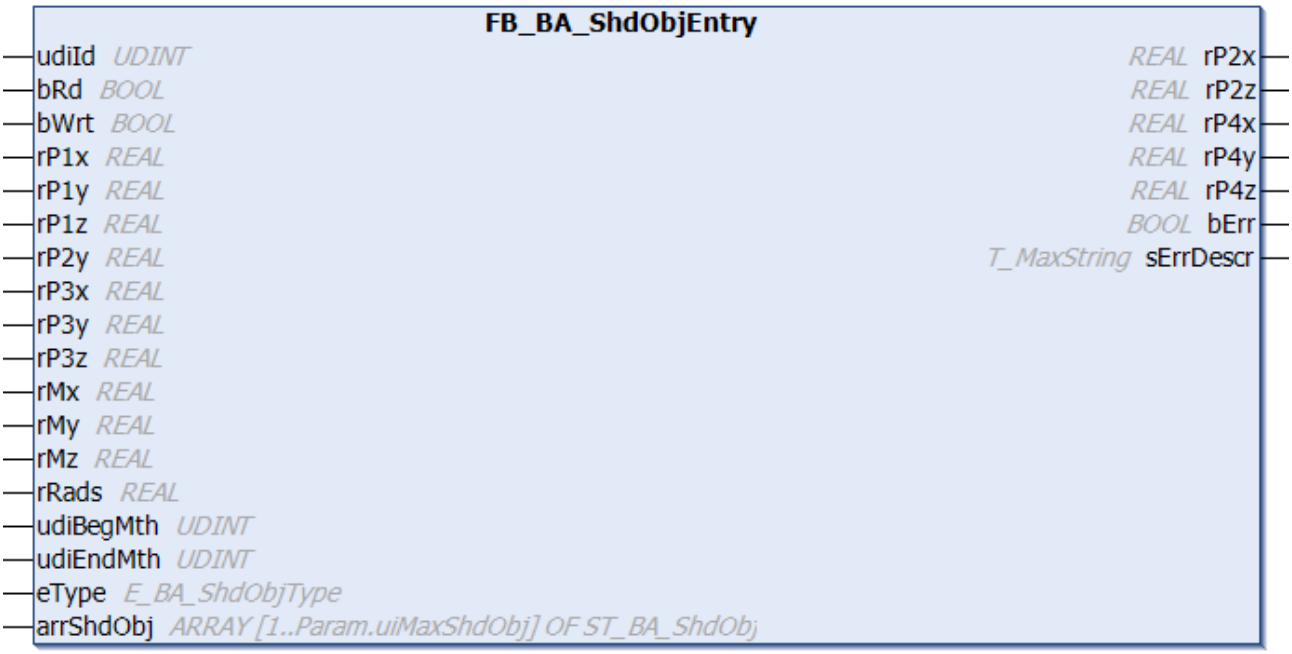
```
arrFcdElem     : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: List of facade elements [▶ 561].

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_ShdObjEntry



This function block serves for the administration of all shading elements in a facade, which are globally saved in a list of shading elements [▶ 561]. 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 [▶ 546].

The shading elements are declared in the global variables:

```
VAR_GLOBAL
     arrShdObj : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Each individual element *arrShdObj[1]* to *arrShdObj [uiMaxShdObj]* carries the information for one shading element (ST_BA_ShdObj [▶ 688]). 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*, r*Rads* 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 [▶ 685] 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 [▶ 688] therefore contains a plausibility bit *bVld*, which forwards precisely this information to the function block FB_BA_ShdCorr [▶ 585] 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 [▶ 685]).

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 [▶ 588]" above.

rP2z: Calculated Z-coordinate of point 2 of the shading element (rectangle) [m]. See "Note [▶ 588]" above.

rP4x: Calculated X-coordinate of point 4 of the shading element (rectangle) [m]. See "Note [▶ 588]" above.

rP4y: Calculated Y-coordinate of point 4 of the shading element (rectangle) [m]. See "Note [▶ 588]" above.

rP4z: Calculated Z-coordinate of point 4 of the shading element (rectangle) [m]. See "Note [▶ 588]" 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 *udiId* is outside the permissible limits 1..*uiMaxShdObj*.
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 *eType* - neither rectangle nor ball.
15: Error: Month input error.

VAR_IN_OUT

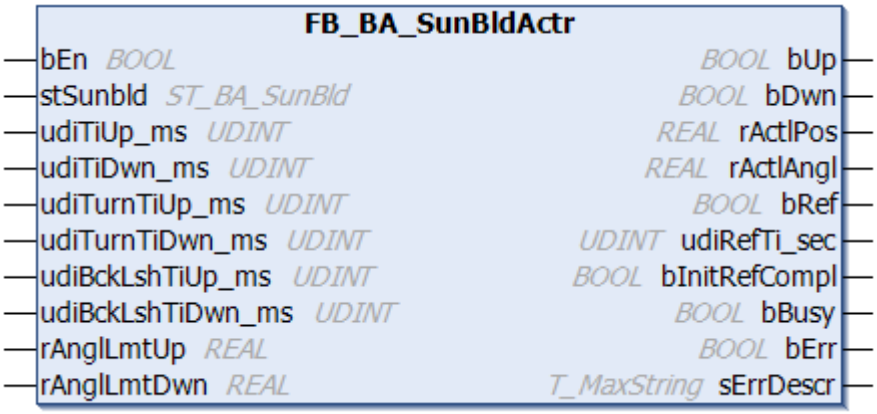
```
arrShdObj : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;
```

arrShdObj: List of shading objects [▶ 561].

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



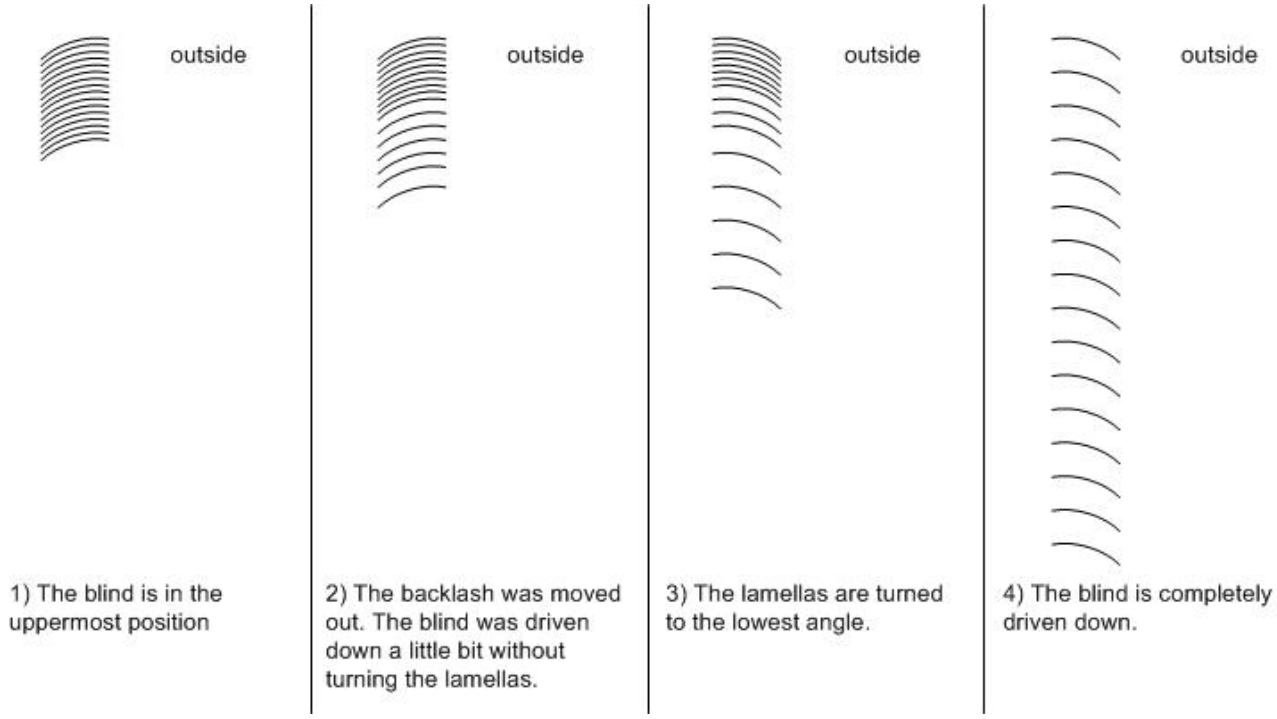
This function block is used for positioning of a louvered blind via two outputs: drive up and drive down. The blind can be driven to any desired (height) position and louvre angle via the positioning telegram stSunBld [▶ 689]. On top of that, the positioning telegram stSunBld [▶ 689] also contains manual commands with which the blind can be moved individually to certain positions. These manual commands are controlled by the function block FB_BA_SunBldSwi [▶ 604].

Structure of the blind positioning telegram stSunBld [▶ 689].

```
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):

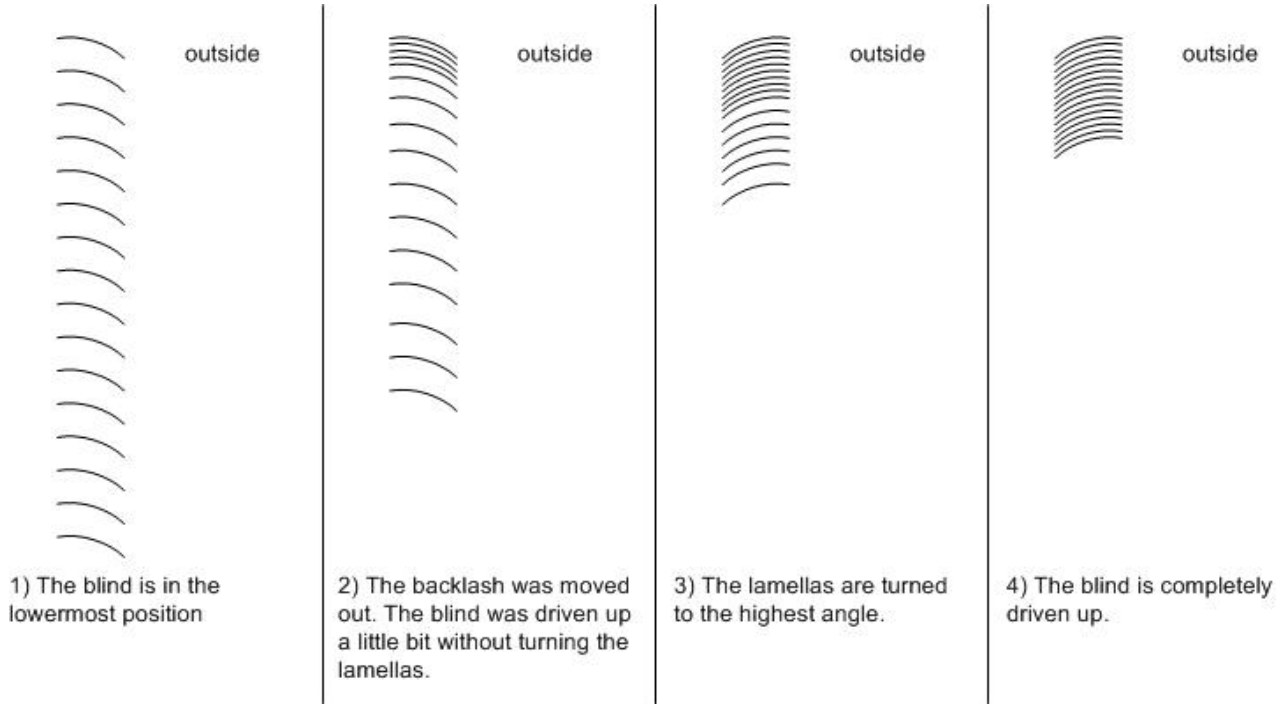
Downward travel profile:



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 louvre 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 louvres start to turn from the "Outward low point" to the "Inward low point". During this time the louvre 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 louvres from the "Outward low point" to the "Inward low point". This time should be entered as *udiTurnTiUp_ms* [ms] at the function block.

Parameterization

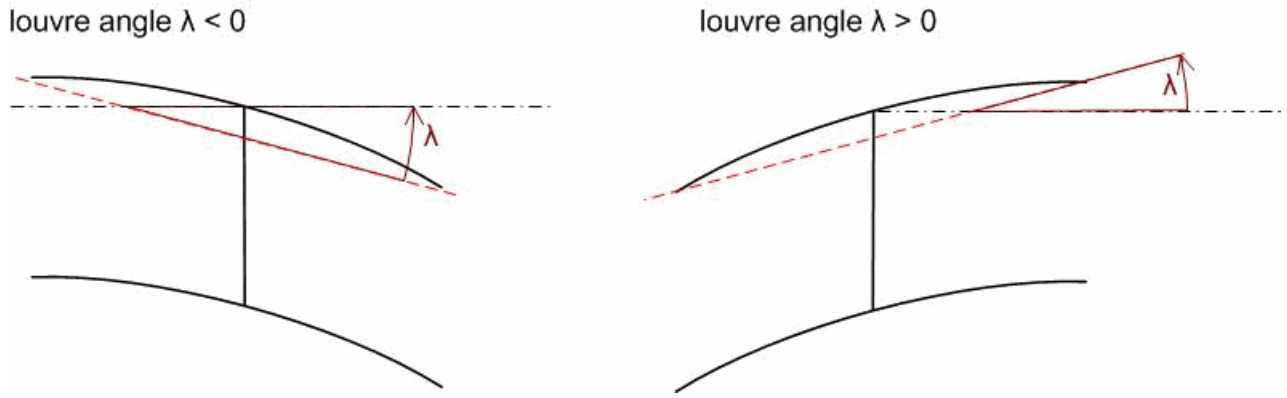
For the calculation of the (height) position and the louvre angle, the following times now have to be determined for both the upward and downward movement:

- 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 louvre angle after turning upwards (*rAnglLmtUp* [°])
- the lowest louvre angle after turning downwards (*rAnglLmtDwn* [°])

The louvre angle λ is defined by a notional straight line through the end points of the louvre to the horizontal.



Functioning

As a rule, the function block controls the blind based on the information from the positioning telegram stSunBld [▶ 689]. 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 +/-4°.

VAR_INPUT

```
bEn                : BOOL;
stSunbld           : ST_BA_SunBld;
udiTiUp_ms         : UDINT;
udiTiDwn_ms        : UDINT;
udiTurnTiUp_ms     : UDINT;
udiTurnTiDwn_ms    : UDINT;
udiBckLshTiUp_ms   : UDINT;
udiBckLshTiDwn_ms  : UDINT;
rAnglLmtUp         : REAL;
rAnglLmtDwn        : 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 [▶ 689]).

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].

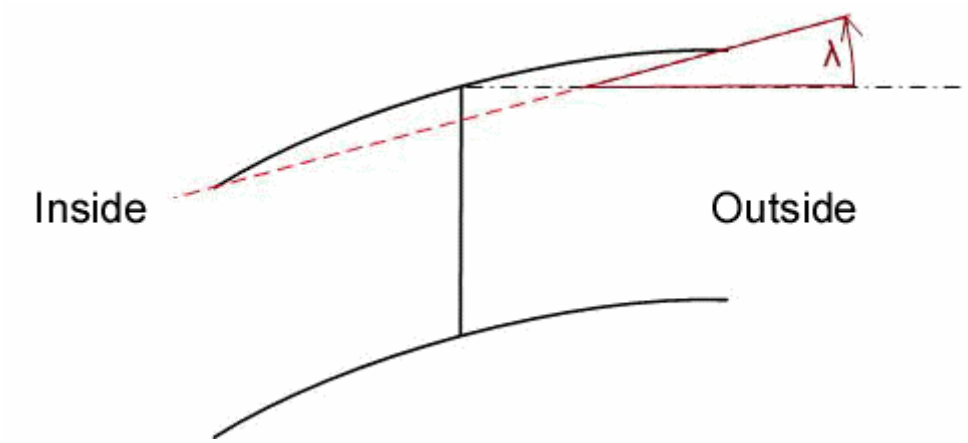
udiBckLshTiUp_ms: Time to traverse the backlash in the upward direction [ms]. This input is internally limited to a minimum value of 0.

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.

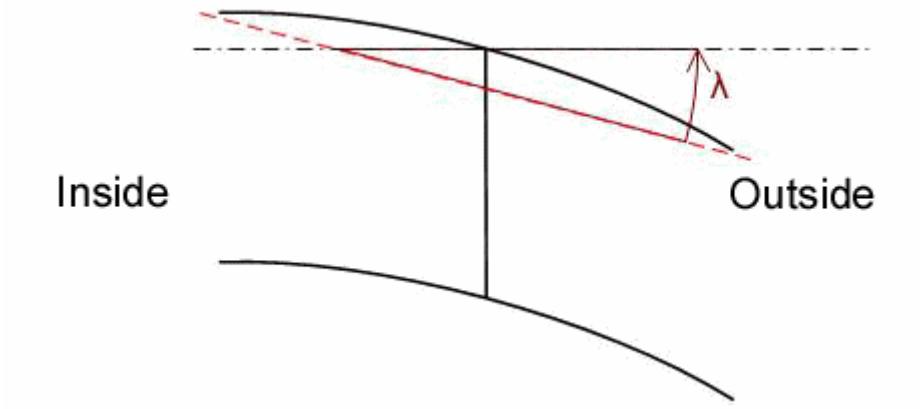
The louvre angle λ, as defined above, is then typically greater than zero.



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].

bInitRefCompl: 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.

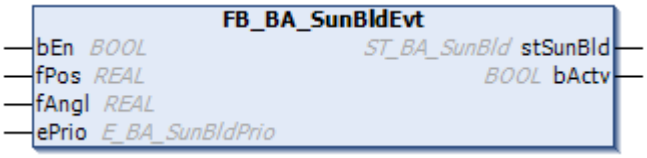
sErrDesc: Contains the error description.

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<=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 [▶ 689] 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 [▶ 689] is reset and the positions for height and angle are set to "0". The priority function block (e.g. FB_BA_SunBldPrioSwi4 [▶ 599]) 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 [▶ 689]. 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;
```

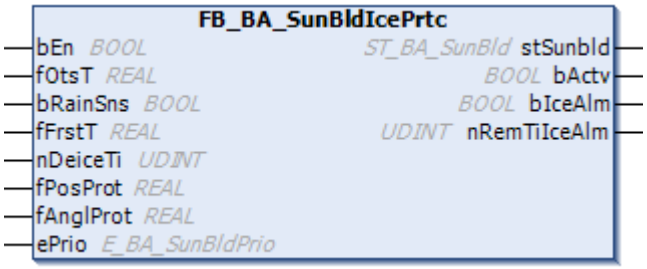
bActv: Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 689] and is solely used to indicate whether the function block sends an active telegram.

stSunBld: Output structure of the blind positions, see ST_BA_SunBld [▶ 689]

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 [▶ 554]) 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 [▶ 689] 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 [▶ 689].

bActv: Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 689] 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 [▶ 689] to the output telegram stOut [▶ 689] 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*)

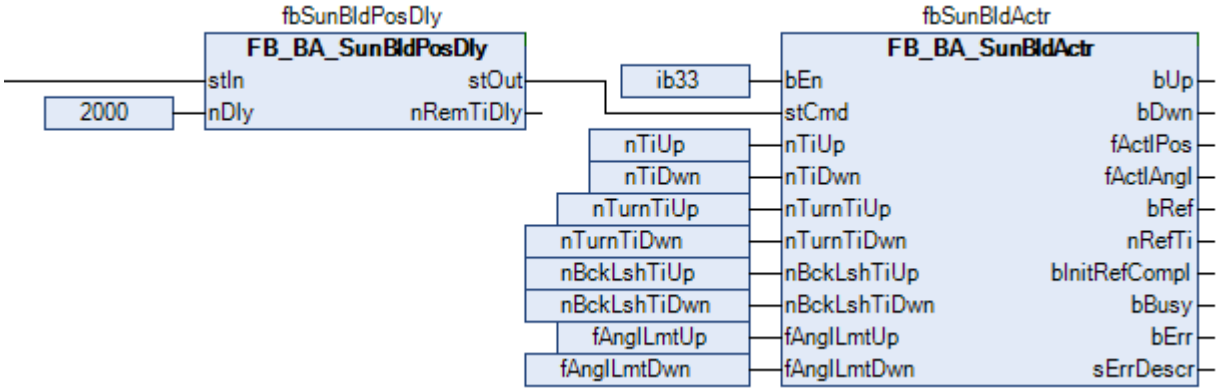
The output telegram *stOut* is always a direct copy of the input telegram *stIn*. In these three cases, however, the output telegram *stOut* is set for the time *udiDly_ms* [ms].

This ensures that the blind controlled via the function block FB_BA_SunBldActr [▶ 591] 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 [▶ 689].

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 [▶ 689].

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 [▶ 689].

Structure of the blind positioning telegram ST_BA_Sunbld [▶ 689].

```
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 [▶ 591] or the roller blind function block FB_BA_RolBldActr [▶ 583] 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

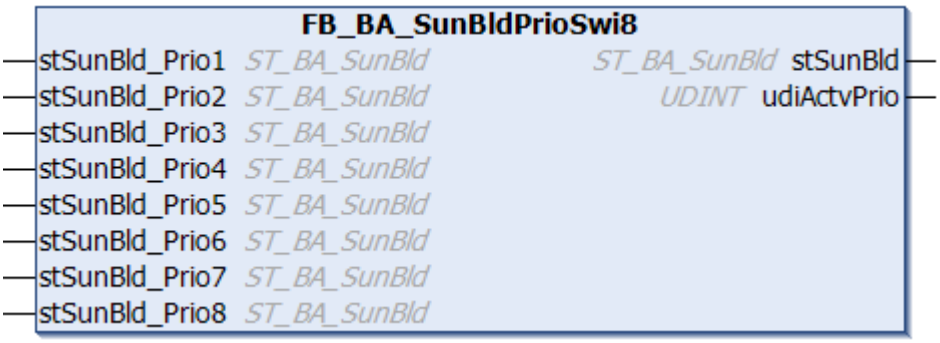
```
stSunBld    : ST_BA_SunBld;
udiActvPrio : UDINT;
```

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 [▶ 689].

Structure of the blind positioning telegram ST_BA_Sunbld [▶ 689].

```
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 [▶ 591] or the roller blind function block FB_BA_RolBldActr [▶ 583] 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

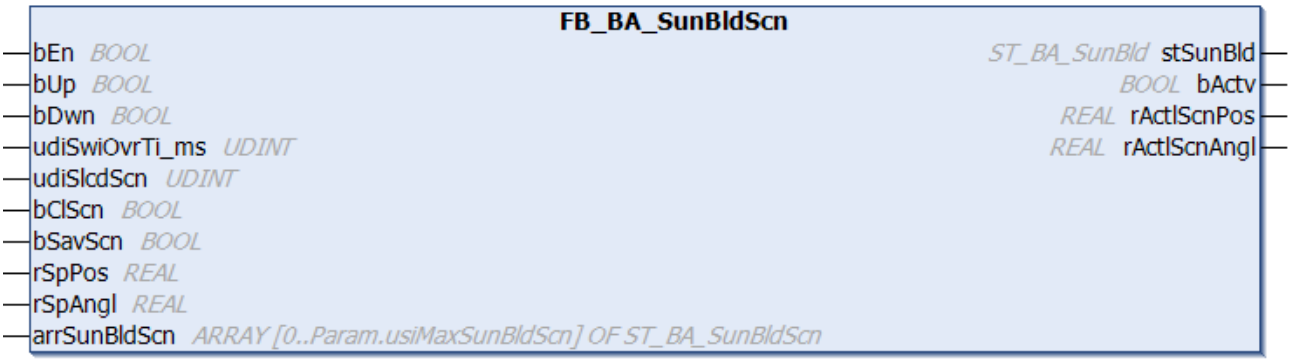
```
stSunBld    : ST_BA_SunBld;
udiActvPrio : UDINT;
```

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 [▶ 604] by a scene memory and a call function. The blind control FB_BA_SunBldActr [▶ 591] or the roller blind control FB_BA_RolBldActr [▶ 583] 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 [▶ 689].

```
TYPE ST_BA_SunBld:
STRUCT
    rPos        : REAL;
    rAngl       : REAL;
    bManUp      : BOOL;
    bManDwn     : BOOL;
```

```
    bManMod     : BOOL;
    bActv       : BOOL;
END_STRUCT
END_TYPE
```

Operation

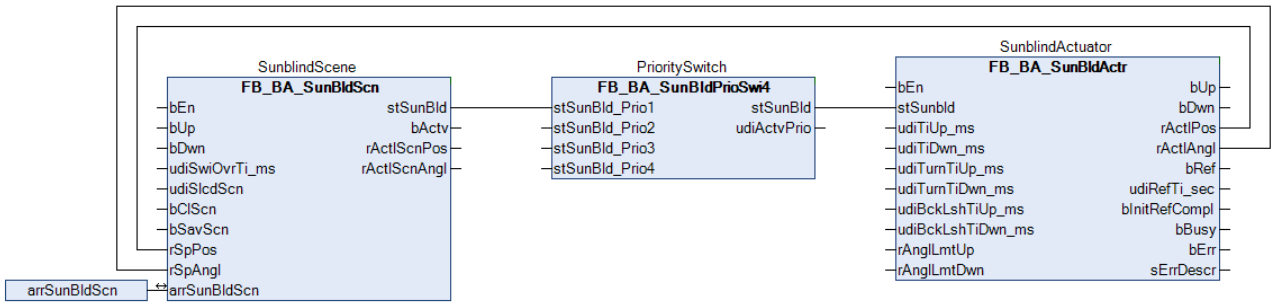
In manual mode, the function block controls the blind function block FB_BA_SunBldActr [▶ 591] or the roller shutter function block FB_BA_RolBldActr [▶ 583] via the command inputs *bUp* and *bDwn*; *bUp* has priority. The commands are passed on to the respective commands *bManUp* and *bManDwn* of the 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 [▶ 599] or FB_BA_SunBldPrioSwi8 [▶ 600]) 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.

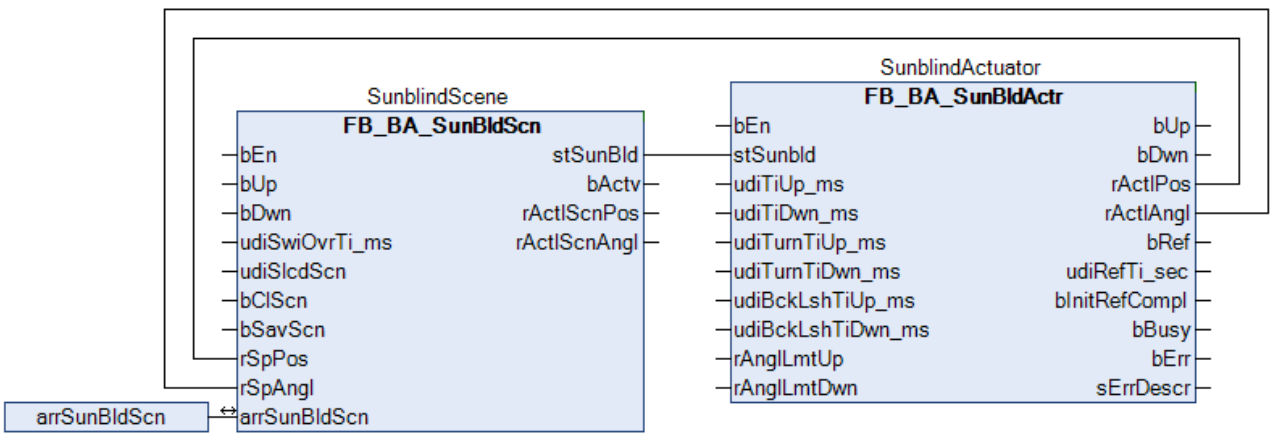
Linking to the blind function block

Like the "normal" manual mode function block FB_BA_SunBldSwi [▶ 604], the scene selection function block can be connected either via an upstream priority control FB_BA_SunBldPrioSwi4 [▶ 599] or FB_BA_SunBldPrioSwi8 [▶ 600], or directly via the blind function block. The connection is established via the positioning telegram ST_BA_Sunbld [▶ 689]. 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 [▶ 689], 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 [▶ 689] 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 [▶ 591] or FB_BA_RolBldActr [▶ 583] 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 [▶ 689].

bActv: Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 689] 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

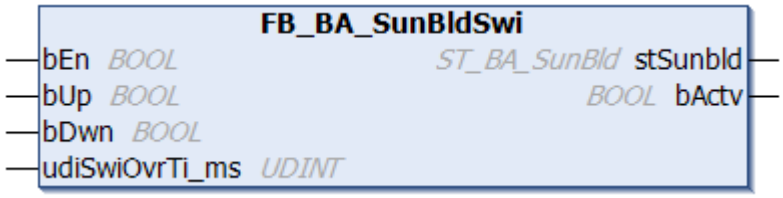
```
arrSunBldScn : ARRAY[0..Param.usiMaxSunBldScn] OF ST_BA_SunBldScn;
```

arrSunBldScn: Table with the scene entries of the type ST_BA_SunBldScn [▶ 690].

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 [▶ 591] or roller shutter FB_BA_RolBldActr [▶ 583] in manual mode. The connection takes place via the positioning telegram ST_BA_Sunbld [▶ 689] either directly or with an additional priority controller.

Structure of the blind positioning telegram ST_BA_Sunbld [▶ 689].

```
TYPE ST_BA_SunBld:
STRUCT
     rPos         : REAL;
     rAngl        : REAL;
     bManUp       : BOOL;
     bManDwn      : BOOL;
     bManMod      : BOOL;
     bActv        : BOOL;
END_STRUCT
END_TYPE
```

Operation

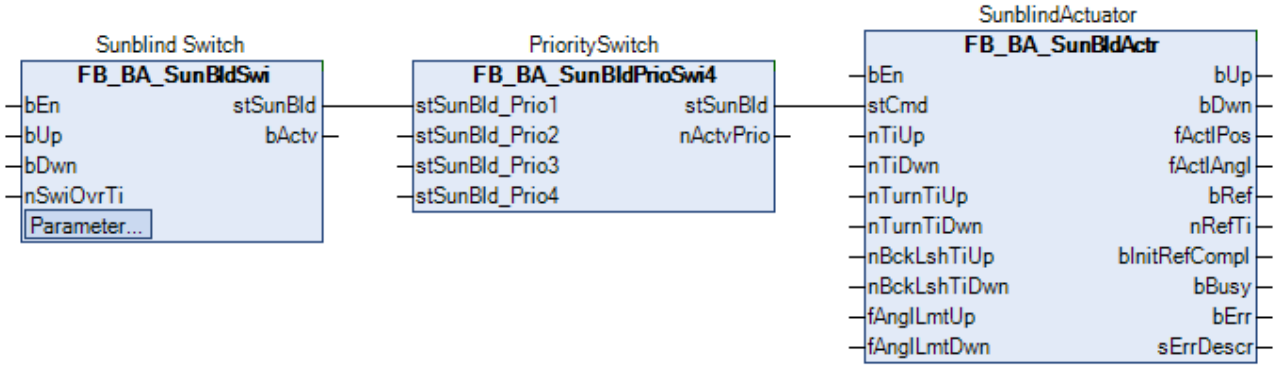
In manual mode, the function block controls the blind function block FB_BA_SunBldActr [▶ 591] or the roller shutter function block FB_BA_RolBldActr [▶ 583] via the command inputs *bUp* and *bDwn*; *bUp* has priority. The commands are passed on to the respective commands *bManUp* and *bManDwn* of the 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 [▶ 599] or FB_BA_SunBldPrioSwi8 [▶ 600]) 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.

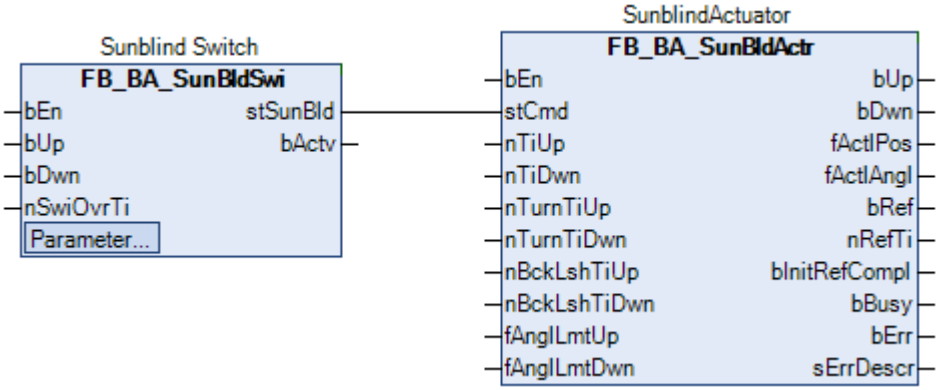
Linking to the blind function block

The manual mode function block can be connected either via an upstream priority control FB_BA_SunBldPrioSwi4 [▶ 599] or FB_BA_SunBldPrioSwi8 [▶ 600], or directly at the blind function block. The connection is established via the positioning telegram ST_BA_Sunbld [▶ 689].

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 [▶ 689], 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 [▶ 689] 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;
```

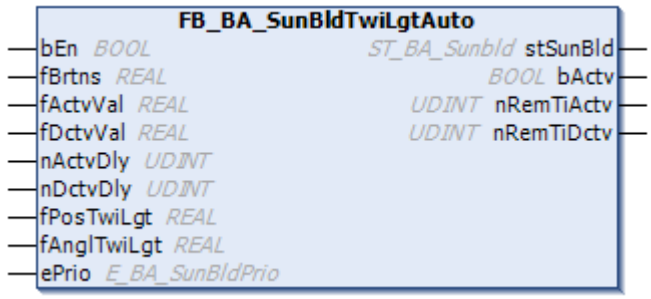
stSunBld: Positioning telegram, see ST_BA_SunBld [▶ 689].

bActv: Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 689] 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 [▶ 689]. 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 [▶ 689] 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 [▶ 689] 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 [▶ 689].

bActv: Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 689] 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.

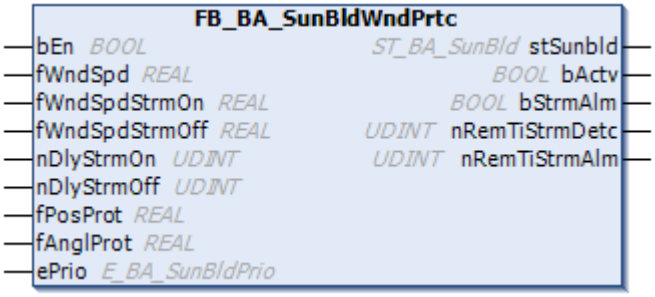
udiRemTiDctv_sec: Shows the time remaining [s] after exceeding of the switching value *rDctvVal* until automatic mode is disabled. This output is 0 as long as no countdown of the time is taking place.

> 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 [▶ 554]) 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 [▶ 689] 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 [▶ 689].

bActv: Corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 689] 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.

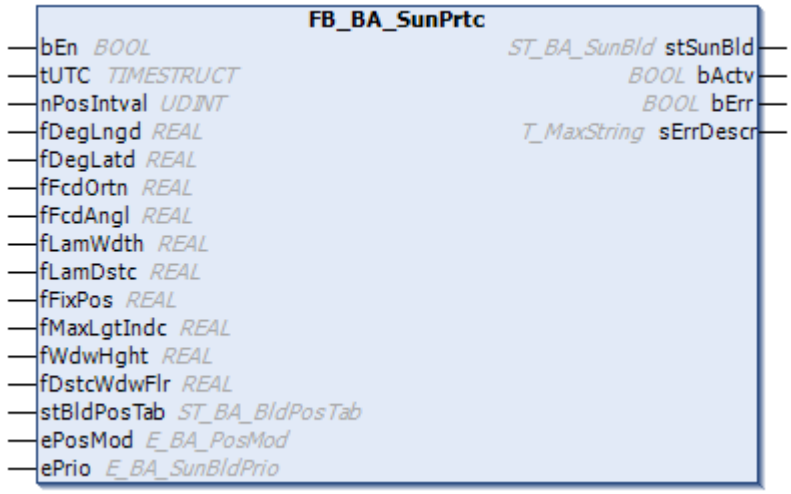
udiRemTiStrmAlm_sec: As soon as the storm alarm is initiated, this second counter initially constantly indicates the deactivation time delay of the storm alarm *udiDlyStrmOff_sec*. If the measured wind speed *rWndSpd* falls below the deactivation limit value *rWndSpdStrmOff*, the seconds to the all-clear 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 [▶ 686]). See also description of FB_BA_BldPosEntry [▶ 561].
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 [▶ 689] 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 [▶ 618] 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	β=0°
East	β=90°
South	β=180°
West	β=270°

The following applies for the southern hemisphere:

Line of sight	Facade orientation
South	β=0°
East	β=90°
North	β=180°
West	β=270°

rFcdAngl: Facade inclination [°]. See facade inclination [▶ 559].

rLamWdth: Width of the louvres in mm, see Louvre adjustment [▶ 556].

rLamDstc: Louvre spacing in mm, see Louvre adjustment [▶ 556].

rFixPos: Fixed (constant) shutter height [0..100%]. Applies if *ePosMod* = *ePosModFix* (see enumerator E_BA_PosMod [▶ 685]).

rMaxLgtIndc: Maximum desired light incidence in mm measured from the outside of the wall (see Height adjustment [▶ 558]). 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 [▶ 685]).

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 [▶ 685]). For a more detailed description please refer to FB_BA_BldPosEntry [▶ 561].

ePosMod: Selection of the positioning mode, see enumerator E_BA_PosMod [▶ 685].

VAR_OUTPUT

```
stSunBld     : ST_BA_SunBld;
bActv        : BOOL;
bErr         : BOOL;
sErrorDescr  : T_MAXSTRING;
```

stSunBld: Output structure of the blind positions, see ST_BA_SunBld [▶ 689]

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 *rFcdAngl* is outside the valid range of -90°..90°.
05: Error: The value for the louvre spacing (*rLamDstc*) is greater than or equal to the value for the louvre width (*rLamWdth*). 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 *rLamWdth* is zero.
07: Error: The value entered for the louvre spacing *rLamDstc* is zero.
08: Error: The value entered for the fixed blind height (*rFixPos*) is greater than 100 or less than 0. At the same time, "fixed blind height" positioning is selected - *ePosMod*=ePosModFix.
09: Error: The bit "values valid" (*bVld*) in the positioning table *stBldPosTab* is not set - invalid values: see FB_BA_BldPosEntry. At the same time, "Table" positioning is selected - *ePosMod*=ePosModTab.
10: Error: The value entered for the maximum required light incidence *rMaxLgtIndc* is less than or equal to zero. At the same time, "maximum light incidence" is selected - *ePosMod*=ePosModMaxIndc.
11: Error: The value entered for the window height *rWdwHght* is less than or equal to zero. At the same time, "maximum light incidence" is selected - *ePosMod*=ePosModMaxIndc.
12: Error: The distance between lower window edge and floor *rDstcWdwFlr* that was entered is less than zero. At the same time, "maximum light incidence" is selected - *ePosMod*=ePosModMaxIndc.
13: Error: An invalid positioning mode is entered at input *ePosMod*.

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

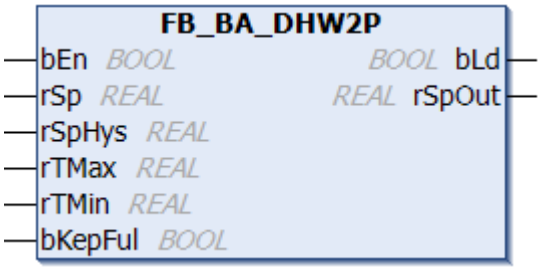
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.3 Provision of hot water

Function blocks

Name	Description
FB_BA_DHW2P [▶ 612]	Charge control for a hot water tank via an on-off controller.
FB_BA_LglPrev [▶ 614]	Function block for disinfecting service water and destroying legionella.

6.1.2.3.2.1.3.1 FB_BA_DHW2P

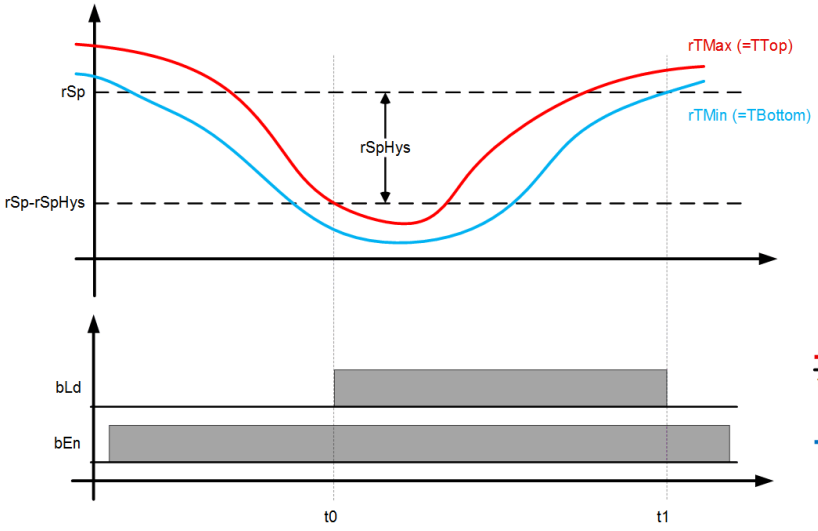


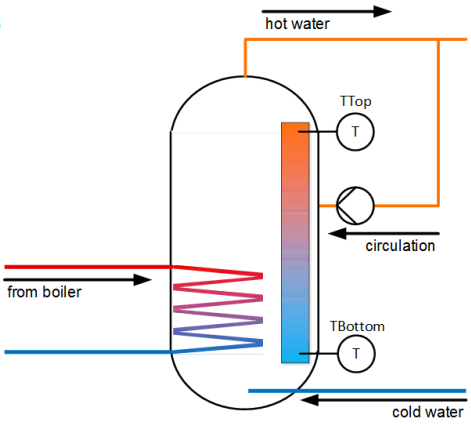
This function block controls the heating of a hot water tank via an on-off controller. Tank heating is activated at input *bEn*. If tank heating is active the output *bLd* is TRUE. The variable *rSp* is used to transfer the setpoint for the hot water temperature to the function block. At input *rTMin* 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

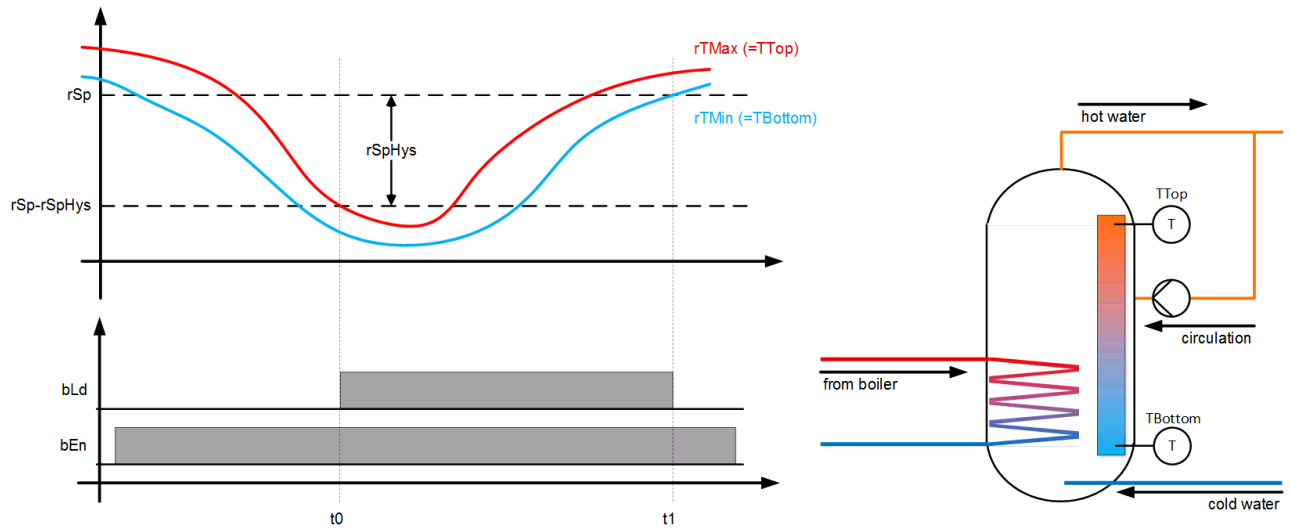
Charging is requested if *rTMax* falls below the value of *rSp-rSpHys*. The charge request is disabled if *rTMin* is above the setpoint of *rSp*.
Due to the fact that the sensor at the top generally measures the highest temperature, the heating is not switched on until the hot water tank has been discharged.





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 = *rTMax* is used to request *bLd*, *rTMin* to switch off

TRUE = *rTMin* alone controls switching on/off of *bLd*

VAR_OUTPUT

```
bLd       : BOOL;
rSpOut    : REAL;
```

bLd: Enable charging mode.

rSpOut: Setpoint transfer to charging circuit:

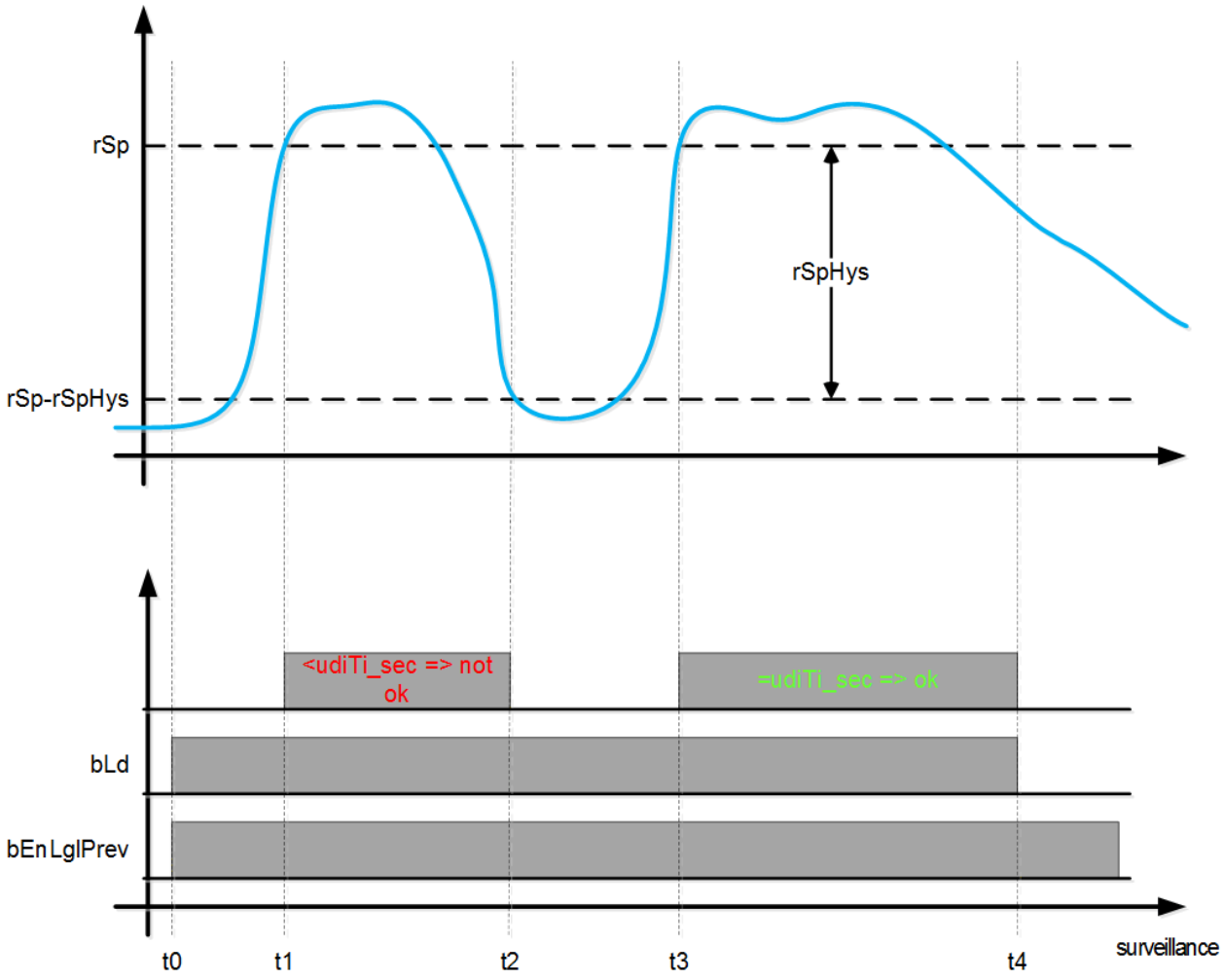
- *rSpOut* = *rSp* (input) if the function block is enabled
- *rSpOut* = 0 if the function block is not enabled

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

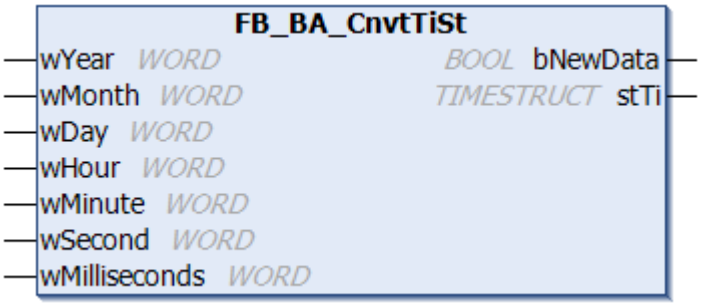
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.4 System

Function blocks

Name	Description
FB_BA_CnvtTiSt [▶ 616]	Conversion year, month, day, hour, minute and second in time structure
FB_BA_ExtTiSt [▶ 617]	Conversion time structure in year, month, day, hour, minute and second
FB_BA_GetTime [▶ 618]	Internal clock with time information - can be synchronized with system time
FB_BA_SetTime [▶ 620]	Setting the system time
FB_BA_WrtPersistDat [▶ 621]	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.

> 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

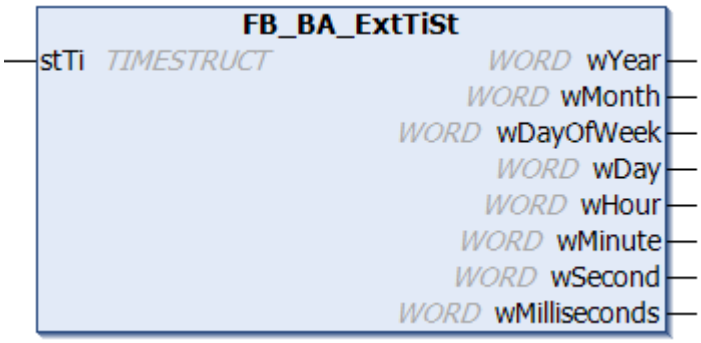
```
stTi      : TIMESTRUCT;
```

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).

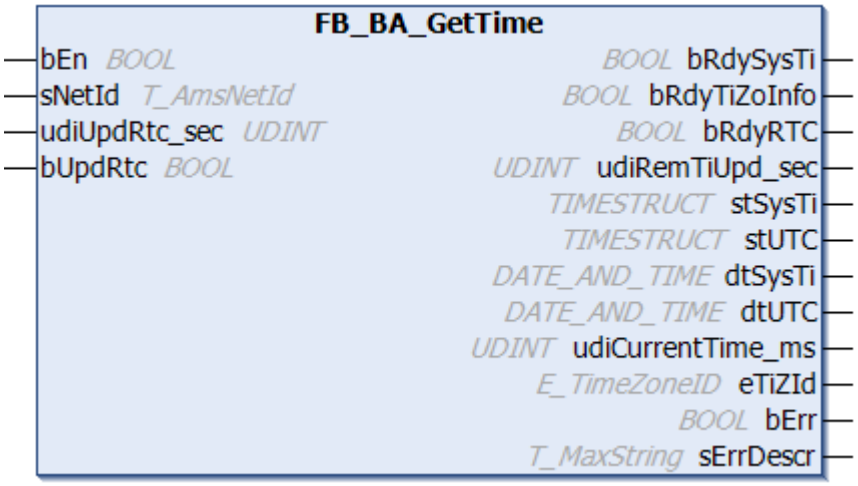
wSecond: The seconds (0..59).

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 [▶ 617] into the components day, month, hour, minute etc.

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 [▶ 617] into its components: day, month, hour, minute etc.

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 [▶ 617] into its components: day, month, hour, minute etc.

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 (*FB 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.

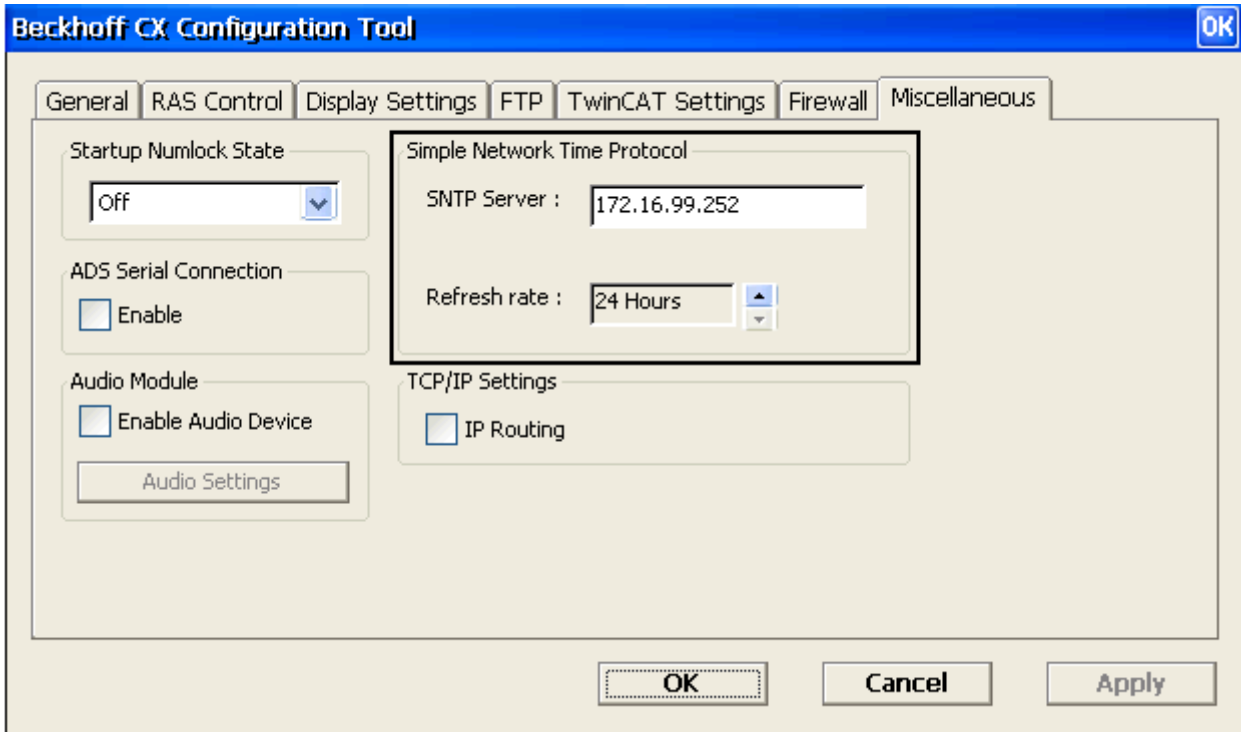
Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	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_CnvtTiSt [▶ 616], 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 [▶ 622])

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).

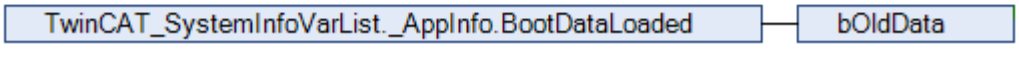
Sample in ST:

```
PROGRAM Example_ST
VAR
    bOldData    :   BOOL;
END_VAR
```

```
bOldData:=TwinCAT_SystemInfoVarList._AppInfo.OldBootData;
```

Sample in CFC:

```
PROGRAM Example_CFC
VAR
    bOldData    :   BOOL;
END_VAR
```



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*.

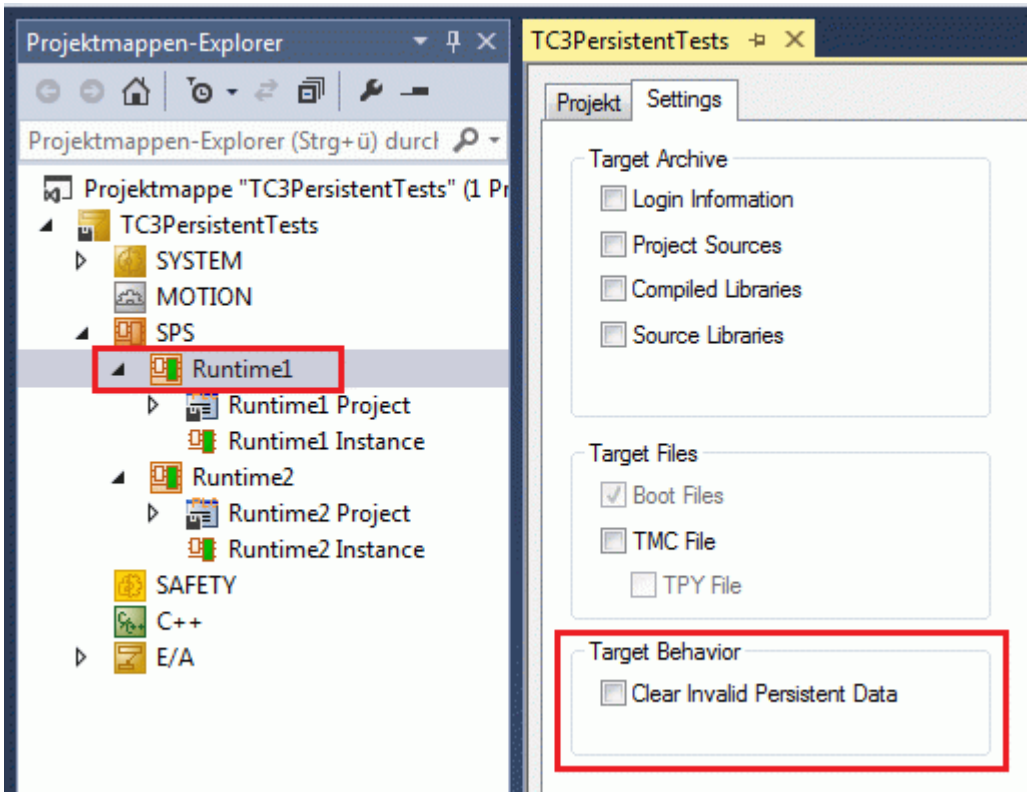
When the system is restarted, i.e. when switching to run mode, this file is read and then saved as Port_xxx.bootdata-old.

If the file Port_xxx.bootdata-old already exists, it is overwritten.

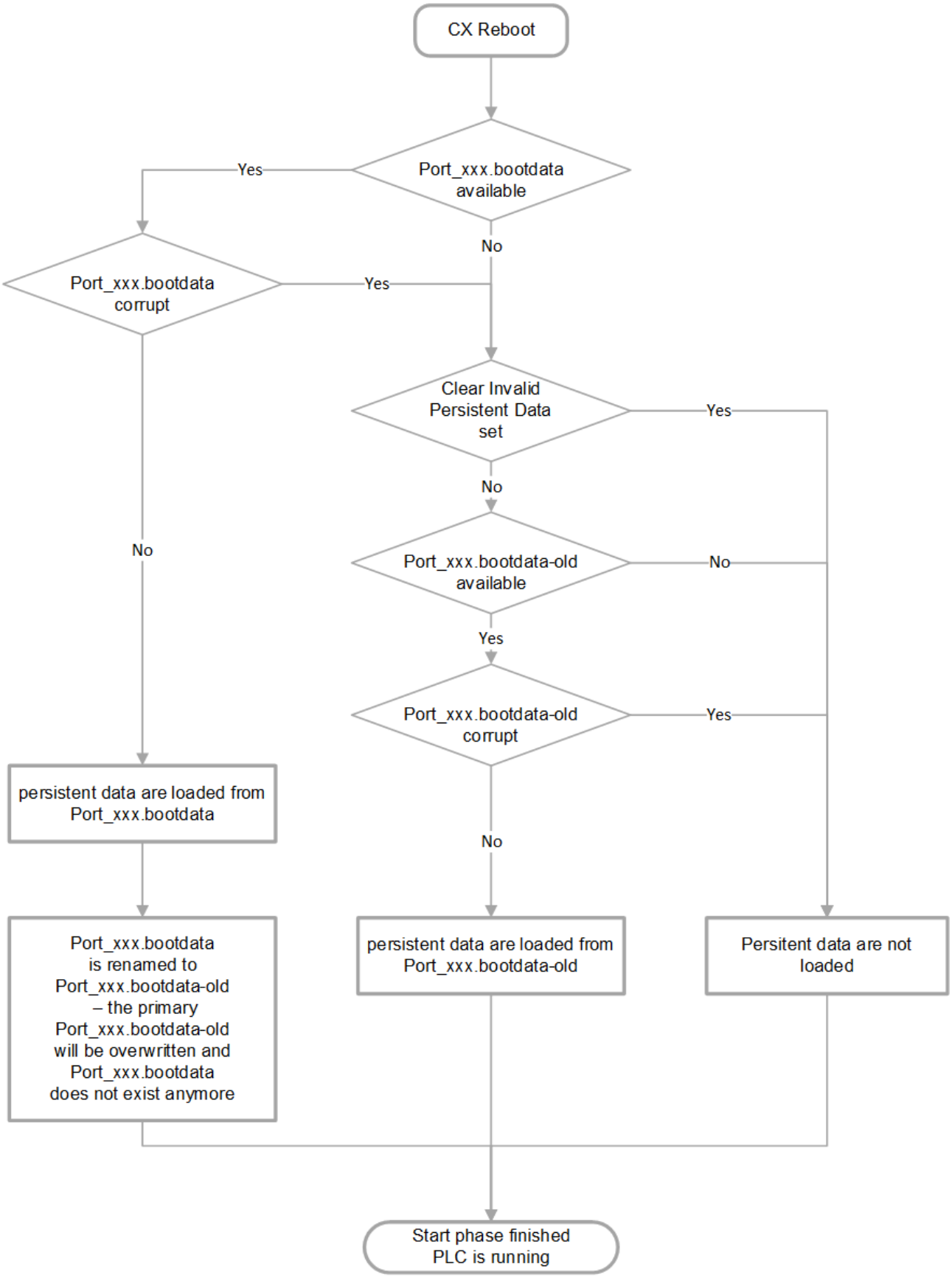
The original file Port_xxx.bootdata then no longer exists. It is created again automatically when switching to Stop mode or by the function block *FB_WritePersistentData* from the TC2_Utilities library.

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 *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 to 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 v3.1.4024.7	Tc3_BA from v1.1.6.0

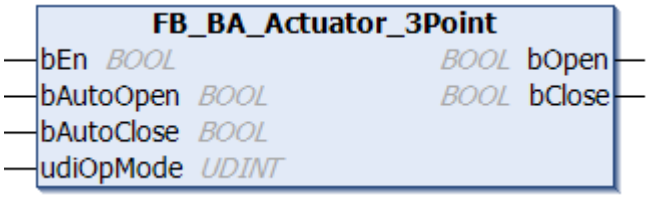
6.1.2.3.2.1.5 Universal

6.1.2.3.2.1.5.1 Actuators

Function blocks

Name	Description
FB_BA_Actuator_3Point [▶ 626]	Control of three-point dampers or valves
FB_BA_Anlg3Pnt [▶ 627]	Analog value for three-point converters
FB_BA_AntBlkg [▶ 628]	Blocking protection for pump or actuators
FB_BA_Motor1St [▶ 629]	Control of single-speed drives
FB_BA_Motor2St [▶ 630]	Control of two-speed drives
FB_BA_PWM [▶ 631]	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
rOut = 0	FALSE	permanently TRUE
rOut = 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 *bCls* is TRUE for the time udiTiOpn_ms

VAR_OUTPUT

```
bCls    : BOOL;
bOpn    : BOOL;
rPos    : REAL;
```

bCls: 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 pumps or actuators from blocking after long periods without movement by outputting a switch-on pulse.
The maximum duration of the standstill until a pulse is output 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 aggregate. The length of the pulse is parameterized with *udiTiImplLngt_sec*. The input *bExe* should be used if the anti-blocking protection pulses are to be issued cyclically based on a 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 of the 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 aggregate.

udiTiImplLngt_sec: Length of the anti-blocking protection pulse [s] at *bQ*.

VAR_OUTPUT

```
bQ                : BOOL;
udiRTiOffMin_sec  : UDINT;
udiRTiImplLngt_sec : UDINT;
```

bQ: Pulse output.

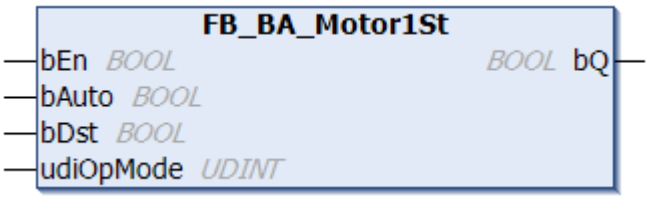
udiRTiOffMin_sec: Remaining time [s] before the next pulse is issued in the absence of movement.

udiTiImplLngt_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

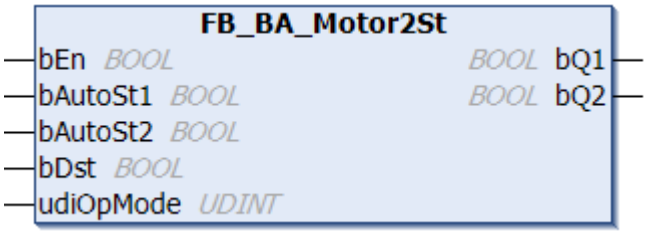
```
bQ  : BOOL;
```

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;
```

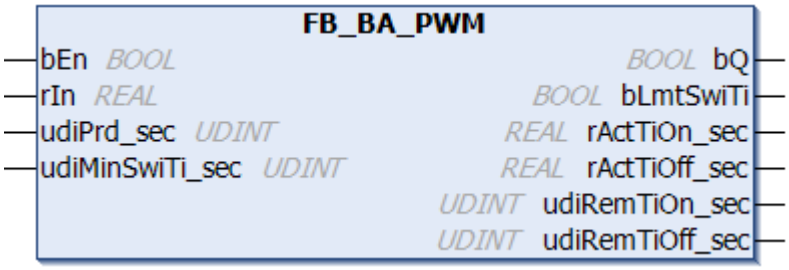
bQ1: Control output stage 1.

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 *rrActTiOn_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

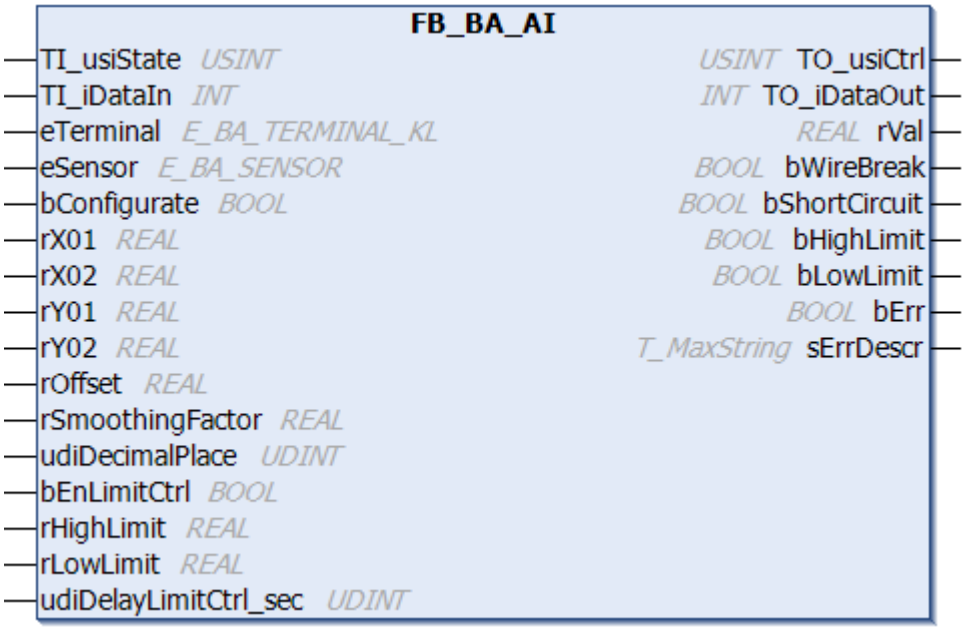
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.2 Analog inputs/outputs

Function blocks

Name	Description
FB_BA_AI [▶ 632]	Acquisition of analog input signals
FB_BA_AO [▶ 635]	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

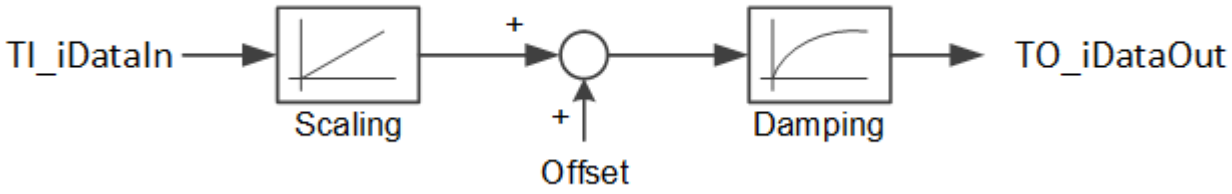
The first step when using this function block is to select the corresponding terminal type eTerminal. Correct functioning of the function block is only guaranteed if the terminal selected with the *eTerminal* variable matches the terminal actually inserted and linked.
With the terminals for resistance 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 *bConfigurate* writes the terminal settings to the terminal using the variables *TO_usiCtrl* and *TO_iDataOut*.

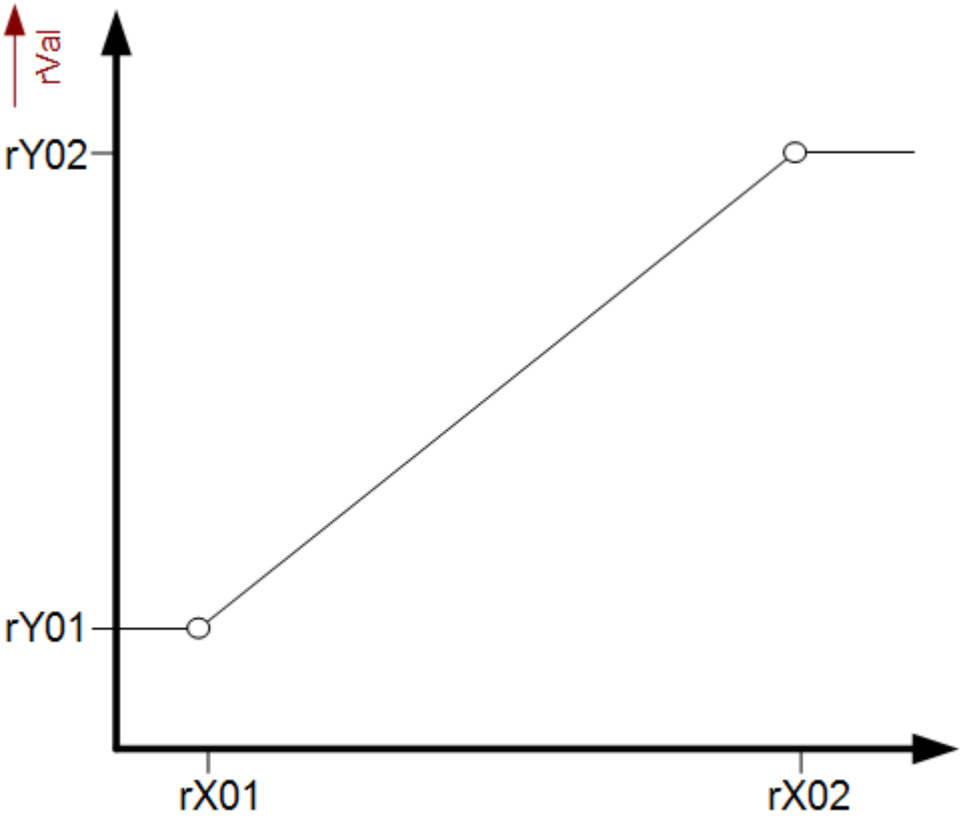
Measured value scaling

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_Chrct02 [▶ 670]. (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:

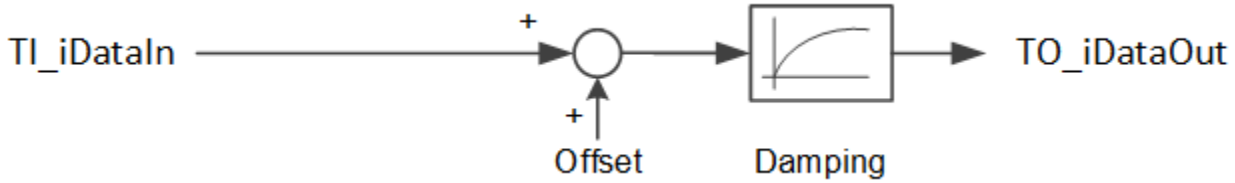


Scaling:



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 *bWireBrake*, 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;
bConfigurate          : 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).

bConfigurate: 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.

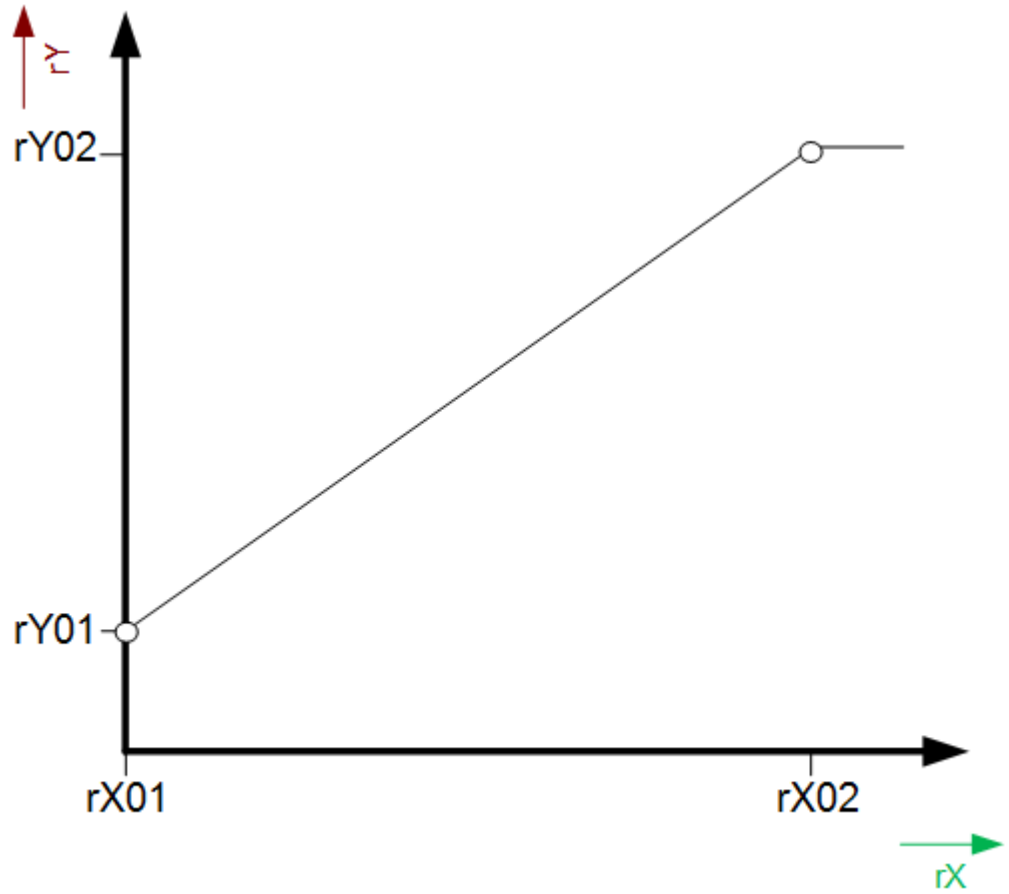
Error description
01: Error: Incorrect scaling parameter *rX01*/*rX02*/*rY01*/*rY02*
02: Error: Check the terminal configuration KL32xx *eTerminal*/*eSensor*/*TI_usiState*/*TI_iDataIn*/*TO_usiCtrl*/*IO_iDataOut*

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

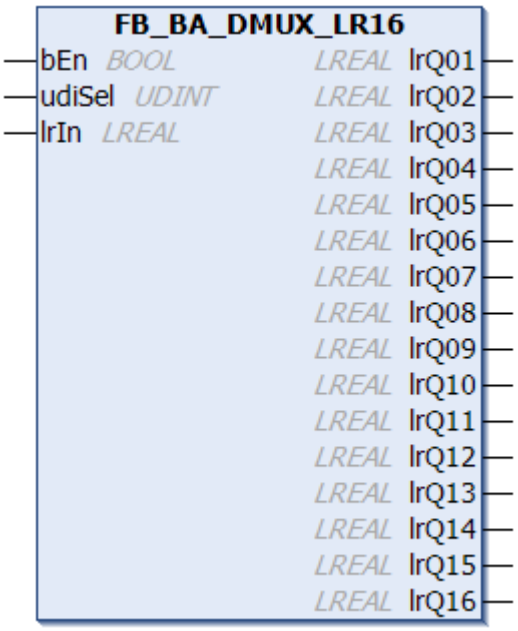
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.3 General control functions

Function blocks

Name	Description
FB_BA_DMUX_XX [▶ 637]	Demultiplexer function blocks
FB_BA_MMUX_XX [▶ 640]	The function blocks activate an input value on the output, depending on a selector and the corresponding input selector condition
FB_BA_MUX_XX [▶ 643]	Multiplexer function blocks
FB_BA_PrioSwi_XX [▶ 645]	Priority switch
FB_BA_Blink [▶ 647]	Simple oscillator function block
FB_BA_FIFO04 [▶ 648]	Sequential control of up to four units
FB_BA_FIFO08 [▶ 650]	Sequential control of up to eight units
FB_BA_StepCtrl08 [▶ 652]	Step sequence function block, 8 steps
FB_BA_StepCtrl12 [▶ 655]	Step sequence function block, 12 steps
FB_BA_FIFO04_XX [▶ 658]	
FB_BA_FIFO08_XX [▶ 659]	

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 *lrIn* at the output (*lrQ01*..*lrQ16*) whose number is entered at input *udiSel*. All other outputs are set to 0 (for boolean demultiplexers to FALSE).

Example:

Inputs	Outputs
bEn = TRUE	lrQ01 = 0.0
udiSel = 5	lrQ02 = 0.0
lrIn = 32.5	lrQ03 = 0.0
	lrQ04 = 0.0
	lrQ05 = 32.5
	lrQ06 = 0.0
	lrQ07 = 0.0
	lrQ08 = 0.0
	lrQ09 = 0.0
	lrQ10 = 0.0
	lrQ11 = 0.0
	lrQ12 = 0.0
	lrQ13 = 0.0
	lrQ14 = 0.0
	lrQ15 = 0.0
	lrQ16 = 0.0

If the value entered at *udiSel* is greater than the number of outputs, the value of *lrIn* is output at the "highest" output:

Inputs	Outputs
bEn = TRUE	lrQ01 = 0.0
udiSel = 25	lrQ02 = 0.0
rIn = 32.5	lrQ03 = 0.0
	lrQ04 = 0.0
	lrQ05 = 0.0
	lrQ06 = 0.0
	lrQ07 = 0.0
	lrQ08 = 0.0
	lrQ09 = 0.0
	lrQ10 = 0.0
	lrQ11 = 0.0
	lrQ12 = 0.0
	lrQ13 = 0.0
	lrQ14 = 0.0
	lrQ15 = 0.0
	lrQ16 = 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, *rReplVal* 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			
rReplVal			

If no priority is active, the value of the global constant Const.udiNoActvPrio [▶ 692] 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 [▶ 692] 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 (*lrIn01..lrIn16*) at output *lrQ*, whose number is entered at input *udiSel*.
Example:

Inputs	Output
bEn = TRUE	lrQ = 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 *lrQ*:

Inputs	Output
bEn = TRUE	lrQ = 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 *lrQ*, 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

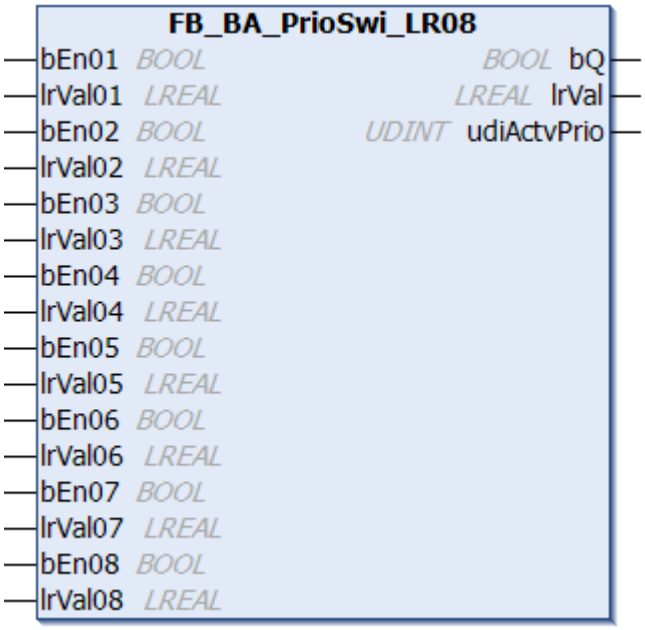
```
lrQ   : LREAL;
```

lrQ: 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 *lrVal* 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.

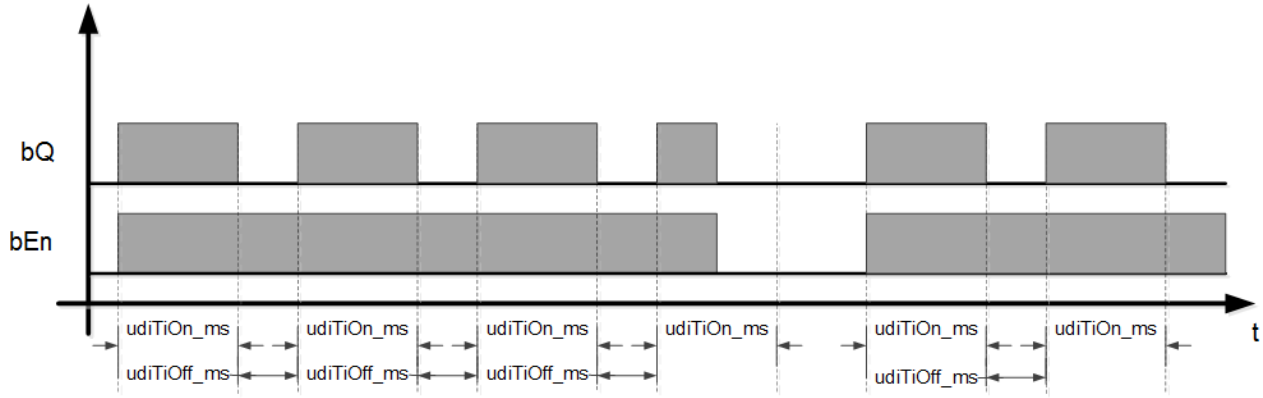
udiActvPrio: Current (highest) priority that is enabled.

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;
```

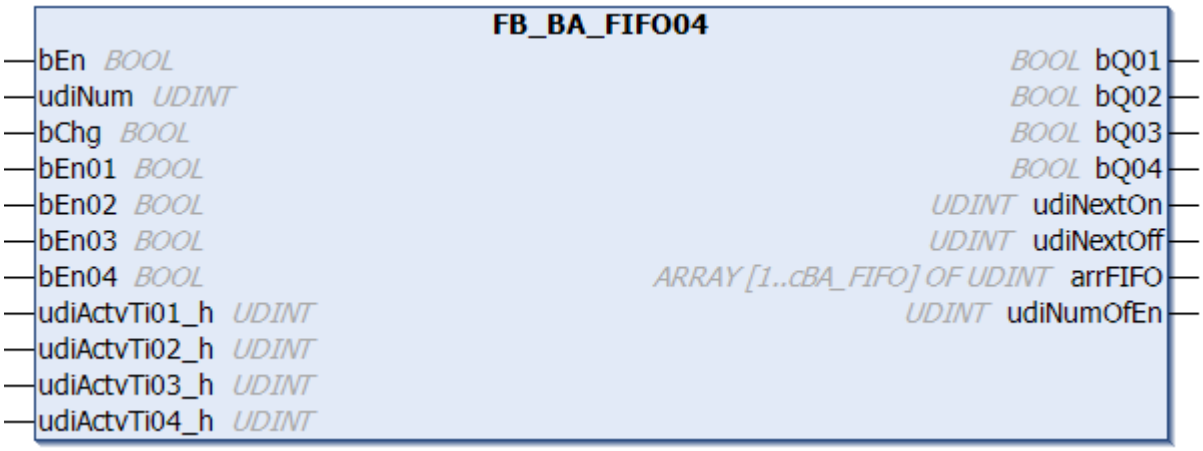
bQ: Oscillator output.

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 [▶ 650] 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.

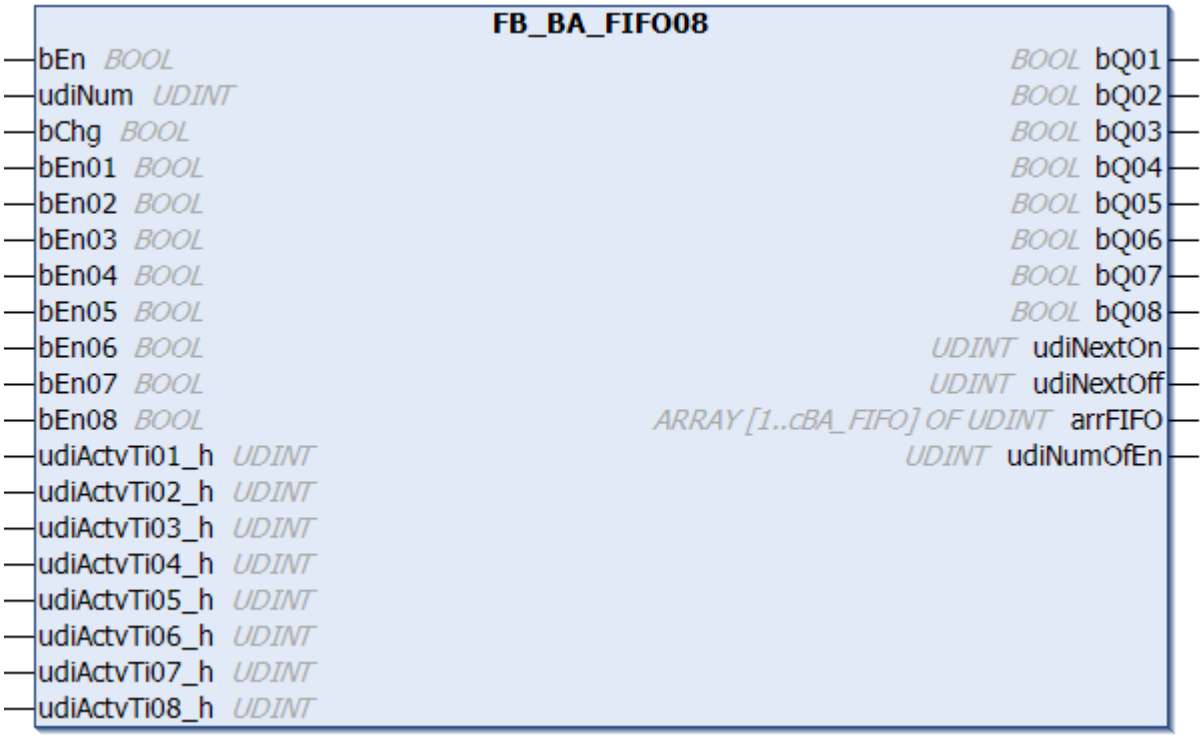
sErrDesc: Contains the error description.

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 [▶ 648] 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.

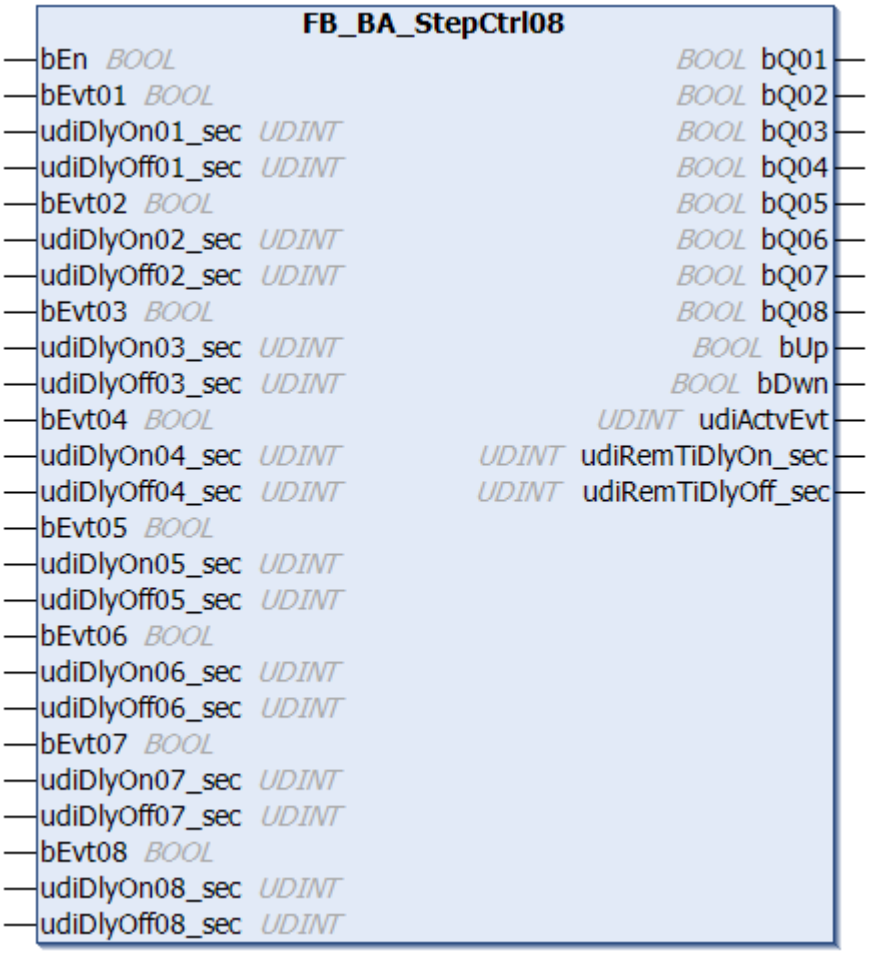
sErrDescr: Contains the error description.

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

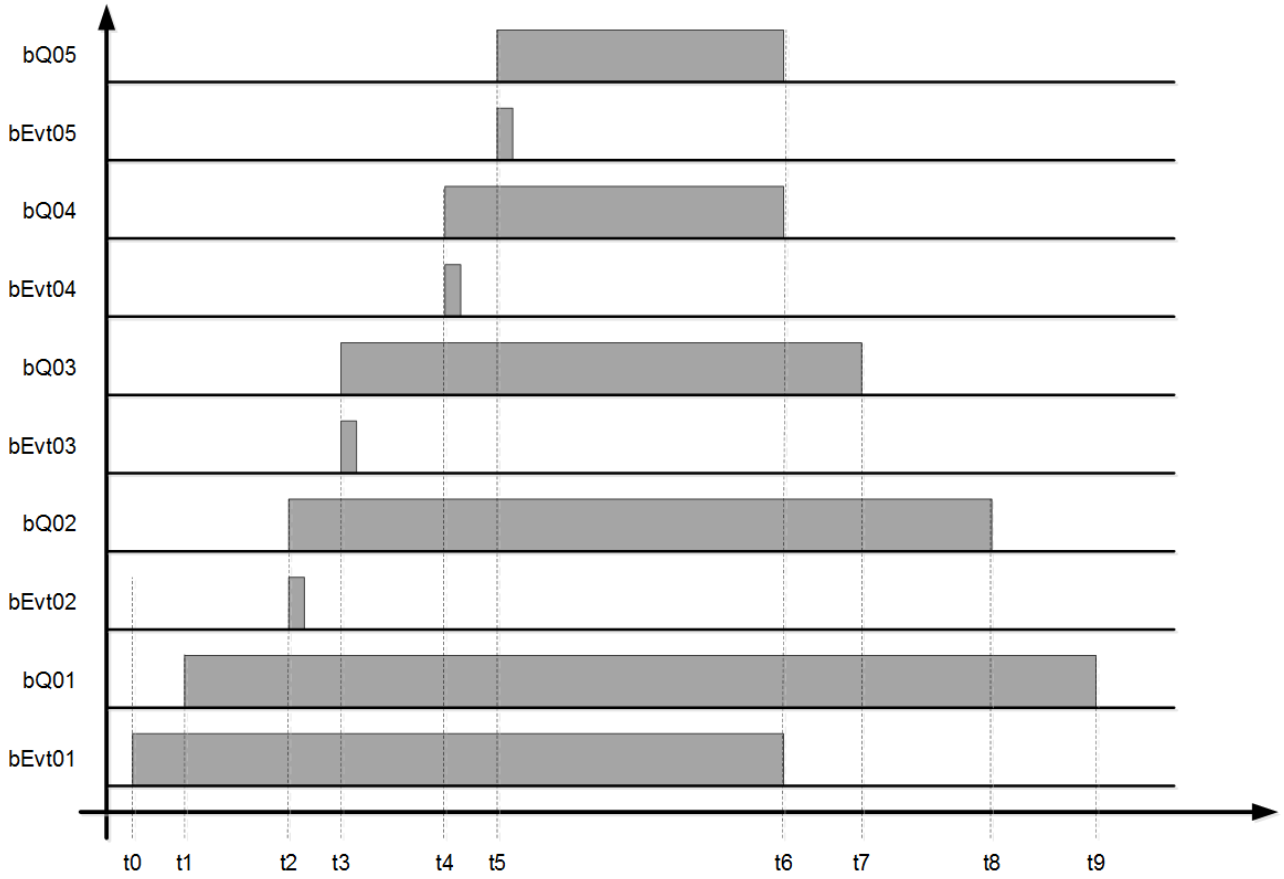


The function block is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs of *bQ01* to *bQ08* are set to FALSE. The control sequence 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.

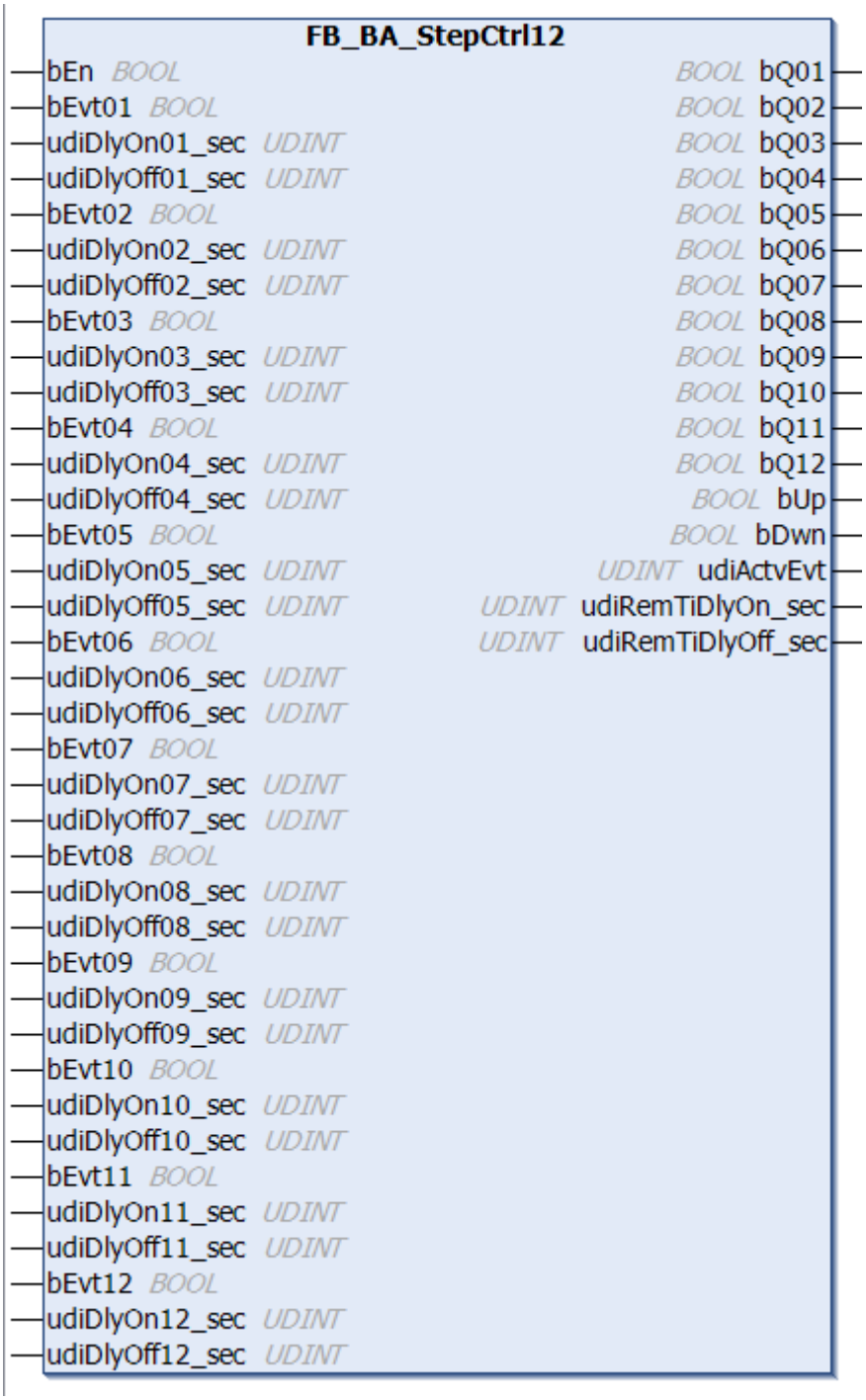
udiRTiDlyOn: Time remaining to up-switching to the next step [s].

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

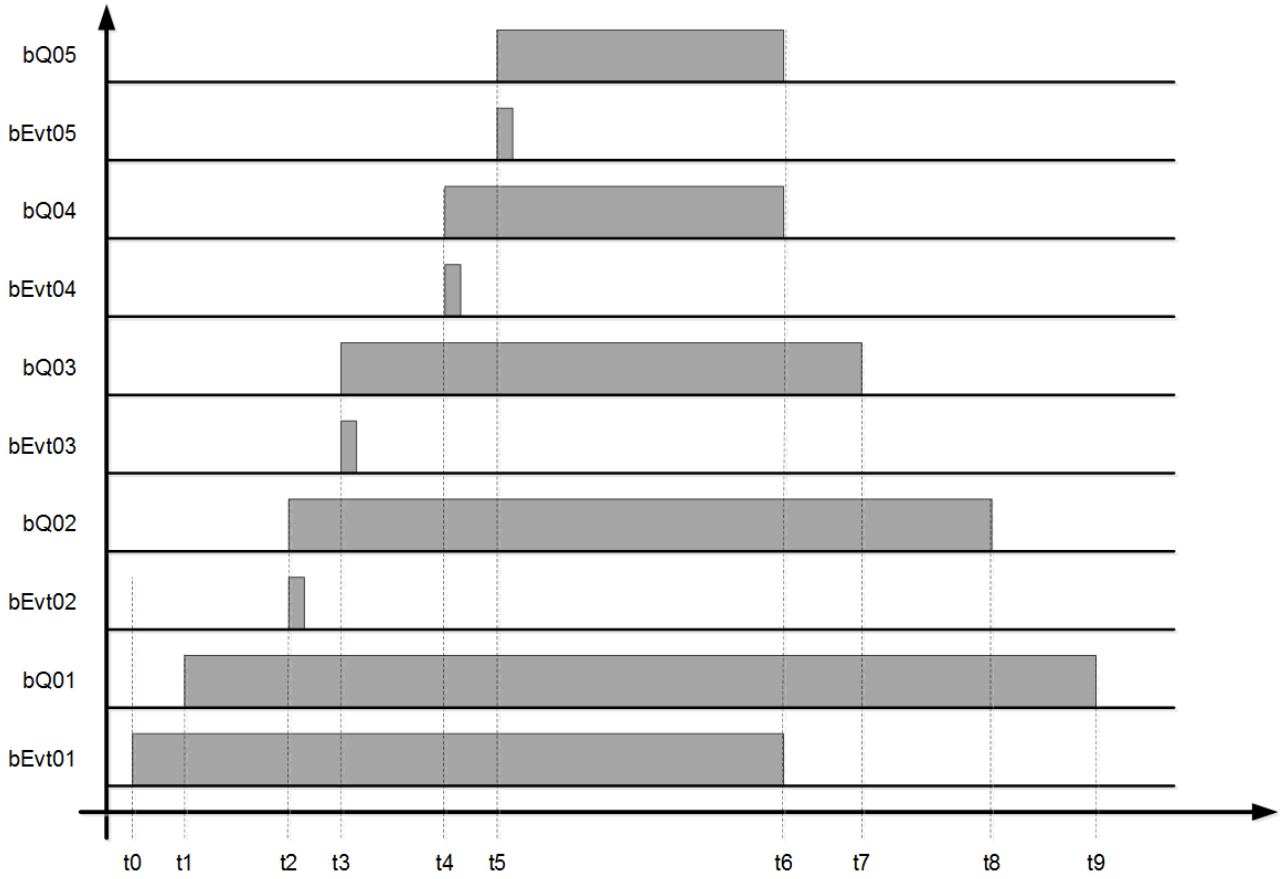


The function block is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs of *bQ01* to *bQ12* are set to FALSE. The control sequence 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.

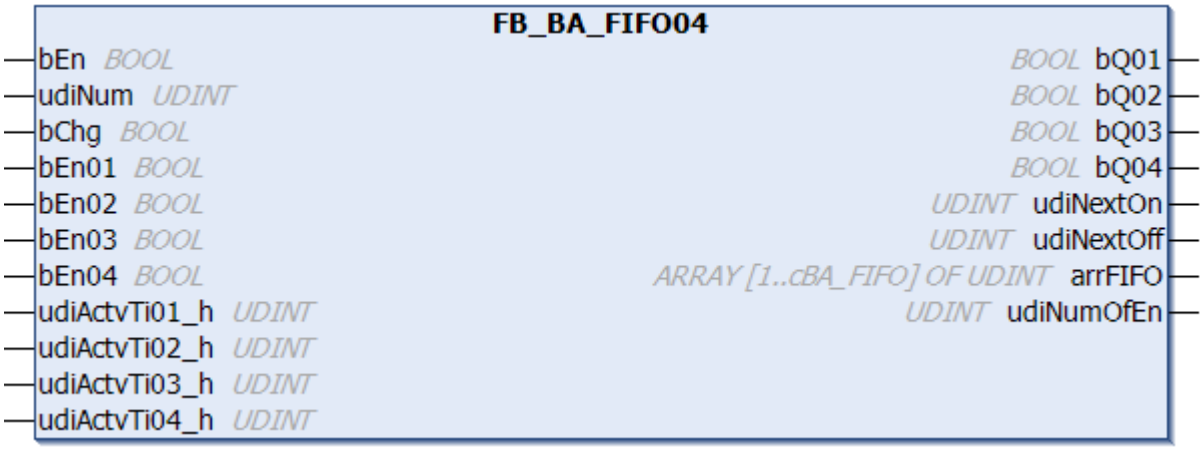
udiRTiDlyOn: Time remaining to up-switching to the next step [s].

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 [▶ 648]. 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

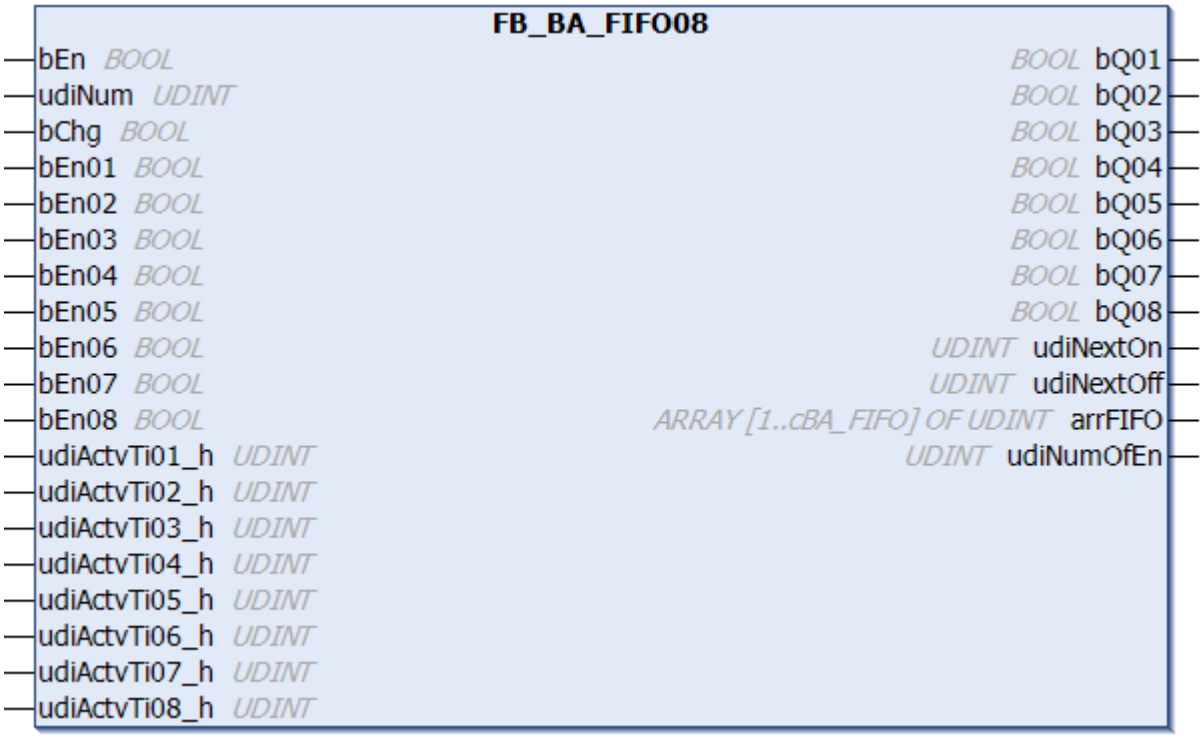
```
rVal01 - rVal04   : REAL;
```

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 [▶ 650]. 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

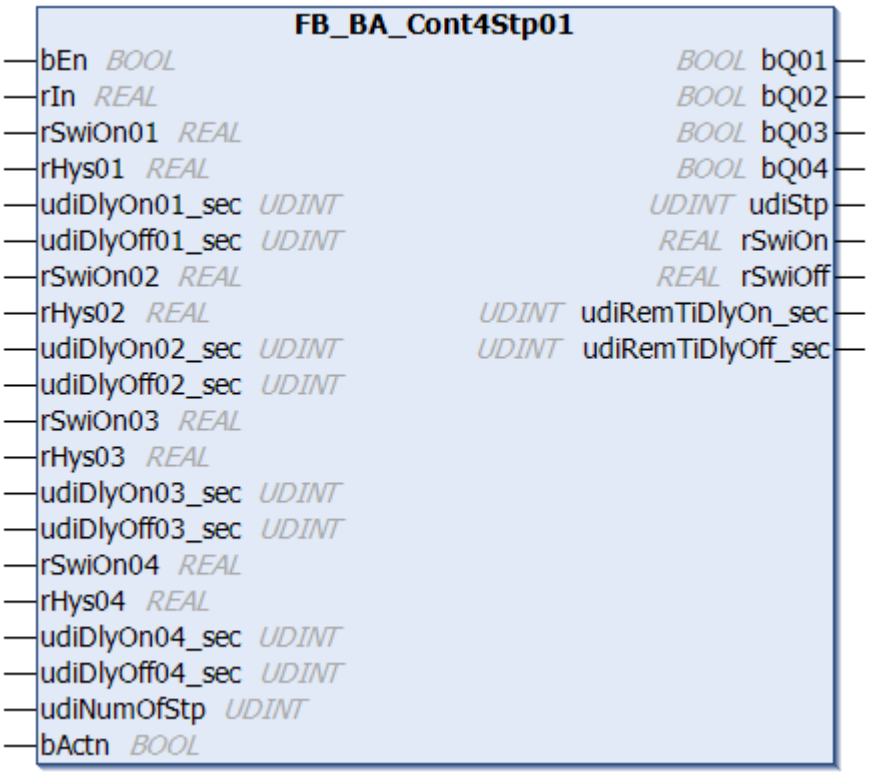
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.4 Hysteresis, 2-point control

Function blocks

Name	Description
FB_BA_Cont4Stp01 [▶ 660]	Step switch with four stages
FB_BA_Swi2P [▶ 665]	Two-point switch
FB_BA_SwiHys2P [▶ 667]	Two-point switch with one switching point

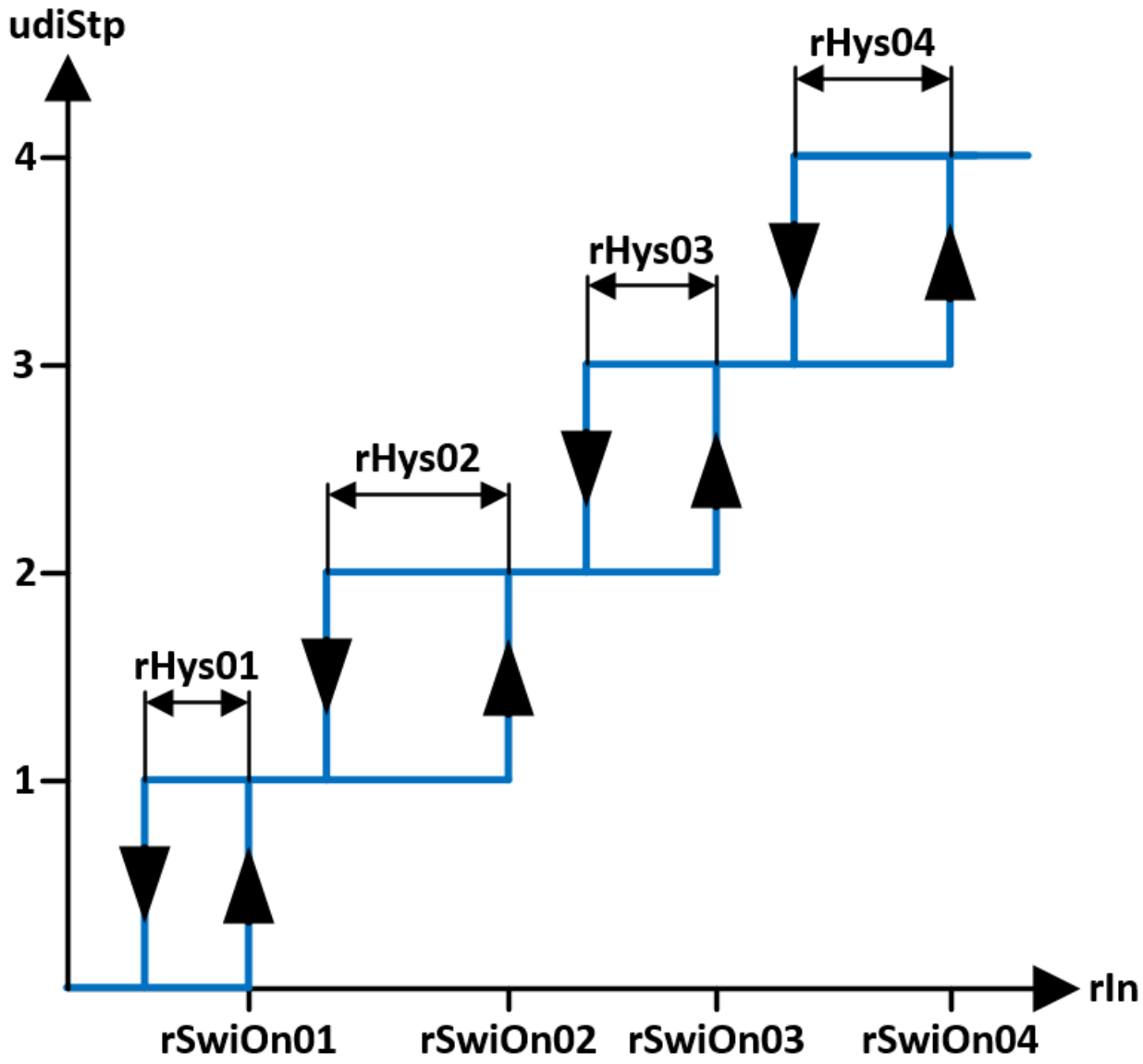
6.1.2.3.2.1.5.4.1 FB_BA_Cont4Stp01



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

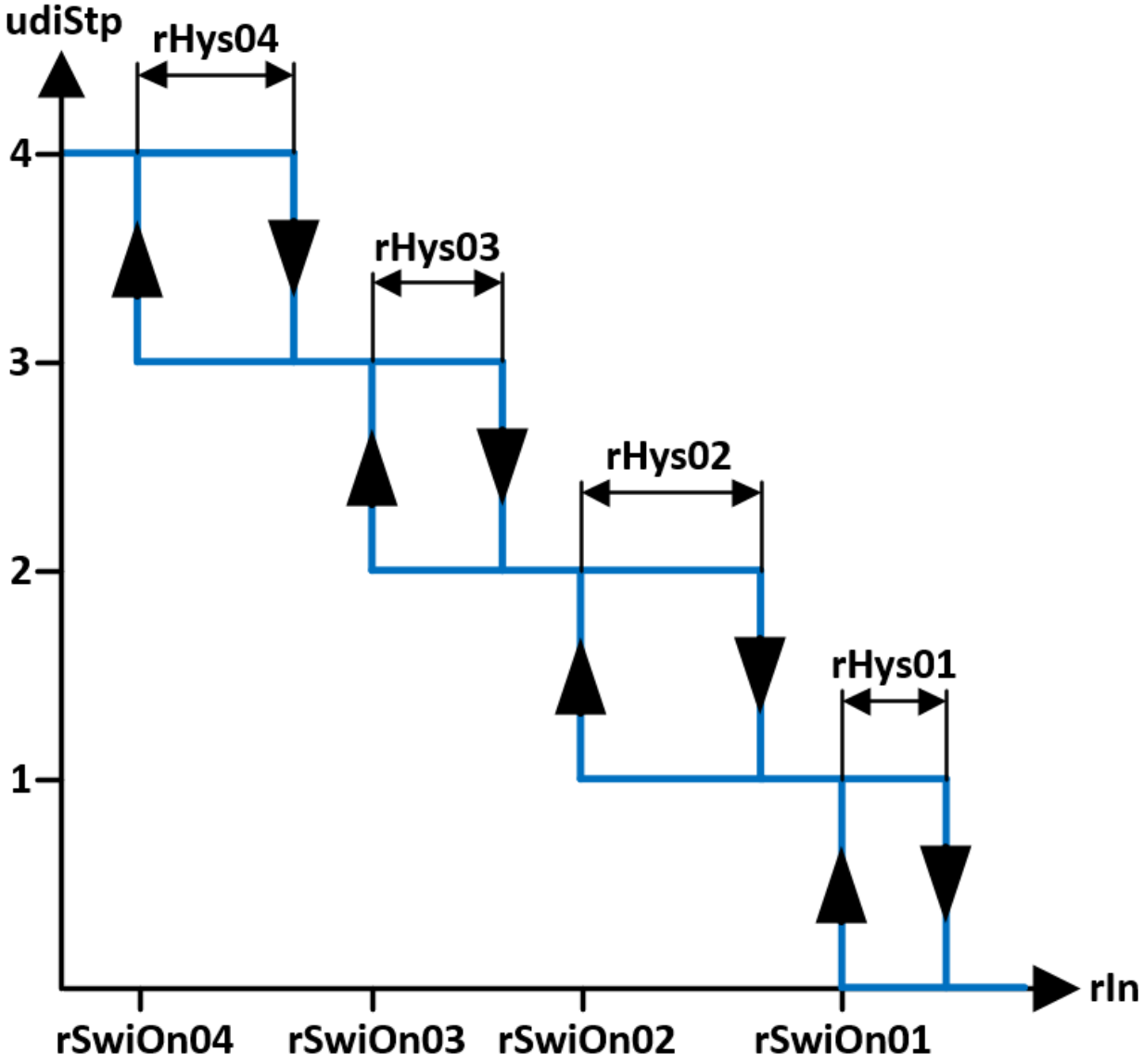
Control direction of parameter *bActn* = FALSE = Reverse = Heating



udiStp	udiNu-mOfStp	rSwiOn	rSwiOff	udi-RemTiD-lyOn_sec	udi-RemTiD-lyOff_sec	bQ01	bQ02	bQ03	bQ04
0	0	rSwiOn01	rSwiOn01 - rHys01	udiDlyOn 01_sec	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	rSwiOn02	rSwiOn01 - rHys01	udiDlyOn 02_sec	udiDlyOff 01_sec	TRUE	FALSE	FALSE	FALSE
2	>= 2	rSwiOn03	rSwiOn02 - rHys02	udiDlyOn 03_sec	udiDlyOff 02_sec	TRUE	TRUE	FALSE	FALSE
3	>= 3	rSwiOn04	rSwiOn03 - rHys03	udiDlyOn 04_sec	udiDlyOff 03_sec	TRUE	TRUE	TRUE	FALSE
4	>= 4	rSwiOn04	rSwiOn04 - rHys04	0	udiDlyOff 04_sec	TRUE	TRUE	TRUE	TRUE

Diagram 02

Control direction parameter *bActn* = TRUE = Direct = Cooling



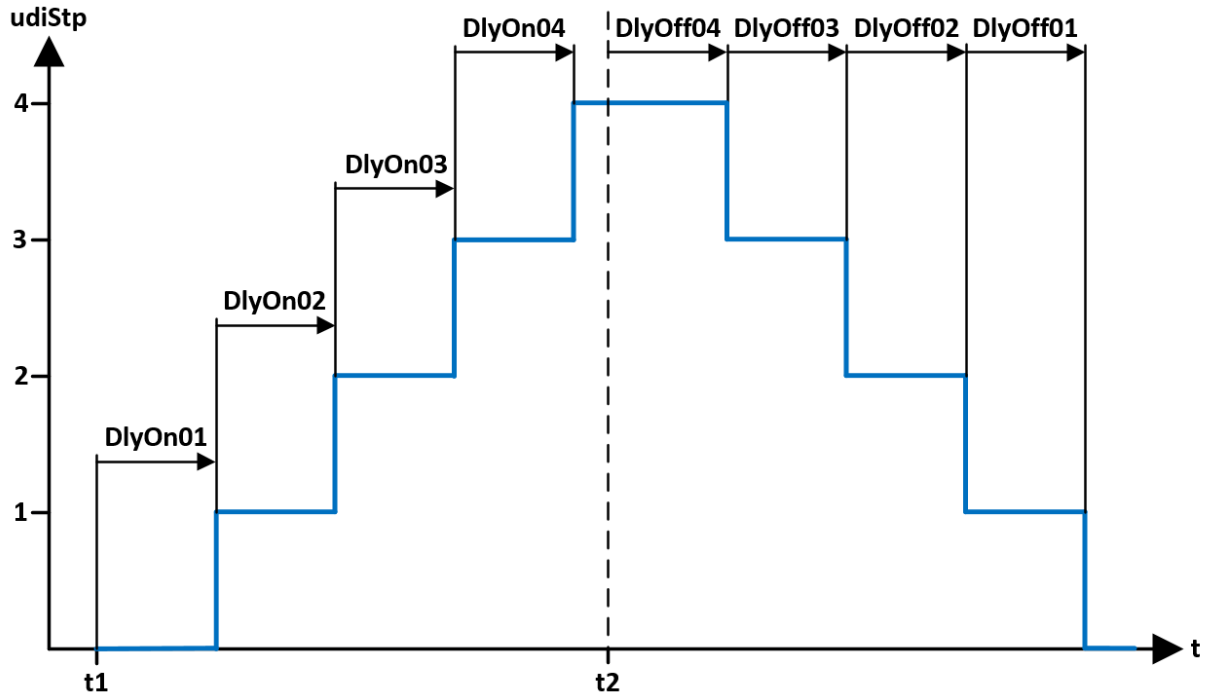
udiStp	udiNu-mOfStp	rSwiOn	rSwiOff	udi-RemTiD-lyOn_sec	udi-RemTiD-lyOff_sec	bQ01	bQ02	bQ03	bQ04
0	0	rSwiOn01	rSwiOn01 + rHys01	udiDlyOn 01_sec	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	rSwiOn02	rSwiOn01 + rHys01	udiDlyOn 02_sec	udiDlyOff 01_sec	TRUE	FALSE	FALSE	FALSE
2	>= 2	rSwiOn03	rSwiOn02 + rHys02	udiDlyOn 03_sec	udiDlyOff 02_sec	TRUE	TRUE	FALSE	FALSE
3	>= 3	rSwiOn04	rSwiOn03 + rHys03	udiDlyOn 04_sec	udiDlyOff 03_sec	TRUE	TRUE	TRUE	FALSE
4	4	rSwiOn04	rSwiOn04 + rHys04	0	udiDlyOff 04_sec	TRUE	TRUE	TRUE	TRUE

Diagram 03

Timing of the switch-on and switch-off delays

At time t1, *rIn* jumps from *rSwiOn01* to *rSwiOn04*

At time t2, *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

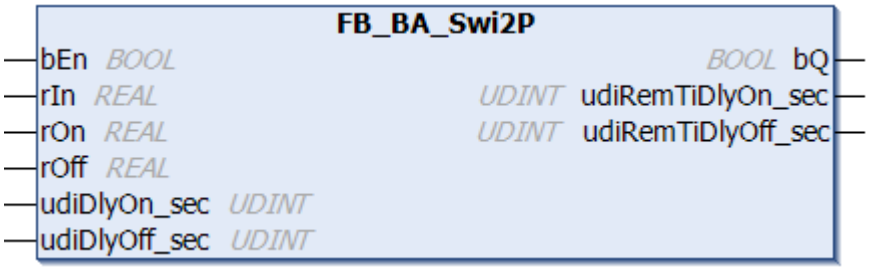
udiRemTiDlyOn_sec: If the switch-on point for switching to the next level is met, the progress of the switch-on delay time is displayed here.

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

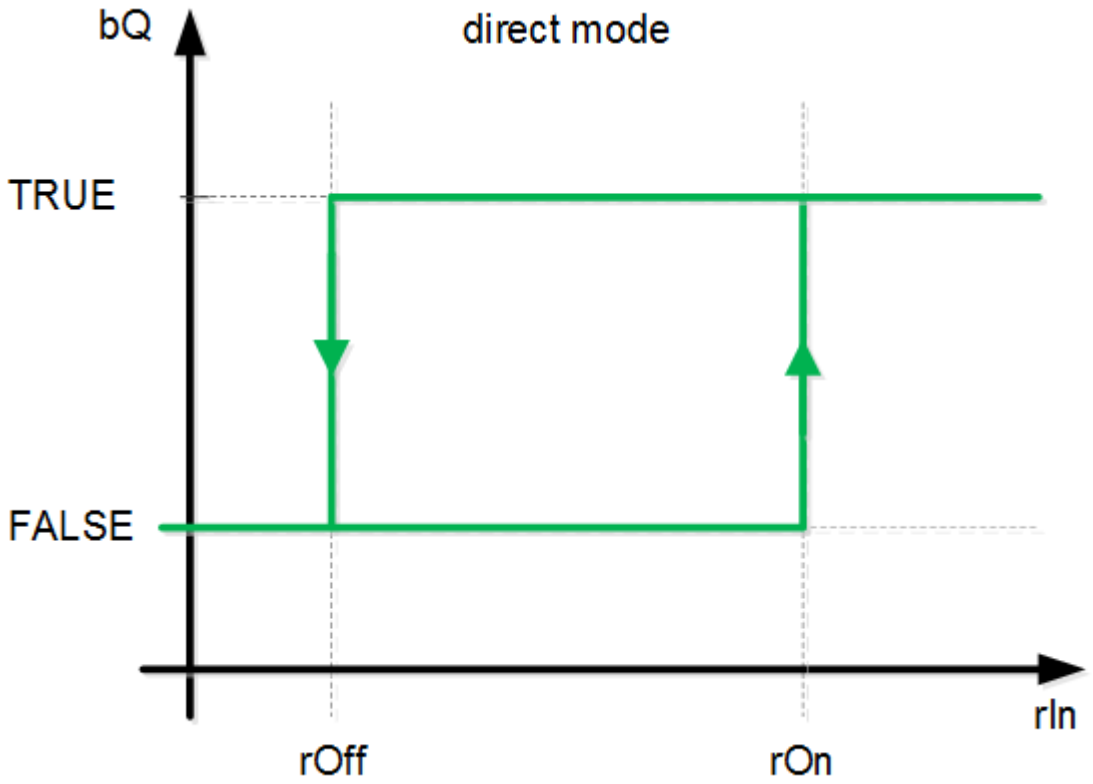
6.1.2.3.2.1.5.4.2 FB_BA_Swi2P



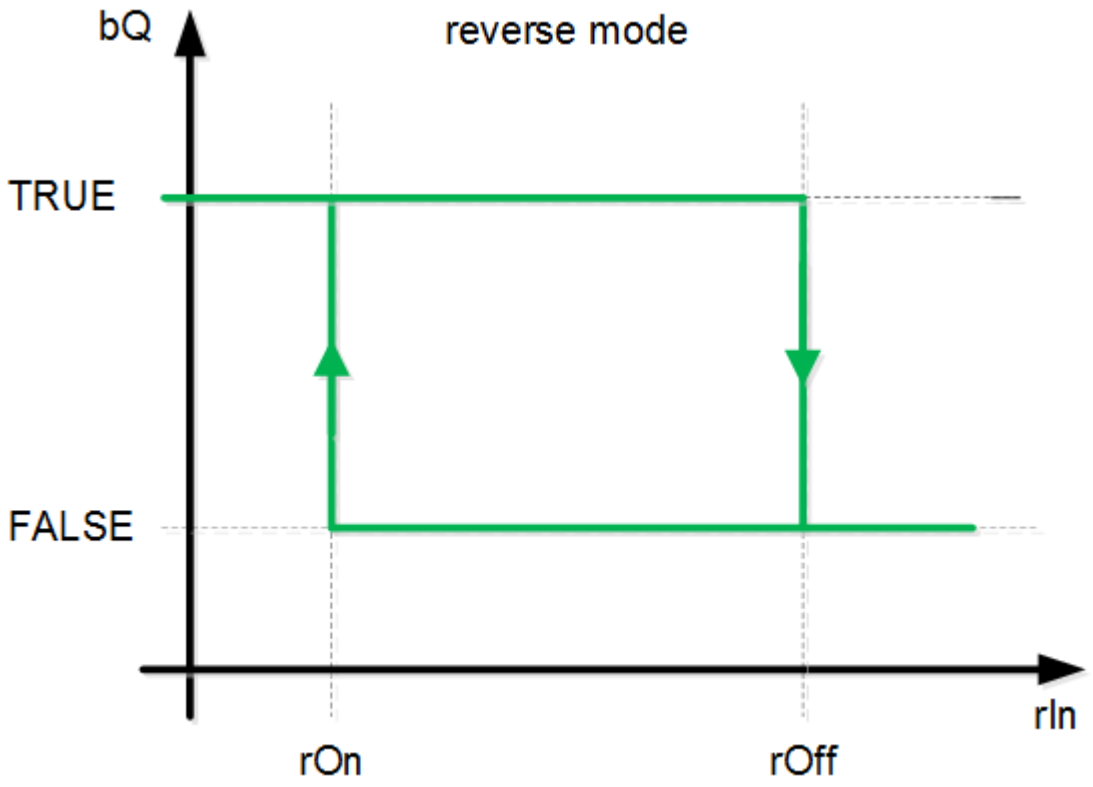
The function block *FB_BA_Swi2P* is a two-point switch with one switch-on point and one switch-off point.

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).



Is 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.

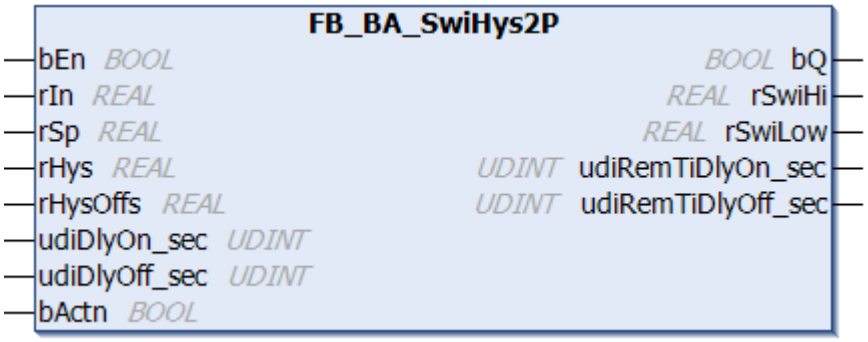
udiRemTiDlyOn_sec: Remaining time of the switch-on delay.

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



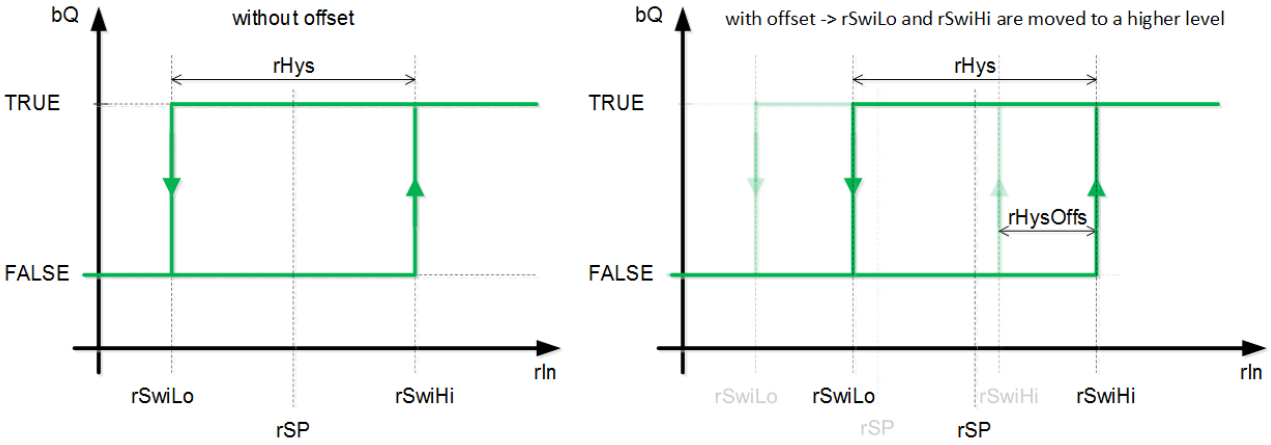
The function block *FB_BA_SwiHys2P* is a two-point switch with adjustable hysteresis and hysteresis offset.

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*.

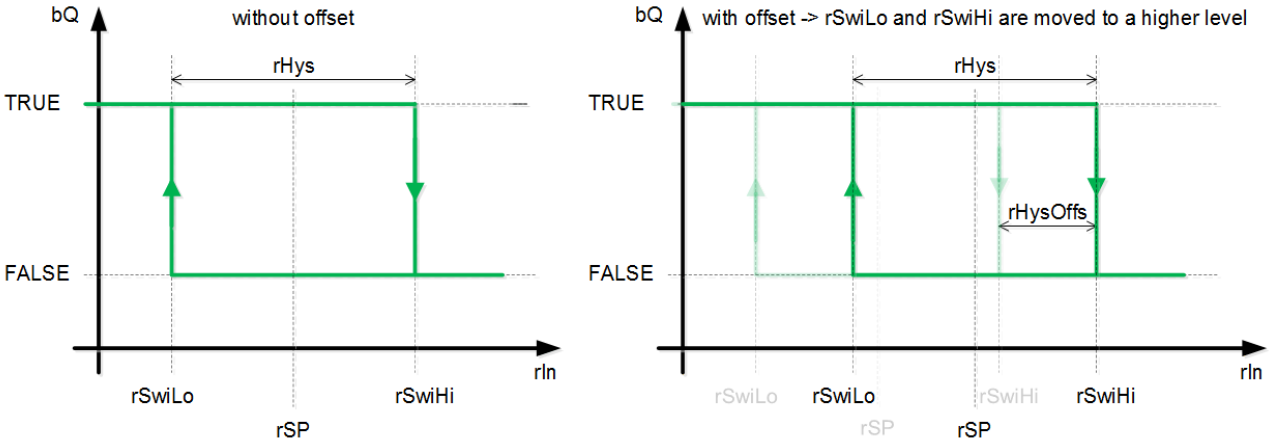
The active switching points result from the setpoint, the hysteresis and the hysteresis offset. They are output at *rSwiHi* and *rSwiLo*.

- 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

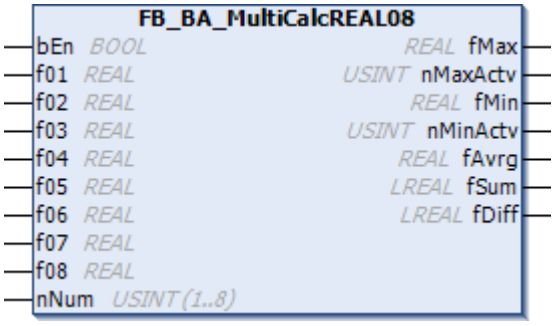
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5 Mathematical functions

Function blocks

Name	Description
FB_BA_MultiCalc_XX [▶ 669]	Multi-calculation function blocks
FB_BA_Chrct02 [▶ 670]	Linear interpolation for 2 interpolation points
FB_BA_Chrct04 [▶ 671]	Linear interpolation for 4 interpolation points
FB_BA_Chrct07 [▶ 673]	Linear interpolation for 7 interpolation points
FB_BA_Chrct32 [▶ 674]	Linear interpolation for 32 interpolation points
FB_BA_TiAvrg [▶ 676]	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 *udiMinActv*
- the minimum value of all inputs *rMin*
- the input at which this minimum value occurs *udiMinActv*
- the mean value of all inputs *rAvrg*
- 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
bEn = TRUE	rMax = 32
r01 = 32	udiMaxActv = 1
r02 = 17	rMin = 5
r03 = 5	udiMinActv = 3
r04 = 9	rAvrg = 18.5
r05 = 16	rSum = 111
r06 = 32	rDiff = 27
r07 = 25	
r08 = 44	
udiNum = 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.

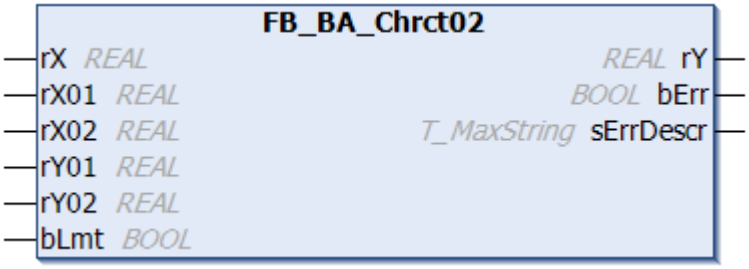
rSum: Sum of all inputs.

rDiff: Difference between maximum and minimum value.

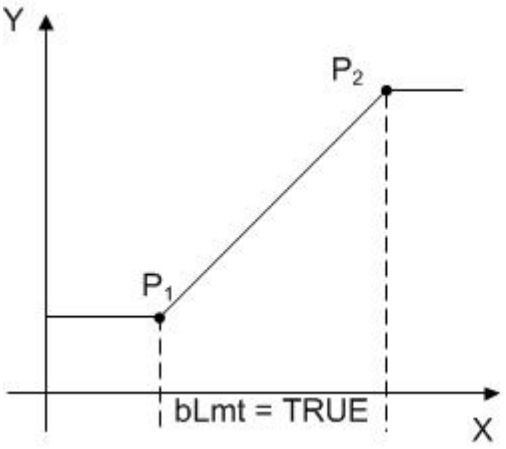
Requirements

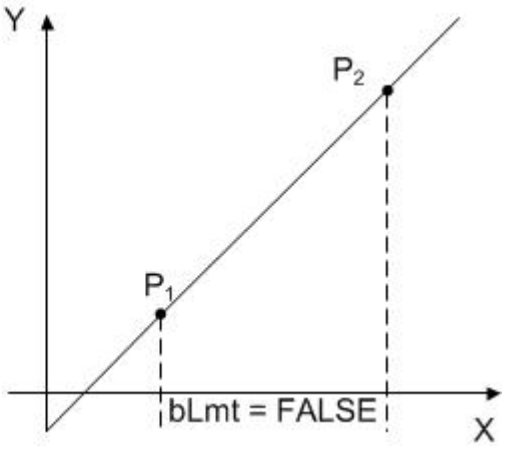
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.5.2 FB_BA_Chrct02



The function block FB_BA_Chrct02 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.

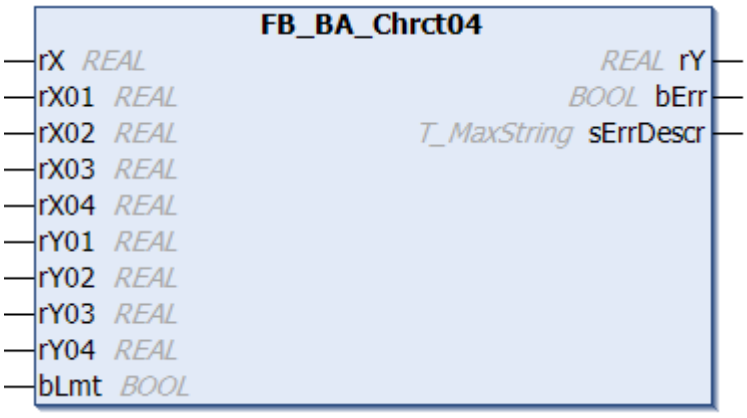
sErrDescr: Contains the error description.

Error description
01: Error: rX01 must not be equal to rX02.

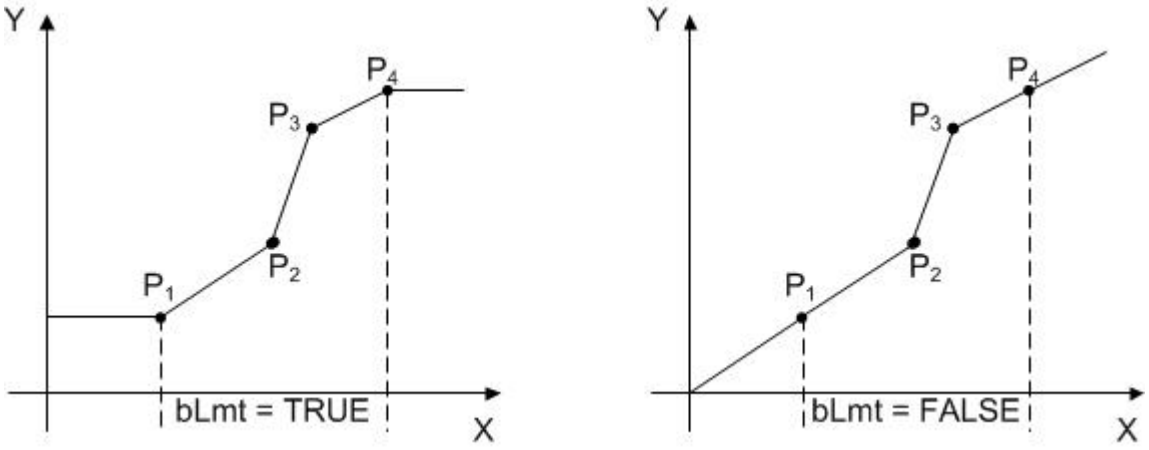
Requirements

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_Chrct04



The function block FB_BA_Chrct04 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.

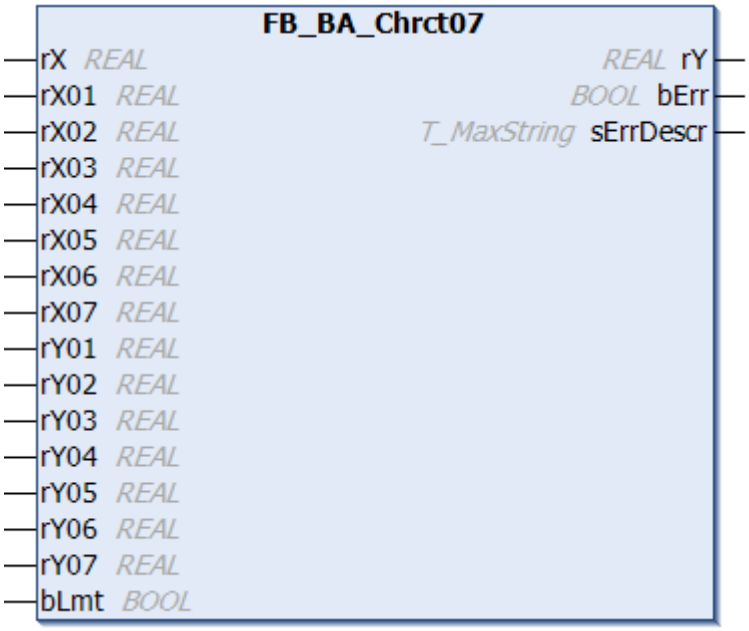
sErrDescr: Contains the error description.

Error description
01: Error: at the specified element. The sequence must always be rX01 > rX02 > rXn or rX01 < rX02 < rXn.

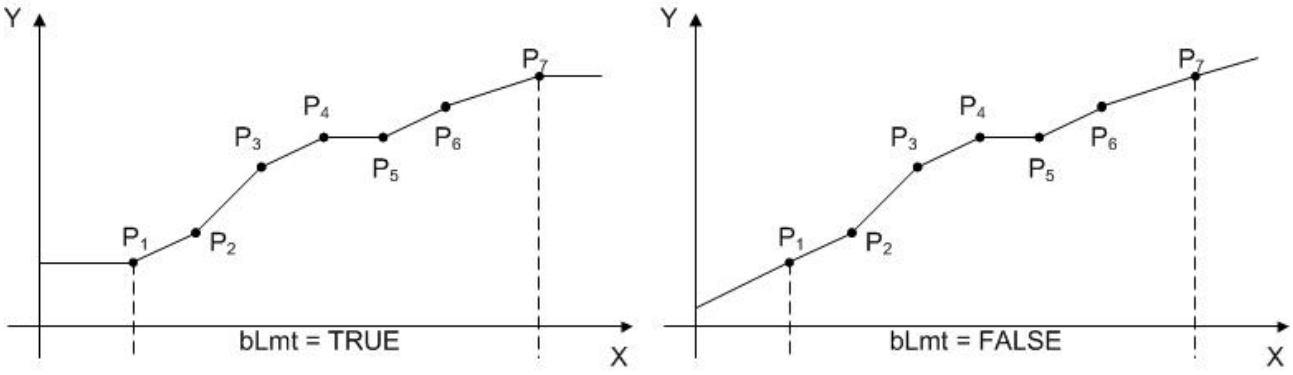
Requirements

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.

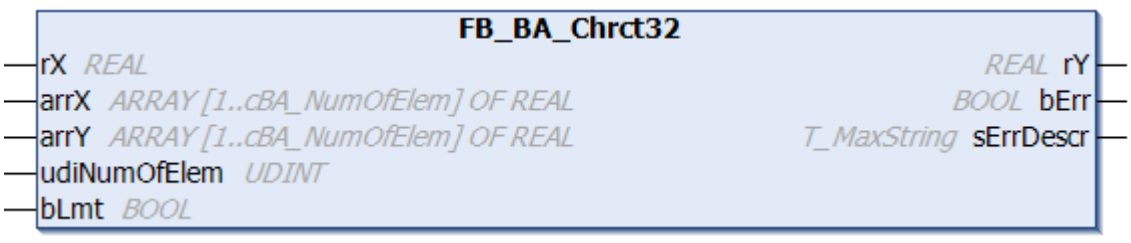
sErrDescr: Contains the error description.

Error description
01: Error: at the specified element. The sequence must always be rX01 > rX02 > rXn or rX01 < rX02 < rXn.

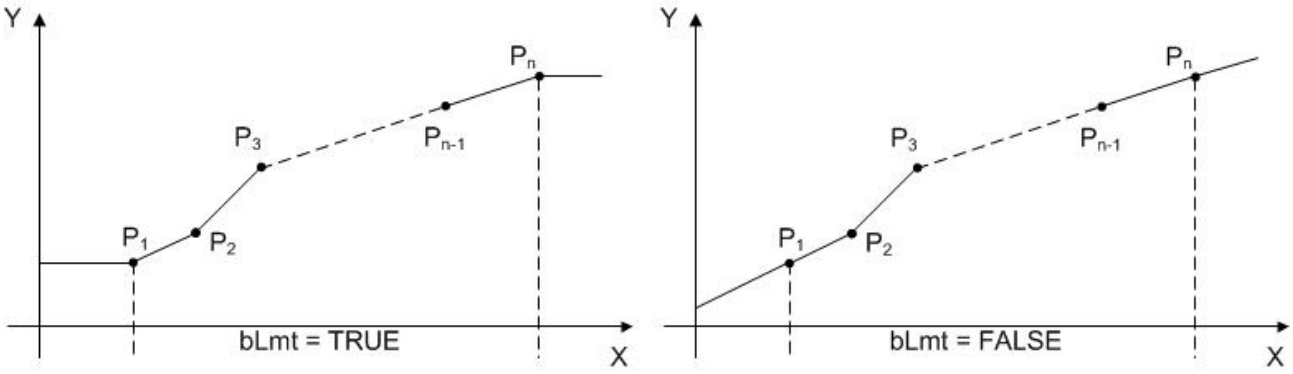
Requirements

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_Chrct32



The function block FB_BA_Chrct32 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_Chrct02 [▶ 670], FB_BA_Chrct04 [▶ 671] and FB_BA_Chrct07 [▶ 673], 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..FB_BA_Chrct32.cBA_NumOfElem] OF REAL;
arrY        : ARRAY [1..FB_BA_Chrct32.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.

nNumOfElem: Number of interpolation points. Internally limited to values between 2 and 32.

bLmt: Limitation of 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.

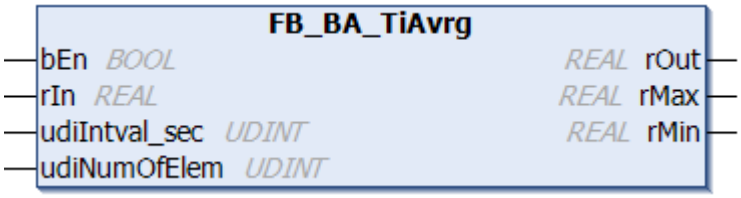
sErrDescr: Contains the error description.

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_TiAvrg



The function block *FB_BA_TiAvrg* 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 [▶ 678]	Feedback monitoring of an actuator by means of digital feedback.
FB_BA_FixedLimitCtrl [▶ 677]	Limit value monitoring of a fixed value.
FB_BA_SlidingLimitCtrl [▶ 679]	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 [▶ 692].

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 [▶ 692].

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 [▶ 692].

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*.

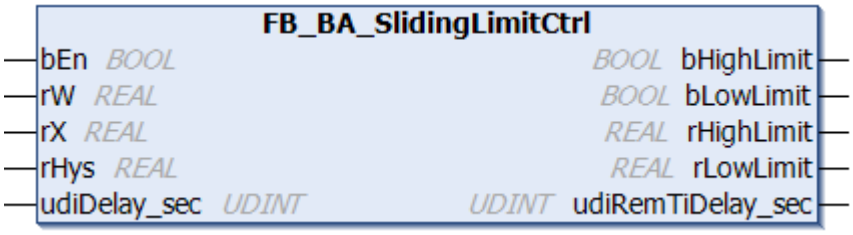
udiRemTiFdbDelay: Remaining time [s] until output *bErrOpn* is set.

udiRemTiInterruptionDelay: 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

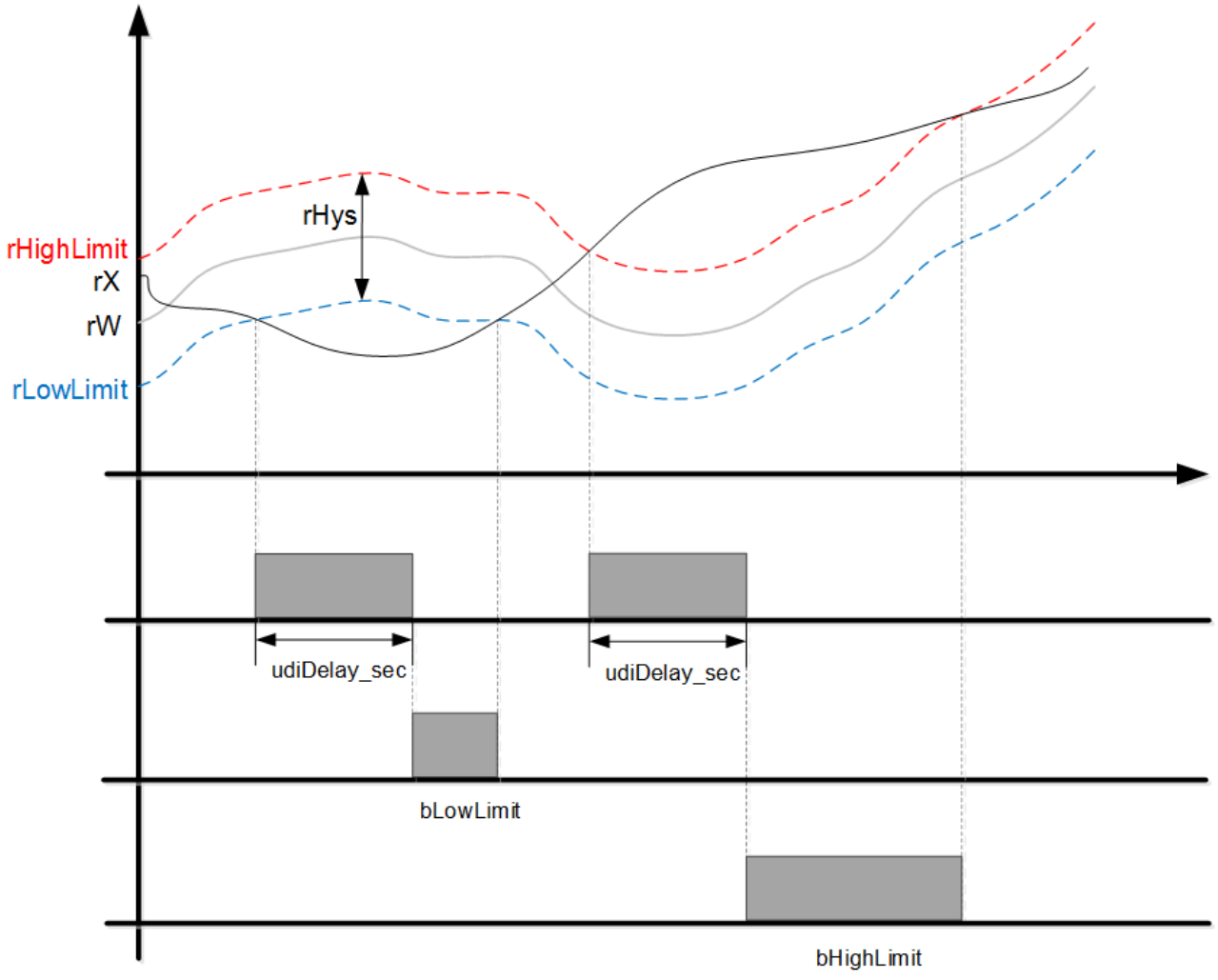


Function block for monitoring a floating setpoint.

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 [▶ 692].

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
FB_BA_FltrPT1 [▶ 680]	First order filter
FB_BA_RampLmt [▶ 681]	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;
bSetActl         : BOOL;
```

rIn: Input signal

udiDampConst_sec: Filter time constant [s]. Internally limited to values between 0 and 86400.

bSetActl: A rising edge at this input sets the output value *rOut* to the input value *rIn*.

VAR_OUTPUT

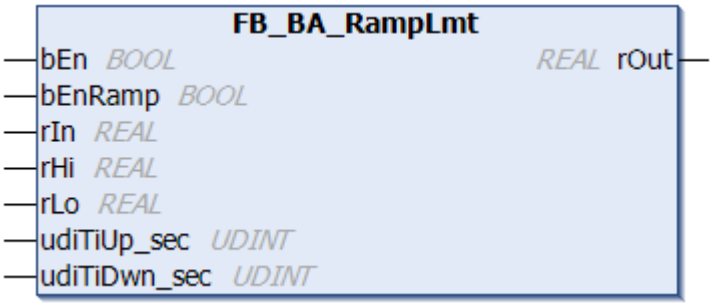
```
rOut    : REAL;
```

rOut: Filtered output signal.

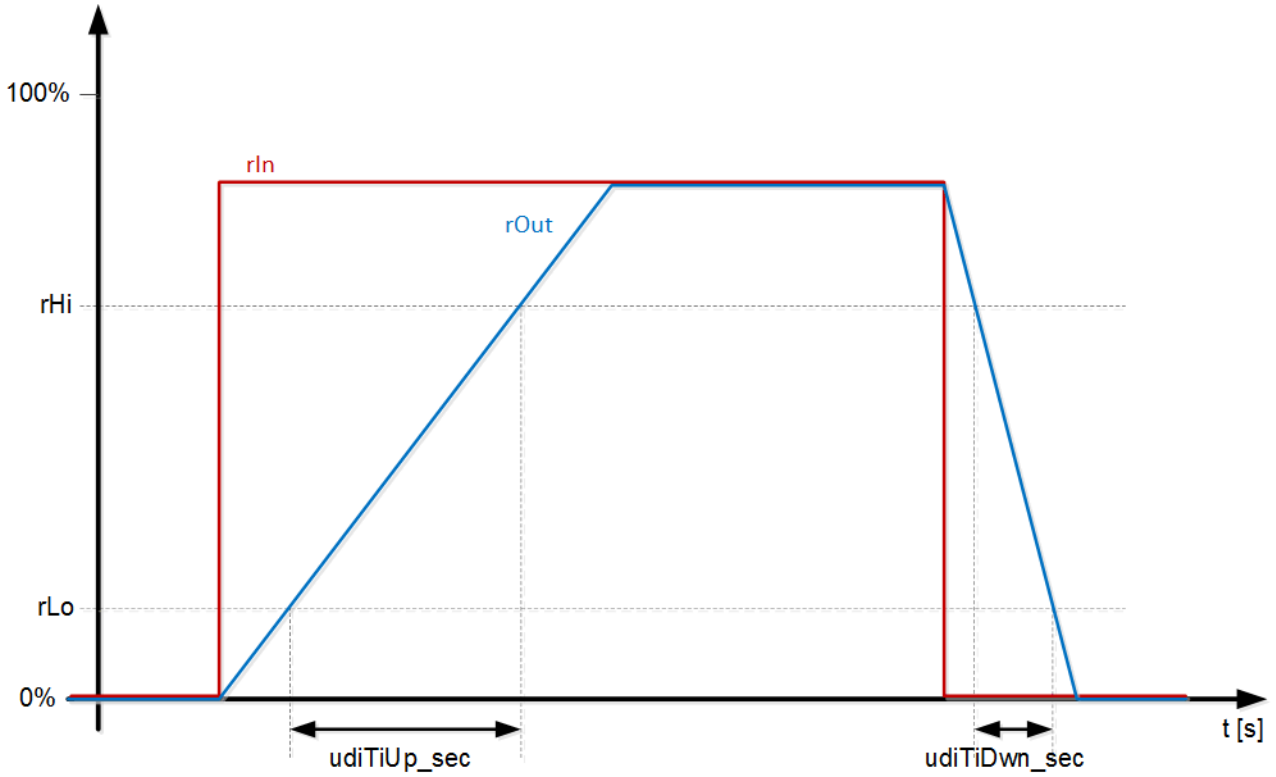
Requirements

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

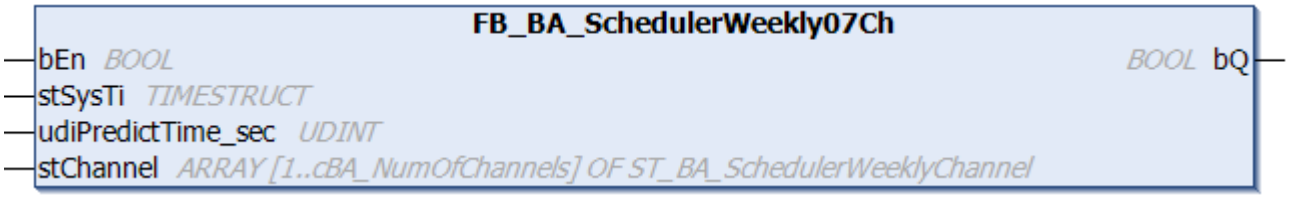
Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.1.5.8 Calendar

Function blocks

Name	Description
FB_BA_SchedulerWeeklyXXCh [▶ 683]	Weekly scheduler
FB_BA_CalenderXXCh [▶ 684]	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

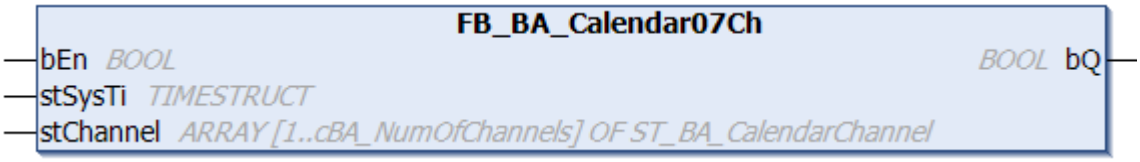
```
arrChannel: ARRAY [1..cBA_NumOfChannels] OF ST_BA_SchedulerWeeklyChannel;
```

arrChannel: Weekly scheduler; with the single-channel function block, the name of the variable is stChannel (see ST_BA_SchedulerWeeklyChannel [▶ 691]). 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_CalendarChannel;
```

arrChannel: Yearly scheduler; with the single-channel function block, the name of the variable is stChannel (see ST_BA_CalenderChannel [▶ 690]). 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 [▶ 685]	Enumerator for the definition of the positioning mode.
E_BA_ShdObjType [▶ 685]	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 [▶ 609] via the value *lrFixPos* [%].

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 [▶ 561].

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_ShdObjType

Enumerator for selecting the shading object type.

```
TYPE E_BA_ShdObjType :
(
  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 [▶ 686]	Structure of the interpolation point entries for the height adjustment of the blind.
ST_BA_Cnr [▶ 686]	Information about window corners.
ST_BA_Fcd [▶ 687]	Facade-specific data for activating the automatic functions.
ST_BA_FcdElem [▶ 687]	List entry for a facade element (window).
ST_BA_ShdObj [▶ 688]	List entry for a shading object
ST_BA_SpRmT [▶ 688]	Room temperature setpoints.
ST_BA_Sunbld [▶ 689]	Structure of the blind positioning telegram.
ST_BA_SunBldScn [▶ 690]	Table entry for a blind scene.
ST_BA_CalenderChannel [▶ 690]	Input of calendar entries.
ST_BA_SchedulerWeeklyChannel [▶ 691]	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 [▶ 609]. It is set to TRUE by the function block FB_BA_BldPosEntry [▶ 561] 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

6.1.2.3.2.2.2.3 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

6.1.2.3.2.2.2.4 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 [▶ 686].

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

6.1.2.3.2.2.2.5 ST_BA_ShdObj

List entry for a shading object.

```
TYPE ST_BA_ShdObj :
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_ShdObjType;
  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_ShdObjType [▶ 685].

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

6.1.2.3.2.2.2.6 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

6.1.2.3.2.2.7 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 [▶ 599] or FB_BA_SunBldPrioSwi8 [▶ 600]. The sun protection actuators FB_BA_SunBldActr [▶ 591] and FB_BA_RolBldActr [▶ 583] ignore it.

Requirements

Development environment	Required PLC library
TwinCAT from v3.1.4024.7	Tc3_BA from v1.1.6.0

6.1.2.3.2.2.2.8 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

6.1.2.3.2.2.2.9 ST_BA_CalenderChannel

Structure for entering calendar entries.

```
TYPE ST_BA_CalendarChannel:
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

6.1.2.3.2.2.2.10 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 [▶ 601].

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 [▶ 575] and FB_BA_RdShdObjLst [▶ 579].

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**

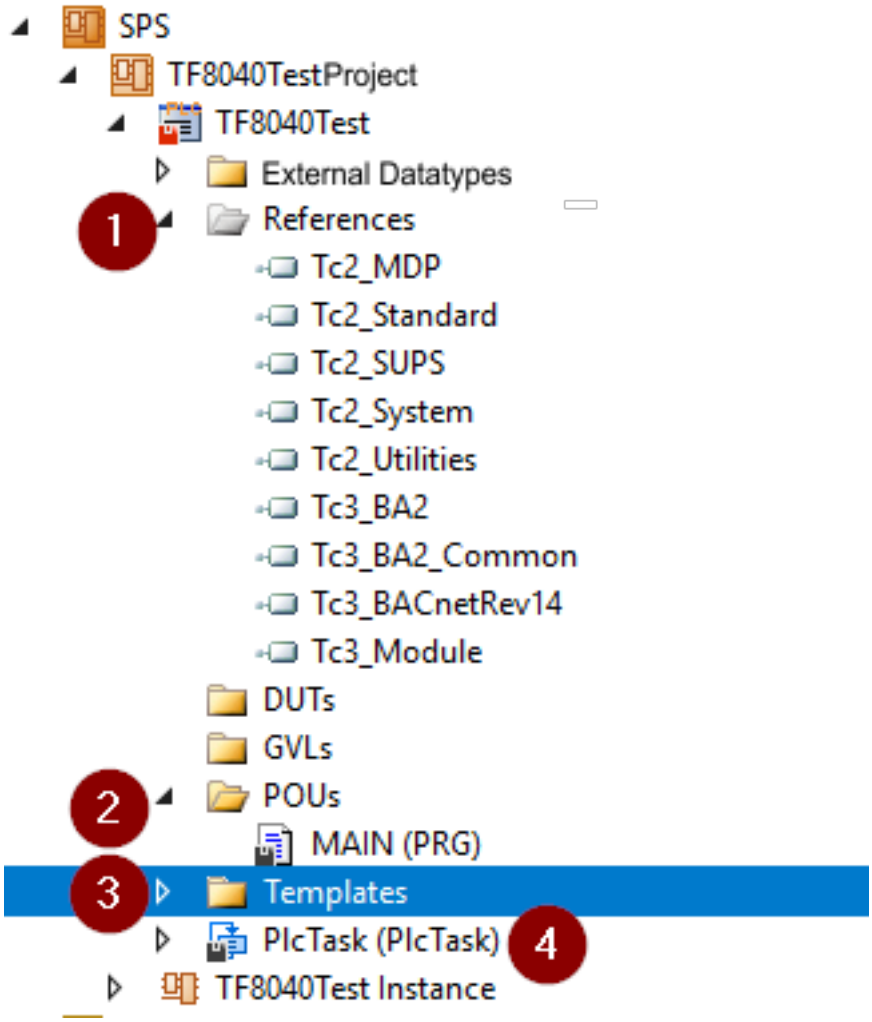
Programs

- **MAIN**
  - FB_BA_DPAD
    Contains preconfigured levels for small projects.
  - Project structure [▶ 38]
    Contains declarations and FB calls for:
    - Control cabinet
    - Device

Templates

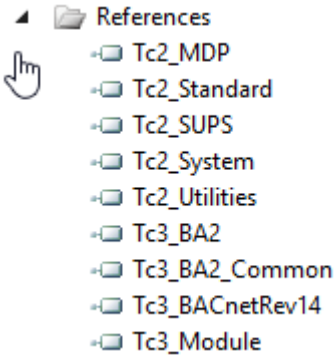
Contains all the templates required for compiling the project template without errors.

Structure in the solution tree



- **References**

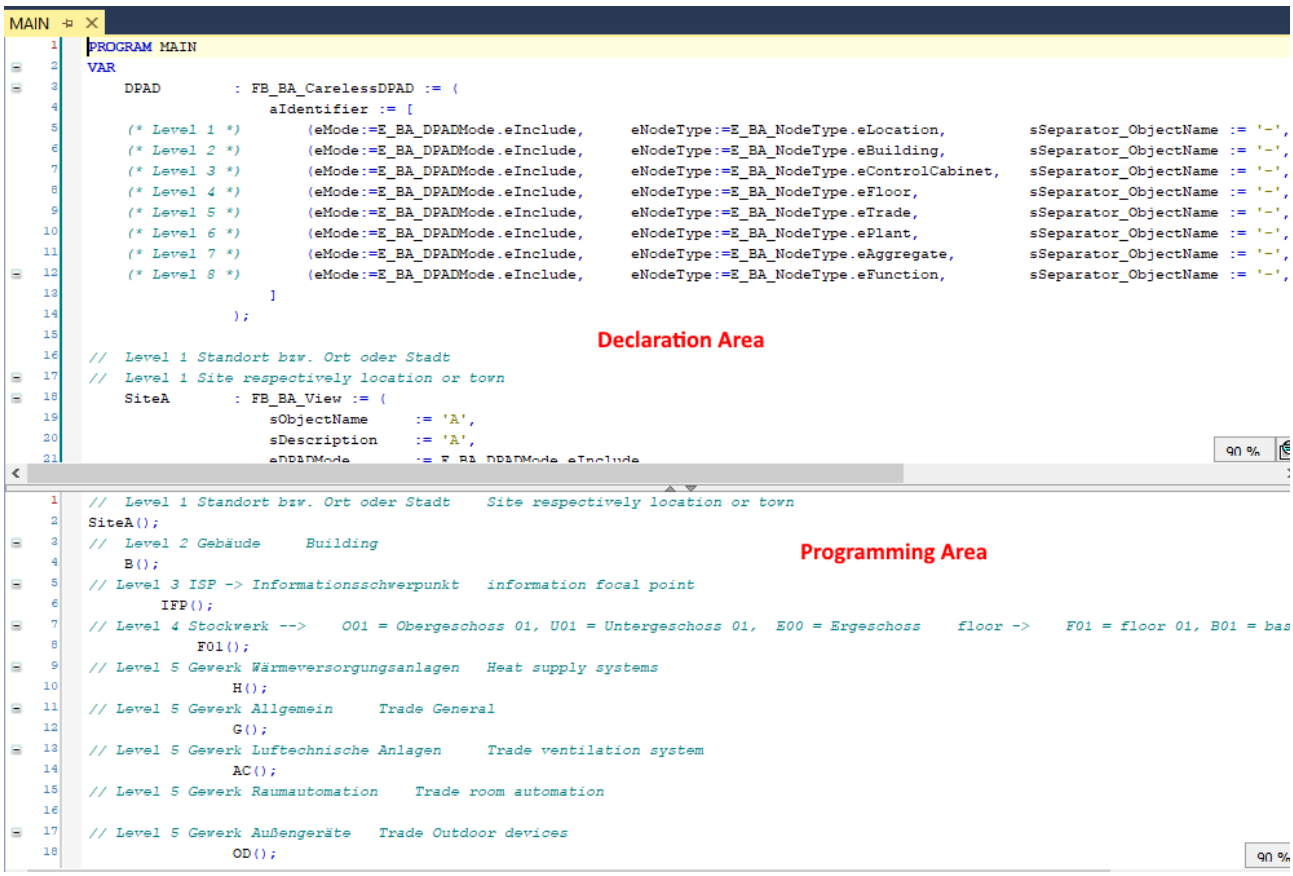
All the libraries required for a TF8040 project are loaded here.



- **MAIN POU**

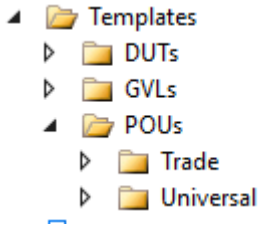
The standard project structure is called in the MAIN POU.

This template provides a project structure as an application example with eight levels based on an example from VDI 3814.



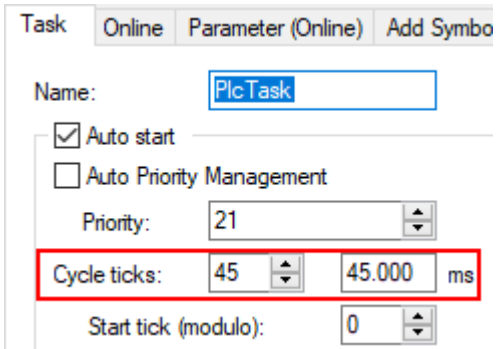
- **Templates**

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 [▶ 1344].



- **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 preparations are complete.

- **Import templates**

Project-specific programming begins with the integration of templates from the Template Repository [▶ 1344].

Now start with the project-specific programming.

6.1.4 Templates

With the templates, Beckhoff provides users of TwinCAT 3 Building Automation with a large systems toolkit. The use of templates facilitates project planning and increases the quality of project execution.

Overview of technical systems

Category	Name	Description
Templates for technical systems	Technical system [▶ 708]	Air conditioning, automation control, building automation, room automation
Universal templates	Universal [▶ 981]	Aggregate, dampers, motor, pump, sensor, controller, smoke detector, valve, weather station

Each template is available in 2 different configurations regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

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_BA_AC_SeqNumber_H

Sequence number controller where the order of the sequence numbers are taken into account.

```
TYPE E_BA_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.

6.1.4.1.1.2.1.2 E_BA_AC_SeqNumber_T

Sequence number controller where the order of the sequence numbers are taken into account.

```
TYPE E_BA_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.

6.1.4.1.1.2.1.3 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	Plant step 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

6.1.4.1.1.2.1.4 E_BA_AC_PlantStep01

Plant steps in the start program of an air conditioning system

```
TYPE E_BA_AC_PlantStep01 :
(
  eOff                                := 1,
  eErc                                := 2,
  ePreRinse                           := 3,
  eDamperOuA                          := 4,
  eFanSupplyAir                       := 5,
  eDamperExhA                         := 6,
  eFanExtractAir                      := 7,
  eCooler                             := 8,
  eReheater                           := 9,
  eMixedAir                           := 10,
  eEnablingTemperatureControl         := 11,
  eEnablingHumidityControl            := 12,
  eOn                                 := 13
) UDINT;
END_TYPE
```

Name	Description
eOff	Plant step Off.
eErc	Plant step Energy recovery.
ePreRinse	Pant step Pre-rinse.
eDamperOuA	Plant step Exhaust air damper.
eFanSupplyAir	Plant step Supply air fan.
eDamperExhA	Plant step External air damper.
eFanExtractAir	Plant step Exhaust air fan.
eCooler	Plant step Cooler.
eReHeater	Plant step Reheater.
eMixedAir	Plant step Mixed air.
eEnablingTemperatureControl	Plant step Enabling temperature control.
eEnablingHumidityControl	Plant step Enabling humidity control.
eOn	On

6.1.4.1.1.2.1.5 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
```

6.1.4.1.1.2.1.6 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	Plant step Off.
eOn	On
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

6.1.4.1.1.3 Universal

6.1.4.1.1.3.1 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
```

6.1.4.1.1.3.2 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	Plant step Off.
eOn	On

6.1.4.1.1.3.3 E_BA_EnergyLvlEx

The enumeration is used to represent the four building energy levels.

A reduced variant without the "*Precomfort*" level is represented by E_BA_EnergyLvl [▶ 700].

```
TYPE E_BA_EnergyLvlEx :
(
  eProtection    := 1,
  eEconomy       := 2,
  ePreComfort    := 3,
  eComfort       := 4
) UDINT;
END_TYPE
```

Name	Description
eProtection	"Protection" energy level.
eEconomy	"Economy" energy level.
ePreComfort	"Precomfort" energy level.
eComfort	"Comfort" energy level.

6.1.4.1.1.3.4 E_BA_EnergyLvl

The enumeration is used to represent the building energy level.

In contrast to E_BA_EnergyLvlEx [▶ 700], this reduced enumeration does not contain the "*Precomfort*" state.

```
TYPE E_BA_EnergyLvl :
(
  eProtection    := 1,
  eEconomy       := 2,
  eComfort       := 3
) UDINT;
END_TYPE
```

Name	Description
eProtection	*Protection* energy level.
eEconomy	*Economy* energy level.
eComfort	*Comfort* energy level.

6.1.4.1.1.3.5 E_BA_EnergyLvlA

The enumeration is used to represent the building energy level. The "Automatic" state is also listed, which allows automatic operation to be viewed.

In contrast to E_BA_EnergyLvlExA [▶ 701], this reduced enumeration does not contain the "Precomfort" state.

```
TYPE E_BA_EnergyLvlA :
(
  eAutomatic     := 1,
  eProtection    := 2,
  eEconomy       := 3,
```

```
  eComfort       := 4
) UDINT;
END_TYPE
```

Name	Description
eAutomatic	Automatic
eProtection	*Protection* energy level.
eEconomy	*Economy* energy level.
eComfort	*Comfort* energy level.

6.1.4.1.1.3.6 E_BA_EnergyLvlExA

The enumeration is used to represent the four building energy levels. The "Automatic" state is also listed, which allows automatic operation to be viewed.

A reduced variant without the "*Precomfort*" level is represented by E_BA_EnergyLvlA [▶ 700].

```
TYPE E_BA_EnergyLvlExA :
(
  eAutomatic     := 1,
  eProtection    := 2,
  eEconomy       := 3,
  ePreComfort    := 4,
  eComfort       := 5
) UDINT;
END_TYPE
```

Name	Description
eAutomatic	Automatic
eProtection	*Protection* energy level.
eEconomy	*Economy* energy level.
ePreComfort	*Precomfort* energy level.
eComfort	*Comfort* energy level.

6.1.4.1.2 Types

6.1.4.1.2.1 Actuator

6.1.4.1.2.1.1 ST_BA_SunblindActuatorFeedback

This structure contains feedback information from a sunblind actuator for a room (zone) user function.

Syntax

```
TYPE ST_BA_SunblindActuatorFeedback:
STRUCT
  bReferencing    : BOOL;
  bErr            : BOOL;
  ePrio           : BYTE;
  fPosition       : REAL;
  fAngle          : REAL;
  fAnglLmtUp      : REAL;
  fAnglLmtDwn     : REAL;
END_STRUCT
END_TYPE
```

Name	Type	Description
bReferencing	BOOL	The sunblind function or the controlled actuator is currently referencing.
bErr	BOOL	The sunblind function or the controlled actuator is in an error state.
ePrio	BYTE	Current priority of the telegram that controls the sunblind actuator.
fPosition	REAL	Current position of the drive.
fAngle	REAL	Currently approached angle of the lamellas.
fAnglLmtUp	REAL	Upper limit value [▶ 965] of the lamella position for further use in connection with the HMI.
fAnglLmtDwn	REAL	Lower limit value [▶ 965] of the lamella position for further use in connection with the HMI.

6.1.4.1.2.1.2 ST_BA_LightActuatorFeedback

This structure contains feedback information from a light actuator for a room (zone) user function.

Syntax

```
TYPE ST_BA_LightActuatorFeedback:
STRUCT
  bInitializing        : BOOL;
  bErr                 : BOOL;
  ePrio                : BYTE;
  fLightValue          : REAL;
  fLightTemperature    : REAL;
END_STRUCT
END_TYPE
```

Name	Type	Description
bInitializing	BOOL	The light function or the controlled actuator is being initialized.
bErr	BOOL	The light function or the controlled actuator is in an error state.
ePrio	BYTE	Current priority of the telegram that controls the light actuator.
fLightValue	REAL	Current light value of the actuator.
fLightTemperature	REAL	Current color temperature of the actuator.

6.1.4.1.2.2 Building

6.1.4.1.2.2.1 ST_BA_BuildingLighting

This structure is used as a transmit telegram to transmit building-specific enables and information concerning light control to other controllers.

Syntax

```
TYPE ST_BA_BuildingLighting:
STRUCT
  stLighting       : ST_BA_Lighting;
  bGlobalReset     : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	Resulting lighting telegram of the building related to the highest priority. This can include the following criteria: fire or burglary (see FB_BA_BuildingLighting [▶ 917]).
bGlobalReset	BOOL	Global reset of the lighting functions.

6.1.4.1.2.2.2 ST_BA_BuildingAlarms

This structure serves as a transmit telegram to transmit building-specific alarms to other controllers.

Syntax

```
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

6.1.4.1.2.2.3 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.

Syntax

```
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 [▶ 274]	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.

6.1.4.1.2.3 Facade

6.1.4.1.2.3.1 ST_BA_Facade

This structure serves as a transmit telegram, for the transmission of facade-specific data to other controllers.

Syntax

```
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 [▶ 274]	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.

6.1.4.1.2.4 General

6.1.4.1.2.4.1 ST_BA_GeneralSettings

This structure contains general weather parameters.

Syntax

```
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 *fTWthLowCrit*.
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 *fTWthLowCrit*.

6.1.4.1.2.5 WeatherStation

6.1.4.1.2.5.1 ST_BA_WeatherStation

This structure serves as a standard transmit telegram for the data of a weather station.

Syntax

```
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	TIMESTRUCT	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]

6.1.4.1.2.5.2 ST_BA_WSC11Data

This structure is used to transfer the data of a Thies WSC11 weather station.

Syntax

```
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 [▶ 707]	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]

6.1.4.1.2.5.3 ST_BA_WSC11Status

This structure is used to transfer the status of a Thies WSC11.

Syntax

```
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_Telegramm	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.

6.1.4.2 POUs

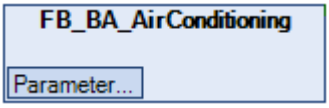
This chapter describes the function blocks contained in the template repository.

6.1.4.2.1 Trade

Templates from the individual technical systems.

6.1.4.2.1.1 AirConditioning

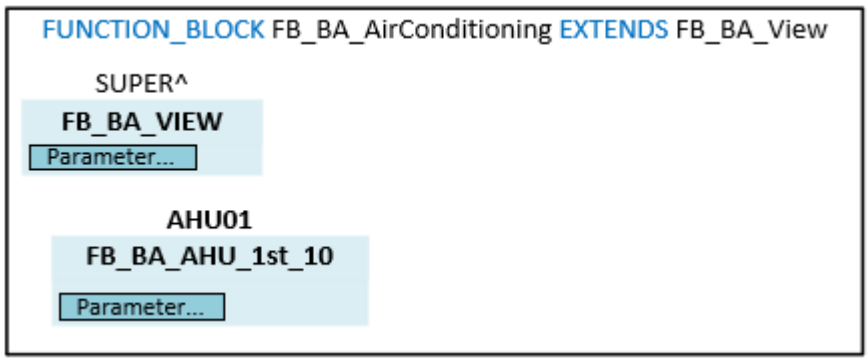
6.1.4.2.1.1.1 FB_BA_AirConditioning



The function block serves as a call template for plants for the *Air conditioning systems*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AirConditioning EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AHU01      : FB_BA_AHU_1st_10;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
AHU01	FB_BA_AHU_1st_10 [▶ 810]	The plant template is an air conditioning ventilation system and essentially consists of the following parts: • Plant control with the different operation modes and setpoints • Temperature control of sequence elements • Night cooling • Pressure-controlled supply and extract air fans • Thermal air treatment using air heaters, coolers and energy recovery by means of plate heat exchangers • External and extract air damper with spring return actuator and end position control • External and extract air filters with analog differential pressure monitoring

Requirements

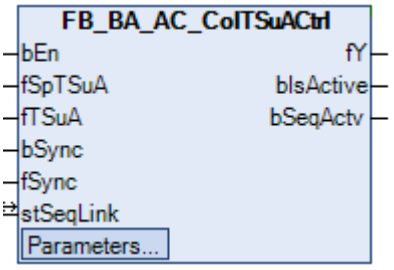
Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2 Aggregates

6.1.4.2.1.1.2.1 Cooler

6.1.4.2.1.1.2.1.1 Temperature

6.1.4.2.1.1.2.1.1.1 FB_BA_AC_ColTSuACtrl



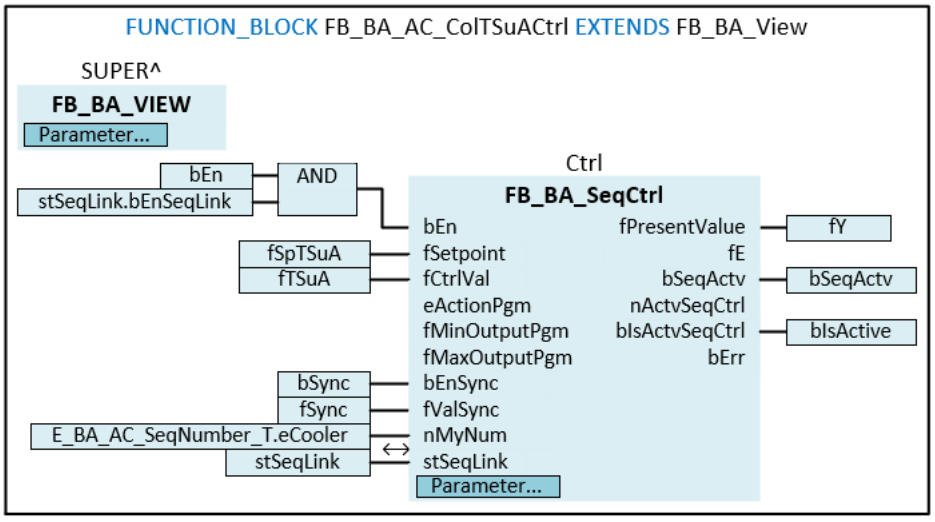
The template represents the supply air temperature control of a cooler with the sequence controller *Ctrl*.

The sequence controller is enabled via the input variable *bEn*.

The sequence controllers are enabled for sequence control using the *stSeqLink* data and command structure. This is indicated by the variable *bSeqActv*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ColTSuACtrl EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
  fSpTSuA            : REAL;
  fTSuA              : REAL;
  bSync              : BOOL;
  fSync              : REAL;
END_VAR
VAR_OUTPUT
  fY                 : REAL;
  bIsActive          : BOOL;
  bSeqActv           : BOOL;
END_VAR
VAR_IN_OUT
```

```
  stSeqLink           : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  Ctrl                : FB_BA_SeqCtrl;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enable for the frequency controller *Ctrl*.
fSpTSuA	REAL	Setpoint of the supply air temperature.
fTSuA	REAL	Measured value of the supply air temperature.
bSync	BOOL	Pulse for synchronization of the sequence controller *Ctrl*.
fSync	REAL	Value for the synchronization of the sequence controller *Ctrl*.

Outputs

Name	Type	Description
fY	REAL	Control value output
bIsActive	BOOL	The sequence controller is the active one in the sequence control.
bSeqActv	BOOL	The sequence controller is implemented in the control sequence.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.

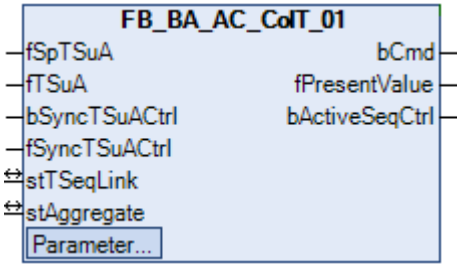
Inputs CONSTANT

Name	Type	Description
Ctrl	FB_BA_SeqCtrl [▶ 168]	The supply air temperature sequence controller *Ctrl* is the core of this template. It is responsible for the supply air temperature control of the cooler. The sequence controller is also part of the supply air temperature sequence control of an air conditioning system, see sample FB_BA_AC_SeqT [▶ 778]. The data exchange within this sequence control takes place via the data and command structure *stSeqLink*. The global variable E_BA_AC_SeqNumber_T.eCooler [▶ 697] gives the sequence controller its sequence number within the temperature sequence control.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.1.1.2 FB_BA_AC_ColT_01



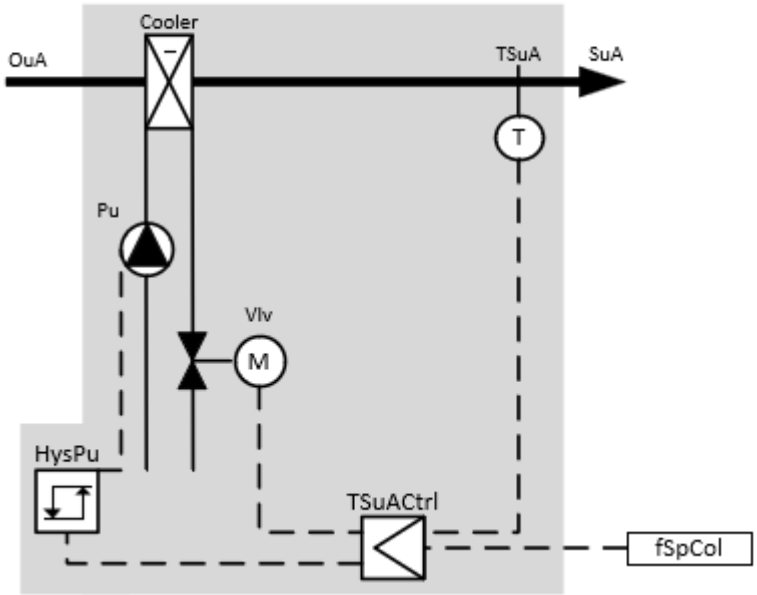
The template represents the open-loop and closed-loop control of a cold water air cooler without dehumidification control.

The main tasks of the template are:

- Control of the supply air temperature, see *TSuACtrl*
- Enable the cooler pump, see *Pu*
- Control of the cooler valve, see *Vlv*
- Collecting and evaluating safety-relevant faults using the *PlantLock*
- To be part of the step sequence control of an air conditioning system, see *Aggregates*

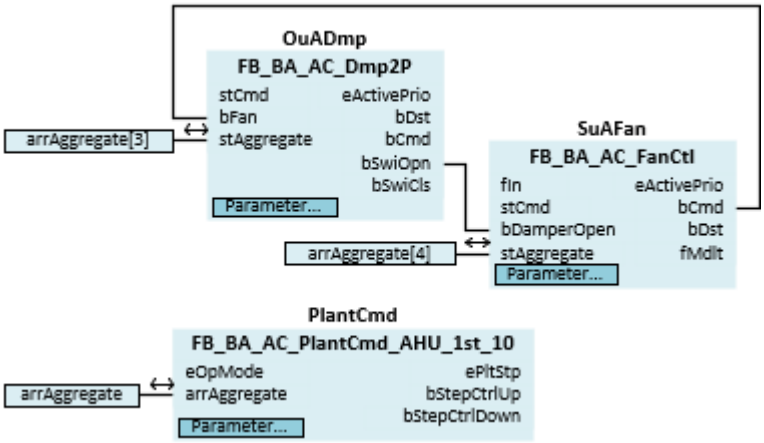
Principle diagram 01

The diagram shows the intended use of the template with the plant elements involved.



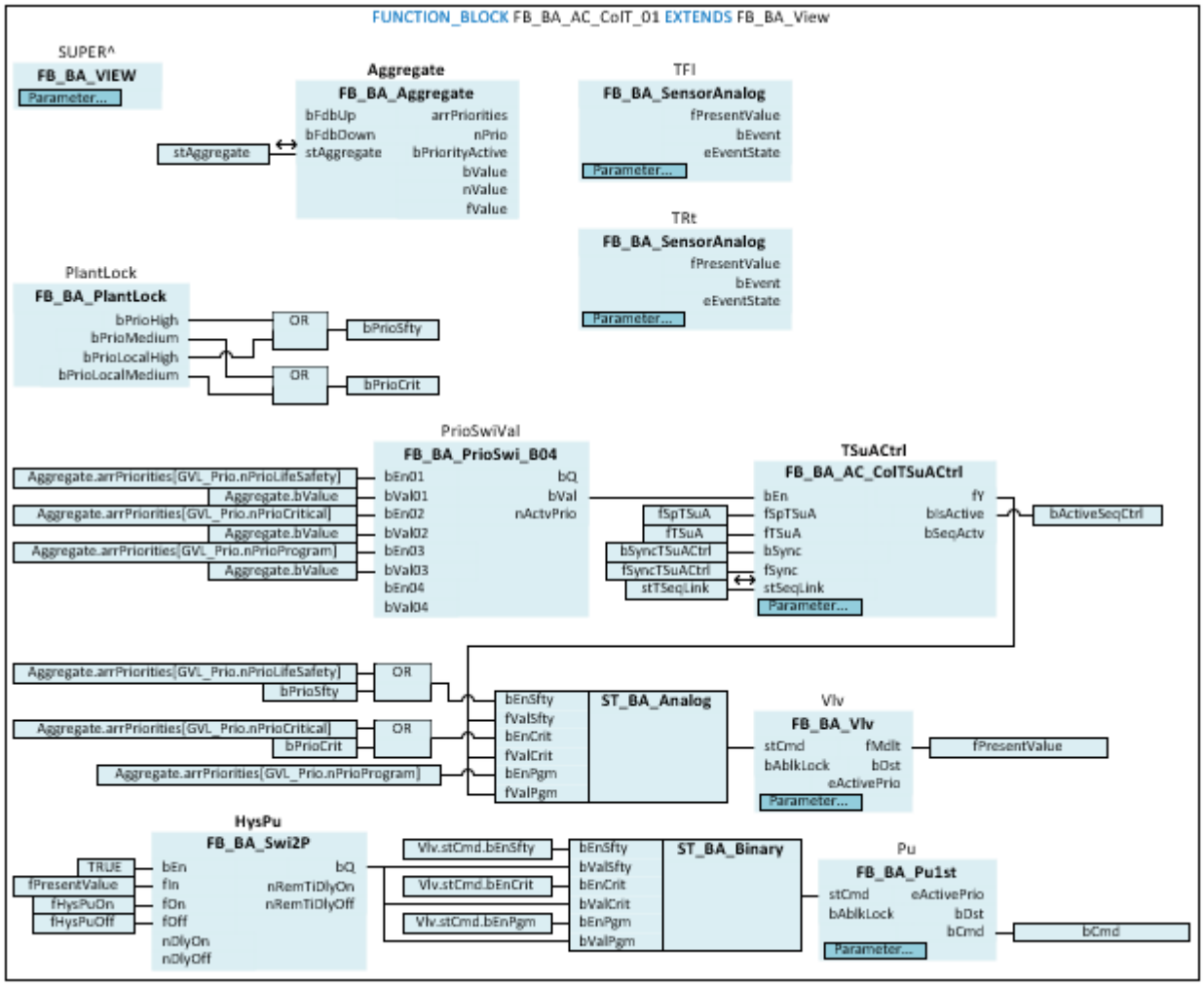
Principle diagram 02

The diagram shows the integration of the template within a plant.



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ColT_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA           : REAL;
  fTSuA             : REAL;
  bSyncTSuACtrl     : BOOL;
  fSyncTSuACtrl     : REAL;
END_VAR
VAR_OUTPUT
```

```
  bCmd              : BOOL;
  fPresentValue     : REAL;
  bActiveSeqCtrl    : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink        : ST_BA_SeqLink;
  stAggregate       : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOn          : REAL := 5.0;
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOff         : REAL := 1.0;
END_VAR
VAR_INPUT CONSTANT
  TFl               : FB_BA_SensorAnalog;
  TRt               : FB_BA_SensorAnalog;
  TSuACtrl          : FB_BA_AC_ColTSuACtrl;
  Vlv               : FB_BA_Vlv;
  Pu                : FB_BA_Pu1st;
  PlantLock         : FB_BA_PlantLock;
  Aggregate         : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty         : BOOL;
  bPrioCrit         : BOOL;
  PrioSwiVal        : FB_BA_PrioSwi_B04;
  HysPu             : FB_BA_Swi2P;
END_VAR
```

Inputs

Name	Type	Description
fSpTSuA	REAL	Setpoint of the supply air temperature
fTSuA	REAL	Measured value of the supply air temperature
bSyncTSuACtrl	BOOL	Input for synchronization of the supply air sequence controller in the function block *TSuACtrl*.
fSyncTSuACtrl	REAL	Synchronization value for the supply air sequence controller in the function block *TSuACtrl*.

Outputs

Name	Type	Description
bCmd	BOOL	Current switching status of the single-stage pump
fPresentValue	REAL	Current value of the cooler valve
bActiveSeqCtrl	BOOL	Indicates that the sequence controller of the cooler is the active one in the sequence control.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT PERSISTENT

Name	Type	Description
fHysPuOn	REAL	Upper switching point of the hysteresis to switch on the pump.
fHysPuOff	REAL	Lower switching point of the hysteresis to switch off the pump.

Inputs CONSTANT

Name	Type	Description
TFl	FB_BA_SensorAnalog [▶ 1087]	The function block represents the flow temperature sensor.
TRt	FB_BA_SensorAnalog [▶ 1087]	The function block represents the return temperature sensor.
TSuACtrl	FB_BA_AC_ColTSuACtrl [▶ 710]	The function block represents the supply air temperature control of the cooler and is part of the temperature sequence control of an air conditioning system. The control signal is forwarded to the valve *Vlv*.
Vlv	FB_BA_Vlv [▶ 1092]	The function block represents the valve.
Pu	FB_BA_Pu1st [▶ 1074]	The function block represents the cooler pump.
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the template. These relevant faults cause specific switching actions via the variables *bPrioSfty* and *bPrioCrit* in the template. The parameterization of the lock priority of the event-enabled objects can be found in the *FB_init* of this template.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

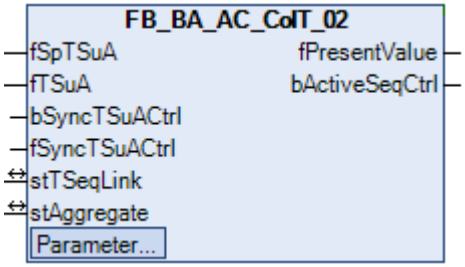
Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch *PrioSwiVal* determines the switching conditions and enables for the supply air temperature control *TSuACtrl* using the aggregate function block *Aggregate* and the global variable list Priority [▶ 1119].
HysPu	FB_BA_Swi2P	The two-position switch *HysPu* switches the pump *Pu* on and off depending on the valve position *fPresentValue* and the switching points of the hysteresis *fHysPuOn*/ *fHysPuOff*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.1.1.3 FB_BA_AC_ColT_02



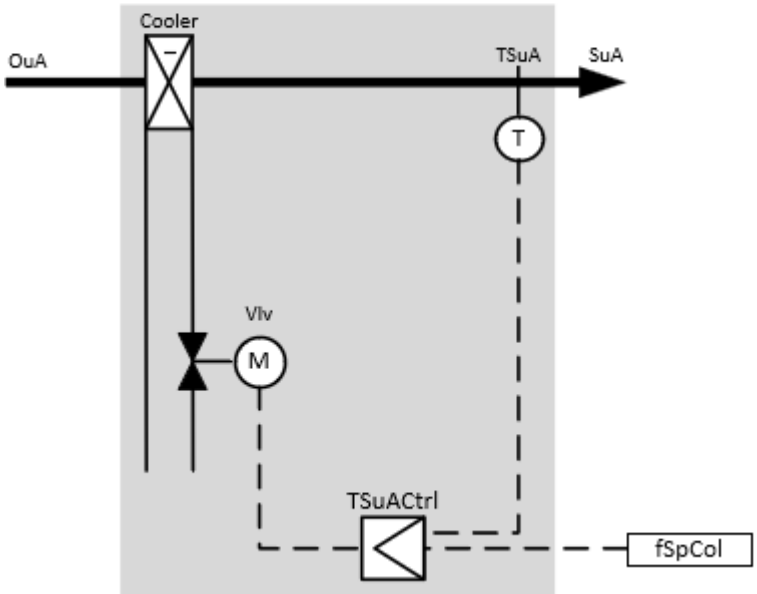
The template represents the open-loop and closed-loop control of a cold water air cooler without dehumidification control.

The main tasks of the template are:

- Control of the supply air temperature, see *TSuACtrl*
- Control of the cooler valve, see *Vlv*
- Collecting and evaluating safety-relevant faults using the *PlantLock*
- To be part of the step sequence control of an air conditioning system, see *Aggregates*

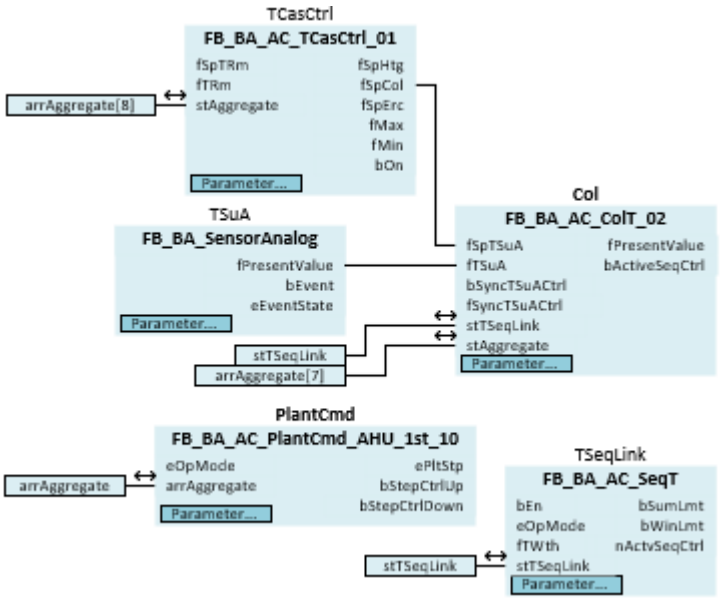
Principle diagram 01

The diagram shows the intended use of the template with the plant elements involved.



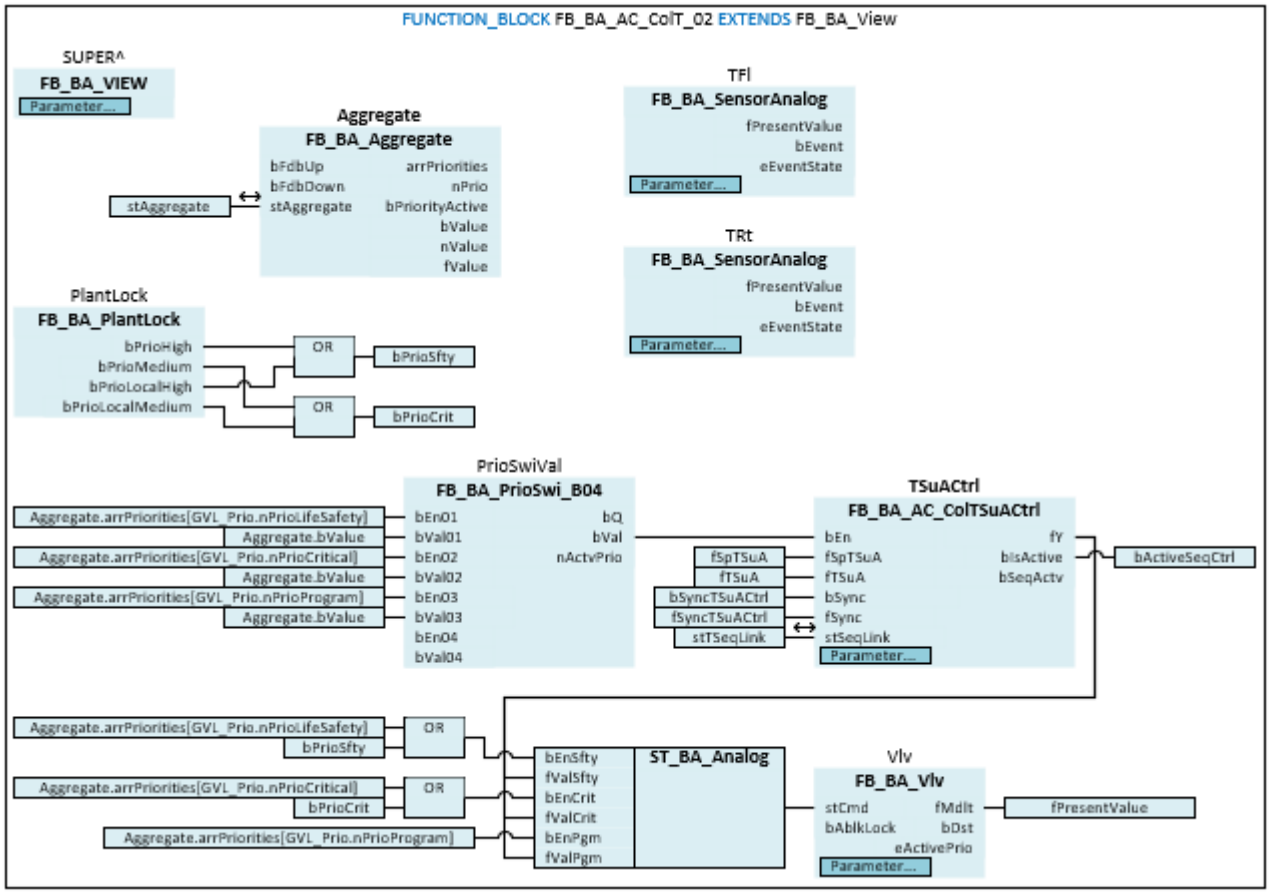
Principle diagram 02

The diagram shows the integration of the template within a plant.



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ColT_02 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA           : REAL;
  fTSuA             : REAL;
  bSyncTSuACtrl     : BOOL;
  fSyncTSuACtrl     : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue     : REAL;
  bActiveSeqCtrl    : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink        : ST_BA_SeqLink;
  stAggregate       : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  TFl               : FB_BA_SensorAnalog;
  TRt               : FB_BA_SensorAnalog;
  TSuACtrl          : FB_BA_AC_ColTSuACtrl;
  Vlv               : FB_BA_Vlv;
  PlantLock         : FB_BA_PlantLock;
  Aggregate         : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty         : BOOL;
  bPrioCrit         : BOOL;
  PrioSwiVal        : FB_BA_PrioSwi_B04;
END_VAR
```

Inputs

Name	Type	Description
fSpTSuA	REAL	Setpoint of the supply air temperature
fTSuA	REAL	Measured value of the supply air temperature
bSyncTSuACtrl	BOOL	Input for synchronization of the supply air sequence controller in the function block *TSuACtrl*.
fSyncTSuACtrl	REAL	Synchronization value for the supply air sequence controller in the function block *TSuACtrl*.

Outputs

Name	Type	Description
fPresentValue	REAL	Current value of the cooler valve.
bActiveSeqCtrl	BOOL	Indicates that the sequence controller of the cooler is the active one in the sequence control.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

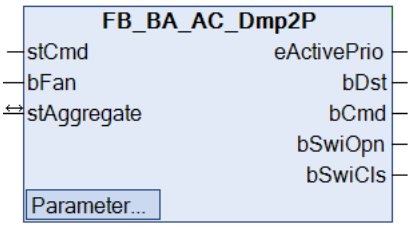
Name	Type	Description
TFl	FB_BA_SensorAnalog [▶ 1087]	The function block represents the flow temperature sensor.
TRt	FB_BA_SensorAnalog [▶ 1087]	The function block represents the return temperature sensor.
TSuACtrl	FB_BA_AC_ColTSuACtrl [▶ 710]	The function block represents the supply air temperature control of the cooler and is part of the temperature sequence control of an air conditioning system. The control signal is forwarded to the valve *Vlv*.
Vlv	FB_BA_Vlv [▶ 1092]	The function block represents the valve.
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the template. These relevant faults cause specific switching actions via the variables *bPrioSfty* and *bPrioCrit* in the template. The parameterization of the lock priority of the event-enabled objects can be found in the *FB_init* of this template.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch *PrioSwiVal* determines the switching conditions and enables for the supply air temperature control *TSuACtrl* using the aggregate function block *Aggregate* and the global variable list Priority [▶ 1119].

6.1.4.2.1.1.2.2 Damper

6.1.4.2.1.1.2.2.1 FB_BA_AC_Dmp2P



The aggregate template represents the control of a two-point damper with integrated monitoring of both end positions.

Typical applications of the template are the fresh air and exhaust air dampers of an air conditioning system.

An essential component of the damper control is the template

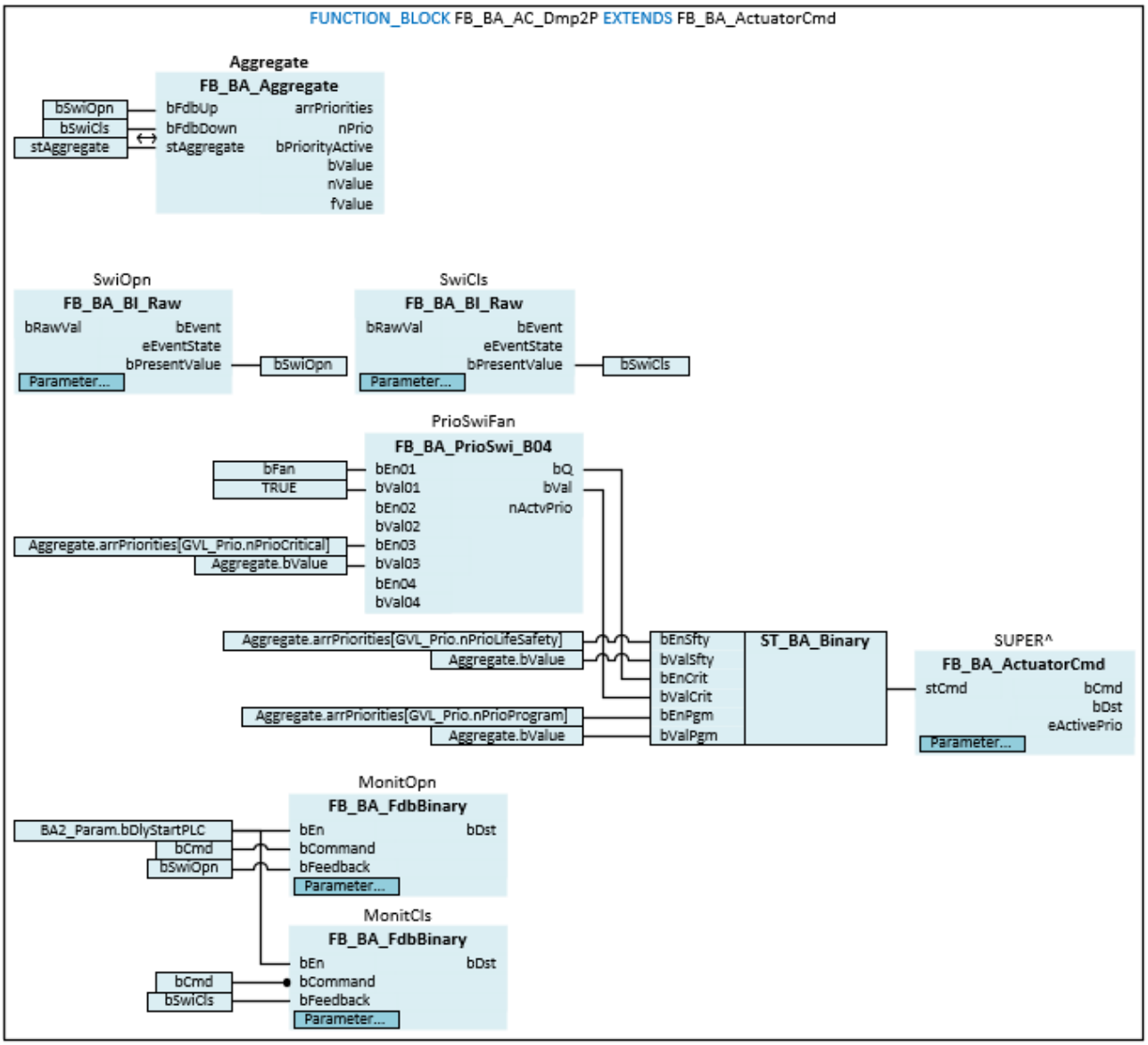
FB_BA_ActuatorCmd [▶ 986]

The damper is integrated into the system control program of the ventilation system using the function block *Aggregate*.

Each template is available in 2 different configurations regarding the connection to the I/O process level, see I/O mapping. This documentation refers to the RAW variant.

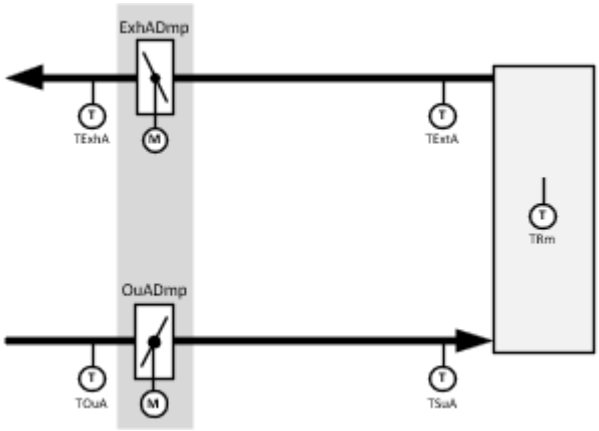
The initialization of the template takes place within the method FB_Init.

Block diagram



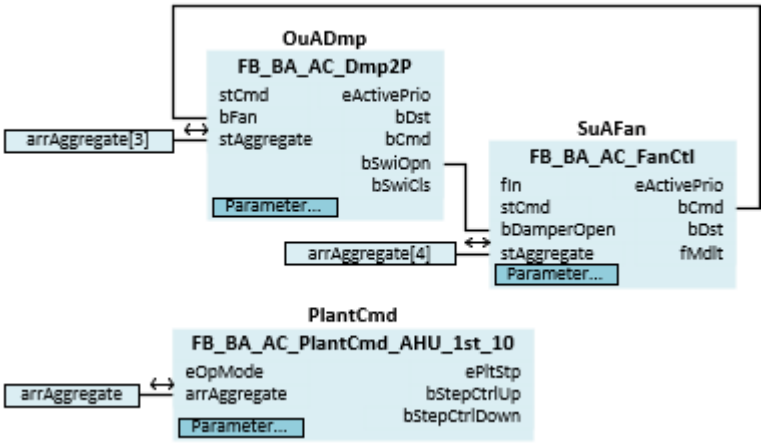
Principle diagram 01

The diagram shows the intended use of the template with the plant elements involved.



Principle diagram 02

The diagram shows the integration of the template within a plant.



Syntax

```
FUNCTION_BLOCK FB_BA_AC_Dmp2P EXTENDS FB_BA_ActuatorCmd
VAR_INPUT
  bFan                    : BOOL;
END_VAR
VAR_OUTPUT
  bSwiOpn                 : BOOL;
  bSwiCls                 : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate             : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  SwiOpn                  : FB_BA_BI_Raw;
  SwiCls                  : FB_BA_BI_Raw;
  MonitOpen               : FB_BA_FdbBinary;
  MonitClose              : FB_BA_FdbBinary;
  Aggregate               : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiFan              : FB_BA_PrioSwi_B04;
END_VAR
```

Inputs

Name	Type	Description
bFan	BOOL	The "Fan On" input signal is used within a ventilation and air conditioning system to protect the connected damper. As long as the input signal "Fan On" is TRUE, the priority "Critical" is enabled on the priority switch *PrioSwiFan* within the template and the damper is switched on.

Outputs

Name	Type	Description
bSwiOpn	BOOL	End position "open" of the damper has been reached.
bSwiCls	BOOL	End position "closed" of the damper has been reached.

Inputs/outputs

Name	Type	Description
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control Aggregate. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
SwiOpn	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the Open limit switch.
SwiCls	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the limit switch Closed.
MonitOpn	FB_BA_FdbCtrlBinary [▶ 467]	The template monitors the end position “open” of the damper.
MonitCls	FB_BA_FdbCtrlBinary [▶ 467]	The template monitors the end position “Closed” of the damper.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

Variables

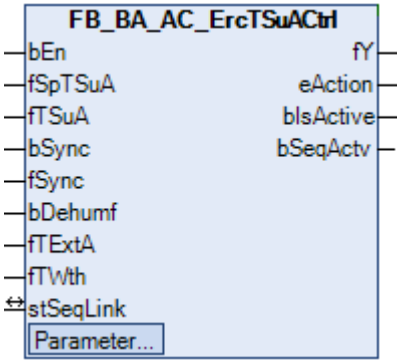
Name	Type	Description
PrioSwiFan	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch *PrioSwiFan* uses the aggregate function block *Aggregate* and the input variable *bFan* to determine the enable conditions and switch-on values for the priority critical of the damper.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.2.3 ERC

6.1.4.2.1.1.2.3.1 FB_BA_AC_ErcTSuACtrl



The template represents the supply air temperature control of an energy recovery system with the sequence controller *Ctrl*.

The sequence controller is enabled via the input variable *bEn*. If the control direction of the energy recovery system is indirect (heating mode) and the air conditioning system is in dehumidification mode *bDehumf* at the same time, energy recovery is blocked.

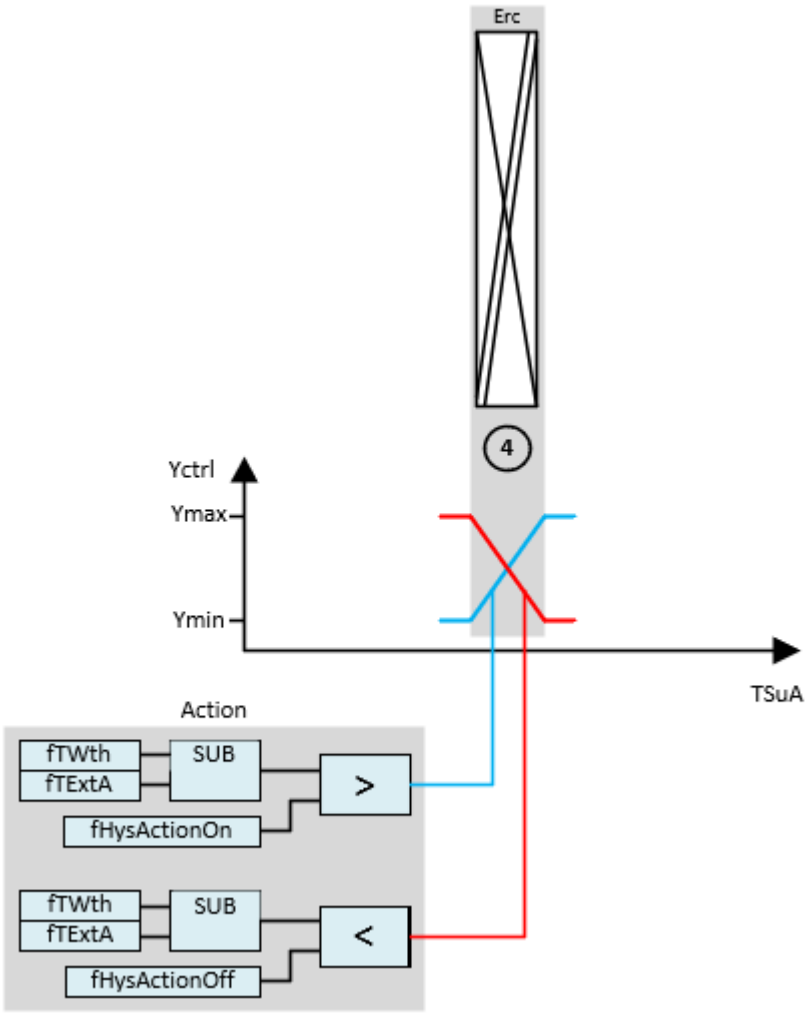
The sequence controllers are enabled for sequence control using the *stSeqLink* data and command structure. This is indicated by the variable *bSeqActv*.

Control direction

The control direction of the sequence controller *Ctrl* is selected based on a comparison of the outside temperature with the extract air temperature.

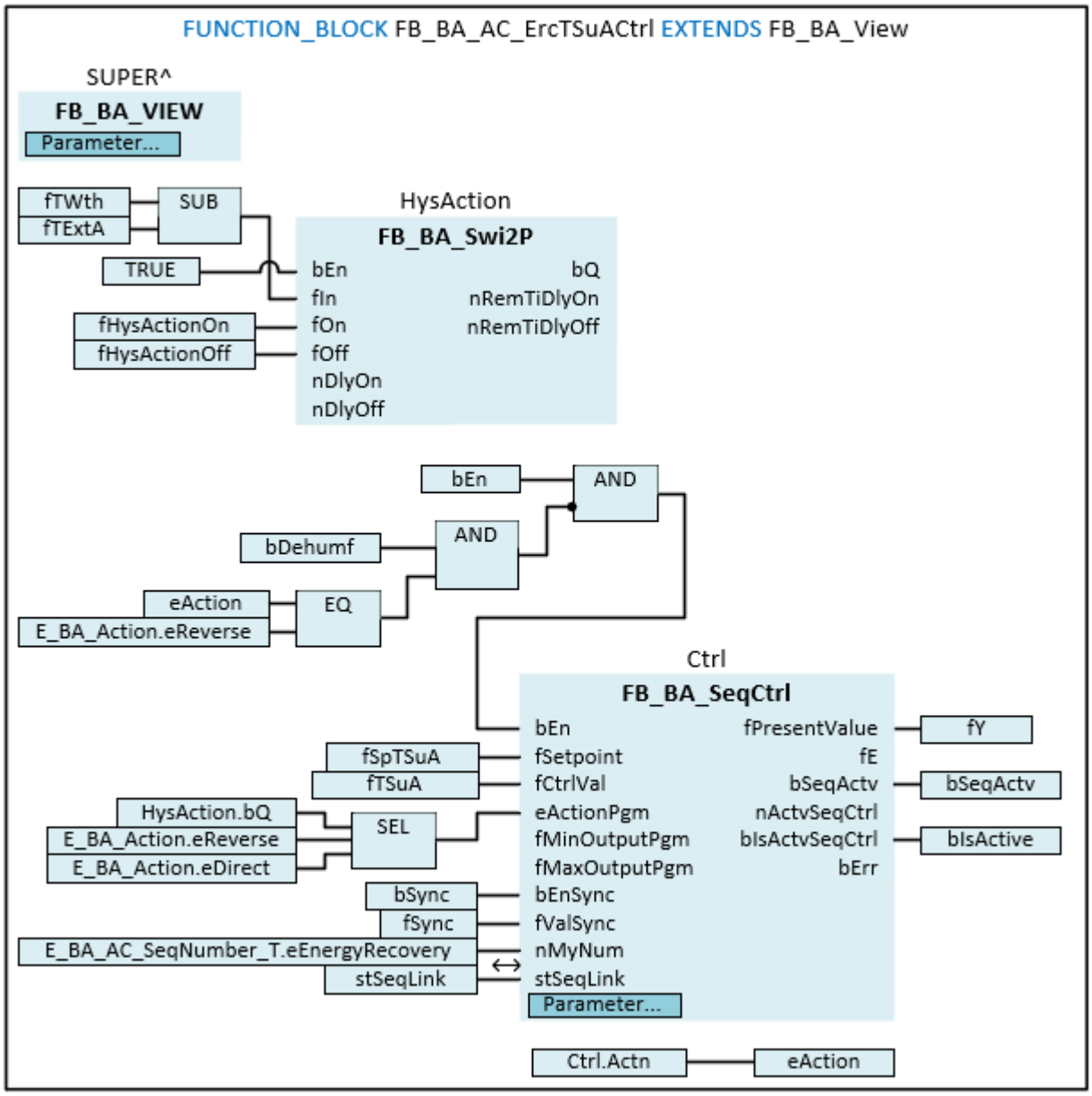
If the outside temperature is lower than the extract air temperature, the control direction of the sequence controller *Ctrl* is indirect (heating mode), see E_BA_Action.

If the outside temperature is higher than the extract air temperature, the control direction of the sequence controller *Ctrl* is direct (cooling mode), see E_BA_Action.



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcTSuACtrl EXTENDS FB_BA_View
VAR_INPUT
  bEn                 : BOOL;
  fSpTSuA             : REAL;
  fTSuA               : REAL;
  bSync               : BOOL;
  fSync               : REAL;
  bDehumf             : BOOL;
  fTExtA              : REAL;
  fTWth               : REAL;
END_VAR
VAR_OUTPUT
  fY                  : REAL;
  eAction             : E_BA_Action;
  bIsActive           : BOOL;
  bSeqActv            : BOOL;
END_VAR
VAR_IN_OUT
  stSeqLink           : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysActionOn        : REAL := 0.25;
  {attribute 'parameterCategory' := 'Behaviour'}
```

```
  fHysActionOff       : REAL := -0.25;
END_VAR
VAR_INPUT CONSTANT
  Ctrl                : FB_BA_SeqCtrl;
END_VAR
VAR
  HysAction           : FB_BA_Swi2P;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enable for the frequency controller *Ctrl*.
fSpTSuA	REAL	Setpoint of the supply air temperature.
fTSuA	REAL	Measured value of the supply air temperature.
bSync	BOOL	Pulse for synchronization of the sequence controller *Ctrl*.
fSync	REAL	Value for the synchronization of the sequence controller *Ctrl*.
bDehumf	BOOL	Indicates with a TRUE that the air conditioning system is in dehumidification mode.
fTExtA	REAL	Measured value extract air temperature
fTWth	REAL	Measured value weather temperature

Outputs

Name	Type	Description
fY	REAL	Control value output
eAction	E_BA_Action	The output of the control direction of the supply air controller *Ctrl* is required within an air conditioning system for the setpoint strategy.
bIsActive	BOOL	The sequence controller is the active one in the sequence control.
bSeqActv	BOOL	The sequence controller is implemented in the control sequence.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.

Inputs CONSTANT PERSISTENT

Name	Type	Description
fHysActionOn	REAL	Upper switching point of the hysteresis *HysAction* to determine the control direction of the supply air controller *Ctrl*.
fHysActionOff	REAL	Lower switching point of the hysteresis *HysAction* to determine the control direction of the supply air controller *Ctrl*.

Inputs CONSTANT

Name	Type	Description
Ctrl	FB_BA_SeqCtrl [▶ 168]	The supply air temperature sequence controller *Ctrl* is the core of this template. It is responsible for the supply air temperature control of the cooler. The sequence controller is also part of the supply air temperature sequence control of an air conditioning system, see sample FB_BA_AC_SeqT [▶ 778]. The data exchange within this sequence control takes place via the data and command structure *stSeqLink*. The global variable E_BA_AC_SeqNumber_T.eEnergyRecovery [▶ 697] gives the sequence controller its sequence number within the temperature sequence control.

Variables

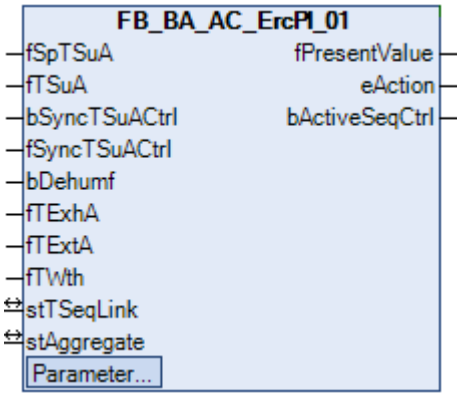
Name	Type	Description
HysAction	FB_BA_Swi2P	The two-position switch determines the control direction of the sequence controller depending on the outside temperature *fTWth* and the extract air temperature *fTExtA* and the switching points of the hysteresis *fHysActionOn*/ *fHysActionOff*. If the subtraction of the outside temperature *fTWth* and the extract air temperature *fTExtA* > *fHysActionOn*, then the output of the function block *HysAction* is TRUE. The control direction of the sequence controller *Ctrl* is therefore direct and energy recovery is in cooling mode. If the subtraction of the outside temperature *fTWth* and the extract air temperature *fTExtA* < *fHysActionOff*, then the output of the function block *HysAction* is FALSE. The control direction of the sequence controller *Ctrl* is therefore indirect and energy recovery is in heating mode.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.3.2 Plate

6.1.4.2.1.1.2.3.2.1 FB_BA_AC_ErcPl_01



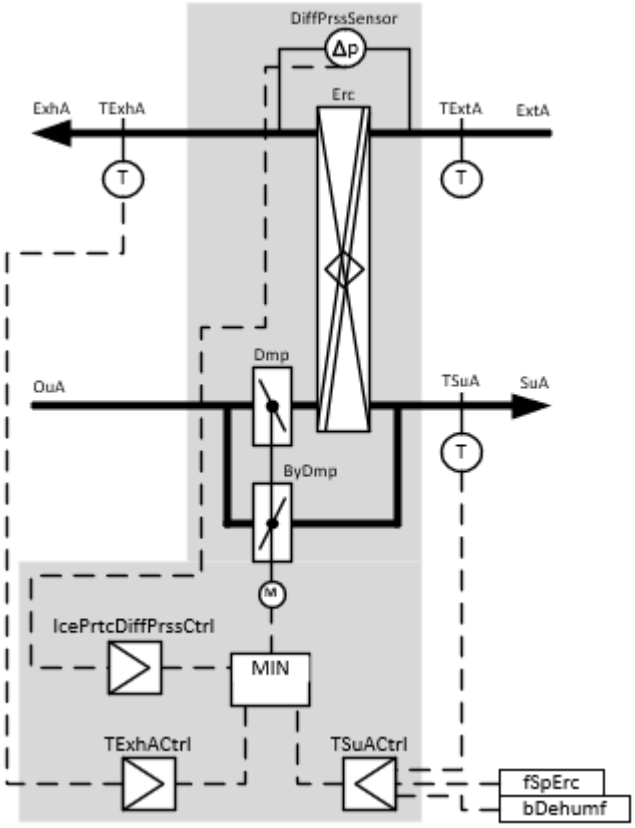
The template represents the open-loop and closed-loop control of an energy recovery system with plate heat exchanger.

The main tasks of the template are:

- Control of the supply air temperature
- Minimum limitation of exhaust air temperature
- Anti-icing of the heat exchanger by means of a differential pressure sensor above the heat exchanger
- Control of a bypass damper system
- Collecting and evaluating safety-relevant faults using the *PlantLock*
- To be part of the step sequence control of an air conditioning system, see *Aggregates*

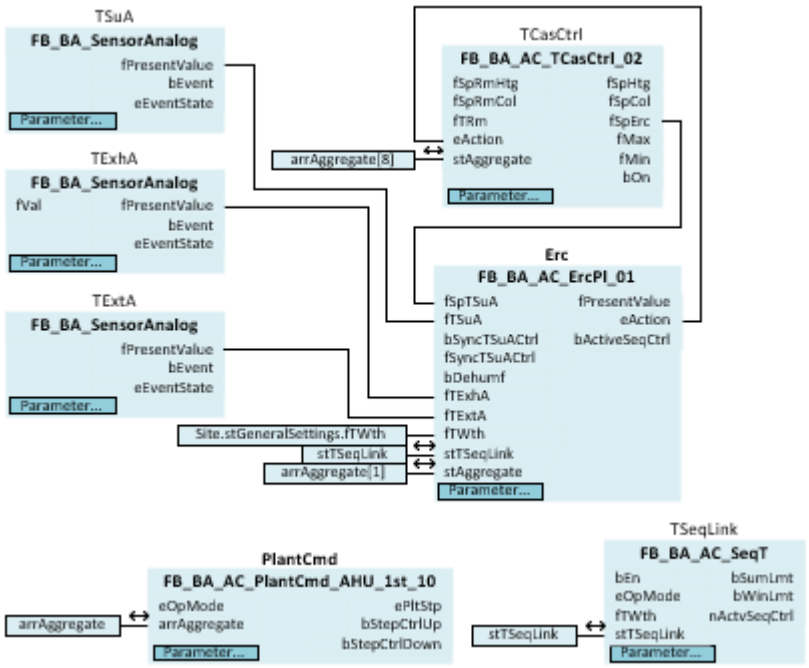
The exhaust air minimum limiter *TExhACtrl* and the frost protection program *IcePrtcDiffPrssCtrl* limit the control value of the temperature sequence controller *TSuACtrl* of the energy recovery system via a minimum selection to prevent the heat exchanger from icing up.

Principle diagram 01



The diagram shows the intended use of the template with the plant elements involved.

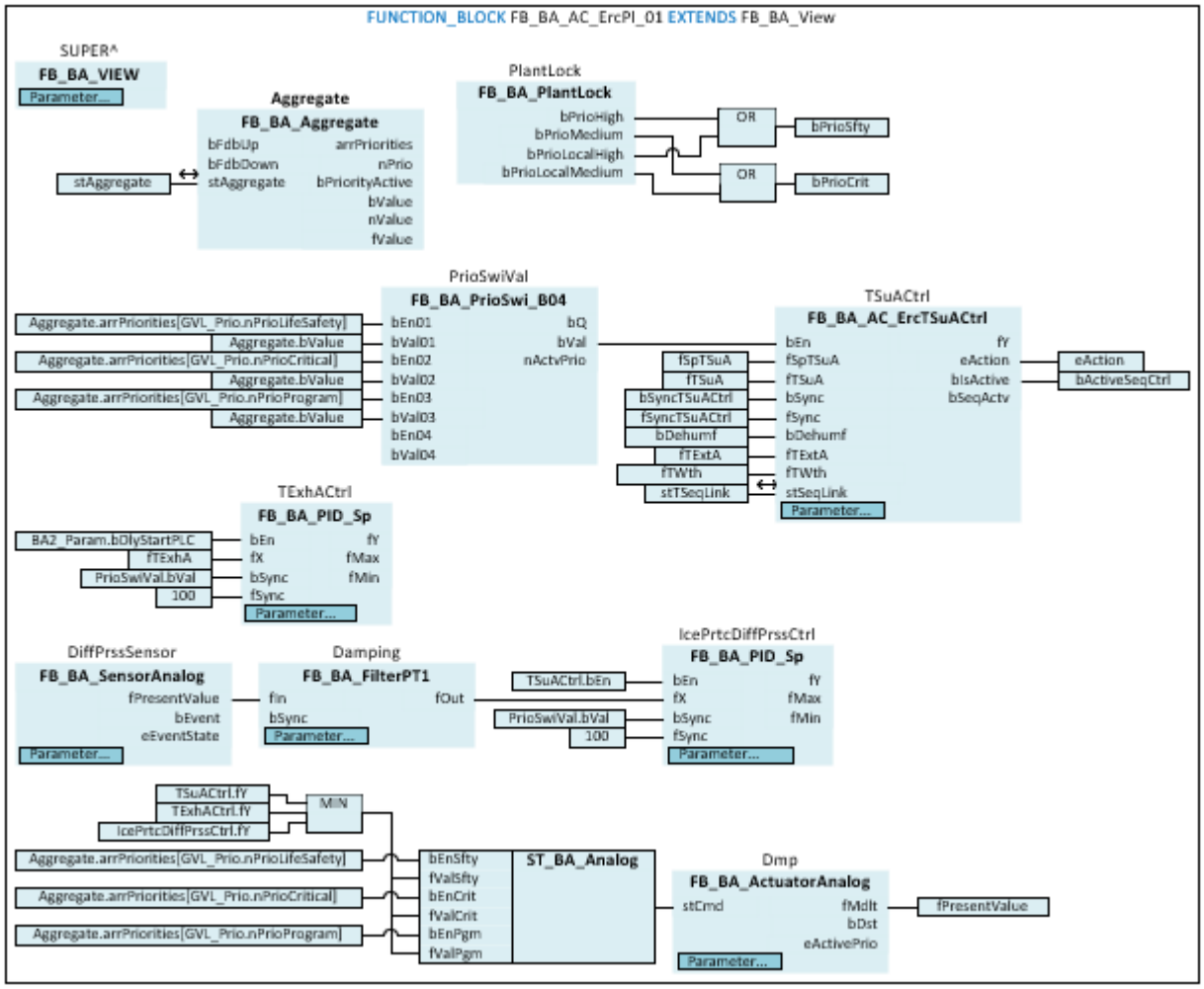
Principle diagram 02



The diagram shows the integration of the template within a plant.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcPl_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA                 : REAL;
  fTSuA                   : REAL;
  bSyncTSuACtrl           : BOOL;
  fSyncTSuACtrl           : REAL;
  bDehumf                 : BOOL;
  fTExhA                  : REAL;
  fTExtA                  : REAL;
  fTWth                   : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue           : REAL;
  eAction                 : E_BA_Action;
  bActiveSeqCtrl          : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink              : ST_BA_SeqLink;
  stAggregate             : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  TSuACtrl                : FB_BA_AC_PreHtrTSuACtrl;
  TExhACtrl               : FB_BA_PID_Sp;
  Dmp                     : FB_BA_ActuatorAnalog;
  DiffPrssSensor          : FB_BA_SensorAnalog;
  Damping                 : FB_BA_FilterPT1;
  IcePrtcDiffPrssCtrl     : FB_BA_PID_Sp;
  PlantLock               : FB_BA_PlantLock;
  Aggregate               : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty               : BOOL;
```

```
  bPrioCrit               : BOOL;
  PrioSwiVal              : FB_BA_PrioSwi_B04;
END_VAR
```

Inputs

Name	Type	Description
fSpTSuA	REAL	Setpoint of the supply air temperature
fTSuA	REAL	Measured value of the supply air temperature
bSyncTSuACtrl	BOOL	Pulse for synchronization of the sequence controller *TSuACtrl*.
fSyncTSuACtrl	REAL	Value for the synchronization of the supply air sequence controller *TSuACtrl*.
bDehumf	BOOL	Input dehumidification mode active. This state has an effect on the sequence control in the supply air temperature control *TSuACtrl* template.
fTExhA	REAL	Measured value exhaust air temperature
fTExtA	REAL	Measured value extract air temperature
fTWth	REAL	Measured value weather temperature

Outputs

Name	Type	Description
fPresentValue	REAL	Current value of the bypass damper system
eAction	E_BA_Action	The output of the control direction of the supply air controller *TSuACtrl* is required within an air conditioning system for the setpoint strategy.
bActiveSeqCtrl	BOOL	The sequence controller *TSuACtrl* is the active one in the sequence control.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 746]	The function block represents the supply air temperature control of energy recovery and is part of the temperature sequence control of an air conditioning system. The control signal is forwarded to the analog control of damper *Dmp* via a minimum selection.
TExhACtrl	FB_BA_PID_Sp [▶ 1035]	The function block represents the exhaust air temperature control for the anti-icing of the plate heat exchanger.
Dmp	FB_BA_ActuatorAnalog [▶ 982]	The function block is used to control the analog damper of the plate heat exchanger.
DiffPrssSensor	FB_BA_SensorAnalog [▶ 1087]	The analog input object represents a differential pressure sensor above the heat exchanger. This is the control variable for the differential pressure control *IcePrtcDiffPrssCtrl*.
Damping	FB_BA_FilterPT1 [▶ 1003]	The function block is used to damp the differential pressure sensor *DiffPrssSensor*.
IcePrtcDiffPrssCtrl	FB_BA_PID_Sp [▶ 1035]	The function block represents the differential pressure control for the anti-icing of the heat exchanger.
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the template. These relevant faults cause specific switching actions via the variables *bPrioSfty* and *bPrioCrit* in the template. The parameterization of the lock priority of the event-enabled objects can be found in the *FB_init* of this template.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

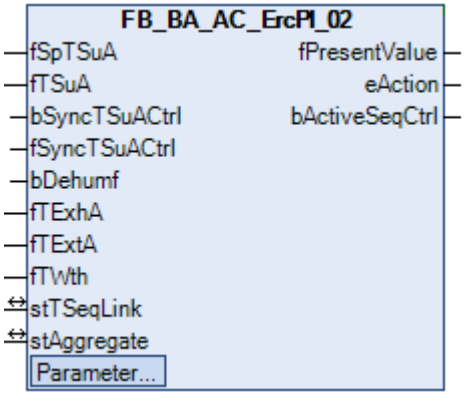
Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the Safety lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
bPrioCrit	BOOL	The variable is an evaluation of the Critical lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch *PrioSwiVal* determines the switching conditions and enables for the supply air temperature control *TSuACtrl* and the differential pressure control for anti-icing *IcePrtcDiffPrssCtrl* using the aggregate function block *Aggregate* and the global variable list Priority [▶ 1119].

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.3.2.2 FB_BA_AC_ErcPl_02



The template represents the open-loop and closed-loop control of an energy recovery system with a plate heat exchanger.

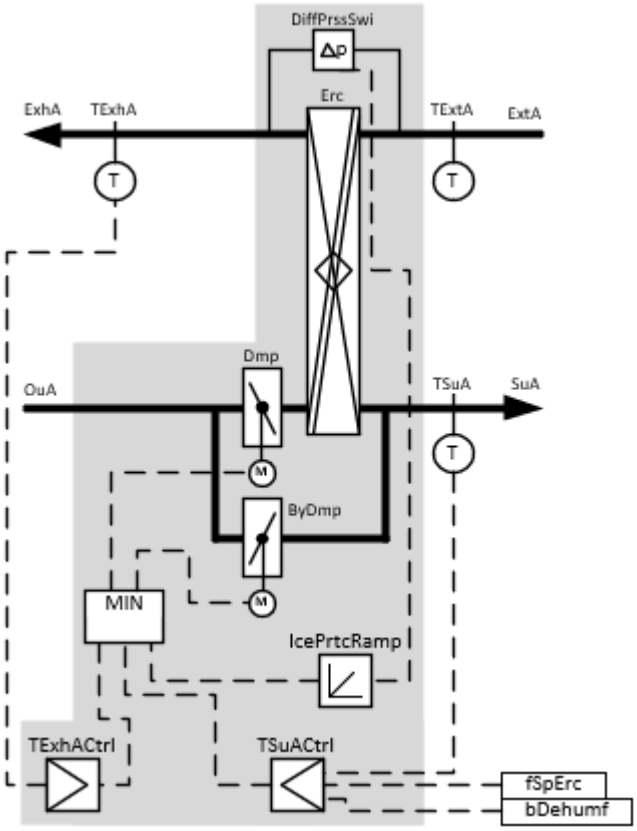
The main tasks of the template are:

- Control of the supply air temperature
- Minimum limitation of exhaust air temperature
- Anti-icing of the heat exchanger by means of a differential pressure monitor above the heat exchanger
- Control of a damper
- Control of a bypass damper
- Collecting and evaluating safety-relevant faults using the *PlantLock*
- To be part of the step sequence control of an air conditioning system, see *Aggregates*

The exhaust air minimum limiter *TExhACtrl* and the frost protection program *IcePrtcRamp* limit the control value of the temperature sequence controller *TSuACtrl* of the energy recovery system via a minimum selection to prevent the heat exchanger from icing up.

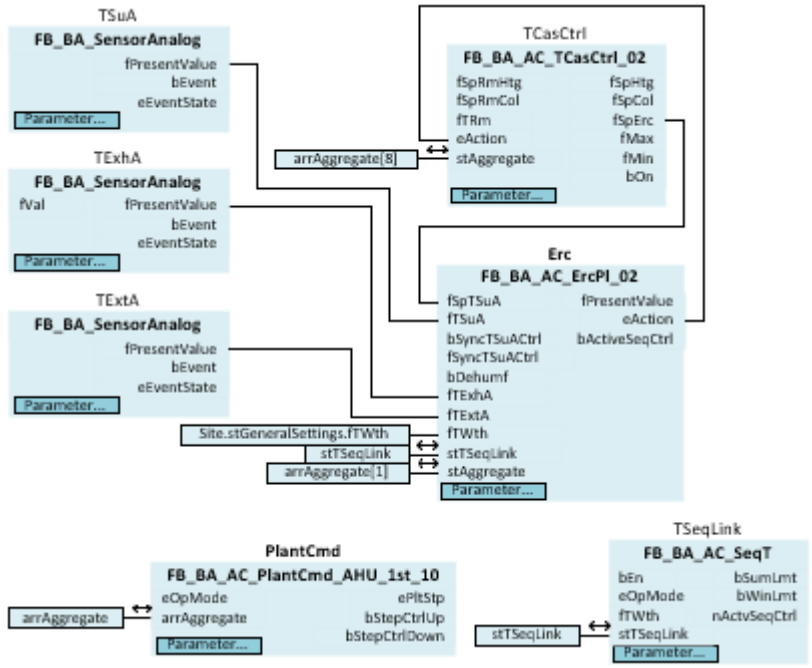
Principle diagram 01

The diagram shows the intended use of the template with the plant elements involved.



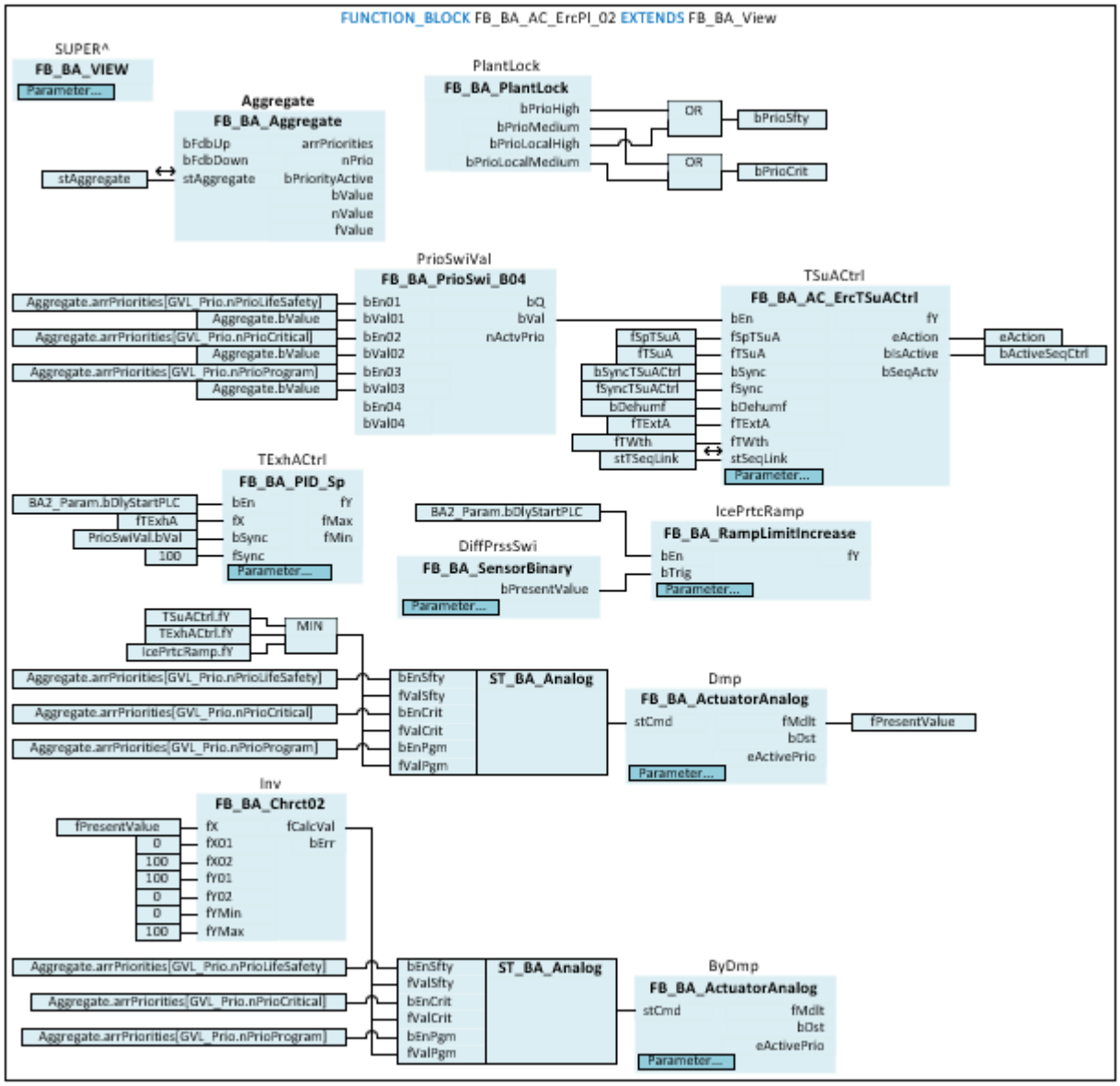
Principle diagram 02

The diagram shows the integration of the template within a plant.



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcPl_02 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA           : REAL;
  fTSuA             : REAL;
  bSyncTSuACtrl     : BOOL;
  fSyncTSuACtrl     : REAL;
  bDehumf           : BOOL;
  fTExhA            : REAL;
  fTExtA            : REAL;
  fTWth             : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue     : REAL;
  eAction           : E_BA_Action;
  bActiveSeqCtrl    : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink        : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  TSuACtrl          : FB_BA_AC_PreHtrTSuACtrl;
  TExhACtrl         : FB_BA_PID_Sp;
  DiffPrssSwi       : FB_BA_SensorBinary;
  IcePrtcRamp       : FB_BA_PID_Sp;
  Dmp               : FB_BA_ActuatorAnalog;
  ByDmp             : FB_BA_ActuatorAnalog;
```

```
  PlantLock         : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty         : BOOL;
  bPrioCrit         : BOOL;
  PrioSwiVal        : FB_BA_PrioSwi_B04;
  Inv               : FB_BA_Chrct02;
END_VAR
```

Inputs

Name	Type	Description
fSpTSuA	REAL	Setpoint of the supply air temperature
fTSuA	REAL	Measured value of the supply air temperature
bSyncTSuACtrl	BOOL	Pulse for synchronization of the sequence controller *TSuACtrl*.
fSyncTSuACtrl	REAL	Value for the synchronization of the supply air sequence controller *TSuACtrl*.
bDehumf	BOOL	Input dehumidification mode active. This state has an effect on the sequence control in the supply air temperature control *TSuACtrl* template.
fTExhA	REAL	Measured value exhaust air temperature
fTExtA	REAL	Measured value extract air temperature
fTWth	REAL	Measured value weather temperature

Outputs

Name	Type	Description
fPresentValue	REAL	Current value of the bypass damper system
eAction	E_BA_Action	The output of the control direction of the supply air controller *TSuACtrl* is required within an air conditioning system for the setpoint strategy.
bActiveSeqCtrl	BOOL	The sequence controller *TSuACtrl* is the active one in the sequence control.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 746]	The function block represents the supply air temperature control of energy recovery and is part of the temperature sequence control of an air conditioning system. The control signal is forwarded to the analog control of damper *Dmp* via a minimum selection.
TExhACtrl	FB_BA_PID_Sp [▶ 1035]	The function block represents the exhaust air temperature control for the anti-icing of the plate heat exchanger.
DiffPrssSwi	FB_BA_SensorBinary [▶ 1088]	The binary input object represents a differential pressure monitor above the heat exchanger. This is used for anti-icing and activates the ramp function *IcePrtcRamp*.
IcePrtcRamp	FB_BA_RampLimitIncrease [▶ 1040]	The function block represents the anti-icing of the plate heat exchanger using an increasing ramp function. This function is activated by means of the differential pressure monitor *DiffPrssSwi*.
Dmp	FB_BA_ActuatorAnalog [▶ 982]	The function block is used to control the analog damper of the plate heat exchanger.
ByDmp	FB_BA_ActuatorAnalog [▶ 982]	The function block is used to control the analog bypass damper of the plate heat exchanger.
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the template. These relevant faults cause specific switching actions via the variables *bPrioSfty* and *bPrioCrit* in the template. The parameterization of the lock priority of the event-enabled objects can be found in the *FB_init* of this template.

Variables

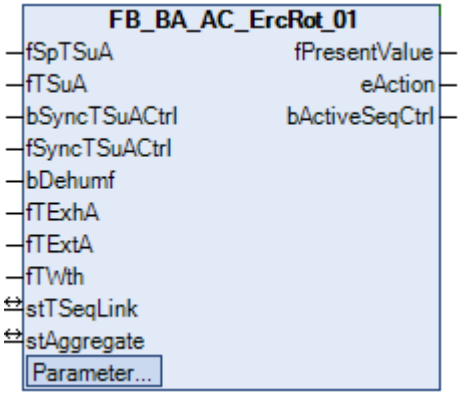
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the Safety lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
bPrioCrit	BOOL	The variable is an evaluation of the Critical lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch *PrioSwiVal* determines the switching conditions and enables for the supply air temperature control *TSuACtrl* and the differential pressure control for anti-icing *IcePrtcDiffPrssCtrl* using the aggregate function block *Aggregate* and the global variable list Priority [▶ 1119].

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.3.3 Rotation

6.1.4.2.1.1.2.3.3.1 FB_BA_AC_ErcRot_01



The template represents the open-loop and closed-loop control of an energy recovery system with a rotary heat exchanger.

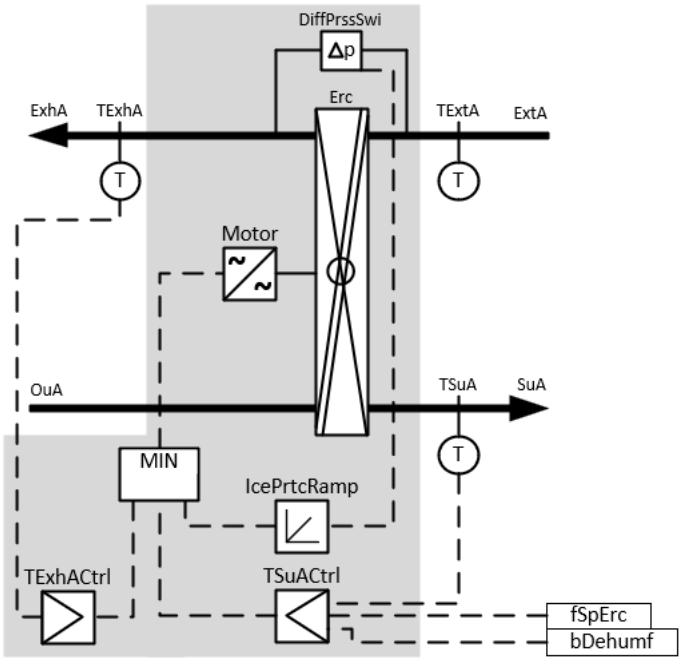
The main tasks of the template are:

- Control of the supply air temperature
- Minimum limitation of exhaust air temperature
- Anti-icing of the heat exchanger by means of a differential pressure monitor
- Control of the rotary heat exchanger motor
- Collecting and evaluating safety-relevant faults using the *PlantLock*
- To be part of the step sequence control of an air conditioning system, see *Aggregates*

The exhaust air minimum limiter *TExhACtrl* and the frost protection program *IcePrtcRamp* limit the control value of the temperature sequence controller *TSuACtrl* of the energy recovery system via a minimum selection to prevent the heat exchanger from icing up.

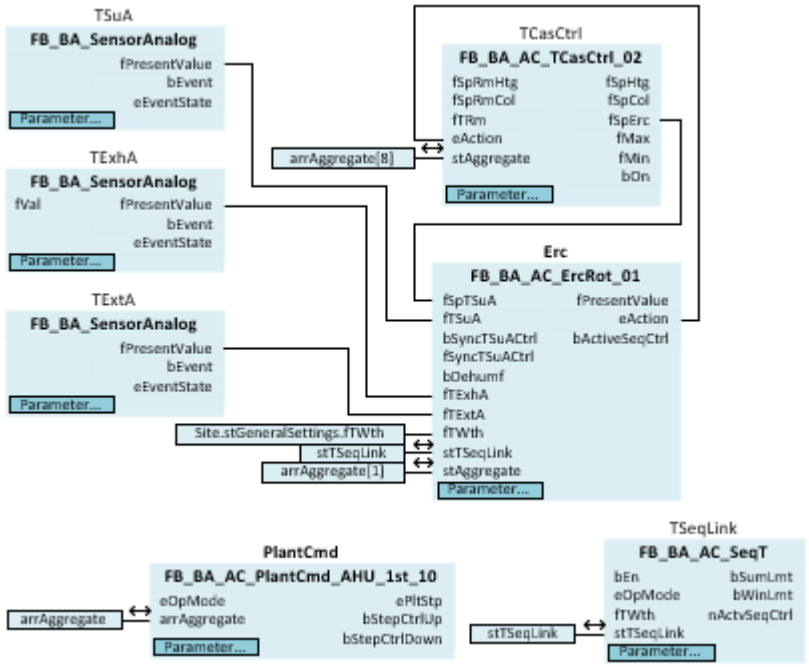
Principle diagram 01

The diagram shows the intended use of the template with the plant elements involved.



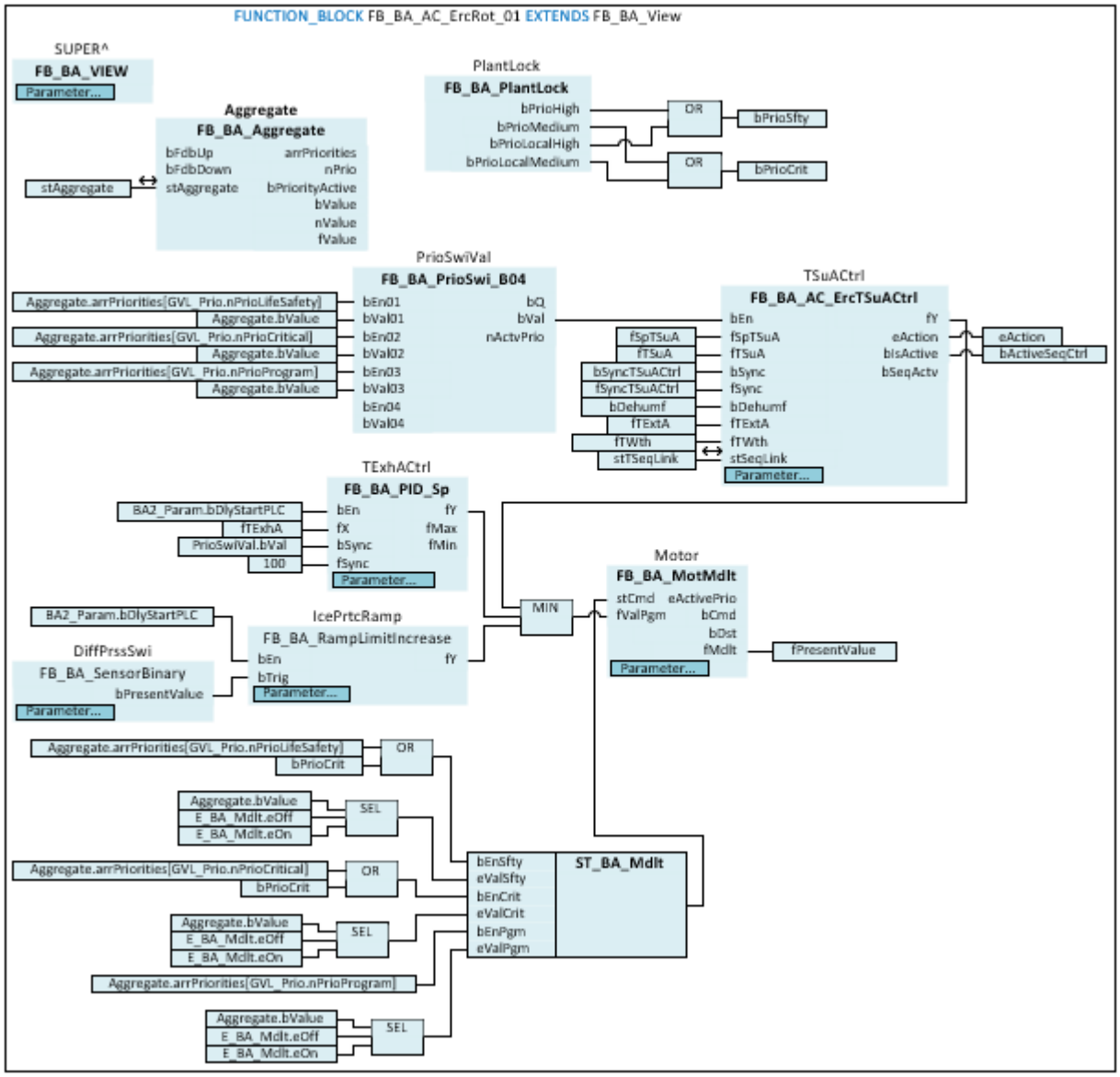
Principle diagram 02

The diagram shows the integration of the template within a plant.



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcRot_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA           : REAL;
  fTSuA             : REAL;
  bSyncTSuAActrl    : BOOL;
  fSyncTSuAActrl    : REAL;
  bDehumf           : BOOL;
  fTExhA            : REAL;
  fTExtA            : REAL;
  fTWth             : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue     : REAL;
  eAction           : E_BA_Action;
  bActiveSeqCtrl    : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink        : ST_BA_SeqLink;
  stAggregate       : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  TSuACtrl          : FB_BA_AC_PreHtrTSuACtrl;
  TExhACtrl         : FB_BA_PID_Sp;
  DiffPrssSwi       : FB_BA_SensorBinary;
  IcePrtcRamp       : FB_BA_RampLimitIncrease;
  Motor             : FB_BA_MotMdlt;
  PlantLock         : FB_BA_PlantLock;
```

```
  Aggregate          : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty          : BOOL;
  bPrioCrit          : BOOL;
  PrioSwiVal         : FB_BA_PrioSwi_B04;
END_VAR
```

Inputs

Name	Type	Description
fSpTSuA	REAL	Setpoint of the supply air temperature
fTSuA	REAL	Measured value of the supply air temperature
bSyncTSuACtrl	BOOL	Pulse for synchronization of the sequence controller *TSuACtrl*.
fSyncTSuACtrl	REAL	Value for the synchronization of the supply air sequence controller *TSuACtrl*.
bDehumf	BOOL	Input dehumidification mode active. This state has an effect on the sequence control in the supply air temperature control *TSuACtrl* template.
fTExhA	REAL	Measured value exhaust air temperature
fTExtA	REAL	Measured value extract air temperature
fTWth	REAL	Measured value weather temperature

Outputs

Name	Type	Description
fPresentValue	REAL	Current value of the bypass damper system
eAction	E_BA_Action	The output of the control direction of the supply air controller *TSuACtrl* is required within an air conditioning system for the setpoint strategy.
bActiveSeqCtrl	BOOL	The sequence controller *TSuACtrl* is the active one in the sequence control.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 746]	The function block represents the supply air temperature control of energy recovery and is part of the temperature sequence control of an air conditioning system. The control signal is forwarded to the analog control of damper *Dmp* via a minimum selection.
TExhACtrl	FB_BA_PID_Sp [▶ 1035]	The function block represents the exhaust air temperature control for the anti-icing of the plate heat exchanger.
DiffPrssSwi	FB_BA_SensorBinary [▶ 1088]	The binary input object represents a differential pressure monitor above the heat exchanger. This is used for anti-icing and activates the ramp function *IcePrtcRamp*.
IcePrtcRamp	FB_BA_RampLimitIncrease [▶ 1040]	The function block represents the anti-icing of the rotary heat exchanger using an increasing ramp function. This function is activated by means of the differential pressure monitor *DiffPrssSwi*.
Motor	FB_BA_MotMdlt [▶ 1070]	The function block is used to control a frequency converter. This then controls the motor of the rotary heat exchanger.
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the template. These relevant faults cause specific switching actions via the variables *bPrioSfty* and *bPrioCrit* in the template. The parameterization of the lock priority of the event-enabled objects can be found in the *FB_init* of this template.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

Variables

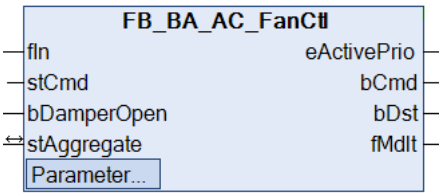
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the Safety lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
bPrioCrit	BOOL	The variable is an evaluation of the Critical lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch *PrioSwiVal* determines the switching conditions and enables for the supply air temperature control *TSuACtrl* and the differential pressure control for anti-icing *IcePrtcDiffPrssCtrl* using the aggregate function block *Aggregate* and the global variable list Priority [▶ 1119].

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 Fan

6.1.4.2.1.1.2.4.1 FB_BA_AC_FanCtl



The aggregate template represents the control and regulation of a pressure-controlled fan. Typical applications for this template are the supply and exhaust air fans of an air conditioning system with variable volume flow rate.

The basic template FB_BA_MotCtl [▶ 1066] is extended by two binary inputs for detecting an additional fault and a maintenance switch.

The input *bDamperOpen* prevents the fan from starting when the damper is closed.

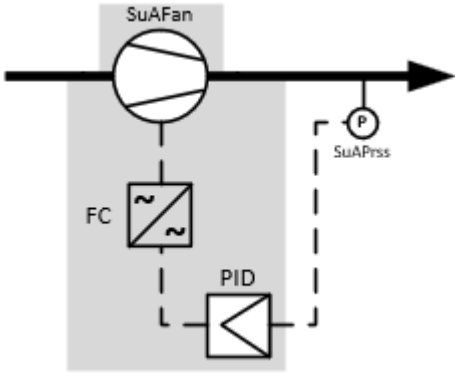
The pressure-controlled fan is integrated into the control program of the air conditioning system using the function block *Aggregate* and the inputs and outputs structure stAggregate [▶ 278].

Each template is available in 2 different configurations 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.

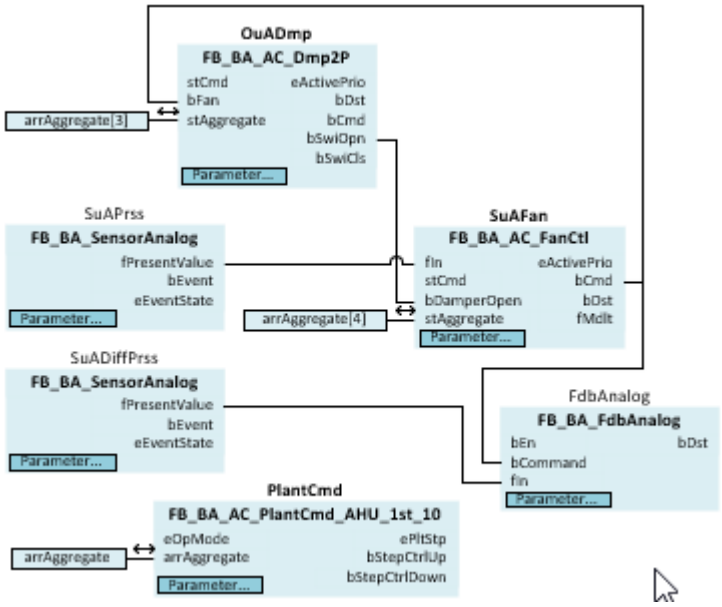
Principle diagram 01

The diagram shows the intended use of the template with the plant elements involved.

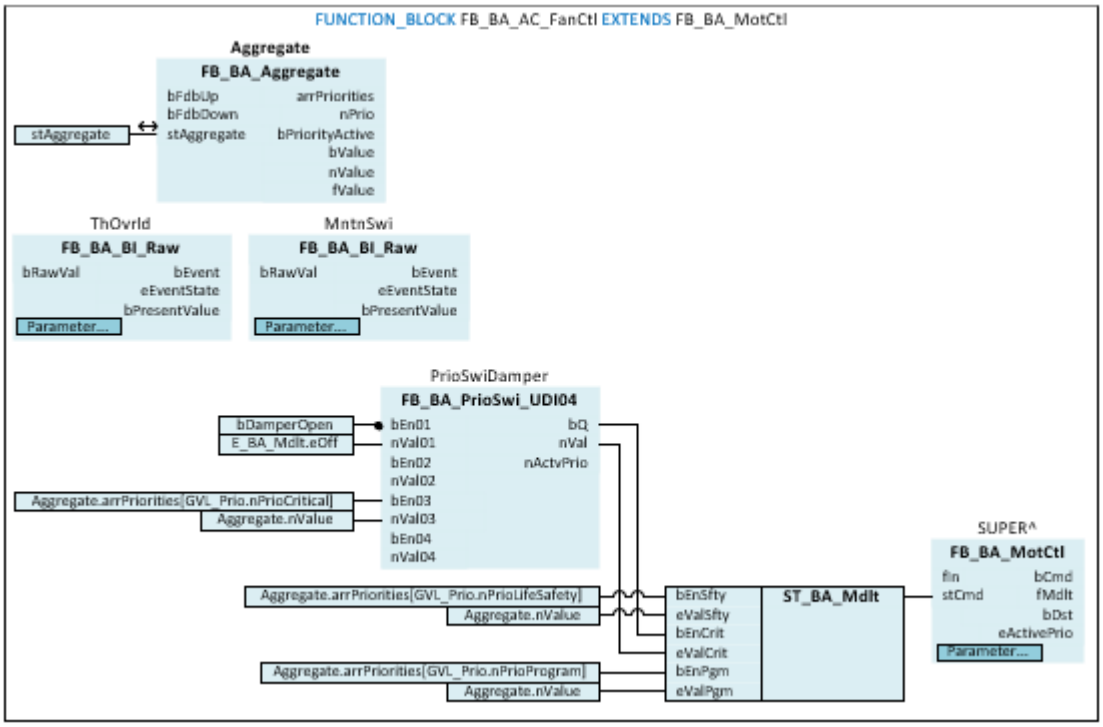


Principle diagram 02

The diagram shows the integration of the template within a plant.



Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_FanCtl EXTENDS FB_BA_MotCtl
VAR_INPUT
  bDamperOpen       : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate       : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  ThOvrld           : FB_BA_BI_Raw;
  MntnSwi           : FB_BA_BI_Raw;
  Aggregate         : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiDamper     : FB_BA_PrioSwi_UDI04;
END_VAR
```

Inputs

Name	Type	Description
bDamperOpen	BOOL	The input signal "Damper open" is used within a air conditioning system to protect the connected damper. As long as the input signal "Damper open" is not TRUE, the priority "Critical" is enabled on the priority switch *PrioSwiDamper* within the template and the fan is switched off.

Inputs/outputs

Name	Type	Description
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control Aggregate. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
ThOvrld	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a maintenance switch.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

Variables

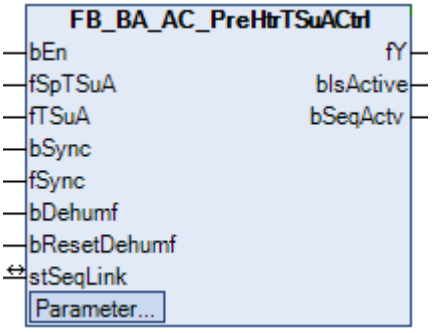
Name	Type	Description
PrioSwiDamper	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch *PrioSwiDamper* uses the aggregate function block *Aggregate* and the input variable *bDamperOpen* to determine the enable conditions and switch-on values for the priority "Critical" of the fan.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.2.5 PreHeater

6.1.4.2.1.1.2.5.1 FB_BA_AC_PreHtrTSuACtrl



The template represents the supply air temperature control of a preheater in a ventilation system. The supply air controller *Ctrl* is the core of the template. It is integrated into a supply air temperature control sequence.

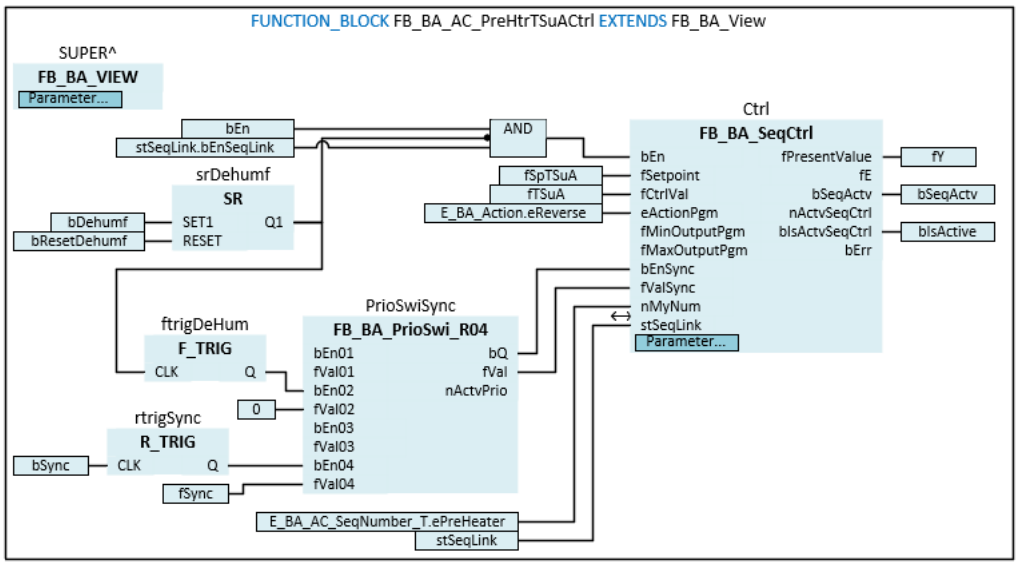
The sequence controller is enabled using the input variable *bEn* and the variable *stSeqLink.bEnSeqLink* of the data and command structure *stSeqLink*.

In dehumidification mode *bDehumf*, the sequence controller *Ctrl* is disabled and thus removed from the sequence control.

The sequence controllers are enabled for sequence control via the data and command structure *stSeqLink*. This is indicated by the variable *bSeqActv*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  fSpTSuA               : REAL;
  fTSuA                 : REAL;
  bSync                 : BOOL;
  fSync                 : REAL;
  bDehumf               : BOOL;
  bResetDehumf          : BOOL;
END_VAR
VAR_OUTPUT
```

```
  fY                    : REAL;
  bIsActive             : BOOL;
  bSeqActv              : BOOL;
END_VAR
VAR_IN_OUT
  stSeqLink             : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  Ctrl                  : FB_BA_SeqCtrl;
END_VAR
VAR
  srDehumf              : SR;
  ftrigDeHum            : F_TRIG;
  rtrigSync             : R_TRIG;
  PrioSwiSync           : FB_BA_PrioSwi_R04;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enable the sequence controller *Ctrl*.
fSpTSuA	REAL	Setpoint of the supply air temperature.
fTSuA	REAL	Measured value of the supply air temperature.
bSync	BOOL	Pulse for synchronization of the sequence controller *Ctrl*.
fSync	REAL	Value for the synchronization of the sequence controller *Ctrl*.
bDehumf	BOOL	Input to set the dehumidification mode on the RS flip-flop *srDehumf*.
bResetDehumf	BOOL	Input to reset the dehumidification mode on the RS flip-flop *srDehumf*.

Outputs

Name	Type	Description
fY	REAL	Control value output control valve.
bIsActive	BOOL	The sequence controller is the active one in the sequence control.
bSeqActv	BOOL	The sequence controller is implemented in the control sequence.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure *stSeqLink* is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence control TSeqLink [▶ 778] of an air conditioning system.

Inputs CONSTANT

Name	Type	Description
Ctrl	FB_BA_SeqCtrl	The supply air temperature sequence controller *Ctrl* is the core of this template. It is responsible for the supply air temperature control of the preheater. The sequence controller is also part of the supply air temperature sequence control of an air conditioning system. The data exchange within this sequence control takes place via the data and command structure *stSeqLink*. The global variable E_BA_AC_SeqNumber_T.ePreHeater [▶ 697] gives the sequence controller its sequence number within the temperature sequence control.

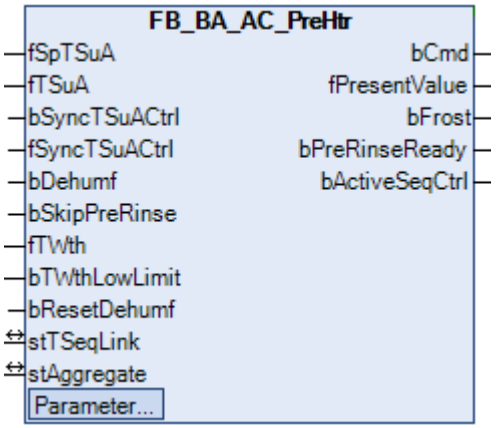
Variables

Name	Type	Description
srDehumf	SR	Setting the RS flip-flop *srDehumf* indicates that dehumidification mode is active within the air conditioning system. The sequence controller *Ctrl* is no longer enabled. The RS flip-flop *srDehumf* is reset via the input *bResetDehumf*.
ftrigDeHum	F_TRIG	A falling edge at input *CLK* of the function block *ftrigDeHum* synchronizes the sequence controller *Ctrl* to the value 0. The falling edge is triggered by resetting the dehumidification mode on the RS flip-flop *srDehumf*.
rtrigSync	R_TRIG	On a rising edge at input *CLK* of the function block *rtrigSync*, the sequence controller *Ctrl* is synchronized to the value of *fSync*.
PrioSwiSync	FB_BA_PrioSwi_R04 [▶ 433]	The priority switch prioritizes the synchronization of the sequence controller *Ctrl*

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.5.2 FB_BA_AC_PreHtr



The template represents the open-loop and closed-loop control of a hot water air heater.

The main tasks of the template are:

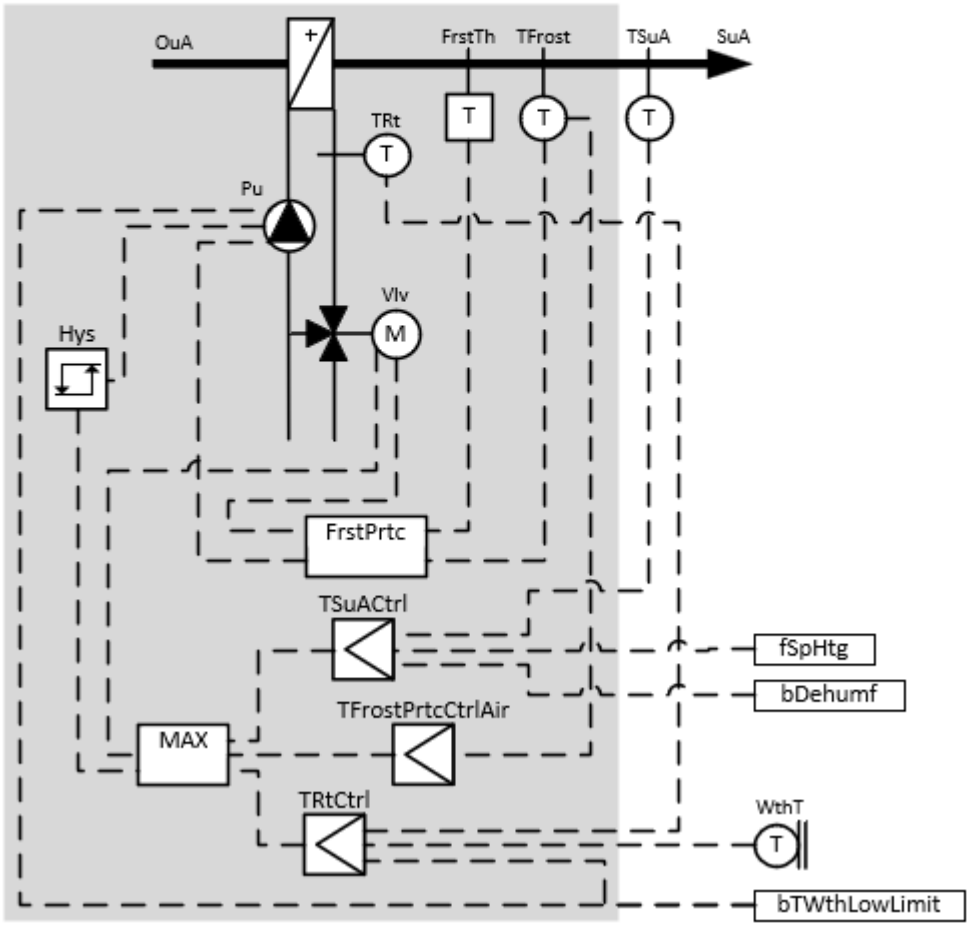
- Control of the supply air temperature (*TSuACtrl*).
- Control of the return temperature (*TRtCtrl*).
- Frost monitoring on the air side with frost protection thermostat (*FrostThermostat*).
- Frost monitoring air side with analog frost protection sensor (*TFrost*).
- Enable the heater pump (*Pu*).
- Control of the heater valve (*Vlv*).
- Collecting and evaluating safety-relevant faults using the *PlantLock*
- To be part of the step sequence control of an air conditioning system, see *Aggregates*

A maximum choice of control signals from the temperature sequence controller *TSuACtrl* and the frost protection functions *TFrostPrtcCtrlAir* / *TRtCtrl* prevent the hot water air heater from icing up.

The initialization of the template takes place within the method FB_Init.

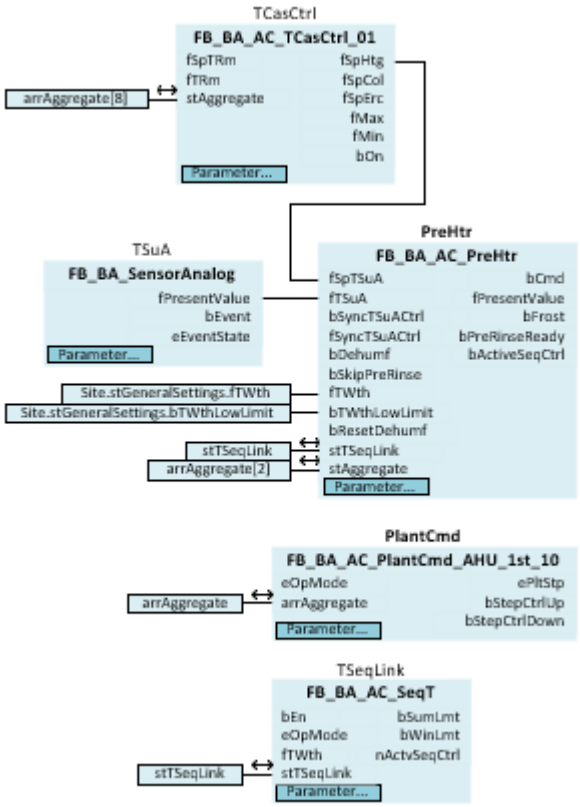
Principle diagram 01

The principle diagram shows the intended use of the template with the plant elements involved.

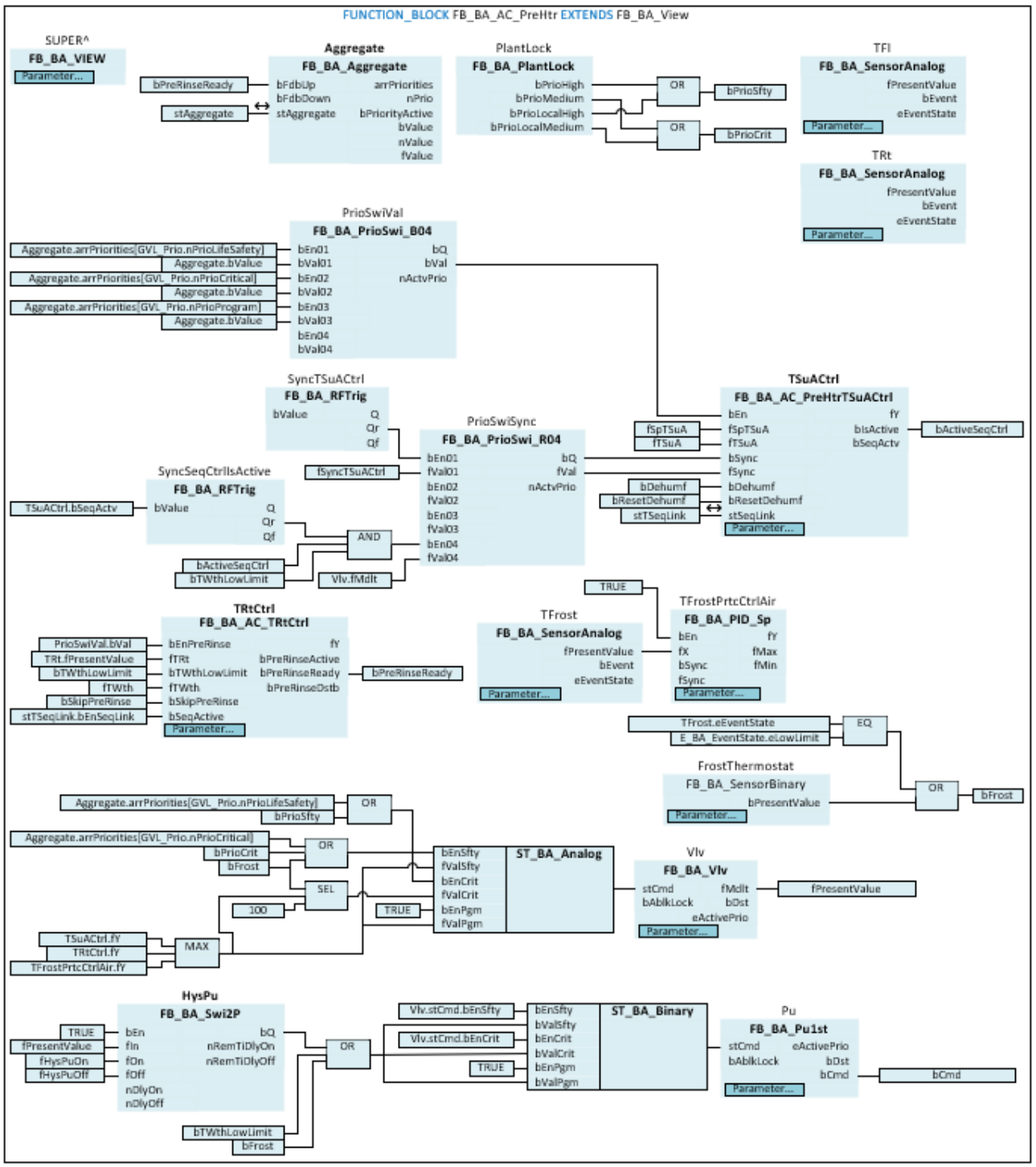


Principle diagram 02

The diagram shows the integration of the template within a plant.


TCasCtrl
FB_BA_AC_TCasCtrl_01
fSpTRm
fTRm
stAggregate
fSpHtg
fSpCol
fSpErc
fMax
fMin
bOn
arrAggregate[8]
Parameter...
TSuA
FB_BA_SensorAnalog
fPresentValue
bEvent
eEventState
Parameter...
PreHtr
FB_BA_AC_PreHtr
fSpTSuA
fTSuA
bSyncTSuACtrl
fSyncTSuACtrl
bDehumf
bSkipPreRinse
fTWth
bTWthLowLimit
bResetDehumf
stTSeqLink
stAggregate
bCmd
fPresentValue
bFrost
bPreRinseReady
bActiveSeqCtrl
Parameter...
Site.stGeneralSettings.fTWth
Site.stGeneralSettings.bTWthLowLimit
stTSeqLink
arrAggregate[2]
PlantCmd
FB_BA_AC_PlantCmd_AHU_1st_10
eOpMode
arrAggregate
ePltStp
bStepCtrlUp
bStepCtrlDown
Parameter...
arrAggregate
TSeqLink
FB_BA_AC_SeqT
bEn
eOpMode
fTWth
stTSeqLink
bSumLmt
bWinLmt
nActvSeqCtrl
Parameter...
stTSeqLink

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_PreHtr EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA                 : REAL;
  fTSuA                   : REAL;
  bSyncTSuACtrl           : BOOL;
  fSyncTSuACtrl           : REAL;
  bDehumf                 : BOOL;
  bSkipPreRinse           : BOOL;
  fTWth                   : REAL;
  bTWthLowLimit           : BOOL;
  bResetDehumf            : BOOL;
END_VAR
VAR_OUTPUT
  bCmd                    : BOOL;
  fPresentValue           : REAL;
```

```
  bFrost                  : BOOL;
  bPreRinseReady          : BOOL;
  bActiveSeqCtrl          : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink              : ST_BA_SeqLink;
  stAggregate             : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOn                : REAL := 5.0;
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOff               : REAL := 1.0;
END_VAR
VAR_INPUT CONSTANT
  TFl                     : FB_BA_SensorAnalog;
  TRt                     : FB_BA_SensorAnalog;
  TSuACtrl                : FB_BA_AC_PreHtrTSuACtrl;
  TRtCtrl                 : FB_BA_AC_TRtCtrl;
  TFrost                  : FB_BA_SensorAnalog;
  TFrostPrtcCtrlAir       : FB_BA_PID_Sp;
  FrostThermostat         : FB_BA_SensorBinary;
  Vlv                     : FB_BA_Vlv;
  Pu                      : FB_BA_Pulst;
  PlantLock               : FB_BA_PlantLock;
  Aggregate               : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty               : BOOL;
  bPrioCrit               : BOOL;
  PrioSwiVal              : FB_BA_PrioSwi_B04;

  HysPu                   : FB_BA_Swi2P;
  SyncTSuACtrl            : FB_BA_RFTrig;
  SyncSeqCtrlIsActive     : FB_BA_RFTrig;
  PrioSwiSync             : FB_BA_PrioSwi_R04;
END_VAR
```

Inputs

Name	Type	Description
fSpTSuA	REAL	Setpoint of the supply air temperature
fTSuA	REAL	Measured value of the supply air temperature
bSyncTSuACtrl	BOOL	Input for synchronization of the supply air sequence controller in the function block *TSuACtrl*.
fSyncTSuACtrl	REAL	Synchronization value for the supply air sequence controller in the function block *TSuACtrl*.
bDehumf	BOOL	Input dehumidification mode active. This state has an effect on the sequence control in the supply air temperature control *TSuACtrl* template.
bSkipPreRinse	BOOL	Input to skip the pre-rinse process in the function block *TRtCtrl*.
fTWth	REAL	Measured value weather temperature
bTWthLowLimit	BOOL	The variable indicates that the outside temperature has fallen below the lower limit value. The following actions are triggered when the outside temperature falls below a critical value: • Heater pump activation, forced pump operation • Enabling the return temperature control in the function block *TRtCtrl* • During plant start-up, pre-rinse mode is active in the function block *TRtCtrl*
bResetDehumf	BOOL	Input to reset the dehumidification mode in the template *TSuACtrl*.

Outputs

Name	Type	Description
bCmd	BOOL	Current switching status of the single-stage pump.
fPresentValue	REAL	Current control value of the heater valve.
bFrost	BOOL	Frost protection active display. The message is triggered either by the frost protection thermostat *FrostThermostat* or by falling below the lower limit value of the analog frost protection sensor *TFrost*. The signal Frost protection active triggers the plant operation mode Frost protection in the template FB_BA_AC_OpMod1St_Prio [▶ 763].
bPreRinseReady	BOOL	Display, pre-rinse process has reached set temperature. This signal is used as a switching condition for the step sequence control of a ventilation system.
bActiveSeqCtrl	BOOL	Indicates that the sequence controller *TSuACtrl* is the active one in the sequence control.

Inputs/outputs

Name	Type	Description
stSeqLink	ST_BA_SeqLink [▶ 123]	The data and command structure is the link between the supply air temperature sequence controller *Ctrl* and the supply air temperature sequence controller *TSeqLink* of an air conditioning system.
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT PERSISTENT

Name	Type	Description
fHysPuOn	REAL	Upper switching point of the hysteresis to switch on the pump.
fHysPuOff	REAL	Lower switching point of the hysteresis to switch off the pump.

Inputs CONSTANT

Name	Type	Description
TFl	FB_BA_SensorAnalog [▶ 1087]	The function block represents the flow temperature sensor.
TRt	FB_BA_SensorAnalog [▶ 1087]	The function block represents the return temperature sensor.
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 746]	The function block represents the supply air temperature control of a preheater and is part of the temperature sequence control of an air conditioning system. The control signal is forwarded to the heater valve *Vlv* via a maximum selection.
TRtCtrl	FB_BA_AC_TRtCtrl [▶ 756]	The function block represents the open-loop and closed-loop control of the return temperature of a hot water air heater. The control signal is forwarded to the heater valve *Vlv* via a maximum selection.
TFrost	FB_BA_SensorAnalog [▶ 1087]	The function block represents a frost protection sensor on the air side.
TFrostPrtcCtrlAir	FB_BA_PID_Sp [▶ 1035]	The function block represents the analog frost protection monitoring on the air side of the preheater. The control signal is forwarded to the heater valve *Vlv* via a maximum selection.
FrostThermostat	FB_BA_SensorBinary [▶ 1088]	The function block represents a frost protection thermostat.
Vlv	FB_BA_Vlv [▶ 1092]	The function block represents the heater valve.
Pu	FB_BA_Pu1st [▶ 1074]	The function block represents the heater pump.
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the template. These relevant faults cause specific switching actions via the variables *bPrioSfty* and *bPrioCrit* in the template. The event-enabled objects of the template that trigger faults with *bPrioCrit* are listed below. *TRt.MV, TFrost.MV, FrostThermostat.Input, Pu.Dst* The parameterization of the lock priority of the event-enabled objects can be found in the FB_init of this template.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

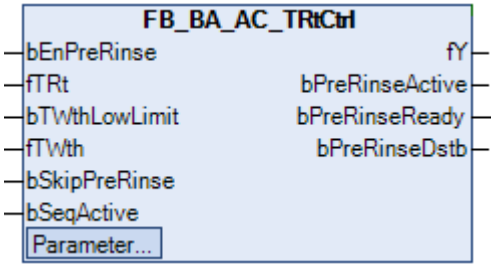
Variables

Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the event-enabled objects in the project structure and causes targeted switching actions in the template when a relevant fault is triggered.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch uses the command structure *stCmd* to determine the enable conditions for the supply air temperature control *TSuACtrl* and for the pre-rinse process of the return air temperature control *TRtCtrl*.
HysPu	FB_BA_Swi2P	The two-position switch switches the heater pump on and off depending on the valve position *fPresentValue* and the switching point of the hysteresis *fHysPuOn*/*fHysPuOff*.
SyncTSuACtrl	FB_BA_RFTrig	A rising edge at the input *bValue* of the function block synchronizes the supply air temperature control *TSuACtrl* to the value of *fSyncTSuACtrl*.
SyncSeqCtrlIsActive	FB_BA_RFTrig	The variable *TSuACtrl.bSeqActv* indicates that the sequence control is active. This rising edge triggers a pulse at the input *bValue* of the function block *SyncSeqCtrlIsActive*. If the sequence controller of the preheater is the active one in the sequence control *bActiveSeqCtrl* and the lower limit of the outside temperature *bTWthLowLimit* is active, the pulse of *SyncSeqCtrlIsActive* triggers a synchronization of the supply air temperature controller *TSuACtrl* to the value of the valve position *Vlv.fMdlt*.
PrioSwiSync	FB_BA_PrioSwi_R04 [▶ 433]	The priority switch prioritizes the synchronization of the supply air temperature control *TSuACtrl*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.2.5.3 FB_BA_AC_TRtCtrl



The template is used for open-loop and closed-loop control of the return temperature of a hot water air heater.

To prevent frost damage, the heating coil of the ventilation system is first preheated with warm water when the outside temperature is low.

To do this, the flip-flop *rsPreRinseActive* is first set in the template *FB_BA_AC_TRtCtrl*. The return temperature controller *Ctrl* receives the pre-rinse setpoint from the setpoint curve *PreRinseSp* via the selector. To prevent the heating coil from overheating during preheating, the pre-rinse setpoint is varied by

the characteristic curve *PreRinseSp* depending on the outside temperature. When the desired return temperature is reached, the hysteresis module *PreRinseHys* switches on. The flip-flop *rsPreRinseReady* is then set.

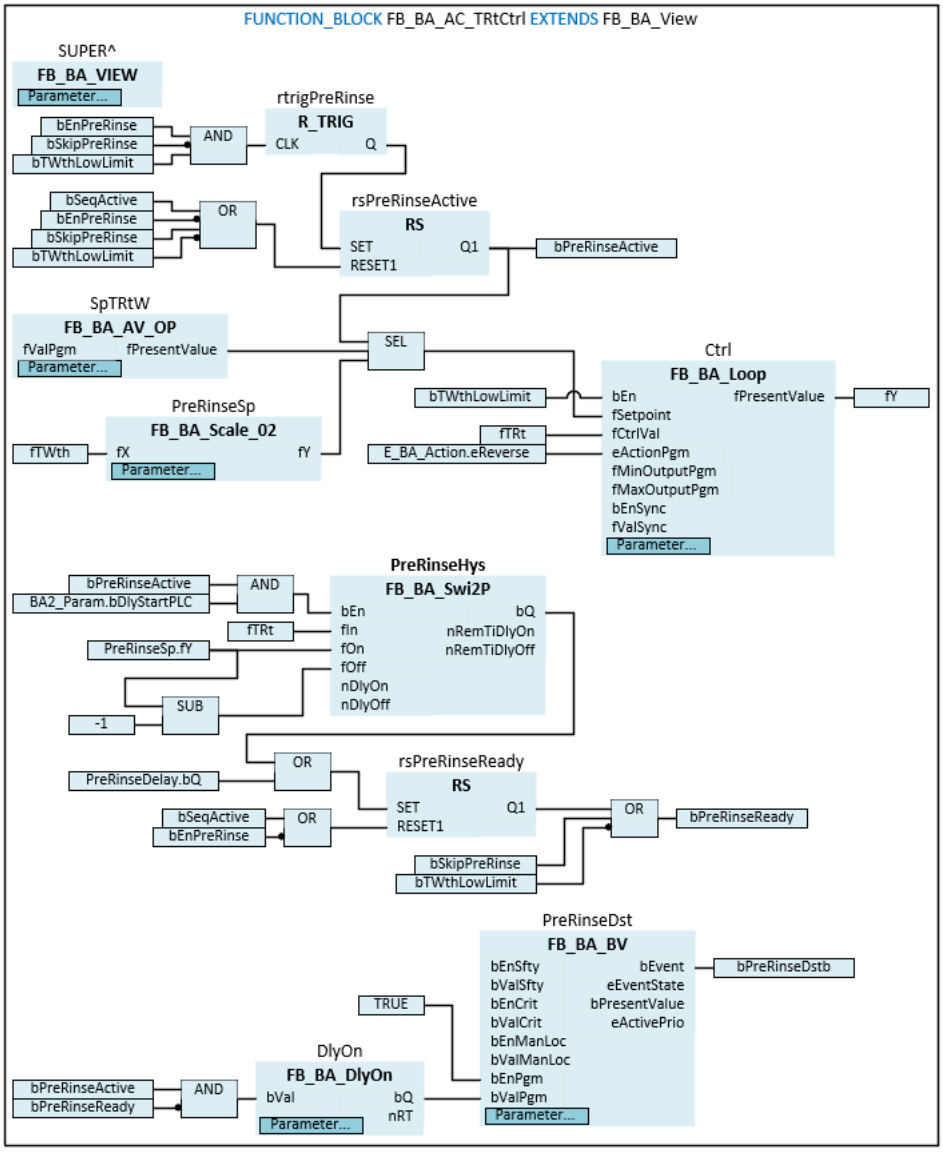
The system start program FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 786] is informed via the output *bPreRinseReady* that it can continue with the next step, e.g. Opening the outdoor air damper.

The flip-flop rsPreRinseActive remains set so that the pre-rinse setpoint is still present at the return temperature controller.

If the system start program has switched on the fans and enabled the supply air temperature control in its subsequent steps, the flip-flop *rsPreRinseActive* is reset using the input *bSeqActive*. The pre-rinse process is now complete. For permanent frost protection operation of the heating coil during system operation and system standstill, the return temperature controller receives the reduced setpoint of the AV object *SpTRtW*. The output *bQ* of the function block *PreRinseDelay* becomes TRUE if the pre-rinse process is started and the pre-rinse setpoint is not reached after a delay time has elapsed. The fault is reported using the BV object *PreRinseDst*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEnPreRinse               : BOOL;
  fTRt                      : REAL;
  bTWthLowLimit             : BOOL;
  fTWth                     : REAL;
  bSkipPreRinse             : BOOL;
  bSeqActive                : BOOL;
END_VAR
VAR_OUTPUT
  fY                        : REAL;
  bPreRinseActive           : BOOL;
  bPreRinseReady            : BOOL;
  bPreRinseDstb             : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SpTRtW                    : FB_BA_AV_Op;
  PreRinseDelay             : FB_BA_DlyOn;
  PreRinseDst               : FB_BA_BV;
  PreRinseSp                : FB_BA_Scale_02;
  Ctrl                      : FB_BA_Loop;
END_VAR
VAR
  rtrigPreRinse             : R_TRIG;
  rsPreRinseReady           : RS;
  rsPreRinseActive          : RS;
  PreRinseHys               : FB_BA_Swi2P;
END_VAR
```

Inputs

Name	Type	Description
bEnPreRinse	BOOL	Enabling the pre-rinse process.
fTRt	REAL	Measured value of the return temperature of the hot water air heater.
bTWthLowLimit	BOOL	The variable indicates that the outside temperature has fallen below the lower limit value. This state is used to enable the return temperature controller *Ctrl*.
fTWth	REAL	Current value of the outside temperature.
bSkipPreRinse	BOOL	Input for skipping the pre-rinse process.
bSeqActive	BOOL	This input is used to inform the controller that the supply air temperature control of the ventilation has gone into operation or that the supply air temperature sequence controller has been enabled.

Outputs

Name	Type	Description
fY	REAL	Control value output of the return temperature controller *Ctrl*.
bPreRinseActive	BOOL	Display, pre-rinse process active.
bPreRinseReady	BOOL	Display, pre-rinse process has reached set temperature. This signal is used as a switching condition for the step sequence control of a ventilation system, see sample FB_BA_AC_PlantCmd_AHU_1st_10.
bPreRinseDstb	BOOL	Display, pre-rinse process faulty.

Inputs CONSTANT

Name	Type	Description
SpTRtW	FB_BA_AV_Op [▶ 202]	The analog value object is used to enter the return temperature setpoint when there is a risk of frost or low outside temperatures.
PreRinseDelay	FB_BA_DlyOn [▶ 1054]	The template represents a start-up delay and triggers the message "Pre-rinse process faulty" at the object *PreRinseDst* if the result of the pre-rinse hysteresis *PreRinseHys* is not achieved.
PreRinseDst	FB_BA_BV [▶ 213]	The binary object displays the message "Pre-rinse process faulty".
PreRinseSp	FB_BA_Scale_02 [▶ 1042]	Template for calculating the pre-rinse setpoint as a function of the outside temperature *fTWth* for return temperature control during system start-up.
Ctrl	FB_BA_Loop [▶ 220]	PID controller for controlling the return temperature of the heating coil.

Variables

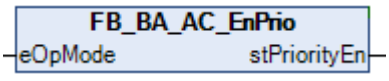
Name	Type	Description
rtrigPreRinse	R_TRIG	rtrigPreRinse activates the pre-rinse at the RS flip-flop rsPreRinseActive by a rising edge at input CLK.
rsPreRinseReady	RS	Setting the RS flip-flop indicates that the pre-rinse process has reached its set temperature.
rsPreRinseActive	RS	The pre-rinse process is activated by setting the RS flip-flop.
PreRinseHys	FB_BA_Swi2P	The result of the pre-rinse hysteresis *PreRinseHys* indicates that the pre-rinse process has reached its set temperature.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.3 General

6.1.4.2.1.1.3.1 FB_BA_AC_EnPrio



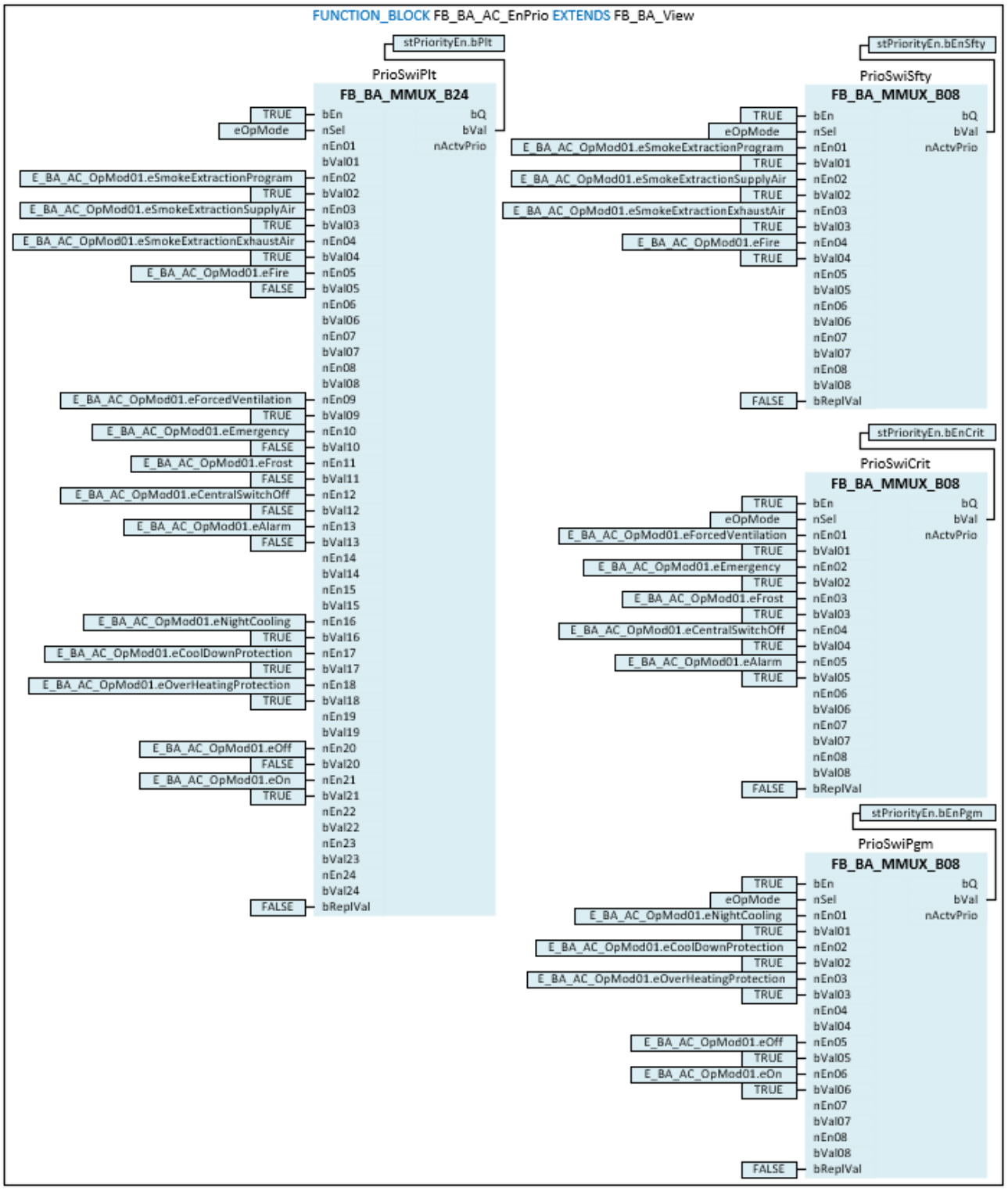
The template represents the system enable and the enable of the priorities Safty, Critcial and Program of an air conditioning system.

The multiplexers *PrioSwiPlt*, *PrioSwiSfty*, *PrioSwiCrit* and *PrioSwiPgm* use the plant operation mode *eOpMode* to define the system enable for activating the step sequence control of a system and enabling the priorities "Safety", "Critcial" and "Program", see command structure *stPriorityEn*.

eOpmode	Value	Operation mode	System status	stPriorityEn.bpLT	stPriorityEn.bEnSfty	stPriorityEn.bEnCrit	stPriorityEn.bPgm
E_BA_AC_OpMod01.eOff	1	Off	Plant shutdown	FALSE	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eOn	2	On	Plant startup	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eEmergency	3	Emergency	Plant shutdown	FALSE	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eFrost	4	Frost	Plant shutdown	FALSE	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eSmokeExtractionProgram	5	Smoke extraction program	Plant startup	**TRUE**	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir	6	Smoke extraction supply air	Plant startup	**TRUE**	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir	7	Smoke extraction exhaust air	Plant startup	**TRUE**	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eFire	8	Fire	Plant shutdown	FALSE	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eNightCooling	9	Night cooling	Plant startup	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eCoolDownProtection	10	Support operation, cooling protection	Plant startup	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eOverHeatingProtection	11	Overheating protection	Plant startup	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eAlarm	12	Fault	Plant shutdown	FALSE	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eForcedVentilation	13	Forced ventilation	Plant startup	**TRUE**	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eCentralSwitchOff	14	Central shutdown	Plant shutdown	FALSE	FALSE	**TRUE**	FALSE

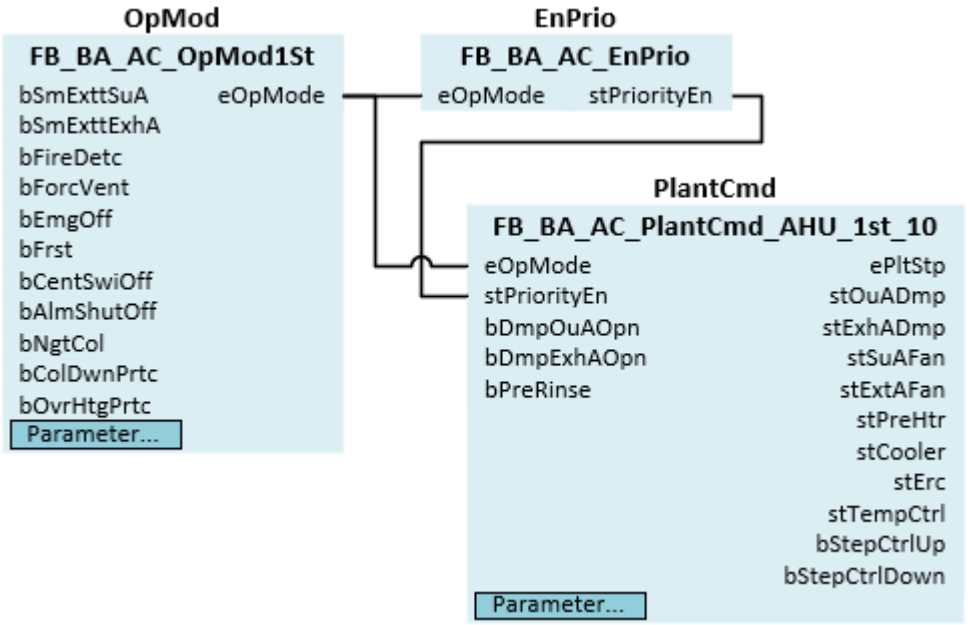
The initialization of the template takes place within the method FB_Init.

Block diagram



Principle diagram

The diagram shows the integration of the template within a plant.



Syntax

```
FUNCTION_BLOCK FB_BA_AC_EnPrio
VAR_INPUT
  eOpMode           : E_BA_AC_OpMod01;
END_VAR
VAR_OUTPUT
  stPriorityEn      : ST_BA_PriorityEn;
END_VAR
VAR
  PrioSwiPlt        : FB_BA_MMUX_B24;
  PrioSwiSfty       : FB_BA_MMUX_B08;
  PrioSwiCrit       : FB_BA_MMUX_B08;
  PrioSwiPgm        : FB_BA_MMUX_B08;
END_VAR
```

Inputs

Name	Type	Description
eOpMode	E_BA_AC_OpMod01 [▶ 699]	Input of the current operation mode *eOpMode*.

Outputs

Name	Type	Description
stPriorityEn	ST_BA_PriorityEn [▶ 278]	Output of the current command structure *stPriorityEn*. This includes the system enable for the step sequence control of an air conditioning system and the associated enables for the priorities "Safety", "Critical" and "Program".

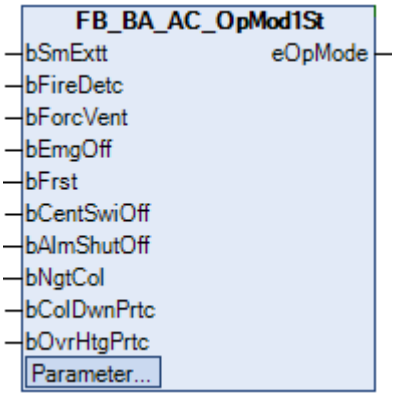
Variables

Name	Type	Description
PrioSwiPlt	FB_BA_MMUX_B24 [▶ 428]	The multiplexer uses the plant operation mode *eOpMode* to define the system enable for controlling the aggregates of a plant.
PrioSwiSfty	FB_BA_MMUX_B08 [▶ 428]	The multiplexer uses the plant operation mode *eOpMode* to define the enable of the priority "Safety" for controlling the aggregates of a plant.
PrioSwiCrit	FB_BA_MMUX_B08 [▶ 428]	The multiplexer uses the plant operation mode *eOpMode* to define the enable of the priority "Critical" for controlling the aggregates of a plant.
PrioSwiPgm	FB_BA_MMUX_B08 [▶ 428]	The multiplexer uses the plant operation mode *eOpMode* to define the enable of the priority "Program" for controlling the aggregates of a plant.

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_AC_OpMod1St

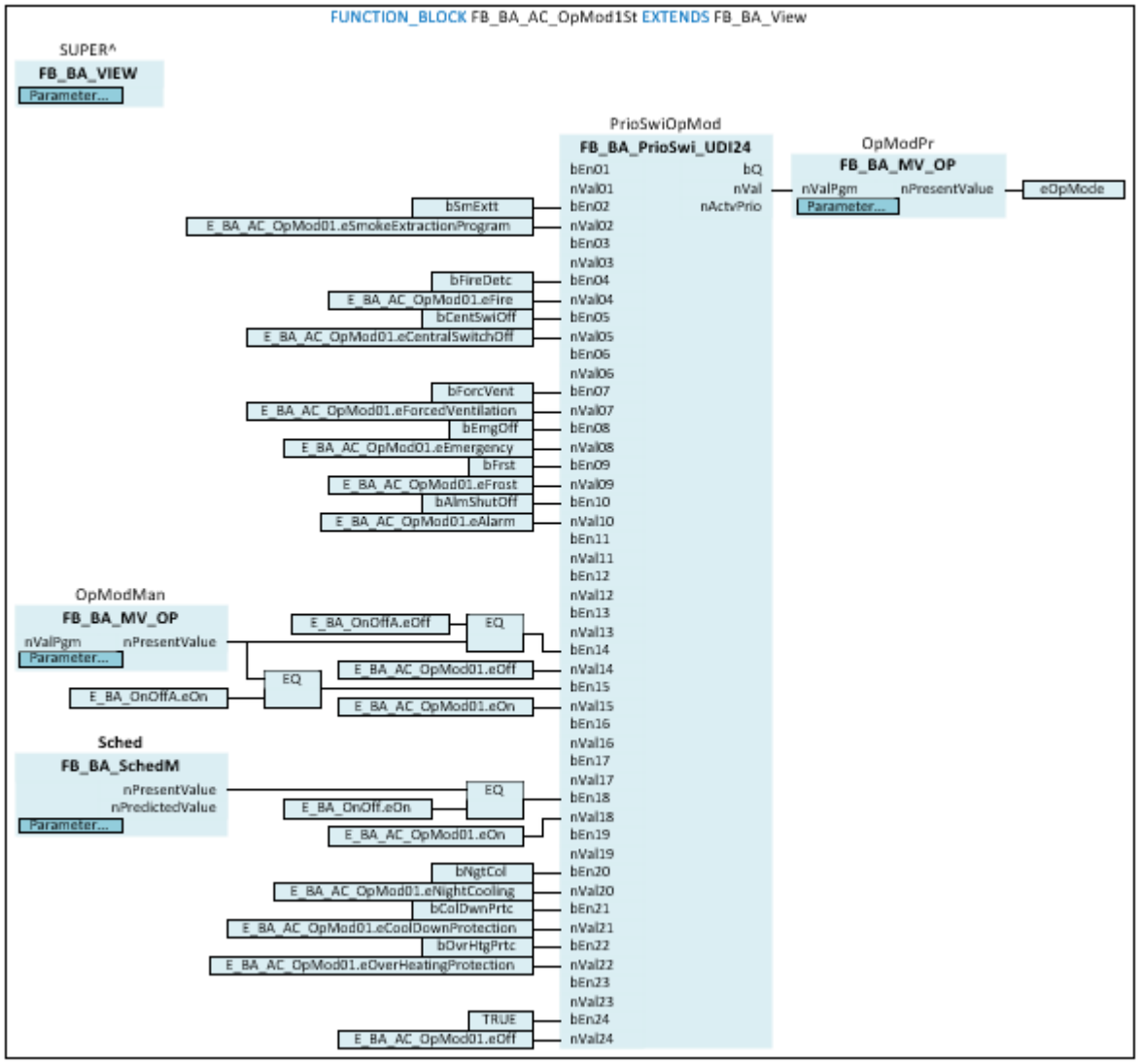


The template represents the operation mode of an air conditioning system.

The priority switch *PrioSwiOpMod* prioritizes various events or commands such as fire alarms, requests from the time schedule or requests from the plant selector switch and writes a resulting operation mode or system status to the variable eOpMode [▶ 699].

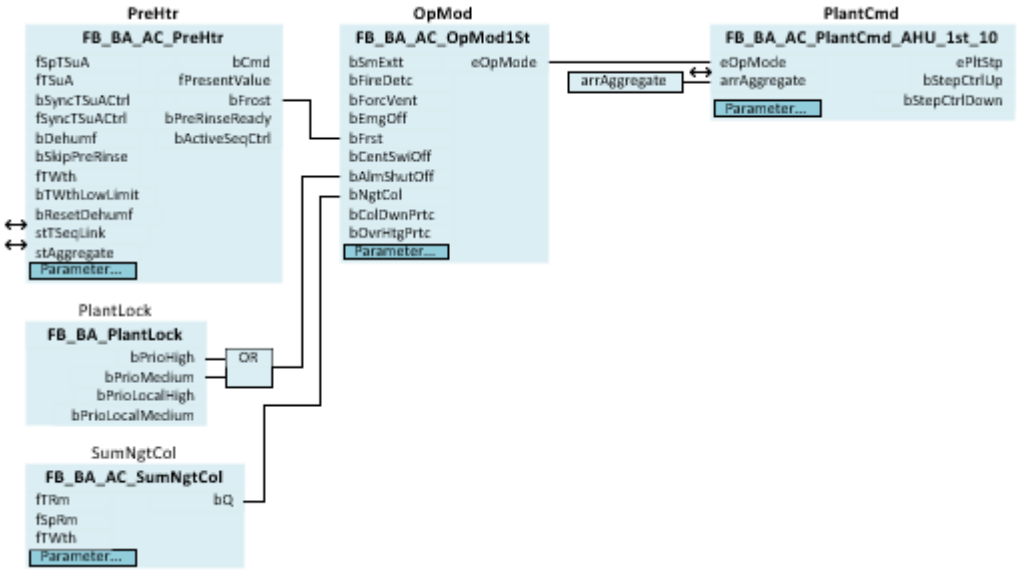
The initialization of the template takes place within the method FB_Init.

Block diagram



Principle diagram

The diagram shows the integration of the template within a plant.



Syntax

```
FUNCTION_BLOCK FB_BA_AC_OpMod1St EXTENDS FB_BA_View
VAR_INPUT
  bSmExtt                   : BOOL;
  bFireDetc                 : BOOL;
  bForcVent                 : BOOL;
  bEmgOff                   : BOOL;
  bFrst                     : BOOL;
  bCentSwiOff               : BOOL;
  bAlmShutOff               : BOOL;
  bNgtCol                   : BOOL;
  bColDwnPrtc               : BOOL;
  bOvrHtgPrtc               : BOOL;
END_VAR
VAR_OUTPUT
  eOpMode                   : E_BA_AC_OpMod01;
END_VAR
VAR_INPUT CONSTANT
  OpModMan                  : FB_BA_MV_Op;
  Sched                     : FB_BA_SchedM;
  OpModPr                   : FB_BA_MV_Op;
END_VAR
VAR
  PrioSwiOpMod              : FB_BA_PrioSwi_UDI24;
END_VAR
```

Inputs

Name	Type	Description
bSmExtt	BOOL	Smoke extraction of the plant requested
bFireDetc	BOOL	Fire alarm message from fire alarm center
bForceVent	BOOL	Forced ventilation request
bEmgOff	BOOL	Emergency stop
bFrst	BOOL	Frost protection program active
bCentSwiOff	BOOL	Central switch-off
bAlmShutOff	BOOL	Collective error message - shut down plant
bNgtCol	BOOL	Request from the summer night cooling program (see FB_BA_AC_SumNgtCol [▶ 783])
bColDwnPrtc	BOOL	Request program cooling protection
bOvrHtgPrtc	BOOL	Request program overheating protection

Outputs

Name	Type	Description
eOpMode	E_BA_AC_OpMod01 [▶ 699]	Output of the current operation mode eOpMode

Inputs CONSTANT

Name	Type	Description
OpModMan	FB_BA_MV_Op [▶ 239]	The Multistate-Value object represents an operating modes switch with the operating modes Auto, Manual-Off and Manual-On.
Sched	FB_BA_SchedM [▶ 225]	Schedule object (automatic) for the "BuildingEnergyLevel".
OpModPr	FB_BA_MV_Op [▶ 239]	The Multistate-Value object indicates the state of the currently valid plant operation mode.

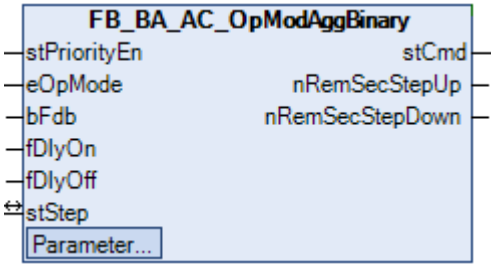
Variables

Name	Type	Description
PrioSwiOpMod	FB_BA_PrioSwi_UDI24 [▶ 433]	The priority switch determines the current operation mode *eOpMode* from the pending events or commands of the plant.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.3.3 FB_BA_AC_OpModAggBinary



The template serves as a link between steps within the step sequence control of an air conditioning system.

The function block *Step* is the core of the template and represents the binary receive block of a step sequence controller.

The connected aggregate receives its commands via the command structure *stCmd*.

The data exchange to the control block of the step sequence control (*StepCtrlAgg*) takes place via the data and command structure *stStep*.

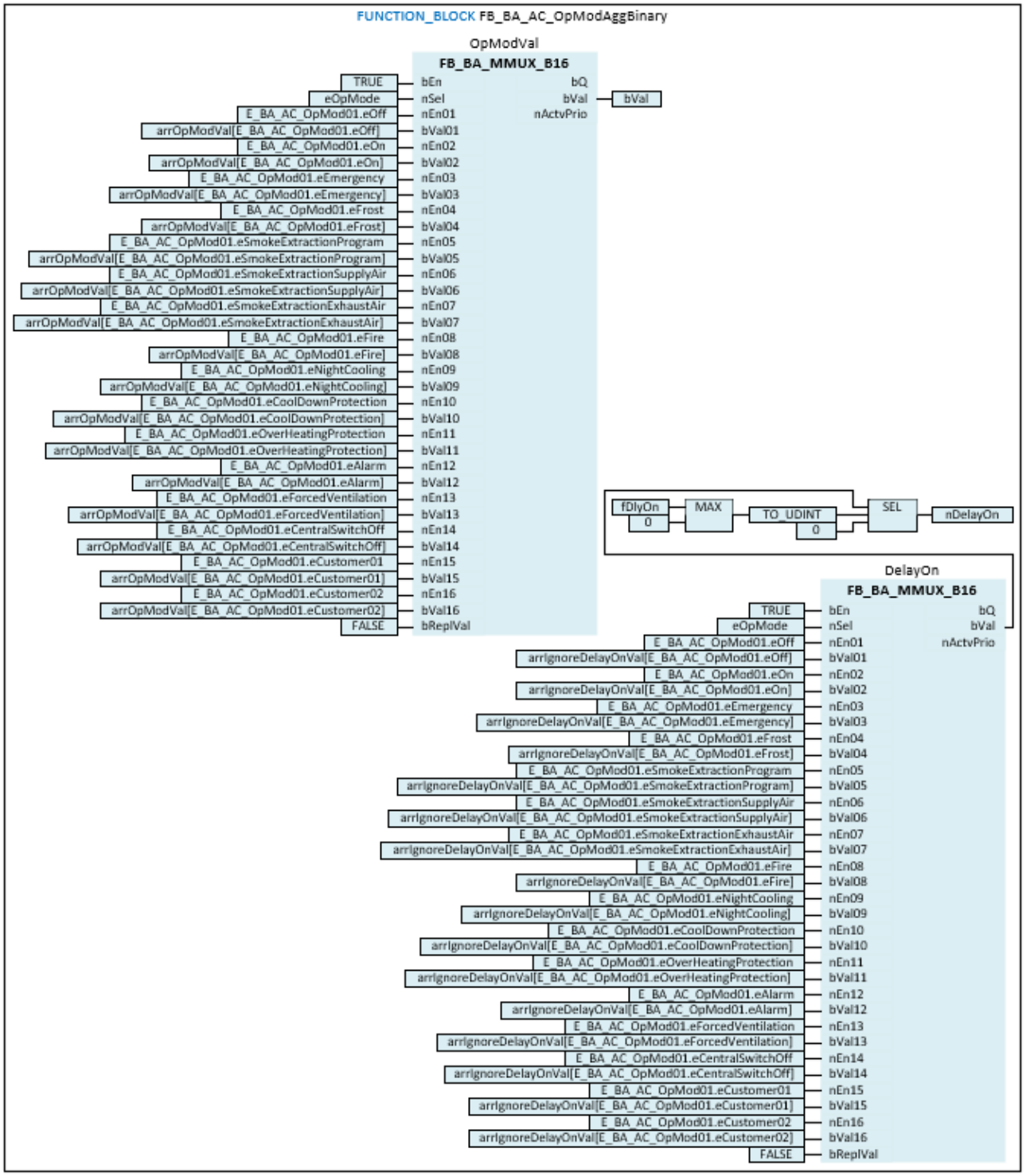
The four multiplexers *OpModVal*, *DelayOn*, *DelayOff* and *IgnoreFdb* define the switch-on and switch-off conditions of the plant step based on the plant operation mode *eOpMode*. Each of these multiplexers has an array to be parameterized for the output value of the multiplexers (*arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal*).

The resulting values of the multiplexers *bVal*, *_bFdb*, *nDelayOn* and *nDelayOff* are transmitted to the receive block of the step sequence controller *Step*, see block diagram part 02.

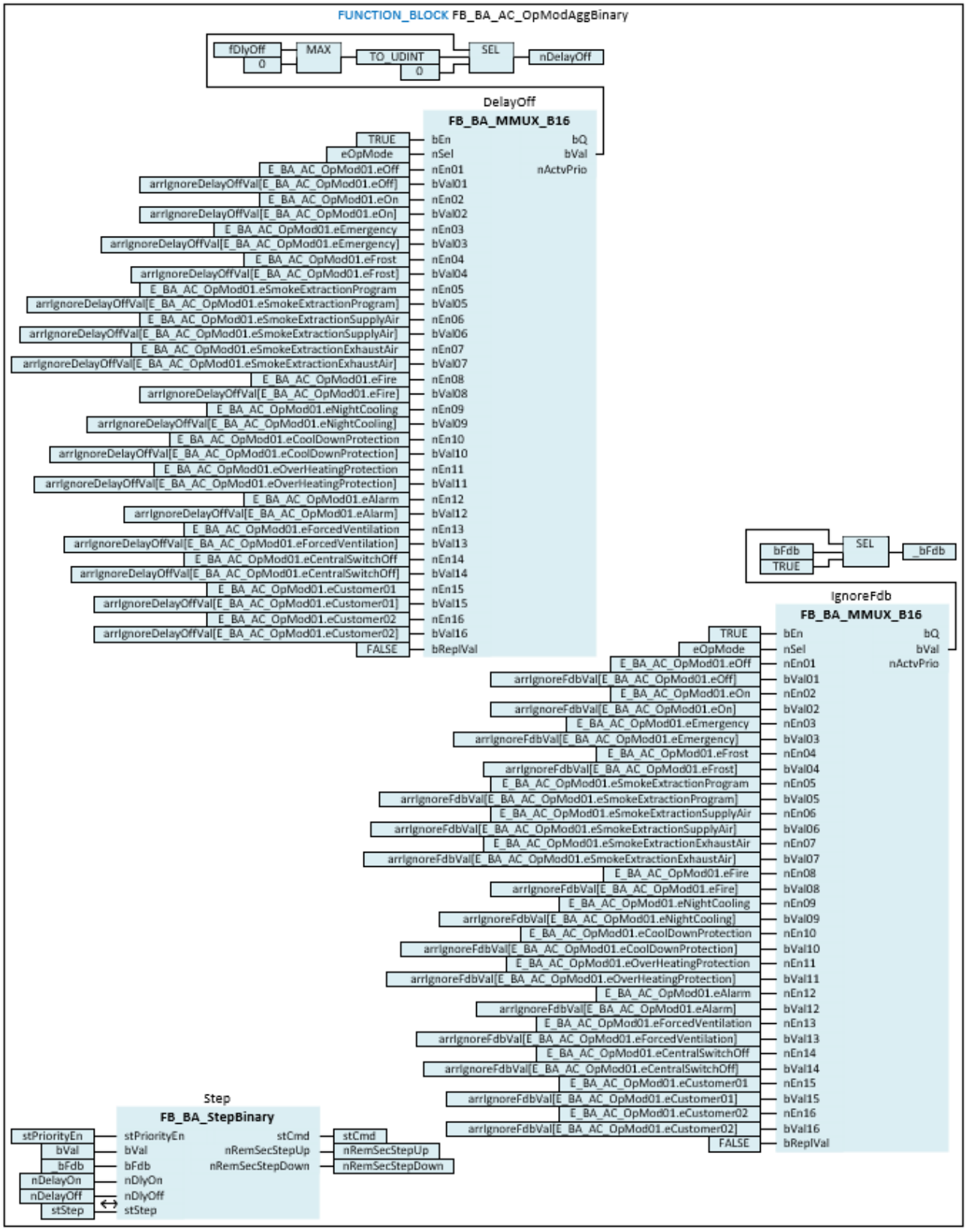
The parameterization of the arrays can be found in an air conditioning system in the methods below the example template FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 786], see Methods.

Block diagram

Part 01


FUNCTION_BLOCK FB_BA_AC_OpModAggBinary
OpModVal
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrOpModVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrOpModVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrOpModVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrOpModVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrOpModVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrOpModVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrOpModVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrOpModVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrOpModVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrOpModVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrOpModVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrOpModVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrOpModVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio
bVal
fDlyOn
0
MAX
TO_UDINT
SEL
0
nDelayOn
DelayOn
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio

Part 02



Syntax

```
FUNCTION_BLOCK FB_BA_AC_OpModAggBinary
VAR_INPUT
  stPriorityEn              : ST_BA_PriorityEn;
  eOpMode                   : E_BA_AC_OpMod01;
  bFdb                      : BOOL;
  fDlyOn                    : REAL;
  fDlyOff                   : REAL;
END_VAR
VAR_OUTPUT
```

```
  stCmd                    : ST_BA_Binary;
  nRemSecStepUp            : UDINT;
  nRemSecStepDown          : UDINT;
END_VAR
VAR_IN_OUT
  stStep                   : ST_BA_Step;
END_VAR
VAR_INPUT CONSTANT
  OpModVal                 : FB_BA_MMUX_B16;
  DelayOn                  : FB_BA_MMUX_B16;
  DelayOff                 : FB_BA_MMUX_B16;
  IgnoreFdb                : FB_BA_MMUX_B16;
  Step                     : FB_BA_StepBinary;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bInit                    : BOOL := TRUE;
  arrOpModVal              : ARRAY [1..16] OF BOOL;
  arrIgnoreDelayOnVal      : ARRAY [1..16] OF BOOL;
  arrIgnoreDelayOffVal     : ARRAY [1..16] OF BOOL;
  arrIgnoreFdbVal          : ARRAY [1..16] OF BOOL;
END_VAR
VAR
  bVal                     : BOOL;
  _bFdb                    : BOOL;
  nDelayOn                 : UDINT;
  nDelayOff                : UDINT;
END_VAR
```

Inputs

Name	Type	Description
stPriorityEn	ST_BA_PriorityEn [▶ 278]	Input of the current command structure stPriorityEn. This includes the system enable for the step sequence control of an air conditioning system and the associated enables for the priorities "Safety", "Critical" and "Program".
eOpMode	E_BA_AC_OpMod01 [▶ 699]	Input of the current operation mode.
bFdb	BOOL	Feedback from the connected aggregate of the step sequence controller. The feedback is required for switching to the next step. *bFdb* is only considered if the aggregate is the active one in the step sequence control. By parameterizing the individual elements of the array *arrIgnoreFdbVal* with a TRUE, the feedback from the aggregate can be ignored for the different operation modes *eOpMode*.
fDlyOn	REAL	Time specification of the start-up delay [s]. The time specification is required for switching to the next step. By parameterizing the individual elements of the array *arrIgnoreDelayOnVal* with a TRUE, the start-up delay of the step can be ignored for the different operation modes *eOpMode*.
fDlyOff	REAL	Time specification of switch-off delay [s]. The time specification is needed to switch off the active step. By parameterizing the individual elements of the array *arrIgnoreDelayOffVal* with a TRUE, the switch-off delay of the step can be ignored for the different operation modes *eOpMode*.

Outputs

Name	Type	Description
stCmd	ST_BA_Binary [▶ 277]	The command structure *stCmd* transmits the enables and switching values of the priorities to the connected aggregate.
nRemSecStepUp	UDINT	Countdown to switch on the next step [s].
nRemSecStepDown	UDINT	Countdown to switching off the active step [s].

Inputs/outputs

Name	Type	Description
stStep	ST_BA_Step [▶ 277]	Data and command structure between the step sequence function block *FB_BA_StepBinary* and the control block of the step sequence FB_BA_StepCtrlAgg16 [▶ 471].

Inputs CONSTANT

Name	Type	Description
OpModVal	FB_BA_MMUX_B16 [▶ 428]	The multiplexer determines the switch-on value of the connected aggregate. The resulting value is sent to the output *bVal*.
DelayOn	FB_BA_MMUX_B16 [▶ 428]	The multiplexer and the connected network of functions determine the switching delay to the next step. The resulting value is sent to the output *nDelayOn*.
DelayOff	FB_BA_MMUX_B16 [▶ 428]	The multiplexer and the connected network of functions determine the switch-off delay of the current step. The resulting value is sent to the output *nDelayOff*.
IgnoreFdb	FB_BA_MMUX_B16 [▶ 428]	The multiplexer and the connected network of functions determine the feedback of the aggregate. The resulting value is sent to the output *_bFdb*.
Step	FB_BA_StepBinary [▶ 473]	The function block represents the binary receive block of a step sequence controller and controls the connected aggregate via the command structure *stCmd*. The data exchange to the control block of the step sequence control (FB_BA_StepCtrlAgg16 [▶ 471]) takes place via the data and command structure *stStep*.

Inputs CONSTANT PERSISTENT

Name	Type	Description
bInit	BOOL	The variable is used to initialize the arrays *arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal* in the methods below the template FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 786]. After starting the controller, the variable is set to FALSE after a PLC cycle and this state is saved persistently.
arrOpModVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array, the switch-on value of the connected aggregate can be defined on the step sequence function block for the different operation modes *eOpMode*.
arrIgnoreDelayOnVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array with a TRUE, the switching delay to the next step within the step sequence control can be ignored for the different operation modes *eOpMode*.
arrIgnoreDelayOffVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array with a TRUE, the switch-off delay of the current step within the step sequence control can be ignored for the different operation modes *eOpMode*.
arrIgnoreFdbVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array with a TRUE, the feedback from the aggregate *bFdb* can be ignored for the different operation modes *eOpMode*.

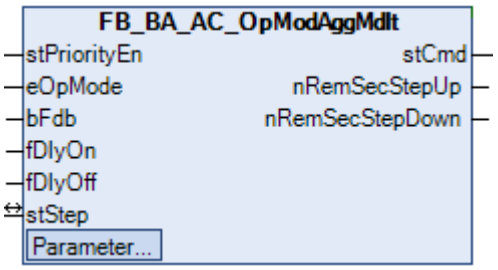
Variables

Name	Type	Description
bVal	BOOL	The variable is the result of the operation mode *eOpMode*, the multiplexer *OpModVal* and the array *arrOpModVal*. *bVal* defines the switch-on value at the step sequence function block *Step*.
_bFdb	BOOL	The variable is the result of the operation mode *eOpMode*, the multiplexer *IgnoreFdb* and the array *arrIgnoreFdbVal*. *_bFdb* defines the switching condition to the next step in the step sequence function block *Step*.
nDelayOn	UDINT	The variable is the result of the operation mode *eOpMode*, the multiplexer *DelayOn* and the array *arrIgnoreDelayOnVal*. *nDelayOn* defines the switching delay to the next step on the step sequence function block *Step*.
nDelayOff	UDINT	The variable is the result of the operation mode *eOpMode*, the multiplexer *DelayOff* and the array *arrIgnoreDelayOffVal*. *nDelayOff* defines the switch-off delay of the current step on the step sequence function block *Step*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.3.4 FB_BA_AC_OpModAggMdlt



The template serves as a link between the steps within the step sequence control of an air conditioning system.

The function block *Step* is the core of the template and represents the binary receive block of a step sequence controller.

The connected aggregate receives its commands via the command structure *stCmd*.

The data exchange to the control block of the step sequence control (*StepCtrlAgg*) takes place via the data and command structure *stStep*.

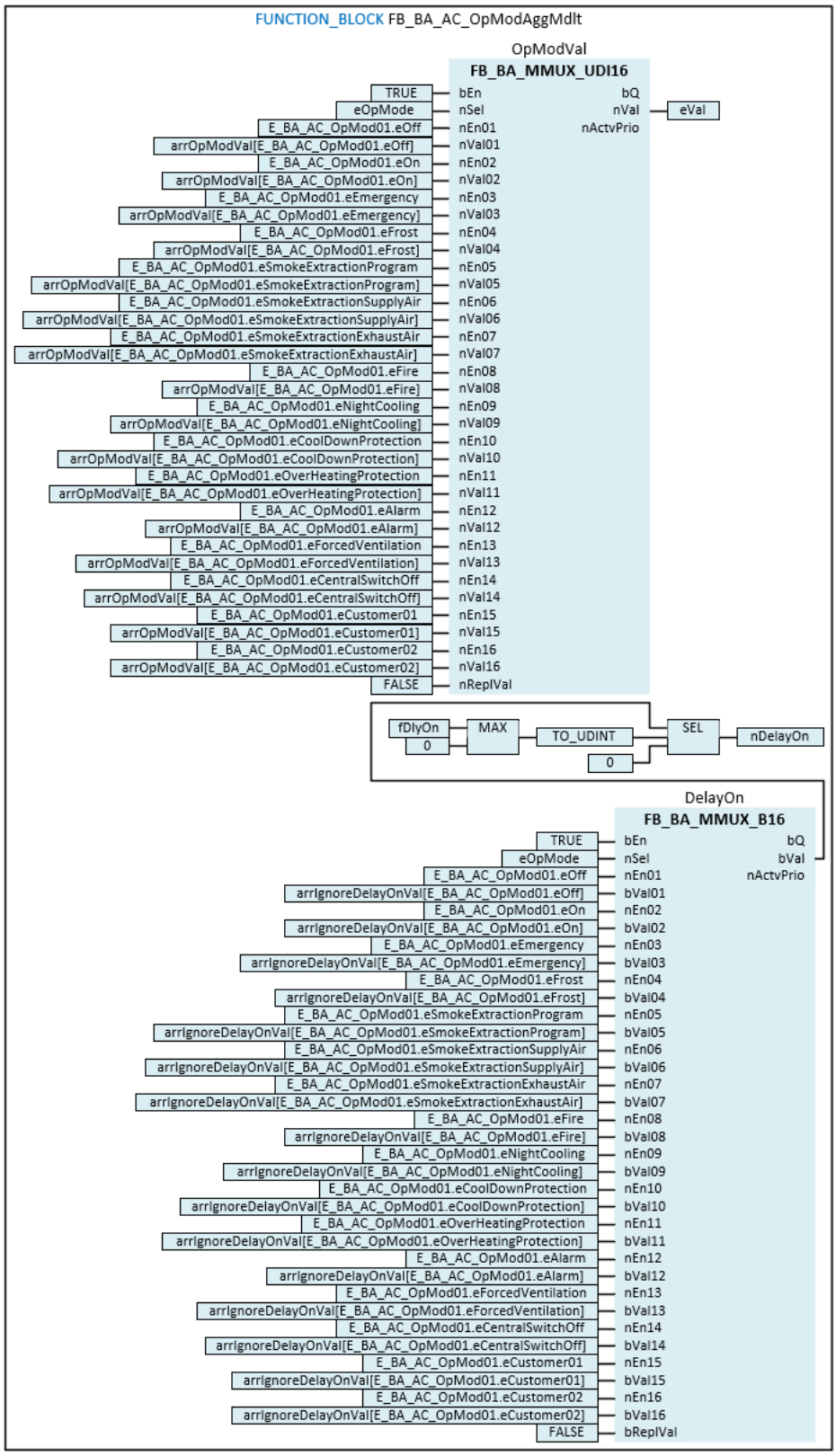
The 4 multiplexers *OpModVal*, *DelayOn*, *DelayOff* and *IgnoreFdb* define the switch-on and switch-off conditions of the system step based on the plant operation mode *eOpMode*. Each of these multiplexers has an array to be parameterized for the output value of the multiplexers (*arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal*).

The resulting values of the multiplexers *eVal*, *_bFdb*, *nDelayOn* and *nDelayOff* are transmitted to the receive block of the step sequence controller *Step*, see block diagram part 02.

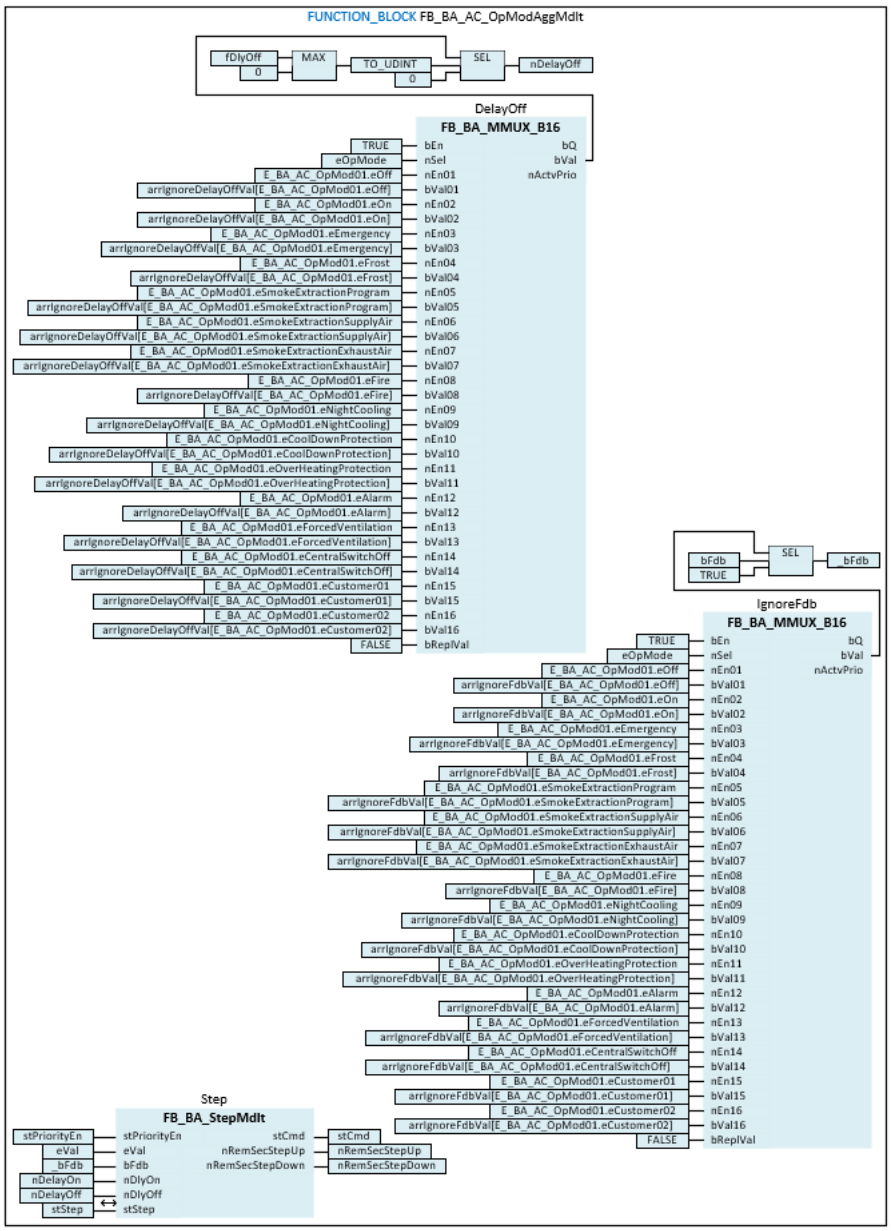
The parameterization of the arrays can be found in an air conditioning system in the methods below the example template FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 786], see Methods.

Block diagram

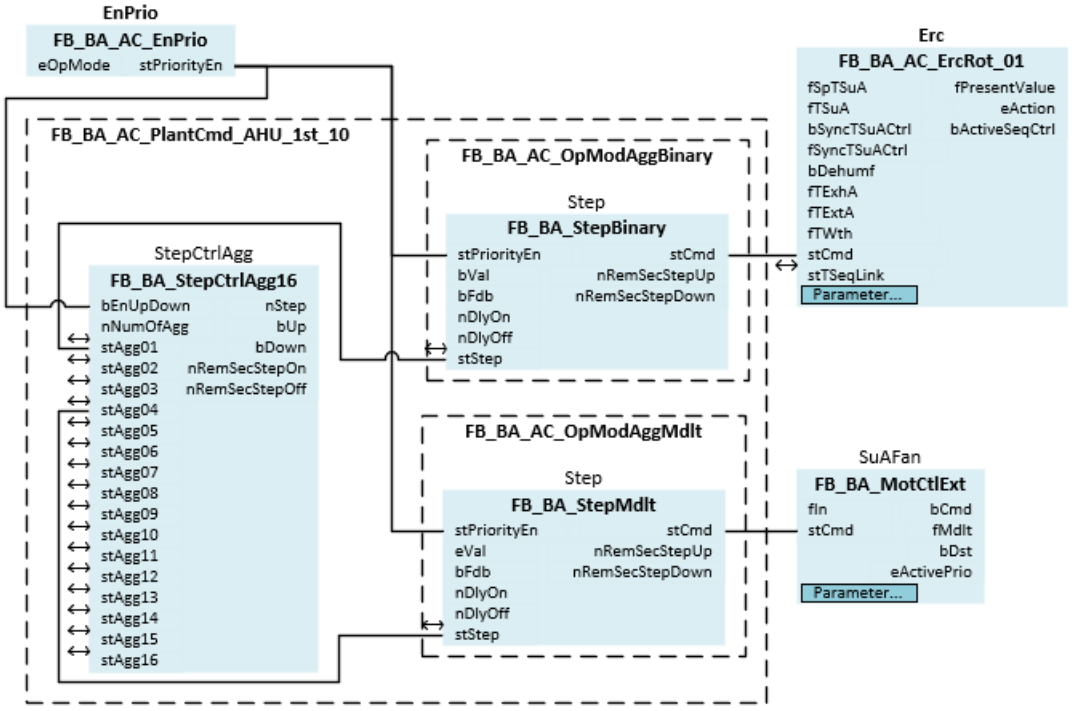
Part 01


FUNCTION_BLOCK FB_BA_AC_OpModAggMdlt
OpModVal
FB_BA_MMUX_UDI16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrOpModVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrOpModVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrOpModVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrOpModVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrOpModVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrOpModVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrOpModVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrOpModVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrOpModVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrOpModVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrOpModVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrOpModVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrOpModVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
nVal01
nEn02
nVal02
nEn03
nVal03
nEn04
nVal04
nEn05
nVal05
nEn06
nVal06
nEn07
nVal07
nEn08
nVal08
nEn09
nVal09
nEn10
nVal10
nEn11
nVal11
nEn12
nVal12
nEn13
nVal13
nEn14
nVal14
nEn15
nVal15
nEn16
nVal16
nReplVal
bQ
nVal
nActvPrio
eVal
fDlyOn
0
MAX
TO_UDINT
SEL
0
nDelayOn
DelayOn
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio

Part 02


FUNCTION_BLOCK FB_BA_AC_OpModAggMdlt
fDlyOff
0
MAX
TO_UDINT
0
SEL
nDelayOff
DelayOff
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio
bFdb
TRUE
SEL
_bFdb
IgnoreFdb
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrIgnoreFdbVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrIgnoreFdbVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrIgnoreFdbVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrIgnoreFdbVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrIgnoreFdbVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrIgnoreFdbVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrIgnoreFdbVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrIgnoreFdbVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrIgnoreFdbVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrIgnoreFdbVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrIgnoreFdbVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrIgnoreFdbVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio
Step
FB_BA_StepMdlt
stPriorityEn
eVal
_bFdb
nDelayOn
nDelayOff
stStep
stPriorityEn
eVal
bFdb
nDlyOn
nDlyOff
stStep
stCmd
nRemSecStepUp
nRemSecStepDown
stCmd
nRemSecStepUp
nRemSecStepDown

Principle diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_OpModAggMdlt
VAR_INPUT
  stPriorityEn              : ST_BA_PriorityEn;
  eOpMode                   : E_BA_AC_OpMod01;
  bFdb                      : BOOL;
  fDlyOn                    : REAL;
  fDlyOff                   : REAL;
END_VAR
VAR_OUTPUT
  stCmd                     : ST_BA_Mdlt;
  nRemSecStepUp             : UDINT;
  nRemSecStepDown           : UDINT;
END_VAR
VAR_IN_OUT
  stStep                    : ST_BA_Step;
END_VAR
VAR_INPUT CONSTANT
  OpModVal                  : FB_BA_MMUX_UDI16;
  DelayOn                   : FB_BA_MMUX_B16;
  DelayOff                  : FB_BA_MMUX_B16;
  IgnoreFdb                 : FB_BA_MMUX_B16;
  Step                      : FB_BA_StepMdlt;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bInit                     : BOOL := TRUE;
  arrOpModVal               : ARRAY [1..16] OF E_BA_Mdlt;
  arrIgnoreDelayOnVal       : ARRAY [1..16] OF BOOL;
  arrIgnoreDelayOffVal      : ARRAY [1..16] OF BOOL;
  arrIgnoreFdbVal           : ARRAY [1..16] OF BOOL;
END_VAR
VAR
  eVal                      : E_BA_Mdlt;
  _bFdb                     : BOOL;
  nDelayOn                  : UDINT;
  nDelayOff                 : UDINT;
END_VAR
```

Inputs

Name	Type	Description
stPriorityEn	ST_BA_PriorityEn [▶ 278]	Input of the current command structure stPriorityEn. This includes the system enable for the step sequence control of an air conditioning system and the associated enables for the priorities "Safety", "Critical" and "Program".
eOpMode	E_BA_AC_OpMod01 [▶ 699]	Input of the current operation mode.
bFdb	BOOL	Feedback from the connected aggregate of the step sequence controller. The feedback is required for switching to the next step. *bFdb* is only considered if the aggregate is the active one in the step sequence control. By parameterizing the individual elements of the array *arrIgnoreFdbVal* with a TRUE, the feedback from the aggregate can be ignored for the different operation modes *eOpMode*.
fDlyOn	REAL	Time specification of the start-up delay [s]. The time specification is required for switching to the next step. By parameterizing the individual elements of the array *arrIgnoreDelayOnVal* with a TRUE, the start-up delay of the step can be ignored for the different operation modes *eOpMode*.
fDlyOff	REAL	Time specification of switch-off delay [s]. The time specification is needed to switch off the active step. By parameterizing the individual elements of the array *arrIgnoreDelayOffVal* with a TRUE, the switch-off delay of the step can be ignored for the different operation modes *eOpMode*.

Outputs

Name	Type	Description
stCmd	ST_BA_Binary [▶ 277]	The command structure *stCmd* transmits the enables and switching values of the priorities to the connected aggregate.
nRemSecStepUp	UDINT	Countdown to switch on the next step [s].
nRemSecStepDown	UDINT	Countdown to switching off the active step [s].

Inputs/outputs

Name	Type	Description
stStep	ST_BA_Step [▶ 277]	Data and command structure between the step sequence function block *FB_BA_StepBinary* and the control block of the step sequence FB_BA_StepCtrlAgg16 [▶ 471].

Inputs CONSTANT

Name	Type	Description
OpModVal	FB_BA_MMUX_B16 [▶ 428]	The multiplexer determines the switch-on value of the connected aggregate. The resulting value is sent to the output *bVal*.
DelayOn	FB_BA_MMUX_B16 [▶ 428]	The multiplexer and the connected network of functions determine the switching delay to the next step. The resulting value is sent to the output *nDelayOn*.
DelayOff	FB_BA_MMUX_B16 [▶ 428]	The multiplexer and the connected network of functions determine the switch-off delay of the current step. The resulting value is sent to the output *nDelayOff*.
IgnoreFdb	FB_BA_MMUX_B16 [▶ 428]	The multiplexer and the connected network of functions determine the feedback of the aggregate. The resulting value is sent to the output *_bFdb*.
Step	FB_BA_StepBinary [▶ 473]	The function block represents the binary receive block of a step sequence controller and controls the connected aggregate via the command structure *stCmd*. The data exchange to the control block of the step sequence control (FB_BA_StepCtrlAgg16 [▶ 471]) takes place via the data and command structure *stStep*.

Inputs CONSTANT PERSISTENT

Name	Type	Description
bInit	BOOL	The variable is used to initialize the arrays *arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal* in the methods below the template FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 786]. After starting the controller, the variable is set to FALSE after a PLC cycle and this state is saved persistently.
arrOpModVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array, the switch-on value of the connected aggregate can be defined on the step sequence function block for the different operation modes *eOpMode*.
arrIgnoreDelayOnVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array with a TRUE, the switching delay to the next step within the step sequence control can be ignored for the different operation modes *eOpMode*.
arrIgnoreDelayOffVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array with a TRUE, the switch-off delay of the current step within the step sequence control can be ignored for the different operation modes *eOpMode*.
arrIgnoreFdbVal	ARRAY [1..16] OF BOOL	By parameterizing the individual elements of the array with a TRUE, the feedback from the aggregate *bFdb* can be ignored for the different operation modes *eOpMode*.

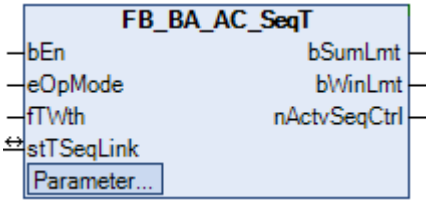
Variables

Name	Type	Description
bVal	BOOL	The variable is the result of the operation mode *eOpMode*, the multiplexer *OpModVal* and the array *arrOpModVal*. *bVal* defines the switch-on value at the step sequence function block *Step*.
_bFdb	BOOL	The variable is the result of the operation mode *eOpMode*, the multiplexer *IgnoreFdb* and the array *arrIgnoreFdbVal*. *_bFdb* defines the switching condition to the next step in the step sequence function block *Step*.
nDelayOn	UDINT	The variable is the result of the operation mode *eOpMode*, the multiplexer *DelayOn* and the array *arrIgnoreDelayOnVal*. *nDelayOn* defines the switching delay to the next step on the step sequence function block *Step*.
nDelayOff	UDINT	The variable is the result of the operation mode *eOpMode*, the multiplexer *DelayOff* and the array *arrIgnoreDelayOffVal*. *nDelayOff* defines the switch-off delay of the current step on the step sequence function block *Step*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.3.5 FB_BA_AC_SeqT



The template represents the start and control of the supply air temperature sequence control of an air conditioning (HVAC) system.

The initialization of the template takes place within the method FB_Init.

Start behavior of the sequence

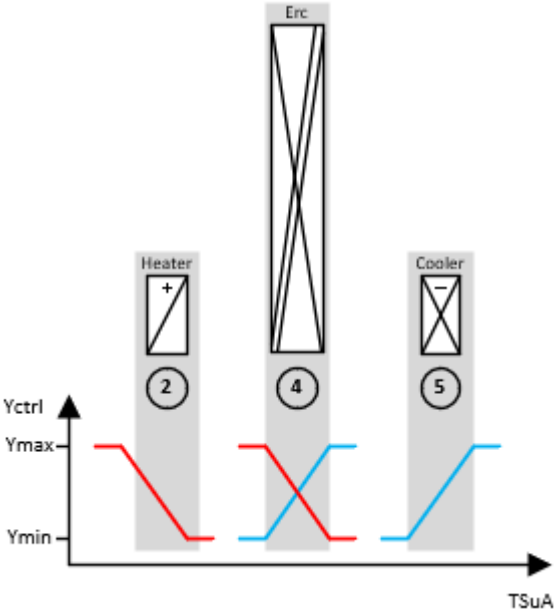
On startup of the air-conditioning plant the system determines whether to start with the heating, cooling or heat recovery sequence. The start sequence is selected depending on the plant operation mode *eOpMode* and the outside temperature *fTWth*. The result of the network at the input *nNumStartCtrl* of the function block *SeqLink* determines the start controller of the control sequence.

If the sequence controller specified at *nNumStartCtrl* is not ready for operation, the parameter enum eNoStartCtrlFound of the function block *SeqLink* is used to define the procedure for finding a new start controller. This ensures that the sequence always starts.

Control of the sequence control

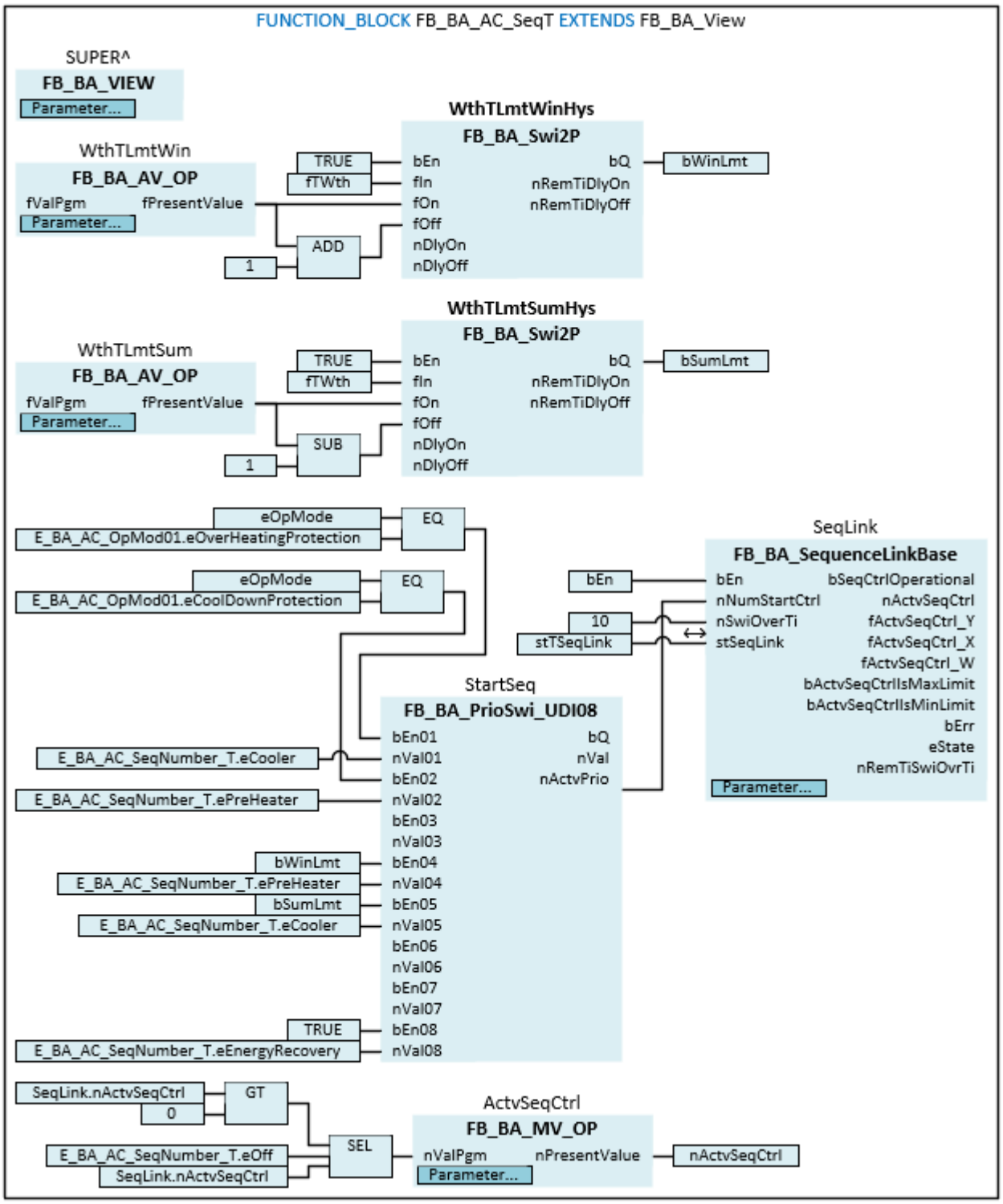
Only one element of the sequence can be controlling. If a regulating sequence controller reaches its maximum or minimum value (Ymax,Ymin), the next controller in the sequence is switched to depending on the control direction.

Switching to another sequence controller can be delayed by specifying the time *nSwiOverTi* of the function block *SeqLink*.

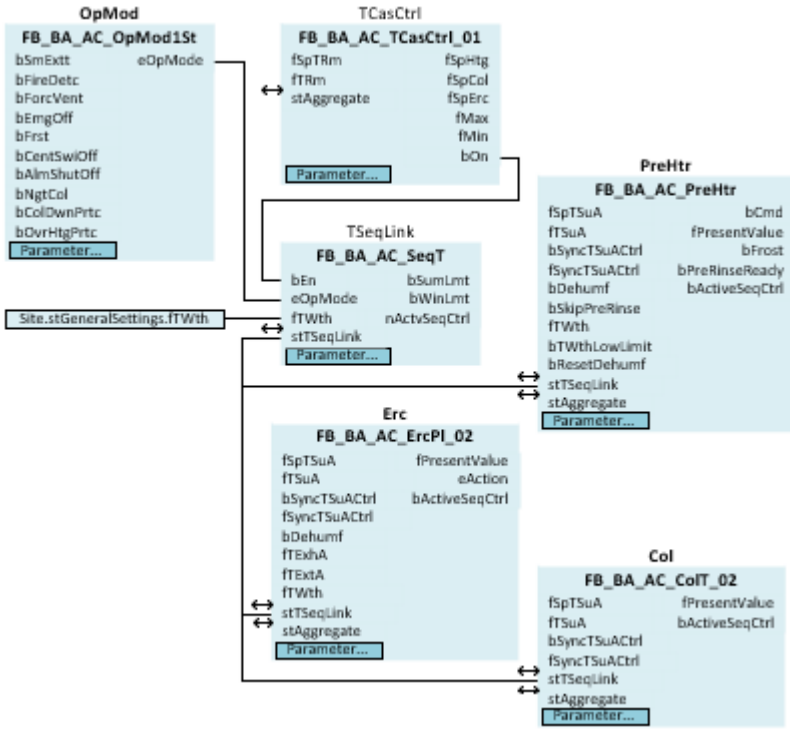


The order of the temperature control sequences is specified by the globally defined enumeration variable E_BA_SeqNumber_T [▶ 697].

Block diagram



Principle diagram plant



Syntax

```
FUNCTION_BLOCK FB_BA_AC_SeqT EXTENDS FB_BA_View
VAR_INPUT
  bEn                 : BOOL;
  eOpMode             : E_BA_AC_OpMod01;
  fTWth               : REAL;
END_VAR
VAR_OUTPUT
  bSumLmt             : BOOL;
  bWinLmt             : BOOL;
  nActvSeqCtrl        : UDINT;
END_VAR
VAR_IN_OUT
  stTSeqLink          : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  WthTLmtSum          : FB_BA_AV_Op;
  WthTLmtWin          : FB_BA_AV_Op;
  SeqLink             : FB_BA_SequenceLinkBase;
  ActvSeqCtrl         : FB_BA_MV_Op;
END_VAR
VAR
  WthTLmtWinHys       : FB_BA_Swi2P;
  WthTLmtSumHys       : FB_BA_Swi2P;
  StartSeq            : FB_BA_PrioSwi_UDI08;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Activation of the supply air temperature sequence control.
eOpMode	E_BA_AC_OpMod01 [▶ 699]	The plant operation mode is used for the selection of the start sequence controller.
fTWth	REAL	Input for the weather temperature.

Outputs

Name	Type	Description
bSumLmt	BOOL	The output variable indicates that the ventilation system starts with the cooling sequence.
bWinLmt	BOOL	The output variable indicates that the ventilation system starts with the heating sequence.
nActvSeqCtrl	UDINT	The output variable shows the number of the active sequence controller.

Inputs/outputs

Name	Type	Description
stTSeqLink	ST_BA_SeqLink [▶ 123]	Data and command structure between the individual sequence controllers FB_BA_SeqCtrl of the temperature control sequence (preheater, energy recovery, cooler) and the control block *SeqLink*.

Inputs CONSTANT

Name	Type	Description
WthTLmtSum	FB_BA_AV_Op [▶ 202]	AV object for input of an outside temperature limit value from which the air-conditioning plant starts in the heating sequence.
WthTLmtWin	FB_BA_AV_Op [▶ 202]	AV object for input of an outside temperature limit value from which the air-conditioning plant starts in the cooling sequence
SeqLink	FB_BA_SequenceLinkBase [▶ 171]	The function block *SeqLinkH* is the core of the template *FB_BA_AC_SeqT*. The sequence linker is connected to the supply air controllers of the sequence via the data structure stTSeqLink [▶ 123]. It is the central control element, and responsible for switching between the sequence controllers and starting of the control sequence.
ActvSeqCtrl	FB_BA_MV_Op [▶ 239]	The MV object indicates the currently active sequence controller.

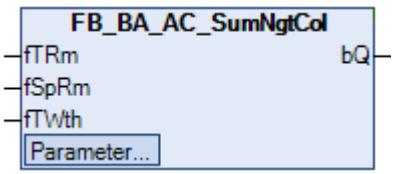
Variables

Name	Type	Description
WthTLmtWinHys	FB_BA_Swi2P	Two-position switch which generates the binary switching signal for *bWinLmt* based on the weather temperature *fTWth* and the winter weather temperature limit value *WthTLmtWin*.
WthTLmtSumHys	FB_BA_Swi2P	Two-position switch which generates the binary switching signal for *bWinLmt* based on the weather temperature *fTWth* and the limit value weather temperature summer *WthTLmtSum*.
StartSeq	FB_BA_PrioSwi_UDI08 [▶ 433]	The priority switch is used to select the start sequence. **Prio 1: bEN01** In the overheating protection *E_BA_AC_OpMod01.eOverHeatingProtection*, the cooling sequence (*E_BA_AC_SeqNumber.eCooler*) is always started. **Prio 2: bEN02** In the cool-down protection mode *E_BA_AC_OpMod01.eCoolDownProtection*, the heating sequence (*E_BA_AC_SeqNumber.ePreHeater*) is always started. **Prio 4: bEN04** The upstream function block *WthTLmtWinHys* checks whether the actual outside temperature is below the critical value of AV object *WthTLmtSum*. If this is the case, the air-conditioning system starts with the sequence controller of the preheater (*E_BA_AC_SeqNumber.ePreHeater*). **Prio 5: bEN05** The upstream function block *WthTLmtSumHys* checks whether the actual outside temperature is above the value of AV object *WthTLmtSum*. If this is the case, the air conditioning system starts with the sequence controller of the cooler (*E_BA_AC_SeqNumber.eCooler*). **Prio 8: bEN08** In the transitional periods between winter and summer, the inputs of *StartSeq.bEn04* and *StartSeq.bEn05* are FALSE on account of the weather. In this case, priority 8 applies to the priority switch, meaning that the air-conditioning system is started in the energy recovery sequence (*E_BA_AC_SeqNumber.eEnergyRecovery*).

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

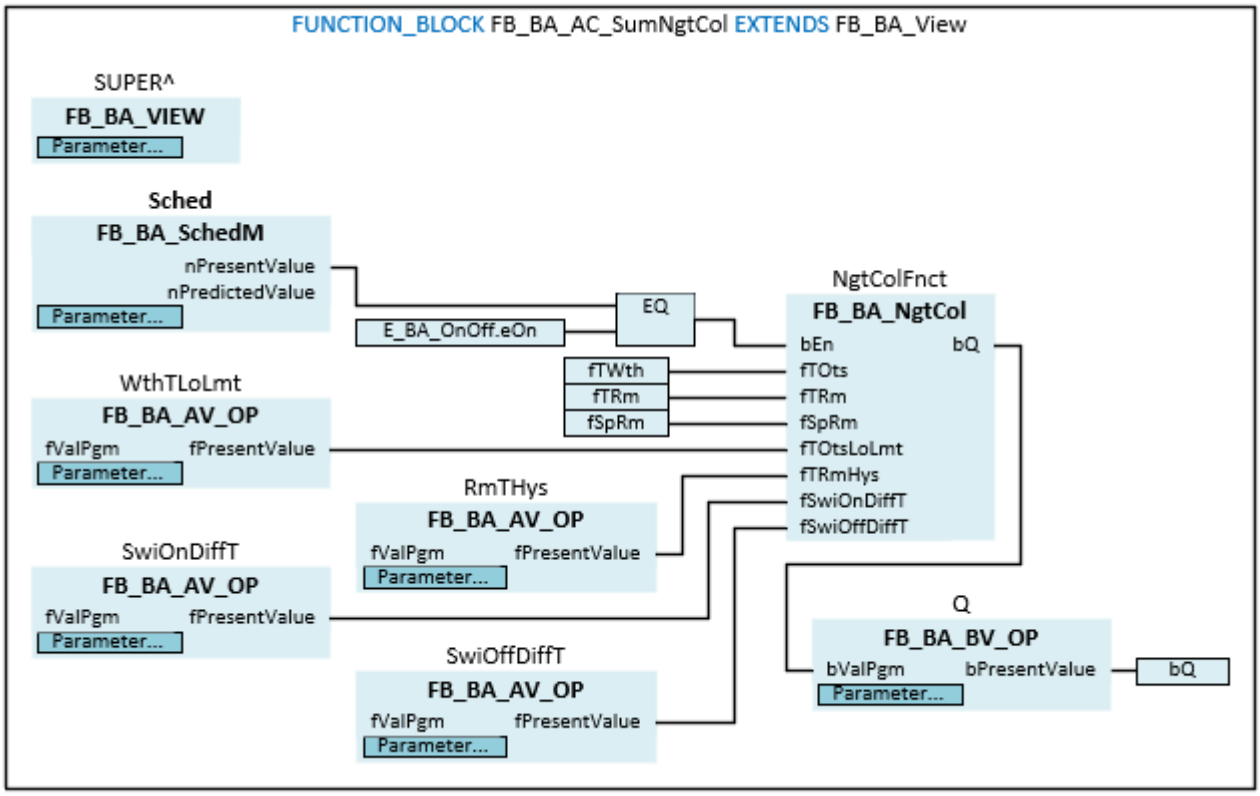
6.1.4.2.1.1.3.6 FB_BA_AC_SumNgtCol



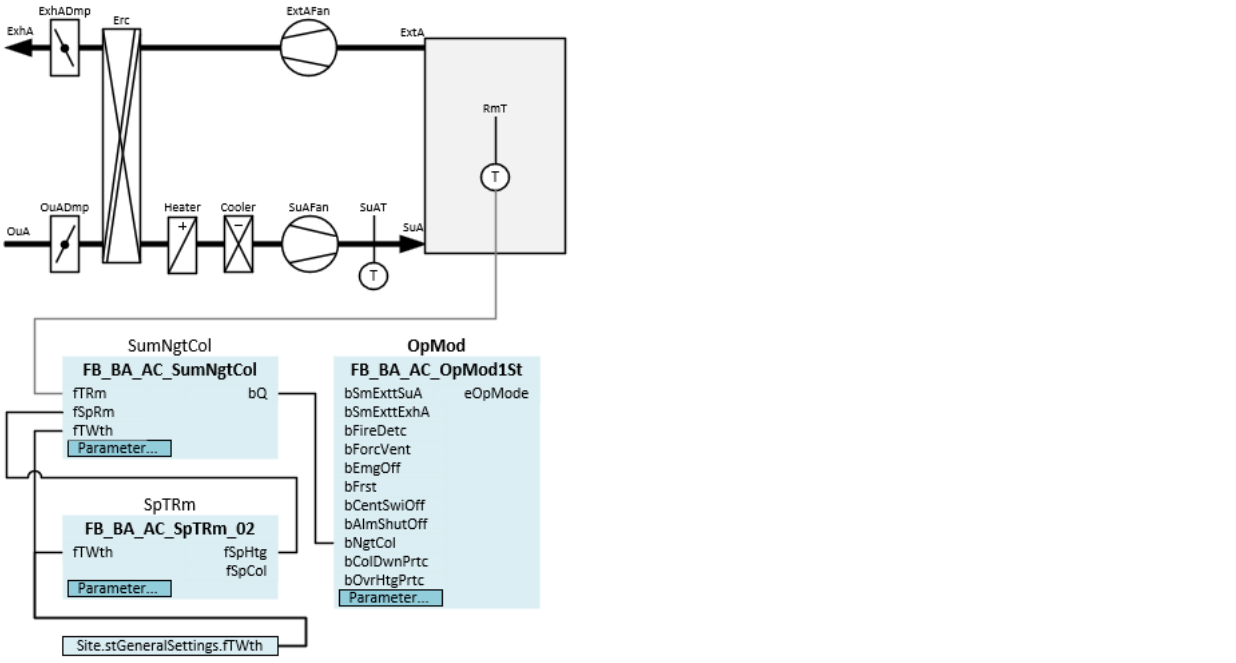
The template represents a night cooling function for ventilation systems. It is used to cool down rooms that have been heated up during the day with cool outside air at night. The summer night cooling function serves to improve the quality of the air and to save electrical energy. Electrical energy for cooling is saved during the first hours of the next summer day.

The initialization of the template takes place within the method FB_Init.

Block diagram



Principle diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_SumNgtCol EXTENDS FB_BA_View
VAR_INPUT
  fTRm                : REAL;
  fSpRm               : REAL;
  fTWth               : REAL;
END_VAR
VAR_OUTPUT
  bQ                  : BOOL;
END_VAR
VAR_INPUT CONSTANT
  WthTLoLmt           : FB_BA_AV_Op;
  RmTHys              : FB_BA_AV_Op;
  SwiOnDiffT          : FB_BA_AV_Op;
  SwiOffDiffT         : FB_BA_AV_Op;
  Q                   : FB_BA_BV_Op;
  Sched               : FB_BA_SchedM;
END_VAR
VAR  NgtColFnct       : FB_BA_NgtCol;
END_VAR
```

Inputs

Name	Type	Description
fTRm	REAL	Input measured value room temperature.
fSpRm	REAL	Input setpoint room temperature.
fTWth	REAL	Input for the weather temperature.

Outputs

Name	Type	Description
bQ	BOOL	Summer night cooling switched on.

Inputs CONSTANT

Name	Type	Description
WthTLoLmt	FB_BA_AV_Op [▶ 202]	Input of the value for the lower outside temperature. The value is intended to prevent excessive cooling.
RmTHys	FB_BA_AV_Op [▶ 202]	Input of the room temperature hysteresis.
SwiOnDiffT	FB_BA_AV_Op [▶ 202]	Switch-on value Temperature difference between the room temperature and the outside temperature [K].
SwiOffDiffT	FB_BA_AV_Op [▶ 202]	Switch-off value Temperature difference between the room temperature and the outside temperature [K].
Q	FB_BA_BV_Op [▶ 216]	The BV object indicates that night cooling is switched on.
Sched	FB_BA_SchedM [▶ 225]	The schedule object can be used to define periods in which night cooling is enabled.

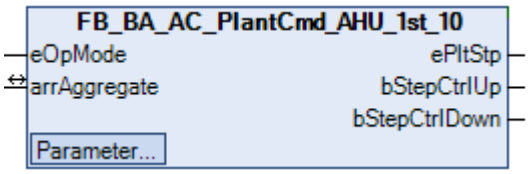
Variables

Name	Type	Description
NgtColFct	FB_BA_NgtCol [▶ 289]	The function block is the core of the template and includes the actual control process of the night cooling program.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.3.7 FB_BA_AC_PlantCmd_AHU_1st_10



The template represents the plant command of an air conditioning system without humidification and dehumidification.

The function block *PlantControl* is the core of the plant command system. It serves as a higher-level plant control unit for the connected aggregates.

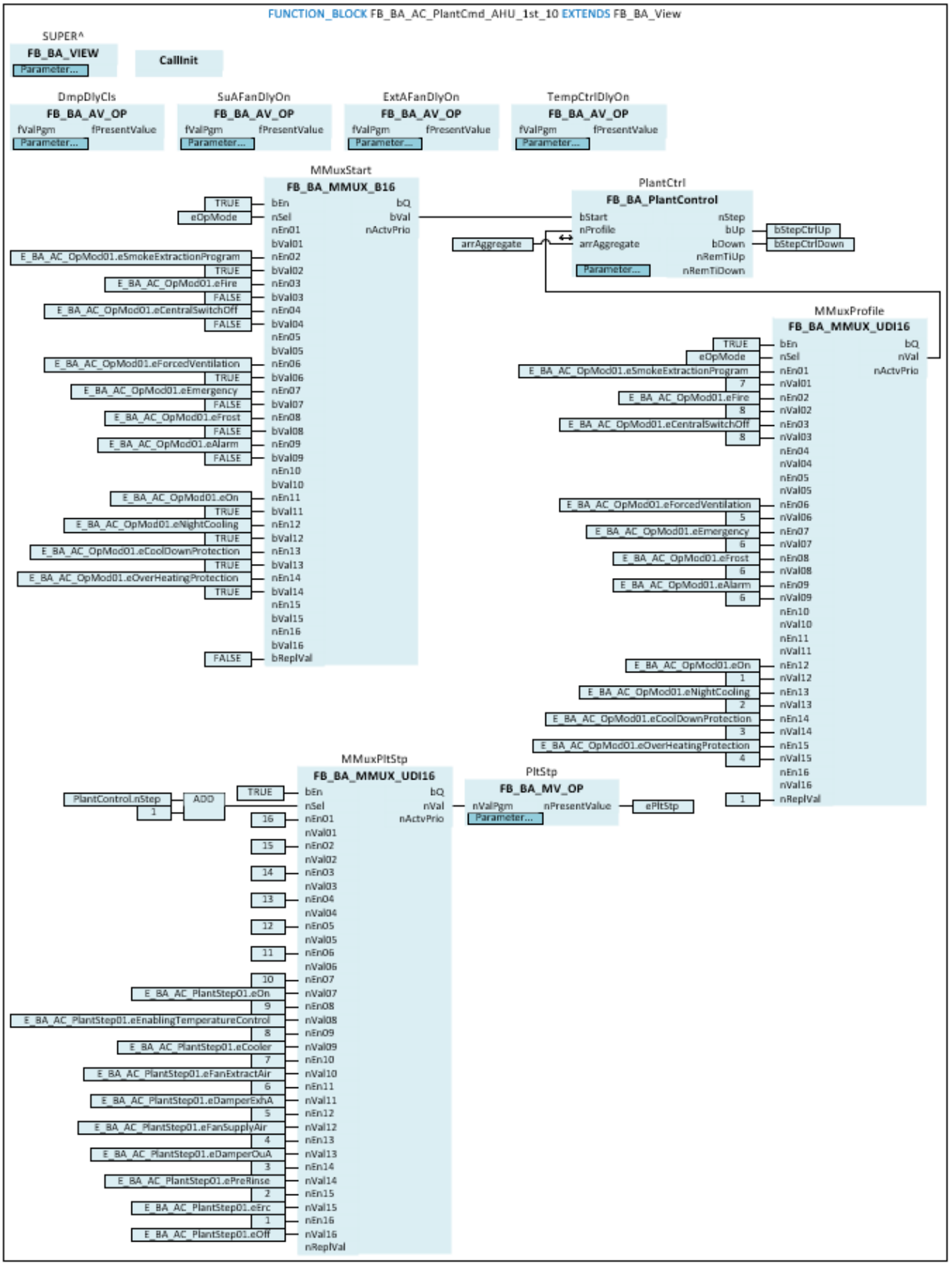
The corresponding switch-on or switch-off commands and their priority are determined for each aggregate according to the specified plant operation mode *eOpMode* and the information in the control matrix of the *PlantControl* unit.

The bidirectional communication to the aggregates of the air conditioning system takes place via the array *arrAggregate*, see Principle diagram plant.

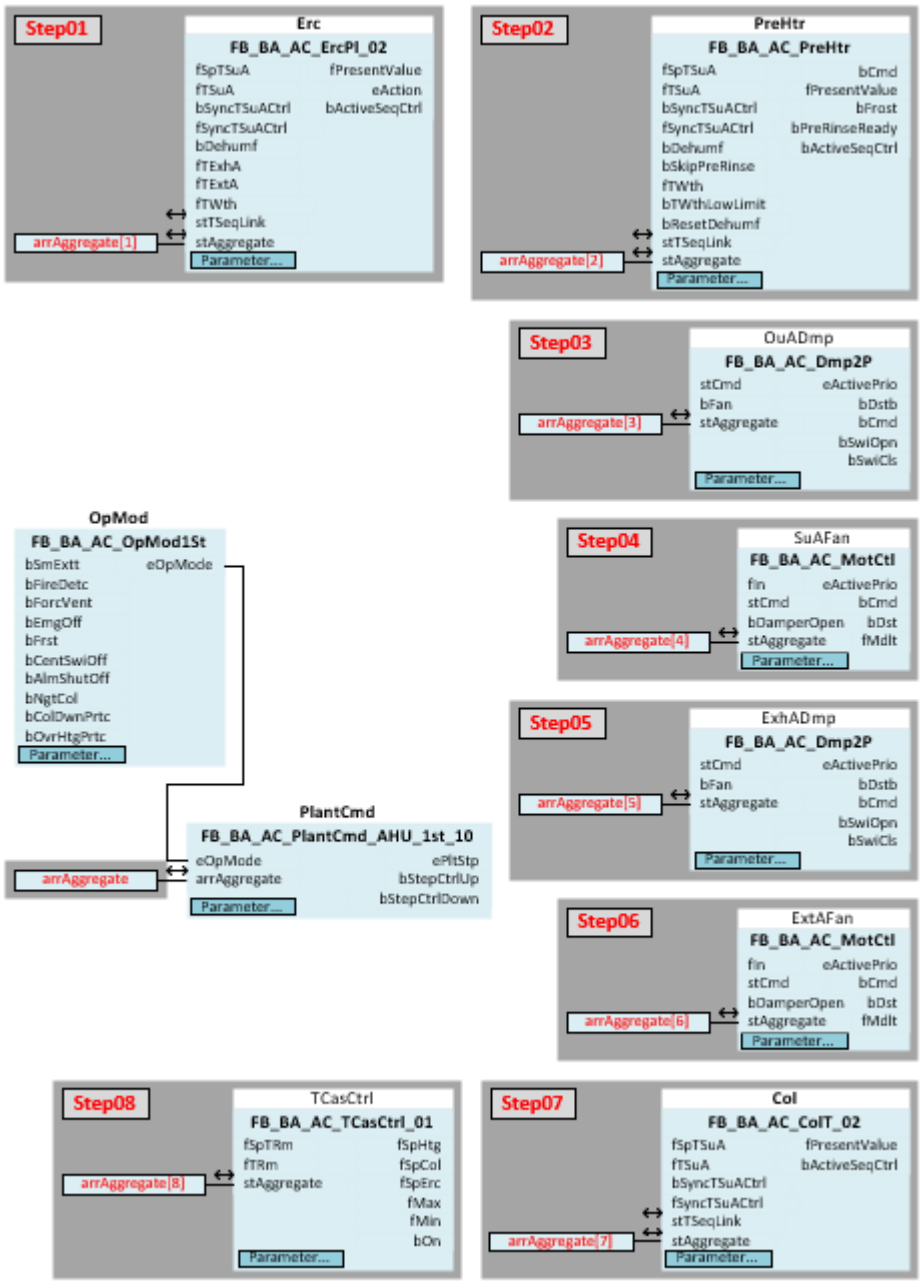
The parameterization of the control matrix is done in the methods below this template.

The initialization of the template takes place within the method FB_Init.

Block diagram



Principle diagram plant



Syntax

```
FUNCTION_BLOCK FB_BA_AC_PlantCmd_AHU_1st_10 EXTENDS FB_BA_View
VAR_INPUT
  eOpMode                : E_BA_AC_OpMod01;
END_VAR
VAR_OUTPUT
  ePltStp                : E_BA_AC_PlantStep01;
  bStepCtrlUp            : BOOL;
  bStepCtrlDown          : BOOL;
END_VAR
VAR_IN_OUT
  arrAggregate           : ARRAY [1..MAX(1,BA_Param.nMaxAggregate)] OF ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  DmpDlyCls              : FB_BA_AV_Op;
  SuAFanDlyOn            : FB_BA_AV_Op;
  ExtAFanDlyOn           : FB_BA_AV_Op;
  TempCtrlDlyOn          : FB_BA_AV_Op;
  PltStp                 : FB_BA_MV_Op;
  PlantControl           : FB_BA_PlantControl;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bInit                  : BOOL := TRUE;
```

```
END_VAR
VAR
  MMuxStart             : FB_BA_MMUX_B16;
  MMuxProfile           : FB_BA_MMUX_UDI16;
  MMuxPltStp            : FB_BA_MMUX_UDI16;
END_VAR
```

Inputs

Name	Type	Description
eOpMode	E_BA_AC_OpMod01 [▶ 699]	Input of the current operation mode *eOpMode*, see template FB_BA_AC_OpMod1St [▶ 763].

Outputs

Name	Type	Description
ePltStp	E_BA_AC_PlantStep01 [▶ 698]	Output of the current plant step of the air conditioning system.
bStepCtrlUp	BOOL	Indicates that the order of the step sequence control is in the rising state.
bStepCtrlDown	BOOL	Indicates that the order of the step sequence control is in the falling state.

Inputs/outputs

Name	Type	Description
arrAggregates	ARRAY [1..MAX(1,BA_Param.nMaxAggregate [▶ 284])] OF ST_BA_Aggregate [▶ 278]	The array contains a bidirectional data and command structure for each aggregate. When the step sequence control is switched up or down, the switching command determined for each aggregate is transferred via the bidirectional communication structure *arrAggregate*. Feedback from the aggregates for switching the step sequence control up or down is transmitted to the *PlantControl* via the communication structure.

Inputs CONSTANT

Name	Type	Description
DmpDlyCls	FB_BA_AV_Op [▶ 202]	AV object for entering the value to delay the closing of the dampers during shutdown of the air conditioning plant, to take into account the overrun time of the fans during plant shutdown.
SuAFanDlyOn	FB_BA_AV_Op [▶ 202]	AV object for entering a value for the start-up delay of the supply air fan when the air conditioning plant is switched on.
ExtAFanDlyOn	FB_BA_AV_Op [▶ 202]	AV object for entering a value for the start-up delay of the extract air fan when the air conditioning plant is switched on.
TempCtrlDlyOn	FB_BA_AV_Op [▶ 202]	AV object for entering a value for delayed temperature control enable.
PltStp	FB_BA_MV_Op [▶ 239]	The MV object shows the current step of the plant's step sequence control.
PlantControl	FB_BA_PlantControl [▶ 480]	The function block represents the higher-level plant control of a step sequence control of aggregates. The function block ensures that the individual aggregates of a plant are switched on or off sequentially one after the other in a certain order.

Inputs CONSTANT PERSISTENT

Name	Type	Description
bInit	BOOL	The variable is used to call the initialization methods for the control matrix in the CallInit [▶ 790] method. After starting the controller, the variable is set to FALSE after a PLC cycle and this state is saved persistently.

Variables

Name	Type	Description
MMuxStart	FB_BA_MMUX_B16 [▶ 428]	The multiplexer uses the plant operation mode *eOpMode* to define whether the air conditioning system is started or stopped.
MMuxProfile	FB_BA_MMUX_UDI16 [▶ 428]	The multiplexer uses the plant operation mode *eOpMode* to define the control profile of the *PlantControl*.
MMuxPltStp	FB_BA_MMUX_UDI16 [▶ 428]	The multiplexer evaluates the steps of the step sequence control and transfers them to the MV object *PltStp* for display at management level.

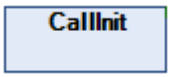
Methods

Name	Definition location	Description
CallInit [▶ 790]	Local	Calling the initialization methods for parameterization of the control matrix of the plant control unit.
ProfileDescription [▶ 791]	Local	Used to initialize the parameter array *arrProfileDescription* of the plant control unit.
ProfileParameter0x [▶ 791]	Local	Initialization of a part of the multidimensional profile parameter array *arrProfileParameter* of the control unit.
StepDescription [▶ 792]	Local	Used to initialize the parameter array *arrStepDescription* of the plant control unit.

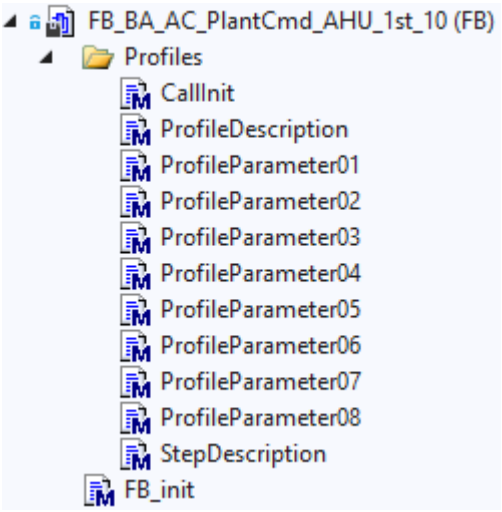
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.3.7.1 CallInit



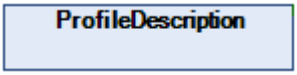
The method is used to call the initialization methods for the parameterization of the control matrix of the *PlantControl* unit.



Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.3.7.2 ProfileDescription

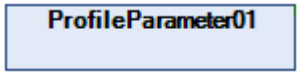


The method is used to initialize the parameter array *arrProfileDescription* of the *PlantControl* unit. Each element of the array represents a profile. Accordingly, the names of the profiles are assigned in this method and the associated priority *nPrio* is defined.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.3.7.3 ProfileParameter0x



The method is used to initialize a part of the multidimensional profile parameter array *arrProfileParameter* of the PlantControl [▶ 480] unit.

This description is representative of all eight methods of this type.

The first element of the array of the ProfileParameter01 method represents the "Auto" plant profile.

The first element of the array of the ProfileParameter02 method represents the "NightCooling" plant profile.

The first element of the array of the ProfileParameter03 method represents the "CoolDownProtection" plant profile.

The first element of the array of the ProfileParameter04 method represents the "OverHeatingProtection" plant profile.

The first element of the array of the ProfileParameter05 method represents the "ForcedVentilation" plant profile.

The first element of the array of the ProfileParameter06 method represents the "CriticalOff" plant profile.

The first element of the array of the ProfileParameter07 method represents the "SmokeExtractionProgram" plant profile.

The first element of the array of the ProfileParameter08 method represents the "LifeSafetyOff" plant profile.

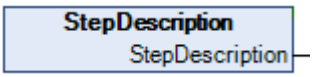
The second element stands for the steps within the step sequence control.

The switching values for the aggregates (*bValue*, *nValue*, *fValue*), the delay times for the step sequence control (*fDelayUp*, *fDelayDown*) and the releases for the use of the process feedback of the aggregates in the step (*bEnFdbUp*, *bEnFdbDown*) are parameterized.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.3.7.4 StepDescription



The method is used to initialize the parameter array *arrStepDescription* of the *PlantControl* unit.

The plain text of the plant steps is defined in the parameter array. This information is purely for display purposes.

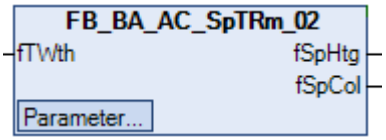
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.4 Setpoint

6.1.4.2.1.1.4.1 Temperature

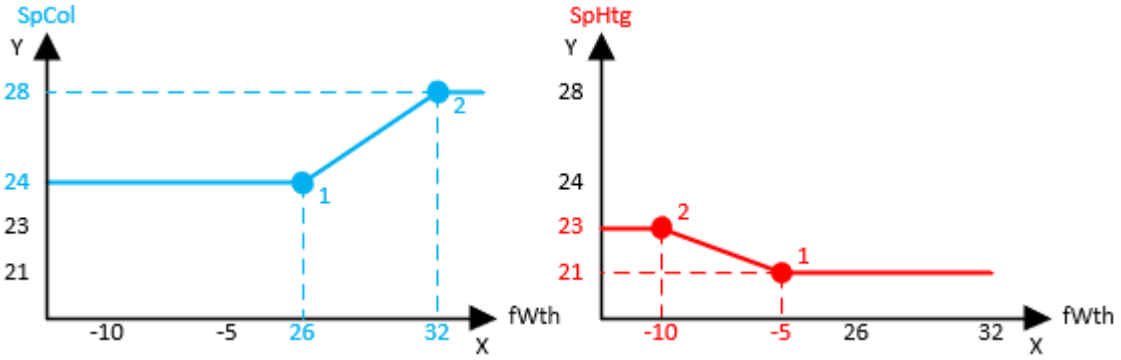
6.1.4.2.1.1.4.1.1 FB_BA_AC_SpTRm_02



The template is a setpoint program for an air conditioning system.

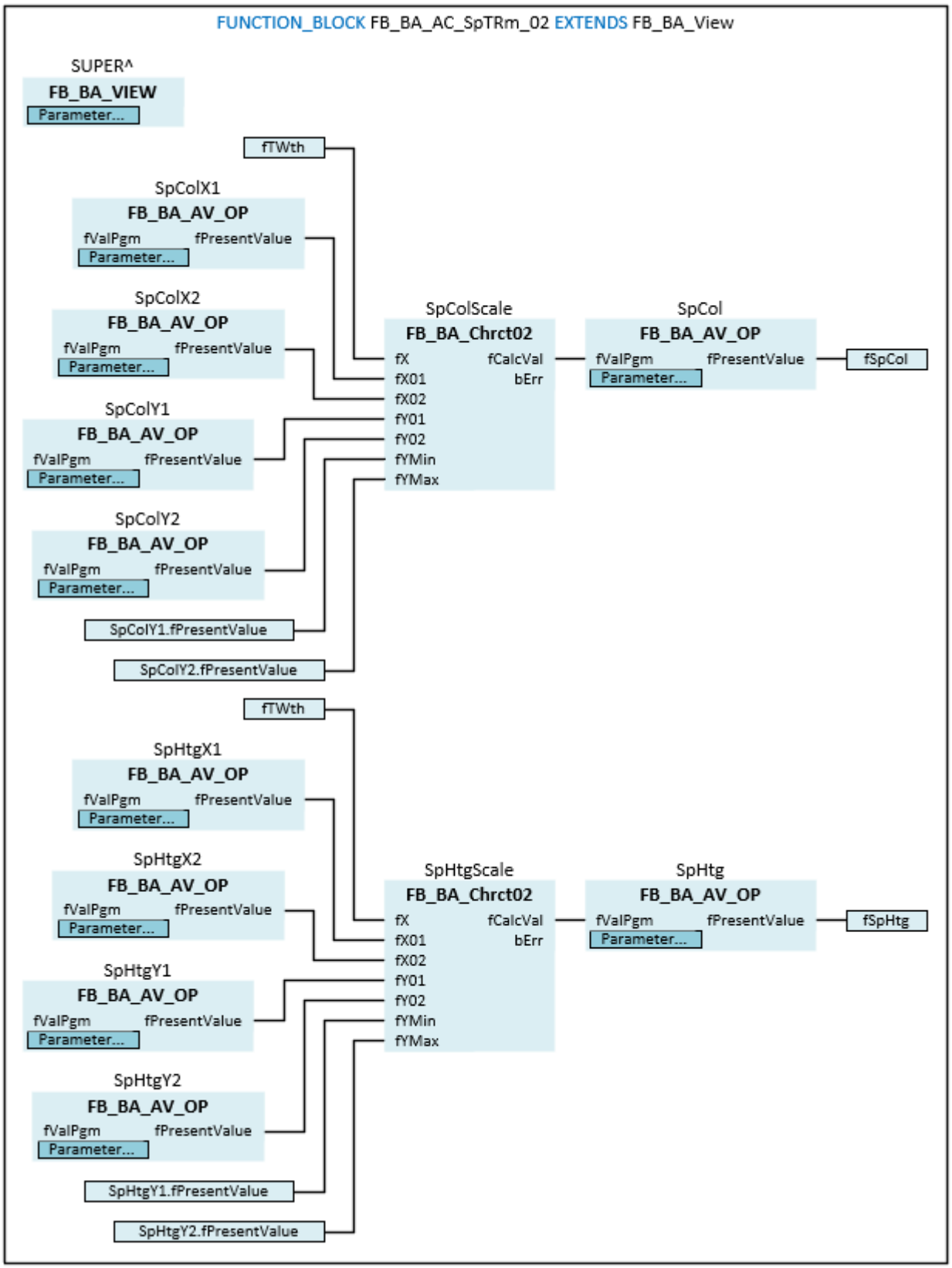
It serves as a setpoint program for an extract air/supply air cascade with a separate room temperature setpoint for heating and cooling mode, including summer and winter compensation.

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).

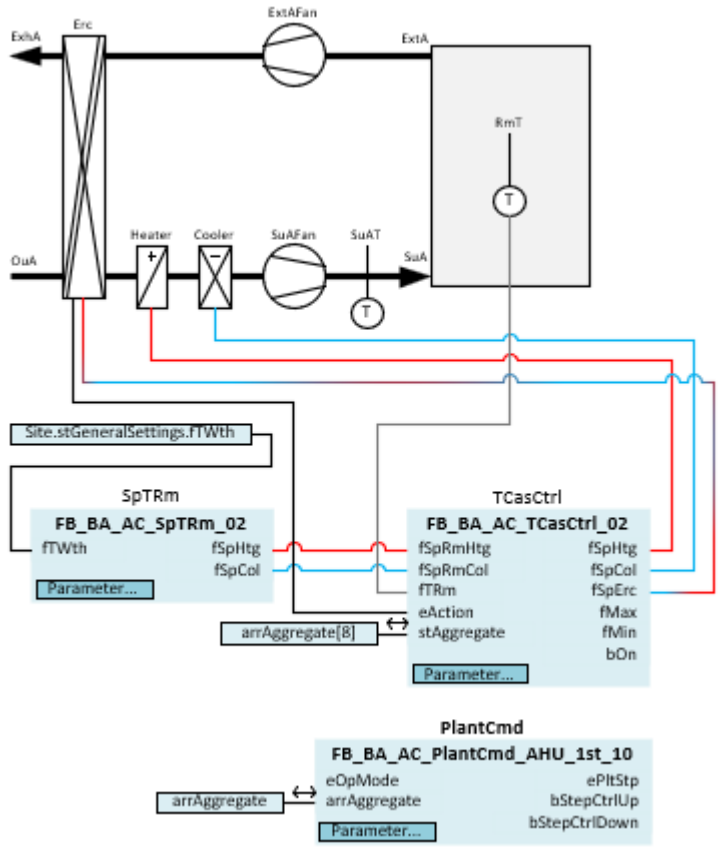


The initialization of the template takes place within the method FB_Init.

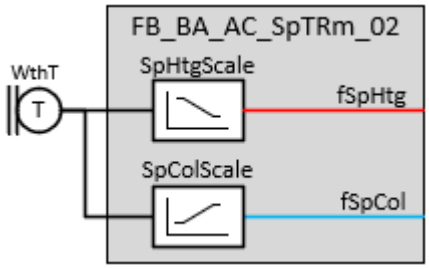
Block diagram



Principle diagram01



Principle diagram02



Syntax

```
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_Chrct02;
  SpHtgScale    : FB_BA_Chrct02;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
fTWth	REAL	Current value of the outside temperature.

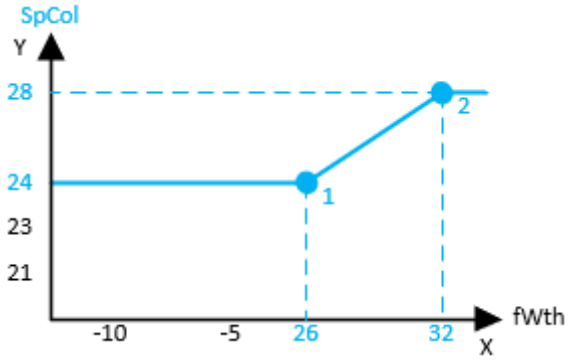
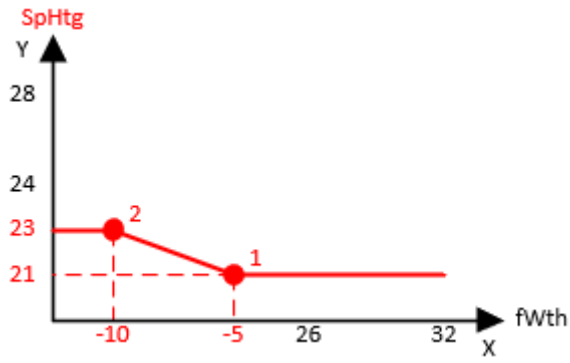
Outputs

Name	Type	Description
fSpHtg	REAL	Calculated setpoint of the supply air temperature for the heater.
fSpCol	REAL	Calculated setpoint of the supply air temperature for the cooler.

Inputs CONSTANT

Name	Type	Description
SpColX1	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X1 of the summer compensation for the cooling setpoint.
SpColX2	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X2 of the summer compensation for the cooling setpoint.
SpColY1	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point Y1 of the summer compensation for the cooling setpoint. *SpColY1* is the base cooling setpoint.
SpColY2	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point Y2 of the summer compensation for the cooling setpoint.
SpHtgX1	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X1 of the winter compensation for the heating setpoint.
SpHtgX2	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X2 of the winter compensation for the heating setpoint.
SpHtgY1	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point Y1 of the winter compensation for the heating setpoint. *SpHtgY1* is the base heating setpoint.
SpHtgY2	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point Y2 of the winter compensation for the heating setpoint.
SpHtg	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated room temperature setpoint heating.
SpCol	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated room temperature setpoint cooling.

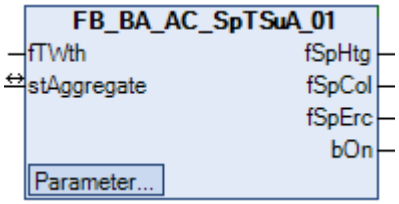
Variables

Name	Type	Description
SpColScale	FB_BA_Chrct02 [▶ 456]	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_Chrct02 [▶ 456]	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 v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.1.4.1.2 FB_BA_AC_SpTSuA_01

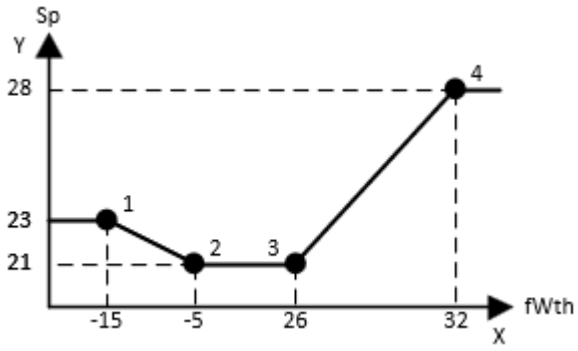


The template generates the supply air temperature setpoint of an air conditioning system, including summer and winter compensation.

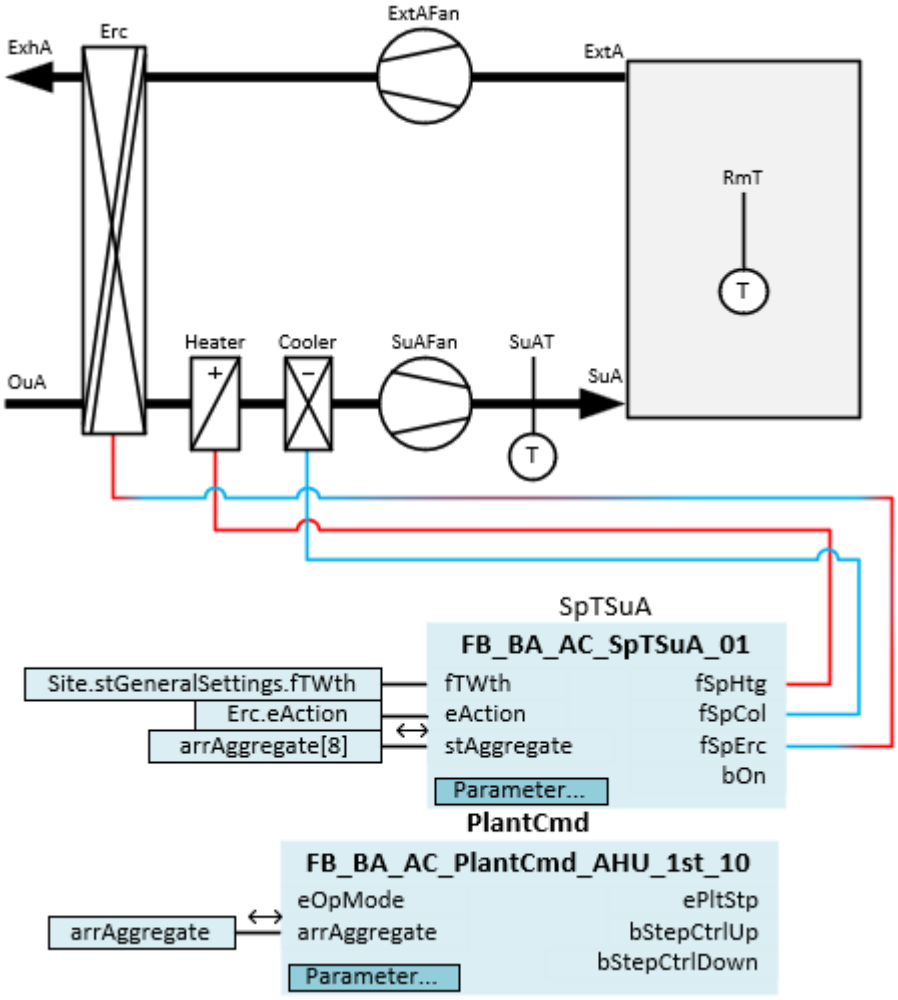
The supply air temperature setpoint is determined by the function block Scale [▶ 459] depending on the outside temperature.

The supply air temperature sequence control is enabled with the variable *bOn* by evaluating the receive block of the step sequence control *Aggregate*.

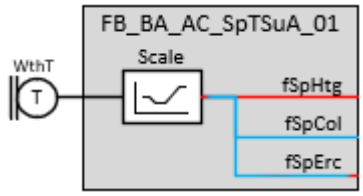
The setpoint linked to the variables *fSpHtg*, *fSpCol* and *fSpErc* is transferred to the supply air controllers of the aggregates heater, cooler and energy recovery.



Principle diagram 01

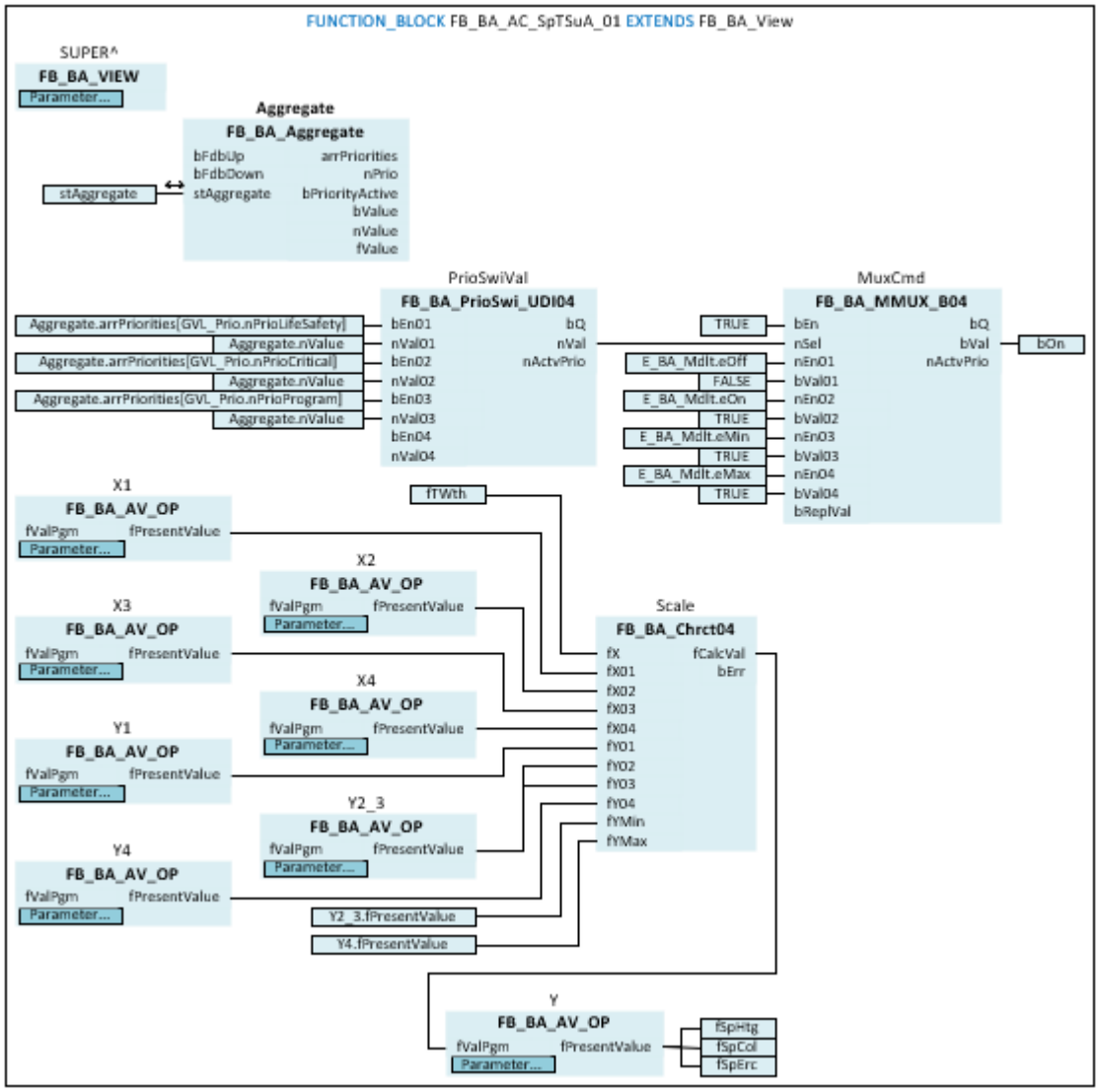


Principle diagram 02



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_SpTSuA_01 EXTENDS FB_BA_View
VAR_INPUT
  fTWth          : REAL;
END_VAR
VAR_OUTPUT
  fSpHtg         : REAL;
  fSpCol         : REAL;
  fSpErc         : REAL;
  bOn            : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate    : ST_BA_Aggregate;
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;
  Sp             : FB_BA_AV_Op;
  Aggregate      : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiVal     : FB_BA_PrioSwi_UDI04;
  MuxCmd         : FB_BA_MMUX_B04;
  Scale          : FB_BA_Chrct04;
END_VAR
```

Inputs

Name	Type	Description
fTWth	REAL	Current value of the outside temperature.

Outputs

Name	Type	Description
fSpHtg	REAL	Calculated setpoint of the supply air temperature for the heater.
fSpCol	REAL	Calculated setpoint of the supply air temperature for the cooler.
fSpErc	REAL	Calculated setpoint of the supply air temperature. for energy recovery.
bOn	BOOL	The output shows the resulting result of the multiplexer *MuxCmd*. This is further processed within the ventilation system and is used to enable the supply air temperature sequence control.

Inputs/outputs

Name	Type	Description
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
X1	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X2.
X3	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X3.
X4	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point X4.
Y1	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point Y1.
Y2_3	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point Y2/Y3.
Y4	FB_BA_AV_Op [▶ 202]	Input of the value for interpolation point Y4.
Sp	FB_BA_AV_Op [▶ 202]	Output of the calculated, simple supply air temperature setpoint. This is output via the *fSpHtg*, *fSpCol* and *fSpErc* outputs.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

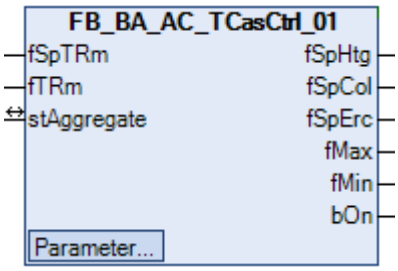
Variables

Name	Type	Description
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch PrioSwiVal [▶ 433] uses the command structure *stCmd* to determine the modulation command for the multiplexers *MuxCmd* and *MuxMdlt*.
MuxCmd	FB_BA_MMUX_B04 [▶ 428]	The multiplexer MuxCmd [▶ 428] determines the current switch value from the command of the priority switch *PrioSwiVal*. The resulting output is sent to the binary output object *bOn*.
Scale	FB_BA_Chrct04 [▶ 459]	The function block calculates the supply air temperature setpoint depending on the outside temperature.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.4.1.3 FB_BA_AC_TCasCtrl_01

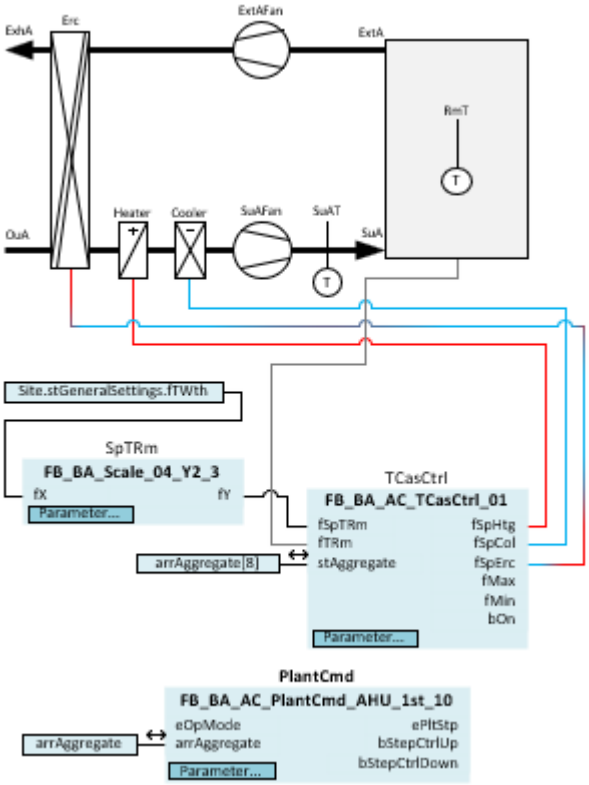


The template is used for room temperature control by means of room-supply air cascade.

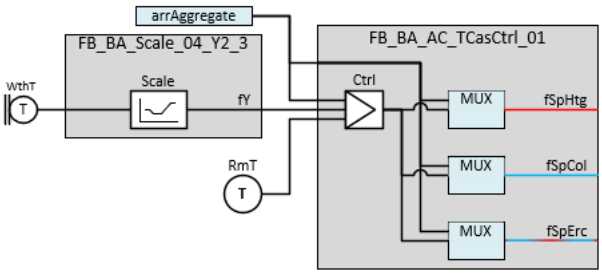
It consists of a master controller for calculating the supply air temperature setpoint for the aggregates heater, cooler and energy recovery.

The supply air temperature sequence control is enabled with the variable *bOn* by evaluating the receive block of the step sequence control *Aggregate*.

Principle diagram01

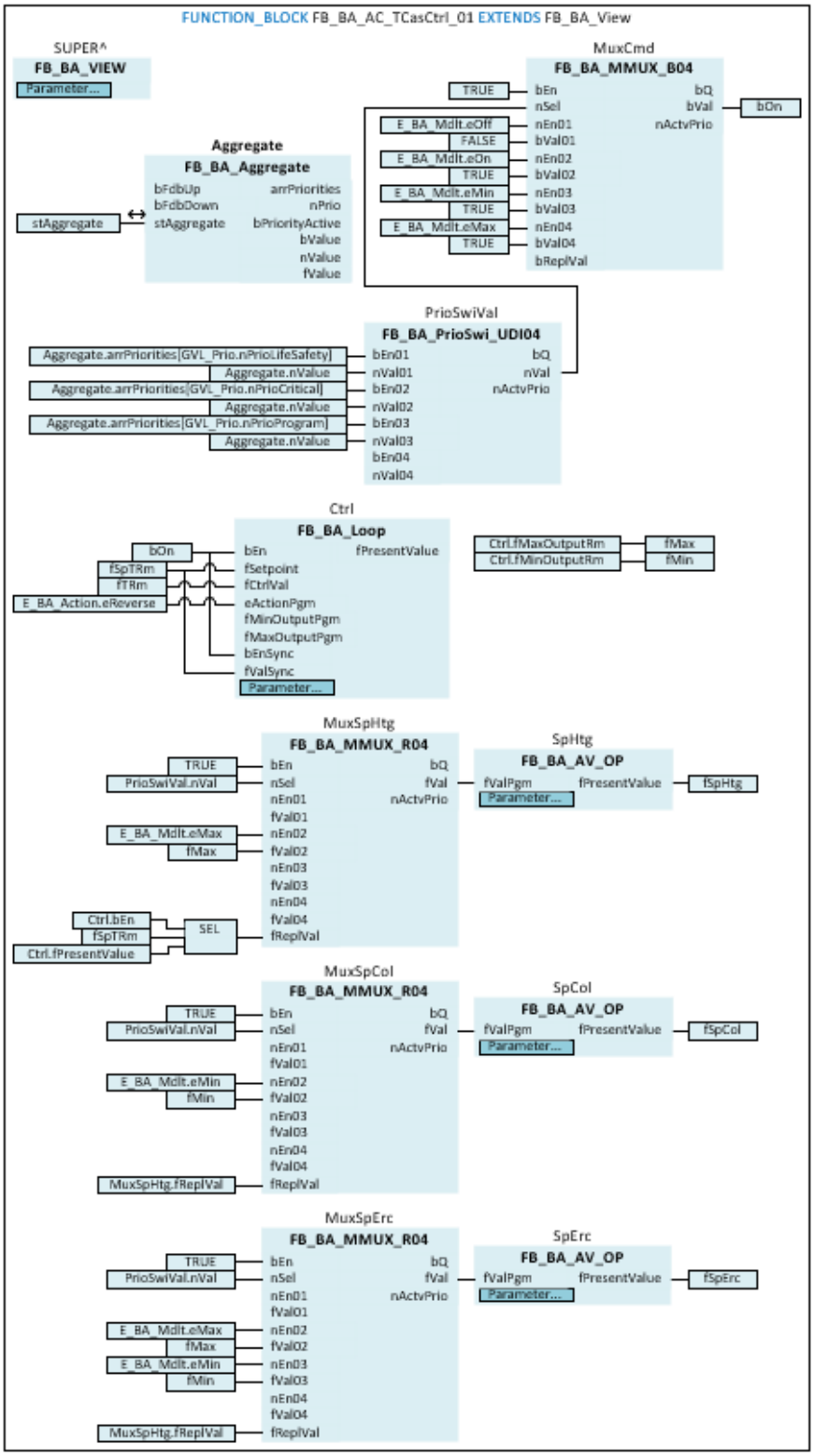


Principle diagram02



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AC_TCasCtrl_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTRm          : REAL;
  fTRm            : REAL;
END_VAR
VAR_OUTPUT
  fSpHtg          : REAL;
  fSpCol          : REAL;
  fSpErc          : REAL;
  fMax            : REAL;
  fMin            : REAL;
```

```
  bOn           : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate   : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  SpHtg         : FB_BA_AV_Op;
  SpCol         : FB_BA_AV_Op;
  SpErc         : FB_BA_AV_Op;
  Ctrl          : FB_BA_Loop;
  Aggregate     : FB_BA_Aggregate;
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 *Ctrl*. If no room temperature is available, the outlet air temperature of a ventilation system can be used as control value.

Outputs

Name	Type	Description
fSpHtg	REAL	Calculated setpoint of the supply air temperature for the heater.
fSpCol	REAL	Calculated setpoint of the supply air temperature for the cooler.
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 *MuxCmd*. This is further processed within the ventilation system and is used to enable the supply air temperature sequence control.

Inputs/outputs

Name	Type	Description
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
SpHtg	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated room temperature setpoint heating.
SpCol	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated room temperature setpoint cooling.
SpErc	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated supply air temperature setpoint Energy recovery.
Ctrl	FB_BA_Loop [▶ 220]	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.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

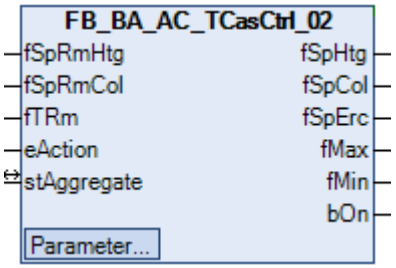
Variables

Name	Type	Description
PrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch uses the command structure *stCmd* to determine the modulation command for the multiplexer *MuxCmd*.
MuxCmd	FB_BA_MMUX_B04 [▶ 428]	The multiplexer determines the current switching command from the modulation command of the priority switch *PrioSwiVal*. The resulting output is sent to the output *bOn*. The multiplexer defines the enabling conditions of the PID master controller *Ctrl*.
MuxSpHtg	FB_BA_MMUX_R04 [▶ 428]	The multiplexer defines the setpoint heating *fSpHtg* depending on the priority switch *PrioSwiVal*.
MuxSpCol	FB_BA_MMUX_R04 [▶ 428]	The multiplexer defines the cooling setpoint *fSpCol* depending on the priority switch *PrioSwiVal*.
MuxSpErc	FB_BA_MMUX_R04 [▶ 428]	The multiplexer defines the setpoint energy recovery *fSpErc* depending on the priority switch *PrioSwiVal*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.4.1.4 FB_BA_AC_TCasCtrl_02



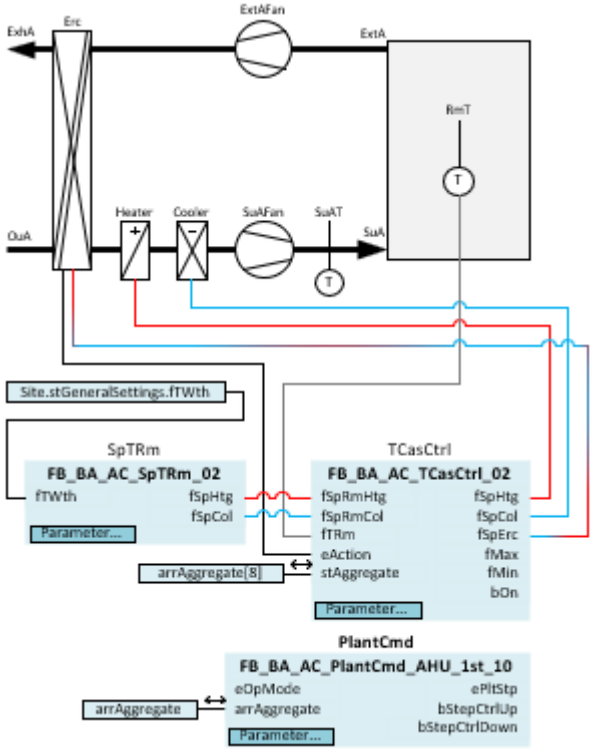
The template is used for room temperature control by means of room-supply air cascade.

It consists of two master controllers for calculating the supply air temperature setpoints for the heater and cooler aggregates. Each of the two master controllers receives a separate setpoint. The setpoint of the master controller for the heating function is below the setpoint of the master controller for cooling. There is an energy-neutral zone between the two setpoints.

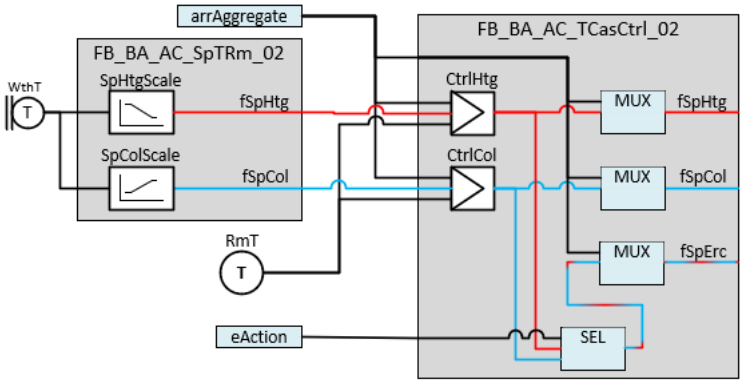
In addition, it determines the supply air temperature setpoint for energy recovery depending on its control direction.

The supply air temperature sequence control is enabled with the variable *bOn* by evaluating the receive block of the step sequence control *Aggregate*.

Principle diagram01

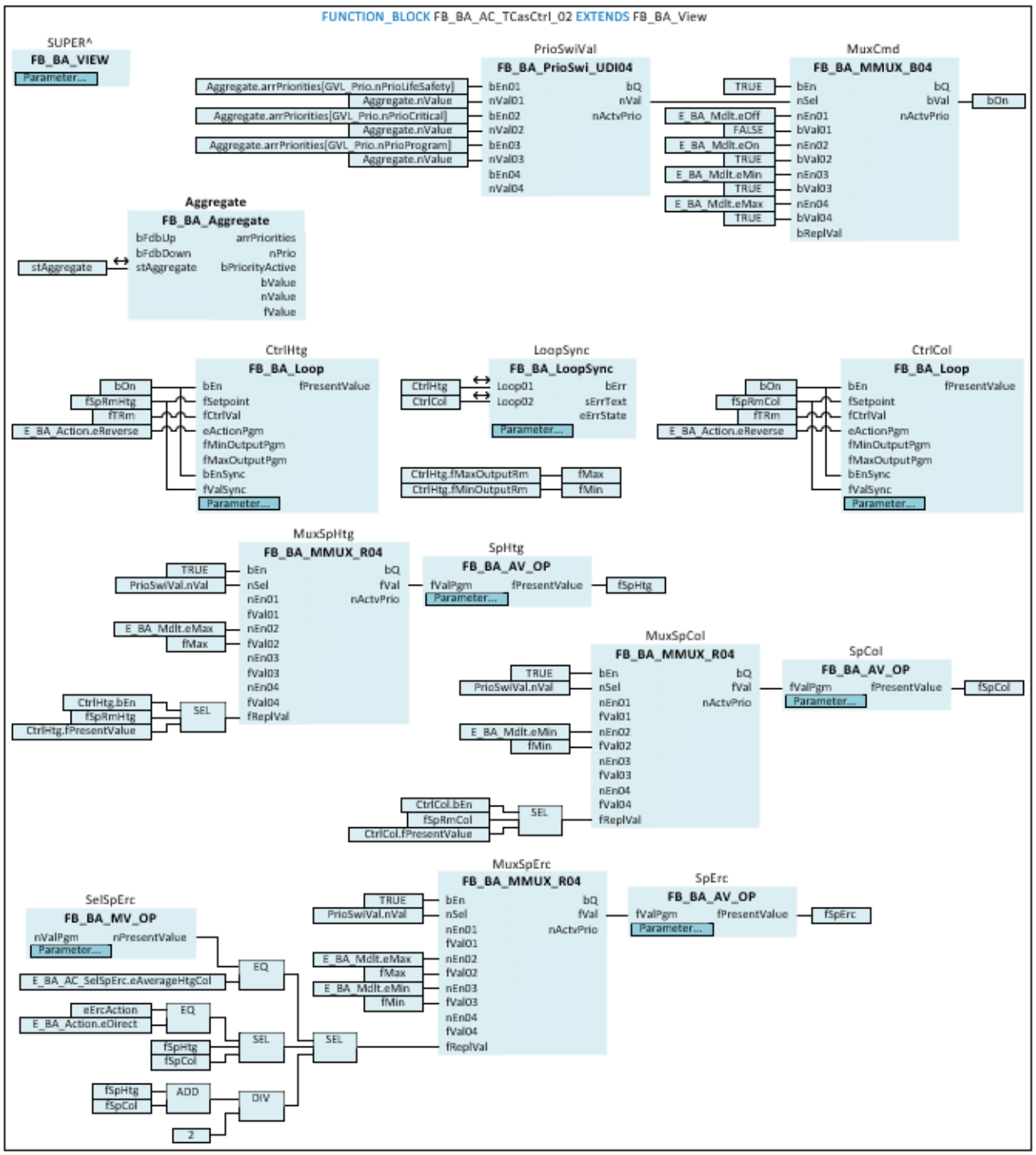


Principle diagram02



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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;
END_VAR
VAR_OUTPUT
  fSpHtg        : REAL;
  fSpCol        : REAL;
  fSpErc        : REAL;
  fMax          : REAL;
  fMin          : REAL;
  bOn           : BOOL;
END_VAR
VAR_IN_OUT
```

```
  stAggregate    : ST_BA_Aggregate;
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;
  Aggregate      : FB_BA_Aggregate;
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

Name	Type	Description
fSpRmHtg	REAL	Input variable for the setpoint room temperature Heating [▶ 792].
fSpRmCol	REAL	Input variable for the setpoint room temperature Cooling. [▶ 792]
fTRm	REAL	Input variable to which the room temperature is applied. The room temperature is the control value of the PID master controller *Ctrl*. 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.

Outputs

Name	Type	Description
fSpHtg	REAL	Calculated setpoint of the supply air temperature for the heater.
fSpCol	REAL	Calculated setpoint of the supply air temperature for the cooler.
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 *MuxCmd*. This is further processed within the ventilation system and is used to enable the supply air temperature sequence control.

Inputs/outputs

Name	Type	Description
stAggregate	ST_BA_Aggregate [▶ 278]	Bidirectional aggregate structure via which the enables and switching values are transmitted to the evaluation function block of the step sequence control *Aggregate*. The aggregate structure transmits the recorded feedback signals up and down to the central plant control.

Inputs CONSTANT

Name	Type	Description
SpHtg	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated room temperature setpoint heating.
SpCol	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated room temperature setpoint cooling.
SpErc	FB_BA_AV_Op [▶ 202]	Analog value object for displaying the calculated supply air temperature setpoint Energy recovery.
SelSpErc	FB_BA_MV_Op [▶ 239]	The Multistate value object is used to select the strategy setpoint energy recovery. E_BA_AC_SelSpErc.eAction [▶ 699]: = 1: Depends on the control direction, defined by input *eErcAction*. E_BA_AC_SelSpErc.eAverageHtgCol [▶ 699]:= 2: Average of input variables *fSpHtg* and *fSpCol*.
CtrlHtg	FB_BA_Loop [▶ 220]	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 [▶ 220]	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 [▶ 165]	The function block synchronizes the parameters for the two master controllers *CtrlHtg/CtrlCol*.
Aggregates	FB_BA_Aggregate [▶ 478]	The aggregate function block represents a receive block of a step sequence controller. Within the function block, the aggregate structure *stAggregate* is evaluated and integrated into the template using the inputs and outputs.

Variables

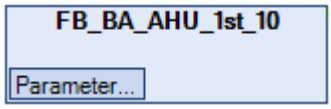
Name	Type	Description
PrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch uses the command structure *stCmd* to determine the modulation command for the multiplexer *MuxCmd*.
MuxCmd	FB_BA_MMUX_B04 [▶ 428]	The multiplexer determines the current switching command from the modulation command of the priority switch *PrioSwiVal*. The resulting output is sent to the output *bOn*. The multiplexer defines the enabling conditions of the PID master controller *Ctrl*.
MuxSpHtg	FB_BA_MMUX_R04 [▶ 428]	The multiplexer defines the setpoint heating *fSpHtg* depending on the priority switch *PrioSwiVal*.
MuxSpCol	FB_BA_MMUX_R04 [▶ 428]	The multiplexer defines the cooling setpoint *fSpCol* depending on the priority switch *PrioSwiVal*.
MuxSpErc	FB_BA_MMUX_R04 [▶ 428]	The multiplexer defines the setpoint energy recovery *fSpErc* depending on the priority switch *PrioSwiVal*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.1.5 Plants

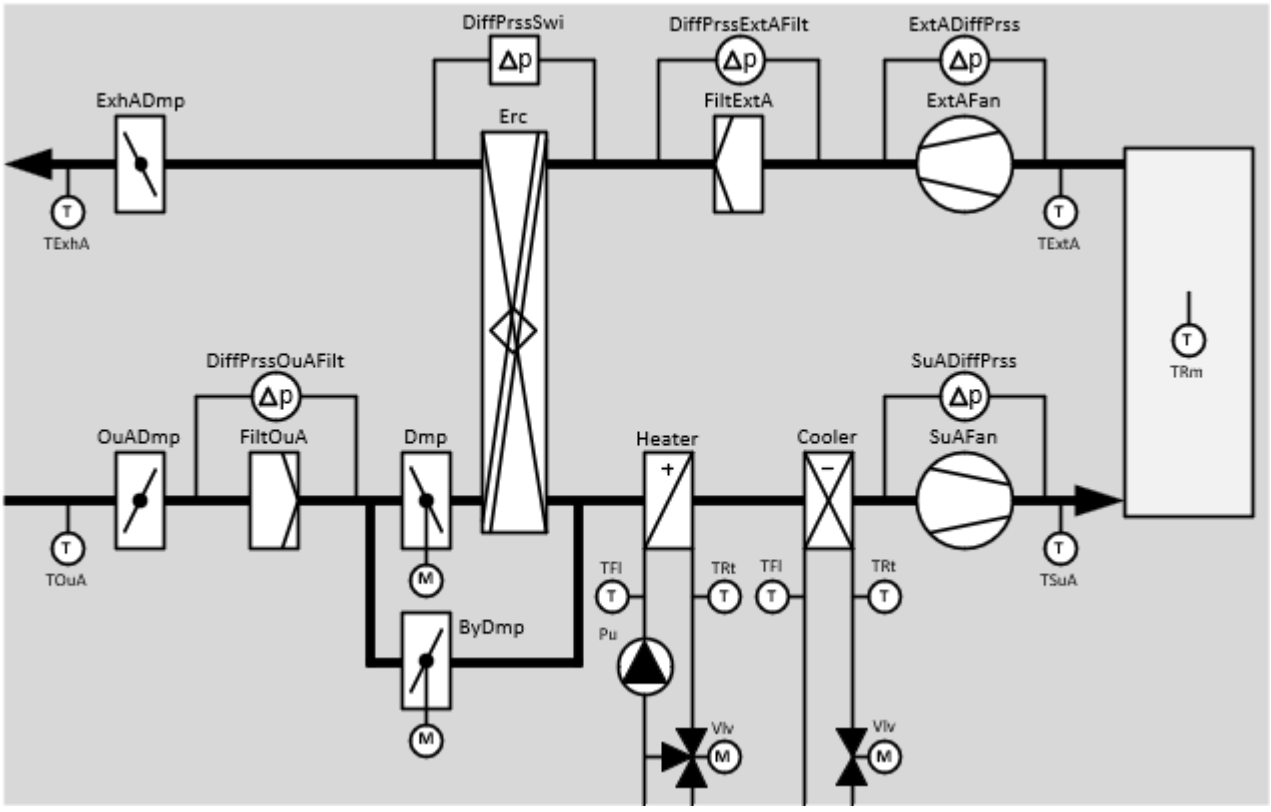
6.1.4.2.1.1.5.1 FB_BA_AHU_1st_10



Template of an air conditioning system.

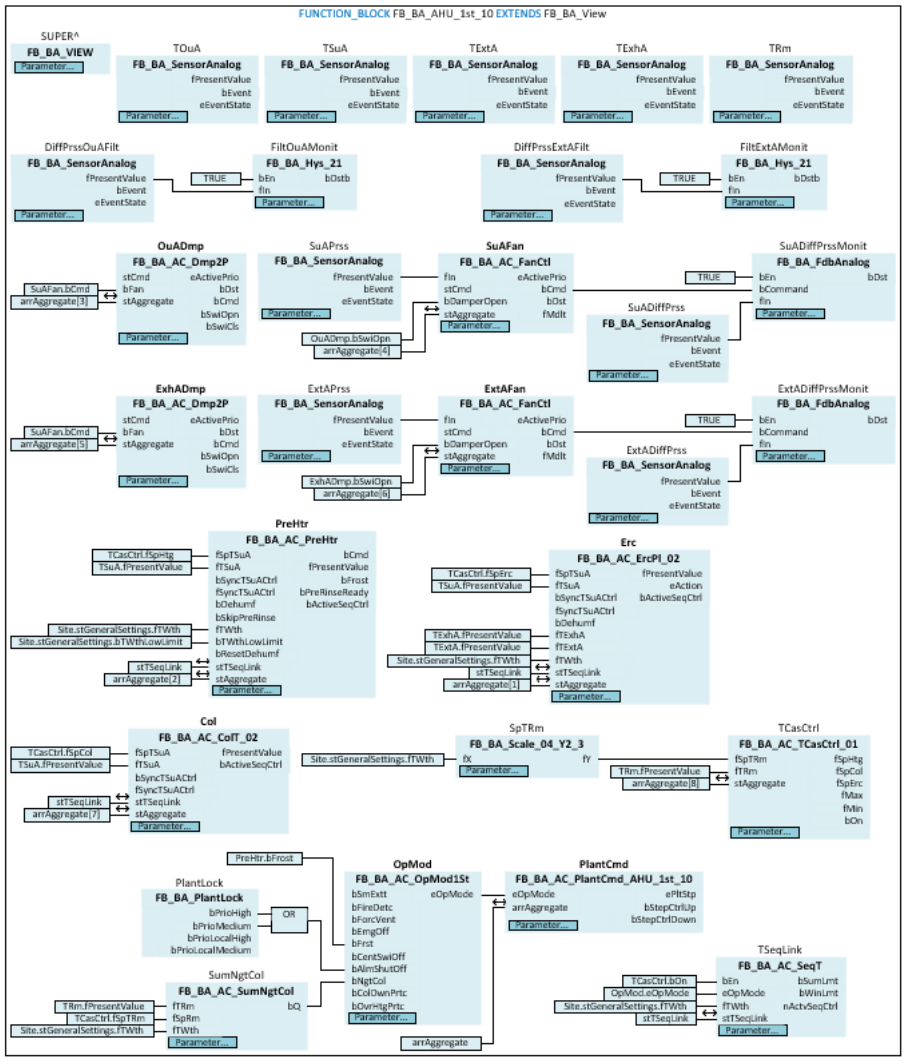
The main components of the template are:

- Plant control and start program
- Supply air temperature sequence control
- Room temperature cascade control
- Summer night cooling
- Pressure control of the supply and extract air fan
- Thermal air treatment using air heaters, coolers and energy recovery by means of plate heat exchangers
- External and exhaust air damper with spring return actuator and end position monitor
- External and extract air filters with analog differential pressure monitoring

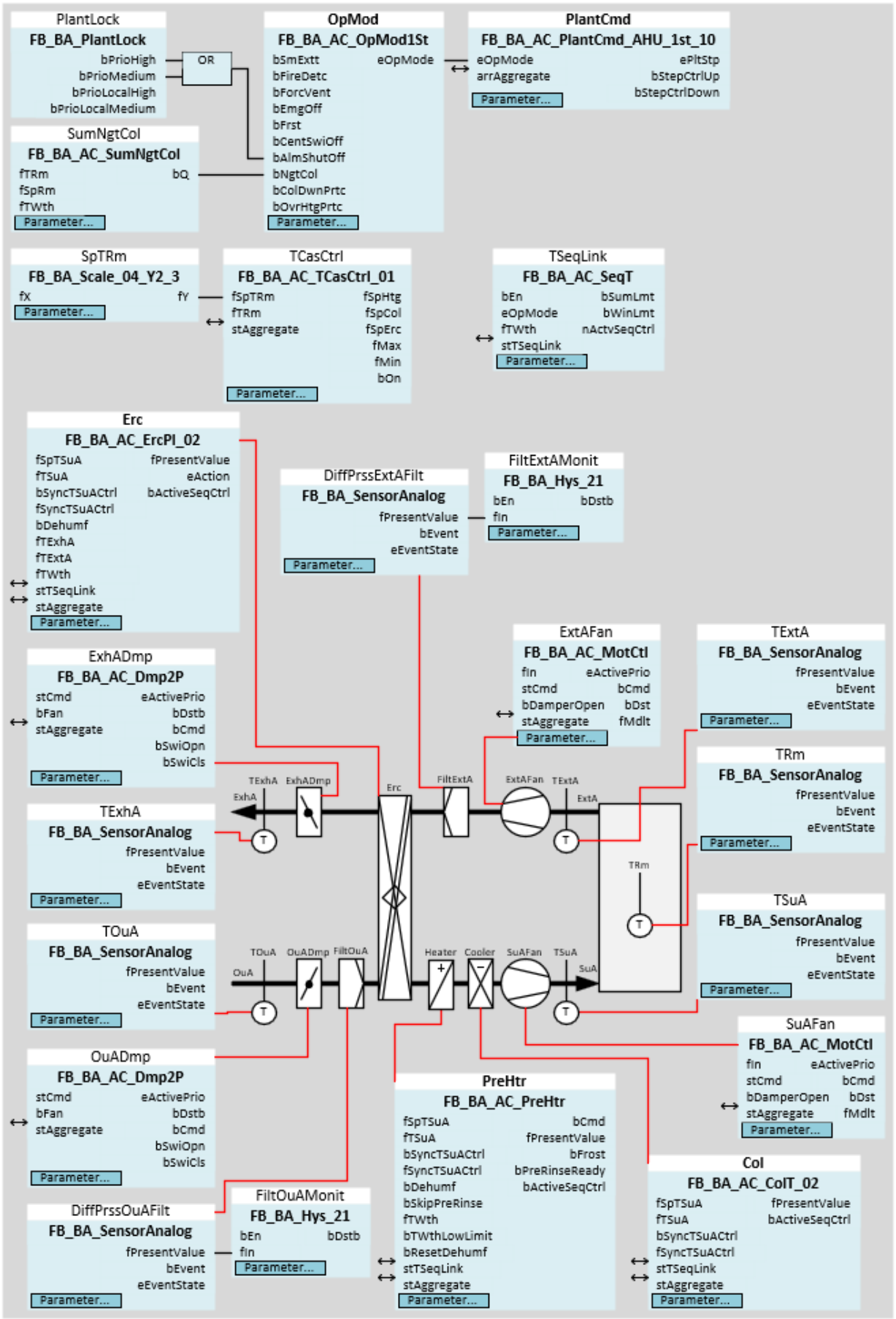


The initialization of the template takes place within the method FB_Init.

Block diagram



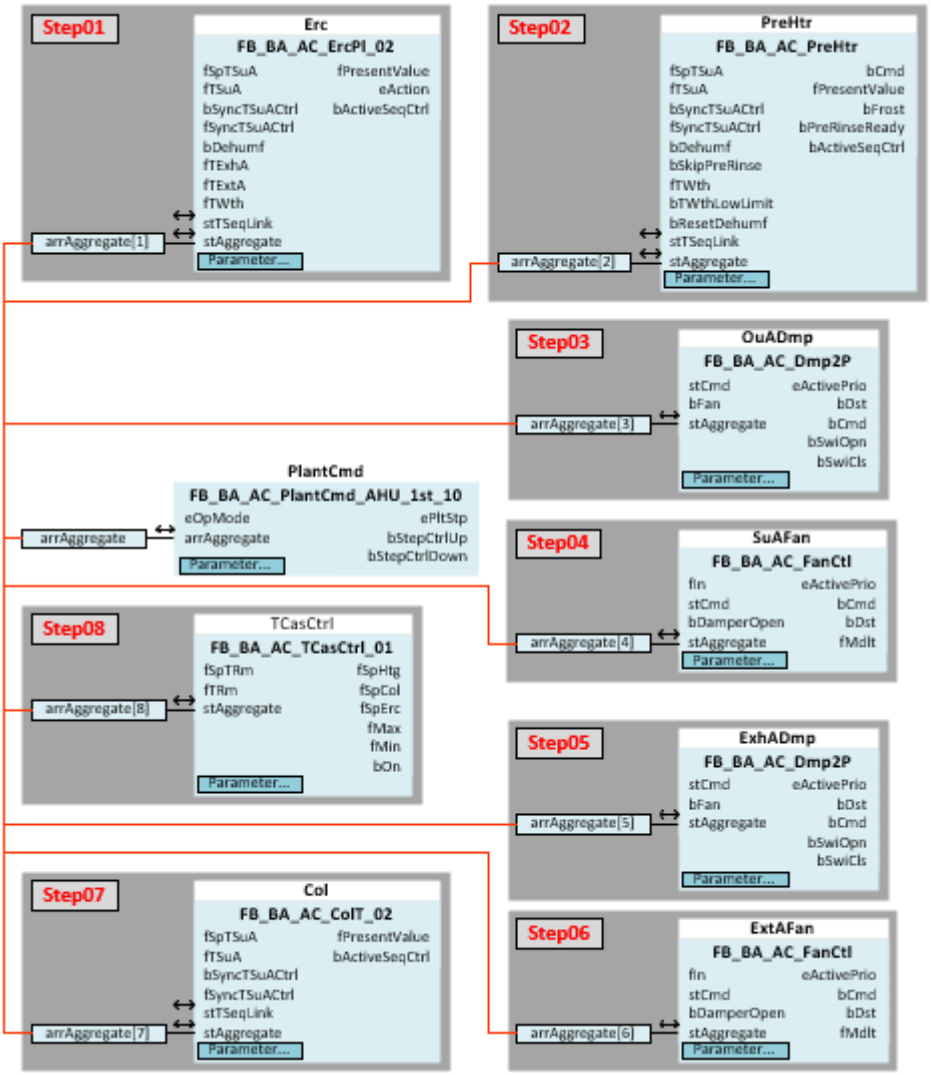
Plant diagram



Switch-on sequence of the aggregates of the plant

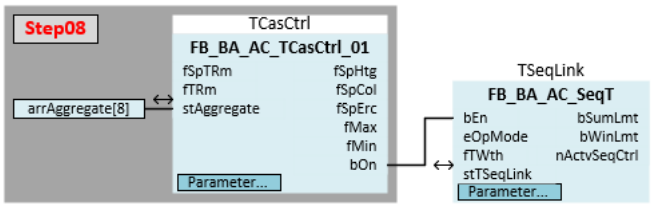
Within *PlantCmd*, a step sequence controller sequentially controls the switching on and off of the aggregates one after the other. When switching from aggregate to aggregate, predefined delay times and feedback from the aggregates are taken into account.

The order of the aggregates is determined by the ordinal number of the individual elements of the bidirectional array *arrAggregate*.



Step	Stages of plant start-up	Aggregate	Function blocks
00	E_BA_AC_PlantStep01.eOff	Plant Off	
01	E_BA_AC_PlantStep01.eErc	Energy recovery	*Erc*
02	E_BA_AC_PlantStep01.ePreRinse	Prerinsing of the hot water air heater	*PreHtr*
03	E_BA_AC_PlantStep01.eDamperOuA	Outside air damper	*OuADmp*
04	E_BA_AC_PlantStep01.eFanSupplyAir	Supply air fan	*SuAFan*
05	E_BA_AC_PlantStep01.eDamperExhA	Exhaust air damper	*ExhADmp*
06	E_BA_AC_PlantStep01.eFanExtractAir	Extract air fan	*ExtAFan*
07	E_BA_AC_PlantStep01.eCooler	Cold water air cooler	*Col*
08	E_BA_AC_PlantStep01.eOn	Room supply air cascade, plant On	*TCasCtrl*

If the plant has reached step 08 *E_BA_AC_PlantStep01.eOn*, the supply air temperature sequence control *TSeqLink* is enabled when the cascade control *TCasCtrl* is switched on.



Starting up the plant in cold weather

When the system is switched on in cold weather, the step sequence control of the *PlantCmd* command includes the plant step PreRinse of the preheater *E_BA_AC_PlantStep01.ePreRinse*. In this step, the return temperature control of the hot water air heater is activated. This procedure is intended to prevent the freezing of the hot water air heater during the plant start-up.

Fault shutdown

The plant switches off at:

- Fire alarm
- Frost alarm

In the event of a frost alarm, the *PreHtr* hot water air heater remains in the frost protection position even when the plant is switched off!

- Fan faults (feedback monitoring, thermoelectric overload, maintenance switch)
- Incorrect position of the outside and exhaust air damper.
- Filter blockages

The following table lists the event-capable objects of the template that can trigger relevant *PlantLock* faults. The parameterization of these events can be found in *FB_init*.

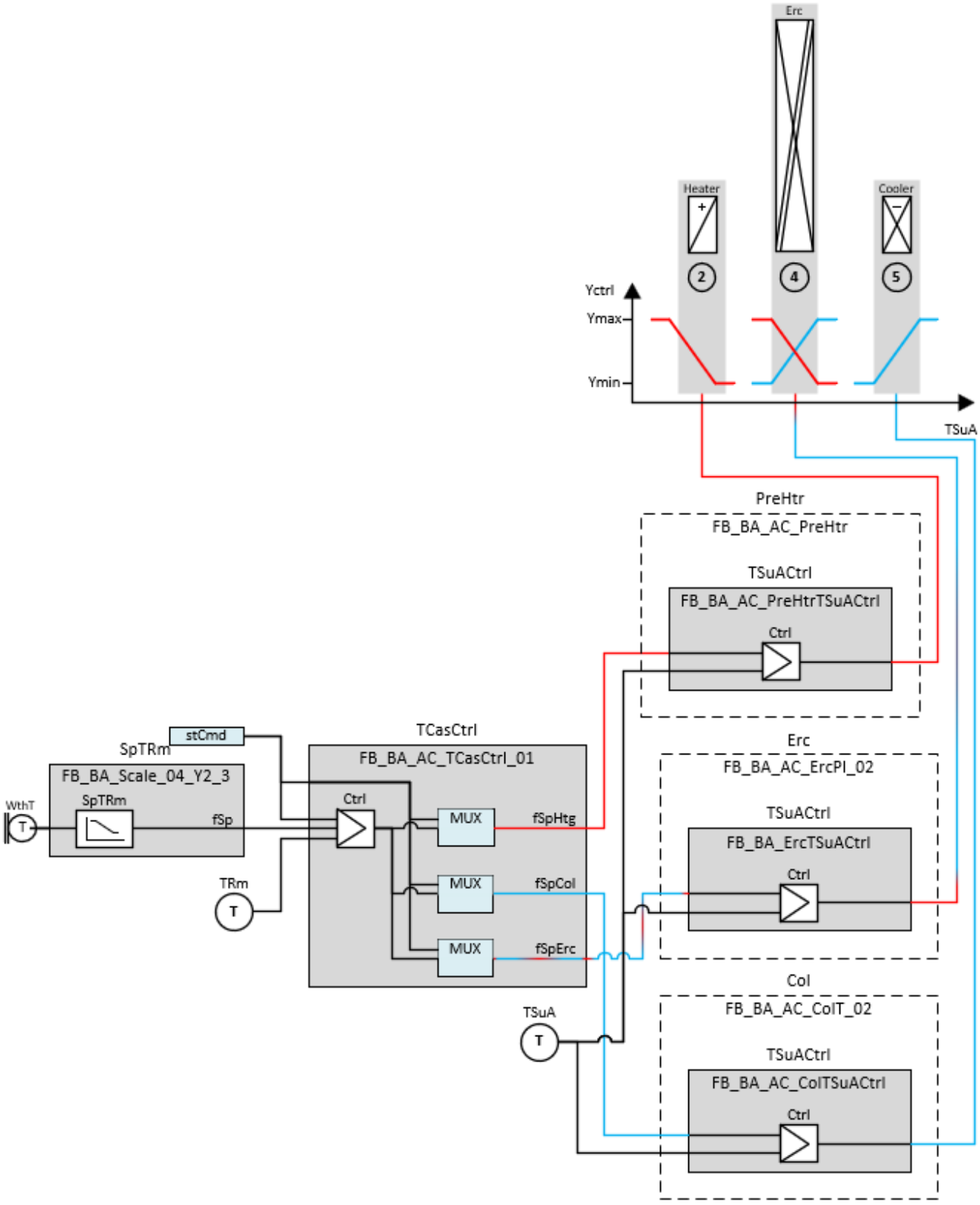
Function block	eEnPlantLock	Function block	bPrioHigh	bPrioMedium
TSuA.MV	E_BA_LockPriority.eMedium	PlantLock		x
FiltOuAMonit.DstMainAlarm	E_BA_LockPriority.eMedium			x
OuADmp.MonitOpen.Dst	E_BA_LockPriority.eMedium			x
SuAFan.Dst	E_BA_LockPriority.eMedium			x
SuAFan.ThOvrld	E_BA_LockPriority.eMedium			x
SuAFan.MntnSwi	E_BA_LockPriority.eMedium			x
SuADiffPrssMonit.Dst	E_BA_LockPriority.eMedium			x
ExhADmp.MonitOpen.Dst	E_BA_LockPriority.eMedium			x
ExtAFan.Dst	E_BA_LockPriority.eMedium			x
ExtAFan.ThOvrld	E_BA_LockPriority.eMedium			x
ExtAFan.MntnSwi	E_BA_LockPriority.eMedium			x
ExtADiffPrssMonit.Dst	E_BA_LockPriority.eMedium			x
PreHtr.TRt.MV	E_BA_LockPriority.eMedium			x
PreHtr.TFrost.MV	E_BA_LockPriority.eMedium			x
PreHtr.FrostThermostat.Input	E_BA_LockPriority.eMedium			x
PreHtr.Pu.Dst	E_BA_LockPriority.eMedium			x

The parameterization of the lock priority of the event-enabled objects can be found in the *FB_init* of this template.

Temperature control

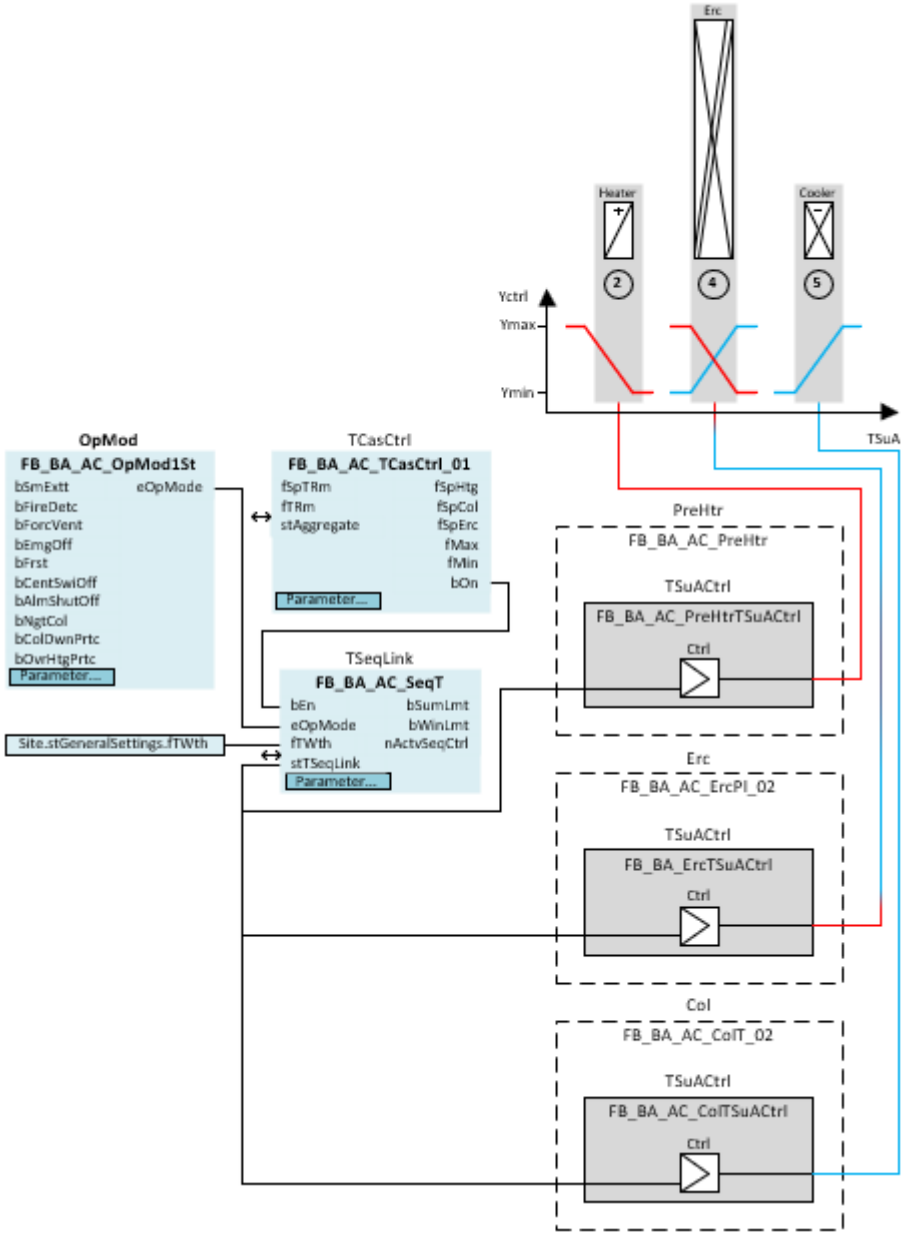
In this plant template a simple room air supply air cascade is realized as control strategy.

The setpoints for the plant are generated in the templates *SpTRm* / *TCasCtrl* and forwarded to the sequence controllers within the templates *PreHtr*, *Col* and *Erc*.



Sequence control

The number of sequence controllers in the control sequence varies and depends on the number of aggregates in the ventilation system. The PID sequence controllers, which contribute to increasing the supply air temperature, are arranged on the left of the control sequence. The controllers of the aggregates that cool down the supply air temperature are located on the right.
Several heating or cooling aggregates can be integrated into the control sequence.
Only one PID controller is active at any one time in the sequence control. All others remain at the value of their maximum or minimum control value.



The sequence control *TSeqLink* is enabled when *bOn* is received from the cascade control *TCasCtrl*.
The start sequence controller with which the plant starts up is determined depending on the weather temperature Site.stGeneralSettings.fTWth [▶ 1116] or the operation mode *OpMod*.

The *stTSeqLink* data structure is used to transmit the plant enable and the number of the start controller to the sequence controllers of the *PreHtr* air heater, *Col* air cooler and *Erc* energy recovery unit.
The ventilation system in the sequence diagram is equipped with an air heater, an energy recovery unit and an air cooler.
All aggregates are integrated into a sequence controller for continuous control of the supply air temperature *TSuA* at very low outdoor temperatures in winter through to high outdoor temperatures in summer.

The sequence on the left starts with the control range of the *PreHtr* air heater. At low outside temperatures, the output of the air heater is increased until the supply air temperature *TSuA* has reached its setpoint.

If the outside temperature *TOuA* rises, the sequence controller *TSuACtrl* of the *PreHtr* air heater reduces its actuating variable until it reaches the lower limit of its control range.

If the setpoint of the supply air temperature *TCasCtrl* is still not reached, the sequence controller *TSuACtrl* of the air heater PreHtr switches to the sequence controller *TSuACtrl* of the energy recovery *Erc*.
The behavior of the energy recovery *Erc* depends on the ratio of the weather temperature Site.stGeneralSettings.fTWth [▶ 1116] and the extract air temperature *TExtA*.

If the weather temperature Site.stGeneralSettings.fTWth [▶ 1116] is lower than the extract air temperature *TExtA*, energy recovery recovers heat energy from the extract air and transfers it to the outside air. The red characteristic curve is then activated in the control sequence.

The control direction of the sequence controller *TSuACtrl* is therefore indirect (heating mode).

In summer, the extract air temperature is often lower than the weather temperature. The energy flow of the energy recovery *Erc* is then inverted. The heat from the outside air is transferred to the exhaust air *TExhA* and the supply air temperature is reduced as a result. In this operating case, the sequence follows the blue characteristic curve. The control direction of the sequence controller *TSuACtrl* is therefore direct (cooling mode).

At high outside temperatures, if the recovery capacity of the *Erc* is not sufficient to cool the outside air *OuA* down far enough, the sequence controller *TSuACtrl* of the energy recovery *Erc* switches on to the sequence controller *TSuACtrl* of the cooler *Col*. This increases the capacity of the cooler until the setpoint of the supply air temperature is reached.

Filter monitoring

The outside and extract air filters are monitored via the function blocks *FiltOuAMonit* and *FiltExtAMonit* with the analog differential pressure sensors *DiffPrssOuAFilt* / *DiffPrssExtAFilt*. The filter monitors are enabled when *bOn* is received from the cascade control *TCasCtrl*.

Night cooling

Night cooling can be controlled via the function block **SumNgtCol**.

Syntax

```
FUNCTION_BLOCK FB_BA_AHU_1st_10 EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  TOuA                    : FB_BA_SensorAnalog;
  TExhA                   : FB_BA_SensorAnalog;
  TSuA                    : FB_BA_SensorAnalog;
  TExtA                   : FB_BA_SensorAnalog;
  TRm                     : FB_BA_SensorAnalog;

  DiffPrssOuAFilt         : FB_BA_SensorAnalog;
  FiltOuAMonit            : FB_BA_Hys_21;
  DiffPrssExtAFilt        : FB_BA_SensorAnalog;
  FiltExtAMonit           : FB_BA_Hys_21;

  OuADmp                  : FB_BA_AC_Dmp2P;
  SuAPrss                 : FB_BA_SensorAnalog;
  SuADiffPrss             : FB_BA_SensorAnalog;
  SuAFan                  : FB_BA_AC_FanCtl;
  SuADiffPrssMonit        : FB_BA_FdbAnalog;

  ExhADmp                 : FB_BA_AC_Dmp2P;
  ExtAPrss                : FB_BA_SensorAnalog;
  ExtADiffPrss            : FB_BA_SensorAnalog;
  ExtAFan                 : FB_BA_AC_FanCtl;
  ExtADiffPrssMonit       : FB_BA_FdbAnalog;

  PreHtr                  : FB_BA_AC_PreHtr;
  Erc                     : FB_BA_AC_ErcPl_02;
  Col                     : FB_BA_AC_ColT_02;

  SpTRm                   : FB_BA_Scale_04_Y2_3;
  TCasCtrl                : FB_BA_AC_TCasCtrl_01;

  SumNgtCol               : FB_BA_AC_SumNgtCol;
  OpMod                   : FB_BA_AC_OpMod1St;
  EnPrio                  : FB_BA_AC_EnPrio;
```

```
  PlantLock               : FB_BA_PlantLock;

  PlantCmd                : FB_BA_AC_PlantCmd_AHU_1st_10;

  TSeqLink                : FB_BA_AC_SeqT;
  stTSeqLink              : ST_BA_SeqLink;

  arrAggregate            : ARRAY [1..BA_Param.nMaxAggregate] OF ST_BA_Aggregate;
END_VAR
```

VAR_INPUT CONSTANT

Name	Type	Description
TOuA	FB_BA_SensorAnalog [▶ 1087]	Represents the outside air temperature
TExhA	FB_BA_SensorAnalog [▶ 1087]	Represents the exhaust air temperature
TSuA	FB_BA_SensorAnalog [▶ 1087]	Represents the supply air temperature
TExtA	FB_BA_SensorAnalog [▶ 1087]	Represents the extract air temperature
TRm	FB_BA_SensorAnalog [▶ 1087]	Represents the room temperature
DiffPrssOuAFilt	FB_BA_SensorAnalog [▶ 1087]	Analog differential pressure via the outside air filter
FiltOuAMonit	FB_BA_Hys_21 [▶ 1010]	Analog monitoring of the outside air filter. The filter monitoring uses the analog differential pressure *DiffPrssOuAFilt* and can trigger both a pre-alarm and a main alarm.
DiffPrssExtAFilt	FB_BA_SensorAnalog [▶ 1087]	Analog differential pressure via the extract air filter
FiltExtAMonit	FB_BA_Hys_21 [▶ 1010]	Analog monitoring of the extract air filter. Filter monitoring uses the analog differential pressure *FiltExtAMonit* and can trigger both a pre-alarm and a main alarm.
OuADmp	FB_BA_AC_Dmp2P [▶ 719]	Activation of the outside air damper with a spring return actuator and end position monitor.
SuAPrss	FB_BA_SensorAnalog [▶ 1087]	Supply air pressure
SuADiffPrss	FB_BA_SensorAnalog [▶ 1087]	Differential pressure via the supply air fan
SuAFan	FB_BA_AC_FanCtl [▶ 743]	Activation and control of the speed-controlled supply air fan.
SuADiffPrssMonit	FB_BA_FdbAnalog [▶ 1016]	Analog differential pressure monitoring via the supply air fan. Differential pressure monitoring uses the *SuADiffPrss* analog differential pressure sensor and can trigger a plant shutdown alarm.
ExhADmp	FB_BA_AC_Dmp2P [▶ 719]	Control of the exhaust air damper with a spring return actuator and end position monitor.
ExtAPrss	FB_BA_SensorAnalog [▶ 1087]	Extract air pressure. The extract air pressure is the controlled variable of the pressure controller of the extract air fan *ExtAFan*.
ExtADiffPrss	FB_BA_SensorAnalog [▶ 1087]	Differential pressure via the extract air fan
ExtAFan	FB_BA_AC_FanCtl [▶ 743]	Control and regulation of the speed-controlled extract air fan.
ExtADiffPrssMonit	FB_BA_FdbAnalog [▶ 1016]	Analog differential pressure monitoring via the extract air fan. Differential pressure monitoring uses the analog differential pressure sensor *ExtADiffPrss* and can trigger a plant shutdown alarm.
PreHtr	FB_BA_AC_PreHtr [▶ 748]	Activation and control of a hot water air heater. The function block is regarded as an aggregate and is part of the temperature sequence control of this plant due to its implemented sequence controller.

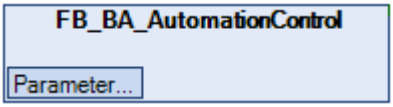
Name	Type	Description
Erc	FB_BA_AC_ErcPl_02 [▶ 733]	Activation and control of a plate heat exchanger for the energy recovery of a ventilation system. The function block is considered as an aggregate and is part of the temperature sequence control of this plant due to its implemented sequence controller.
Col	FB_BA_AC_ColT_02 [▶ 716]	Activation and control of a temperature-controlled cold water air cooler. The function block is considered as an aggregate and is part of the temperature sequence control of this plant due to its implemented sequence controller.
SpTRm	FB_BA_Scale_04_Y2_3 [▶ 1048]	Setpoint program for supply air temperature control with a supply air temperature setpoint, including summer/winter compensation via a characteristic curve.
TCasCtrl	FB_BA_AC_TCasCtrl_01 [▶ 801]	Room temperature control by means of room-supply air cascade. It consists of a master controller for setpoint calculation for heating, cooling and energy recovery.
SumNgtCol	FB_BA_AC_SumNgtCol [▶ 783]	Night cooling control
OpMod	FB_BA_AC_OpMod1St [▶ 763]	The function block is used to switch on the plant. It represents the choice of plant operation mode, a timer program and a plant selector switch.
EnPrio	FB_BA_AC_EnPrio [▶ 759]	Defines the plant release for controlling the step sequence control of a plant and the release of the priorities "Safety", "Critcial" and "Program" based on the plant operation mode *OpMod*. These releases are transferred to *PlantCmd*.
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the plant. These relevant faults trigger the operation mode Fault *E_BA_AC_OpMod01.eAlarm* in the operation mode program *OpMod*. As a result, the air conditioning system is shut down. A list of the faults that shut down the plant can be found under Fault shutdowns.
PlantCmd	FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 786]	The function block represents the plant command of the aggregates and functions of this air conditioning system. The integrated step sequence controller fixes the order in which the individual aggregates of the plant are switched on or off sequentially one after the other.
TSeqLink	FB_BA_AC_SeqT [▶ 778]	Represents the start and control of the supply air temperature sequence control.
stTSeqLink	ST_BA_SeqLink [▶ 123]	Data and command structure between the individual sequence controllers of the temperature control sequence *PreHtr*, *Col*, *Erc* and the supply air temperature sequence control *TSeqLink*.
arrAggregates	ST_BA_Aggregate [▶ 278]	The array contains a bidirectional data and command structure for each aggregate. When the step sequence controller is switched up or down, the switching command determined for each aggregate is transferred via the bidirectional communication structure *arrAggregate*. Feedback from the aggregates for switching the step sequence control up or down is transmitted to the plant control unit via the communication structure.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.2 AutomationControl

6.1.4.2.1.2.1 FB_BA_AutomationControl



The template contains basic functions and project information relevant for an automation station.

In the sub-template FB_BA_Device [▶ 830] information about the project and diagnostic options are offered. A parameterization of the BACnet device and general BACnet parameters takes place.

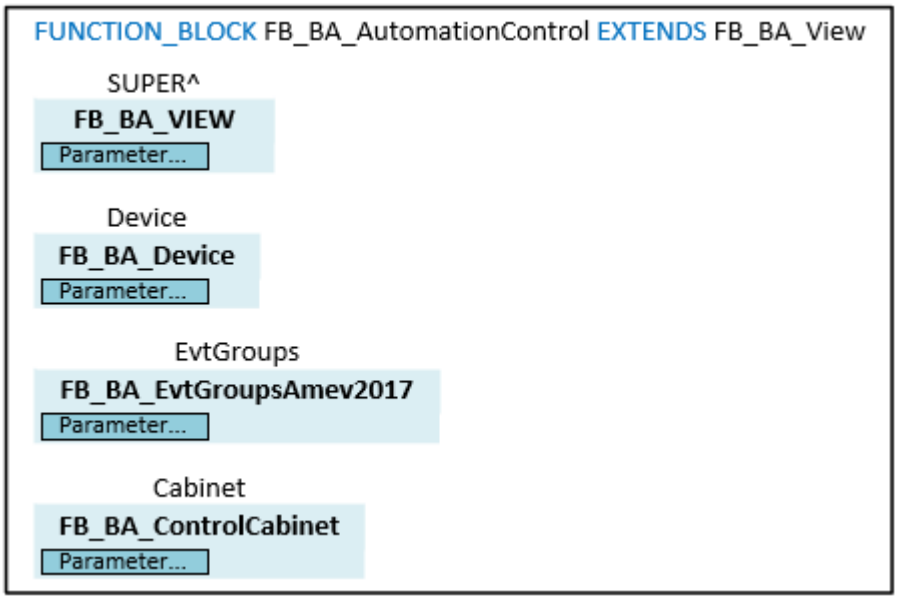
Below *FB_BA_Device* there are two possibilities to integrate the BACnet system time or the local NT system time of the TwinCAT system.

The sub-template FB_BA_EvtGroupsAMEV2017 [▶ 844] provides event classes (notification classes) for the intrinsic reporting.

The sub-template FB_BA_ControlCabinet [▶ 829] 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.

Block diagram



Syntax

```
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

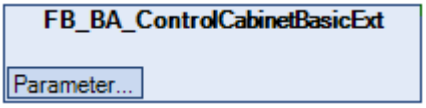
Name	Type	Description
Device	FB_BA_Device [▶ 830]	Information about the project and diagnostic options are offered in the template. A parameterization of the BACnet device and general BACnet parameters takes place. Below FB_BA_Device there are two possibilities to integrate the BACnet system time or the local NT system time of the TwinCAT system.
EvtGroups	FB_BA_EvtGroupsAMEV2017 [▶ 844]	The template provides event classes (notification classes) for intrinsic reporting.
Cabinet	FB_BA_ControlCabinet [▶ 829]	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

6.1.4.2.1.2.2 ControlCabinet

6.1.4.2.1.2.2.1 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 [▶ 134] used in the template.

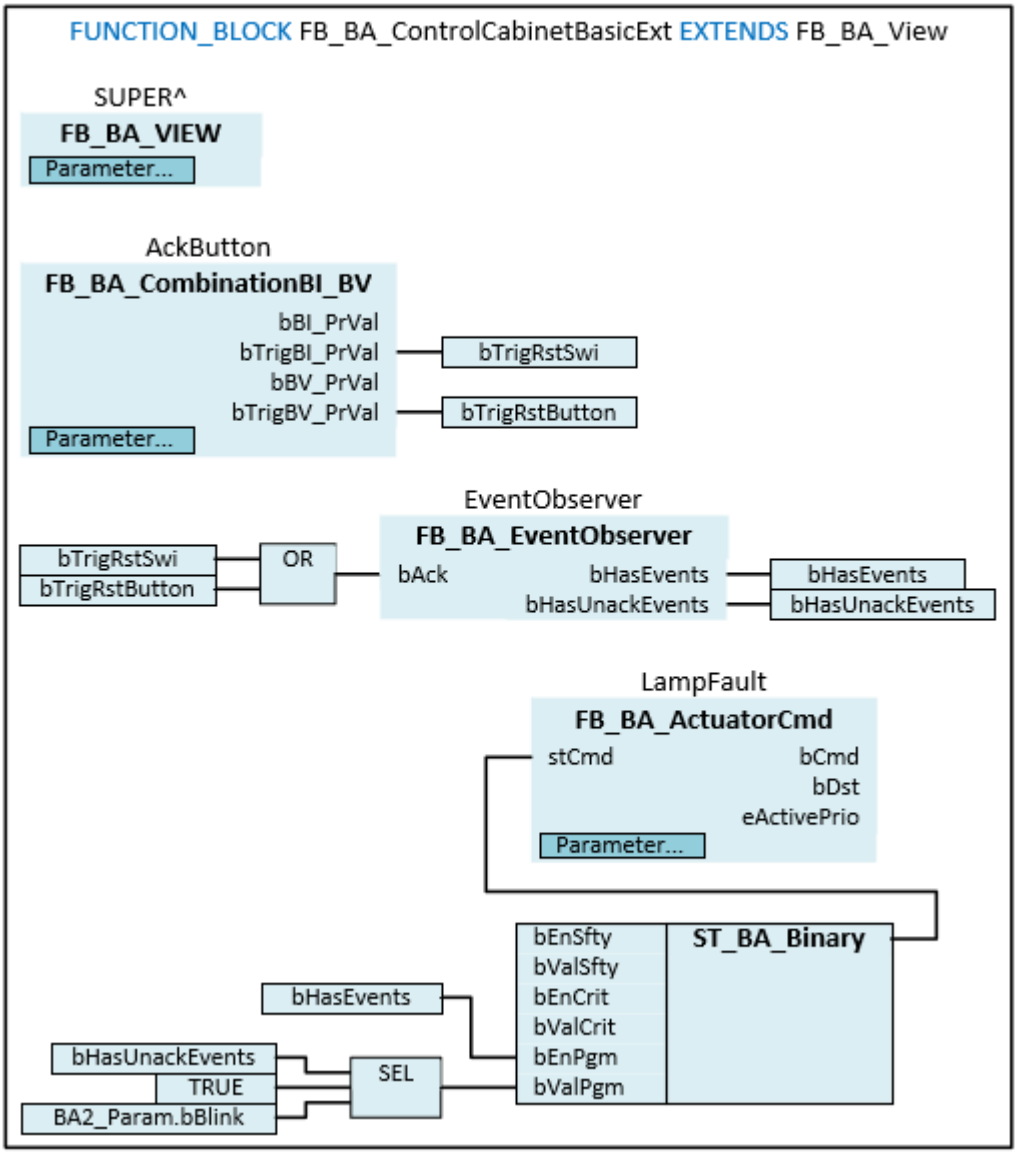
The resulting collective alarm is used in the template LampFault [▶ 986] to control a collective error signal lamp. In case of a new, unacknowledged alarm or event, the signal lamp is flashing.

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 button press, 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.

Block diagram



Syntax

```
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 [▶ 986]	The template is used to control a collective error signal lamp.
AckButton	FB_BA_CombinationBI_BV [▶ 1084]	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 [▶ 134]	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 *Parent* 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.

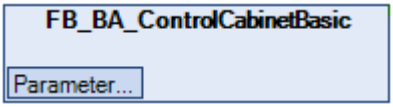
VAR

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 *Input* (see FB_BA_CombinationBI_BV).
bTrigRstButton	BOOL	Display of the acknowledgement signal from the BV object *Value* (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

6.1.4.2.1.2.2.2 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 [▶ 134] used in the template.

The resulting collective alarm is used in the template LampFault [▶ 986] to control a collective error signal lamp. In case of a new, unacknowledged alarm or event, the signal lamp is flashing.

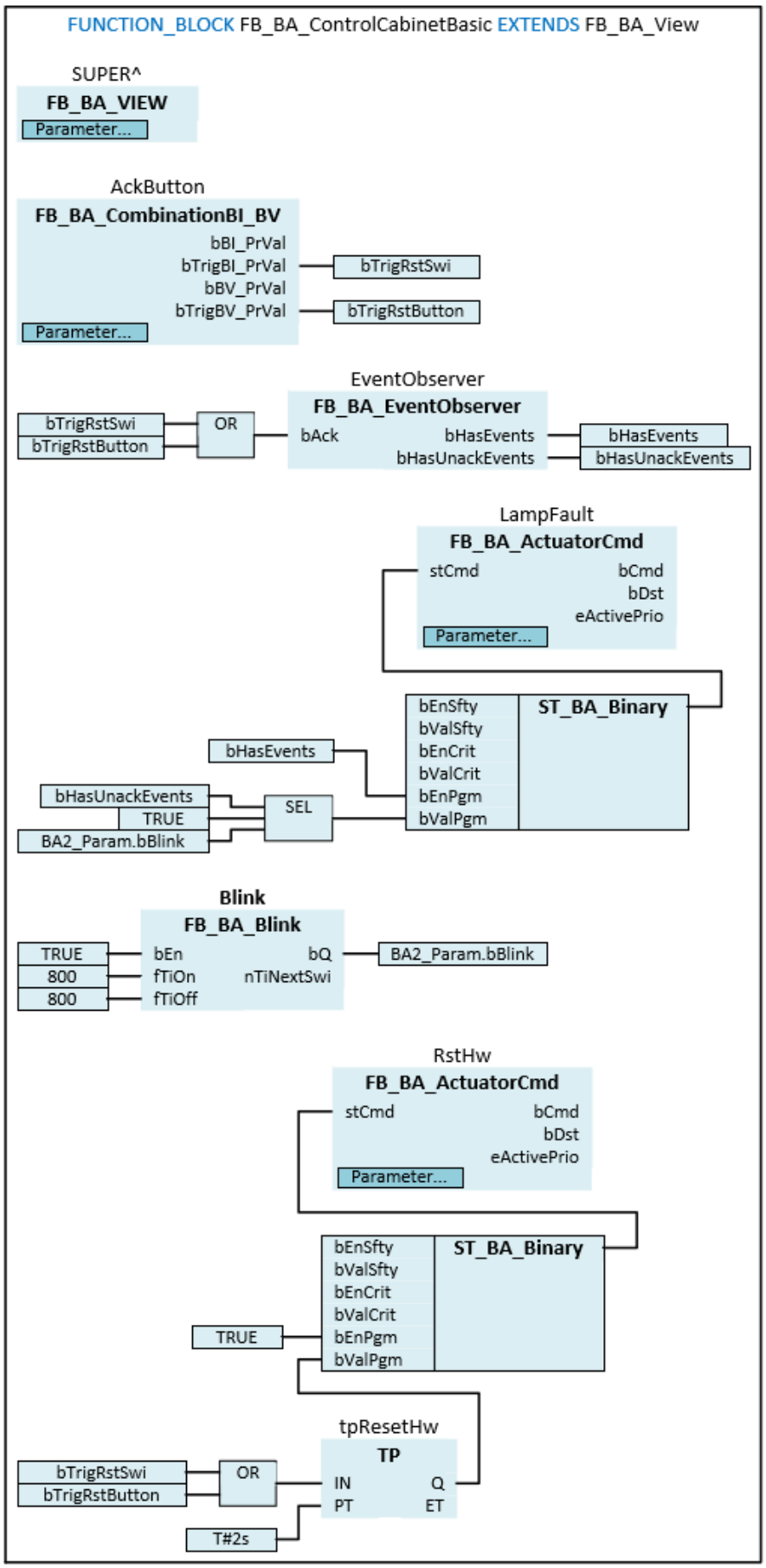
If only acknowledged alarms are present, the lamp is permanently on (new value message).

A momentary pulse for resetting hardware, is output via the function block RstHw [▶ 986]. 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 button press, 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.

Block diagram



Syntax

```
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 [▶ 986]	The template is used to control a collective error signal lamp.
AckButton	FB_BA_CombinationBI_BV [▶ 1084]	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 [▶ 986]	The template can be used to wipe in relay circuits (e.g. frost protection relay).
EventObserver	FB_BA_EventObserver [▶ 134]	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 *Parent* 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.

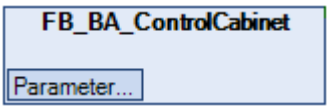
VAR

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 [▶ 435]	The function block generates a blink pulse. This blink pulse is written to the global variable *BA2_Param.bBlink*.
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 *Input* (see FB_BA_CombinationBI_BV).
bTrigRstButton	BOOL	Display of the acknowledgement signal from the BV object *Value* (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

6.1.4.2.1.2.2.3 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 [▶ 825] 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

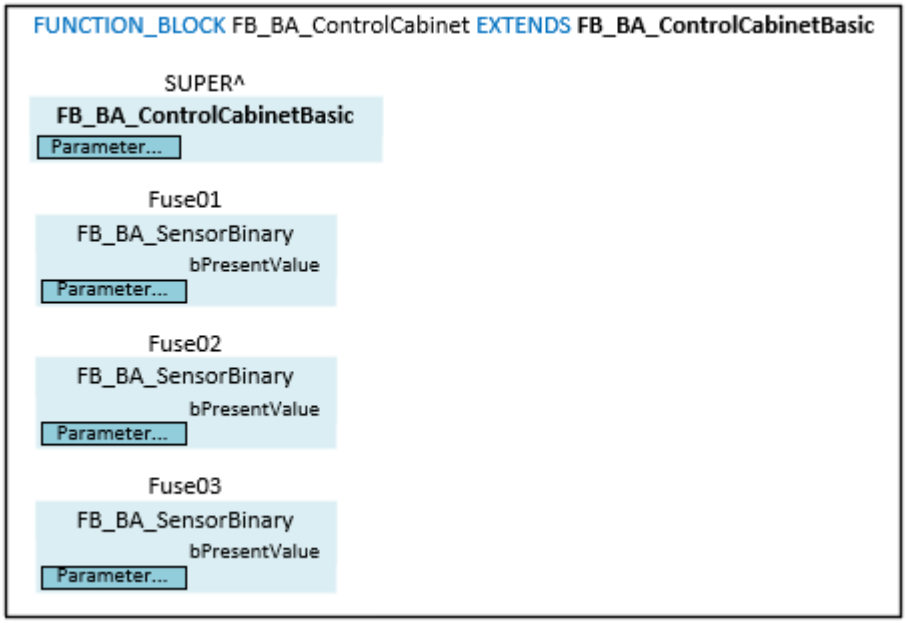
FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_View [▶ 241]

FB_BA_ControlCabinetBasic [▶ 825]

Block diagram



Syntax

```
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

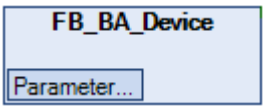
Name	Type	Description
Fuse01	FB_BA_SensorBinary [▶ 1088]	The template contains a binary input for connecting a fuse message.
Fuse02	FB_BA_SensorBinary [▶ 1088]	The template contains a binary input for connecting a fuse message.
Fuse03	FB_BA_SensorBinary [▶ 1088]	The template contains a binary input for connecting a fuse message.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.3 Device

6.1.4.2.1.2.3.1 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.

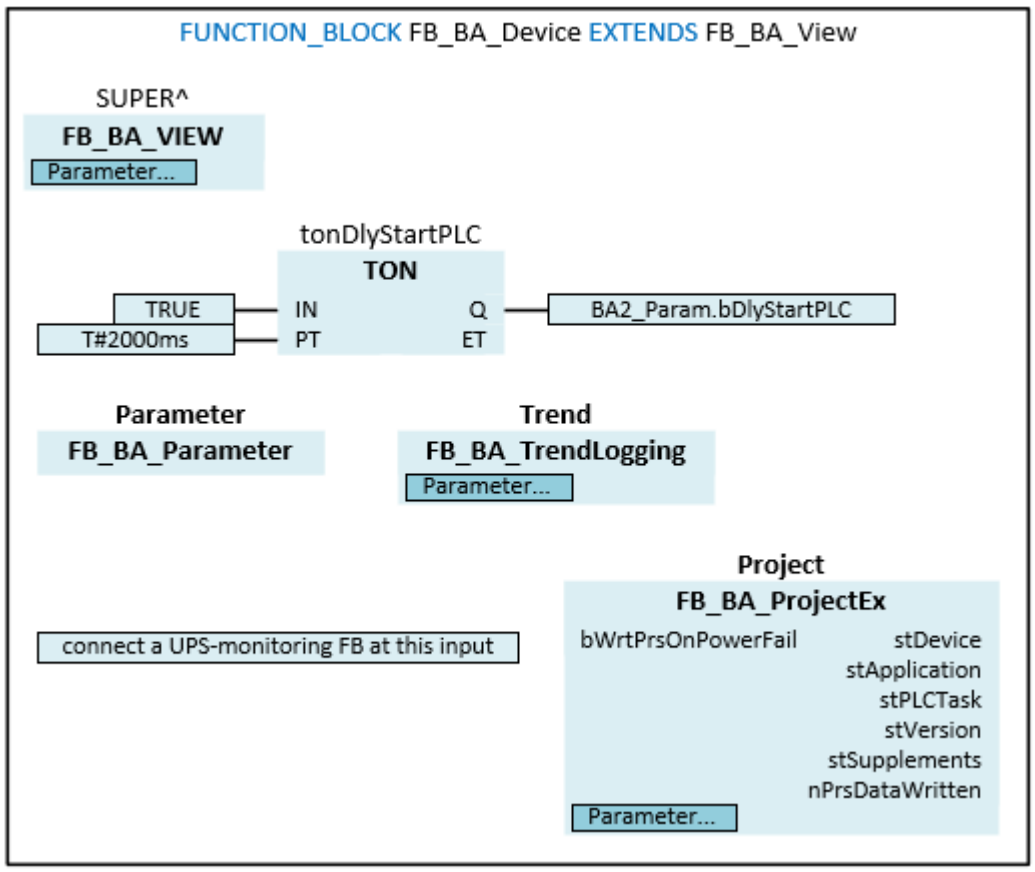
The template FB_BA_TrendLogging contains various Trendlog objects.

The function block FB_BA_Parameter [▶ 833] is used to parameterize global parameter lists of the library Tc3_XBA [▶ 99].

Within the TimeSettings folder there are two 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.

Block diagram



Syntax

```
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 [▶ 1056]	Calling the Trendlog objects.
Project	FB_BA_ProjectEx [▶ 841]	The function block provides information about the project and offers information and diagnostic options. It also stores the persistent data. It has the input *bWrtPrsOnPowerFail*, 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 [▶ 833]	Within the function block global parameter lists of the libraries Tc3_XBA [▶ 99], Tc3_BACnetRev14, Tc3_BA2_Common and Tc3_BA2 [▶ 267] are parameterized.

Variables

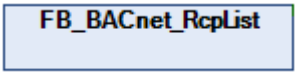
Name	Type	Description
tonDlyStartPLC	TON	The timing element sets the global variable bDlyStartPLC [▶ 1118] delayed after a PLC restart.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.3.2 BACnetSettings

6.1.4.2.1.2.3.2.1 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.

Syntax

```
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

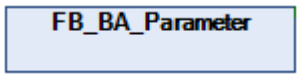
```
IF (XBA_Globals.Top.ProjectState = E_BA_ProjectState.eFirstOpCycle) THEN
  bWrite := TRUE;
END_IF
```

```
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

6.1.4.2.1.2.3.2.2 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 [▶ 126] of the library Tc3_XBA [▶ 99]. 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 [▶ 159]!

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.

Syntax

```
FUNCTION_BLOCK FB_BA_Parameter
VAR
//    Recipient    : FB_BACnet_RcpList;
END_VAR
```

Implementation part

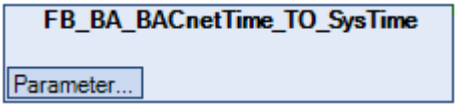
```
// Recipient list to write (to all event classes):
// Recipient();
```

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.3.3 TimeSettings

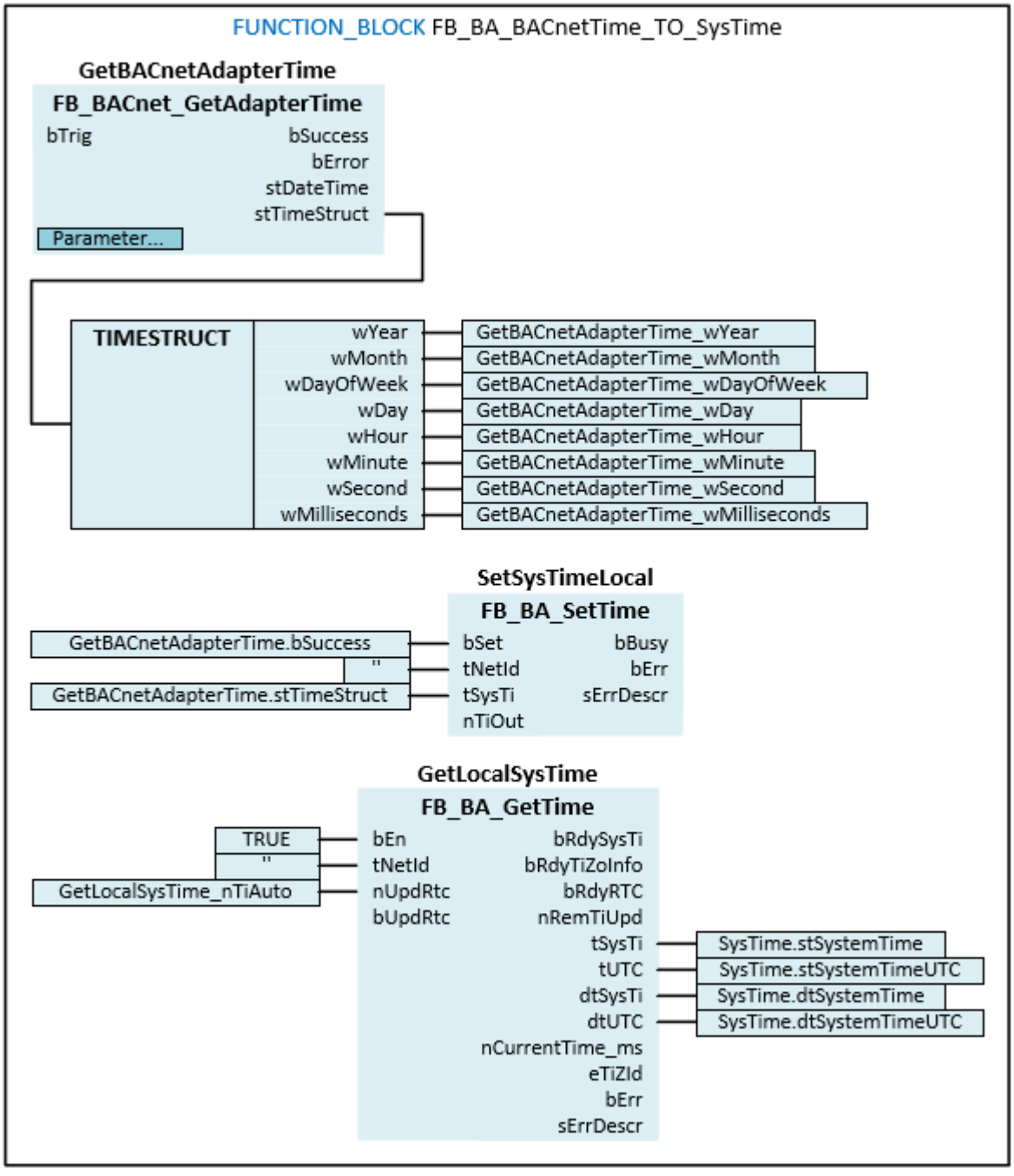
6.1.4.2.1.2.3.3.1 FB_BA_BACnetTime_TO_SysTime



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 [▶ 1117].

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_BACnetTime_TO_SysTime
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterUnit':= 's'}
  GetLocalSysTime_nTiAuto                : UDINT := 60;
END_VAR
VAR_INPUT CONSTANT
  SetSysTimeLocal                        : FB_BA_SetTime;
  GetLocalSysTime                        : FB_BA_GetTime;
  GetBACnetAdapterTime                   : FB_BACnet_GetAdapterTime;
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 PERSISTENT

Name	Type	Description
GetLocalSysTime_nTiAuto	UDINT	Based on this time information, the local NT system time of a TwinCAT system is regularly determined.

Inputs CONSTANT

Name	Type	Description
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 [▶ 1117].
GetBACnetAdapterTime	FB_BACnet_GetAdapterTime	The function block GetBACnetAdapterTime determines the local BACnet system time. Important parameters are explained below.

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.

Parameter of GetBACnetAdapterTime

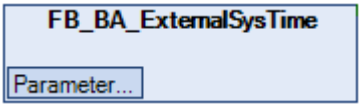
These parameters are used to configure time synchronization.

Name	Type	Description
GetBACnetAdapterTime.bTrig	BOOL	A rising edge at this input determines the local BACnet system time of a TwinCAT system.
GetBACnetAdapterTime.nTiAuto	UDINT	The local BACnet system time of a TwinCAT system is automatically determined on a regular basis using this time specification from the parameter menu. The time specification must be greater than or equal to 500ms. The default value is 1800s.
GetBACnetAdapterTime.fbAdapter	fbAdapter	The function block *fbAdapter* represents a BACnet device. This is pre-initialized to the local BACnet device of the TwinCAT system under Devices.
GetBACnetAdapterTime.bSuccess	BOOL	The system time was read successfully from the target system. The variable is TRUE for one cycle.
GetBACnetAdapterTime.stDateTime	ST_BA_DateTime	Current BACnet system time of a TwinCAT system.
GetBACnetAdapterTime.stTimeStruct	TIMESTRUCT	Current BACnet system time of a TwinCAT system.
GetBACnetAdapterTime.bError	BOOL	Error while reading the BACnet system time.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.3.3.2 FB_BA_ExternalSysTime

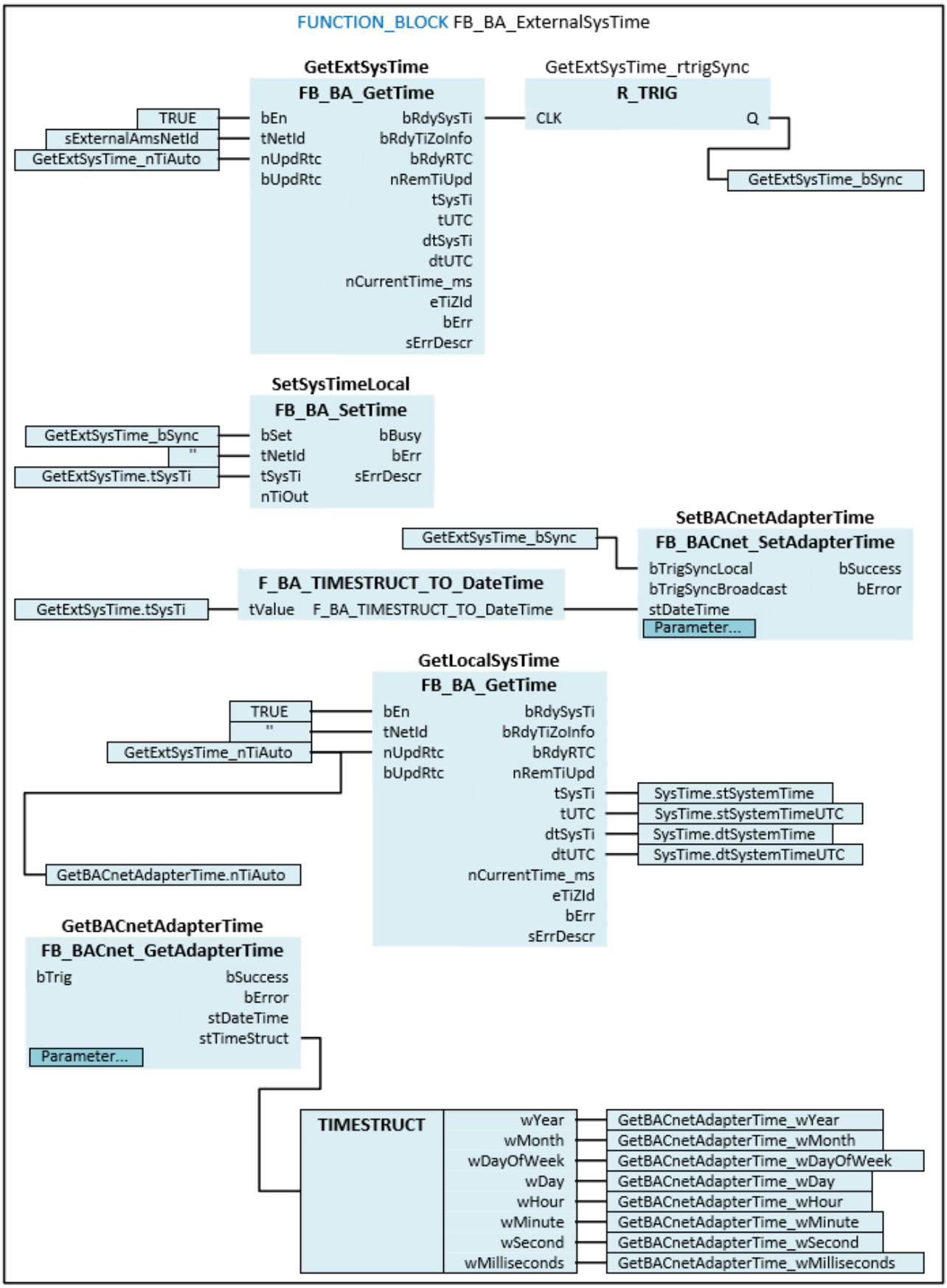


The template reads the NT system time of an external TwinCAT system at regular intervals.

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



Syntax

```
FUNCTION_BLOCK FB_BA_ExternalSysTime
VAR_INPUT CONSTANT PERSISTET
  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_BACnet_SetAdapterTime;

  GetLocalSysTime                       : FB_BA_GetTime;
  GetBACnetAdapterTime                  : FB_BACnet_GetAdapterTime;
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 PERSISTENT

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.

Inputs CONSTANT

Name	Type	Description
GetExtSysTime	FB_BA_GetTime	The function block GetExtSysTime determines the external NT system time of a TwinCAT system.
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.
SetBACnetAdapterTime	FB_BACnet_SetAdapterTime	The function block SetBACnetAdapterTime can set both the local BACnet system time of a TwinCAT system (*SetBACnetAdapterTime.bTrigSyncLocal*) and, by triggering a broadcast command (*SetBACnetAdapterTime.bTrigSyncBroadcast*), all BACnet system times of a network. Important parameters are explained below.
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 [▶ 1117].
GetBACnetAdapterTime	FB_BACnet_GetAdapterTime	The function block "GetBACnetAdapterTime" determines the local BACnet system time. Important parameters are explained below.

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_rtrigSync	R_TRIG	The function block generates a rising edge after successfully reading the external system time GetExtSysTime and passes this on to *GetExtSysTime_bSync*.
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.

Parameter of SetBACnetAdapterTime

These parameters are used to configure time synchronization.

Name	Type	Description
SetBACnetAdapterTime.bTrigSyncLocal	BOOL	On a rising edge at this input, the content of the structure *SetBACnetAdapterTime.stDateTime* is written to the local BACnet system time of the TwinCAT system.
SetBACnetAdapterTime.bTrigSyncBroadcast	BOOL	On a rising edge at this input, the content of the *SetBACnetAdapterTime.stDateTime* structure is written to all BACnet system times of a network as well as the local BACnet system time of the TwinCAT system.
SetBACnetAdapterTime.stDateTime	ST_BA_DateTime	Time specification which is written to the local BACnet system time.
SetBACnetAdapterTime.fbAdapter	fbAdapter	The function block *fbAdapter* represents a BACnet device. This is pre-initialized to the local BACnet device of the TwinCAT system under Devices.
SetBACnetAdapterTime.bSuccess	BOOL	The system time was read successfully from the target system. The variable is TRUE for one cycle.
SetBACnetAdapterTime.bError	BOOL	Error while reading the BACnet system time.

Parameter of GetBACnetAdapterTime

These parameters are used to configure time synchronization.

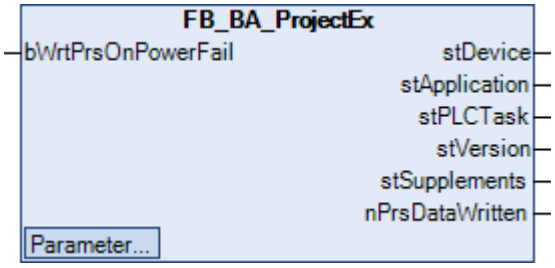
Name	Type	Description
GetBACnetAdapterTime.bTrig	BOOL	A rising edge at this input determines the local BACnet system time of a TwinCAT system.
GetBACnetAdapterTime.nTiAuto	UDINT	The local BACnet system time of a TwinCAT system is automatically determined on a regular basis using this time specification from the parameter menu. The time specification must be greater than or equal to 500ms. The default value is 1800s.
GetBACnetAdapterTime.fbAdapter	fbAdapter	The function block *fbAdapter* represents a BACnet device. This is pre-initialized to the local BACnet device of the TwinCAT system under Devices.
GetBACnetAdapterTime.bSuccess	BOOL	The system time was read successfully from the target system. The variable is TRUE for one cycle.
GetBACnetAdapterTime.stDateTime	ST_BA_DateTime	Current BACnet system time of a TwinCAT system.
GetBACnetAdapterTime.stTimeStruct	TIMESTRUCT	Current BACnet system time of a TwinCAT system.
GetBACnetAdapterTime.bError	BOOL	Error while reading the BACnet system time.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.2.3.4 Project

6.1.4.2.1.2.3.4.1 FB_BA_ProjectEx



This template is derived from the basic function block FB_BA_Project [▶ 227].

It provides the project version number for information purposes and ensures that persistent data is saved.

The writing of persistent data has 2 modes:

- **Normal state of the system:**

Commissioning is complete and the system is running.

The persistent data is written at a fixed interval *tPrsData_DefaultPeriod*, pre-set to 30 minutes. This protects the memory card. Parameterizing *tPrsData_DefaultPeriod* below 10 minutes resets this value to 30 minutes as a precaution.

- **Commissioning mode**:

By automatically detecting an online change, the system switches to commissioning mode and the persistent data is now saved at shorter intervals. The interval *tPrsData_ComissioningPeriod*, which is pre-set to 10 minutes, applies here. Parameterizing *tPrsData_ComissioningPeriod* below 30 seconds resets this value to 5 minutes as a precaution.

If no online change is detected after a fixed *DETECT_COMISSIONING_DURATION* time of 60 minutes, the routine returns to normal operation.

Syntax

```
FUNCTION_BLOCK FB_BA_ProjectEx EXTENDS FB_BA_Project
VAR_INPUT CONSTANT PERSISTENT
  tPrsData_DefaultPeriod         : TIME := T#30M;
  tPrsData_ComissioningPeriod    : TIME := T#10M;
END_VAR
VAR_INPUT CONSTANT
  bPrsData_ManualWriteData       : BOOL;
END_VAR
VAR_INPUT
  bWrtPrsOnPowerFail             : BOOL;
END_VAR
VAR_OUTPUT
  nPrsDataWritten                : UINT;
END_VAR
```

Inputs CONSTANT PERSITENT

Name	Type	Description
tPrsData_DefaultPeriod	TIME	Storage interval in normal state.
tPrsData_ComissioningPeriod	TIME	Storage interval in commissioning state.

Inputs CONSTANT

Name	Type	Description
bPrsData_ManualWriteData	BOOL	This variable can be set to TRUE for commissioning purposes. This triggers a one-time saving of the persistent variables.

Inputs

Name	Type	Description
bWrtPrsOnPowerFail	BOOL	Input variable for linking with a UPS function block

Outputs

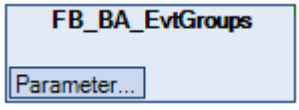
Name	Type	Description
nPrsDataWritten	UINT	Counter output for the executed write cycles.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.62	TF8040 | TwinCAT Building Automation from V5.9.0.0

6.1.4.2.1.2.4 EventClasses

6.1.4.2.1.2.4.1 FB_BA_EvtGroups



The template provides a group of notification classes.

The initialization of the template takes place within the method FB_Init.

Notification class matrix

Event category	Meaning	Priority	Notification class	NC object EC object	Sample
Hazard notification (Life Safety)	Danger to life	00 - 29	EC_ID.NC10	LifeSafety.EC	Fire alarm, robbery
Hazard notification (Property Safety)	Safety message	30 - 59	EC_ID.NC20	SafetyMsg.EC	Burglary, unauthorized entry
Alarm message	Message indicates system failure or requires immediate intervention.	60 - 89	EC_ID.NC30	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.NC40	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.NC50	MaintenanceMsg01.EC	Operating hours, tank level, repair switch, etc.
Maintenance message	Indication of maintenance activity or similar.	120 - 149	EC_ID.NC51	MaintenanceMsg02.EC	Filter end reached, filter dirty, etc.
System message	Fault message from the GA system.	150 - 219	EC_ID.NC60	SystemMsg.EC	Device malfunction, battery message, communication interruption, etc.
Manual intervention	Manual intervention	220	EC_ID.NC70	ManualOp.EC	Manual intervention
Subject to confirmation	Other messages	221 - 255	EC_ID.NC80	OtherMsg.EC	Operating state change, operation modes, trend memory full, etc.

Syntax

```
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
```

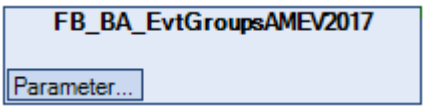
VAR_INPUT CONSTANT

Name	Type	Description
LifeSafety	FB_BA_EvtCategory [▶ 1002]	The template calls the "Danger to life" event category.
SafetyMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the "Danger message, safety message" event category.
AlarmMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the "Alarm message" event category.
FaultMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the "Fault message" event category.
MaintenanceMsg01	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "Maintenance message".
MaintenanceMsg02	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "Maintenance message".
SystemMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "System message".
ManualOp	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "Manual intervention".
OtherMsg	FB_BA_EvtCategory [▶ 1002]	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

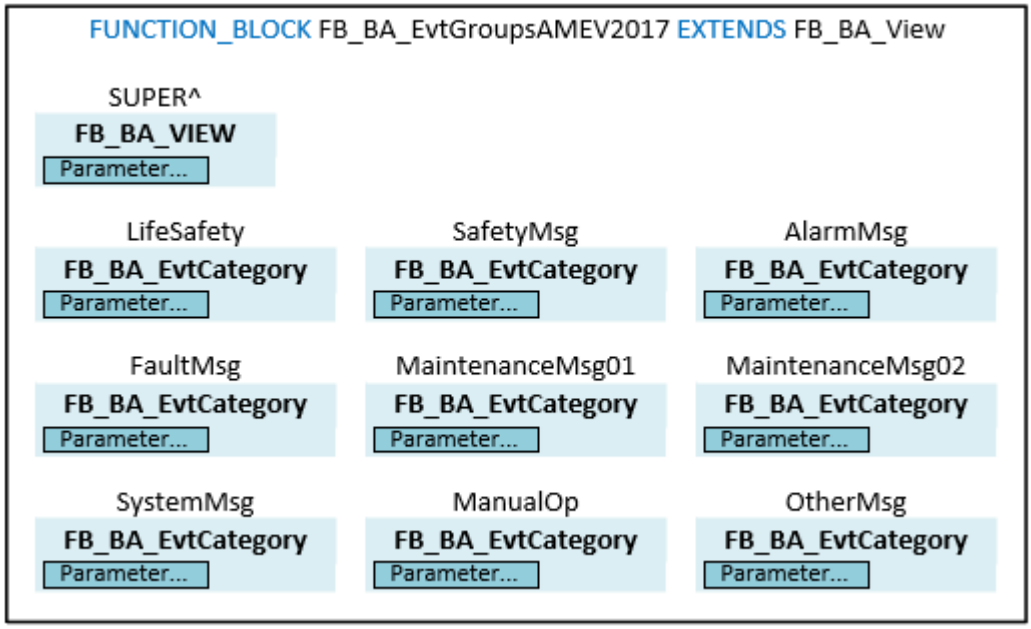
6.1.4.2.1.2.4.2 FB_BA_EvtGroupsAMEV2017



The template generates the notification classes according to the requirements of the AMEV (Arbeitskreis Maschinen- und Elektrotechnik staatlicher und kommunaler Verwaltungen, Working Group Mechanical and Electrical Engineering of Public and Municipal Administrations).

The initialization of the template takes place within the method FB_Init.

Block diagram



Notification class list

Event category	Meaning	Priority	Notification class	NC object EC object	Example
Hazard notification (Life Safety)	Danger to life	00 - 29	EC_ID.NC10	LifeSafety.EC	Fire alarm, robbery
Hazard notification (Property Safety)	Safety message	30 - 59	EC_ID.NC20	SafetyMsg.EC	Burglary, unauthorized entry
Alarm message	Message indicates system failure or requires immediate intervention.	60 - 89	EC_ID.NC30	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.NC40	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.NC50	MaintenanceMsg01.EC	Operating hours, tank level, repair switch, etc.
Maintenance message	Indication of maintenance activity or similar.	120 - 149	EC_ID.NC51	MaintenanceMsg02.EC	Filter end reached, filter dirty, etc.
System message	Fault message from the GA system.	150 - 219	EC_ID.NC60	SystemMsg.EC	Device malfunction, battery message, communication interruption, etc.
Manual message	Manual intervention	220	EC_ID.NC70	ManualOp.EC	Manual intervention
Subject to confirmation	Other messages	221 - 255	EC_ID.NC80	OtherMsg.EC	Operating state change, operation modes, trend memory full, etc.

Syntax

```
FUNCTION_BLOCK FB_BA_EvtGroupsAMEV2017 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 [▶ 1002]	The template calls the "Danger to life" event category.
SafetyMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the "Danger message, safety message" event category.
AlarmMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the "Alarm message" event category.
FaultMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the "Fault message" event category.
MaintenanceMsg01	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "Maintenance message".
MaintenanceMsg02	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "Maintenance message".
SystemMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "System message".
ManualOp	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "Manual intervention".
OtherMsg	FB_BA_EvtCategory [▶ 1002]	The template calls the event category "Other messages".

Requirements

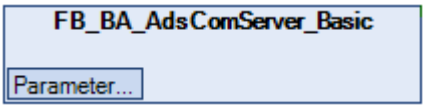
Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3 BuildingAutomation

Templates for cross-system functions.

6.1.4.2.1.3.1 Communication

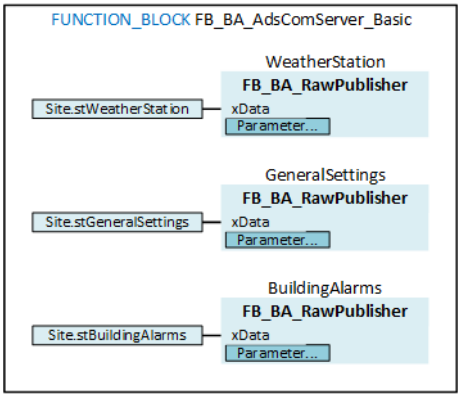
6.1.4.2.1.3.1.1 FB_BA_AdsComServer_Basic



The purpose of the template is to read data and parameters assigned to the "General Building Automation" trade from the global variable list Site [▶ 1116] and make them available within the BA network via FB_BA_RawPublisher [▶ 139].

The counter-block that receives this data as a client is FB_BA_AdsComClient_Basic [▶ 848].

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComServer
VAR_INPUT CONSTANT
  WeatherStation     : FB_BA_RawPublisher;
  GeneralSettings    : FB_BA_RawPublisher;
  BuildingAlarms     : FB_BA_RawPublisher;
END_VAR
```

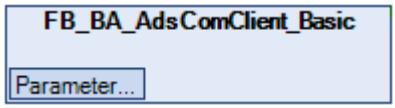
VAR_INPUT CONSTANT

Name	Type	Description
WeatherStation	FB_BA_RawPublisher [▶ 139]	The publisher *WeatherStation* publishes the structure of the weather station data Site.stWeatherStation. [▶ 1116]
GeneralSettings	FB_BA_RawPublisher [▶ 139]	The publisher *GeneralSettings* publishes the structure of the weather parameters Site.stSettings [▶ 1116], which are created in the template FB_BA_WeatherData [▶ 862].
BuildingAlarms	FB_BA_RawPublisher [▶ 139]	The publisher *BuildingAlarms* publishes the structure of the building-wide alarms Site.stBuildingAlarms [▶ 1116], which are created in the templates FB_BA_BurglarAlarmSystem and FB_BA_FireAlarmSystem.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

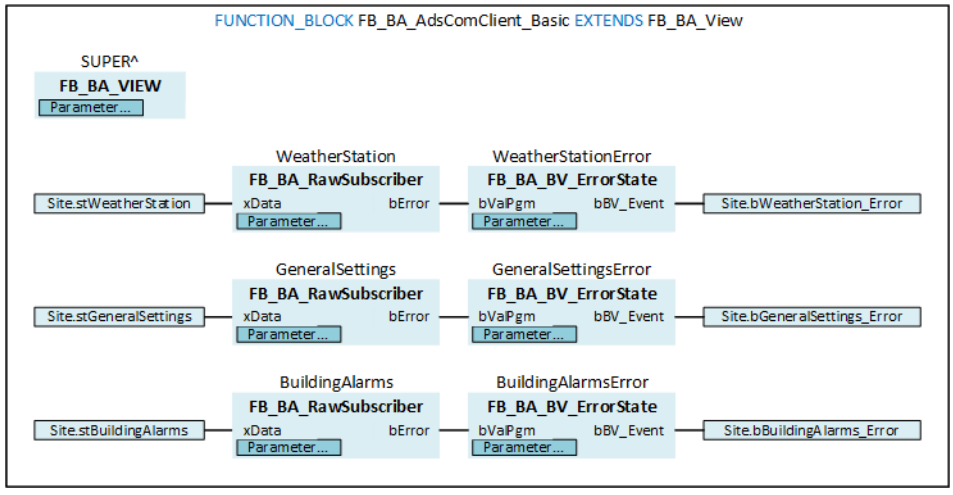
6.1.4.2.1.3.1.2 FB_BA_AdsComClient_Basic



The template reads global data assigned to the "General building automation" trade from another controller, which makes this data available with the counter-block FB_BA_AdsComServer_Basic [▶ 847]. The information read is copied to the global variable list Site [▶ 1116].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  WeatherStation          : FB_BA_RawSubscriber;
  WeatherStationError     : FB_BA_BV_ErrorState;
  GeneralSettings         : FB_BA_RawSubscriber;
  GeneralSettingsError    : FB_BA_BV_ErrorState;
  BuildingAlarms          : FB_BA_RawSubscriber;
  BuildingAlarmsError     : FB_BA_BV_ErrorState;
END_VAR
```

VAR_INPUT CONSTANT

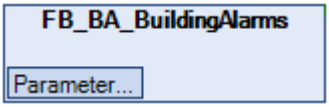
Name	Type	Description
WeatherStation	FB_BA_RawSubscriber [▶ 158]	The subscriber accesses the TwinCAT network structure *WeatherStation* and saves the data in the created structure *stWeatherStation* of the GVL Site [▶ 1116].
WeatherStationError	FB_BA_BV_ErrorState [▶ 1030]	Binary object indicating communication failure of the subscriber *WeatherStation*.
GeneralSettings	FB_BA_RawSubscriber [▶ 158]	The subscriber accesses the TwinCAT network structure *GeneralSettings* and saves the data in the created structure *stGeneralSettings* of the GVL Site [▶ 1116].
GeneralSettingsError	FB_BA_BV_ErrorState [▶ 1030]	Binary object indicating communication failure of the subscriber *GeneralSettings*.
BuildingAlarms	FB_BA_RawSubscriber [▶ 158]	The subscriber accesses the TwinCAT network structure *BuildingAlarms* and saves the data in the created structure *stBuildingAlarms* of the GVL Site [▶ 1116].
BuildingAlarmsError	FB_BA_BV_ErrorState [▶ 1030]	Binary object indicating communication failure of the subscriber *BuildingAlarms*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.3.2 GlobalEventsAndSettings

6.1.4.2.1.3.2.1 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 [▶ 1116]).

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.

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
```

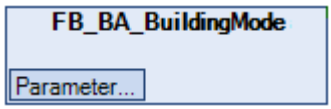
Inputs CONSTANT

Name	Type	Description
FireAlert	FB_BA_BI_IO [▶ 205]	Binary input object "Fire alert".
Bgly	FB_BA_BI_IO [▶ 205]	Binary input object "Burglary".
CentSwiOff	FB_BA_BV_Op [▶ 216]	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

6.1.4.2.1.3.2.2 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.

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 [▶ 239]	The Multistate-Value object represents an operating modes switch with the operating modes Auto, Manual-Off and Manual-On.
OpModPr	FB_BA_MV_Op [▶ 239]	The Multistate-Value object indicates the state of the currently valid plant operation mode.
Sched	FB_BA_SchedM [▶ 225]	Schedule object (automatic) for the "BuildingEnergyLevel".

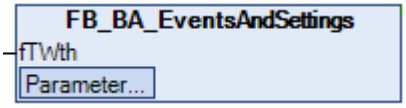
Variables

Name	Type	Description
DeMuxManMode	FB_BA_DMUX_B04 [▶ 637]	Conversion of the multistate value of the manual selection to a binary output.
DeMuxSched	FB_BA_DMUX_B04 [▶ 426]	Conversion of the multistate value of the schedule to a binary output.
PrioSwi	FB_BA_PrioSwi_UDI08 [▶ 433]	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

6.1.4.2.1.3.2.3 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 [▶ 849] is used to record global events that are required in a building or property at all automation stations of the system.

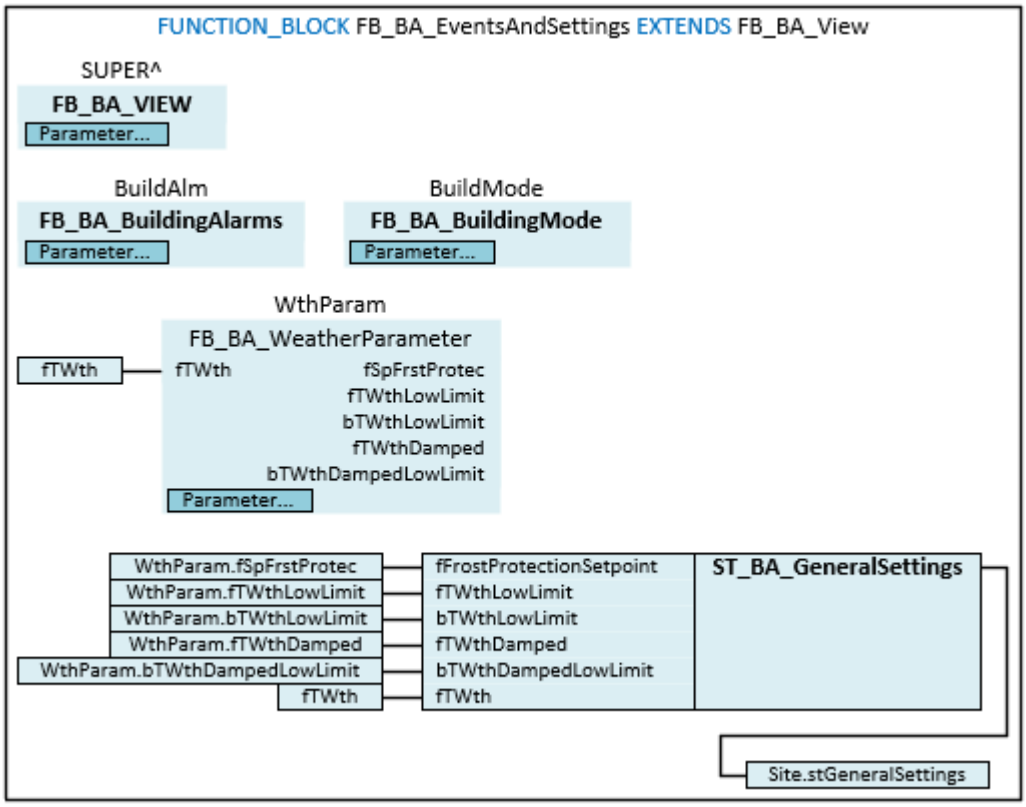
The function block FB_BA_WeatherParameter is used for the global provision of weather-specific parameters and setpoints.

The function block FB_BA_BuildingMode [▶ 850] is used for the global provision of building operating modes.

The global building data is stored in the GVL site [▶ 1116].

The initialization of the template takes place within the method FB_Init.

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	Current value of the outside temperature.

Inputs CONSTANT

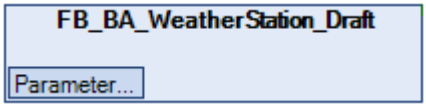
Name	Type	Description
BuildAlm	FB_BA_BuildingAlarms [▶ 849]	Template for the building alarms.
BuildMode	FB_BA_BuildingMode [▶ 850]	Template for the building operation modes.
WthParam	FB_BA_WeatherParameter [▶ 863]	Template for the evaluation of the outside temperature.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

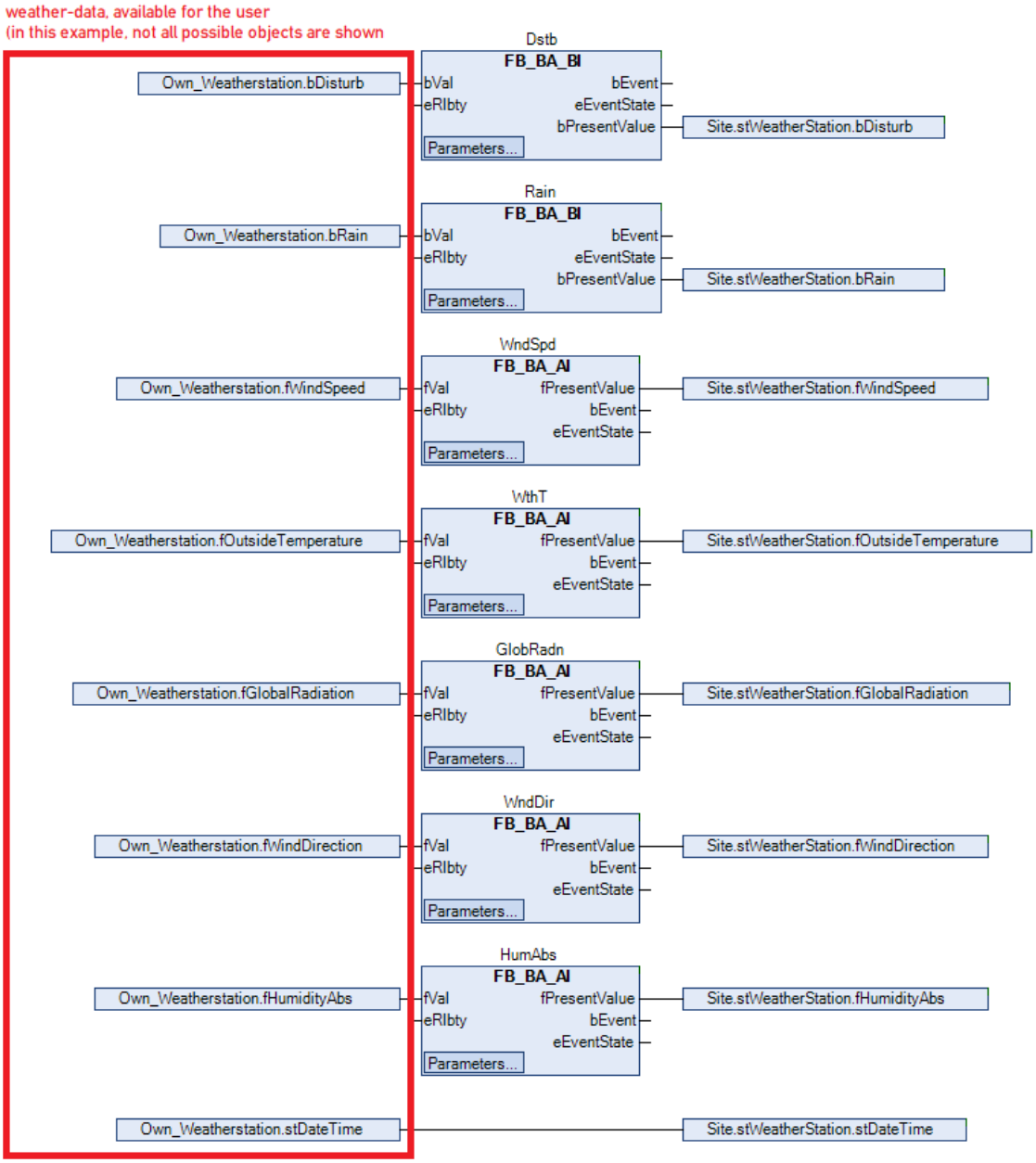
6.1.4.2.1.3.3 WeatherStation

6.1.4.2.1.3.3.1 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 [▶ 1116]).



The initialization of the template takes place within the method FB_Init.

Syntax

```
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 [▶ 204]	The weather station reports a malfunction.
Rain	FB_BA_BI [▶ 204]	Rain sensor.
WthT	FB_BA_AI [▶ 191]	Outside temperature [°C].
DewPtT	FB_BA_AI [▶ 191]	Dew point temperature [°C].
PrssAbs	FB_BA_AI [▶ 191]	Absolute air pressure [hPa].
PrssRel	FB_BA_AI [▶ 191]	Relative air pressure [hPa].
HumAbs	FB_BA_AI [▶ 191]	Absolute humidity [g/m³].
HumRel	FB_BA_AI [▶ 191]	Relative Absolute Humidity [g/m³].
Brightness_N	FB_BA_AI [▶ 191]	Directional light sensor north [Lux].
Brightness_S	FB_BA_AI [▶ 191]	Directional light sensor south [Lux].
Brightness_E	FB_BA_AI [▶ 191]	Directional light sensor east [Lux].
Brightness_W	FB_BA_AI [▶ 191]	Directional light sensor west [Lux].
Dawn	FB_BA_AI [▶ 191]	Dawn [Lux].
GlobRadn	FB_BA_AI [▶ 191]	Global radiation [W/m²].
WndDir	FB_BA_AI [▶ 191]	Wind direction [°].
WndSpd	FB_BA_AI [▶ 191]	Wind speed [m/s]
Latd	FB_BA_AI [▶ 191]	Latitude of the installation site [°].
Lngt	FB_BA_AI [▶ 191]	Longitude of the installation site [°].
SunAzm	FB_BA_AI [▶ 191]	Current position of the sun [°].
SunElv	FB_BA_AI [▶ 191]	Current sun elevation [°].

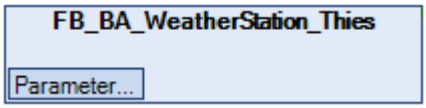
Variables

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

6.1.4.2.1.3.3.2 FB_BA_Weatherstation_Thies



This template prepares the data which are read from a Thies weather station via the function block FB_BA_ThiesWSC11 [▶ 1114] and makes them available in a globally declared variable structure *stWeatherstation* (see global variable list Site [▶ 1116]).

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 [▶ 1097] can be found here.

The initialization of the template takes place within the method FB_Init.

Syntax

```
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 [▶ 216]	Restart the serial communication if it is permanently faulty due to a configuration error or during operation.
WthStRst	FB_BA_Bv_OP [▶ 216]	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 [▶ 213]	Message requiring acknowledgement: Serial communication or configuration permanently faulty.
Dstb	FB_BA_BV [▶ 213]	The weather station reports a malfunction.
Rain	FB_BA_BI [▶ 204]	Rain sensor.
WthT	FB_BA_AI [▶ 191]	Outside temperature [°C].
DewPtT	FB_BA_AI [▶ 191]	Dew point temperature [°C].
PrssAbs	FB_BA_AI [▶ 191]	Absolute air pressure [hPa].
PrssRel	FB_BA_AI [▶ 191]	Relative air pressure [hPa].
HumAbs	FB_BA_AI [▶ 191]	Absolute humidity [g/m^3].
HumRel	FB_BA_AI [▶ 191]	Relative Absolute Humidity [g/m^3].
Brightness_N	FB_BA_AI [▶ 191]	Directional light sensor north [Lux].
Brightness_S	FB_BA_AI [▶ 191]	Directional light sensor south [Lux].
Brightness_E	FB_BA_AI [▶ 191]	Directional light sensor east [Lux].
Brightness_W	FB_BA_AI [▶ 191]	Directional light sensor west [Lux].
Dawn	FB_BA_AI [▶ 191]	Dawn [Lux].
GlobRadn	FB_BA_AI [▶ 191]	Global radiation [W/m²].
WndDir	FB_BA_AI [▶ 191]	Wind direction [°].
WndSpd	FB_BA_AI [▶ 191]	Wind speed [m/s]
Latd	FB_BA_AI [▶ 191]	Latitude of the installation site [°].
Lngt	FB_BA_AI [▶ 191]	Longitude of the installation site [°].
SunAzm	FB_BA_AI [▶ 191]	Current position of the sun [°].
SunElv	FB_BA_AI [▶ 191]	Current sun elevation [°].

Variables

Name	Type	Description
ThiesWSC11	FB_BA_ThiesWSC11 [▶ 1114]	Function block for unloading the data from the Thies WSC11
bResetWeatherStation	BOOL	Reset weather station.
bResetSerialCommunication	BOOL	Reset serial communication.
DataConversion	FB_BA_ThiesData [▶ 1115]	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 [▶ 1097]).
stTxBuff	ComBuffer	Communication variable between the *fbSerialCtrl* of the fast task and the *ThiesWSC11* of the normal PLC task.
stRxBuff	ComBuffer	Communication variable between the *fbSerialCtrl* of the fast task and the *ThiesWSC11* 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

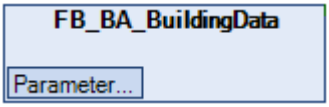
Name	Type	Description
nSerialCommErrDelay	UDINT	Error delay communication error [s].
nDstbDelay	UDINT	Error delay weather station error [s].

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.4 General

6.1.4.2.1.3.4.1 FB_BA_BuildingData

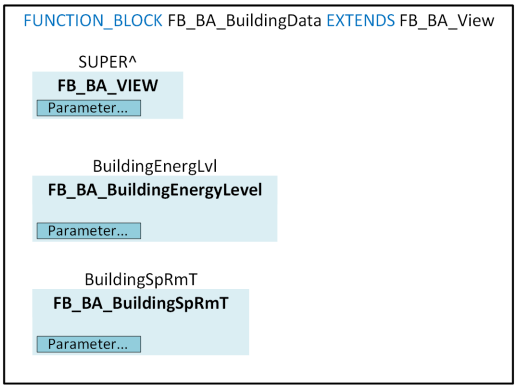


This template at system level is used to call up sub-templates that provide building-wide data and operation modes.

Typically, this template is called up in the building calculator.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
VAR_INPUT CONSTANT
  BuildingMode        : FB_BA_BuildingMode;
  BuildingEnergLvl    : FB_BA_BuildingEnergyLevel;
  BuildingSpRmT       : FB_BA_BuildingSpRmT;
  BuildingLight       : FB_BA_BuildingLighting;
END_VAR
```

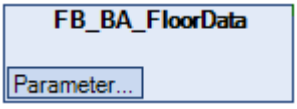
VAR_INPUT CONSTANT

Name	Type	Description
BuildingMode	FB_BA_BuildingMode [▶ 850]	The sub-template *BuildingMode* defines the building operation mode using a schedule with manual intervention.
BuildingEnergLvl	FB_BA_BuildingEnergyLevel [▶ 889]	The sub-template *BuildingEnergLvl* defines the building energy level using a schedule with manual intervention.
BuildingSpRmT	FB_BA_BuildingSpRmT [▶ 860]	The sub-template *BuildingSpRmT* is used to enter the heating/cooling room temperature setpoints for the various building energy levels. In addition, a summer and a winter compensation value for the room temperature is determined based on the outside temperature.
BuildingLight	FB_BA_BuildingLighting [▶ 917]	The sub-template *BuildingLight* compiles cross-building control telegrams for the light functions.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.4.2 FB_BA_FloorData

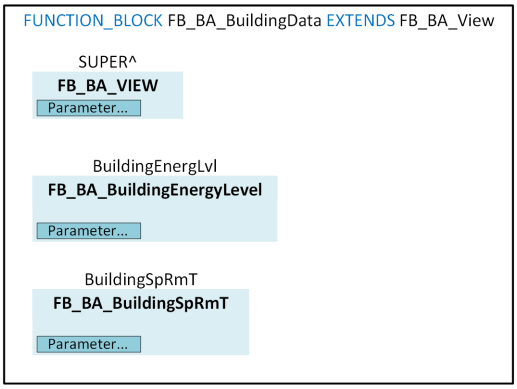


This template at system level is used to call up sub-templates that provide floor-wide data and operation modes.

This template is typically called up in the floor controllers.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
VAR_INPUT CONSTANT
  FloorLighting    : FB_BA_FloorLighting;
END_VAR
```

VAR_INPUT CONSTANT

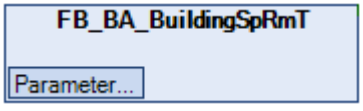
Name	Type	Description
FloorLighting	FB_BA_FloorLighting [▶ 919]	The sub-template calls up functions assigned to the illumination in the floor area.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.4.3 Functions

6.1.4.2.1.3.4.3.1 FB_BA_BuildingSpRmT

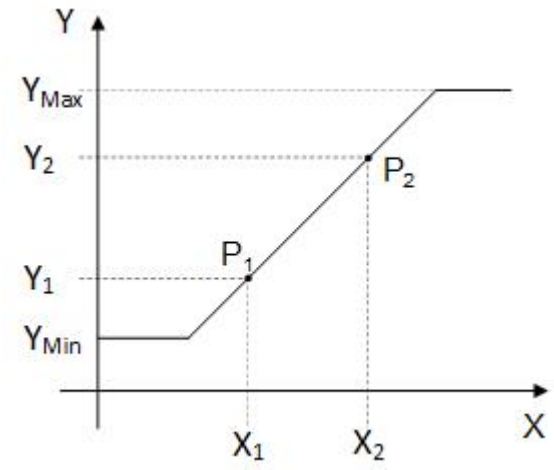


In this template the basic setpoints Protection Cooling ... Comfort Cooling and Protection Heating ... Comfort Heating are applied with correction values and made available as building-wide setpoints in a globally declared structure variable *stBuildingSpRmT* (see global variable list Site [▶ 1116]).

The correction affects the Precomfort and Comfort values respectively, see FB_BA_SpRmT [▶ 305]. These are:

- **central setpoint value shift**
  These two values, ShiftHtg and ShiftCol, are each added to the corresponding values for Precomfort and Comfort.

- **weather-compensated setpoint value shift (summer or winter compensation)**
  Using the outside temperature from the weather station data *stWeatherStation* (see global variable list Site [▶ 1116]), 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 [▶ 1042], which can be parameterized in the *FB_init* method.



The initialization of the template takes place within the method FB_Init.

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;
  ShiftCol     : FB_BA_AV_Op;
END_VAR

VAR
  SpRmT        : FB_BA_SpRmT;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
SumCpsn	FB_BA_Scale_02 [▶ 1042]	Value Summer compensation [K]
WinCpsn	FB_BA_Scale_02 [▶ 1042]	Value Winter compensation [K]
PrtcHtg	FB_BA_AV_Op [▶ 202]	Basic setpoint Protection Heating [°C]
EcoHtg	FB_BA_AV_Op [▶ 202]	Basic setpoint Economy Heating [°C]
PreCmfHtg	FB_BA_AV_Op [▶ 202]	Basic setpoint Precomfort Heating [°C]
CmfHtg	FB_BA_AV_Op [▶ 202]	Basic setpoint Comfort Heating [°C]
PrtcCol	FB_BA_AV_Op [▶ 202]	Basic setpoint Protection Cooling [°C]
EcoCol	FB_BA_AV_Op [▶ 202]	Basic setpoint Economy Cooling [°C]
PreCmfCol	FB_BA_AV_Op [▶ 202]	Basic setpoint Precomfort Cooling [°C]
CmfCol	FB_BA_AV_Op [▶ 202]	Basic setpoint Comfort Cooling [°C]
ShiftHtg	FB_BA_AV_Op [▶ 202]	Setpoint value shift Heating [K]
ShitfCol	FB_BA_AV_Op [▶ 202]	Setpoint value shift Cooling [K]

Variables

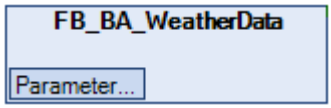
Name	Type	Description
SpRmT	FB_BA_SpRmT [▶ 305]	Function block for calculating the applied temperature setpoints.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.5 WeatherData

6.1.4.2.1.3.5.1 FB_BA_WeatherData



The template is used to prepare and distribute global data that is required throughout the system or building to control and regulate the system automation.

The function block FB_BA_WeatherParameter [▶ 863] is used for the global provision of weather-specific parameters and setpoints.

The global building data is stored in the GVL Site [▶ 1116].

The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

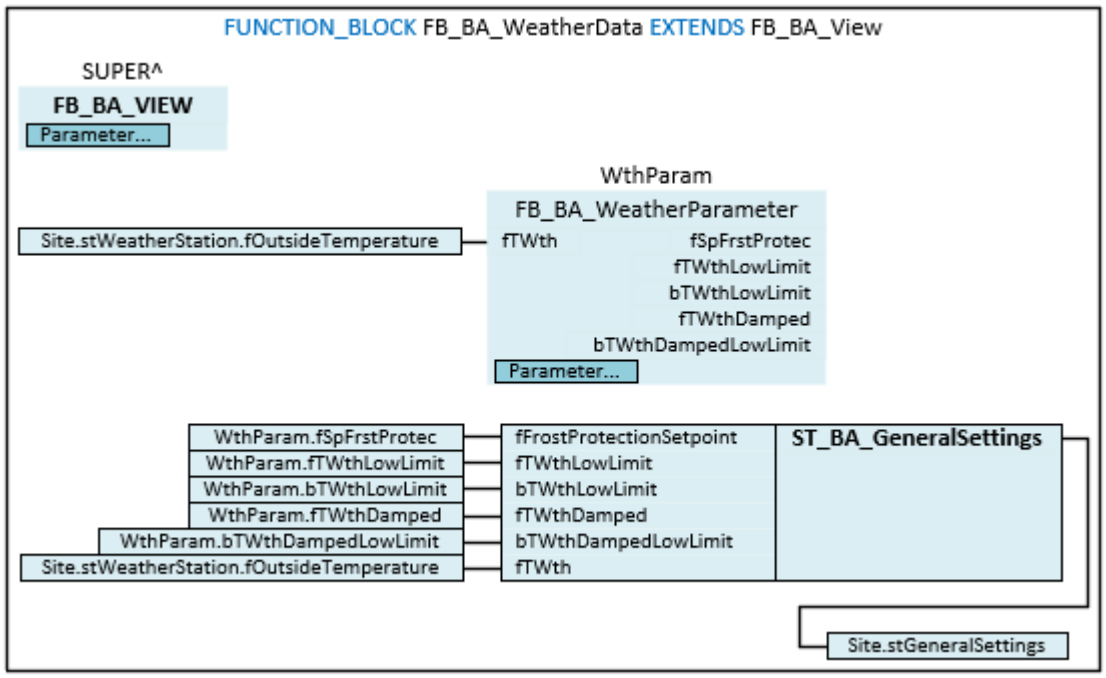
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_View [▶ 241]

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_WeatherData EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  WthParam    : FB_BA_WeatherParameter;
END_VAR
```

Inputs CONSTANT

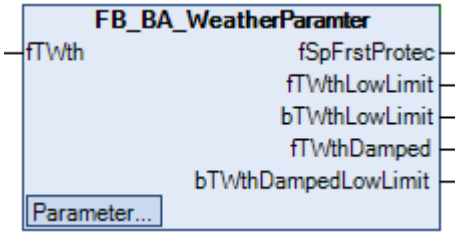
Name	Type	Description
WthParam	FB_BA_WeatherParameter [▶ 863]	Sub-template for evaluating and creating weather parameters.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.5.2 Parameter

6.1.4.2.1.3.5.2.1 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.

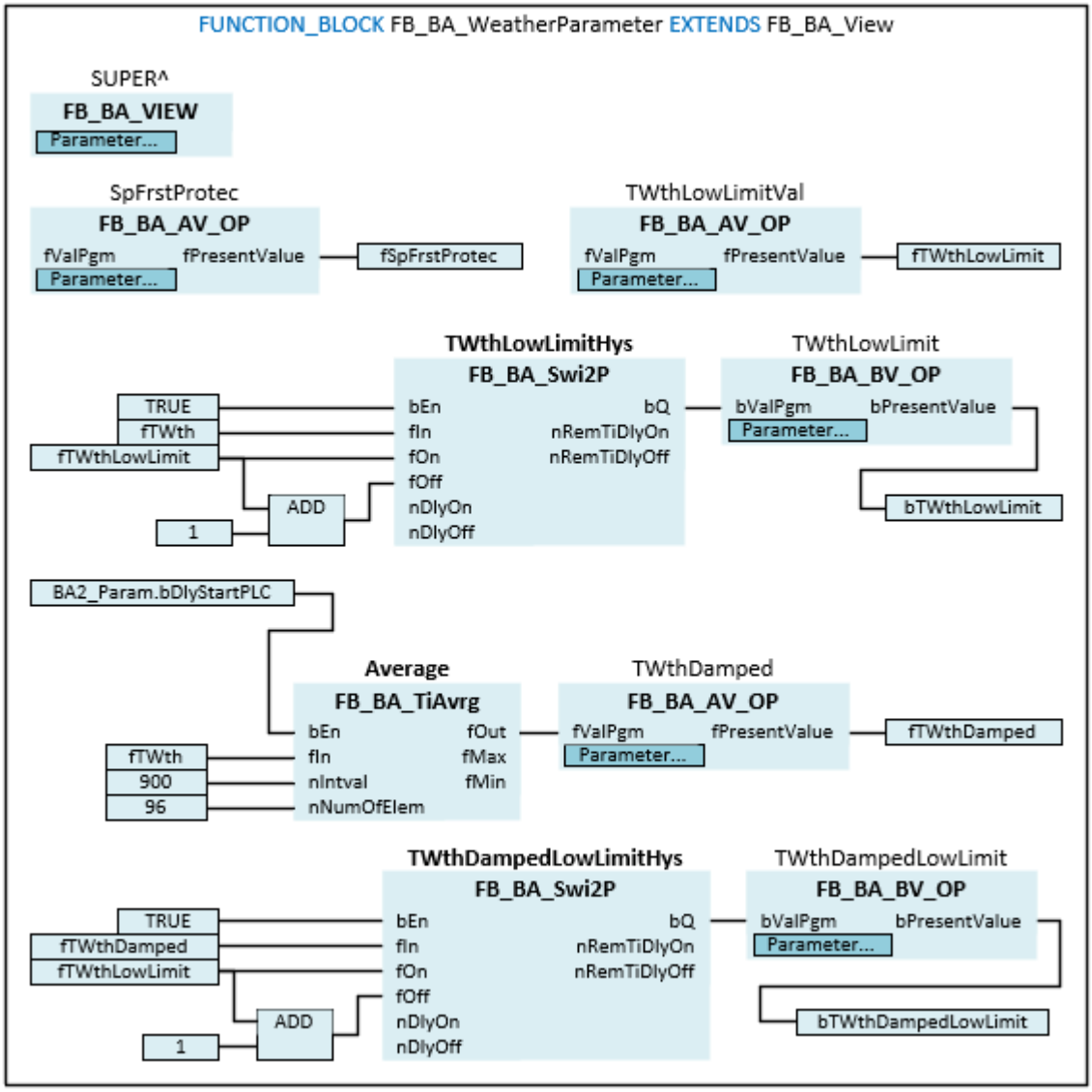
The function block *TWthLowLimitHys* calculates whether the outside temperature is below the critical value of *TWthLowLimitVal*. Depending on the value of the variable *bTWthLowLimit*, the frost protection function of the HVAC systems in the building is activated.

The weather-related enable of the heating systems is not dependent on the current, but on the damped outside temperature. The outside temperature is attenuated using the function block *Average* (formation of a sliding average value). 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 "TWthDampedLowLimit" and written to the variable *bTWthDampedLowLimit* for further processing in other templates.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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	Current 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 *Outside temperature lower limit*. The variable is TRUE if the outside temperature is below the value of *fTWthLowCrit*.
fTWthDamped	REAL	Current value of the damped outside temperature.
bTWthDampedLowLimit	BOOL	The output shows the operating message *Outside temperature damped lower limit*. The variable is TRUE if the damped outside temperature is below the value of *fTWthLowCrit*.

Inputs CONSTANT

Name	Type	Description
SpFrstProtec	FB_BA_AV_Op [▶ 202]	Analog value object for entering the frost protection setpoint, e.g. for heating circuits in protection mode.
TWthLowLimitVal	FB_BA_AV_Op [▶ 202]	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 [▶ 216]	Binary object displaying the *Outside temperature lower limit* operating message.
TWthDamped	FB_BA_AV_Op [▶ 202]	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 [▶ 216]	Binary object displaying the *Outside temperature damped lower limit* operating message.

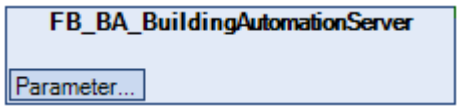
Variables

Name	Type	Description
TWthLowLimitHys	FB_BA_Swi2P	Two-point switch which converts the outside temperature *fWth* into a binary switching signal for *TWthLowLimit*.
Average	FB_BA_TiAvrg [▶ 464]	The Average uses the outside temperature *fWth* to determine the damped outside temperature *fTWthDamped*.
TWthDampedLowLimitHys	FB_BA_Swi2P	Two-point switch which converts the damped outside temperature *fTWthDamped* into a binary switching signal for *TWthDampedLowLimit*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.3.6 FB_BA_BuildingAutomationServer



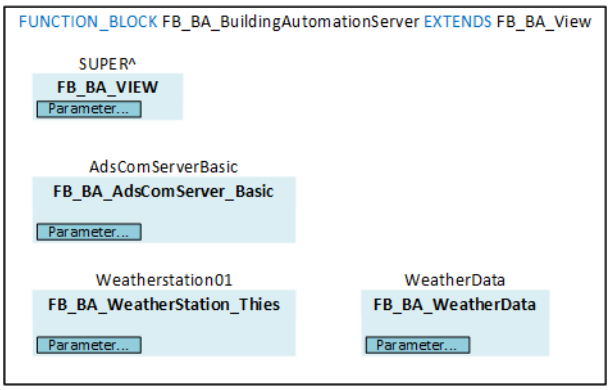
Call template trade "Building automation" - server version.

This template at the "trade" level calls up all sub-templates whose data is ideally processed in a building calculator.

In addition, an ADS server template FB_BA_AdsComServer_Basic [▶ 847] makes basic data available throughout the building.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_BuildingAutomationServer EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComServerBasic    : FB_BA_AdsComServer_Basic;
  Weatherstation01     : FB_BA_WeatherStation_Thies;
  WeatherData          : FB_BA_WeatherData;
END_VAR
```

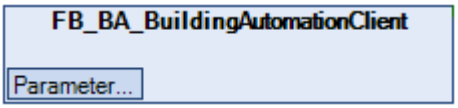
VAR_INPUT CONSTANT

Name	Type	Description
AdsComServerBasic	FB_BA_AdsComServer_Basic [▶ 847]	The sub-template has the task of transmitting data and parameters within the BA network.
Weatherstation01	FB_BA_WeatherStation_Thies [▶ 855]	The sub-template is used for data exchange with a weather station type WSC11 from Thies. The suffix "01" at the instance name is intended to indicate that it is quite possible that a building may have multiple weather stations, which should be entered here.
WeatherData	FB_BA_WeatherData [▶ 862]	The sub-template is used to prepare and distribute global data that is required throughout the system or building for the control and regulation of the system automation.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.3.7 FB_BA_BuildingAutomationClient



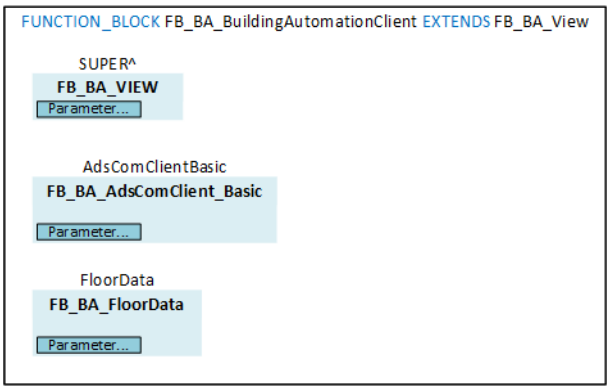
Call template trade "Building automation" - client version.

This template at the "trade" level calls up all sub-templates whose data is ideally prepared in a floor computer.

These are floor data from the FloorData [▶ 859] template as well as basic data that is transmitted and read by a building computer (FB_BA_AdsComClient_Basic [▶ 848]).

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_BuildingAutomation EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComClientBasic    : FB_BA_AdsComClient_Basic;
  FloorData            : FB_BA_FloorData;
END_VAR
```

VAR_INPUT CONSTANT

Name	Type	Description
AdsComClientBasic	FB_BA_AdsComClient_Basic [▶ 848]	The sub-template has the task of reading data and parameters sent from other computers, ideally from the building computer.
FloorData	FB_BA_FloorData [▶ 859]	The sub-template calls up further templates that provide floor-wide data and operation modes.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4 RoomAutomation

Templates for creating room automation solutions.

6.1.4.2.1.4.1 Common

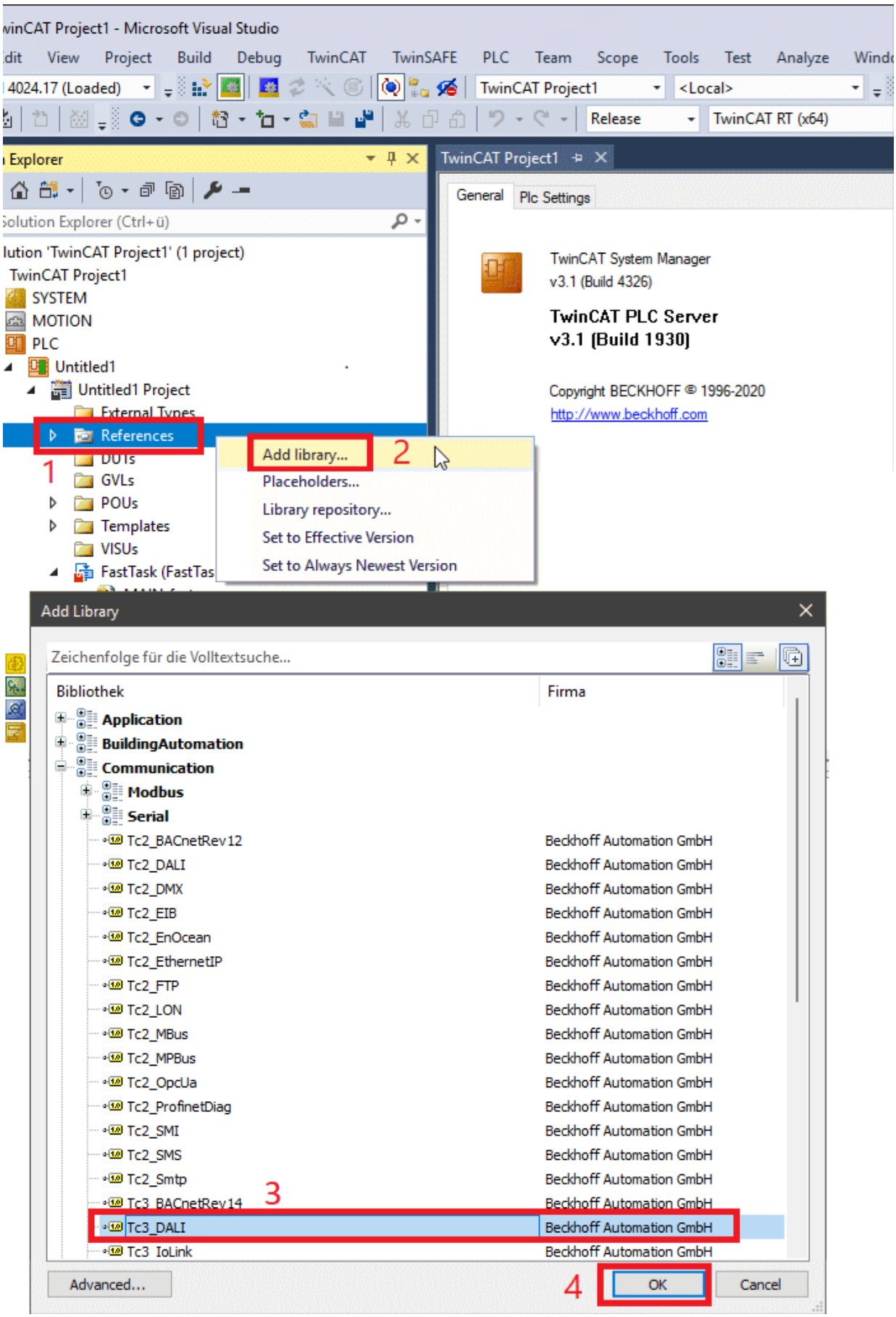
6.1.4.2.1.4.1.1 DALIFunctionBlocks

6.1.4.2.1.4.1.1.1 DALI communication

These instructions and the DALI templates provided refer to the application with a KL6821 in conjunction with the library TC3_DALI.

Adding the library

Simply adding DALI-based templates to a project does not automatically include the required TC3_DALI library. This must be inserted manually:



Right-click on "References" (1) and select "Add Library" (2). In the window that opens, select the library "TC3_DALI" (3) and confirm with "OK" (4).

Communication blocks and fast task

All DALI-based templates require the communication block FB_KL6821Communication, which runs in a fast communication task and establishes the connection to the DALI terminal to which the DALI line of the relevant device is connected.

The exact procedure can be found in the Beckhoff Information System within the description of the TC3_DALI.

One communication block must be instantiated for each DALI line. Once these have been inserted and called up in the fast task, they are available to the DALI-based templates.

Linking in the project

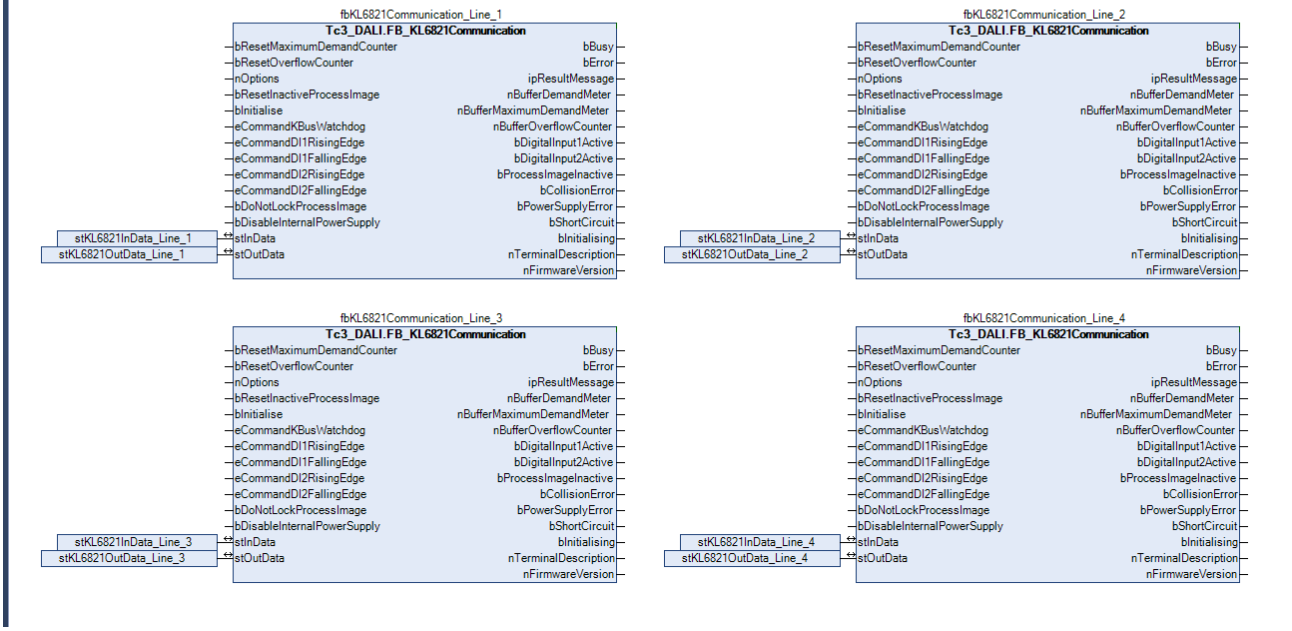
The data exchange between the communication block and the DALI-based templates takes place via an interface provided by the communication block. All DALI-based templates have implemented this interface as a parameter (VAR_INPUT CONSTANT PERSISTENT), so that only the respective communication block needs to be linked there in FB_Init.

Example:

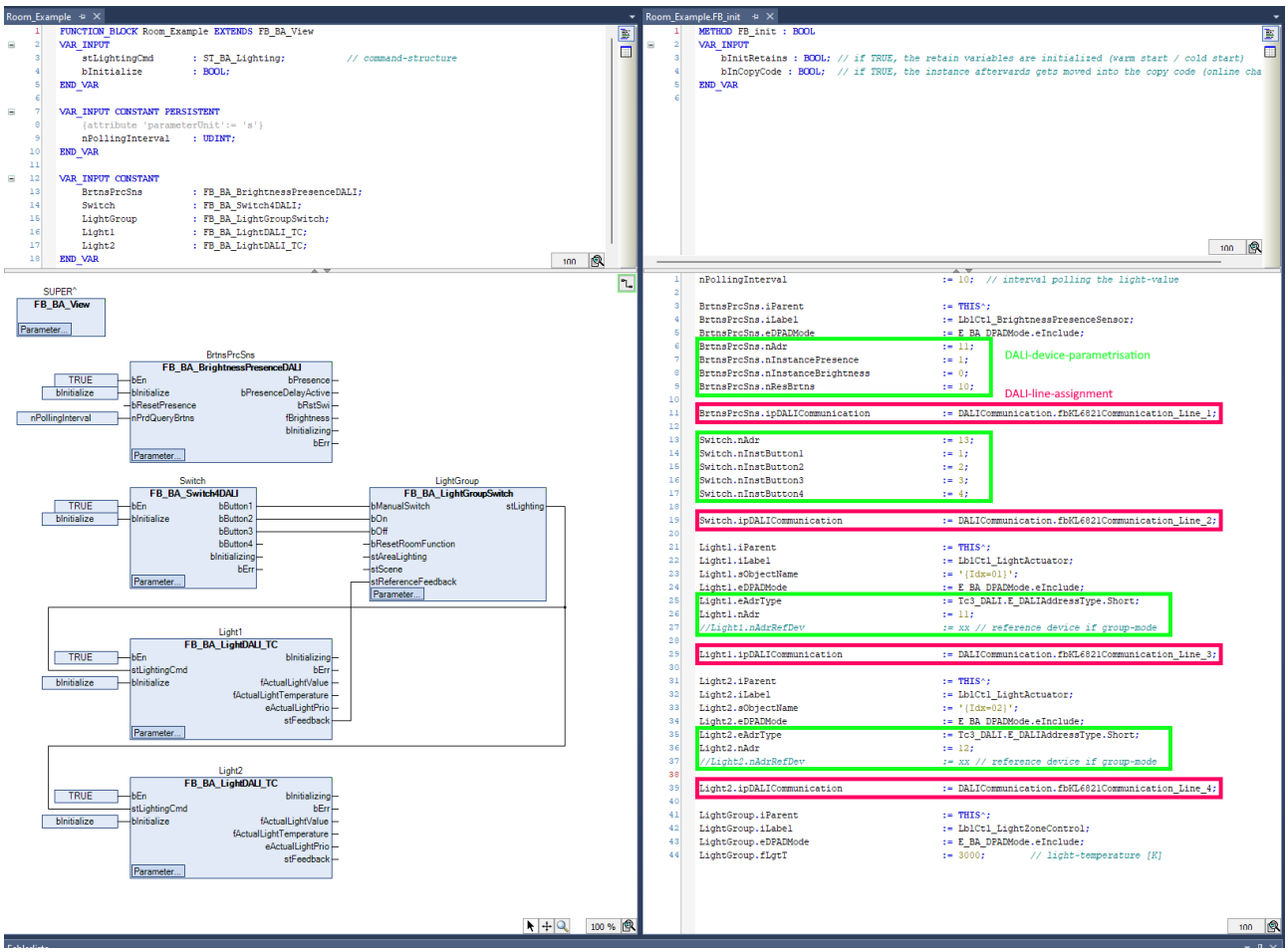
Communication blocks are called up for 4 DALI lines in a fast task:

DALICommunication

```
PROGRAM DALICommunication
VAR
    stKL6821InData_Line_1          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_1         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_1           : Tc3_DALI.FB_KL6821Communication;

    stKL6821InData_Line_2          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_2         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_2           : Tc3_DALI.FB_KL6821Communication;

    stKL6821InData_Line_3          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_3         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_3           : Tc3_DALI.FB_KL6821Communication;

    stKL6821InData_Line_4          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_4         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_4           : Tc3_DALI.FB_KL6821Communication;

END_VAR
```



DALI templates are then called up in a room template. These are pre-initialized in the FB_Init of the room template, including the link to the corresponding DALI communication block:



6.1.4.2.1.4.1.1.2 FB_BA_BrtnsPrcDALI

This function block is used to read a combined DALI brightness and occupancy sensor.

For information on the function of the DALI device used, please refer to the manufacturer's device documentation.

Function

The device is specified by the instance number of the presence, *nInstancePresence*, and the instance number of the brightness, *nInstanceBrightness*. Added to this is the device-specific resolution, *nResBrtns*. These three parameters vary from manufacturer to manufacturer.

The brightness is queried cyclically (interval *nPrdQueryBrtns* [s]). With presence, on the other hand, the device's internal notification system is used, which automatically sends a telegram as soon as presence or absence is detected.

The DALI brightness and presence function block can be initialized via the input *bInitialize* with partly parameterizable (see VAR_INPUT CONSTANT PERSISTENT) and partly **fixed pre-set** values (see VAR CONSTANT).

Syntax

```
FUNCTION_BLOCK FB_BA_BrtnsPrcDALI
VAR_INPUT
  bEn                   : BOOL;
  bInitialize           : BOOL;
  nPrdQueryBrtns        : UDINT;
  nAdr                  : BYTE;
  nInstPrc              : BYTE;
  nInstBrtns            : BYTE;
  nResBrtns             : BYTE := 10;
  ipDALICommunication   : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_OUTPUT
  bPrc                  : BOOL;
  nBrtns                : UINT;
  bInitializing         : BOOL;
  bErr                  : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nDeadtime             : BYTE := 2;
  nHold                 : BYTE := 90;
  nReport               : BYTE := 60;
END_VAR
VAR CONSTANT
  nEventFilter          : DWORD := 7;
  eEventPriority        : Tc3_DALI.E_DALIEventPriority := Tc3_DALI.E_DALIEventPriority.Middle;
  eEventScheme          : Tc3_DALI.E_DALIEventScheme := Tc3_DALI.E_DALIEventScheme.DeviceInstance;
END_VAR
```

VAR_INPUT

Name	Type	Description
bEn	BOOL	Enabling the function block: a TRUE signal at this input enables the function.
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the DALI device, see FB_BA_Swi4DALI [▶ 880].
nPrdQueryBrtns	UDINT	Brightness query interval in seconds. The declaration of this variable as an input is intended to enable a variable query speed. In the case of constant light regulation, for example, a slow query interval is initially sufficient until a larger deviation to be compensated is detected, at which point a shorter interval is used. A "0" at this input disables the query.
nAdr	BYTE	DALI short address of the sensor.
nInstPrc	BYTE	Vendor-specific number of the presence instance to be queried.
nInstBrtns	BYTE	Vendor-specific number of the brightness instance to be queried.
nResBrtns	BYTE	Vendor-specific brightness resolution.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

VAR_OUTPUT

Name	Type	Description
bPrc	BOOL	Occupancy signal output.
nBrtns	BOOL	Measured brightness. The unit is vendor-specific.
bInitializing	BOOL	The sensor is in the DALI initialization phase, i.e. the entered parameters are transferred to the DALI device.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.

VAR_INPUT CONSTANT PERSISTENT

The following parameters are transferred to the DALI control gear or the DALI devices by the initialization routine.

Name	Type	Description
nDeadtime	BYTE	Dead time of the occupancy sensor as a multiple of 50 ms, pre-set to 2 (=100 ms).
nHold	BYTE	Hold time of the occupancy sensor as a multiple of 10 s, pre-set to 90 (=900 s).
nReport	BYTE	Report time of the occupancy sensor in seconds, pre-set to 60 s.

VAR CONSTANT

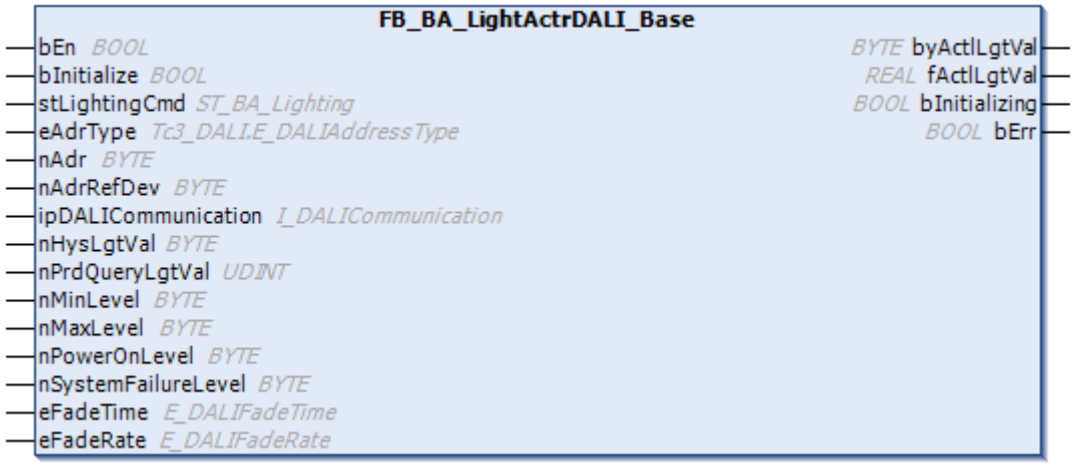
The following parameters cannot be changed and are transferred to the DALI device by the initialization routine.

Name	Type	Description
nEventFilter	DWORD	DALI event filter mask for occupancy sensors. This value is set to 7 and thus queries the *Occupied*, *Vacant* and *Repeat Event* states.
eEventPriority	Tc3_DALI.E_DALIEventPriority	DALI event priority with which input notification events are sent by the instance of the DALI control device, set to *Middle*.
eEventScheme	Tc3_DALI.E_DALIEventScheme	DALI event scheme. This is set to *DeviceInstance* (device/instance addressing with short address and instance number).

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_LightActrDALI_Base



This function block is used to control a DALI light actuator, a DALI group or DALI devices with a broadcast, whereby only the light brightness value is changed.

Function

Two routines are executed in the function block:

- Setting the light value if it deviates from the current value. This is subjected to a hysteresis range *nHysLgtVal*, see below.
- Reading the light value. This is done cyclically at the interval *nPrdQueryLgtVal* in seconds.

The "current value" is always initially assumed to be the value that was sent without errors. The cyclic query of the read function will correct the value if a telegram is nevertheless "lost" or the control gear has been controlled in another way.

This procedure ensures that, on the one hand, the DALI bus is not overloaded, but on the other hand, the actual light value is always read once.

The function block is controlled via values from the light telegram stLightingCmd [▶ 272]. The control values of the light are limited to the limits that were previously transferred to the control gear or the devices by an initialization routine with other configuration parameters, see VAR_INPUT CONSTANT PERSISTENT. Initialization takes place via the input *bInitialize*. **Fixed** values are also transferred here, see VAR CONSTANT.

The light value is set internally exclusively via the DALI command FB_DALI102DirectArcPowerControl. This means that the FadeTime is the time taken to dim to the new light value.

This dimming time is pre-set to 0 s so that dimming ramps can be realized in the PLC alone.

If group or broadcast control is selected, the control and initialization commands are directed to all devices. Read commands, on the other hand, are only sent to the reference device.

Syntax

```
FUNCTION_BLOCK FB_BA_LightActrDALI_Base
VAR_INPUT
  bEn                           : BOOL;
  bInitialize                   : BOOL;
  stLightingCmd                 : ST_BA_Lighting;
  eAdrType                      : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                          : BYTE;
  nAdrRefDev                    : BYTE;
  ipDALICommunication           : I_DALICommunication;
END_VAR
VAR_OUTPUT
  byActlLgtVal                  : BYTE;
  fActlLgtVal                   : REAL;
  bInitializing                 : BOOL;
  bErr                          : BOOL;
END_VAR
```

```
VAR_INPUT CONSTANT PERSISTENT
  nHysLgtVal                    : BYTE;
  nPrdQueryLgtVal               : UDINT;

  nMinLevel                     : BYTE := 126;
  nMaxLevel                     : BYTE := 254;
  nPowerOnLevel                 : BYTE := 254;
  nSystemFailureLevel           : BYTE := 254;
  eFadeTime                     : Tc3_DALI.E_DALIFadeTime := Tc3_DALI.E_DALIFadeTime.Disabled;
  eFadeRate                     : Tc3_DALI.E_DALIFadeRate := Tc3_DALI.E_DALIFadeRate.N358StepsPerSe
c;
END_VAR
VAR CONSTANT
  eExtendedFadeTimeBase         : Tc3_DALI.E_DALIExtendedFadeTimeBase := Tc3_DALI.E_DALIExtendedFad
eTimeBase.Base01;
  eExtendedFadeTimeMultiplier   : Tc3_DALI.E_DALIExtendedFadeTimeMultiplier := Tc3_DALI.E_DALIExten
dedFadeTimeMultiplier.Disabled;
END_VAR
```

VAR_INPUT

Name	Type	Description
bEn	BOOL	Enabling the function block: a TRUE signal at this input enables the function.
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the DALI device, see FB_BA_Swi4DALI [▶ 880].
stLightingCmd	ST_BA_Lighting [▶ 272]	Light control telegram
eAdrType	Tc3_DALI.E_DALIAddressType	Defines whether the input *nAdr* contains a short address (0...63) or a group address (0...31). The input *nAdr* has no meaning if a broadcast is sent
nAdr	BYTE	DALI short address of the sensor.
nAdrRefDev	BYTE	DALI address of a reference device if group or broadcast control is selected. This reference device then represents all other controlled devices.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

VAR_OUTPUT

Name	Type	Description
byActlLgtVal	BYTE	Current light value in DALI format (0...254).
fActlLgtVal	REAL	Current light value in percent.
bInitializing	BOOL	The sensor is in the DALI initialization phase, i.e. the entered parameters are transferred to the DALI device.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.

VAR_INPUT CONSTANT PERSISTENT

Name	Type	Description
nHysLgtVal	BYTE	Hysteresis range around the light control value. A new set command is started if the setpoint *nHysLgtVal/2* is above or below the actual value.
nPrdQueryLgtVal	UDINT	Query interval of the light value [s].

The following parameters are transferred to the DALI control gear or the DALI devices by the initialization routine.

Name	Type	Description
nMinLevel	BYTE	Minimum light control value in DALI output format (0...254). Pre-set to 126.
nMaxLevel	BYTE	Maximum light control value in DALI output format (0...254). Pre-set to 254.
nPowerOnLevel	BYTE	Light control value when the supply voltage is applied to the control gear. Pre-set to 254.
nSystemFailureLevel	BYTE	Light control value in the event of an error on the DALI bus. Pre-set to 254.
eFadeTime	Tc3_DALI.E_DALIFadeTime	The FadeTime defines the time the current output value takes to reach the required value. The FadeTime is pre-set to "disabled", which together with the deactivated "Extended FadeTime" (VAR CONSTANT) corresponds to a fade time of 0 s.
eFadeRate	Tc3_DALI.E_DALIFadeRate	The Fade Rate determines the rate of change (in steps per second) of the output value. It is not decisive for the behavior of this function block. It is pre-set to 358 steps/s.

VAR CONSTANT

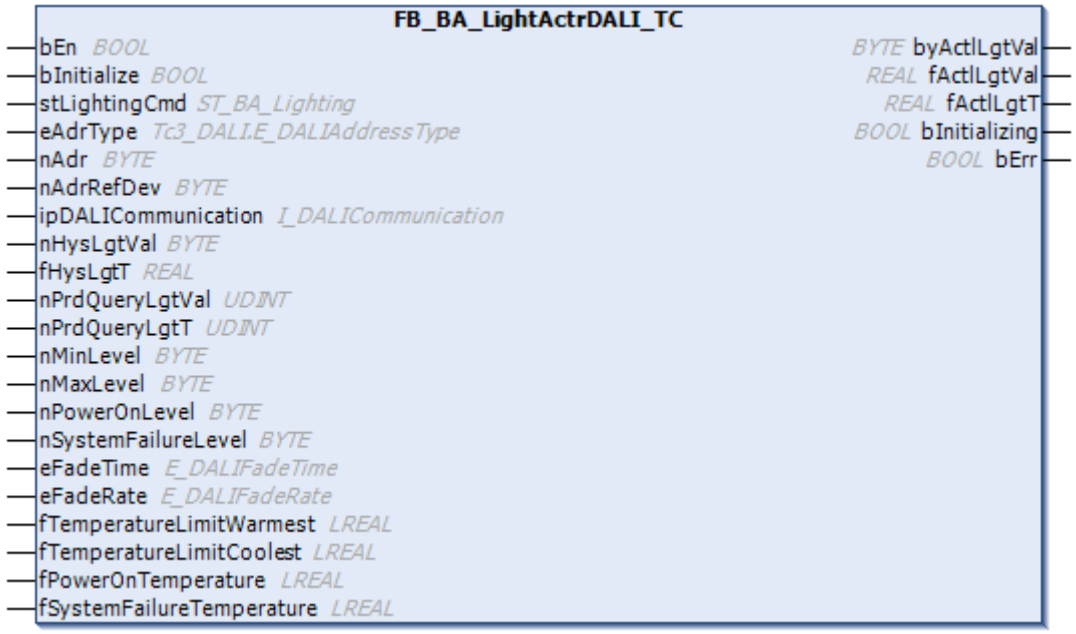
The following parameters cannot be changed and are transferred to the DALI device by the initialization routine.

Name	Type	Description
eExtendedFadeTimeBase	Tc3_DALI.E_DALIExtendedFadeTimeBase	The Extended Fade Time is intended for very long dimming ramps of up to 16 minutes. It is disabled for this use case. *eExtendedFadeTimeBase* designates the time base and is pre-set to the smallest value "Base01".
eExtendedFadeTimeMultiplier	Tc3_DALI.E_DALIExtendedFadeTimeMultiplier	*eExtendedFadeTimeMultiplier* is the multiplier for the time base and is set to the value "Disabled".

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_LightActrDALI_TC



This function block is used to control a DALI light actuator, a DALI group or DALI devices with a broadcast, whereby color temperature control is supported.

Function

Four routines are executed in the function block:

- Setting the light value if it deviates from the current value. This is subjected to a hysteresis range *nHysLgtVal*, see below.
- Setting the color temperature if it deviates from the current value. This is subjected to a hysteresis range *fHysLgtT*, see below.
- Reading the light value. This is done cyclically at the interval *nPrdQueryLgtVal* in seconds.
- Reading the color temperature. This is done cyclically at the interval *nPrdQueryLgtT* in seconds.

The "current value" is always initially assumed to be the value that was sent without errors. The cyclic query of the respective read function will correct the value if a telegram is nevertheless "lost" or the control gear has been controlled in another way.

This procedure ensures that, on the one hand, the DALI bus is not overloaded, but on the other hand, the actual light value is always read.

The function block is controlled via values from the light telegram stLightingCmd [▶ 272]. The control values of the light and the color temperature are limited to the limits that were previously transferred to the control gear or the devices by an initialization routine with other configuration parameters, see VAR_INPUT CONSTANT PERSISTENT. Initialization takes place via the input *bInitialize*. Fixed values are also transferred here, see VAR CONSTANT.

A color temperature setpoint of "0" is explicitly ignored. This allows telegrams to be generated that only influence the brightness, but not the color temperature previously set elsewhere.

The light value is set internally exclusively via the DALI command FB_DALI102DirectArcPowerControl. This means that the FadeTime is the time taken to dim to the new light value.

This dimming time is pre-set to 0 s so that dimming ramps can only be implemented in the PLC.

If group or broadcast control is selected, the control and initialization commands are directed to all devices. Read commands, on the other hand, are only sent to the reference device.

Syntax

```
FUNCTION_BLOCK FB_BA_LightDALI_TC
VAR_INPUT
  bEn                          : BOOL;
  bInitialize                  : BOOL;
  stLightingCmd                : ST_BA_Lighting;
  eAdrType                     : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                         : BYTE;
  nAdrRefDev                   : BYTE;
  ipDALICommunication          : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_OUTPUT
  byActlLgtVal                 : BYTE;
  fActlLgtVal                  : REAL;
  fActlLgtT                    : REAL;
  bInitializing                : BOOL;
  bErr                         : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nHysLgtVal                   : BYTE;
  fHysLgtT                     : REAL;
  nPrdQueryLgtVal              : UDINT;
  nPrdQueryLgtT                : UDINT;

  nMinLevel                    : BYTE := 126;
  nMaxLevel                    : BYTE := 254;
  nPowerOnLevel                : BYTE := 254;
  nSystemFailureLevel          : BYTE := 254;
  eFadeTime                    : Tc3_DALI.E_DALIFadeTime := Tc3_DALI.E_DALIFadeTime.Disabled;
  eFadeRate                    : Tc3_DALI.E_DALIFadeRate := Tc3_DALI.E_DALIFadeRate.N358StepsPerSec;
  fTemperatureLimitWarmest     : LREAL := 2700;
  fTemperatureLimitCoolest     : LREAL := 6500;
  fPowerOnTemperature          : LREAL := 3000;
  fSystemFailureTemperature    : LREAL := 3000;
END_VAR
VAR CONSTANT
  eExtendedFadeTimeBase        : Tc3_DALI.E_DALIExtendedFadeTimeBase := Tc3_DALI.E_DALIExtendedFadeT
imeBase.Base01;
  eExtendedFadeTimeMultiplier  : Tc3_DALI.E_DALIExtendedFadeTimeMultiplier := Tc3_DALI.E_DALIExtende
dFadeTimeMultiplier.Disabled;
END_VAR
```

VAR_INPUT

Name	Type	Description
bEn	BOOL	Enabling the function block: a TRUE signal at this input enables the function.
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the DALI device, see FB_BA_Swi4DALI [▶ 880].
stLightingCmd	ST_BA_Lighting [▶ 272]	Light control telegram
eAdrType	Tc3_DALI.E_DALIAddressType	Defines whether the input *nAdr* contains a short address (0...63) or a group address (0...31). The input *nAdr* has no meaning if a broadcast is sent
nAdr	BYTE	DALI short address of the sensor.
nAdrRefDev	BYTE	DALI address of a reference device if group or broadcast control is selected. This reference device then represents all other controlled devices.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

VAR_OUTPUT

Name	Type	Description
byActlLgtVal	BYTE	Current light value in DALI format (0...254).
fActlLgtVal	REAL	Current light value in percent.
fActlLgtT	REAL	Current color temperature in Kelvin.
bInitializing	BOOL	The sensor is in the DALI initialization phase, i.e. the entered parameters are transferred to the DALI device.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.

VAR_INPUT CONSTANT PERSISTENT

Name	Type	Description
nHysLgtVal	BYTE	Hysteresis range around the light control value. A new set command is started if the setpoint *nHysLgtVal/2* is above or below the actual value.
fHysLgtT	REAL	Hysteresis range around the color temperature default value. A new set command is started if the setpoint *nHysLgtVal/2* is above or below the actual value.
nPrdQueryLgtVal	UDINT	Query interval of the light value [s].
nPrdQueryLgtT	UDINT	Query interval of the color temperature [s].

The following parameters are transferred to the DALI control gear or the DALI devices by the initialization routine.

Name	Type	Description
nMinLevel	BYTE	Minimum light control value in DALI output format (0...254). Pre-set to 126.
nMaxLevel	BYTE	Maximum light control value in DALI output format (0...254). Pre-set to 254.
nPowerOnLevel	BYTE	Light control value when the supply voltage is applied to the control gear. Pre-set to 254.
nSystemFailureLevel	BYTE	Light control value in the event of an error on the DALI bus. Pre-set to 254.
eFadeTime	Tc3_DALI.E_DALIFadeTime	The FadeTime defines the time the current output value takes to reach the required value. The FadeTime is pre-set to "disabled", which together with the deactivated "Extended FadeTime" (VAR CONSTANT) corresponds to a fade time of 0 s.
eFadeRate	Tc3_DALI.E_DALIFadeRate	The Fade Rate determines the rate of change (in steps per second) of the output value. It is not decisive for the behavior of this function block. It is pre-set to 358 steps/s.
fTemperatureLimitWarmest	LREAL	Lower limit of the color temperature range in Kelvin. Pre-set to 2700 K.
fTemperatureLimitCoolest	LREAL	Upper limit of the color temperature range in Kelvin. Pre-set to 6500 K.
fPowerOnTemperature	LREAL	Color temperature value when the supply voltage is applied to the control gear. Pre-set to 3000 K.
fSystemFailureTemperature	LREAL	Color temperature value in the event of an error on the DALI bus. Pre-set to 3000 K.

VAR CONSTANT

The following parameters cannot be changed and are transferred to the DALI device by the initialization routine.

Name	Type	Description
eExtendedFadeTimeBase	Tc3_DALI.E_DALIExtendedFadeTimeBase	The Extended Fade Time is intended for very long dimming ramps of up to 16 minutes. It is disabled for this use case. *eExtendedFadeTimeBase* designates the time base and is pre-set to the smallest value "Base01".
eExtendedFadeTimeMultiplier	Tc3_DALI.E_DALIExtendedFadeTimeMultiplier	*eExtendedFadeTimeMultiplier* is the multiplier for the time base and is set to the value "Disabled".

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_Swi4DALI

This function block is used to read a DALI 4-fold push button.

For information on the function of the DALI device used, please refer to the manufacturer's device documentation.

Function

The device itself is specified by the instance numbers of the buttons, *nInstButton1...nInstButton4*.

The device's internal notification system is always used, which automatically sends a telegram as soon as a button press or release is detected.

The states of the buttons are also queried cyclically. This ensures that the state of the buttons is always corrected if a telegram is "lost".

The DALI push button function block can be initialized with **fixed** parameters via the input *bInitialize*, see VAR CONSTANT.

Syntax

```
FUNCTION_BLOCK FB_BA_Swi4DALI
VAR_INPUT
  bEn                   : BOOL;
  bInitialize           : BOOL;
  nAdr                  : BYTE;
  nInstButton1          : BYTE;
  nInstButton2          : BYTE;
  nInstButton3          : BYTE;
  nInstButton4          : BYTE;
  ipDALICommunication   : Tc3_DALI.I_DALICommunication;
END_VAR
```

```
VAR_OUTPUT
  bButton1               : BOOL;
  bButton2               : BOOL;
  bButton3               : BOOL;
  bButton4               : BOOL;
  bInitializing          : BOOL;
  bErr                   : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nPrdQuery              : UDINT := 60;
END_VAR
VAR CONSTANT
  nEventFilter           : DWORD := 131;
  eEventPriority         : Tc3_DALI.E_DALIEventPriority := Tc3_DALI.E_DALIEventPriority.Middle;
  eEventScheme           : Tc3_DALI.E_DALIEventScheme := Tc3_DALI.E_DALIEventScheme.DeviceInstance;
  nRepeatTimer           : UINT := 160;
  nShortTimer            : UINT := 500;
  nDoubleTimer           : UINT := 0;
  nStuckTimer            : BYTE := 20;
END_VAR
```

VAR_INPUT

Name	Type	Description
bEn	BOOL	Enabling the function block: a TRUE signal at this input enables the function.
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the DALI device, see FB_BA_Swi4DALI [▶ 880].
nAdr	BYTE	DALI short address of the push button sensor.
nInstButton1... nInstButton4	BYTE	Numbers of the push button instances to be queried.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

VAR_OUTPUT

Name	Type	Description
bButton1...bButton4	BOOL	State of the individual push buttons (TRUE = pressed).
bInitializing	BOOL	The sensor is in the DALI initialization phase, i.e. the entered parameters are transferred to the DALI device.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.

VAR_INPUT CONSTANT PERSISTENT

Name	Type	Description
nPrdQuery	UDINT	Query interval of the switch states. This query is only used for correction if an incorrect state is detected due to a telegram error.

VAR CONSTANT

The following parameters cannot be changed and are transferred to the DALI device by the initialization routine.

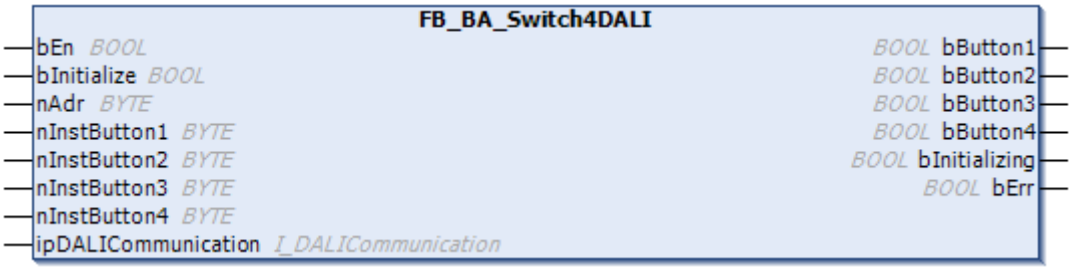
Name	Type	Description
nEventFilter	DWORD	DALI event filter mask for occupancy sensors. This value is set to 7 and thus queries the *Occupied*, *Vacant* and *Repeat Event* states.
eEventPriority	Tc3_DALI.E_DALIEventPriority	DALI event priority with which input notification events are sent by the instance of the DALI control device, set to *Middle*.
eEventScheme	Tc3_DALI.E_DALIEventScheme	DALI event scheme. This is set to *DeviceInstance* (device/instance addressing with short address and instance number).
nRepeatTimer	UINT	Repeat Timer as a multiple of 20 ms. Set to 8 (=160 ms).
nShortTimer	UINT	Short Timer as a multiple of 20 ms. Set to 25 (=500 ms).
nDoubleTimer	UINT	Double Timer as a multiple of 20 ms. Set to 0 (=0 ms). This means that this value is set to the vendor-specific minimum value.
nStuckTimer	BYTE	Stuck Timer in seconds. Set to 20 s.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.1.2 Sensors

6.1.4.2.1.4.1.2.1 FB_BA_Switch4DALI



The template is used to read a 4-fold switch.

Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_BaseSwitch* (internal function block). The use of the following variable is visible in this template:

- **bInitializeRm:** Start DALI initialization routine from the HMI.

Function

The device itself is specified by the instance numbers of the push buttons, *nInstButton1* ... *nInstButton4*.

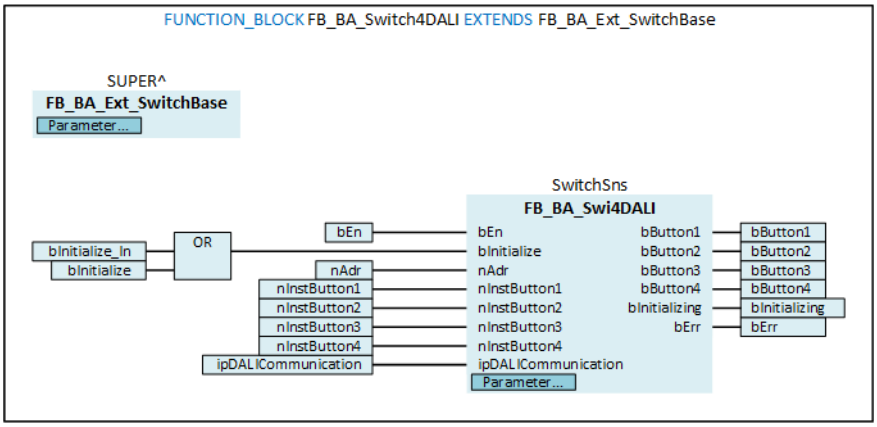
The device's internal notification system is always used, which automatically sends a telegram as soon as a button press or release is detected.

The states of the buttons are also queried cyclically in the internal function block FB_BA_Swi4DALI [▶ 880]. This ensures that the state of the buttons is always corrected if a telegram is "lost".

The DALI push button function block can be initialized with its set parameters via the input *bInitialize* or via the HMI (*bInitializeRm*), see FB_BA_Swi4DALI [▶ 880].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Switch4DALI EXTENDS FB_BA_Ext_SwitchBase
VAR_INPUT
  bEn                   : BOOL;
  bInitialize           : BOOL;
END_VAR
VAR_OUTPUT
  bButton1              : BOOL;
  bButton2              : BOOL;
  bButton3              : BOOL;
  bButton4              : BOOL;
  bInitializing         : BOOL;
  bErr                  : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nAdr                  : BYTE;
  nInstButton1          : BYTE := 0;
  nInstButton2          : BYTE := 1;
  nInstButton3          : BYTE := 2;
  nInstButton4          : BYTE := 3;
  ipDALICommunication   : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_INPUT CONSTANT
  SwitchSns             : FB_BA_Swi4DALI;
END_VAR
```

VAR_INPUT

Name	Type	Description
bEn	BOOL	Enabling the function block: a TRUE signal at this input enables the function.
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the DALI device, see FB_BA_Swi4DALI [▶ 880].

VAR_OUTPUT

Name	Type	Description
bButton1...bButton4	BOOL	State of the individual push buttons (TRUE = pressed).
bInitializing	BOOL	The push button sensor is in the DALI initialization phase, i.e. the entered parameters are transferred to the DALI device.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.

VAR_INPUT CONSTANT PERSISTENT

Name	Type	Description
nAdr	BYTE	DALI individual address of the push button sensor.
nInstButton1...nInstButton4	BYTE	Numbers of the push button instances to be queried.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

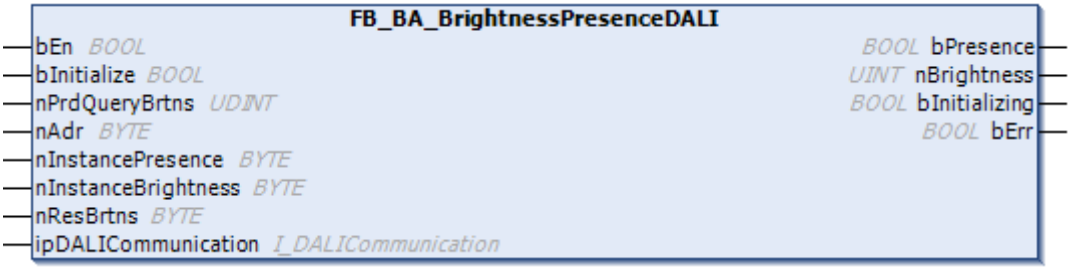
VAR_INPUT CONSTANT

Name	Type	Description
SwitchSns	FB_BA_Swi4DALI [▶ 880]	Function block for reading the push button sensor.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.1.2.2 FB_BA_BrightnessPresenceDALI



This template is used to read a combined DALI brightness and occupancy sensor.

Data exchange HMI

The data exchange with the HMI is realized in the base class *FB_BA_BaseBrightnessPresence* (internal function block). The use of the following variable is visible in this template:

- **bInitializeRm:** Start DALI initialization routine from the HMI.

Function

The device itself is specified by the instance number of the presence, *nInstancePresence*, and the instance number of the brightness, *nInstanceBrightness*. Added to this is the device-specific resolution, *nResBrtns*. These three parameters vary from manufacturer to manufacturer.

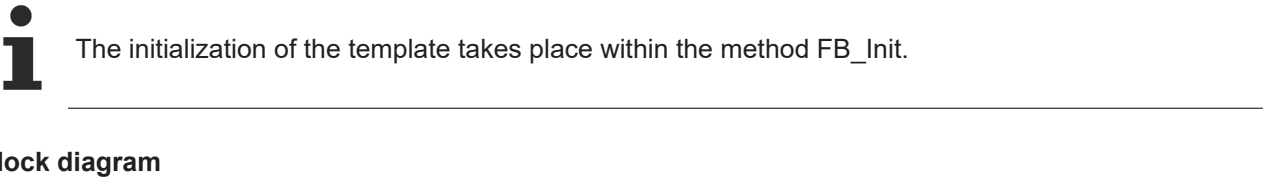
The brightness is queried cyclically (interval *nPrdQueryBrtns* [s]) in the internal function block BrightnessPresenceSns [▶ 871]. With presence, on the other hand, the device's internal notification system is used, which automatically sends a telegram as soon as presence or absence is detected.

The DALI-specific parameters of this function block (*nInstPrc*, *nInstBrtns* and *nResBrtns*) are transferred to the relevant device via a TRUE signal to *bInitialize* or via the HMI via *bInitialize_In*. The device is addressed by naming the addressx (*nAdr*) and linking the communication block of the corresponding DALI line via *ipDALICommunication*.

The function block PresenceDelay [▶ 1025] provides a delayed signal *bPresence*. After the delay time has elapsed, a positive edge is also emitted at *bRstSwi*, which can be used to cancel the flag circuits for light and blinds.

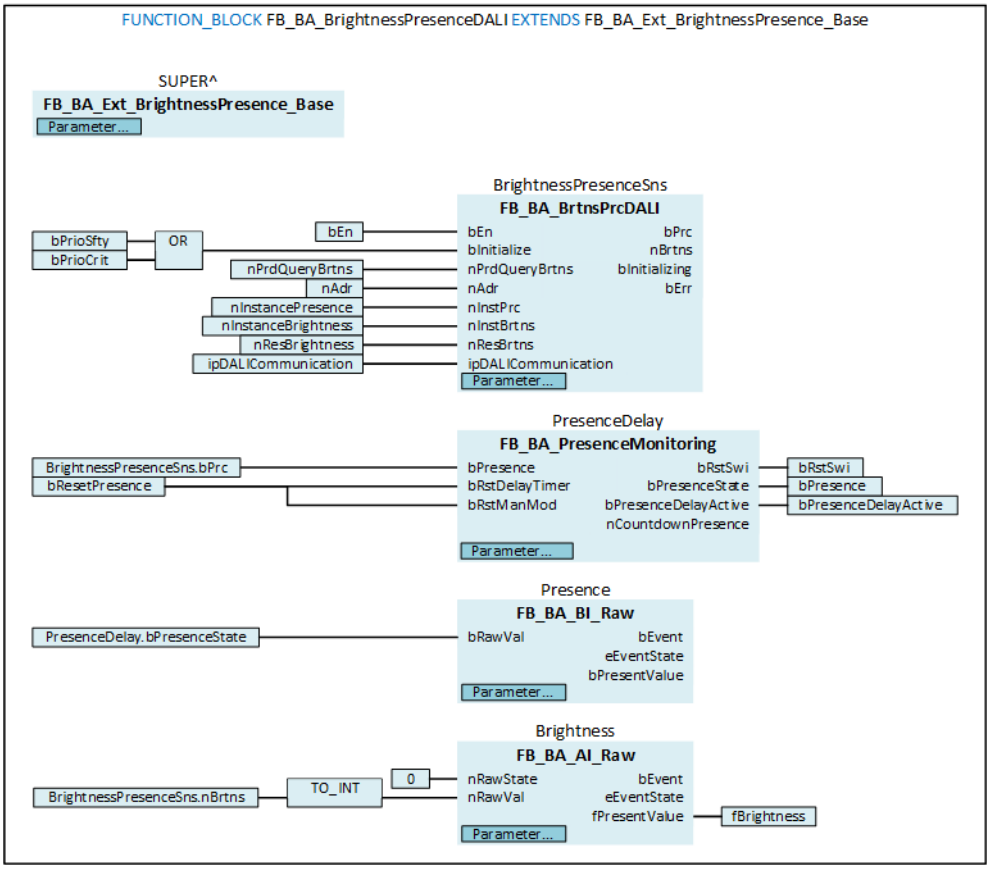
Presence and brightness values are linked to the objects *Presence* and *Brightness* and are thus available in BACnet.

Any scaling of the brightness can be carried out by parameterization of *Brightness* in FB_Init using the parameters *fResolution* and *fScaleOffset*.



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_BrightnessPresenceDALI EXTENDS FB_BA_Ext_BrightnessPresence_Base
VAR_INPUT
  bEn                       : BOOL;
  bInitialize               : BOOL;
  bResetPresence            : BOOL;
  nPrdQueryBrtns            : UDINT  := 1;
END_VAR
VAR_OUTPUT
  bPresence                 : BOOL;
  bPresenceDelayActive      : BOOL;
```

```
  bRstSwi                 : BOOL;
  fBrightness             : REAL;
  bInitializing           : BOOL;
  bErr                    : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nAdr                    : BYTE;
  nInstancePresence       : BYTE;
  nInstanceBrightness     : BYTE;
  nResBrtns               : BYTE;
  ipDALICommunication     : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_INPUT CONSTANT
  BrightnessPresenceSns   : FB_BA_BrtnsPrcDALI;
  PresenceDelay           : FB_BA_PresenceMonitoring;
  Brightness              : FB_BA_AI_RAW;
  Presence                : FB_BA_BI_RAW;
END_VAR
```

VAR_INPUT

Name	Type	Description
bEn	BOOL	Enabling the function block: a TRUE signal at this input enables the function.
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the DALI device, see FB_BA_Swi4DALI [▶ 880].
bResetPresence	BOOL	Resets the internal presence delay and thus triggers the *bRstSwi* output once.
nPrdQueryBrtns	UDINT	Brightness query interval in seconds. The declaration of this variable as an input is intended to enable a variable query speed. In the case of constant light regulation, for example, a slow query interval is initially sufficient until a larger deviation to be compensated is detected, at which point a shorter interval is used.

VAR_OUTPUT

Name	Type	Description
bPresence	BOOL	Occupancy signal output.
bPresenceDelayActive	BOOL	This output is then set to TRUE if no more presence is detected and the output *bPresence* is only set by the internal delay.
bRstSwi	BOOL	Reset output for manual overrides. Is set for a PLC cycle after the internal presence delay has elapsed (see PresenceDelay [▶ 1025]) or when a TRUE signal is received at the input bResetPresence,
fBrightness	REAL	Measured brightness. The unit is vendor-specific. Conversion to lux must be requested from the manufacturer if required.
bInitializing	BOOL	The sensor is in the DALI initialization phase, i.e. the entered parameters are transferred to the DALI device.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.

VAR_INPUT CONSTANT PERSISTENT

Name	Type	Description
nAdr	BYTE	DALI single address of the sensor.
nInstancePresence	BYTE	Number of the presence instance to be queried.
nInstanceBrightness	BYTE	Number of the brightness instance to be queried.
nResBrtns	BYTE	Vendor-specific brightness resolution.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

VAR_INPUT CONSTANT

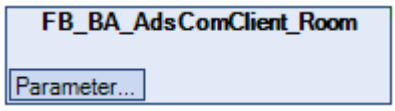
Name	Type	Description
BrightnessPresenceSns	FB_BA_BrtnsPrcDALI [▶ 871]	Function block for reading the DALI brightness and occupancy sensor.
PresenceDelay	FB_BA_PresenceMonitoring [▶ 1025]	Universal presence monitoring template with reset inputs for delay timer and manual function.
Brightness	FB_BA_AI_RAW [▶ 195]	This function block is used to convert the vendor-specific raw brightness value to a desired unit, for example lux, and also to display the value in BACnet.
Presence	FB_BA_BI_RAW [▶ 206]	Function block for displaying the presence state in BACnet.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.2 Communication

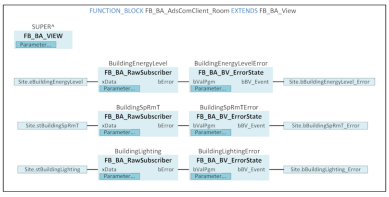
6.1.4.2.1.4.2.1 FB_BA_AdsComClient_Room



The template reads global data that is assigned to the "Room automation" trade from another controller, which makes this data available with the FB_BA_AdsComServer_Room [▶ 888] counter-block. The information read is copied to the global variable list Site [▶ 1116].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  BuildingEnergyLevel          : FB_BA_RawSubscriber;
  BuildingEnergyLevelError     : FB_BA_BV_ErrorState;
  BuildingSpRmT                : FB_BA_RawSubscriber;
  BuildingSpRmTError           : FB_BA_BV_ErrorState;
  BuildingLighting             : FB_BA_RawSubscriber;
  BuildingLightingError        : FB_BA_BV_ErrorState;
END_VAR
```

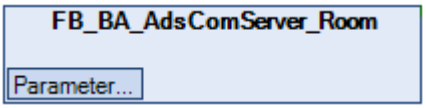
VAR_INPUT CONSTANT

Name	Type	Description
BuildingEnergyLevel	FB_BA_RawSubscriber [▶ 158]	The subscriber accesses the TwinCAT network structure *eBuildingEnergyLevel* and saves the data in the created enumeration *eBuildingEnergyLevel* of the GVL Site [▶ 1116].
BuildingEnergyLevelError	FB_BA_BV_ErrorState [▶ 1030]	Binary object indicating communication failure of the subscriber *BuildingEnergyLevel*.
BuildingSpRmT	FB_BA_RawSubscriber [▶ 158]	The subscriber accesses the TwinCAT network structure *BuildingSpRmT* and saves the data in the created structure *stBuildingSpRmT* of the GVL Site [▶ 1116].
BuildingSpRmTError	FB_BA_BV_ErrorState [▶ 1030]	Binary object indicating communication failure of the subscriber *BuildingSpRmT*.
BuildingLighting	FB_BA_RawSubscriber [▶ 158]	The subscriber accesses the TwinCAT network structure *BuildingLighting* and saves the data in the created structure *stBuildingLighting* of the GVL Site [▶ 1116].
BuildingLightingError	FB_BA_BV_ErrorState [▶ 1030]	Binary object indicating communication failure of the subscriber *BuildingLighting*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.2.2 FB_BA_AdsComServer_Room

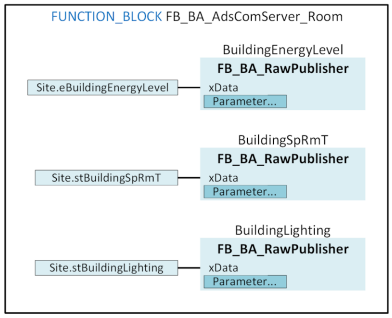


The template has the task of reading data and parameters that are assigned to the "Room automation" trade from the global variable list Site and making them available within the BA network via FB_BA_RawPublisher [▶ 139].

The counter-block that receives this data as a client is FB_BA_AdsComClient_Room [▶ 887].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComServer
VAR_INPUT CONSTANT
  BuildingEnergyLevel     : FB_BA_RawPublisher;
  BuildingSpRmT           : FB_BA_RawPublisher;
  BuildingLighting        : FB_BA_RawPublisher;
END_VAR
```

VAR_INPUT CONSTANT

Name	Type	Description
BuildingEnergyLevel	FB_BA_RawPublisher [▶ 139]	The publisher publishes the current building energy level Site.eBuildingEnergyLevel [▶ 1116], which is created in the template FB_BA_BuildingEnergyLevel [▶ 889].
BuildingSpRmT	FB_BA_RawPublisher [▶ 139]	The publisher publishes the structure of the building-wide room temperature data Site.stBuildingSpRmT [▶ 1116], which is created in the template FB_BA_BuildingSpRmT [▶ 860].
BuildingLighting	FB_BA_RawPublisher [▶ 139]	The publisher publishes the structure of the building-wide lighting data Site.stBuildingLighting [▶ 1116], which is created in the template FB_BA_BuildingLighting [▶ 917].

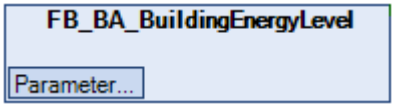
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.3 HeatingCooling

Templates for heating and cooling.

6.1.4.2.1.4.3.1 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.

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 [▶ 239]	Input object Hand for the "BuildingEnergyLevel".
EnergLvlPr	FB_BA_MV_Op [▶ 239]	Resulting mode.
Sched	FB_BA_SchedM [▶ 225]	Schedule object (automatic) for the "BuildingEnergyLevel".

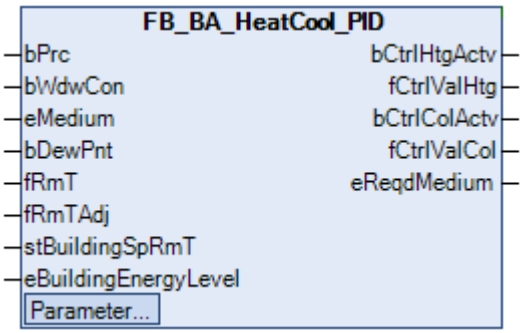
Variables

Name	Type	Description
DeMuxEnergLvl	FB_BA_DMUX_B08 [▶ 426]	Conversion of the multistate value of the manual mode to a binary output.
DeMuxSched	FB_BA_DMUX_B04 [▶ 426]	Conversion of the multistate value of the schedule to a binary output.
PrioSwi	FB_BA_PrioSwi_UDI08 [▶ 433]	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

6.1.4.2.1.4.3.2 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* is used for a local shift of the room temperature setpoint for the energy levels "Pre-Comfort" and "Comfort".

The setpoint is shifted via a local input value *fRmTAdj* or via an external value *fRmTAdj_In* which is read via the base class *FB_BA_Ext_SunblindAngle* (internal function block), for example via a HMI. The value that was last changed is always valid; the function block FB_BA_LastWriterWins_R04 [▶ 447] is responsible for this. The room temperature value is limited to +/- *fRmTAdjLimit* (predefined to 1.0 in FB_init).

The instance *FnctSel* of the function block FB_BA_FnctSel [▶ 299] enables the heating or cooling controller of the temperature zone.

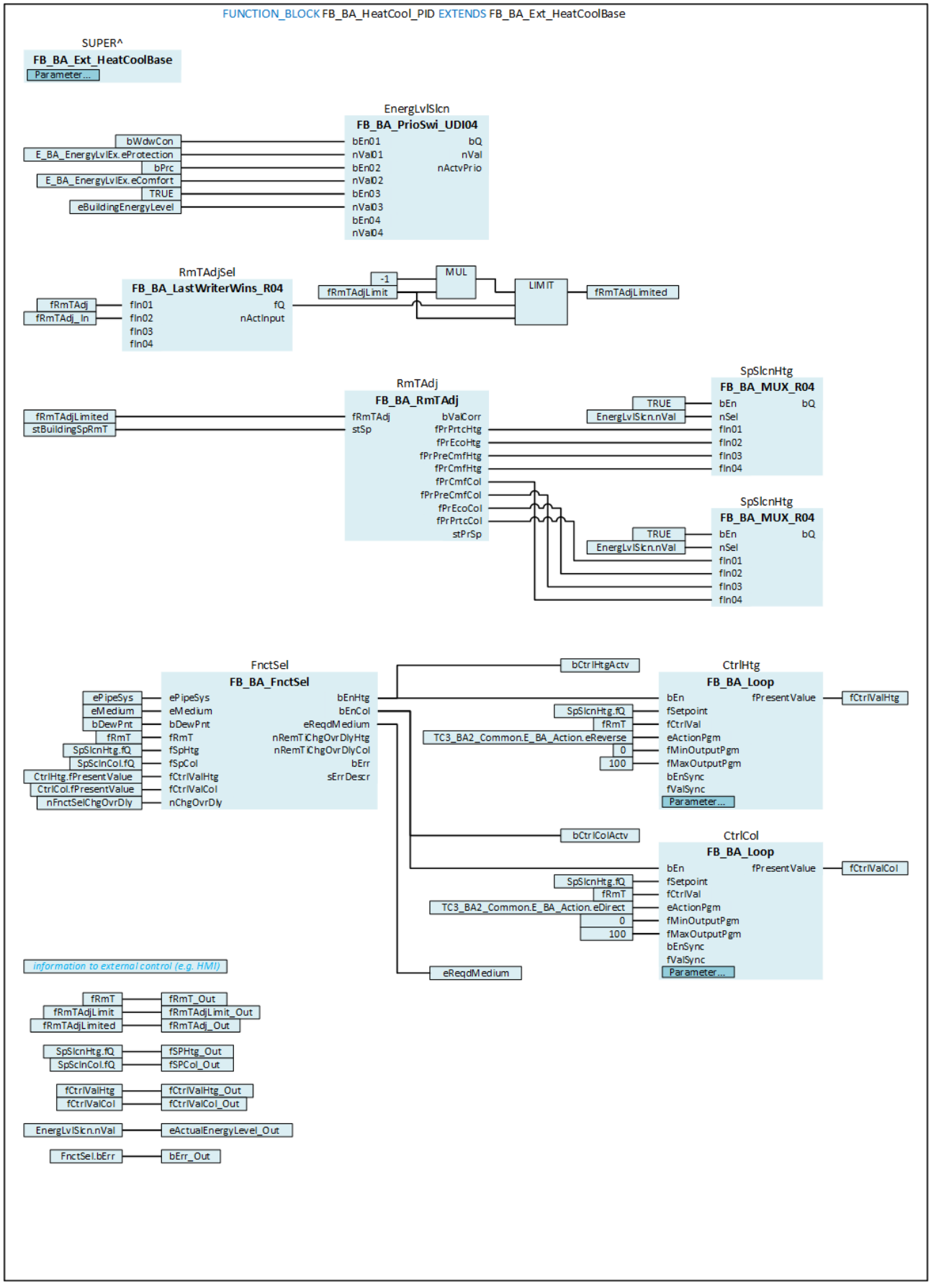
Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_Ext_HeatCoolBase* (internal function block). The use of the following variables is visible in this template:

- **fRmTAdj_In**: Input room setpoint shift from the HMI.
- **fRmTAdj_Out**: Output of current room setpoint shift to the HMI.
- **fRmTAdjLimit_Out**: Output of the +/- range limits of the room setpoint shift to the HMI.
- **fRmT_Out**: Output of current room temperature to the HMI.
- **fSPHtg_Out**: Output of heating setpoint to the human-machine interface.
- **fSPCol_Out**: Output of setpoint cooling to the HMI.
- **fCtrlValHtg_Out**: Output heating controller output to the HMI.
- **fCtrlValCol_Out**: Output cooling controller output to the HMI.
- **eActualEnergyLevel_Out**: Output of the current energy level to the HMI.
- **bErr_Out**: Output status "Error to *FnctSel*" to the HMI.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_HeatCool_PID EXTENDS FB_BA_Ext_HeatCoolBase
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;
  fRmTAdjLimit            : REAL;
END_VAR
VAR_INPUT CONSTANT
  CtrlHtg                 : FB_BA_Loop;
  CtrlCol                 : FB_BA_Loop;
END_VAR
VAR
  RmTAdj                  : FB_BA_RmTAdj;
  EnergLvlSlcn            : FB_BA_PrioSwi_UDI04;
  SpSlcnHtg               : FB_BA_MUX_R04;
  SpSclnCol               : 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 [▶ 267]	Medium present (heating or cooling medium, only important for two-pipe operation)
bDewPnt	BOOL	Dew point sensor (alarm = TRUE)
fRmT	BOOL	Room temperature [°C]
fRmTAdj	REAL	Room setpoint shift [K]
stBuildingSpRmT	ST_BA_SpRmT [▶ 272]	Structure of room setpoints (Protection Cooling..Comfort Cooling and Protection Heating ... Comfort Heating)
eBuildingEnergyLevel	E_BA_EnergyLvlEx [▶ 700]	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 [▶ 267]	Requested medium (heating or cooling medium)

Inputs CONSTANT PERSISTENT

Name	Type	Description
ePipeSys	E_BA_PipeSys [▶ 267]	Selection of two or four-pipe system.
nFnctSelChgOvrDly	UDINT	Switchover delay [s] from heating to cooling or vice versa.
fRmTAdjLimit	REAL	Adjustable range [K] of the setpoint value shift *fRmTAdj* : *-fRmTAdjLimit <= fRmTAdj <= +fRmTAdjLimit*

Inputs CONSTANT

Name	Type	Description
CtrlHtg	FB_BA_Loop [▶ 220]	Heating controller
CtrlCol	FB_BA_Loop [▶ 220]	Cooling controller

Variables

Name	Type	Description
RmTAdj	FB_BA_RmTAdj [▶ 302]	Function block that applies the sh to the corresponding setpoints and outputs them explicitly.
EnergLvlSlcn	FB_BA_PrioSwi_UDI04 [▶ 433]	Prioritizing selection and conversion of the possible energy levels into a numerical value.
SpSlcnHtg	FB_BA_MUX_R04 [▶ 431]	Selection of the heating setpoint
SpSclnCol	FB_BA_MUX_R04 [▶ 431]	Selection of the cooling setpoint
FnctSel	FB_BA_FnctSel [▶ 299]	Function selection heating or cooling.

Requirements

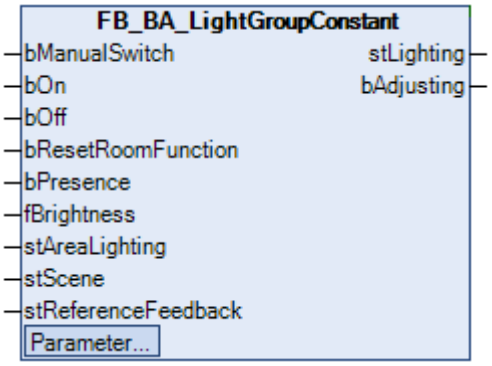
Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.4 Lighting

6.1.4.2.1.4.4.1 Functions

6.1.4.2.1.4.4.1.1 Zone

6.1.4.2.1.4.4.1.1.1 FB_BA_LightGroupConstant



One or more light actuators are combined in a light zone for simultaneous control, depending on local conditions.

This template represents a light zone (room) with constant light regulation and manual override.

Function

If fully automatic is set via the parameter *eLightActivationMode*, the light function ConstLgtCtrl [▶ 317] switches to control mode when presence is detected at the input *bPresence*. However, if semi-automatic is selected, the first press of a button on the template *bManualSwitch* will set the function block to control mode. Further button presses or a value specification via the HMI (*fSetLightValueMan_In* / *SetLightValueMan_In*) abort the control mode and the function block is in manual override.

In this mode, the light can be switched on and off alternately via the template input *bManualSwitch* or dimmed up and down with a long keystroke (*t>SwiOvrTi*). The inputs *bOn* and *bOff* allow the light to be switched on and off selectively, while the light value *fSetLightValueMan_In* is set directly via a HMI using the *bSetLightValueMan_In* command.

The light control block [▶ 317] displays its current state "Control mode" or "Manual override" at the outputs *bControlMode* and *bManualMode*; both outputs are not set simultaneously. These outputs are used either to generate a light output telegram with the priority eAutomaticLight [▶ 268] via the CnstLgtAuto [▶ 309] function block or a telegram with the priority eManual [▶ 268] via ManMode [▶ 309]. The different telegram priority is assigned to the internal HMI variable *eActualPrio_Out* on the PrioSwi [▶ 310] function block. For example, a HMI can be used to display a distinction between control mode and manual override.

On a falling edge at the template input *bPresence* or by setting *bResetRoomFunction*, the light control block is disabled, whereby, if selected, it still announces its shutdown via a switch-off ramp with a dwell time at a pre-switch-off value. Both function block outputs *bControlMode* and *bManualMode* are then deleted.

In addition to the internal telegrams from *bControlMode* and *ManMode*, the input-side *stAreaLighting* and *stScene* are also routed to the selector (*PrioSwi*).

Here, *stArea* represents the resulting telegram from the higher-level, higher-order levels (area, floor and building), while stScene is reserved for any lighting scene to override the constant light regulation.

The telegram selector (*PrioSwi*) forwards the telegram with the highest priority - with the same priority, the last telegram sent (last writer wins).

The setpoint for constant light regulation is set using an analog object SpBrightness [▶ 202].

Input stReferenceFeedback

Information about the controlled light actuator or the reference actuator of a group is fed back into the light control function via this input.

The light value of the reference actuator is of particular importance here: If the light actuator is already controlled by another function of the same or lower priority and the template described here now "takes over" control, it must control the actuator based on its existing light value.

In addition, current values and states are transmitted and can be displayed via the HMI.

Data exchange HMI

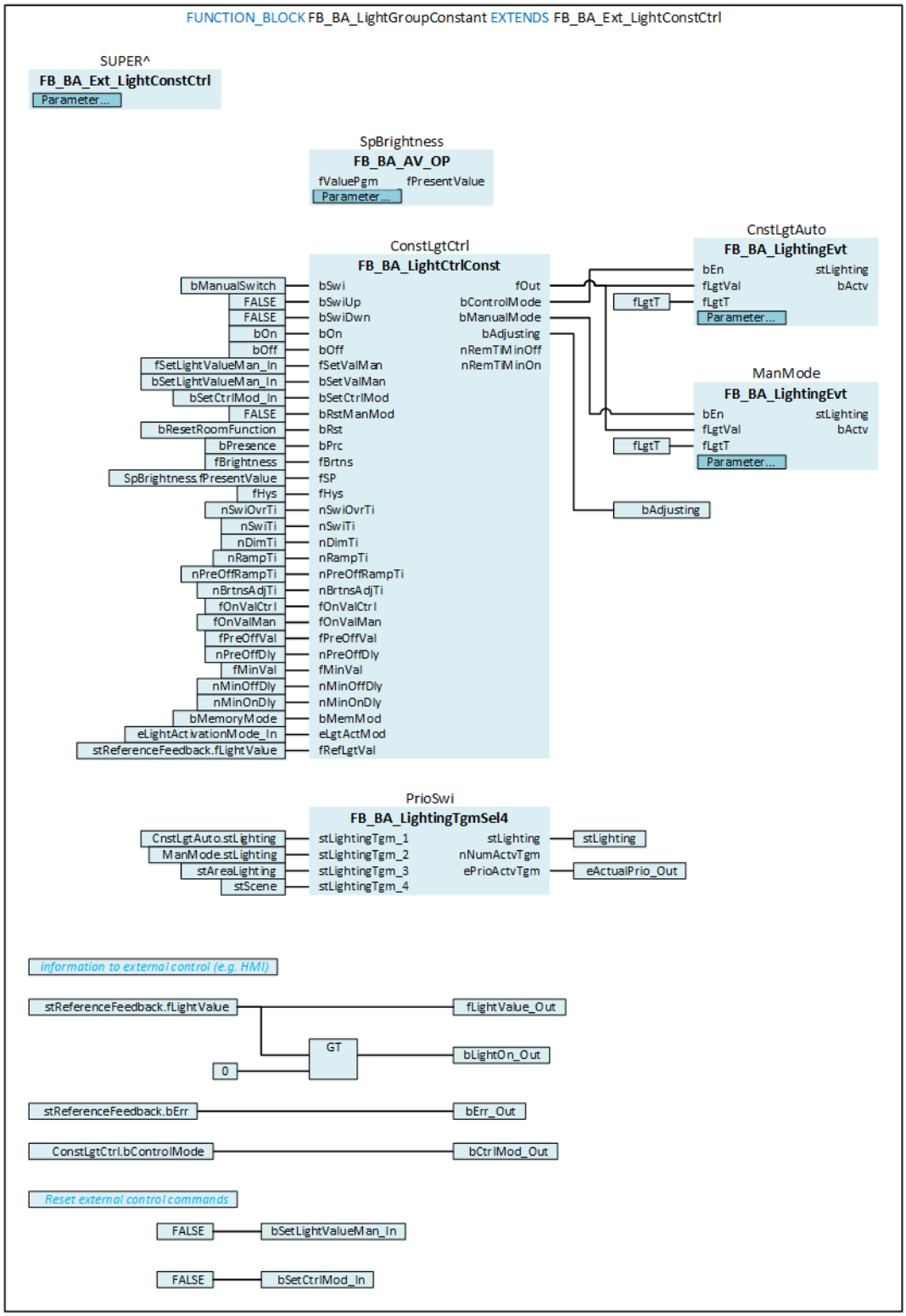
The data exchange with the HMI is realized here in the base class *FB_BA_Ext_LightConstCtrl* (internal function block). The use of the following variables is visible in this template:

- **bSetLightValueMan_In**: Command to switch to manual mode.
- **fSetLightValueMan_In**: Light value to be adopted.
- **bSetCtrlMod_In**: Command to switch to constant light control mode.
- **eLightActivationMode_In**: Activation mode of the light control function [▶ 317] from the HMI (fully automatic or semi-automatic). The mode selected via the HMI is saved persistently.
- **fLightValue_Out**: Output of percentage light value to the HMI.
- **bLightOn_Out**: Output information "Light On" to the HMI.
- **bErr_Out**: Output information "Reference actuator faulty" to the HMI.
- **eActualPrio_Out**: Currently controlling telegram priority to the HMI.

- **bCtrlMod_Out**: Output information "Light function is in control mode" to the HMI.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupConstant EXTENDS FB_BA_Ext_LightConstCtrl
VAR_INPUT
  bManualSwitch          : BOOL;
```

```
  bOn                   : BOOL;
  bOff                  : BOOL;
  bResetRoomFunction    : BOOL;
  bPresence             : BOOL;
  fBrightness           : REAL;

  stAreaLighting        : ST_BA_Lighting;
  stScene               : ST_BA_Lighting;
  stReferenceFeedback   : ST_BA_LightActuatorFeedback;
END_VAR
VAR_OUTPUT
  stLighting            : ST_BA_Lighting;
  bAdjusting            : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  fHys                  : REAL;
  nSwiOvrTi             : UDINT;
  nSwiTi                : UDINT;
  nDimTi                : UDINT;
  nRampTi               : UDINT;
  nPreOffRampTi         : UDINT;
  nBrtnsAdjTi           : UDINT;
  fOnValCtrl            : REAL;
  fOnValMan             : REAL;
  fPreOffVal            : REAL;
  nPreOffDly            : UDINT;
  fMinVal               : REAL;
  nMinOffDly            : UDINT;
  nMinOnDly             : UDINT;
  bMemoryMode           : BOOL;
  fLgtT                 : REAL;
END_VAR
VAR_INPUT CONSTANT
  SpBrightness          : FB_BA_AV_Op;
  ManMode               : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eManual);
  CnstLgtAuto           : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eAutomaticLight);
END_VAR
VAR
  ConstLgtCtrl          : FB_BA_LightCtrlConst;
  PrioSwi               : FB_BA_LightingTgmSel4;
END_VAR
```

Inputs

Name	Type	Description
bManualSwitch	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.
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.
bResetRoomFunction	BOOL	This input switches off the constant light control function block. Switch-off takes place via a ramp and dwell time on a basic light value, see parameters: *fPreOffVal*, *nPreOffDly* and *nPreOffRampT*.
bPresence	BOOL	Presence signal input. If the constant light control block is configured in fully automatic mode, a rising edge can switch the function on and a falling edge can switch it off via this input. In semi-automatic mode, only a falling edge at this input switches off the constant light control block.
fBrightness	REAL	Current brightness for constant light automatic: Range and unit depend on the light sensor.
stAreaLighting	ST_BA_Lighting [▶ 272]	Resulting telegram from the higher levels (area, floor and building), which is formed in the function block FB_BA_AreaLighting [▶ 920].
stScene	ST_BA_Lighting [▶ 272]	Reserved telegram input for scene control.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback input of the controlled light actuator or the guide light of the controlled group.

Outputs

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	Resulting output program.
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.

Inputs CONSTANT PERSISTENT

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 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 depend on the used light sensor, see above: *fBrtnsSP* and *fBrtnsSen*. Pre-set to 50 in *FB_Init*.
nSwiOvrTi	UDINT	Distinction time [ms] between short and long button press. Pre-set to 250 ms in *FB_Init*.
nSwiTi	UDINT	Ramp for the switching functions in seconds, related to a dimming from 0 to 100%. Pre-set to 2 s in *FB_Init*.
nDimTi	UDINT	Ramp for the dimming functions in seconds, related to a dimming from 0 to 100%. Pre-set to 10 s in *FB_Init*.
nRampTi	UDINT	Control ramp of the constant light automatic in seconds, related to a dimming from 0 to 100%. Pre-set to 60 s in *FB_Init*.
nPreOffRampTi	UDINT	Ramp used to drive to a base value *fPreOffVal* before switching off. Pre-set to 2 s in *FB_Init*.
nBrtnsAdjTi	UDINT	Waiting time in seconds 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. Pre-set to 50% in *FB_Init*.
fOnValMan	REAL	Switch-on value of the manual function, if "Memory mode" is not selected via the parameter *eOperationalMode*. Pre-set to 100% in *FB_Init*.
fPreOffVal / nPreOffDly	REAL / 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. Pre-set to 4% and 10 s in *FB_Init*.
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 *nOffDly* (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 *nOnDly* (in seconds) to initially *fMinVal*. Pre-set to 4% in *FB_Init*
nMinOffDly	UDINT	Constant light automatic: Switch-off waiting time in seconds, see *fMinVal*. Pre-set to 300 s in *FB_Init*.
nMinOnDly	UDINT	Constant light automatic: Switch-on waiting time in seconds, see *fMinVal*. Pre-set to 300 s in *FB_Init*.
bMemoryMode	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 *fOnValMan*. Pre-set to FALSE in *FB_Init*.
fLgtT	REAL	Light temperature: Pre-set to 3000 K in *FB_Init*.

Inputs CONSTANT

Name	Type	Description
SpBrightness	FB_BA_AV_Op [▶ 202]	Analog input object: Setpoint of the control [lx].
ManMode	FB_BA_LightingEvt [▶ 309]	Telegram generator for the light values in manual mode (*bManualMode* = TRUE). The priority is pre-set to *eManualActuator*.
CnstLgtAuto	FB_BA_LightingEvt [▶ 309]	Telegram generator for the light values in control mode (*bControlMode* = TRUE). The priority is pre-set to *eConstantLightControl*.

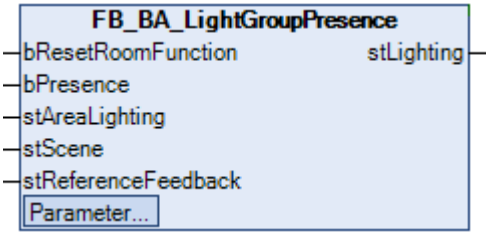
VAR

Name	Type	Description
ConstLgtCtrl	FB_BA_LightCtrlConst [▶ 317]	Constant light control block.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 310]	Telegram selection.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.4.1.1.2 FB_BA_LightGroupPresence



One or more light actuators are combined in a light zone for simultaneous control, depending on local conditions.

This template represents a light zone (room) that is only switched via presence. Typical applications are sanitary areas or interior corridors.

Function

A rising edge at *bPresence* switches the light on via the internal light control block LightSwitch [▶ 311].

The output *bAutomaticMode* changes to TRUE and the light value is output via the function block LightEvent [▶ 309] as the telegram ST_BA_Lighting [▶ 272].

On a falling edge at *bPresence*, the light is first dimmed to a minimum value via a ramp (if parameterized, see FB_BA_LightCtrl [▶ 311]) and then switched off. The output *bAutmaticMode* changes to FALSE and the light telegram is deactivated on the function block LightEvent [▶ 309].

A rising edge at input *bResetRoomFunction* can be used to achieve premature switch-off and deactivation.

Input stReferenceFeedback

Information about the controlled light actuator or the reference actuator of a group is fed back into the light control function via this input.

The light value of the reference actuator is of particular importance here: If the light actuator is already controlled by another function of the same or lower priority and the template described here now "takes over" control, it must control the actuator based on its existing light value.

In addition, current values and states are transmitted and can be displayed via the HMI.

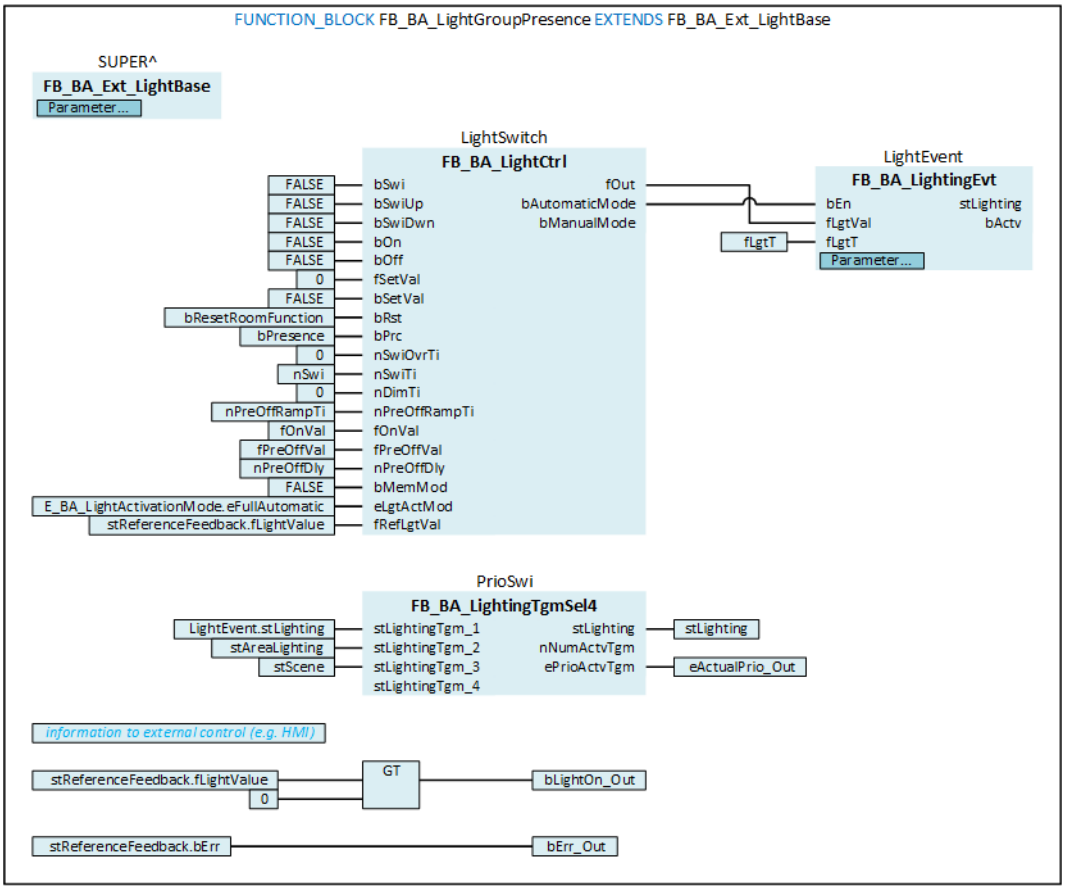
Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_Ext_LightBase* (internal function block). The use of the following variables is visible in this template:

- **bLightOn_Out**: Output information "Light On" to the HMI.
- **bErr_Out**: Output information "Reference actuator faulty" to the HMI.
- **eActualPrio_Out**: Currently controlling telegram priority to the HMI.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupPresence EXTENDS FB_BA_Ext_LightBase
VAR_INPUT
  bResetRoomFunction     : BOOL;
  bPresence              : BOOL;
  stAreaLighting         : ST_BA_Lighting;
  stScene                : ST_BA_Lighting;
  stReferenceFeedback    : ST_BA_LightActuatorFeedback;
END_VAR
VAR_OUTPUT
  stLighting             : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiTi                 : UDINT;
```

```
  nPreOffRampTi          : UDINT;
  fOnVal                 : REAL;
  fPreOffVal             : REAL;
  nPreOffDly             : UDINT;
  fLgtT                  : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  LightEvent             : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eSimple);
END_VAR
VAR
  LightSwitch            : FB_BA_LightCtrl;
  PrioSwi                : FB_BA_LightingTgmSel4;
END_VAR
```

Inputs

Name	Type	Description
bResetRoomFunction	BOOL	This input switches off the light function block *LightSwitch* - it is then no longer active (output *bAutmaticMode*/ *bManualMode* = FALSE). Switch-off takes place via a ramp and dwell time on a basic light value, see parameters: *fPreOffVal*, *nPreOffDly* and *nPreOffRampT*.
bPresence	BOOL	Presence signal input. A rising edge switches the light on and a falling edge switches it off. Switching on and off via presence is accompanied by activation and deactivation of the light function block [▶ 311].
stAreaLighting	ST_BA_Lighting [▶ 272]	Resulting telegram from the higher levels (area, floor and building), which is formed in the function block FB_BA_AreaLighting [▶ 920].
stScene	ST_BA_Lighting [▶ 272]	Reserved telegram input for scene control.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback input of the controlled light actuator or the guide light of the controlled group.

Outputs

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	Resulting output program.

Inputs CONSTANT PERSISTENT

Name	Type	Description
nSwiTi	UDINT	Ramp for the switching functions in seconds, related to a dimming from 0 to 100%. Pre-set to 2 s in *FB_Init*.
nPreOffRampTi	UDINT	Ramp used to drive to a base value *fPreOffVal* before switching off. Pre-set to 2 s in *FB_Init*.
fOnVal	REAL	Switch-on value. Pre-set to 100% in *FB_Init*.
fPreOffVal / nPreOffDly	REAL / 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. Pre-set to 4% and 10 s in *FB_Init*.
fLgtT	REAL	Light temperature: Pre-set to 3000 K in *FB_Init*.

Inputs CONSTANT

Name	Type	Description
LightEvent	FB_BA_LightingEvt [▶ 309]	Telegram generator for the light values. The priority is pre-set to *eManualActuator*.

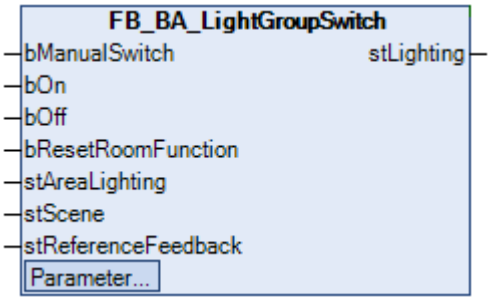
VAR

Name	Type	Description
LightSwitch	FB_BA_LightCtrl [▶ 311]	Light control block.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 310]	Telegram selection.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.4.1.1.3 FB_BA_LightGroupSwitch



One or more light actuators are combined in a light zone for simultaneous control, depending on local conditions.

This template represents a light zone (room) which is only operated via switches.

Function

The light can be switched on and off alternately via the template input *bManualSwitch* using the control block LightSwitch [▶ 311] or dimmed up and down with a long button press (t > *SwiOvrTi*).

The inputs *bOn* and *bOff* allow the light to be switched on and off selectively, while the light value *fSetLightValueMan_In* is set directly via a HMI using the *bSetLightValueMan_In* command.

Switched by manual commands (not presence), the light function block FB_BA_LightCtrl [▶ 311] normally works in such a way that it is always active and indicates this at the output *bManualMode*. This is due to the fact that the user may want to switch off the light in a targeted manner and this must not lead to a situation where the function block becomes inactive and a lower priority light telegram switches the actuators.

Normally, a falling edge at presence input *bPrc* deactivates the function. As this is not possible without presence detection, the light function is preceded by a logic that sends a trigger pulse to the reset input *bRst* when 0% is reached at the light output and thus deactivates the light control input anyway. Alternatively, deactivation can also be carried out via the template input *bResetRoomFunction*.

If the light is switched on, the function block LightEvent [▶ 309] is activated by *bManualMode* = TRUE and a light telegram [▶ 272] with the set light value is output.

In addition to the internal telegram from *LightEvent*, the input-side telegrams *stAreaLighting* and *stScene* are also routed to a selector (*PrioSwi*).

Here, *stArea* represents the resulting telegram from the higher levels (area, floor and building), while *stScene* is reserved for any lighting scene to override the constant light regulation.

The telegram selector (*PrioSwi*) forwards the telegram with the highest priority - with the same priority, the last telegram sent (last writer wins).

Input stReferenceFeedback

Information about the controlled light actuator or the reference actuator of a group is fed back into the light control function via this input.

The light value of the reference actuator is of particular importance here: If the light actuator is already controlled by another function of the same or lower priority and the template described here now "takes over" control, it must control the actuator based on its existing light value.

In addition, current values and states are transmitted and can be displayed via the HMI.

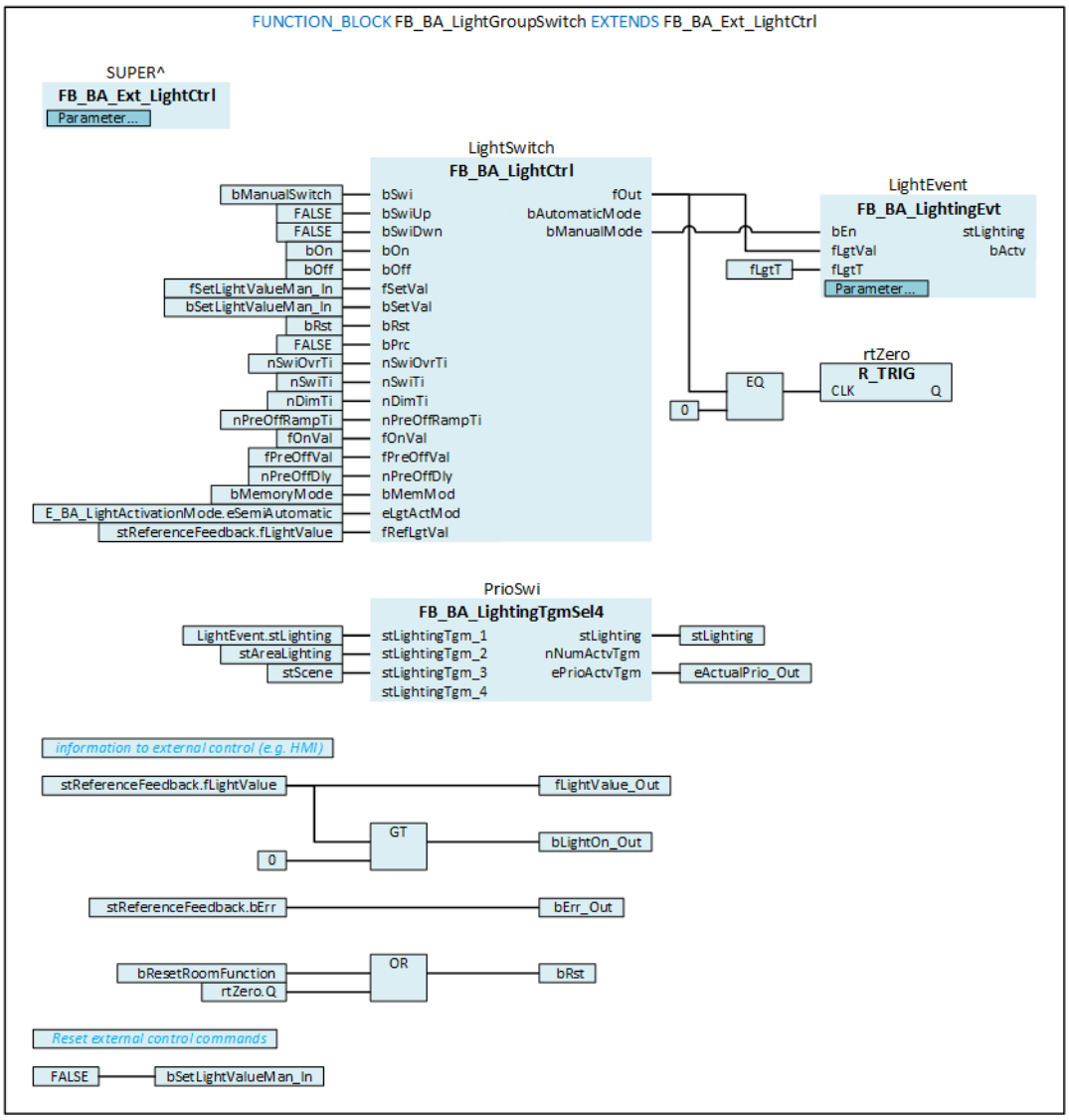
Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_Ext_LightCtrl* (internal function block). The use of the following variables is visible in this template:

- **fSetLightValueMan_In**: Light value to be set via the HMI.
- **bSetLightValueMan_In**: Command from the HMI that the above light value is to be set.
- **bLightOn_Out**: Output information "Light On" to the HMI.
- **fLightValue_Out**: Output information light value to the HMI.
- **bErr_Out**: Output information "Reference actuator faulty" to the HMI.
- **eActualPrio_Out**: Currently controlling telegram priority to the HMI.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupSwitch EXTENDS FB_BA_Ext_LightCtrl
VAR_INPUT
  bManualSwitch           : BOOL;
  bOn                     : BOOL;
  bOff                    : BOOL;
  bResetRoomFunction      : BOOL;
  stAreaLighting          : ST_BA_Lighting;
  stScene                 : ST_BA_Lighting;
  stReferenceFeedback     : ST_BA_LightActuatorFeedback;
END_VAR
VAR_OUTPUT
  stLighting              : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiOvrTi               : UDINT;
  nSwiTi                  : UDINT;
  nDimTi                  : UDINT;
  nPreOffRampTi           : UDINT;
  fOnVal                  : REAL;
  fPreOffVal              : REAL;
  nPreOffDly              : UDINT;
  bMemoryMode             : BOOL;
  eLightActivationMode    : E_BA_LightActivationMode;
  fLgtT                   : REAL;
END_VAR
VAR_INPUT CONSTANT
  LightEvent              : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eSimple);
END_VAR
VAR
```

```
  LightSwitch             : FB_BA_LightCtrl;
  PrioSwi                 : FB_BA_LightingTgmSel4;
  rtZero                  : R_TRIG;
  bRst                    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bManualSwitch	BOOL	Short button presses at this input switch the light on and off. Long button presses (t > *SwiOvrTi*) cause the light to dim up and down.
bOn	BOOL	Switches the light on and activates the light control block [▶ 920].
bOff	BOOL	Switches the light off. Due to an additional circuit that sends a trigger pulse to the *bRst* input when 0% is reached on the light control block [▶ 920], the block is also deactivated at the same time.
bResetRoomFunction	BOOL	This input switches off the light function block *LightSwitch* - it is then no longer active (output *bAutmaticMode*/ *bManualMode* = FALSE). Switch-off takes place via a ramp and dwell time on a basic light value, see parameters: *fPreOffVal*, *nPreOffDly* and *nPreOffRampT*.
stAreaLighting	ST_BA_Lighting [▶ 272]	Resulting telegram from the higher levels (area, floor and building), which is formed in the function block FB_BA_AreaLighting [▶ 920].
stScene	ST_BA_Lighting [▶ 272]	Reserved telegram input for scene control.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback input of the controlled light actuator or the guide light of the controlled group.

Outputs

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	Resulting output program.

Inputs CONSTANT PERSISTENT

Name	Type	Description
nSwiOvrTi	UDINT	Distinction time [ms] between short and long button press. Pre-set to 250 ms in *FB_Init*.
nSwiTi	UDINT	Ramp for the switching functions in seconds, related to a dimming from 0 to 100%. Pre-set to 2 s in *FB_Init*.
nDimTi	UDINT	Ramp for the dimming functions in seconds, related to a dimming from 0 to 100%. Pre-set to 10 s in *FB_Init*.
nPreOffRampTi	UDINT	Ramp used to drive to a base value *fPreOffVal* before switching off. Pre-set to 2 s in *FB_Init*.
fOnVal	REAL	Switch-on value. Pre-set to 100% in *FB_Init*.
fPreOffVal / nPreOffDly	REAL / 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. Pre-set to 4% and 10 s in *FB_Init*.
bMemoryMode	BOOL	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 *fOnValMan*. Pre-set to FALSE in *FB_Init*.
eLightActivationMode	E_BA_LightActivationMode [▶ 267]	Selection of the automatic lighting function: • Semi-automatic: Activation of the light control function only by pressing a button; it is automatically deactivated when not occupied. • Fully automatic: The light function can be activated either by presence detection or by pressing a button; it is automatically deactivated if there is no presence.
fLgtT	REAL	Light temperature: Pre-set to 3000 K in *FB_Init*.

Inputs CONSTANT

Name	Type	Description
LightEvent	FB_BA_LightingEvt [▶ 309]	Telegram generator for the light values. The priority is pre-set to *eManualActuator*.

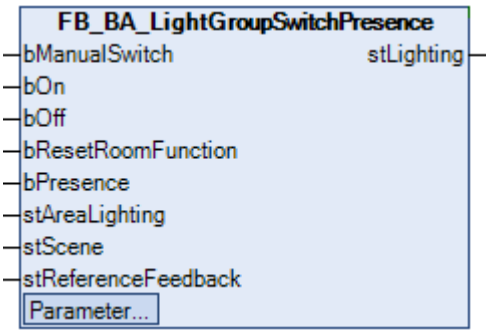
VAR

Name	Type	Description
LightSwitch	FB_BA_LightCtrl [▶ 311]	Light control block.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 310]	Telegram selection.
rtZero	R_TRIG	Trigger that is activated by the light output value "0".
bRst	BOOL	Internal variable for deactivating the overall function. A valid light control telegram *stLighting* is no longer output.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.4.1.1.4 FB_BA_LightGroupSwitchPresence



One or more light actuators are combined in a light zone for simultaneous control, depending on local conditions.

This template represents a light zone (room) that is operated both via switches and occupancy sensors.

Function

The light can be switched on and off alternately via the template input *bManualSwitch* using the control block LightSwitch or dimmed up and down with a long button press (t > *SwiOvrTi*).

The inputs *bOn* and *bOff* allow the light to be switched on and off selectively, while the light value *fSetLightValueMan_In* is set directly via a HMI using the *bSetLightValueMan_In* command.

If fully automatic mode is set via the parameter *eLightActivationMode*, the light switches on automatically when presence is detected at the input *bPresence* and switches off automatically on a falling edge. However, if semi-automatic mode is selected, the presence input is only used to switch off.

The light control block distinguishes whether it has been activated or overridden manually or by presence. In the first case, the output *bAutomaticMode* is set, in the second case *bManualMode*; both outputs are not set at the same time. These outputs are used either to generate a light output telegram with the priority eAutomaticLight [▶ 268] via the function block AutoMode [▶ 309] or a telegram with the priority eManual [▶ 268] via ManMode [▶ 309]. The different telegram priority is assigned to the internal HMI variable *eActualPrio_Out* on the PrioSwi function block. For example, an HMI can be used to display a distinction between automatic or manual/manual override.

On a falling edge at the template input *bPresence* or by setting *bResetRoomFunction*, the light control block is deactivated, whereby, if selected, it still announces its switch-off via a switch-off ramp with a dwell time at a pre-switch-off value. Both function block outputs *bAutomaticMode* and *bManualMode* are then deleted.

In addition to the internal telegrams from *AutoMode* and *ManMode*, the input-side stAreaLighting and stScene are also routed to the selector (PrioSwi).

Here, *stArea* represents the resulting telegram from the higher levels (area, floor and building), while *stScene* is reserved for any lighting scene to override the constant light regulation.

The telegram selector (*PrioSwi*) forwards the telegram with the highest priority - with the same priority, the last telegram sent (last writer wins).

Input stReferenceFeedback

Information about the controlled light actuator or the reference actuator of a group is fed back into the light control function via this input.

The light value of the reference actuator is of particular importance here: If the light actuator is already controlled by another function of the same or lower priority and the template described here now "takes over" control, it must control the actuator based on its existing light value.

In addition, current values and states are transmitted and can be displayed via the HMI.

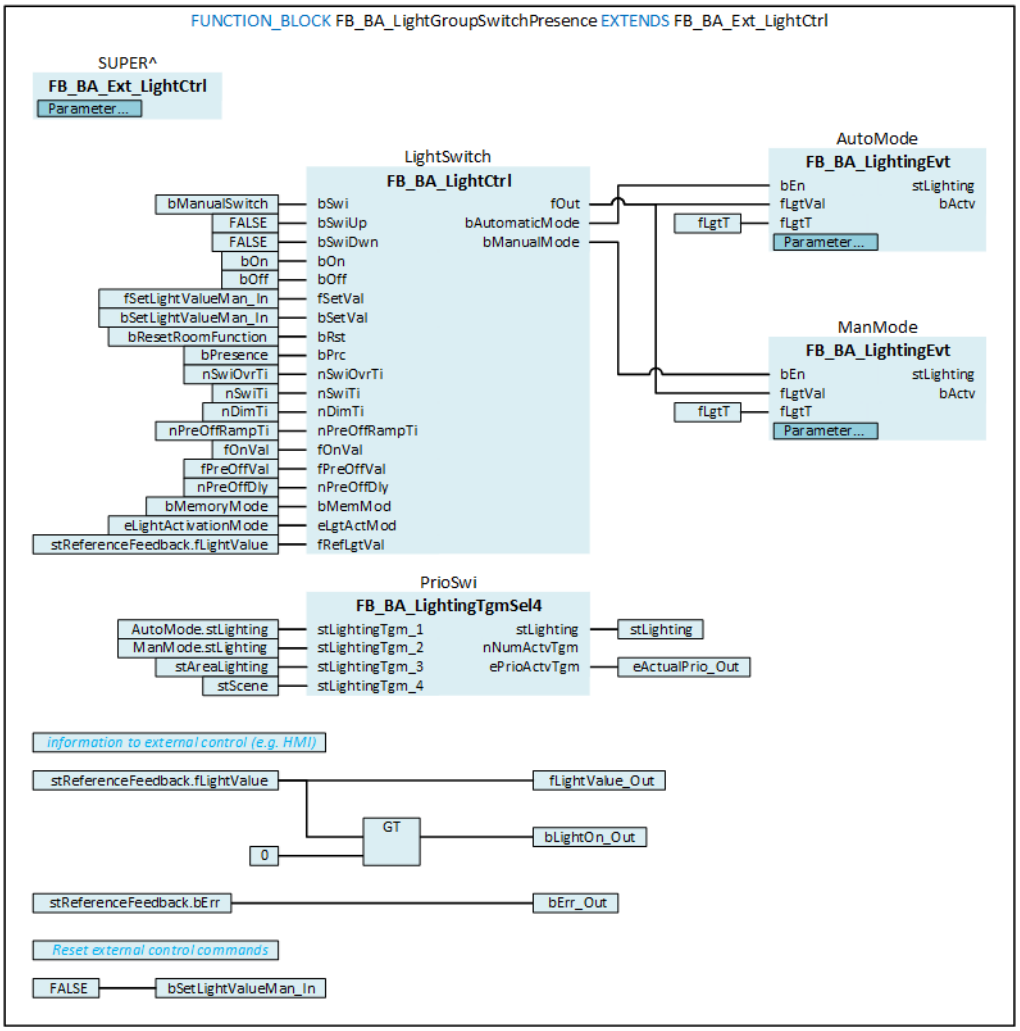
Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_Ext_LightCtrl* (internal function block). The use of the following variables is visible in this template:

- **fSetLightValueMan_In**: Light value to be set via the HMI.
- **bSetLightValueMan_In**: Command from the HMI that the above light value is to be set.
- **bLightOn_Out**: Output information "Light On" to the HMI.
- **fLightValue_Out**: Output information light value to the HMI.
- **bErr_Out**: Output information "Reference actuator faulty" to the HMI.
- **eActualPrio_Out**: Currently controlling telegram priority to the HMI.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupSwitch EXTENDS FB_BA_Ext_LightCtrl
VAR_INPUT
  bManualSwitch           : BOOL;
  bOn                     : BOOL;
  bOff                    : BOOL;
  bResetRoomFunction      : BOOL;
  bPresence               : BOOL;
  stAreaLighting          : ST_BA_Lighting;
  stScene                 : ST_BA_Lighting;
  stReferenceFeedback     : ST_BA_LightActuatorFeedback;
END_VAR
VAR_OUTPUT
  stLighting              : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
```

```
  nSwiOvrTi               : UDINT;
  nSwiTi                  : UDINT;
  nDimTi                  : UDINT;
  nPreOffRampTi           : UDINT;
  fOnVal                  : REAL;
  fPreOffVal              : REAL;
  nPreOffDly              : UDINT;
  bMemoryMode             : BOOL;
  eLightActivationMode    : E_BA_LightActivationMode;
  fLgtT                   : REAL;
END_VAR
VAR_INPUT CONSTANT
  ManMode                 : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eManual);
  AutoMode                : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eAutomaticLight);
END_VAR
VAR
  LightSwitch             : FB_BA_LightCtrl;
  PrioSwi                 : FB_BA_LightingTgmSel4;
END_VAR
```

Inputs

Name	Type	Description
bManualSwitch	BOOL	Short button presses at this input switch the light on and off. Long button presses (t > *SwiOvrTi*) cause the light to dim up and down.
bOn	BOOL	Switches the light on and activates the light control block [▶ 920].
bOff	BOOL	Switches the light off. Due to an additional circuit that sends a trigger pulse to the *bRst* input when 0% is reached on the light control block [▶ 920], the block is also deactivated at the same time.
bResetRoomFunction	BOOL	This input switches off the light function block *LightSwitch* - it is then no longer active (output *bAutmaticMode*/ *bManualMode* = FALSE). Switch-off takes place via a ramp and dwell time on a basic light value, see parameters: *fPreOffVal*, *nPreOffDly* and *nPreOffRampT*.
bPresence	BOOL	Presence signal input. If fully automatic is selected via the parameter eLightActivationMode [▶ 267], the light function block [▶ 311] is activated as "automatic" with a rising edge (output *bAutomaticMode* = TRUE), provided it was not previously set to manual/manual override (output *bManualMode* = TRUE). A falling edge deactivates the light control block (output bAutomaticMode=FALSE, bMaualMode=FALSE).
stAreaLighting	ST_BA_Lighting [▶ 272]	Resulting telegram from the higher levels (area, floor and building), which is formed in the function block FB_BA_AreaLighting [▶ 920].
stScene	ST_BA_Lighting [▶ 272]	Reserved telegram input for scene control.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback input of the controlled light actuator or the guide light of the controlled group.

Outputs

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	Resulting output program.

Inputs CONSTANT PERSISTENT

Name	Type	Description
nSwiOvrTi	UDINT	Distinction time [ms] between short and long button press. Pre-set to 250 ms in *FB_Init*.
nSwiTi	UDINT	Ramp for the switching functions in seconds, related to a dimming from 0 to 100%. Pre-set to 2 s in *FB_Init*.
nDimTi	UDINT	Ramp for the dimming functions in seconds, related to a dimming from 0 to 100%. Pre-set to 10 s in *FB_Init*.
nPreOffRampTi	UDINT	Ramp used to drive to a base value *fPreOffVal* before switching off. Pre-set to 2 s in *FB_Init*.
fOnVal	REAL	Switch-on value. Pre-set to 100% in *FB_Init*.
fPreOffVal / nPreOffDly	REAL / 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. Pre-set to 4% and 10 s in *FB_Init*.
bMemoryMode	BOOL	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 *fOnValMan*. Pre-set to FALSE in *FB_Init*.
eLightActivationMode	E_BA_LightActivationMode [▶ 267]	Selection of the automatic lighting function: • Semi-automatic: Activation of the light control function only by pressing a button; it is automatically deactivated when not occupied. • Fully automatic: The light function can be activated either by presence detection or by pressing a button; it is automatically deactivated if there is no presence.
fLgtT	REAL	Light temperature: Pre-set to 3000 K in *FB_Init*.

Inputs CONSTANT

Name	Type	Description
ManMode	FB_BA_LightingEvt [▶ 309]	Telegram generator for the light values. The priority is pre-set to *eManualActuator*.
AutoMode	FB_BA_LightingEvt [▶ 309]	Telegram generator for the light values if the function block has been activated via the occupancy sensor and is not (yet) manually overridden. The priority is pre-set to *eAutomaticLight*.

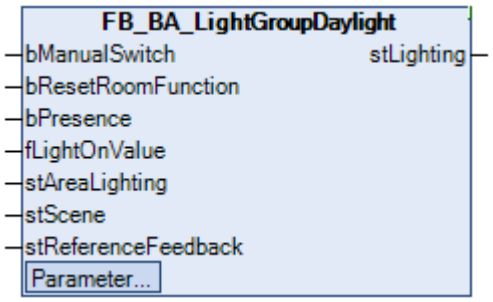
VAR

Name	Type	Description
LightSwitch	FB_BA_LightCtrl [▶ 311]	Light control block.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 310]	Telegram selection.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.4.1.1.5 FB_BA_LightGroupDaylight



This template represents a daylight-dependent light control that automatically switches on and off depending on occupancy. It is intended for corridors and passageways. An upstream, optimizing switch-on function FB_BA_LightCtrlValue [▶ 314] ensures the specification of a light switch-on value that switches the downstream constant light regulation close to the desired setpoint. As soon as the light is switched on via occupancy, it is automatically readjusted according to the setpoint.

Function

In contrast to the FB_BA_LightGroupConstant [▶ 894] template (constant light regulation), only the control mode of the function block FB_BA_LightCtrlConst [▶ 317] is used here. It is not necessary to manually override the constant light regulation for corridors and passageways, which is why there is no switch input. If manual switch-on is nevertheless required, the switch should be treated as an occupancy sensor and **not** linked to the inputs *bSwi*, *bSwiUp* or *bSwiDwn*, as these trigger the override without control.

A rising edge at *bPresence* activates the function block and it is set directly to the switch-on value optimized by FB_BA_LightCtrlValue [▶ 314].

On a falling edge at the template input *bPresence* or by setting *bResetRoomFunction*, the light control block is disabled, whereby, if selected, it still announces its shutdown via a switch-off ramp with a dwell time at a pre-switch-off value. The function is then inactive and the output *bControlMode* is deleted.

In addition to the internal telegrams from *LightEvent* and *OnValue* (teach-in function), the input-side *stAreaLighting* and *stScene* are also routed to the selector (*PrioSwi*).

Here, *stArea* represents the resulting telegram from the higher levels (area, floor and building), while *stScene* is reserved for any lighting scene to override the constant light regulation.

The *OnValue.stTeachLighting* telegram is only active during the teach-in process of *OnValue* (FB_BA_LightCtrlValue [▶ 314]).

The telegram selector (*PrioSwi*) forwards the telegram with the highest priority - with the same priority, the last telegram sent (last writer wins).

The setpoint for constant light regulation is set using an analog object SpBrightness [▶ 202].

Input stReferenceFeedback

Information about the controlled light actuator or the reference actuator of a group is fed back into the light control function via this input.

The light value of the reference actuator is of particular importance here: If the light actuator is already controlled by another function of the same or lower priority and the template described here now "takes over" control, it must control the actuator based on its existing light value.

The light preset block *fOnValue* also requires information on the current light value of the controlled group. Since it can only determine the increase in brightness itself, it can only specify the increase on the current light value - the absolute value is the sum of the current light value and the increase.

In addition, current values and states are transmitted and can be displayed via the HMI.

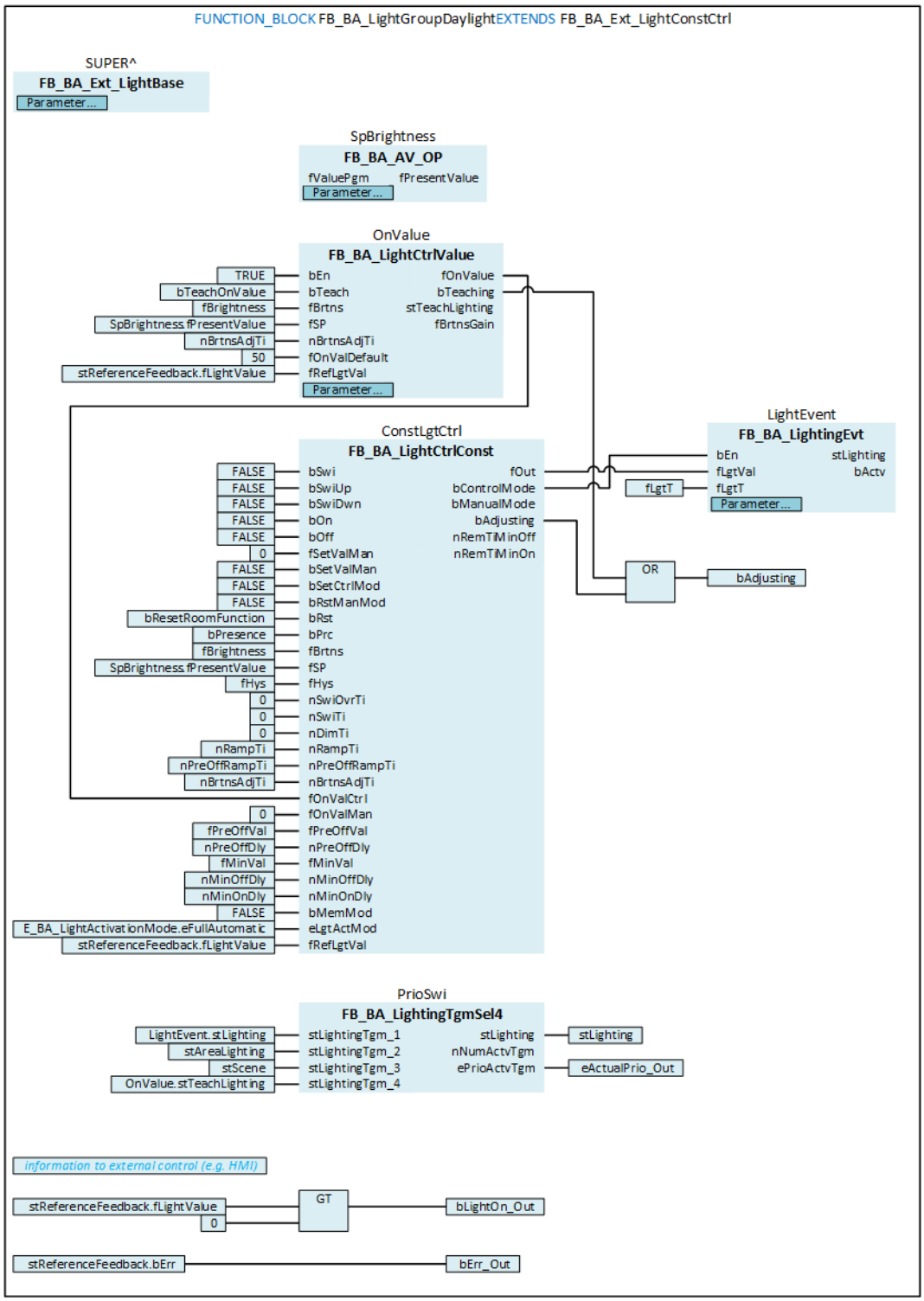
Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_Ext_LightBase* (internal function block). The use of the following variables is visible in this template:

- **bLightOn_Out**: Output information "Light On" to the HMI.
- **bErr_Out**: Output information "Reference actuator faulty" to the HMI.
- **eActualPrio_Out**: Currently controlling telegram priority to the HMI.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupDaylight EXTENDS FB_BA_Ext_LightBase
VAR_INPUT
  bResetRoomFunction     : BOOL;
  bPresence              : BOOL;
  bTeachOnValue          : BOOL;
  fBrightness            : REAL;
  stAreaLighting         : ST_BA_Lighting;
  stScene                : ST_BA_Lighting;
  stReferenceFeedback    : ST_BA_LightActuatorFeedback;
END_VAR

VAR_OUTPUT
  stLighting             : ST_BA_Lighting;
```

```
  bAdjusting              : BOOL;
END_VAR

VAR_INPUT CONSTANT PERSISTENT
  fHys                    : REAL;
  nRampTi                 : UDINT;
  nPreOffRampTi           : UDINT;
  nBrtnsAdjTi             : UDINT;
  fPreOffVal              : REAL;
  nPreOffDly              : UDINT;
  fMinVal                 : REAL;
  nMinOffDly              : UDINT;
  nMinOnDly               : UDINT;
  fLgtT                   : REAL;
END_VAR

VAR_INPUT CONSTANT
  SpBrightness            : FB_BA_AV_Op;
  OnValue                 : FB_BA_LightCtrlValue := ( ePrio := E_BA_LightingPrio.eMaintenance);
  LightEvent              : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eSimple);
END_VAR
VAR
  ConstLgtCtrl            : FB_BA_LightCtrlConst;
  PrioSwi                 : FB_BA_LightingTgmSel4;
END_VAR
```

Inputs

Name	Type	Description
bResetRoomFunction	BOOL	This input switches off the constant light control function block. Switch-off takes place via a ramp and dwell time on a basic light value, see parameters: *fPreOffVal*, *nPreOffDly* and *nPreOffRampT*.
bPresence	BOOL	Presence signal input. In the template, the constant light control block is preconfigured in fully automatic mode so that a rising edge at this input switches the function on and a falling edge switches the function off.
bTeachOnValue	BOOL	Starts the teach-in process on the function block FB_BA_LightCtrlValue [▶ 314]. A light control telegram with priority *eMaintenance* is used to switch the light once to 0% and once to 100%. Due to the high priority of the telegram (*eMaintenance*), manual and automatic telegrams on the priority switch *PrioSwi* are overridden.
fBrightness	REAL	Current brightness for constant light automatic: Range and unit depend on the light sensor.
stAreaLighting	ST_BA_Lighting [▶ 272]	Resulting telegram from the higher levels (area, floor and building), which is formed in the function block FB_BA_AreaLighting [▶ 920].
stScene	ST_BA_Lighting [▶ 272]	Reserved telegram input for scene control.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback input of the controlled light actuator or the guide light of the controlled group.

Outputs

Name	Type	Description
stLighting	ST_BA_Lighting [▶ 272]	Resulting output program.
bAdjusting	BOOL	The constant light control block is in regulation mode or the switch-on function is in teach-in 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.

Inputs CONSTANT PERSISTENT

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 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 depend on the used light sensor, see above: *fBrtnsSP* and *fBrtnsSen*. Pre-set to 50 in FB_Init.
nRampTi	UDINT	Control ramp of the constant light automatic in seconds, related to a dimming from 0 to 100%. Pre-set to 60 s in FB_Init.
nPreOffRampTi	UDINT	Ramp used to drive to a base value *fPreOffVal* before switching off. Pre-set to 2 s in *FB_Init*.
nBrtnsAdjTi	UDINT	Waiting time in seconds after switching on the light for the light sensor to detect the correct value. Is used both by the light preset function *OnValue* and by the light control block *ConstLgtCtrl*.
fPreOffVal / nPreOffDly	REAL / 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. Pre-set to 4% and 10 s in *FB_Init*.
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 *nOffDly* (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 *nOnDly* (in seconds) to initially *fMinVal*. Pre-set to 4% in FB_Init
nMinOffDly	UDINT	Constant light automatic: Switch-off waiting time in seconds, see *fMinVal*. Pre-set to 300 s in FB_Init.
nMinOnDly	UDINT	Constant light automatic: Switch-on waiting time in seconds, see *fMinVal*. Pre-set to 300 s in FB_Init.
fLgtT	REAL	Light temperature: Pre-set to 3000 K in *FB_Init*.

Inputs CONSTANT

Name	Type	Description
SpBrightness	FB_BA_AV_Op [▶ 202]	Analog input object: Setpoint of the control [lx].
OnValue	FB_BA_LightCtrlValue [▶ 314]	Optimizing switch-on function
LightEvent	FB_BA_LightingEvt [▶ 309]	Telegram generator for the light values. The priority is pre-set to *eManualActuator*.

VAR

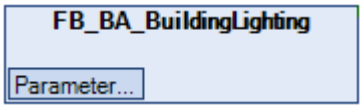
Name	Type	Description
ConstLgtCtrl	FB_BA_LightCtrlConst [▶ 317]	Constant light control block.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 310]	Telegram selection.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.4.1.2 Building

6.1.4.2.1.4.4.1.2.1 FB_BA_BuildingLighting



This template compiles cross-building control telegrams for the light functions.

The "fire" and "burglary" alarms are usually recorded in the building controller (FB_BA_FireAlarmSystem and FB_BA_BurglarAlarmSystem). From there, they are sent with other information via the communication server (FB_BA_AdsComServer_Basic [▶ 847]) to the floor controllers, where they are received in the counter-block FB_BA_AdsComClient_Basic [▶ 848] and made available centrally in the local variable list Site [▶ 1116].

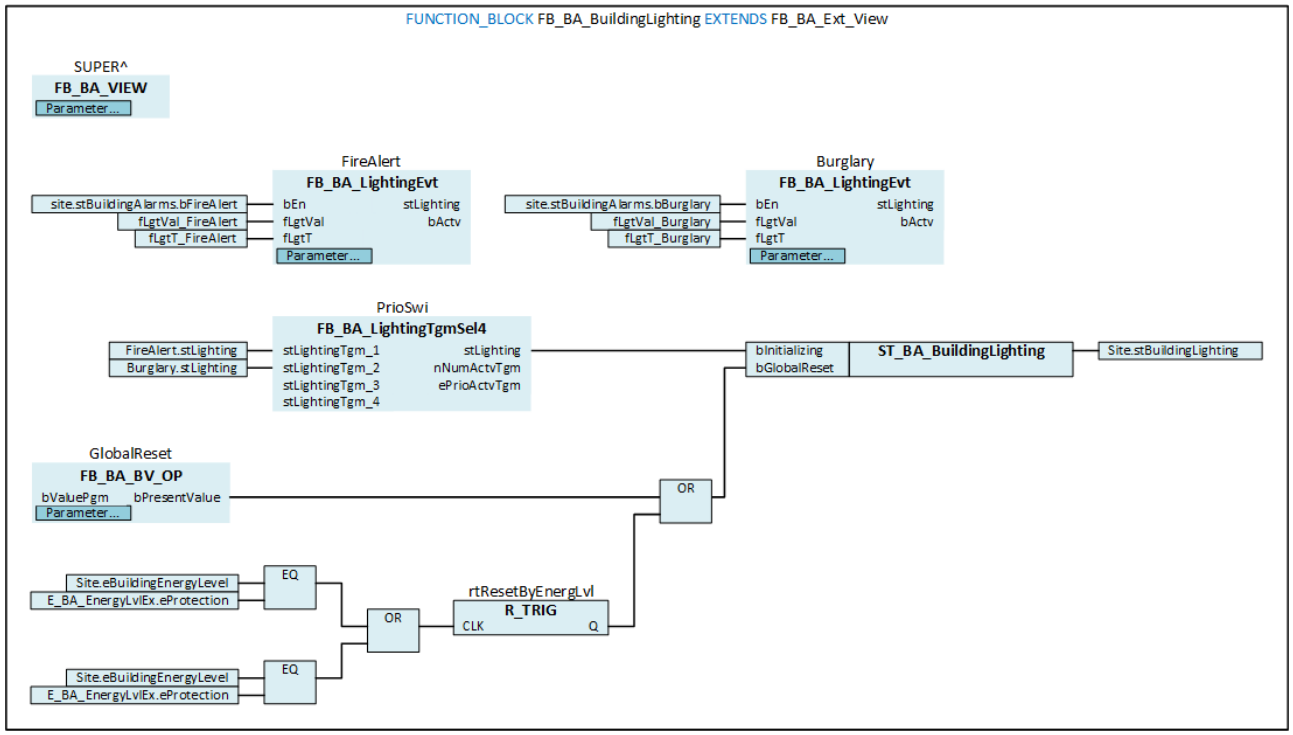
The fire and burglary alarms can each trigger a separate light telegram via the FireAlert and Burglary Event function blocks [▶ 309]. The telegrams are routed to a priority function block PrioSwi [▶ 310].

In addition to the alarms, a global reset of the light functions is provided in this template. On the one hand, it consists of manual triggering via a GlobalReset [▶ 216] input object, which is parameterized in FB_Init as a push button and on the other hand, a trigger signal is generated when the building energy level "Protection" or "Economy" is reached, which is intended to signal "Absence".

The resulting light telegram from *PrioSwi* and the global reset "Light" are made available via the stBuildingLighting [▶ 702] structure in the local variable list Site [▶ 1116].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_BuildingLighting EXTENDS FB_BA_View
VAR_INPUT CONSTANT PERSISTENT
  fLgtVal_FireAlert     : REAL;
  fLgtT_FireAlert       : REAL;
  fLgtVal_Burglary      : REAL;
  fLgtT_Burglary        : REAL;
END_VAR
VAR_INPUT CONSTANT
  FireAlert             : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eFire);
  Burglary              : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eBurglary);
  GlobalReset           : FB_BA_BV_Op;
END_VAR
VAR
  rtResetByEnergLvl     : R_TRIG;
  PrioSwi               : FB_BA_LightingTgmSel4;
END_VAR
```

VAR_INPUT CONSTANT PERSISTENT

Name	Type	Description
fLgtVal_FireAlert	REAL	Input light value [%] in case of fire. Pre-set to 100% in *FB_Init*.
fLgtT_FireAlert	REAL	Input light temperature [K] in case of fire. Pre-set to 4000 K in *FB_Init*.
fLgtVal_Burglary	REAL	Input light value [%] in case of burglary. Pre-set to 100% in *FB_Init*.
fLgtT_Burglary	REAL	Enter the light temperature [K] in case of burglary. Pre-set to 4000 K in *FB_Init*.

VAR_INPUT CONSTANT

Name	Type	Description
FireAlert	FB_BA_LightingEvt [▶ 309]	Event telegram block for the fire alarm.
Burglary	FB_BA_LightingEvt [▶ 309]	Event telegram block for the burglary alarm.
GlobalReset	FB_BA_BV_Op [▶ 216]	Binary input object: Operating option to reset the light functions. This object is parameterized as non-latching in *FB_Init*, i.e. it generates a TRUE pulse.

VAR

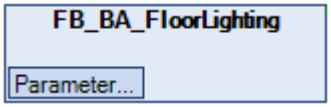
Name	Type	Description
rtResetByEnergLvl	R_TRIG	Triggers the reset signal generated from the energy levels.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 310]	Telegram selection block for the light telegrams.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.4.1.3 Floor

6.1.4.2.1.4.4.1.3.1 FB_BA_FloorLighting



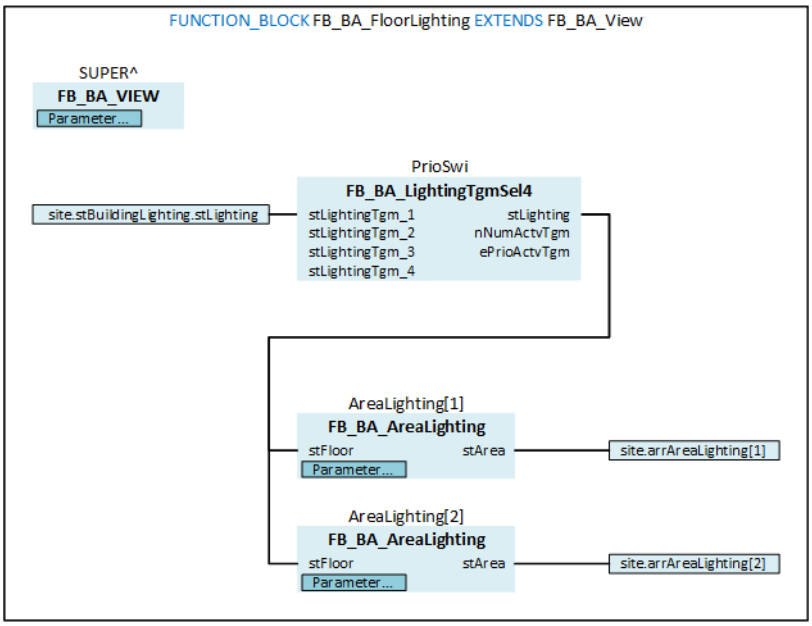
This template is used to implement various light functions at floor level. It has a very simple design and can be expanded as required.

The resulting building light telegram from the local variable list Site [▶ 1116] is routed to the PrioSwi [▶ 310] priority function block, which provides further inputs for light telegrams from schedules, for example. The resulting telegram is sent to the inputs of the function blocks *AreaLighting[1]* and *AreaLighting[2]*. Their output telegrams are routed back to the local variable list Site [▶ 1116].

The lighting area block [▶ 920] contains exemplary schedules and manual controls. The function blocks listed there can be copied and amended or reduced for this template.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_FloorLighting EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AreaLighting    : ARRAY[1..BA2_Param.nMaxNumberOfAreas] OF FB_BA_AreaLighting;
END_VAR
VAR
  PrioSwiLight    : FB_BA_LightingTgmSel4;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
AreaLighting	FB_BA_AreaLighting [▶ 920]	Lighting area block with telegram functions that affect the area within a floor.

VAR

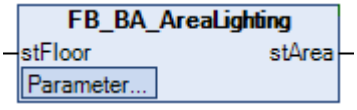
Name	Type	Description
PrioSwiLight	FB_BA_LightingTgmSel4 [▶ 310]	Telegram selection block for the light telegrams.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.4.1.4 Area

6.1.4.2.1.4.4.1.4.1 FB_BA_AreaLighting



Selection of lighting operation for an area based on a schedule (*Sched*) and manual operation (*OpModMan*).

The **manual operation** has the following modes:

Value	Meaning
1	Automatic mode.
2	Manual selection Night watchman tour.
3	Manual selection Building cleaning.
4	Manual selection Maintenance.
5	Manual selection User-defined 1.
6	Manual selection User-defined 2.

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:

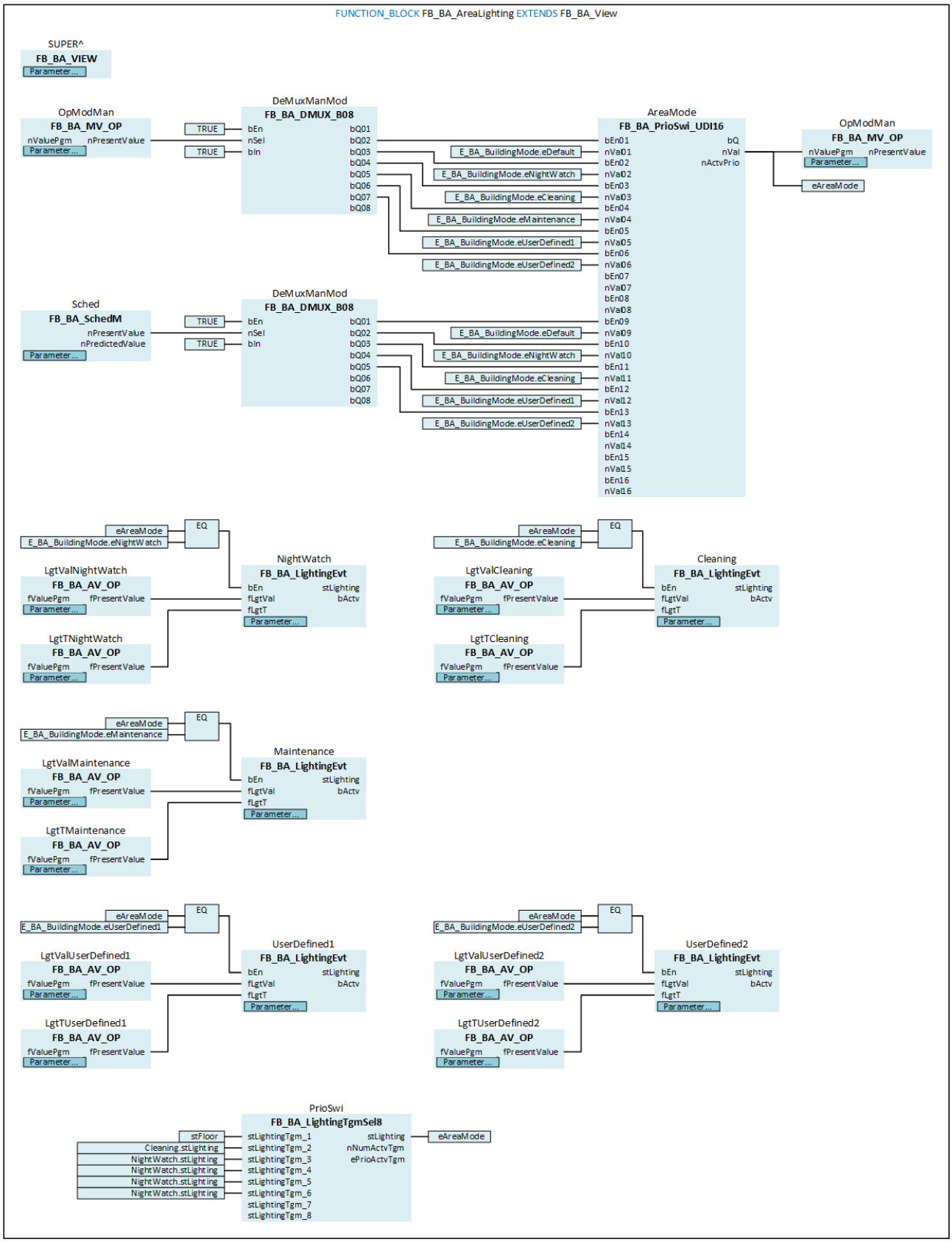
Value	Meaning
1	Default.
2	Night watchman tour.
3	Building cleaning.
4	User-defined 1.
5	User-defined 2.

The currently selected lighting mode *eAreaMode*, which is of the same type as the building mode [▶ 696], is then displayed via the *OpModPr* function block.

One of the Event function blocks [▶ 309] is enabled based on the currently valid lighting mode. All possible light telegrams are routed to the priority function block *PrioSwi* in addition to the floor telegram *stFloor* on the input side. The resulting telegram is available to the room templates at the *stArea* output.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AreaLighting EXTENDS FB_BA_View
VAR_INPUT
  stFloor               : ST_BA_Lighting;
END_VAR
VAR_OUTPUT
```

```
  stArea                : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT
  OpModMan              : FB_BA_MV_Op;
  OpModPr               : FB_BA_MV_Op;
  Sched                 : FB_BA_SchedM;

  LgtValNightWatch      : FB_BA_AV_Op;
  LgtTNightWatch        : FB_BA_AV_Op;
  NightWatch            : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eNightWatch);

  LgtValCleaning        : FB_BA_AV_Op;
  LgtTCleaning          : FB_BA_AV_Op;
  Cleaning              : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eCleaning);

  LgtValMaintenance     : FB_BA_AV_Op;
  LgtTMaintenance       : FB_BA_AV_Op;
  Maintenance           : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eMaintenance);

  LgtValUserDefined1    : FB_BA_AV_Op;
  LgtTUserDefined1      : FB_BA_AV_Op;
  UserDefined1          : FB_BA_LightingEvt := ( ePrio := 10);

  LgtValUserDefined2    : FB_BA_AV_Op;
  LgtTUserDefined2      : FB_BA_AV_Op;
  UserDefined2          : FB_BA_LightingEvt := ( ePrio := 10);
END_VAR
VAR
  DeMuxManMod           : FB_BA_DMUX_B08;
  DeMuxSched            : FB_BA_DMUX_B08;
  AreaMode              : FB_BA_PrioSwi_UDI16;
  eAreaMode             : E_BA_BuildingMode;
  PrioSwi               : FB_BA_LightingTgmSel8;
END_VAR
```

Inputs

Name	Type	Description
stFloor	ST_BA_Lighting [▶ 272]	Light control telegram from the higher level "Floor".

Outputs

Name	Type	Description
stArea	ST_BA_Lighting [▶ 272]	Resulting light control telegram from this template, which represents the "Area" level.

Inputs CONSTANT

Name	Type	Description
OpModMan	FB_BA_MV_Op [▶ 239]	Manual input object for the building mode.
OpModPr	FB_BA_MV_Op [▶ 239]	Resulting mode
Sched	FB_BA_SchedM [▶ 225]	Schedule object (automatic) for the building mode.
LgtValNightWatch	FB_BA_AV_Op [▶ 202]	Input object for the night watchman tour light value.
LgtTNightWatch	FB_BA_AV_Op [▶ 202]	Input object for the light temperature night watchman tour.
NightWatch	FB_BA_LightingEvt [▶ 309]	Event function block Night watchman tour.
LgtValCleaning	FB_BA_AV_Op [▶ 202]	Input object for the building cleaning light value.
LgtTCleaning	FB_BA_AV_Op [▶ 202]	Input object for the building cleaning light temperature.
Cleaning	FB_BA_LightingEvt [▶ 309]	Event function block Building cleaning.
LgtValMaintenance	FB_BA_AV_Op [▶ 202]	Input object for the maintenance light value.
LgtTMaintenance	FB_BA_AV_Op [▶ 202]	Input object for the maintenance light temperature.
Maintenance	FB_BA_LightingEvt [▶ 309]	Event gunction block Maintenance.
LgtValUserDefined1	FB_BA_AV_Op [▶ 202]	Input object for the "User-defined1" light value.
LgtTUserDefined1	FB_BA_AV_Op [▶ 202]	Input object for the "User-defined1" light temperature.
UserDefined1	FB_BA_LightingEvt [▶ 309]	Event function block "User-defined1".
LgtValUserDefined2	FB_BA_AV_Op [▶ 202]	Input object for the "User-defined2" light value.
LgtTUserDefined2	FB_BA_AV_Op [▶ 202]	Input object for the "User-defined2" light temperature.
UserDefined2	FB_BA_LightingEvt [▶ 309]	Event function block "User-defined2".

VAR

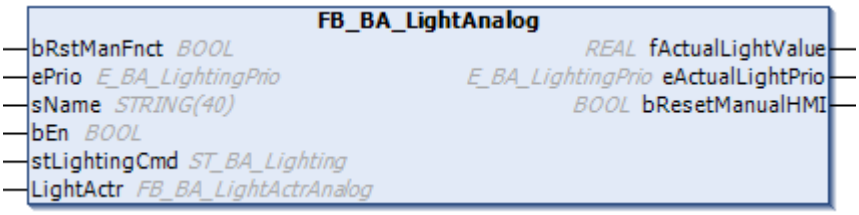
Name	Type	Description
DeMuxManMod	FB_BA_DMUX_B08 [▶ 426]	Conversion of the multistate value of the manual selection to a binary output.
DeMuxSched	FB_BA_DMUX_B08 [▶ 426]	Conversion of the multistate value of the schedule to a binary output.
AreaMode	FB_BA_PrioSwi_UDI16 [▶ 433]	Prioritizing reconversion of the states to a resulting multistate or enumeration value for lighting operation in the building area.
eAreaMode	E_BA_BuildingMode [▶ 696]	Currently valid lighting operation in building area.
PrioSwi	FB_BA_LightingTgmSel8 [▶ 310]	Telegram selection block for the light telegrams.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.4.2 Actuators

6.1.4.2.1.4.4.2.1 FB_BA_LightAnalog



This template is used to control an analog light actuator.

The core element is the analog output function block FB_BA_AO_Raw [▶ 199], which provides the light value in BACnet and also converts the light value from 0 to 100 % to 0 to 32767. The allocated variable *nLgtVal* enables a direct link to a dimmer terminal (KL2751, KL2761).

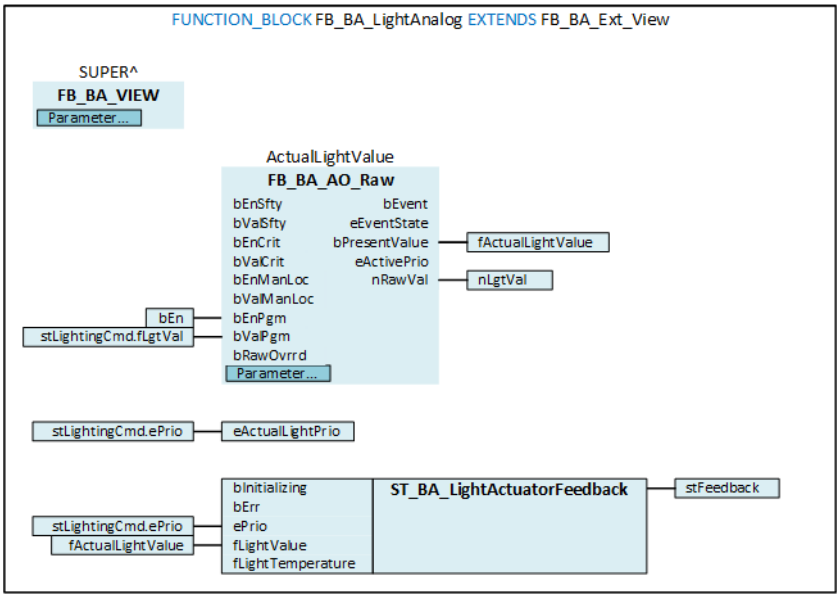
The function block ignores the color temperature when controlling the light actuator.

The template has a feedback structure stFeedback [▶ 702].

If the programmed light actuator is an individual actuator or the reference actuator of a group, this feedback telegram must be linked to the controlling room (zone) user function in order to obtain information about the state of the actuator.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightAnalog  EXTENDS FB_BA_View
VAR_INPUT
  bEn                 : BOOL;
  stLightingCmd       : ST_BA_Lighting;
END_VAR
VAR_OUTPUT
  fActualLightValue   : REAL;
  eActualLightPrio    : BYTE;
  stFeedback          : ST_BA_LightActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT
  ActualLightValue    : FB_BA_AO_Raw;
```

```
END_VAR
VAR
  nLgtVal      AT %Q*  : INT;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enabling the function block.
stLightingCmd	ST_BA_Lighting [▶ 272]	Resulting telegram from the higher-level zone (room).

Outputs

Name	Type	Description
fActualLightValue	REAL	Current light value in percent.
eActualLightPrio	BYTE	Current priority of the command telegram *stLightingCmd*.
stFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback telegram for linking to the controlling room (zone) application function. In this way, information about the state of the light actuator is fed back into the application function.

Inputs CONSTANT

Name	Type	Description
ActualLightValue	FB_BA_AO_Raw [▶ 199]	Function block for converting the percentage light value to the value range 0..32767 and for simultaneous display in BACnet.

VAR

Name	Type	Description
nLgtVal	INT	Output variable for linking with the dimmer terminal.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.4.2.2 FB_BA_LightDALI_Base

This template is used to control a DALI light actuator, a DALI group or DALI devices with a broadcast, whereby only the light brightness value is changed.

The core is the light control block LightActr [▶ 874]. This establishes the connection to a DALI luminaire (or group) and converts changes in the light telegram *stLightingCmd* into DALI control signals. In addition, input *bInitialize* can be used to initialize the DALI luminaire (or group) with the parameters set on the function block (minimum value, maximum value, etc.).

It should be noted that if the light control value is changed in the command telegram, the output values *byActlLgtVal* and *fActlLgtVal* initially correspond to this control value. Internally, however, in the function block FB_BA_LightActrDALI_Base [▶ 874], the actual light value is queried cyclically. This ensures that the true state of the DALI luminaire is displayed without placing an unnecessary load on the DALI bus.

The function block ignores the color temperature when controlling the light actuator.

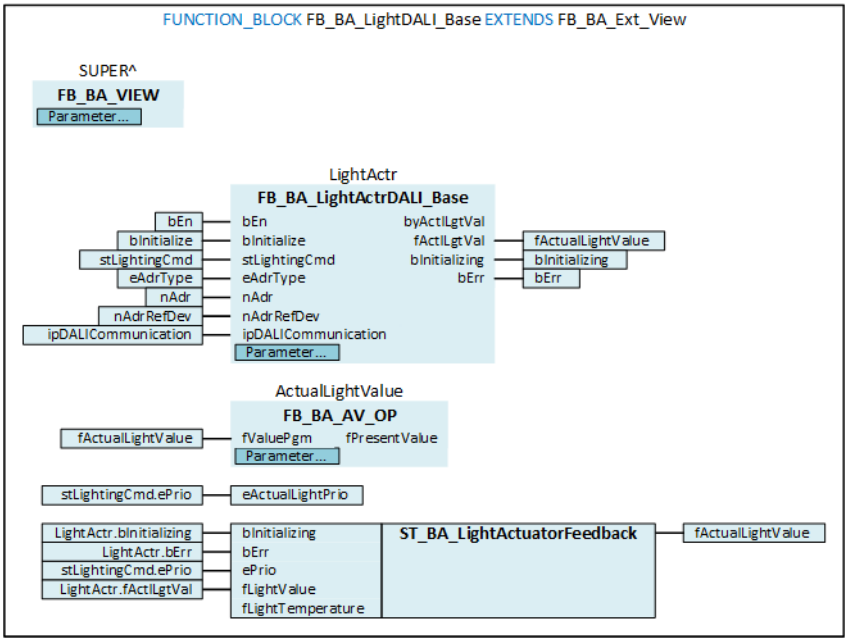
The current light value is made available in BACnet via an analog object FB_BA_AV_Op [▶ 202].

The template has a feedback structure stFeedback [▶ 702].

If the programmed light actuator is an individual actuator or the reference actuator of a group, this feedback telegram must be linked to the controlling room (zone) user function in order to obtain information about the state of the actuator.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightDALI_Base EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  stLightingCmd         : ST_BA_Lighting;
  bInitialize           : BOOL;
END_VAR
VAR_OUTPUT
  bInitializing         : BOOL;
  bErr                  : BOOL;
  fActualLightValue     : REAL;
  eActualLightPrio      : BYTE;
  stFeedback            : ST_BA_LightActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  eAdrType              : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                  : BYTE;
  nAdrRefDev            : BYTE;
  ipDALICommunication   : Tc3_DALI.I_DALICommunication;
```

```
END_VAR
VAR_INPUT CONSTANT
  LightActr              : FB_BA_LightActrDALI_Base;
  ActualLightValue       : FB_BA_AV_Op;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enabling the function block: A TRUE signal activates the function.
stLightingCmd	ST_BA_Lighting [▶ 272]	Command telegram from the higher-level zone (room).
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the control gears (see FB_BA_LightActrDALI_Base [▶ 874]).

Outputs

Name	Type	Description
bInitializing	BOOL	The light actuator or the actuators are in the DALI initialization phase, i.e. the entered parameters are transferred to the controlled control gears.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.
fActualLightValue	REAL	Current light value in percent.
eActualLightPrio	E_BA_LightingPrio [▶ 268]	Current priority of the command telegram *stLightingCmd*.
stFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback telegram for linking to the controlling room (zone) application function. In this way, information about the state of the light actuator is fed back into the application function.

Inputs CONSTANT PERSISTENT

Name	Type	Description
eAdrType	Tc3_DALI.E_DALIAddressType	Selection of the control type: individual, group or broadcast control, pre-set to individual control.
nAdr	BYTE	DALI address of the actuator or group.
nAdrRefDev	BYTE	DALI address of a reference device if group or broadcast control is selected. This reference device then represents all other controlled devices.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

Inputs CONSTANT

Name	Type	Description
LightActr	FB_BA_LightActrDALI_Base [▶ 874]	Light control block without color temperature control.
ActualLightValue	FB_BA_AV_Op [▶ 202]	Function block for displaying the percentage light value in BACnet.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.4.2.3 FB_BA_LightDALI_TC

This template is used to control a DALI light actuator, a DALI group or DALI devices with a broadcast, whereby color temperature control is supported.

The core is the light control block LightActr [▶ 877]. This establishes the connection to a DALI luminaire (or group) and converts changes in the light telegram *stLightingCmd* into DALI control signals. In addition, input *bInitialize* can be used to initialize the DALI luminaire (or group) with the parameters set on the function block (minimum value, maximum value, etc.).

It should be noted that if the light control value or the color temperature is changed in the command telegram, the output values *byActlLgtVal*, *fActlLgtVal* and *fActlLgtT* initially correspond to these control values. Internally, however, in the function block FB_BA_LightActrDALI_TC [▶ 877], the actual light value and the actual color temperature are queried cyclically. This ensures that the true state of the DALI luminaire is displayed without placing an unnecessary load on the DALI bus.

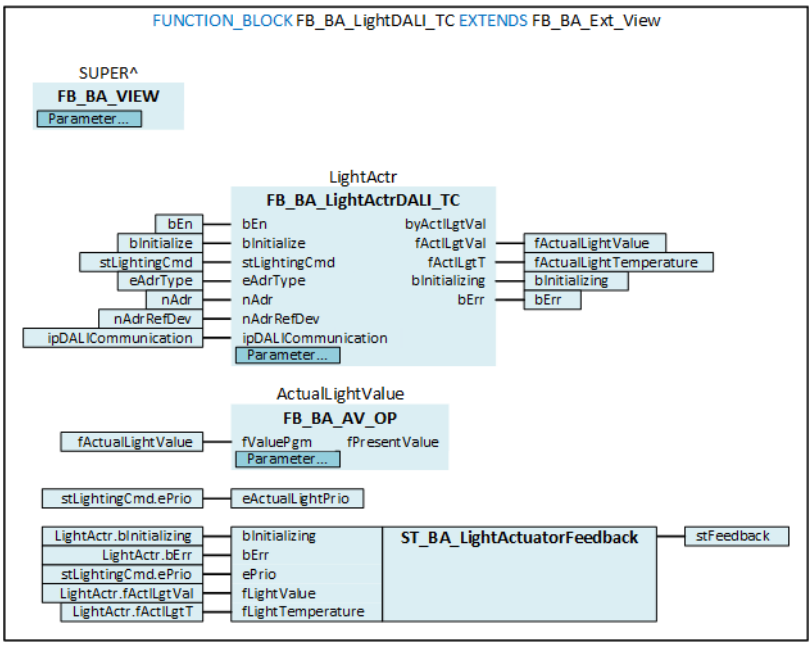
The current light value is made available in BACnet via an analog object FB_BA_AV_Op [▶ 202], while the current color temperature is only displayed as a template output.

The template has a feedback structure stFeedback [▶ 702].

If the programmed light actuator is an individual actuator or the reference actuator of a group, this feedback telegram must be linked to the controlling room (zone) user function in order to obtain information about the state of the actuator.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_LightDALI_TC EXTENDS FB_BA_View
VAR_INPUT
  bEn                          : BOOL;
  stLightingCmd                : ST_BA_Lighting;
  bInitialize                  : BOOL;
END_VAR
VAR_OUTPUT
  bInitializing                : BOOL;
  bErr                         : BOOL;
  fActualLightValue            : REAL;
  fActualLightTemperature      : REAL;
  eActualLightPrio             : BYTE;
  stFeedback                   : ST_BA_LightActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  eAdrType                     : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                         : BYTE;
  nAdrRefDev                   : BYTE;
  ipDALICommunication          : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_INPUT CONSTANT
  LightActr                    : FB_BA_LightActrDALI_Base;
  ActualLightValue             : FB_BA_AV_Op;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enabling the function block: A TRUE signal activates the function.
stLightingCmd	ST_BA_Lighting [▶ 272]	Command telegram from the higher-level zone (room).
bInitialize	BOOL	A positive edge at this input starts the DALI initialization routine, which transfers the set parameters to the control gears (see FB_BA_LightActrDALI_Base [▶ 874]).

Outputs

Name	Type	Description
bInitializing	BOOL	The light actuator or the actuators are in the DALI initialization phase, i.e. the entered parameters are transferred to the controlled control gears.
bErr	BOOL	Error output. A plain text is output in TwinCAT in the error list in the output window.
fActualLightValue	REAL	Current light value in percent.
fActualLightTemperature	REAL	Current color temperature in Kelvin.
eActualLightPrio	E_BA_LightingPrio [▶ 268]	Current priority of the command telegram *stLightingCmd*.
stFeedback	ST_BA_LightActuatorFeedback [▶ 702]	Feedback telegram for linking to the controlling room (zone) user function. In this way, information about the state of the light actuator is fed back into the application function.

Inputs CONSTANT PERSISTENT

Name	Type	Description
eAdrType	Tc3_DALI.E_DALIAddressType	Selection of the control type: individual, group or broadcast control, pre-set to individual control.
nAdr	BYTE	DALI address of the actuator or group.
nAdrRefDev	BYTE	DALI address of a reference device if group or broadcast control is selected. This reference device then represents all other controlled devices.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface pointer to the DALI communication block.

Inputs CONSTANT

Name	Type	Description
LightActr	FB_BA_LightActrDALI_Base [▶ 874]	Light control block without color temperature control.
ActualLightValue	FB_BA_AV_Op [▶ 202]	Function block for displaying the percentage light value in BACnet.

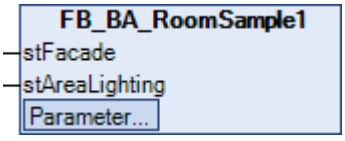
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.5 RoomSamples

Possible room templates.

6.1.4.2.1.4.5.1 FB_BA_RoomSample1



This template represents a simple room with a heating/cooling controller, a sun protection zone function with two blinds and a sun protection zone function with two DALI lights.

The room is also controlled and monitored by a DALI occupancy sensor, which also provides the room brightness.

DALI control

As this room template is an example with DALI components, it requires a fast task with a DALI communication block.

The data exchange with the DALI devices then takes place via an interface and is predefined in FB_Init, here using the example of the brightness and occupancy sensor:

```
BrtnsPrcSns.iParent               := THIS^;
BrtnsPrcSns.iLabel                := LblCtl_BrightnessPresenceSensor;
BrtnsPrcSns.eDPADMode             := E_BA_DPADMode.eInclude;
BrtnsPrcSns.nAdr                  := 11;
BrtnsPrcSns.nInstancePresence     := 1;
BrtnsPrcSns.nInstanceBrightness   := 0;
BrtnsPrcSns.nResBrtns             := 10;
BrtnsPrcSns.ipDALICommunication   := DALICommunication.fbKL6821Communication;
```

In the last line, the DALI communication block is assigned to the communication interface pointer of the brightness and occupancy sensor.

This sample implements the function block *fbKL6821Communication* in the program *DALICommunication*:

```
PROGRAM DALICommunication
VAR

    stKL6821InData          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData         AT %Q*  : Tc3_DALI.ST_KL6821OutData;

    fbKL6821Communication           : Tc3_DALI.FB_KL6821Communication;

    bError                          : BOOL;
    bResetMaximumDemandCounter      : BOOL;
    bResetOverflowCounter           : BOOL;
    nOptions                        : DWORD;
```

```

fbKL6821Communication(  bError                          => bError,
                        bResetMaximumDemandCounter      := bResetMaximumDemandCounter,
                        bResetOverflowCounter           := bResetOverflowCounter,
                        nOptions                        := nOptions,
                        bBusy                           => bBusy,
                        nBufferDemandMeter              => nBufferDemandMeter,
                        nBufferMaximumDemandMeter       => nBufferMaximumDemandMeter,
                        nBufferOverflowCounter          => nBufferOverflowCounter,
                        bResetInactiveProcessImage      := bResetInactiveProcessImage,
                        bInitialise                     := bInitialise,
                        bDigitalInput1Active            => bDigitalInput1Active,
                        bDigitalInput2Active            => bDigitalInput2Active,
                        bProcessImageInactive           => bProcessImageInactive,
                        bCollisionError                 => bCollisionError,
                        bPowerSupplyError               => bPowerSupplyError,
                        bShortCircuit                   => bShortCircuit,
                        bInitialising                   => bInitialising,
                        nTerminalDescription            => nTerminalDescription,
                        nFirmwareVersion                => nFirmwareVersion,
                        stInData                        := stKL6821InData,
                        stOutData                       := stKL6821OutData);
```

The program *DALICommunication* must be called in a fast task - instructions can be found in the chapter: Fast task for serial communication [▶ 54].

Brightness and occupancy sensors

The function block FB_BA_BrightnessPresenceDALI [▶ 884] represents DALI devices that can read both brightness and presence. The function block instructions describe that the presence is output event-driven, but the brightness is read in intervals of the adjustable duration *nPrdQueryBrtns* [s].

In this sample, a short duration *nPollingIntervalFast* and a long duration *nPollingIntervalNormal* are specified: the short duration is intended for the brightness adaptation of a constant light regulation if it receives a readjustment order (*LightZoneConstant.bAdjusting*), the long duration is then used in normal operation to avoid further loading of the DALI bus.

Heating and cooling controller

The core of the heating and cooling function is the function block FB_BA_HeatCool_PID [▶ 891].

The global variable list Site [▶ 1116] informs the function block whether heating or cooling mode is active (*eHeatCoolMedium*), which temperatures apply for which energy level (*stBuildingSpRmT*) and which energy level is currently valid building-wide (*eBuildingEnergyLevel*).

In general, the full heating and cooling output in the *Comfort* energy level should only be available when people are present. This is reported to the function block via the room's occupancy sensor at the *bPrc* input.

Two window contacts *WdwCon1* and *WdwCon2* monitor the heating/cooling. These contacts are designed to supply a TRUE signal when the window is closed. If one of the windows is opened, the function block FB_BA_HeatCool_PID [▶ 891] receives a TRUE signal at the *bWdwCon* input and switches internally to the *Protection* energy level so as not to control unnecessarily against the outside temperature.

Like the window contacts, the *DewPointSensor* is designed according to the quiescent current principle. In the critical case, it will output a FALSE signal, which is negated and sent as TRUE to the *bDewPnt* input of the FB_BA_HeatCool_PID [▶ 891]. This immediately ends and locks the cooling mode.

An analog sensor *RmTSen* provides the room temperature to the control system; the room temperature setpoint can be changed within small limits via a setpoint generator *RmTAdj*. The pre-set values for *fRmTAdjMin* and *fRmTAdjMax* in FB_Init are -5K and 5K.

The room temperature adjustment is only effective if the *Comfort* or *Precomfort* energy level is present, see FB_BA_RmTAdj [▶ 302].

Blind control

In this sample, a zone function block FB_BA_SunblindZone is predefined for the blind controller, which controls 2 blind actuator function blocks FB_BA_SunBld [▶ 970] in parallel.

The zone function block bundles the entire generation of automatic commands, manual control and the link to a visualization (HMI).

The blind actuator function blocks only contain the travel profiles and the link to the hardware. Each blind or blind group that is to be controlled individually therefore requires a zone function block.

The zone function block receives data such as storm protection and anti-icing or sun protection and thermal automatic via the *stFacade* input. It is thus assigned to a specific facade. If it were a corner room with blinds of different orientations, a further zone function block would have to be added.

Via the occupancy sensor input *bPrcDetc*, to which the brightness and occupancy sensor is linked, the function block decides whether the sun protection can be active (occupancy) or whether the thermal automatic can be active (non-occupancy).

The thermal automatic also needs to know the room temperature and the building's current setpoint in order to use the sun's radiant heat or protect against it with the help of the blinds. Sun protection and thermal automatic can also generally be deselected within the FB_BA_SunblindZone.

The inputs *bSunbldSwiUp* and *bSunbldSwiDwn* are declared here as simple variables and it is up to the user to link them to his application.

To assess the status of the blind group and to display it in a visualization, a blind actuator must be selected as a reference. Its output *stFeedback* must be connected to the input *stReferenceFeedback* of the zone function block. In the case of a single blind, the blind itself is the reference actuator.

Light control

Similar to the blinds, the core of the lighting system is a zone function block that combines the generation of higher-level commands (building, floor, area) with local manual operation and the control and display of a possible visualization. The actuator function blocks merely represent the control of the lamps, which can be directly via a bus terminal or via a protocol (DALI).

Here, too, each actuator to be controlled individually requires a zone function block. A reference actuator whose output *stFeedback* must be linked to the input *stReferenceFeedback* of the zone function block must also be selected to assess the status of a group. In the case of a single lamp, this itself is the reference actuator.

In this sample, the light zone is a constant light regulation FB_BA_LightGroupConstant [▶ 894].

If fully automatic is selected, the FB_BA_BrightnessPresenceDALI [▶ 884] brightness and presence function block can activate constant light regulation via the *bPrcDetc* input. If presence is no longer detected, the lighting controllers are reset via the *bResetRoomFunction* input. The automatic function, or its local manual override, is then no longer active and light telegrams of lower priority from the building, floor or area arriving at the *stAreaLighting* input, can take over the lighting controllers.

The *fBrightness* input receives the measured luminous intensity of the brightness sensor for light control. The *bLightSwi* input is declared here as a simple variable and it is up to the user to link it to his application.

The initialization of the template takes place within the method FB_Init.

Block diagram

The https://infosys.beckhoff.com/content/1033/TF8040_TC3_BuildingAutomation/Resources/18447665419.zip can be downloaded if required.

Syntax

```
FUNCTION_BLOCK FB_BA_RoomSample1 EXTENDS FB_BA_View
VAR_INPUT
  stFacade                : ST_BA_Facade;
  stAreaLighting          : ST_BA_Lighting;
END_VAR

VAR_INPUT CONSTANT PERSISTENT
  fRmTAdjMin              : REAL;
  fRmTAdjMax              : REAL;
  fLgtT                   : REAL;
  nPollingIntervalFast    : UDINT;
  nPollingIntervalNormal  : UDINT;
END_VAR

VAR_INPUT CONSTANT
  BrtnsPrcSns             : FB_BA_BrightnessPresenceDALI;
  Ctrl                    : FB_BA_HeatCool_PID;
  VlvHtg                  : FB_BA_ActuatorAnalog;
  VlvCol                  : FB_BA_ActuatorAnalog;
  WdwCon1                 : FB_BA_SensorBinary;
  WdwCon2                 : FB_BA_SensorBinary;
  DewPointSensor          : FB_BA_SensorBinary;
  RmTSen                  : FB_BA_SensorAnalog;
  RmTAdj                  : FB_BA_SensorAnalog;
  SunBldZone              : FB_BA_SunblindZone;
  SunBld1                 : FB_BA_SunBld;
  SunBld2                 : FB_BA_SunBld;
  LightZoneConstant       : FB_BA_LightGroupConstant;
  Light1                  : FB_BA_LightDALI_TC;
  Light2                  : FB_BA_LightDALI_TC;
END_VAR
```

```
VAR
  bSunbldSwiUp            : BOOL;
  bSunbldSwiDwn           : BOOL;
  bLightSwi               : BOOL;
  RmTAvg                  : FB_BA_EnAvrg02;
  bInitializeSensor       : BOOL;
END_VAR
```

Inputs

Name	Type	Description
stFacade	ST_BA_Facade [▶ 704]	Facade-specific blind data and telegrams.
stAreaLighting	ST_BA_Lighting [▶ 272]	Resulting light telegram from the building, the floor and finally the area (prioritization).

Inputs CONSTANT PERSISTENT

Name	Type	Description
fRmTAdjMin	REAL	Minimum value of the temperature setpoint shift, predefined to -5K.
fRmTAdjMax	REAL	Minimum value of the temperature setpoint shift, predefined to 5K.
fLgtT	REAL	Fixed light temperature of the constant light regulation, predefined at 3000K.
nPollingIntervalFast	UDINT	Fast polling interval of the brightness sensor, predefined to 1 s.
nPollingIntervalNormal	UDINT	Slow polling interval of the brightness sensor, predefined to 15 s.

Inputs CONSTANT

Name	Type	Description
BrtnsPrcSns	FB_BA_BrightnessPresenceDALI [▶ 884]	Reading and evaluation function block for a DALI brightness and occupancy sensor.
Ctrl	FB_BA_HeatCool_PID [▶ 891]	Heating-cooling control block.
VlvHtg	FB_BA_ActuatorAnalog [▶ 982]	Analog output object heating valve.
VlvCol	FB_BA_ActuatorAnalog [▶ 982]	Analog output object cooling valve.
WdwCon1	FB_BA_SensorBinary [▶ 1088]	Binary input object window contact.
WdwCon2	FB_BA_SensorBinary [▶ 1088]	Binary input object window contact.
DewPointSensor	FB_BA_SensorBinary [▶ 1088]	Binary input object dew point sensor.
RmTSen	FB_BA_SensorAnalog [▶ 1087]	Analog input object room temperature sensor.
RmTAdj	FB_BA_SensorAnalog [▶ 1087]	Analog input object setpoint adjustment.
SunBldZone	FB_BA_SunblindZone [▶ 964]	Function block blind-specific zone functions.
SunBld1	FB_BA_SunBld [▶ 970]	Actuator function block for a blind.
SunBld2	FB_BA_SunBld [▶ 970]	Actuator function block for a blind.
LightZoneConstant	FB_BA_LightGroupConstant [▶ 894]	Light zone function block of a constant light regulation.
Light1	FB_BA_LightDALI_TC [▶ 929]	Actuator function block for a DALI light.
Light2	FB_BA_LightDALI_TC [▶ 929]	Actuator function block for a DALI light.

VAR

Name	Type	Description
bSunbldSwiUp	BOOL	Variable for the "Blinds manually open" button assignment.
bSunbldSwiDwn	BOOL	Variable for the "Blinds manually down" button assignment.
bLightSwi	BOOL	Variable for "Switch light manually" button assignment.
bInitializeSensor	BOOL	Variable for reinitializing the DALI brightness and occupancy sensor.

Prerequisites

Development environment	Necessary function
TwinCAT from v3.1.4024.64	TF8040 | TwinCAT Building Automation from V5.10.1.0

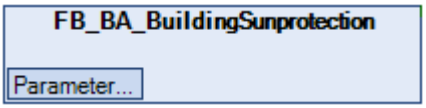
6.1.4.2.1.4.6 SunProtection

Templates for creating a custom sun protection.

6.1.4.2.1.4.6.1 Functions

6.1.4.2.1.4.6.1.1 Building

6.1.4.2.1.4.6.1.1.1 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 [▶ 940]. These are in detail:

- **resulting telegram building alarms**
- **Protection telegram Fire**
  The template FB_BA_BuildingAlarms [▶ 849] supplies the information "Fire alarm" via a globally declared structure variable stBuildingAlarms (see global variable list Site [▶ 1116]). If this alarm is active, the blinds are raised completely via a FB_BA_SunBldEvt [▶ 378].
- **Protection telegram Burglar**
  The template FB_BA_BuildingAlarms [▶ 849] supplies the information "burglar alarm" via a globally declared structure variable stBuildingAlarms (see global variable list Site [▶ 1116]). If this alarm is active, the blinds are raised completely via a FB_BA_SunBldEvt [▶ 378]. 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 pre-set 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 [▶ 379], 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 four 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 lamellas 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 [▶ 390] 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 [▶ 1116]).

The initialization of the template takes place within the method FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_BuildingSunprotection EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  FireAlert                   : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eFire);
  Burglary                    : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eBurglary);
  IceProtection               : FB_BA_IceProtection;
  GlobalThAutoRlse            : FB_BA_Hys_03;
  GlobalTwiLgtAutoRlse        : FB_BA_Hys_03;
  SunPrtcPosIntval            : FB_BA_AV_Op_SP;
  GlobalResetManMode          : FB_BA_BV_Op_Val;
END_VAR
VAR
  BuildingAlarms              : FB_BA_SunBldTgmSel4;
  GlobalTwiLgtAutoValOnOff    : FB_BA_Swi2P;
  rtManResetEnergLvl          : R_TRIG;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
FireAlert	FB_BA_SunBldEvt [▶ 378]	Telegram block for the fire alarm: lets the blind raise completely.
Burglary	FB_BA_SunBldEvt [▶ 378]	Telegram block for burglary: lets the blind raise completely.
IceProtection	FB_BA_IceProtection [▶ 939]	Sub-template for presentation of the ice protection.
GlobalThAutoRlse	FB_BA_Hys_03 [▶ 1008]	Hysteresis template for the presentation of the global automatic thermal release due to global radiation. Contains limit values, delays and a binary object for display.
GlobalTwiLgtAutoRlse	FB_BA_Hys_03 [▶ 1008]	Hysteresis template for the presentation of the global automatic twilight release based on the average brightness. Contains limit values, delays and a binary object for display.
SunPrtcPosIntval	FB_BA_AV_Op_SP [▶ 1029]	Analog input object: Readjustment interval of the lamella angle [min].
GlobalResetManMode	FB_BA_BV_Op_Val [▶ 1031]	Binary input object: Operating option for resetting manual functions.

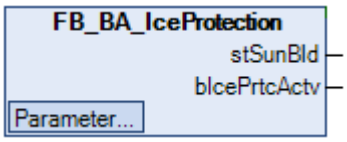
Variables

Name	Type	Description
BuildingAlarms	FB_BA_SunBldTgmSel4 [▶ 390]	Priority selection block.
GlobalTwiLgtAutoValOnOff	FB_BA_Swi2P	Hysteresis block for switching 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

6.1.4.2.1.4.6.1.1.2 FB_BA_IceProtection



Sub-template ice protection.

The function block FB_BA_IceProtection implements the icing protection. If the temperature falls below the freezing point entered on the *SPIce* object and the weather station transmits precipitation via the variable list Site [▶ 1116] (*Site.stWeatherStation.bRain*), the icing condition is given and the blinds are raised completely.

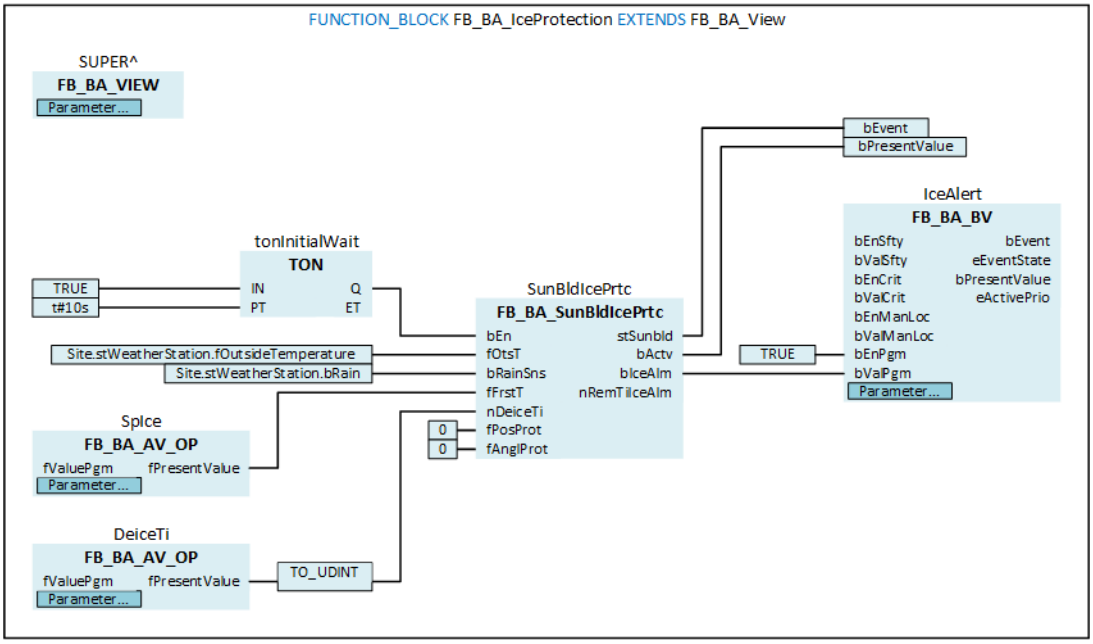
The alarm is only canceled when the outside temperature exceeds the icing temperature again for a deicing time (in seconds) set on the *DeiceTi* object.

The telegram is then no longer active.

To ensure that current temperature values are available at the start of the program, the activation of the icing alarm function block is initially delayed by 10 s.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_IceProtection EXTENDS FB_BA_View
VAR_OUTPUT
  stSunBld          : ST_BA_SunBld;
  bIcePrtcActv      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SpIce             : FB_BA_AV_Op;
  DeiceTi           : FB_BA_AV_Op;
  IceAlert          : FB_BA_BV;
  SunBldIcePrtc     : FB_BA_SunBldIcePrtc;
END_VAR
VAR
  tonInitialWait    : TON;
END_VAR
```

Outputs

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Ice protection telegram.
bIcePrtcActv	BOOL	Control output "Ice protection active".

Inputs CONSTANT

Name	Type	Description
SpIce	FB_BA_AV_Op [▶ 202]	Input limit value for icing [°C].
DeiceTi	FB_BA_AV_Op [▶ 202]	Enter de-icing time [s].
IceAlert	FB_BA_BV [▶ 213]	Display object Status ice protection.
SunBldIcePrtc	FB_BA_SunBldIcePrtc [▶ 379]	Ice protection for the lamellas.

Variables

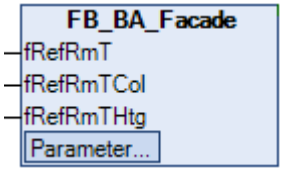
Name	Type	Description
tonInitialWait	TON	Initial function block enable delay.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.6.1.2 Facade

6.1.4.2.1.4.6.1.2.1 FB_BA_Facade



This template compiles the sun protection telegrams that are valid for an entire facade.

The telegrams are largely formed in sub-templates in the main part, then passed to a telegram selector FB_BA_SunBldTgmSel8 [▶ 391] and supplemented by the resulting alarm telegram from the building data (fire, burglary or icing).

The passed telegram from the selector is placed on the output structure stFacade together with the enables from the thermal and twilight automatic and the current sun protection data - the priority of the telegram is displayed in the sub-template FacadeInformation [▶ 946].

Telegrams

The following telegrams are available at the telegram selector [▶ 391] *SunBldTgmResult*, sorted by input:

- **Protection telegram communication error (CommError)**
  The global variable list Site [▶ 1116] 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 (**WindProtection [▶ 957]**)**
- **Positioning telegram Maintenance (**Maintenance [▶ 947]**)**

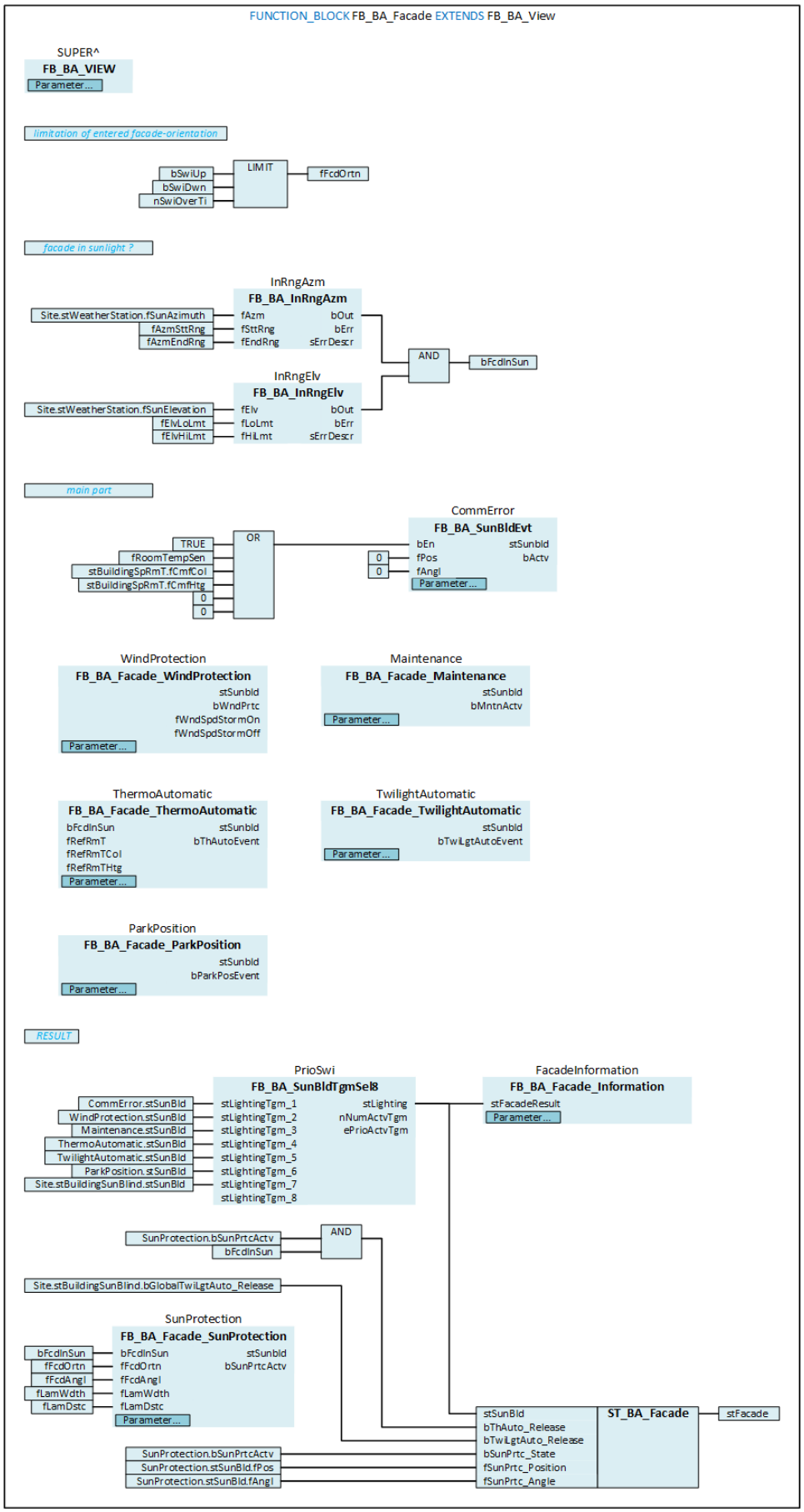
- **Positioning telegram for facade thermal automatic (**ThermoAutomatic [▶ 953]**)**
- **Positioning telegram for twilight automatic (**TwilightAutomatic [▶ 956]**)**
- **Positioning telegram park position (**ParkPosition [▶ 948]**)**
- **Alarm telegram fire/burglary/icing**
  This telegram (*Site.stBuildingSunBlind.stSunBld*) is usually created on the building controller in the template FB_BA_BuildingSunprotection [▶ 937] and placed on the Site variable list [▶ 1116]. It contains the building-wide telegrams for icing, burglary and fire.

Sun protection calculation

Sun protection is calculated separately for each facade. This is done in the sub-template SunProtection [▶ 950]. The calculated values are placed on the output structure stFacade [▶ 704] together with the enables from the thermal and twilight automatic.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade EXTENDS FB_BA_View
VAR_INPUT
  fRefRmT                 : REAL;
  fRefRmTCol              : REAL;
  fRefRmTHtg              : 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;
END_VAR
VAR_INPUT CONSTANT
  CommError               : FB_BA_SunBldEvt :=(ePrio:=E_BA_SunBldPrio.eCommError);
  WindProtection          : FB_BA_Facade_WindProtection;
  Maintenance             : FB_BA_Facade_Maintenance;
  ThermoAutomatic         : FB_BA_Facade_ThermoAutomatic;
  TwilightAutomatic       : FB_BA_Facade_TwilightAutomatic;
  ParkPosition            : FB_BA_Facade_ParkPosition;
 SunProtection            : FB_BA_Facade_SunProtection;
 FacadeInformation        : FB_BA_Facade_Information;
END_VAR
VAR
  InRngAzm                : FB_BA_InRngAzm;
  InRngElv                : FB_BA_InRngElv;
  bFcdInSun               : BOOL;
  SunBldTgmResult         : FB_BA_SunBldTgmSel8;
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 [▶ 704]	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	Lamella width [mm].
fLamDstc	REAL	Lamella 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 *fAzmSttRng* / *fAzmEndRng* 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 *fElvLoLmt* / *fElvHiLmt* the range can be restricted.

Inputs CONSTANT

Name	Type	Description
CommError	FB_BA_SunBldEvt [▶ 378]	Telegram block for the communication error
WindProtection	FB_BA_Facade_WindProtection [▶ 957]	Sub-template storm protection
Maintenance	FB_BA_Facade_Maintenance [▶ 947]	Sub-template maintenance
ThermoAutomatic	FB_BA_Facade_ThermoAutomatic [▶ 953]	Sub-template thermal automatic
TwilightAutomatic	FB_BA_Facade_TwilightAutomatic [▶ 956]	Sub-template twilight automatic
ParkPosition	FB_BA_Facade_ParkPosition [▶ 948]	Sub-template park position
SunProtection	FB_BA_Facade_SunProtection [▶ 950]	Sub-template sun protection
FacadeInformation	FB_BA_Facade_Information [▶ 946]	Sub-template facade information

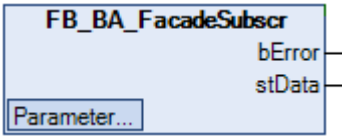
Variables

Name	Type	Description
InRngAzm	FB_BA_InRngAzm [▶ 350]	Sun direction within the defined limits
InRngElv	FB_BA_InRngElv [▶ 352]	Sun elevation within the defined limits
bFcdInSun	BOOL	Facade in the position of the sun
SunBldTgmResult	FB_BA_SunBldTgmSel8 [▶ 391]	Telegram selection block for the resulting blind telegram of the facade

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.6.1.2.2 FB_BA_FacadeSubscr



This template is used to receive the facade data of the type ST_BA_Facade [▶ 704] and is typically used in the floor controllers.

It represents an extension of FB_BA_Subscriber: In case of communication failure, the sunblinds telegram *stSunbld* is overwritten within the facade data with an active telegram position 0%.

Syntax

```
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 [▶ 704]	Read facade telegram, overwritten in case of error.

Variables

Name	Type	Description
_fbHighPrio	FB_BA_SunBldEvt [▶ 378]	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.

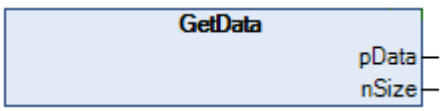
Methods

Name	Description
GetData [▶ 945]	Determines the address and size of the output telegram.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.GetData
4.2.
1.4.
6.1.
2.2.
1



Internally used method that determines the address and size of the output telegram.

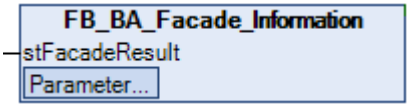
Syntax

```
VAR_OUTPUT
  pData    : PVOID;
  nSize    : DINT;
END_VAR
```

Outputs

Name	Type	Description
pData	pData	Pointer to the subscribed data.
nSize	DINT	Size of the subscribed data.

6.1.4.2.1.4.6.1.2.3 FB_BA_Facade_Information



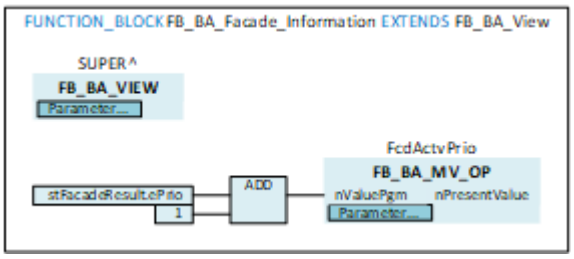
Sub-template facade information.

This template displays the priority of the current sun protection telegram in BACnet.

As the enumeration of the priorities starts with "0", but the texts of the MV object start with "1" (see *FB_Init*), a one must be added to the enumeration.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_Information EXTENDS FB_BA_View
VAR_INPUT
  stFacadeResult    : ST_BA_SunBld;
END_VAR
VAR_INPUT CONSTANT
  FcdActvPrio       : FB_BA_MV_Op;
END_VAR
```

Inputs

Name	Type	Description
stFacadeResult	ST_BA_SunBld [▶ 274]	Resulting telegram from the higher-level facade template [▶ 940].

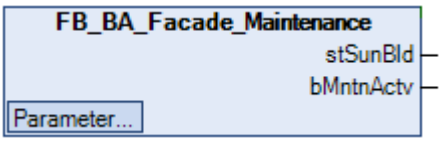
Inputs CONSTANT

Name	Type	Description
FcdActvPrio	FB_BA_MV_Op [▶ 239]	Function block for displaying the current telegram priority in BACnet.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.6.1.2.4 FB_BA_Facade_Maintenance



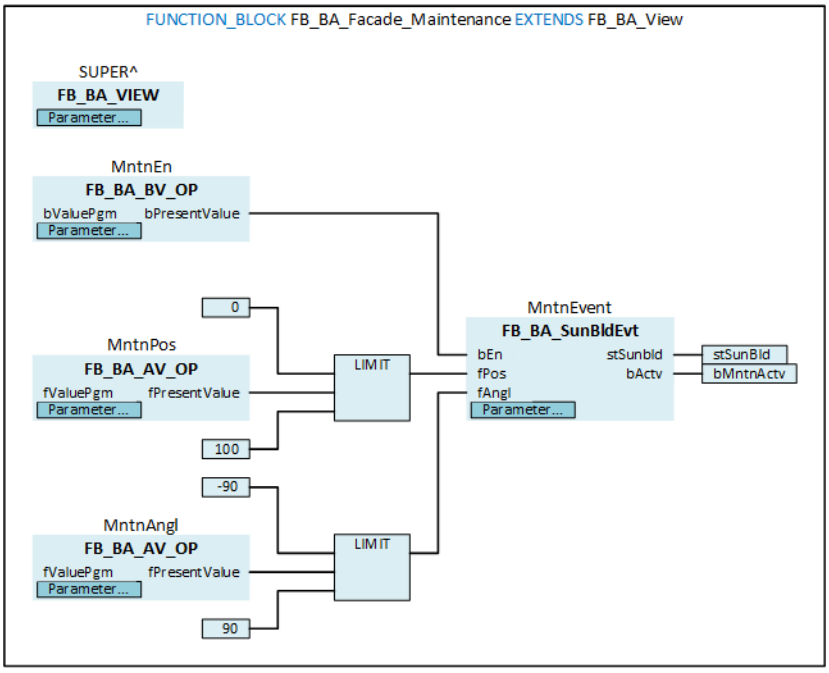
Sub-template Maintenance position.

This function allows you to enter a position and an angle for maintenance work via objects and activate the corresponding sun protection telegram,

which is then available to the higher-level facade template [▶ 940] at the *stSunblind* output.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_Maintenance EXTENDS FB_BA_View
VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  bMntnActv     : BOOL;
END_VAR
VAR_INPUT CONSTANT
  MntnEn        : FB_BA_BV_Op;
  MntnPos       : FB_BA_AV_Op;
  MntnAngl      : FB_BA_AV_Op;
  MntnEvent     : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eMaintenance);
END_VAR
```

Outputs

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Maintenance telegram.
bMntnActv	BOOL	Control output "Maintenance active".

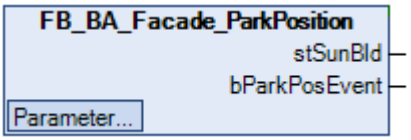
Inputs CONSTANT

Name	Type	Description
MntnEn	FB_BA_BV_Op [▶ 216]	Function block for activating the maintenance telegram.
MntnPos	FB_BA_AV_Op [▶ 202]	Function block for entering the position.
MntnAngl	FB_BA_AV_Op [▶ 202]	Function block for entering the angle.
MntnEvent	FB_BA_SunBldEvt [▶ 378]	Function block for telegram generation.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.6.1.2.5 FB_BA_Facade_ParkPosition



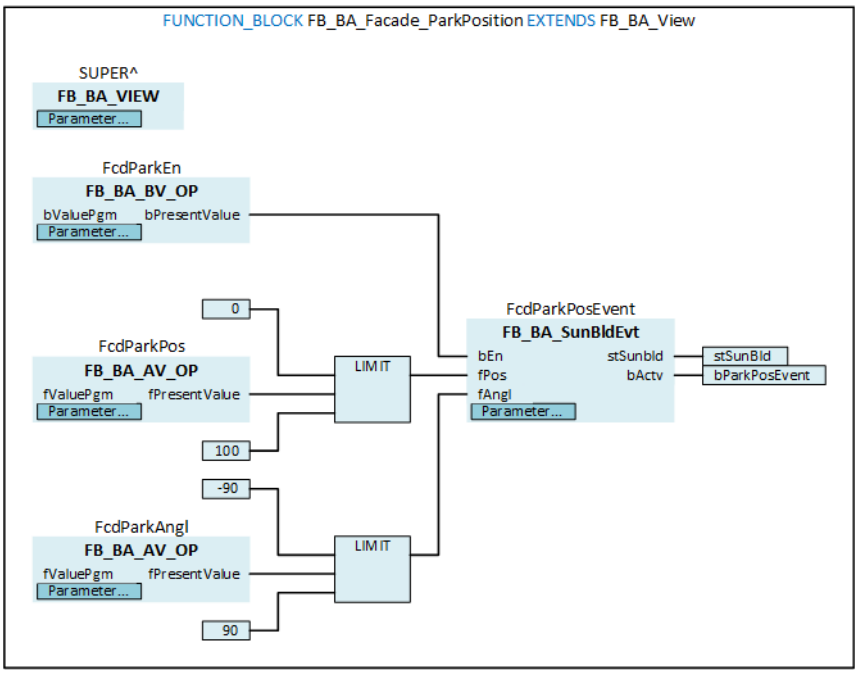
Sub-template Park position.

This function allows you to enter a park position and a park angle and activate the corresponding sun protection telegram,

which is then available to the higher-level facade template [▶ 940] at the *stSunblind* output.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_ParkPosition EXTENDS FB_BA_View
VAR_OUTPUT
  stSunBld           : ST_BA_SunBld;
  bParkPosEvent      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  FcdParkEn          : FB_BA_BV_Op;
  FcdParkPos         : FB_BA_AV_Op;
  FcdParkAngl        : FB_BA_AV_Op;
  FcdParkPosEvent    : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eParkPosition);
END_VAR
```

Outputs

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Parking position telegram.
bParkPosEvent	BOOL	Control output "Parking position enabled".

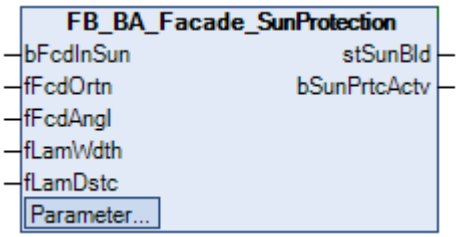
Inputs CONSTANT

Name	Type	Description
FcdParkEn	FB_BA_BV_Op [▶ 216]	Function block for activating the maintenance telegram.
FcdParkPos	FB_BA_AV_Op [▶ 202]	Function block for entering the position.
FcdParkAngl	FB_BA_AV_Op [▶ 202]	Function block for entering the angle.
FcdParkPosEvent	FB_BA_SunBldEvt [▶ 378]	Function block for telegram generation.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.6.1.2.6 FB_BA_Facade_SunProtection



Sub-template Automatic sun protection.

This template calls up a sun protection function with fixed position and lamella setpoint tracing.

In the upper part, the brightness for the facade under consideration is first interpolated using the facade orientation (north = 0°, east = 90°, south = 180°, west = 270°) and the 4 light sensors (*BrtnsCardinal interpolation*).

As the glare effect of the sun also depends on the height of the sun, 7 brightness values are used for the switch-on and switch-off threshold of the sun protection. These are defined as AV objects and describe threshold values at 0°, 15°, 30°, 45°, 60°, 75° and 90° sun elevation.

The switch-on and switch-off thresholds are each interpolated with a function block FB_BA_Chrct07 [▶ 461] based on the current sun elevation.

The calculated threshold values for On and Off are each made available via an AV object in BACnet.

In the subsequent section, a hysteresis block generates the actual enable for the sun protection function.

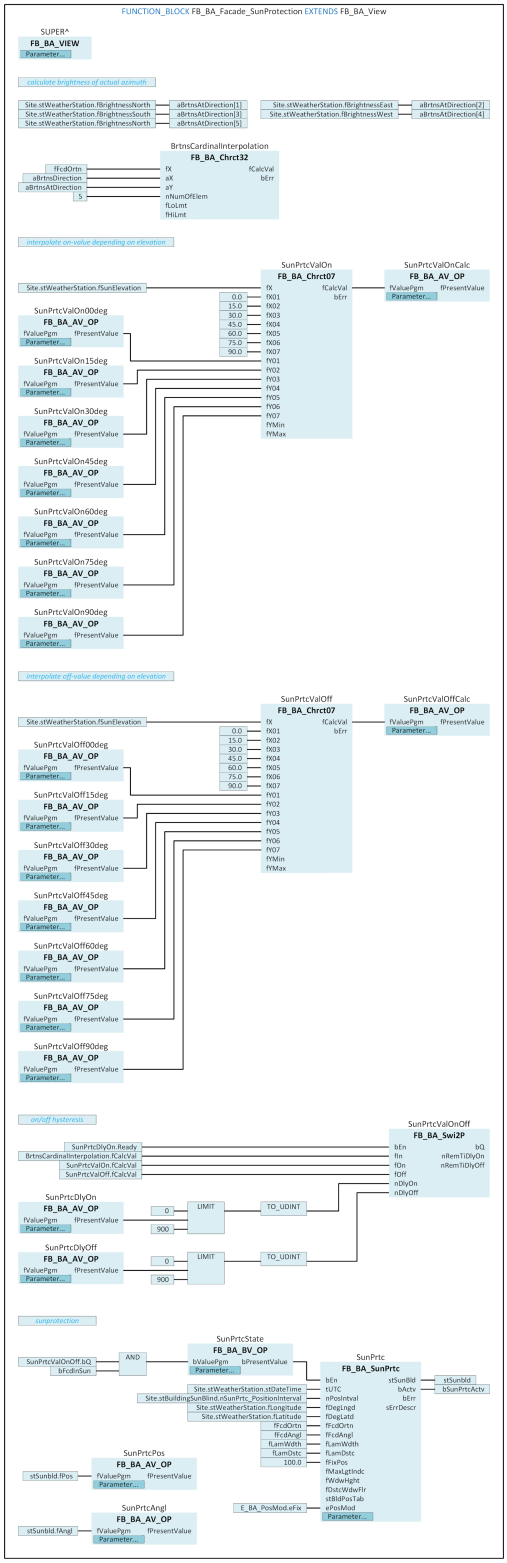
The actual value used for this is the interpolated brightness based on the cardinal points.

In addition, a time delay of the limit values is programmed to take account of a rapid change in brightness.

Enabling sun protection function block is restricted by the criterion "*bFcdInSun*" (facade is in the sun area) and then indicated in BACnet via a BV object. The global data for the sun protection function - date/time, positioning interval, longitude and latitude of the property - comes from the local variable list Site [▶ 1116], while facade orientation, facade inclination, lamella width and lamella spacing come as inputs from the higher-level FB_BA_Facade [▶ 940] as parameters. The determined blind angle and position are indicated as analog objects in BACnet.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_SunProtection EXTENDS FB_BA_View
VAR_INPUT
  bFcdInSun                       : BOOL;
  fFcdOrtn                        : REAL;
  fFcdAngl                        : REAL;
  fLamWdth                        : REAL;
  fLamDstc                        : REAL;
END_VAR
VAR_OUTPUT
  stSunbld                        : ST_BA_SunBld;
  bSunPrtcActv                    : BOOL;
```

```
END_VAR
VAR_INPUT CONSTANT
SunPrtcValOn00deg               : FB_BA_AV_Op;
  SunPrtcValOn15deg             : FB_BA_AV_Op;
  SunPrtcValOn30deg             : FB_BA_AV_Op;
  SunPrtcValOn45deg             : FB_BA_AV_Op;
  SunPrtcValOn60deg             : FB_BA_AV_Op;
  SunPrtcValOn75deg             : FB_BA_AV_Op;
  SunPrtcValOn90deg             : FB_BA_AV_Op;
  SunPrtcValOff00deg            : FB_BA_AV_Op;
  SunPrtcValOff15deg            : FB_BA_AV_Op;
  SunPrtcValOff30deg            : FB_BA_AV_Op;
  SunPrtcValOff45deg            : FB_BA_AV_Op;
  SunPrtcValOff60deg            : FB_BA_AV_Op;
  SunPrtcValOff75deg            : FB_BA_AV_Op;
  SunPrtcValOff90deg            : FB_BA_AV_Op;
  SunPrtcValOnCalc              : FB_BA_AV_Op;
  SunPrtcValOffCalc             : FB_BA_AV_Op;
  SunPrtcDlyOn                  : FB_BA_AV_Op;
  SunPrtcDlyOff                 : FB_BA_AV_Op;
  SunPrtcState                  : FB_BA_BV_Op;
  SunPrtcPos                    : FB_BA_AV_Op;
  SunPrtcAngl                   : FB_BA_AV_Op;
  SunPrtc                       : FB_BA_SunPrtc;
END_VAR
VAR
  aBrtnsDirection               : ARRAY[1..32] OF REAL :=[0.0, 90.0, 180.0, 270.0, 360.0];
  aBrtnsAtDirection             : ARRAY[1..32] OF REAL;
  BrtnsCardinalInterpolation    : FB_BA_Chrct32;
  SunPrtcValOn                  : FB_BA_Chrct07;
  SunPrtcValOff                 : FB_BA_Chrct07;
  SunPrtcValOnOff               : FB_BA_Swi2P;
END_VAR
```

Inputs

Name	Type	Description
bFcdInSun	BOOL	The facade is located in the sun area in relation to the current sun elevation and sun direction. This criterion is generated in the higher-level FB_BA_Facade [▶ 940].
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	Lamella width [mm].
fLamDstc	REAL	Lamella spacing [mm].

Outputs

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Parking position telegram.
bSunPrtcActv	BOOL	Control output "Sun protection active".

Inputs CONSTANT

Name	Type	Description
SunPrtcValOn00deg … SunPrtcValOn90deg	FB_BA_AV_Op [▶ 202]	Brightness thresholds for sun protection On [lx] for elevations of 0°, 15°, 30°, 45°, 60°, 75° and 90°.
SunPrtcValOff00deg … SunPrtcValOff90deg	FB_BA_AV_Op [▶ 202]	Brightness thresholds for sun protection Off [lx] for elevations of 0°, 15°, 30°, 45°, 60°, 75° and 90°.
SunPrtcValOnCalc	FB_BA_AV_Op [▶ 202]	Calculated brightness threshold value for sun protection On [lx] for display in BACnet.
SunPrtcValOffCalc	FB_BA_AV_Op [▶ 202]	Calculated brightness threshold value for sun protection Off [lx] for display in BACnet.
SunPrtcDlyOn	FB_BA_AV_Op [▶ 202]	Delay sun protection on [s].
SunPrtcDlyOff	FB_BA_AV_Op [▶ 202]	Delay sun protection off [s].
SunPrtcState	FB_BA_BV_Op [▶ 216]	Display object state sun protection. 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).
SunPrtcPos	FB_BA_AV_Op [▶ 202]	Calculated sun protection position [%] for display in BACnet.
SunPrtcAngl	FB_BA_AV_Op [▶ 202]	Calculated sun protection angle [°] for display in BACnet.
SunPrtc	FB_BA_SunPrtc [▶ 396]	Calculation block for sun protection.

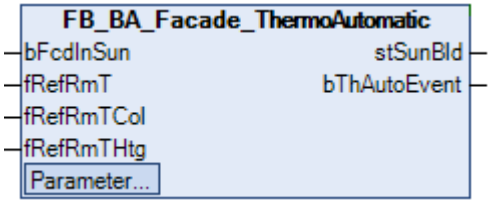
Variables

Name	Type	Description
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.
BrtnsCardinalInterpolation	FB_BA_Chrct32 [▶ 463]	Calculation block for brightness.
SunPrtcValOn	FB_BA_Chrct07 [▶ 461]	Calculation block for the elevation-dependent brightness threshold On.
SunPrtcValOff	FB_BA_Chrct07 [▶ 461]	Calculation block for the elevation-dependent brightness threshold Off.
SunPrtcValOnOff	FB_BA_Swi2P	Hysteresis block for switching on/off the sun protection.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.6.1.2.7 FB_BA_Facade_ThermoAutomatic



Sub-template facade-wide 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 thermal automatic to become active:

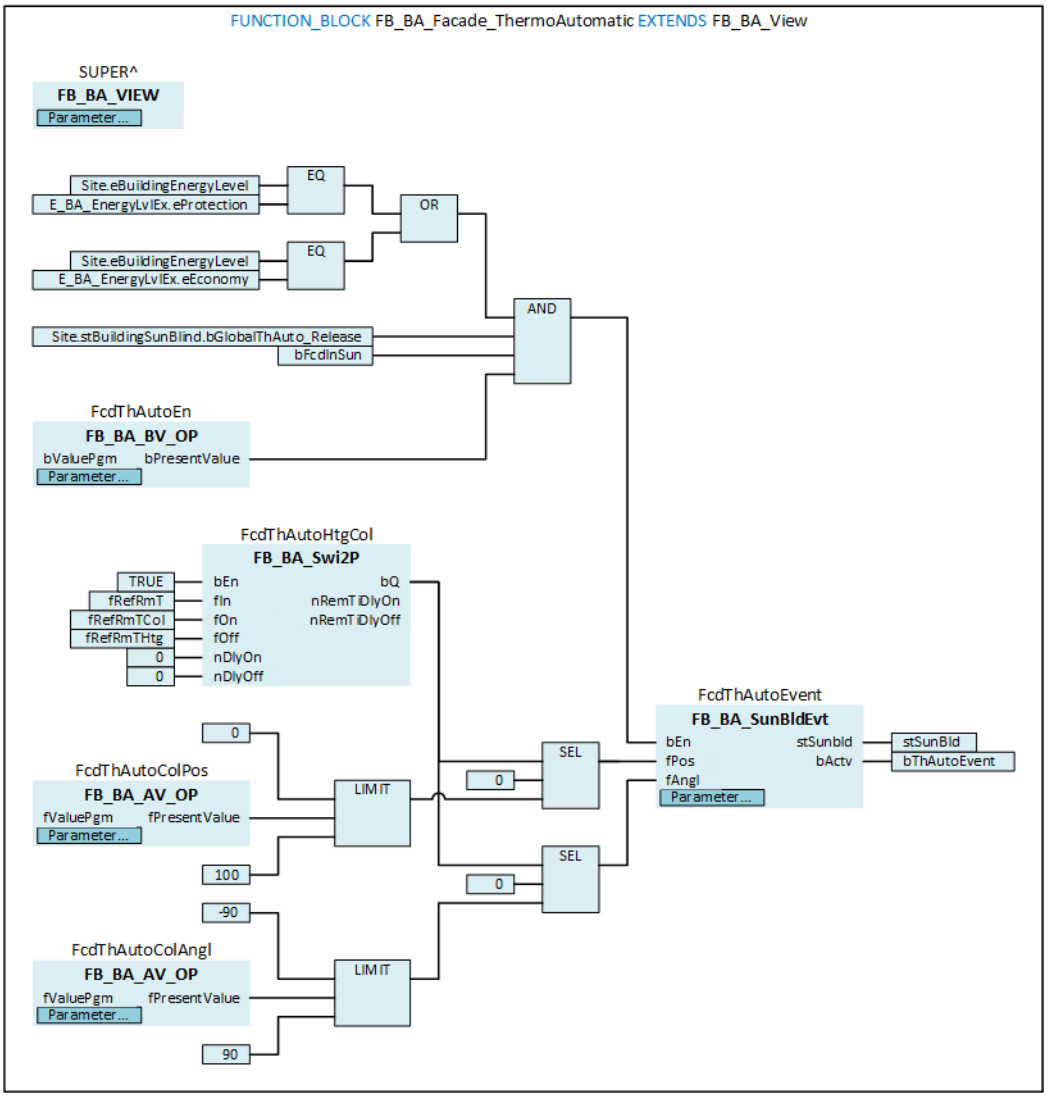
- the building must be unoccupied, this is determined on the basis of the currently valid energy level.
- the global radiation-dependent release from the template FB_BA_BuildingSunprotection [▶ 937] must be given. This criterion is routed to the local variable list Site [▶ 1116] and is read from there (*Site.stBuildingSunBlind.bGlobalThAuto_Release*).
- the facade must be exposed to the sun (*bFcdInSun*).
- the facade-wide thermal automatic is selected via the *FcdThAutoEn* object.

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 *FcdThAutoEn* object.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_ThermoAutomatic EXTENDS FB_BA_View
VAR_INPUT
  bFcdInSun              : BOOL;
  fRefRmT                : REAL;
  fRefRmTCol             : REAL;
  ReffRmTHtg             : REAL;
END_VAR
VAR_OUTPUT
  stSunBld               : ST_BA_SunBld;
  bThAutoEvent           : BOOL;
END_VAR
VAR_INPUT CONSTANT
  FcdThAutoEn            : FB_BA_BV_Op;
  FcdThAutoColPos        : FB_BA_AV_Op;
  FcdThAutoColAngl       : FB_BA_AV_Op;
  FcdThAutoEvent         : FB_BA_SunBldEvt :=(ePrio:=E_BA_SunBldPrio.eFacadeThermoAutomatic);
END_VAR
VAR
  FcdThAutoHtgCol        : FB_BA_Swi2P;
  bFcdThAuto_Enable      : BOOL;
  fFcdThAuto_Position    : REAL;
  fFcdThAuto_Angle       : REAL;
END_VAR
```

Inputs

Name	Type	Description
bFcdInSun	BOOL	The facade is located in the sun area in relation to the current sun elevation and sun direction. This criterion is generated in the higher-level FB_BA_Facade [▶ 940].
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.
RefRmTHtg	REAL	Room temperature setpoint heating of the reference room for the facade-wide thermal automatic.

Outputs

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Thermal automatic telegram.
bThAutoEvent	BOOL	Control output "facade-wide thermal automatic active".

Inputs CONSTANT

Name	Type	Description
FcdThAutoEn	FB_BA_BV_Op [▶ 216]	Facade thermal automatic enable.
FcdThAutoColPos	FB_BA_AV_Op [▶ 202]	Facade thermal automatic cooling position Position [%].
FcdThAutoColAngl	FB_BA_AV_Op [▶ 202]	Facade thermal automatic cooling position Angle [°].
FcdThAutoEvent	FB_BA_SunBldEvt [▶ 378]	Telegram block for facade thermal automatic.

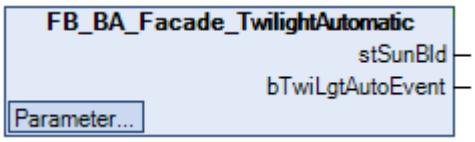
Variables

Name	Type	Description
FcdThAutoHtgCol	FB_BA_Swi2P	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 [°].

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.6.1.2.8 FB_BA_Facade_TwilightAutomatic



Sub-template facade-wide twilight automatic.

This function is designed to be active only when no one is present to prevent unnecessary light from escaping the building.

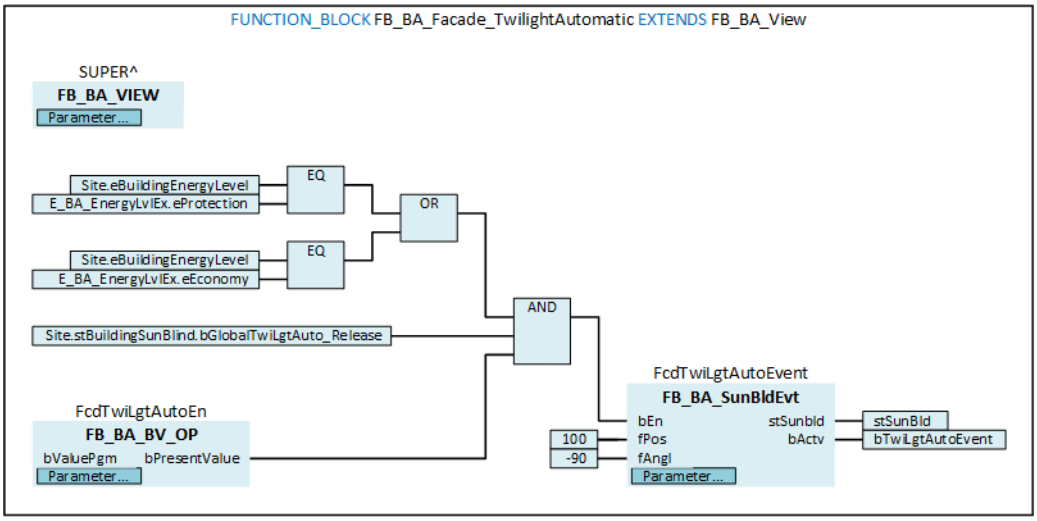
Three conditions must be met for the facade automatic to become active:

- the building must be unoccupied, this is determined on the basis of the currently valid energy level.
- the brightness-dependent release from the template FB_BA_BuildingSunprotection [▶ 937] must be given. This criterion is routed to the local variable list Site [▶ 1116] and is read from there (*Site.stBuildingSunBlind.bGlobalTwiLgtAuto_Release*).
- the facade-wide twilight automatic is selected via the *FcdTwiLgtAutoEn* object.

If it is enabled and selected via the *FcdTwiLgtAutoEn* object, the blinds will be lowered completely in the event of twilight.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_TwilightAutomatic EXTENDS FB_BA_View
VAR_INPUT
END_VAR
VAR_OUTPUT
  stSunBld              : ST_BA_SunBld;
  bTwiLgtAutoEvent      : BOOL;
END_VAR
VAR_INPUT CONSTANT
```

```
  FcdTwiLgtAutoEn       : FB_BA_BV_Op;
  FcdTwiLgtAutoEvent    : FB_BA_SunBldEvt :=(ePrio:=E_BA_SunBldPrio.eFacadeTwilightAutomatic);
END_VAR
```

Outputs

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Twilight automatic telegram.
bTwiLgtAutoEvent	BOOL	Control output "facade-wide twilight automatic active".

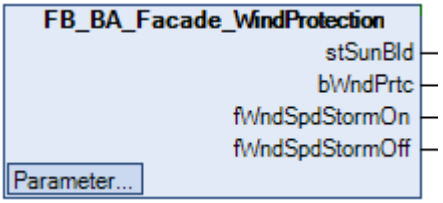
Inputs CONSTANT

Name	Type	Description
FcdTwiLgtAutoEn	FB_BA_BV_Op [▶ 216]	Function block for activating the twilight automatic telegram.
FcdTwiLgtAutoEvent	FB_BA_SunBldEvt [▶ 378]	Function block for telegram generation.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.6.1.2.9 FB_BA_Facade_WindProtection



Sub-template Storm protection.

The data for storm protection - wind speed and wind direction - is first transferred from the weather station template [▶ 855] to the local variable list Site [▶ 1116] together with other weather data. From there, they are available as

- *Site.stWeatherStation.fWindSpeed*
- *Site.stWeatherStation.fWindDirection*

in this template.

In the upper part of the function, the switch-on and switch-off thresholds for activation are weighted according to the wind direction. The array tables *StormOnInterpolation* for the switch-on threshold and *StormOffInterpolation* for the switch-off threshold provide limit values for different wind directions are available in 30° steps.

The idea behind this is that a direct wind impact on the facade is weighted more heavily (lower limit value) than if the wind comes from the side (higher limit value).

As this template is delivered universally, i.e. for all facade directions, the switch-on threshold for **all** wind directions is predefined at 12 m/s (strong wind) and the switch-off threshold at 8 m/s in *FB_Init*. The values can be refined as required.

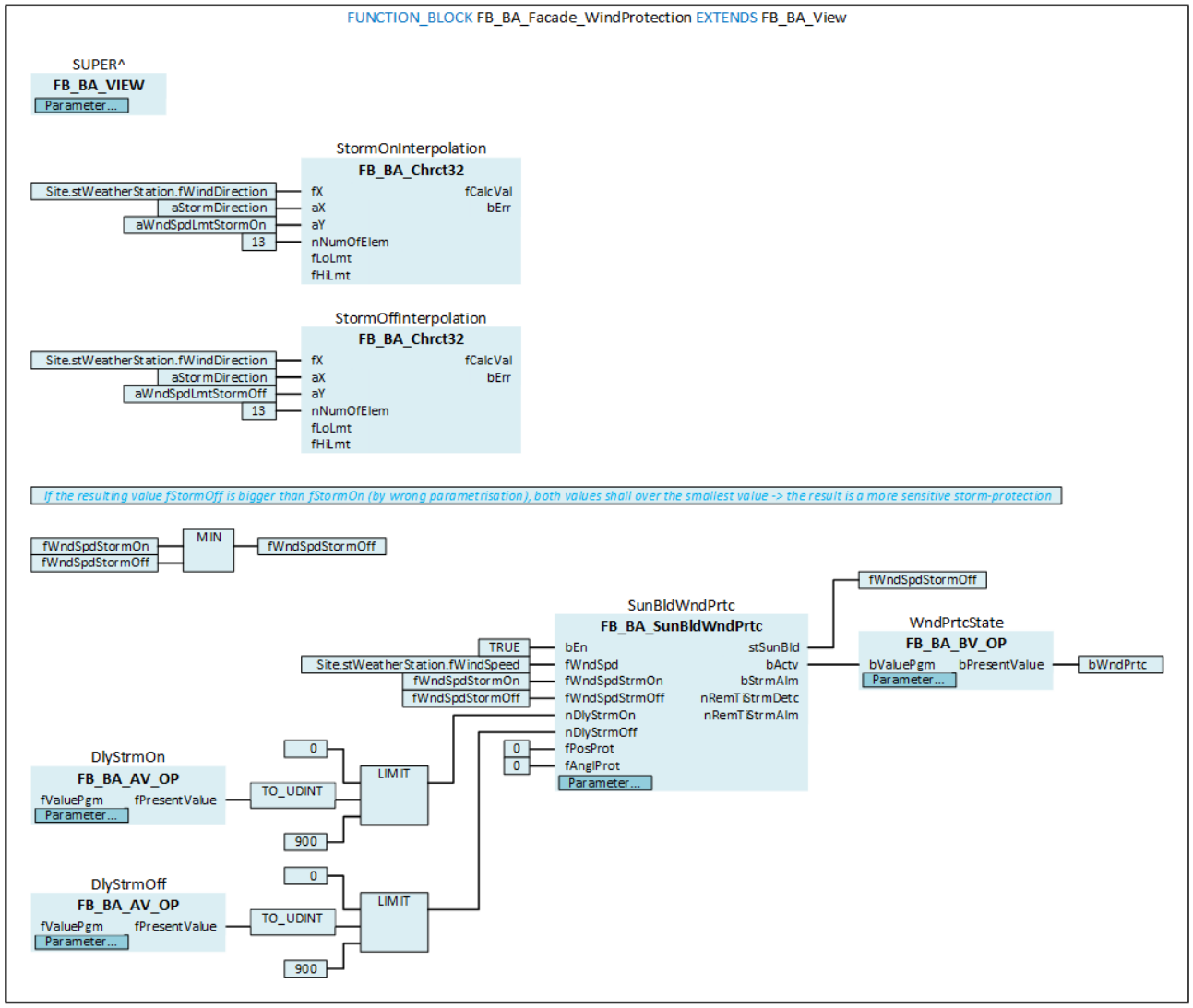
It should also be noted that the limit values are actually unitless and must be selected according to the weather station: if the weather station outputs the wind speed in km/h, the values in the order of 40 km/h for switching on and 28 km/h for switching off are more appropriate.

The function block FB_BA_SunBldWndPrtc [▶ 394] implements storm protection. The current wind speed from the variable list Site [▶ 1116] and the weighted switch-on and switch-off thresholds are sent to it. It also has inputs for a switch-on and switch-off delay, which can be set via the input objects DlyStrmOn [▶ 202] and DlyStrmOff [▶ 202]. Both values are limited to 60 s in this template.

The binary display object WndPrtcState [▶ 216] is used to display the status of the storm protection in BACnet.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_WindProtection EXTENDS FB_BA_View
VAR_OUTPUT
  stSunbld                : ST_BA_SunBld;
  bWndPrtc                : BOOL;
  fWndSpdStormOn          : REAL;
  fWndSpdStormOff         : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  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;
  WndPrtcState             : FB_BA_BV_Op;
  SunBldWndPrtc            : FB_BA_SunBldWndPrtc;
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_Chrct32;
  StormOffInterpolation    : FB_BA_Chrct32;
END_VAR
```

Outputs

Name	Type	Description
stSunBld	ST_BA_SunBld [▶ 274]	Storm protection telegram.
bWndPrtc	BOOL	Control output "Storm protection active".
fWndSpdStormOn	REAL	Calculated value limit value storm detection.
fWndSpdStormOff	REAL	Calculated value limit value Storm abated.

Inputs CONSTANT PERSISTENT

Name	Type	Description
aWndSpdLmtStormOn	ARRAY[1..32] OF REAL	*aWndSpdLmtStormOn[01]* … *aWndSpdLmtStormOn[13]*: These values define the interpolation curve for the direction-dependent switch-on threshold of the storm detection. Here *aWndSpdLmtStormOn[01]* applies to the wind direction at 0° (north). The other field elements represent the interpolation points in 30° steps. The unit of the values selected in *FB_Init* is m/s. If the weather station provides the wind speed in a different unit, the values must be selected accordingly. Only 13 values are described for the interpolation of the wind direction. The corresponding function block is parameterized accordingly.
aWndSpdLmtStormOff	ARRAY[1..32] OF REAL	*aWndSpdLmtStormOff[01]* .. *aWndSpdLmtStormOff[13]*: These values define the interpolation curve for the direction-dependent switch-off threshold of the storm detection. Here *aWndSpdLmtStormOff[01]* applies to the wind direction at 0° (north). The other field elements represent the interpolation points in 30° steps. The unit of the values selected in *FB_Init* is m/s. If the weather station provides the wind speed in a different unit, the values must be selected accordingly. Only 13 values are described for the interpolation of the wind direction. The corresponding function block is parameterized accordingly.

Inputs CONSTANT

Name	Type	Description
DlyStrmOn	FB_BA_AV_Op [▶ 202]	Delay value for storm detection [s].
DlyStrmOff	FB_BA_AV_Op [▶ 202]	Delay value storm abated [s].
WndPrtcState	FB_BA_BV_Op [▶ 216]	Display object state storm protection.
SunBldWndPrtc	FB_BA_SunBldWndPrtc [▶ 394]	Triggering and telegram block for storm protection.

Variables

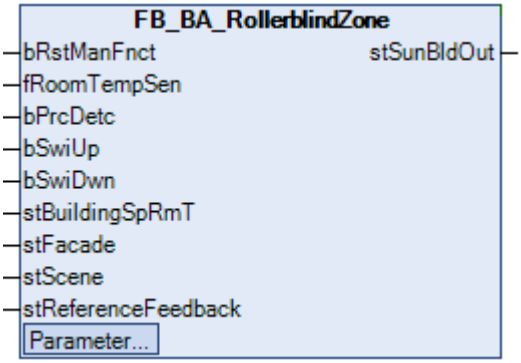
Name	Type	Description
aStormDirection	ARRAY[1..32] OF REAL	Pre-set field of cardinal points for weighting the storm limits.
StormOnInterpolation	FB_BA_Chrct32 [▶ 463]	Calculation block for the limit value storm detection.
StormOffInterpolation	FB_BA_Chrct32 [▶ 463]	Calculation block for the limit value Storm abated.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.6.1.3 Zone

6.1.4.2.1.4.6.1.3.1 FB_BA_RollerblindZone



In a blind zone, one or more blinds are combined for simultaneous control, depending on the local conditions. The template 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 switchover time to latching, *nSwiOverTi* [ms], is pre-parameterized to 250 ms. The manual function is deleted via the input *bRstManFnct* (inherited from the base class *FB_BA_Ext_SunblindPosition* (internal function block)).
- **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 [▶ 940] 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 is 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*).

Input stReferenceFeedback

Information about the controlled blind actuator or the reference actuator of a group is fed back into the blind control function via this input.

These are the position details and status of the reference actuator.

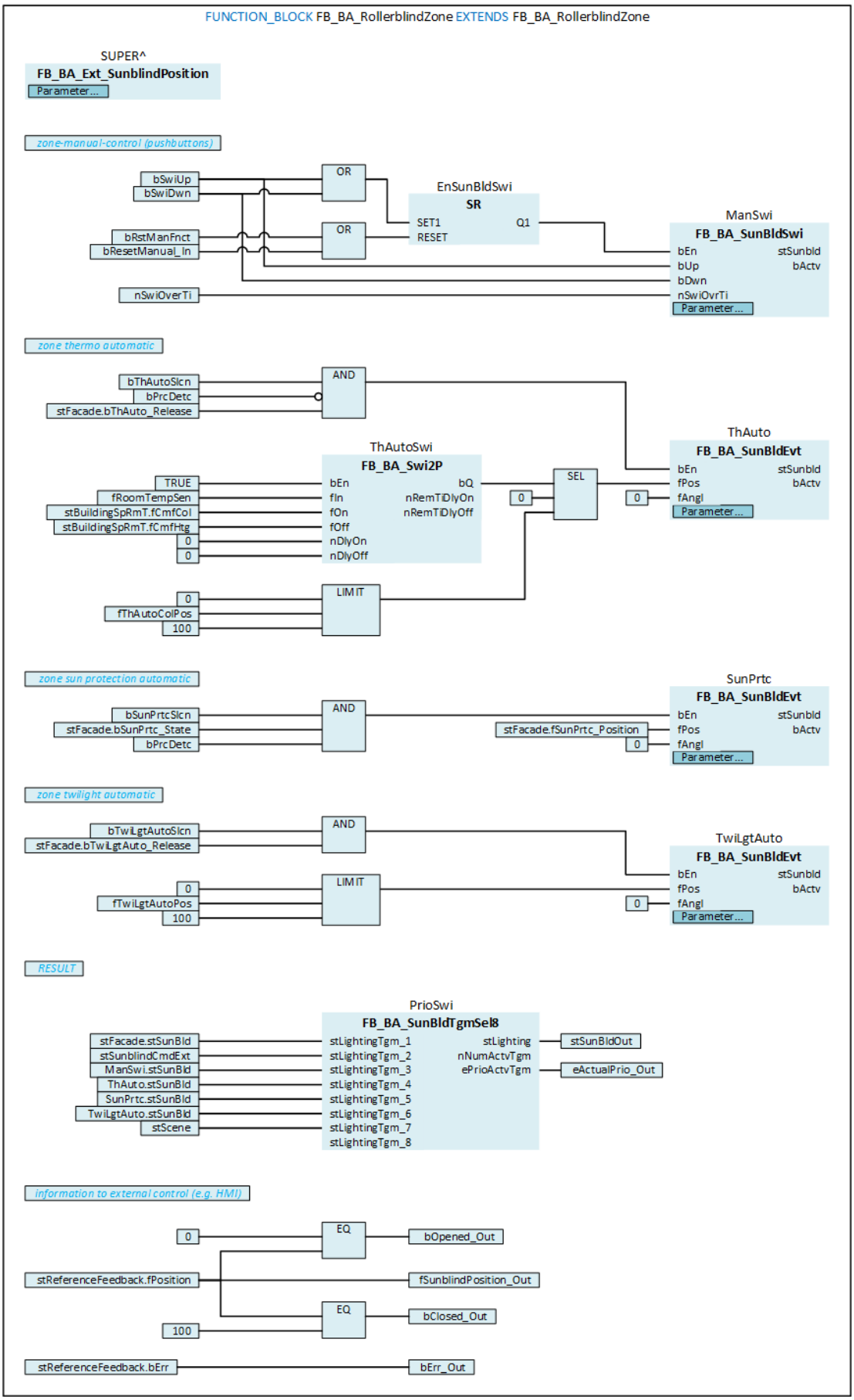
Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_Ext_SunblindPosition* (internal function block). The use of the following variables is visible in this template:

- **bResetManual_In**: Command from the HMI to delete the manual function.
- **fSunblindPosition_Out**: Output information position to the HMI.
- **bOpened_Out**: Output information "Blind completely open" to the HMI.
- **bClosed_Out**: Output information "Blind completely closed" to the HMI.
- **bErr_Out**: Output information "Reference actuator faulty" to the HMI.
- **eActualPrio_Out**: Currently controlling telegram priority to the HMI.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_RollerblindZone EXTENDS FB_BA_Ext_SunblindPosition
VAR_INPUT
  fRoomTempSen           : REAL;
  bPrcDetc               : BOOL;
  bSwiUp                 : BOOL;
  bSwiDwn                : BOOL;
  bRstManFnct            : BOOL;
  stBuildingSpRmT        : ST_BA_SpRmT;
  stFacade               : ST_BA_Facade;
  stScene                : ST_BA_Sunbld;
  stReferenceFeedback    : ST_BA_SunblindActuatorFeedback;
END_VAR
VAR_OUTPUT
  stSunBldOut            : ST_BA_SunBld;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bThAutoSlcn            : BOOL;
  fThAutoColPos          : REAL;
  bSunPrtcSlcn           : BOOL;
  bTwiLgtAutoSlcn        : BOOL;
  fTwiLgtAutoPos         : REAL;
  nSwiOverTi             : UDINT;
END_VAR
VAR_INPUT CONSTANT
  ManSwi                 : FB_BA_SunBldSwi := (ePrio:= E_BA_SunBldPrio.eManualGroup);
  ThAuto                 : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupThermoAuto);
  SunPrtc                : FB_BA_SunBldEvt : = (ePrio:= E_BA_SunBldPrio.eSunProtection);
  TwiLgtAuto             : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupTwiLightAuto);
END_VAR
VAR
  ThAutoSwi              : FB_BA_Swi2P;
  PrioSwi                : FB_BA_SunBldTgmSel8;
  EnSunBldSwi            : SR;
END_VAR
```

Inputs

Name	Type	Description
fRoomTempSen	REAL	Room temperature sensor [°C].
bPrcDetc	BOOL	Presence detection.
bSwiUp	BOOL	Key function Blind "up".
bSwiDwn	BOOL	Key function Blind "down".
bRstManFnct	BOOL	Deletion input for the blind manual flag.
stBuildingSpRmT	ST_BA_SpRmT [▶ 272]	Structure of room setpoints (Protection Cooling..Comfort Cooling and Protection Heating ... Comfort Heating).
stFacade	ST_BA_Facade [▶ 704]	Structure of the facade data (facade telegrams, releases for twilight and thermal automatic, etc.)
stScene	ST_BA_Sunbld [▶ 274]	Reserved telegram input for scene control.
stReferenceFeedback	ST_BA_SunblindActuatorFeedback [▶ 701]	Feedback input of the controlled blind actuator or the reference actuator of the controlled group.

Outputs

Name	Type	Description
stSunBldOut	ST_BA_Sunbld [▶ 274]	Resulting zone telegram.

Inputs CONSTANT PERSISTENT

Name	Type	Description
bThAutoSlcn	BOOL	Thermal automatic enable.
fThAutoColPos	REAL	Thermal automatic cooling position [%].
bSunPrtcSlcn	BOOL	Shading automatic enable.
bTwiLgtAutoSlcn	BOOL	Twilight automatic enable.
fTwiLgtAutoPos	REAL	Twilight automatic cooling position [%].
nSwiOverTi	UDINT	Push button function of the zone: Switchover time [ms] from jogging to latching mode.

Inputs CONSTANT

Name	Type	Description
ManSwi	FB_BA_SunBldSwi [▶ 388]	Group (zone) push button block.
ThAuto	FB_BA_SunBldEvt [▶ 378]	Telegram block for group (zone) thermal automatic.
SunPrtc	FB_BA_SunBldEvt [▶ 378]	Telegram block for group (zone) shading automatic.
TwiLgtAuto	FB_BA_SunBldEvt [▶ 378]	Telegram block for group (zone) twilight automatic.

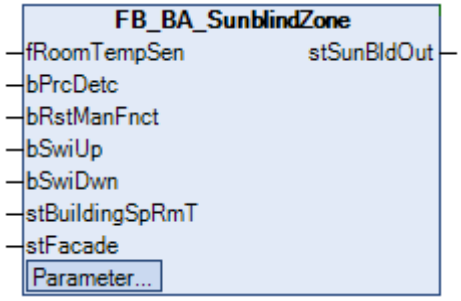
Variables

Name	Type	Description
ThAutoSwi	FB_BA_Swi2P	Hysteresis block for heating/cooling changeover for group (zone) thermal automatic.
PrioSwi	FB_BA_SunBldTgmSel8 [▶ 391]	Selection of the resulting telegram.
EnSunBldSwi	SR	Enable/reset switching.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.6.1.3.2 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 switchover time to latching, *nSwiOverTi* [ms], is pre-parameterized to 250 ms. The manual function is deleted via input *bRstManFnct*.
- **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 [▶ 940] has determined the enable 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*).

Input stReferenceFeedback

Information about the controlled blind actuator or the reference actuator of a group is fed back into the blind control function via this input.

These are the position details and status of the reference actuator.

Data exchange HMI

The data exchange with the HMI is realized here in the base class *FB_BA_Ext_SunblindAngle* (internal function block). The use of the following variables is visible in this template:

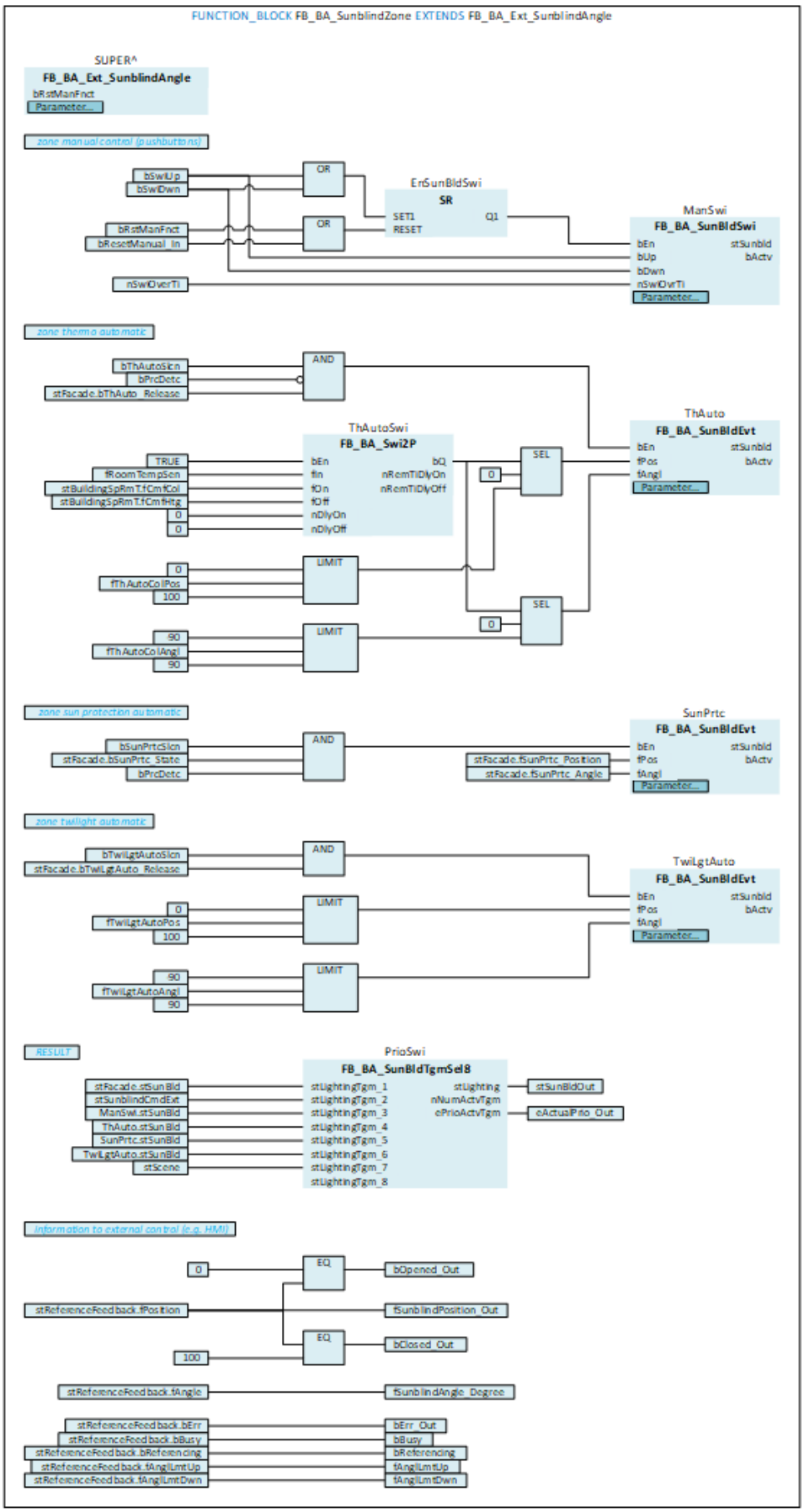
- **bResetManual_In**: Command from the HMI to delete the manual function.
- **fSunblindPosition_Out**: Output information position to the HMI.
- **fSunblindAngle_Degree**: Output information angle in degrees to the HMI.
- **bOpened_Out**: Output information "Blind completely open" to the HMI.
- **bClosed_Out**: Output information "Blind completely closed" to the HMI.
- **bErr_Out**: Output information "Reference actuator faulty" to the HMI.
- **eActualPrio_Out**: Currently controlling telegram priority to the HMI.
- **fAnglLmtUp:** Upper lamella angle of the reference actuator.
- **fAnglLmtDwn:** Lower lamella angle of the reference actuator.

The HMI is used to specify the lamella angles to be controlled in 0% (lamella at upper limit position *fAnglLmtUp*) to 100% (lamella at lower limit position *fAnglLmtDwn*), with 50% being the zero position (0° = lamella horizontal).

It must be ensured that the upper lamella angle is greater than 0° and the lower lamella angle is less than 0° at this point in order to be able to carry out the calculation correctly.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_SunblindZone EXTENDS FB_BA_Ext_SunblindAngle
VAR_INPUT
  fRoomTempSen           : REAL;
  bPrcDetc               : BOOL;
  bSwiUp                 : BOOL;
  bSwiDwn                : BOOL;
  bRstManFnct            : BOOL;
  stBuildingSpRmT        : ST_BA_SpRmT;
  stFacade               : ST_BA_Facade;
  stScene                : ST_BA_Sunbld;
  stReferenceFeedback    : ST_BA_SunblindActuatorFeedback;
END_VAR
VAR_OUTPUT
  stSunBldOut            : ST_BA_SunBld;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bThAutoSlcn            : BOOL;
  fThAutoColPos          : REAL;
  fThAutoColAngl         : REAL;
  bSunPrtcSlcn           : BOOL;
  bTwiLgtAutoSlcn        : BOOL;
  fTwiLgtAutoPos         : REAL;
  fTwiLgtAutoAngl        : REAL;
  nSwiOverTi             : UDINT;
END_VAR
VAR_INPUT CONSTANT
  ManSwi                 : FB_BA_SunBldSwi := (ePrio:= E_BA_SunBldPrio.eManualGroup);
  ThAuto                 : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupThermoAuto);
  SunPrtc                : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eSunProtection);
  TwiLgtAuto             : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupTwiLightAuto);

END_VAR
VAR
  ThAutoSwi              : FB_BA_Swi2P;
  PrioSwi                : FB_BA_SunBldTgmSel8;
  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 *bSwiUp/bSwiDwn*, and external control (e.g. HMI).
stBuildingSpRmT	ST_BA_SpRmT [▶ 272]	Structure of room setpoints (Protection Cooling..Comfort Cooling and Protection Heating ... Comfort Heating).
stFacade	ST_BA_Facade [▶ 704]	Facade-specific blind data and telegrams.
stScene	ST_BA_Sunbld [▶ 274]	Reserved telegram input for scene control.
stReferenceFeedback	ST_BA_SunblindActuatorFeedback [▶ 701]	Feedback input of the controlled blind actuator or the reference actuator of the controlled group.

Outputs

Name	Type	Description
stSunBldOut	ST_BA_SunBld [▶ 274]	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.
bTwiLgtAutoSlcn	BOOL	Twilight automatic enable.
fTwiLgtAutoPos	REAL	Twilight automatic position [%].
fTwiLgtAutoAngl	REAL	Twilight automatic position [°].
nSwiOverTi	UDINT	Changeover time [ms] of the manual switch *SunBldSwi* for latching.

Inputs CONSTANT

Name	Type	Description
ManSwi	FB_BA_SunBldSwi [▶ 388]	Group (zone) push button block.
ThAuto	FB_BA_SunBldEvt [▶ 378]	Telegram block for group (zone) thermal automatic.
SunPrtc	FB_BA_SunBldEvt [▶ 378]	Telegram block for group (zone) shading automatic.
TwiLgtAuto	FB_BA_SunBldEvt [▶ 378]	Telegram block for group (zone) twilight automatic.

VAR

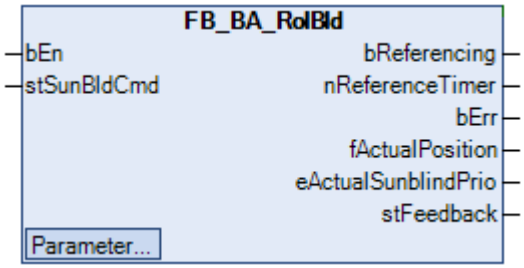
Name	Type	Description
ThAutoSwi	FB_BA_Swi2P	Hysteresis block for heating/cooling changeover for group (zone) thermal automatic.
PrioSwi	FB_BA_SunBldTgmSel8 [▶ 391]	Selection of the resulting telegram.
EnSunBldSwi	SR	Activation memory of the above mentioned function block.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.4.6.2 Actuators

6.1.4.2.1.4.6.2.1 FB_BA_RolBld



This template is used to control a blind actuator without slat adjustment.

The resulting diagram from the zone *stSunBldCmd* is routed via a drive delay FB_BA_SunBldPosDly [▶ 381].

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 current blind position is made available in BACnet via an analog object FB_BA_AV_Op [▶ 202].

The template has a feedback structure stFeedback [▶ 701] at the output.

If the programmed blind actuator is an individual actuator or the reference actuator of a group, this feedback telegram must be linked to the controlling room (zone) user function in order to obtain information about the state of the actuator.

The initialization of the template takes place within the method FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_RolBld EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  stSunBldCmd           : ST_BA_SunBld;
END_VAR
VAR_OUTPUT
  bReferencing          : BOOL;
  nReferenceTimer       : UDINT;
  bErr                  : BOOL;
  fActualPosition       : REAL;
  eActualSunblindPrio   : BYTE;
  stFeedback            : ST_BA_SunblindActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiOverTi            : UDINT;
  nPositioningDelay     : UDINT;
  nTiUp                 : UDINT;
  nTiDwn                : UDINT;
END_VAR
VAR_INPUT CONSTANT
  ActualPosition        : FB_BA_AV_Op;
END_VAR
VAR
  PositioningDelay      : FB_BA_SunBldPosDly;
  RolBldActr            : FB_BA_RolBldActr;
  bCmdUp      AT %Q*    : BOOL;
  bCmdDown    AT %Q*    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enable the function block function.
stSunBldCmd	ST_BA_SunBld [▶ 274]	Resulting telegram from the higher-level zone (room).

Outputs

Name	Type	Description
bReferencing	BOOL	Blind is being referenced.
nReferenceTimer	UDINT	Elapsed referencing time [s].
bErr	BOOL	The internal function block *FB_BA_RolBldActr* is parameterized incorrectly.
fActualPosition	REAL	Current position (calculated).
eActualSunblindPrio	BYTE	Current priority with which the blind actuator is controlled.
stFeedback	ST_BA_SunblindActuatorFeedback [▶ 701]	Feedback telegram for linking to the controlling room (zone) user function. In this way, information about the state of the blind actuator is fed back to the user function.

Inputs CONSTANT PERSISTENT

Name	Type	Description
nSwiOverTi	UDINT	Changeover time [ms] of the manual switch *SunBldSwi* 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].

Inputs CONSTANT

Name	Type	Description
ActualPosition	FB_BA_AV_Op [▶ 202]	Function block for displaying the position [%] in BACnet.

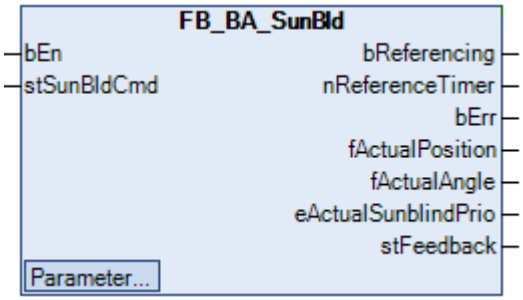
Variables

Name	Type	Description
PositioningDelay	FB_BA_SunBldPosDly [▶ 381]	Delay of telegrams from the zone (group) to avoid high breakaway starting currents due to simultaneity.
RolBldActr	FB_BA_RolBldActr [▶ 365]	Function block for controlling a blind without slat adjustment.
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

6.1.4.2.1.4.6.2.2 FB_BA_SunBld



This template is used to control a blind actuator with slat adjustment.

The resulting diagram from the zone *stSunBldCmd* is routed via a drive delay FB_BA_SunBldPosDly [▶ 381].

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 current blind position and the current angle are made available in BACnet via two analog objects FB_BA_AV_Op [▶ 202].

The template has a feedback structure stFeedback [▶ 701] at the output.

If the programmed blind actuator is an individual actuator or the reference actuator of a group, this feedback telegram must be linked to the controlling room (zone) user function in order to obtain information about the state of the actuator.

The initialization of the template takes place within the method FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_SunBld EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  stSunBldCmd           : ST_BA_SunBld;
END_VAR
VAR_OUTPUT
  bReferencing          : BOOL;
  nReferenceTimer       : UDINT;
  bErr                  : BOOL;
  fActualPosition       : REAL;
  fActualAngle          : REAL;
  eActualSunblindPrio   : BYTE;
  stFeedback            : ST_BA_SunblindActuatorFeedback;
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_INPUT CONSTANT
  ActualPosition        : FB_BA_AV_Op;
  ActualAngle           : FB_BA_AV_Op;
END_VAR
VAR
  PositioningDly        : FB_BA_SunBldPosDly;
  SunBldActr            : FB_BA_SunBldActr;
  bCmdUp      AT %Q*    : BOOL;
  bCmdDown    AT %Q*    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	Enable the function block function.
stSunBldCmd	ST_BA_SunBld [▶ 274]	Resulting telegram from the higher-level zone (room).

Outputs

Name	Type	Description
bReferencing	BOOL	Blind is being referenced.
nReferenceTimer	UDINT	Elapsed referencing time [s].
bErr	BOOL	The internal function block *FB_BA_SunBldActr* is parameterized incorrectly.
fActualPosition	REAL	Current position (calculated).
fActualAngle	REAL	Current angle (calculated).
eActualSunblindPrio	BYTE	Current priority with which the blind actuator is controlled.
stFeedback	ST_BA_SunblindActuatorFeedback [▶ 701]	Feedback telegram for linking to the controlling room (zone) user function. In this way, information about the state of the blind actuator is fed back to the user function.

Inputs CONSTANT PERSISTENT

Name	Type	Description
nSwiOverTi	UDINT	Changeover time [ms] of the manual switch *SunBldSwi* 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.

Inputs CONSTANT

Name	Type	Description
ActualPosition	FB_BA_AV_Op [▶ 202]	Function block for displaying the position [%] in BACnet.
ActualAngle	FB_BA_AV_Op [▶ 202]	Function block for displaying the angle [°] in BACnet.

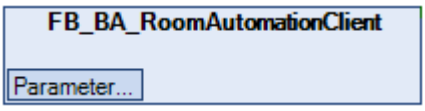
Variables

Name	Type	Description
PositioningDelay	FB_BA_SunBldPosDly [▶ 381]	Delay of telegrams from the zone (group) to avoid high breakaway starting currents due to simultaneity.
SunBldActr	FB_BA_SunBldActr [▶ 374]	Function block for controlling a blind.
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

6.1.4.2.1.4.7 FB_BA_RoomAutomationClient



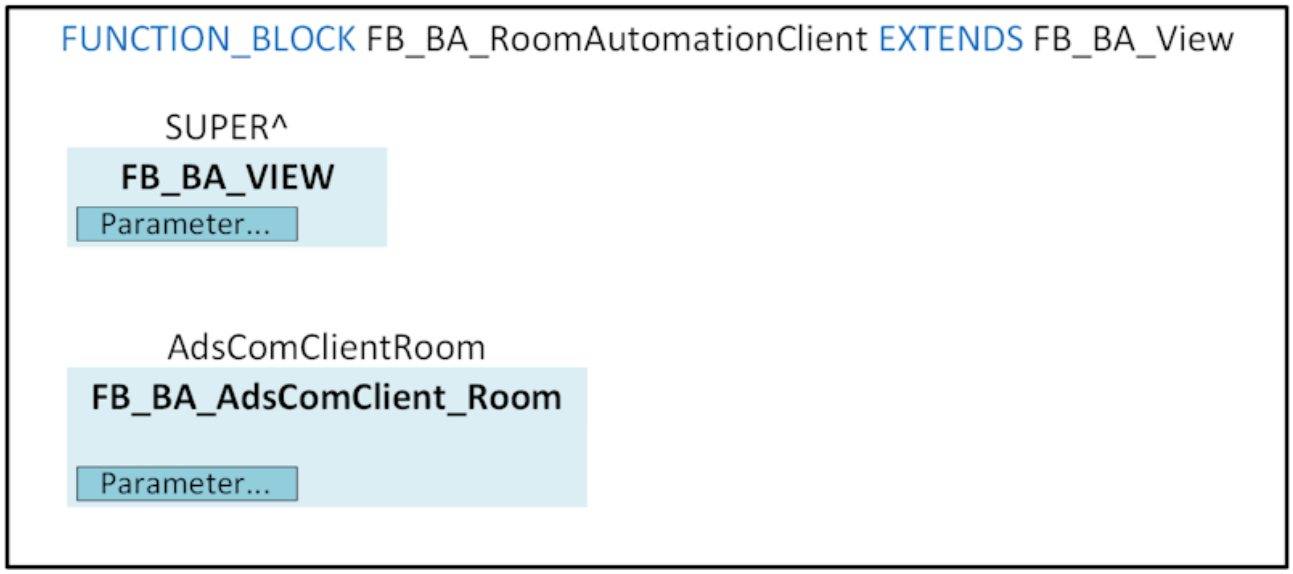
Call template trade "Room automation" - client version.

This template at the "trade" level calls up all sub-templates whose data is ideally prepared in a floor computer.

On the one hand, this is data to be assigned to the "room automation" trade, which is transmitted and read by a building computer (FB_BA_AdsComClient_Room [▶ 887]), and on the other hand, this template contains the calls of all rooms.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_RoomAutomationClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComClientRoom    : FB_BA_AdsComClient_Room;
  SampleRoom          : FB_BA_RoomSample1;
END_VAR
```

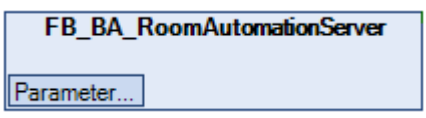
VAR_INPUT CONSTANT

Name	Type	Description
AdsComClientRoom	FB_BA_AdsComClient_Room [▶ 887]	The sub-template has the task of reading data and parameters sent from other computers, ideally from the building computer.
SampleRoom	FB_BA_RoomSample1 [▶ 931]	The sub-template represents an example room with shading, illumination and air conditioning. This room has a shaded area within a facade whose data it receives via *site.arrFacade[2]*. In terms of lighting, the room is assigned to area 1, which is expressed by the receipt of *site.arrAreaLighting[1]*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.4.8 FB_BA_RoomAutomationServer



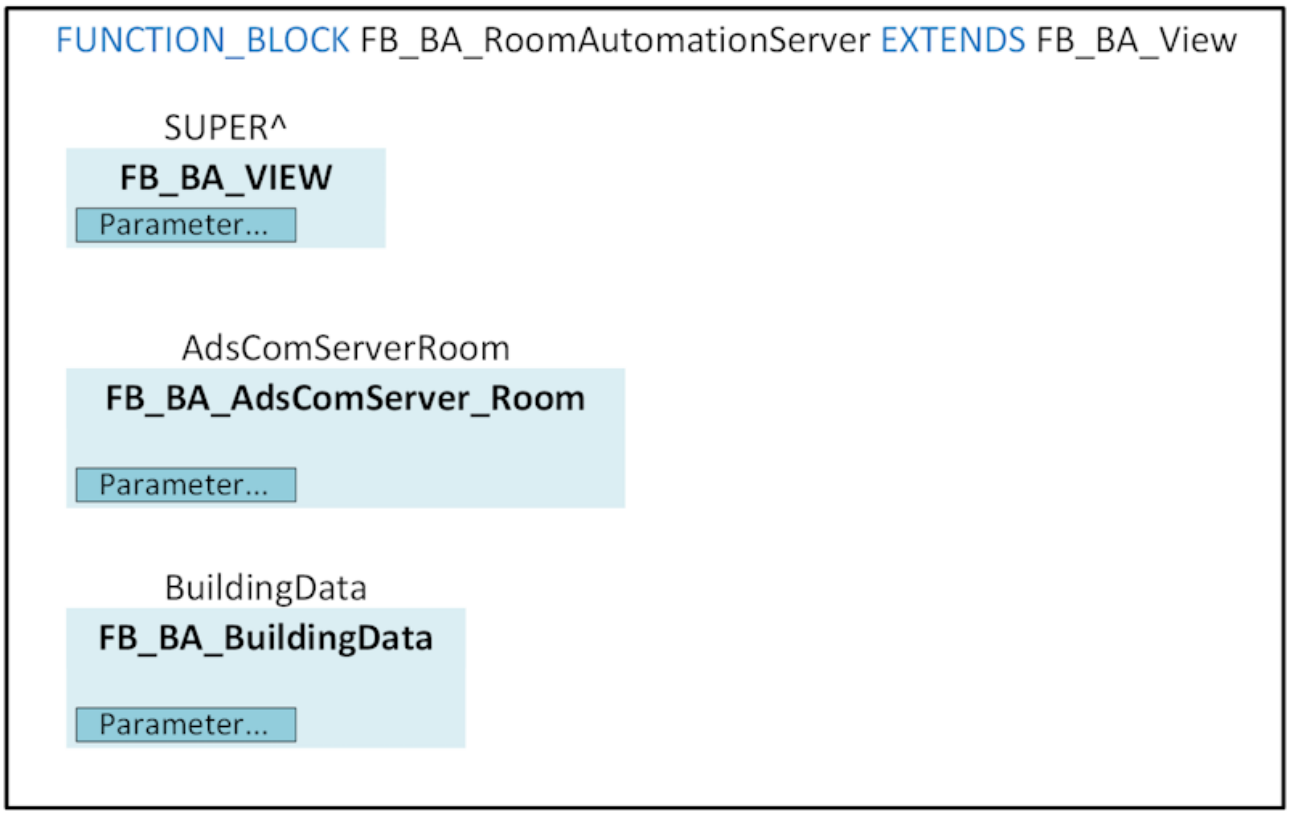
Call template trade "Room automation" - server version.

This template at the "trade" level calls up all sub-templates whose data is ideally processed in a building calculator.

In addition, room automation-specific data is made available building-wide via an ADS server template.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_RoomAutomationServer EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComServerRoom    : FB_BA_AdsComServer_Room;
  BuildingData        : FB_BA_BuildingData;
END_VAR
```

VAR_INPUT CONSTANT

Name	Type	Description
AdsComServerRoom	FB_BA_AdsComServer_Room [▶ 888]	The sub-template has the task of transmitting data and parameters within the BA network.
BuildingData	FB_BA_BuildingData [▶ 858]	The sub-template calls up further templates that provide building-wide data and operation modes.

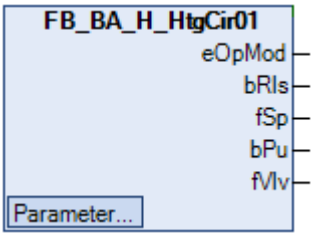
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.5 HeatingSystem

6.1.4.2.1.5.1 Distribution

6.1.4.2.1.5.1.1 FB_BA_H_HtgCir01



Template is used to program a static heating circuit.

The main components of the template are:

- Flow temperature control.
- Heating curve dependent on outside temperature with night setback.
- Operation mode selection.
- Control of the heating circuit pump.
- Control of an analog control valve.

The initialization of the template takes place within the method FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_H_HtgCir01 EXTENDS FB_BA_View
VAR_OUTPUT
  eOpMod       : E_BA_EnergyLvl;
  bRls         : BOOL;
  fSp          : REAL;
  bPu          : BOOL;
  fVlv         : REAL;
END_VAR
VAR_INPUT CONSTANT
  TFl          : FB_BA_SensorAnalog_Raw;
  TRt          : FB_BA_SensorAnalog_Raw;
  Vlv          : FB_BA_Vlv;
  Pu           : FB_BA_Pulst;
  Sp           : FB_BA_H_HtgCir_Sp;
  OpMod        : FB_BA_H_OpMod;
  HtgLmt       : FB_BA_HtgLmt;
  TFlCtrl      : FB_BA_PID;
  PlantLock    : FB_BA_PlantLock;
END_VAR
```

Outputs

Name	Type	Description
eOpMod	E_BA_EnergyLvl [▶ 700]	Operation mode of the heating circuit.
bRls	BOOL	The variable indicates that the heating circuit is in operation.
fSp	REAL	Calculated setpoint of the heating characteristic curve.
bPu	BOOL	Enable the heating circuit pump.
fVlv	REAL	Calculated control value for the valve.

Inputs CONSTANT

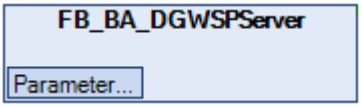
Name	Type	Description
TFl	FB_BA_SensorAnalog [▶ 1087]	The function block represents the flow temperature.
TRt	FB_BA_SensorAnalog [▶ 1087]	The function block represents the return temperature.
Vlv	FB_BA_Vlv [▶ 1092]	Control valve
Pu	FB_BA_Pu1st [▶ 1074]	Heating circuit pump
Sp	FB_BA_H_HtgCir_Sp	The function block calculates the setpoint of the flow temperature depending on the outside temperature.
OpMod	FB_BA_H_OpMod	Operation mode selection of the heating circuit (day, night, protection mode).
HtgLmt	FB_BA_HtgLmt	The function block enables heating operation below a heating limit temperature.
TFlCtrl	FB_BA_PID [▶ 1037]	Return temperature sensor
PlantLock	FB_BA_PlantLock [▶ 135]	When the function block is called, the relevant faults of the event-enabled objects are collected and output at this and lower levels of the heating circuit. These relevant faults trigger the operation mode "Fault". This switches the heating circuit to "Protection" mode.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.1.6 DoorsGatesWindowsSunProtection

6.1.4.2.1.6.1 FB_BA_DGWSPServer



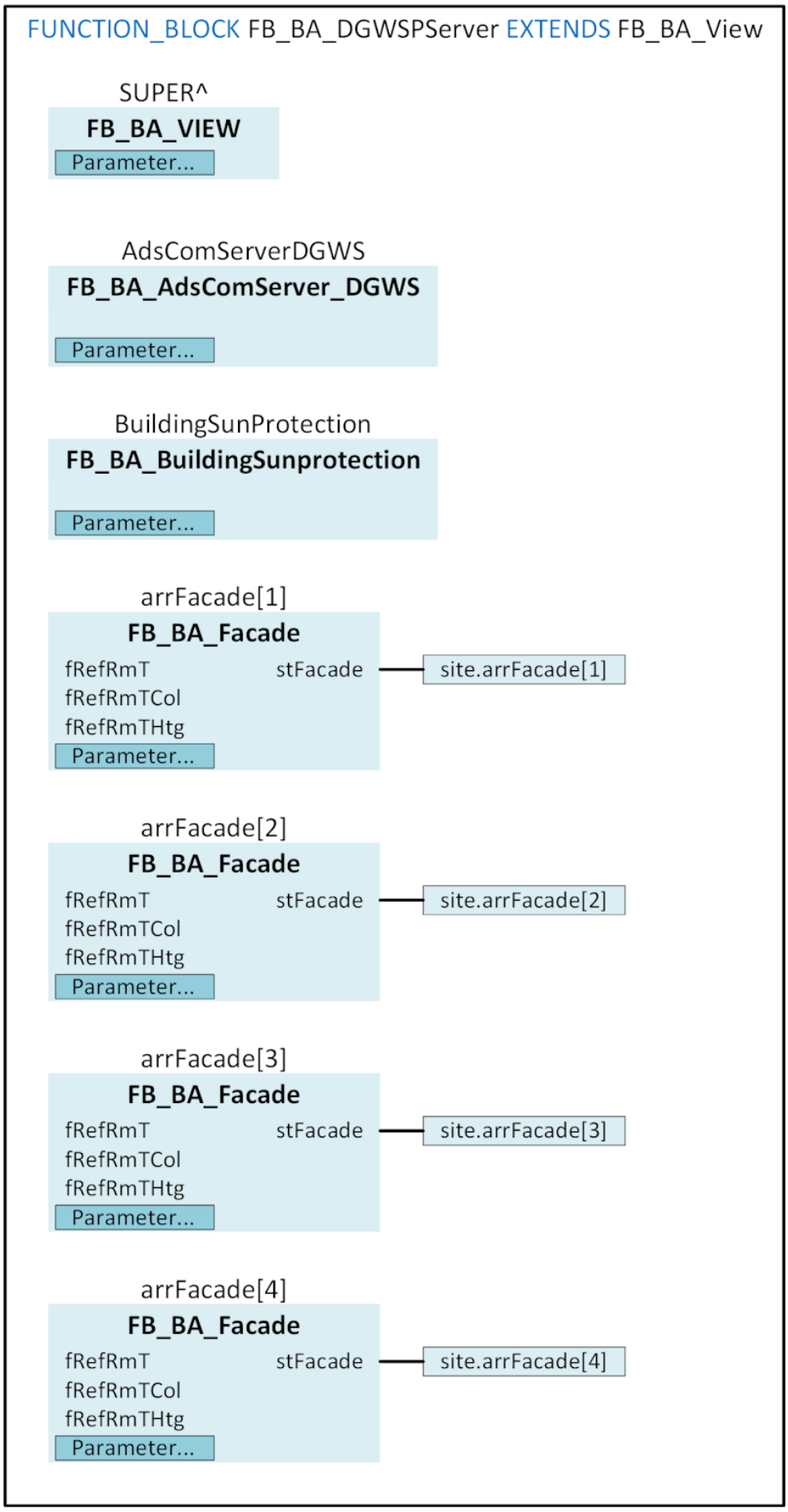
Call template trade "Doors, gates, windows, sun protection" - server version.

This template at the "trade" level calls up all sub-templates whose data is ideally processed in a building calculator. On the one hand, these are general data for the sun protection, as well as the data for all facades.

In addition, trade-specific data is made available building-wide via an ADS server template.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_DGWSPServer EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComServerDGWS          : FB_BA_AdsComServer_DGWS;
  BuildingSunProtection     : FB_BA_BuildingSunprotection;
  arrFacade                 : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_Facade;
END_VAR
```

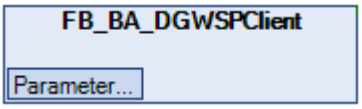
VAR_INPUT CONSTANT

Name	Type	Description
AdsComServerDGWS	FB_BA_AdsComServer_DGWS [▶ 980]	The sub-template has the task of transmitting data and parameters within the BA network.
BuildingSunProtection	FB_BA_BuildingSunprotection [▶ 937]	The sub-template provides building-wide data and operation modes.
arrFacade	FB_BA_Facade [▶ 940]	These templates of type *FB_BA_Facade* compile the sun protection telegrams that are valid for an entire facade. In the global parameter list BA2_Param [▶ 1118], 4 facades are predefined with *nMaxNumberOfFacades*, which are initialized here in *FB_Init* to north, east, south and west as examples.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.6.2 FB_BA_DGWSPClient

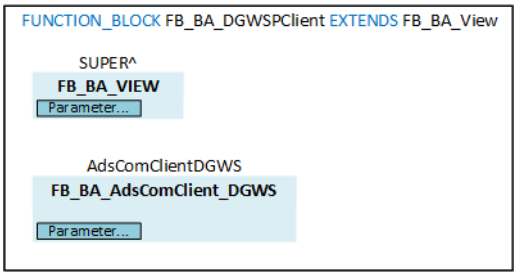


Call template trade "Doors, gates, windows, sun protection" - client version.

This template at the "trade" level calls up all sub-templates whose data is ideally prepared in a floor computer.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_DGWSPClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComClientDGWS    : FB_BA_AdsComClient_DGWS;
END_VAR
```

VAR_INPUT CONSTANT

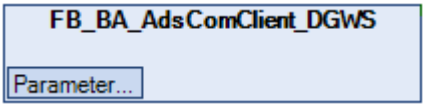
Name	Type	Description
AdsComClientDGWS	FB_BA_AdsComClient_DGWS [▶ 979]	The sub-template has the task of reading data and parameters sent from other computers, ideally from the building computer.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.1.6.3 Communication

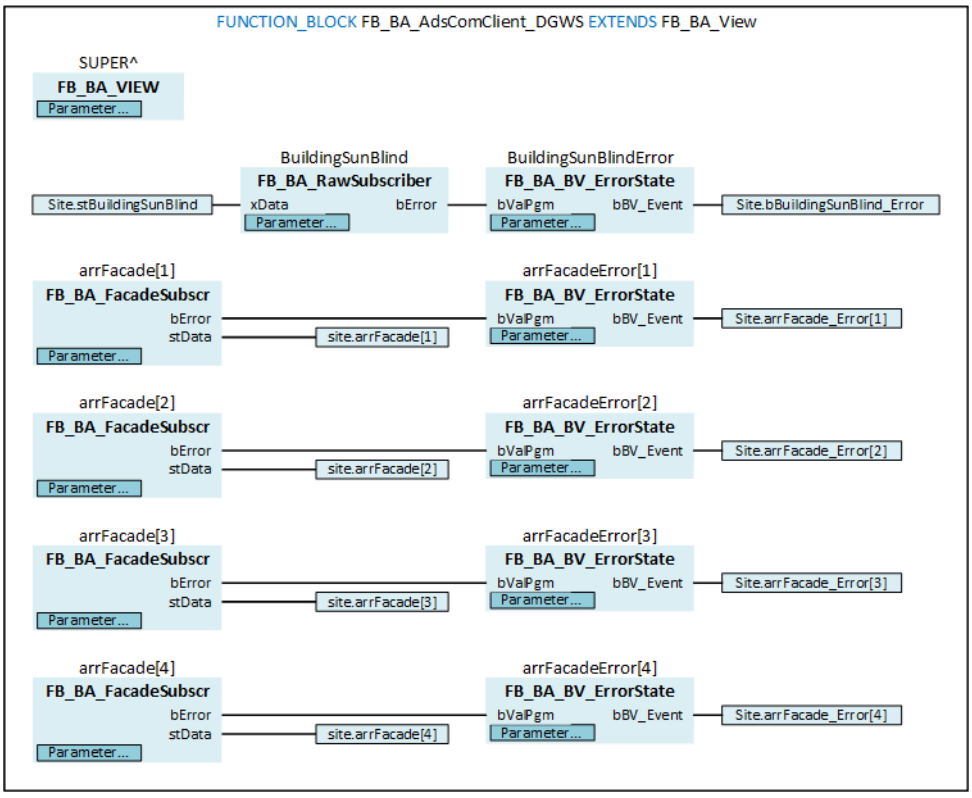
6.1.4.2.1.6.3.1 FB_BA_AdsComClient_DGWS



The template reads global data that is assigned to the "Doors, gates, windows, sun protection" trade from another controller that provides this data with the counter-block FB_BA_AdsComServer_DGWS [▶ 980]. The information read is copied to the global variable list Site [▶ 1116].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  BuildingSunBlind         : FB_BA_RawSubscriber;
  BuildingSunBlindError    : FB_BA_BV_ErrorState;
  arrFacade                : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_FacadeSubscr;
  arrFacadeError           : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_BV_ErrorState;
END_VAR
```

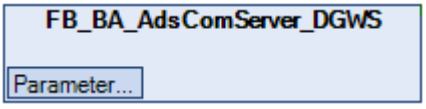
VAR_INPUT CONSTANT

Name	Type	Description
BuildingSunBlind	FB_BA_RawSubscriber [▶ 158]	The subscriber accesses the TwinCAT network structure *BuildingSunBlind* and saves the data in the created structure *stBuildingSunBlind* of the GVL Site [▶ 1116].
BuildingSunBlindError	FB_BA_BV_ErrorState [▶ 1030]	Binary object indicating communication failure of the subscriber *BuildingSunBlind*.
arrFacade	FB_BA_FacadeSubscr [▶ 945]	These templates receive the facade data that has been defined or made available elsewhere via an FB_BA_Facade [▶ 940]. In the global parameter list BA2_Param [▶ 1118], 4 facades are predefined with *nMaxNumberOfFacades*, which are initialized here in *FB_Init* verbally to north, east, south and west.
arrFacadeError	FB_BA_BV_ErrorState [▶ 1030]	Binary objects indicating communication failure of the facade subscriber.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

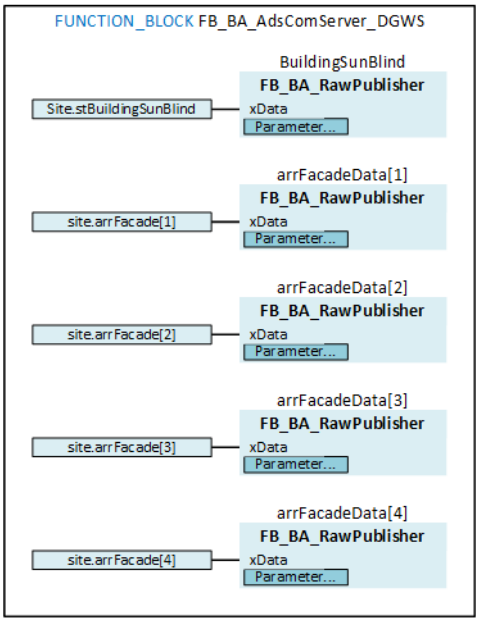
6.1.4.2.1.6.3.2 FB_BA_AdsComServer_DGWS



The template has the task of reading data and parameters that are assigned to the "Doors, gates, windows, sun protection" trade from the global variable list Site [▶ 1116] and making them available within the BA network via FB_BA_RawPublisher [▶ 139].

The counter-block that receives this data as a client is FB_BA_AdsComClient_DGWS [▶ 979].

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComServer
VAR_INPUT CONSTANT
  BuildingSunBlind    : FB_BA_RawPublisher;
  arrFacadeData       : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_RawPublisher;
END_VAR
```

VAR_INPUT CONSTANT

Name	Type	Description
BuildingSunBlind	FB_BA_RawPublisher [▶ 139]	The publisher publishes the structure of the building-wide sun protection data Site.stBuildingSunBlind [▶ 1116], which is generated in the template FB_BA_BuildingSunprotection [▶ 937].
arrFacadeData	FB_BA_RawPublisher [▶ 139]	The publishers *arrFacadeData[1..4]* publish the structures of the building-wide valid shading data site.arrFacade[1..4] [▶ 1116], which is generated in the instances of the template FB_BA_Facade [▶ 940]. The number of facade instances depends on the building floor plan. The facade number "4" has been used here as typical.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

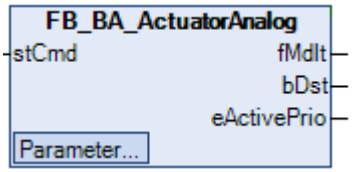
6.1.4.2.2 Universal

Templates for universal use in various building functions.

6.1.4.2.2.1 Aggregates

6.1.4.2.2.1.1 Analog

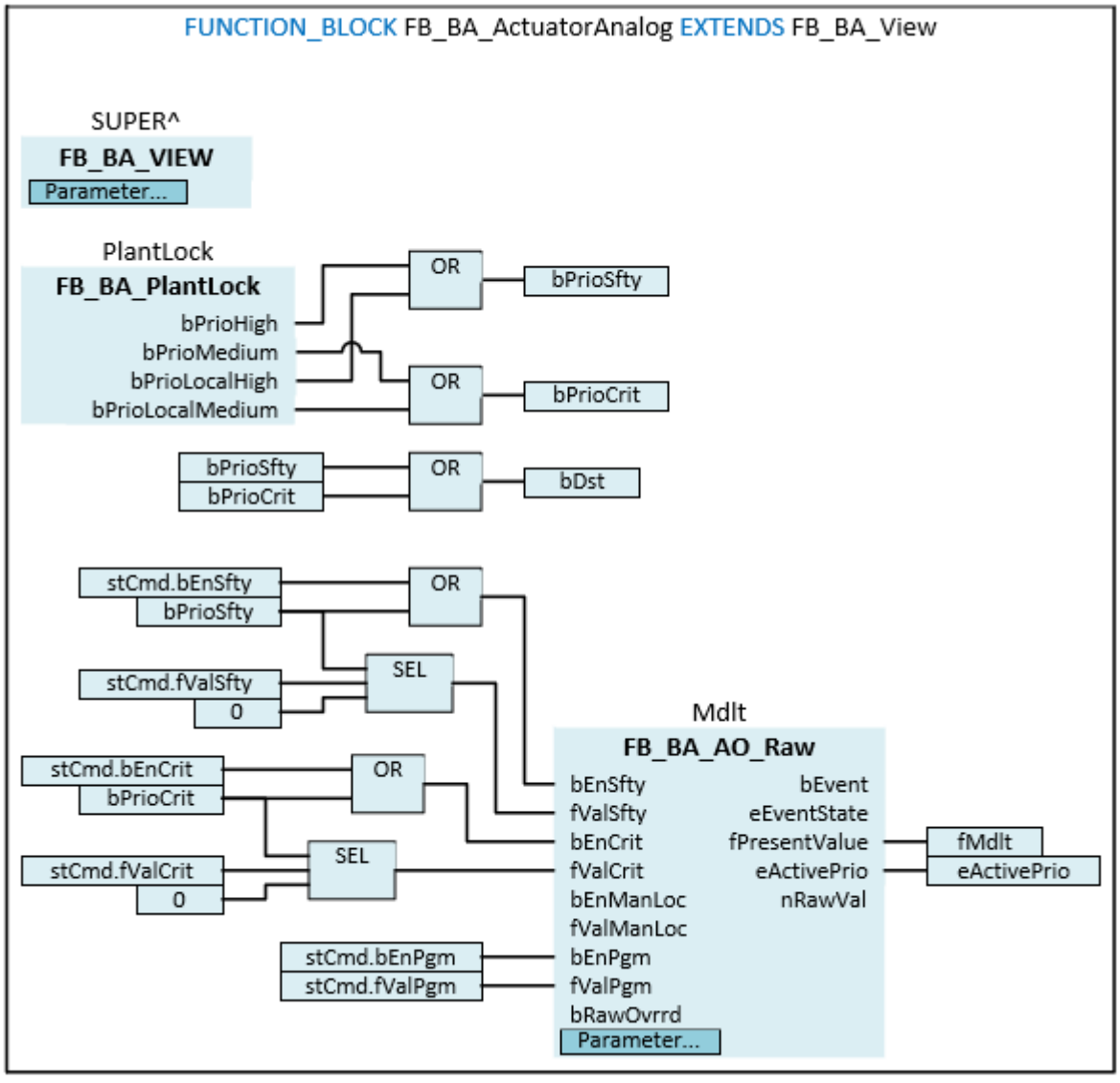
6.1.4.2.2.1.1.1 FB_BA_ActuatorAnalog



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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 [▶ 276]	The enables and switching values are transmitted to the template via the command structure *stCmd* . The command with the highest priority is output at the analog output object *Mdlt*.

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 [▶ 110]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Mdlt	FB_BA_AO_Raw [▶ 199]	Current value of the analog output object.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

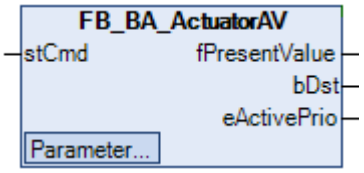
Variables

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

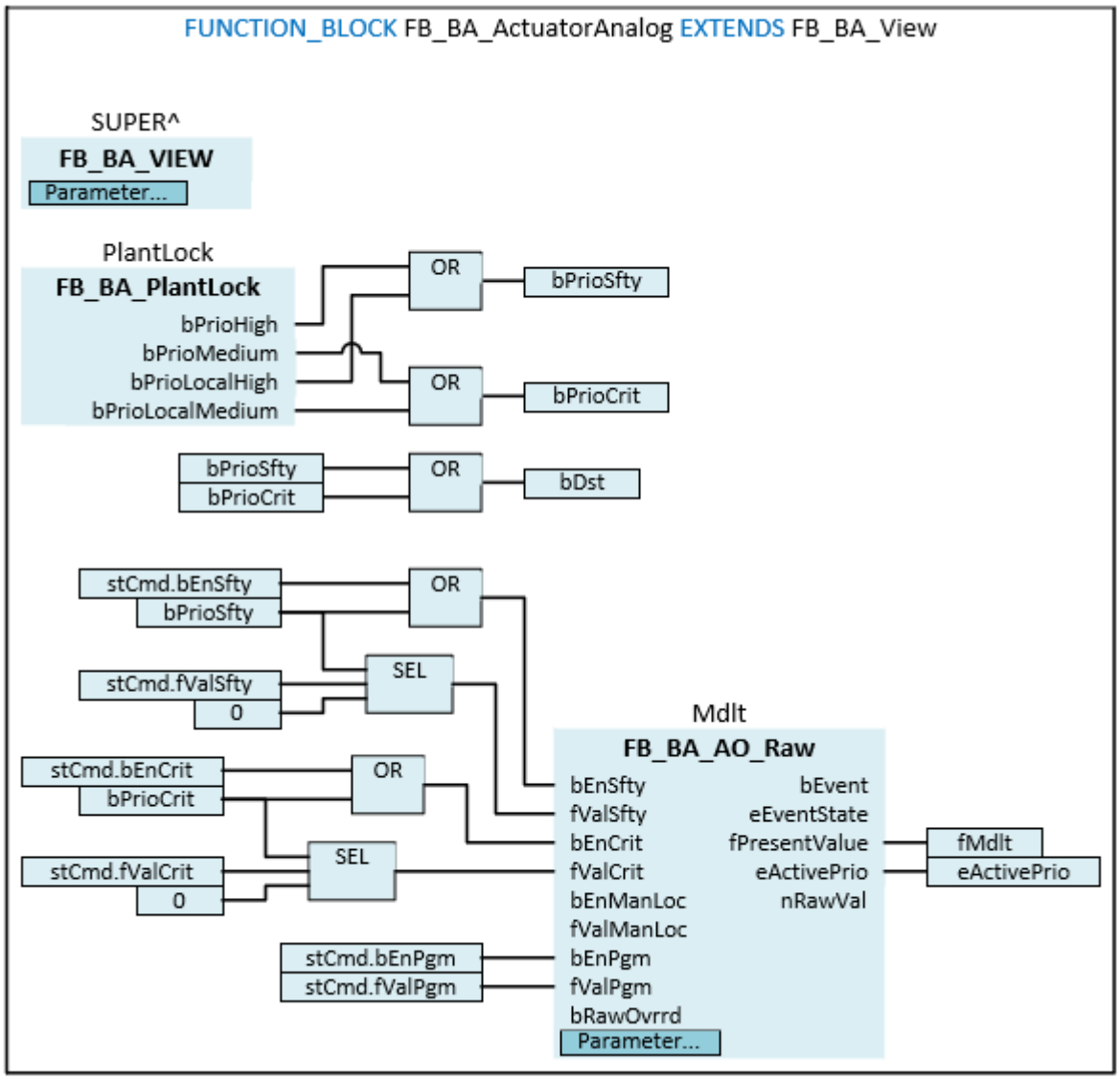
6.1.4.2.2.1.1.2 FB_BA_ActuatorAV



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.

Block diagram



Syntax

```
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 [▶ 276]	The enables and switching values are transmitted to the template via the command structure *stCmd* . The command with the highest priority is output at the analog value object *Val*.

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 [▶ 110]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Val	FB_BA_AV [▶ 200]	The analog value object determines the current control value.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Variables

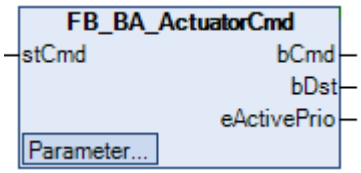
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.

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

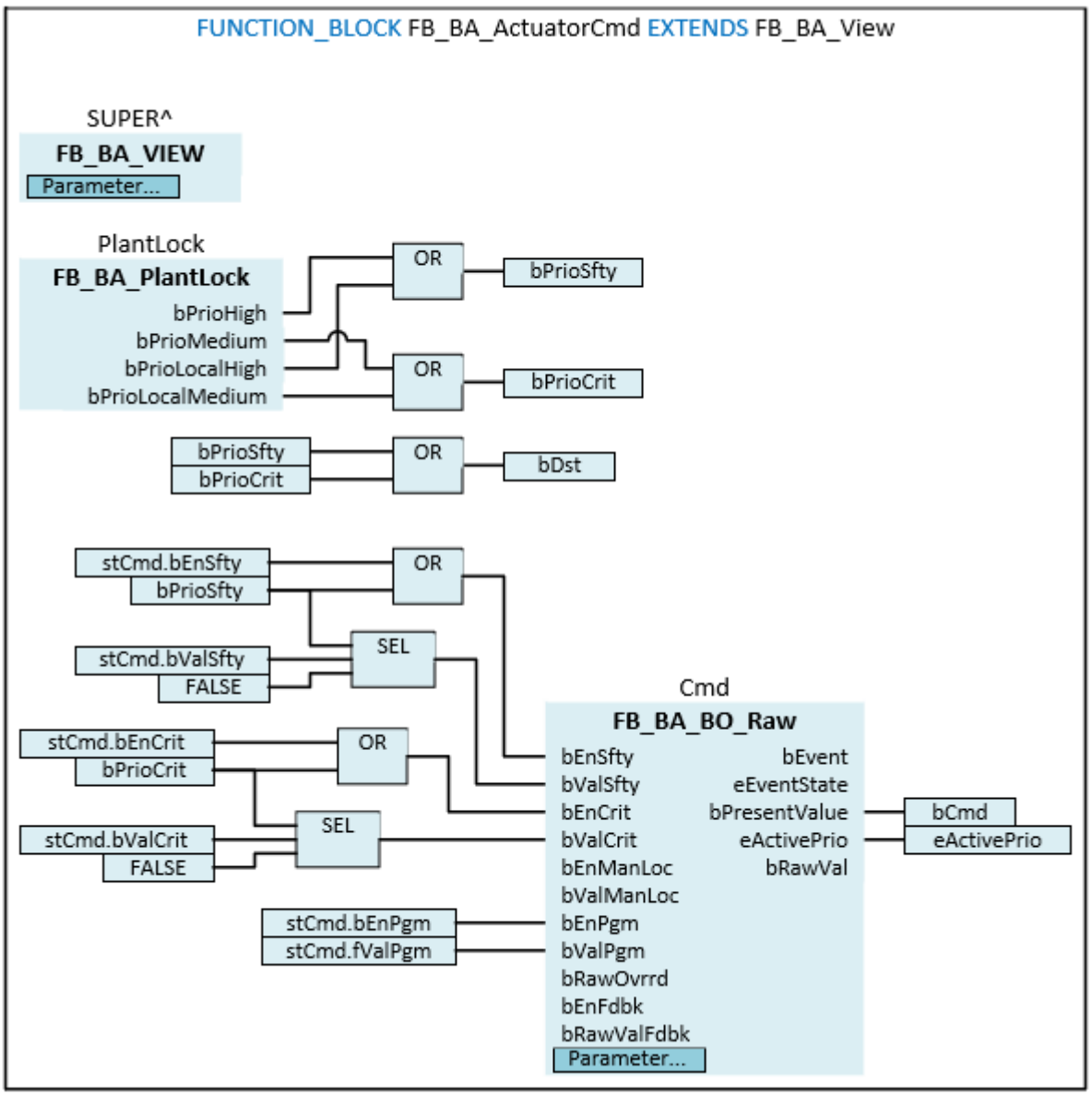
6.1.4.2.2.1.2.1 FB_BA_ActuatorCmd



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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 [▶ 277]	The enables and switching values are transmitted to the template via the command structure *stCmd* . The command with the highest priority is output at the binary output object *Cmd*.

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 [▶ 110]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Cmd	FB_BA_BO_Raw [▶ 211]	The binary output object is used to output a switching command and transmit it to the I/O level.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Variables

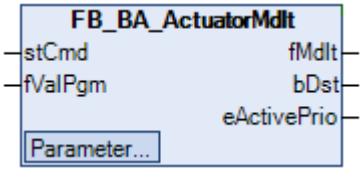
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.

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

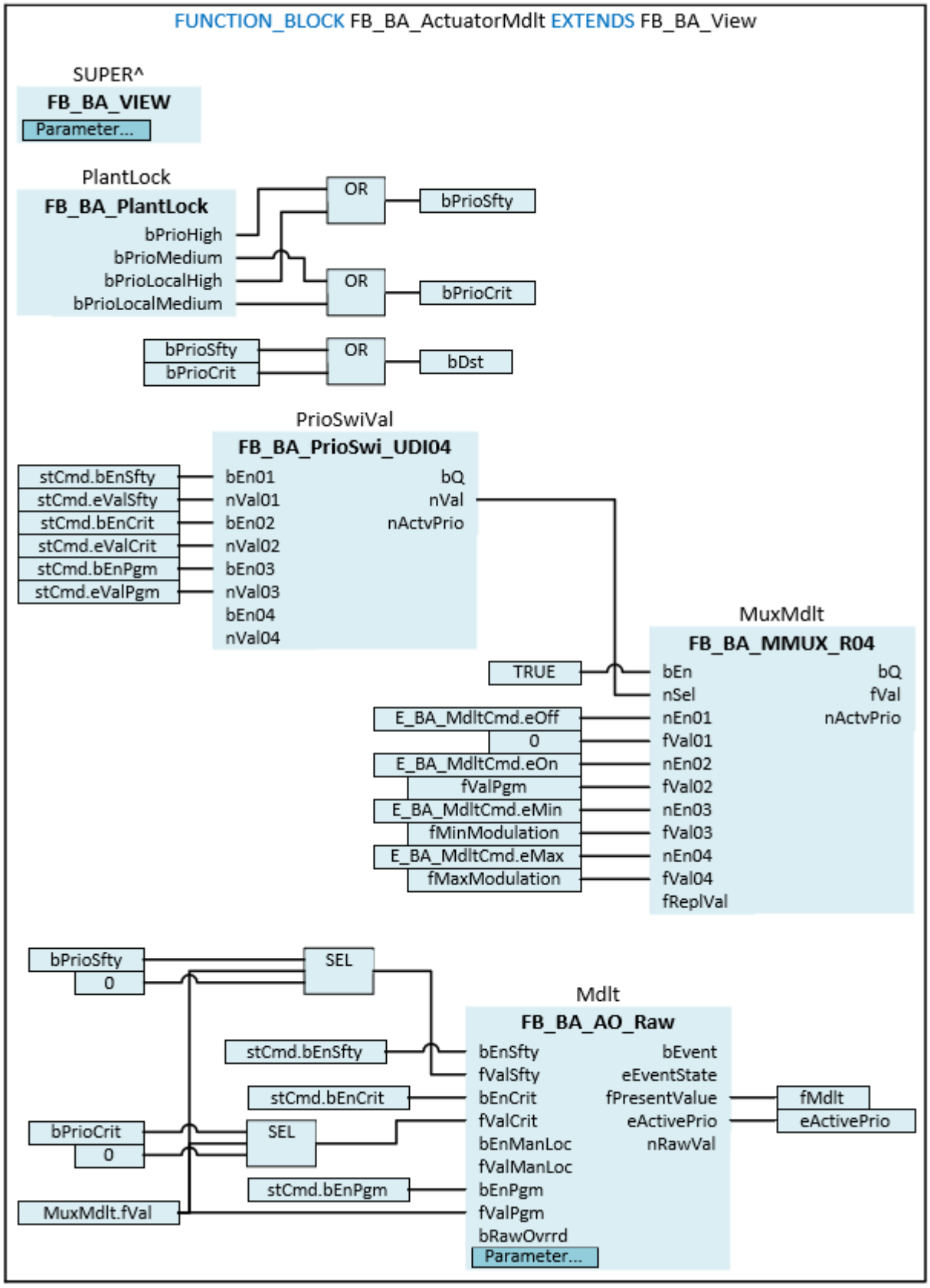
6.1.4.2.2.1.3.1 FB_BA_ActuatorMdlt



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 *MuxMdlt* for determining the control value and the function block *PlantLock*, which collects all safety-related faults.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorMdlt EXTENDS FB_BA_View
VAR_INPUT
  stCmd             : ST_BA_Mdlt;
  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 [▶ 276]	The enables and modulation commands are transmitted to the template via the command structure *stCmd*. The command with the highest priority determines the switching command at the binary output object *Cmd* and the control command at the analog output object *Mdlt*.
fValPgm	REAL	The control signal for modulation command *E_BA_MdltCmd.eOn* is transmitted to the template via the input variable *fValPgm*.

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 [▶ 110]	Display of the active priority.

Inputs CONSTANT PERSISTENT

Name	Type	Description
fMinModulation	REAL	Constant minimum value when modulation command E_BA_MdltCmd.eMin is pending.
fMaxModulation	REAL	Constant maximum value when modulation command E_BA_MdltCmd.eMax is pending.

Inputs CONSTANT

Name	Type	Description
Mdlt	FB_BA_AO_Raw [▶ 199]	Current value of the analog output object.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

Variables

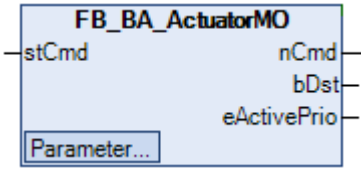
Name	Type	Description
bPrioSfty	BOOL	The variable is an evaluation of the "Safety" lock priority of the project structure.
bPrioCrit	BOOL	The variable is an evaluation of the "Critical" lock priority of the project structure.
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch PrioSwiVal [▶ 433] uses the command structure *stCmd* to determine the modulation command for the multiplexers *MuxCmd* and *MuxMdlt*.
MuxMdlt	FB_BA_MMUX_R04 [▶ 428]	The multiplexer MuxMdlt [▶ 428] determines the current control value from the modulation values *fValPgm, fMinModulation* and *fMaxModulation* and the modulation command of the priority switch *PrioSwiVal.* The result is sent to the analog output object *Mdlt*.

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

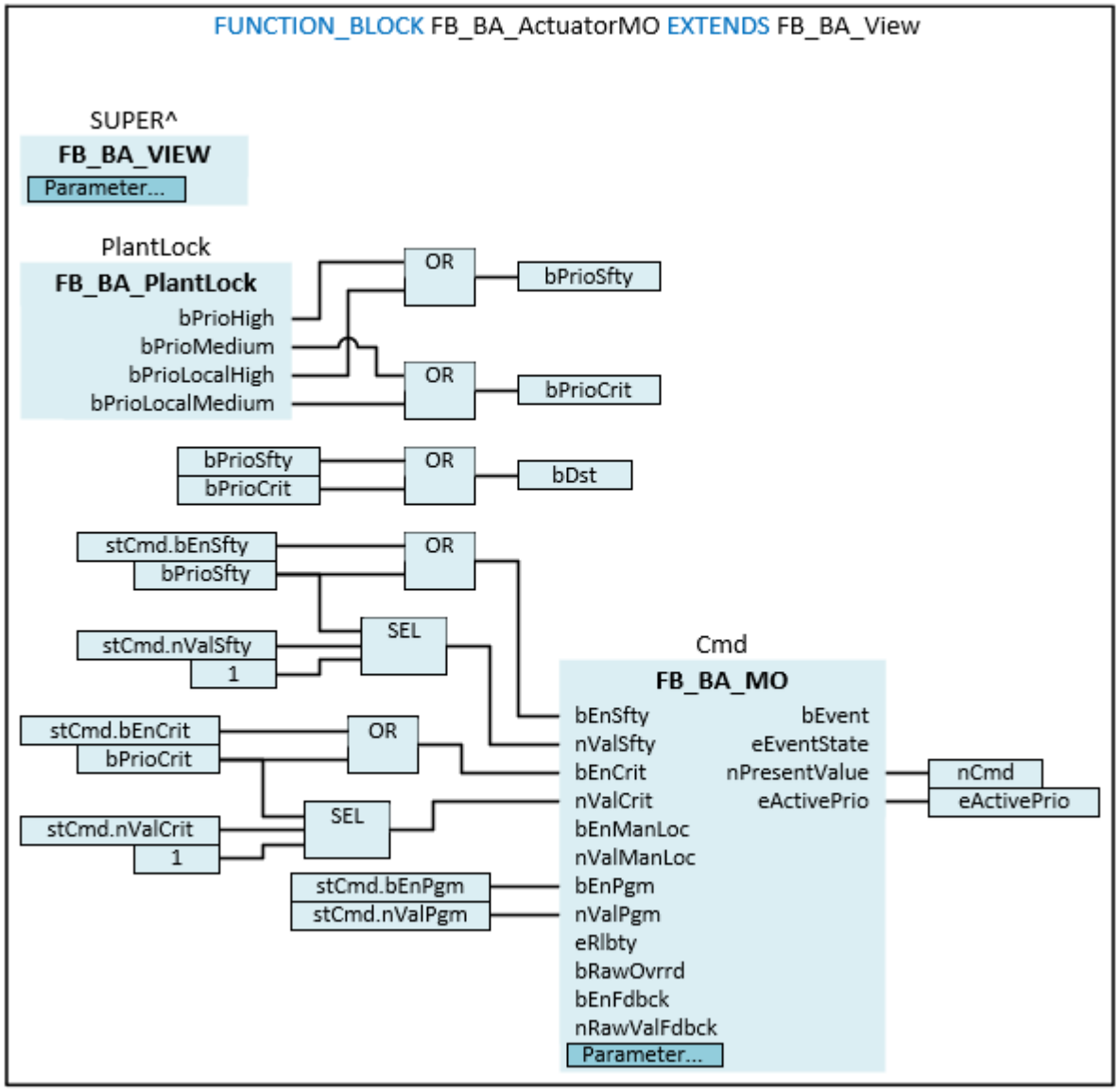
6.1.4.2.2.1.4.1 FB_BA_ActuatorMO



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.

Block diagram



Syntax

```
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 [▶ 275]	The enables and switching values are transmitted to the template via the command structure *stCmd* . The command with the highest priority is output at the multistate output object *Cmd*.

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 [▶ 110]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Cmd	FB_BA_MO [▶ 232]	The multistate output object is used to output the current switch value.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

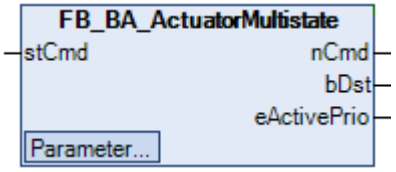
Variables

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

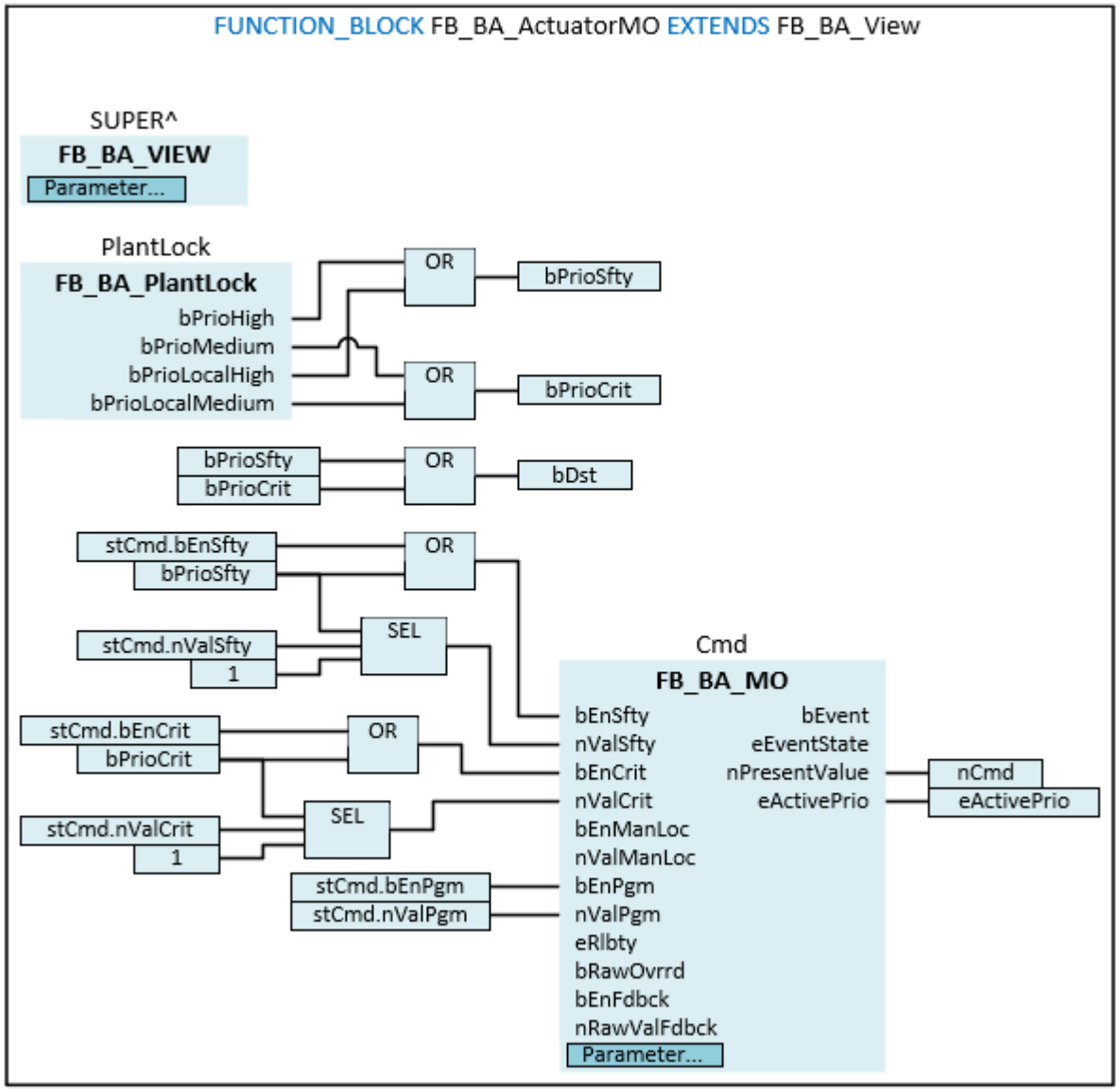
6.1.4.2.2.1.4.2 FB_BA_ActuatorMultistate



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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 [▶ 275]	The enables and switching values are transmitted to the template via the command structure *stCmd* . The command with the highest priority is output at the multistate output object *Cmd*.

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 [▶ 110]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Cmd	FB_BA_MO_Raw [▶ 235]	The multistate output object is used to output the current switch value.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* 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

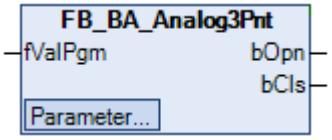
Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2 Control

Templates for presence monitoring and scaling.

6.1.4.2.2.2.1 Analog3Point

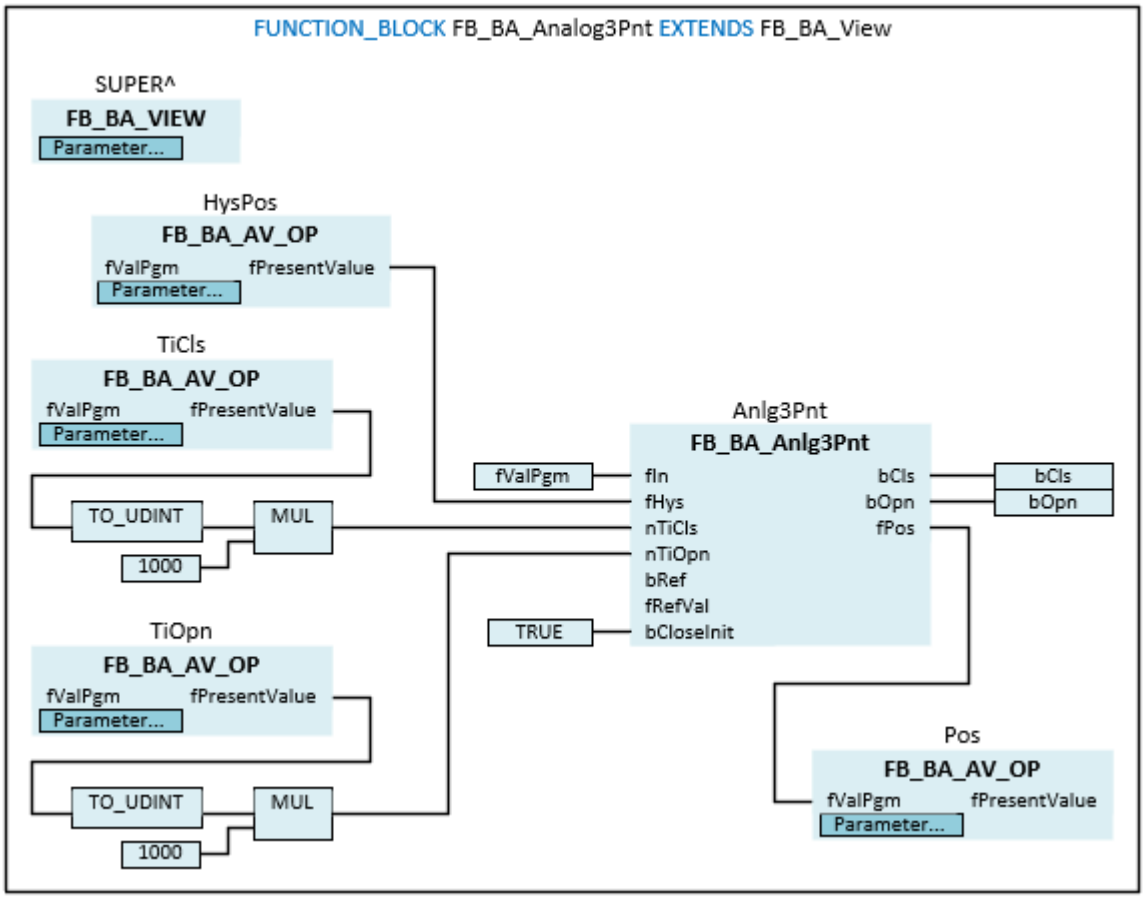
6.1.4.2.2.2.1.1 FB_BA_Analog3Pnt



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.

Block diagram



Syntax

```
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 [▶ 202]	AV object for entering the hysteresis value to start the position change.
TiOpn	FB_BA_AV_Op [▶ 202]	AV object for entering the opening time value.
TiCls	FB_BA_AV_Op [▶ 202]	AV object for entering the closing time value.
Pos	FB_BA_AV_Op [▶ 202]	Display of the calculated position.

VAR

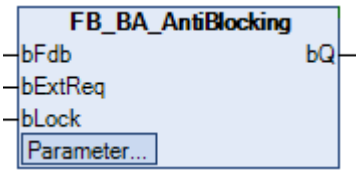
Name	Type	Description
Anlg3Pnt	FB_BA_Anlg3Pnt [▶ 407]	The function block is the core of the template and intended for controlling the three-point actuator. It converts an analog positioning signal into the binary open/close commands.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.2 AntiBlocking

6.1.4.2.2.2.2.1 FB_BA_AntiBlocking



The template prevents blocking of pumps or actuators after prolonged idle periods by issuing a switch-on pulse.

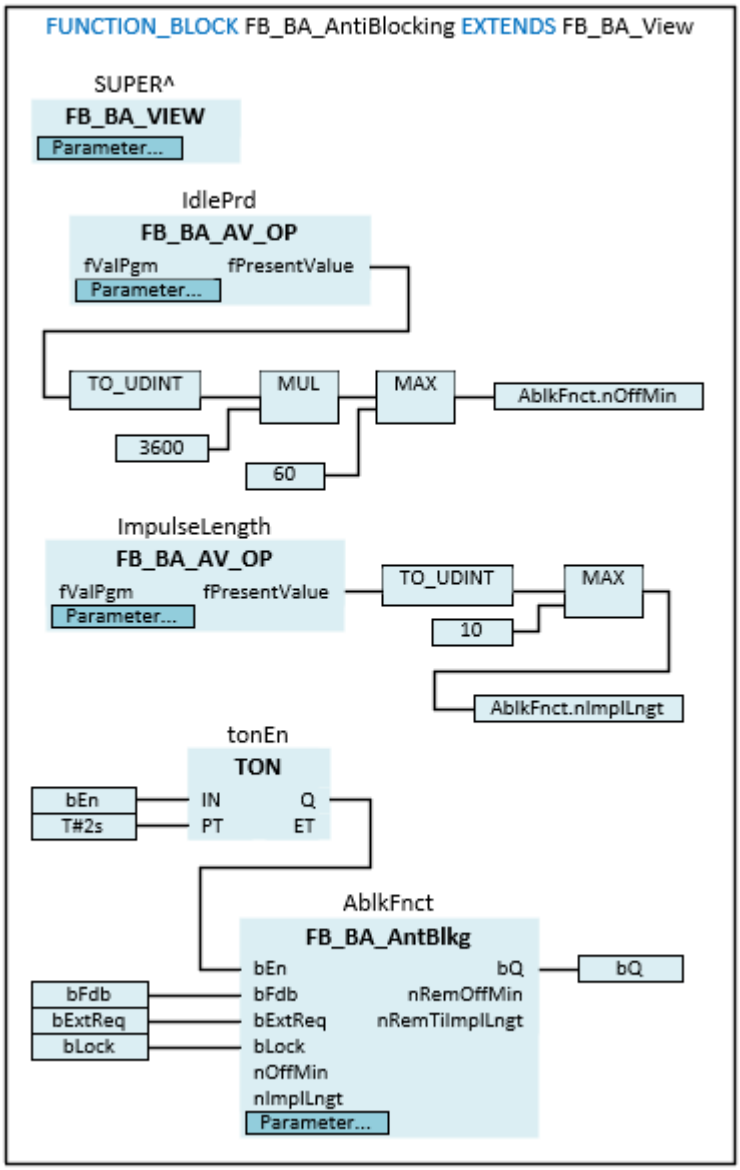
A pulse output generally only occurs if the function block FB_BA_AntBlkg is enabled at *bEn*.

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 anti-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.

Block diagram



Syntax

```
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
  AblkFnct          : 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 *E_BA_AntBlkgMode.eOffTime*.
bExtReq	BOOL	Active in the *E_BA_AntBlkgMode.eExternalRequest* operation mode. External request for a pulse, for example from a schedule. With a rising edge the anti-blocking protection pulse is started.
bLock	BOOL	Active in the operation modes *E_BA_AntBlkgMode.eExternalRequest* or *E_BA_AntBlkgMode.eOffTime*. 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 *bLock* is FALSE again. If *bLock* becomes TRUE during the output of an anti-blocking protection pulse, then the anti-blocking protection pulse is interrupted. After bLock is FALSE again, the anti-lock protection pulse is restarted.

Outputs

Name	Type	Description
bQ	BOOL	Output of the anti-lock protection pulse.

Inputs CONSTANT

Name	Type	Description
IdlePrd	FB_BA_AV_Op [▶ 202]	AV object for input of the maximum pump standstill duration until an anti-blocking protection pulse is issued.
ImpulseLength	FB_BA_AV_Op [▶ 202]	A rising edge at this input switches the output value *fOut* to the input value *fIn*.

Inputs CONSTANT PERSISTENT

Name	Type	Description
bEn	BOOL	General enable of the template. If *bEn* is FALSE, the message output *bQ* is also FALSE.

Variables

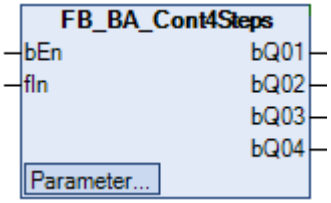
Name	Type	Description
bAblkFnct	FB_BA_AntBlkg [▶ 409]	The function block *AblkFnct* for the output of an anti-blocking protection pulse is the core of this template.
tonEn	TON	Start-up delay of the function after the controller has started up.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.3 ContinuousSteps

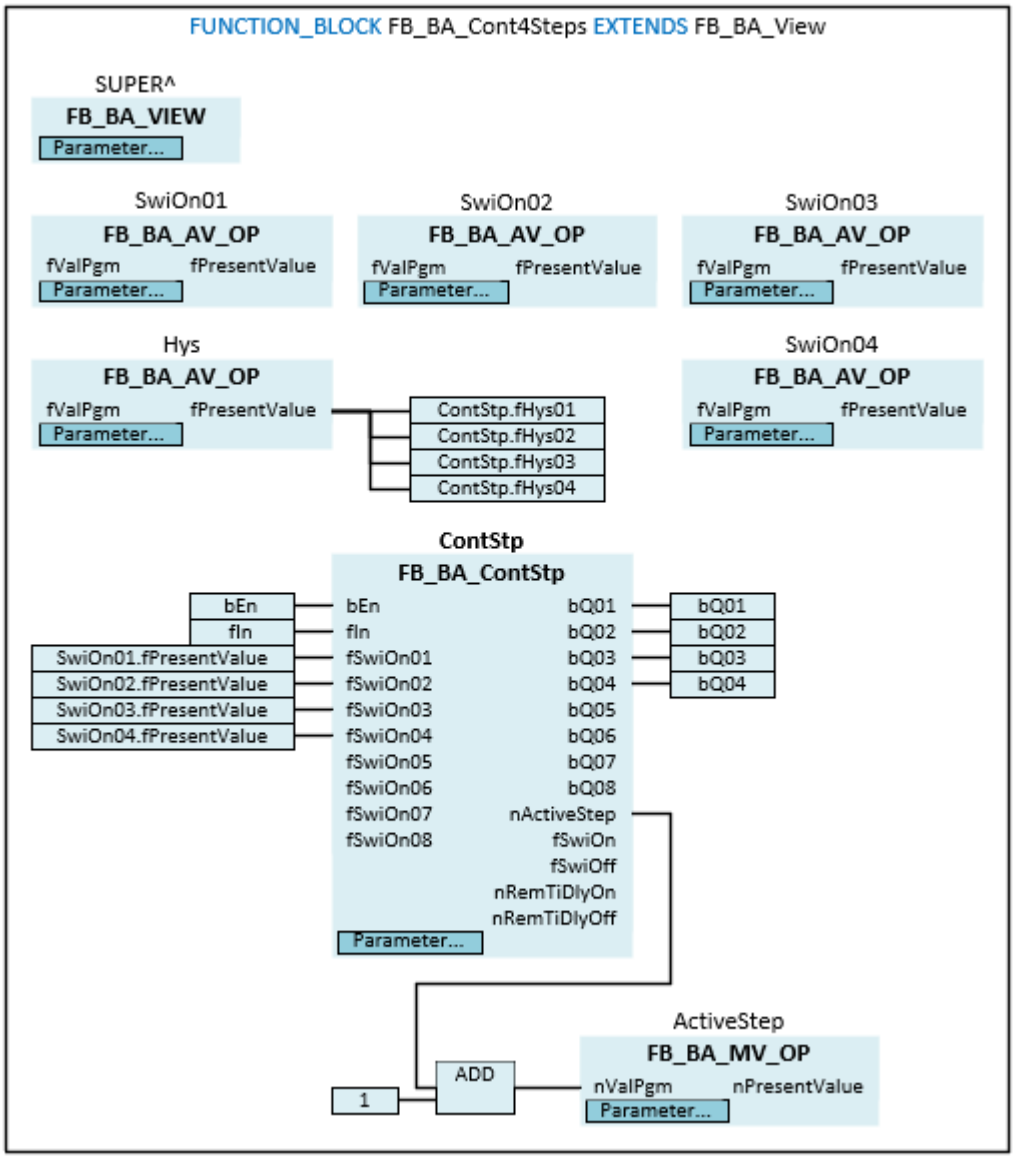
6.1.4.2.2.2.3.1 FB_BA_Cont4Steps



The template determines the resulting control steps of a 4-level aggregate, depending on the continuous input signal *fIn*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 *bEn* is FALSE, all message outputs *bQ0x* 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 [▶ 202]	AV object for entering the switch-on point step 01.
SwitchOn02	FB_BA_AV_Op [▶ 202]	AV object for entering the switch-on point step 02.
SwitchOn03	FB_BA_AV_Op [▶ 202]	AV object for entering the switch-on point step 03.
SwitchOn04	FB_BA_AV_Op [▶ 202]	AV object for entering the switch-on point step 04.
Hys	FB_BA_AV_Op [▶ 202]	AV object for entering the hysteresis for the switch-on points.
ActiveStep	FB_BA_AV_Op [▶ 202]	MV object for displaying how many steps are switched on.

Variables

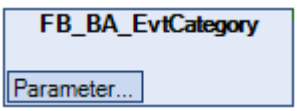
Name	Type	Description
ContStp	FB_BA_ContStp [▶ 417]	The function block determines the resulting control steps of a multi-stage aggregate depending on the continuous input signal *fIn* and is the core of this template

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 EventClasses

6.1.4.2.2.2.4.1 FB_BA_EvtCategory



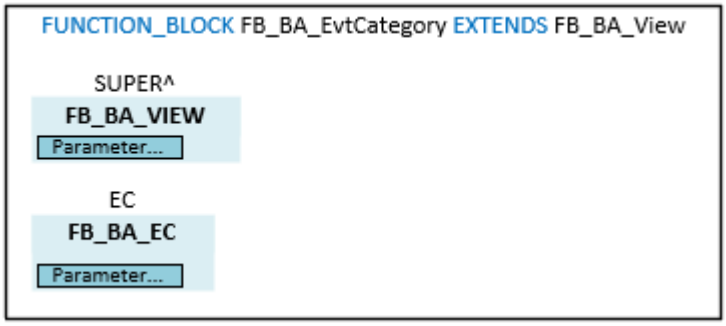
The template contains a Notification Class Object.

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.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_EvtCategory EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  EC    : FB_BA_EC;
END_VAR
```

Inputs CONSTANT

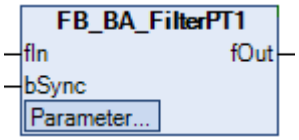
Name	Type	Description
EC	FB_BA_EC [▶ 218]	Notification Class Object.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.5 Filter

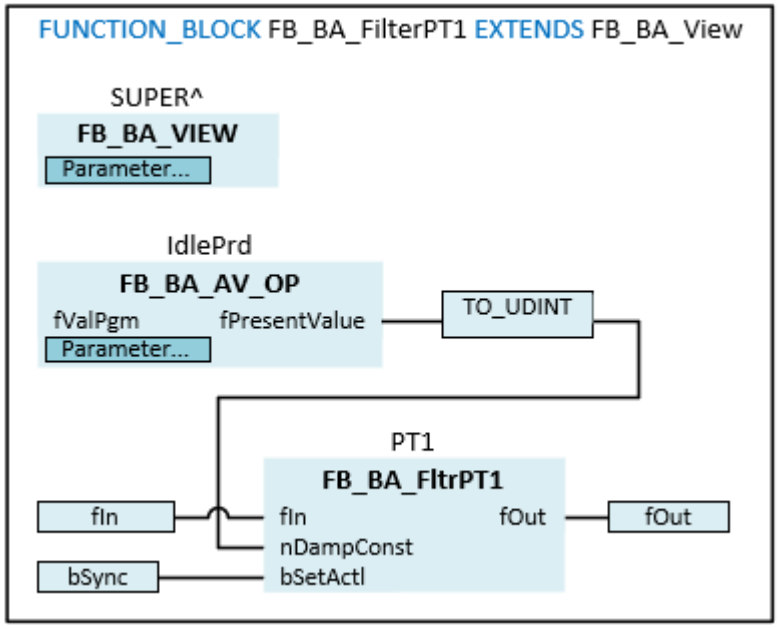
6.1.4.2.2.2.5.1 FB_BA_FilterPT1



First order filter.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_FilterPT1 EXTENDS FB_BA_View
VAR_INPUT
  fIn                : 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 *fOut* to the input value *fIn*.

Outputs

Name	Type	Description
fOut	REAL	Attenuated output signal.

Inputs CONSTANT

Name	Type	Description
DampingConstant	FB_BA_AV_Op [▶ 202]	AV object for entering filter time constant.

Variables

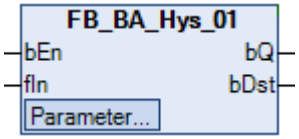
Name	Type	Description
PT1	FB_BA_FltrPT1	Notification Class Object.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.6 Hysteresis

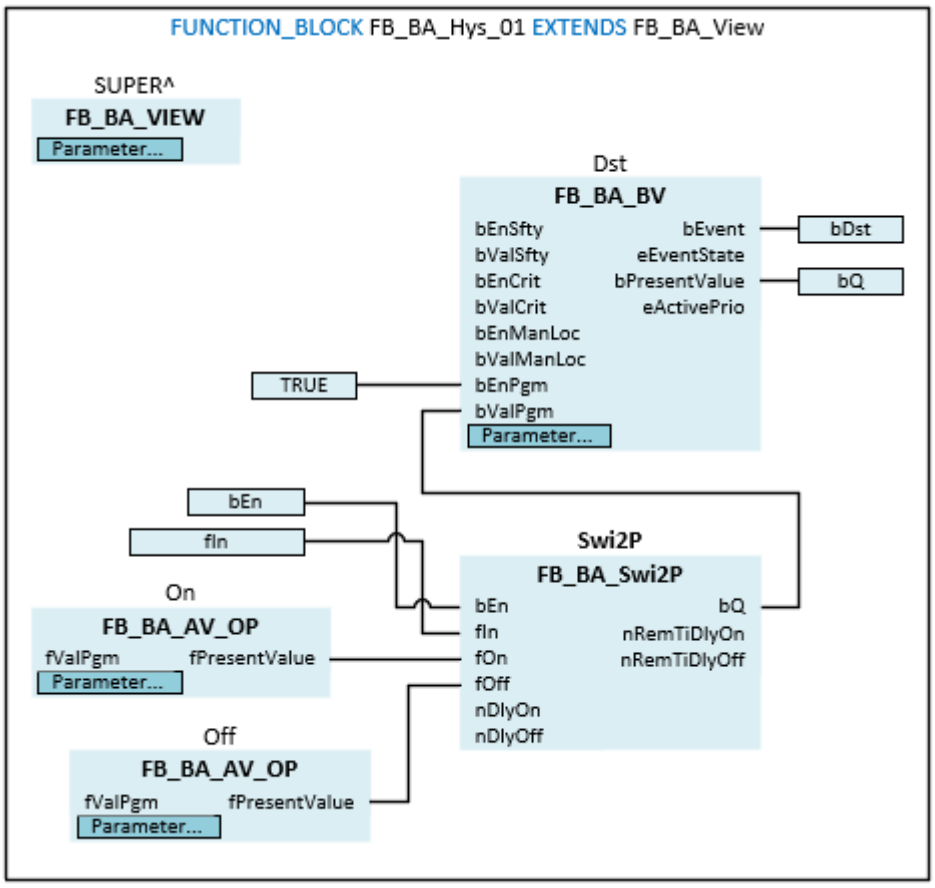
6.1.4.2.2.2.6.1 FB_BA_Hys_01



The template represents a hysteresis function with fixed switching points.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Hys_01 EXTENDS FB_BA_View
VAR_INPUT
  bEn       : BOOL;
  fIn       : 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. *bDst* is only active if the property *bEventDetectionEnable* 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 [▶ 202]	AV object for input of the upper limit of the hysteresis function.
Off	FB_BA_AV_Op [▶ 202]	AV object for input of the lower limit of the hysteresis function.
Q	FB_BA_BV [▶ 213]	The binary object indicates the current state of the hysteresis function.

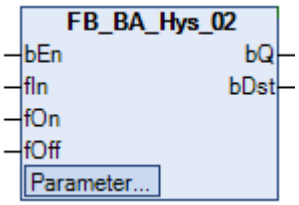
Variables

Name	Type	Description
Swi2P	FB_BA_Swi2P	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

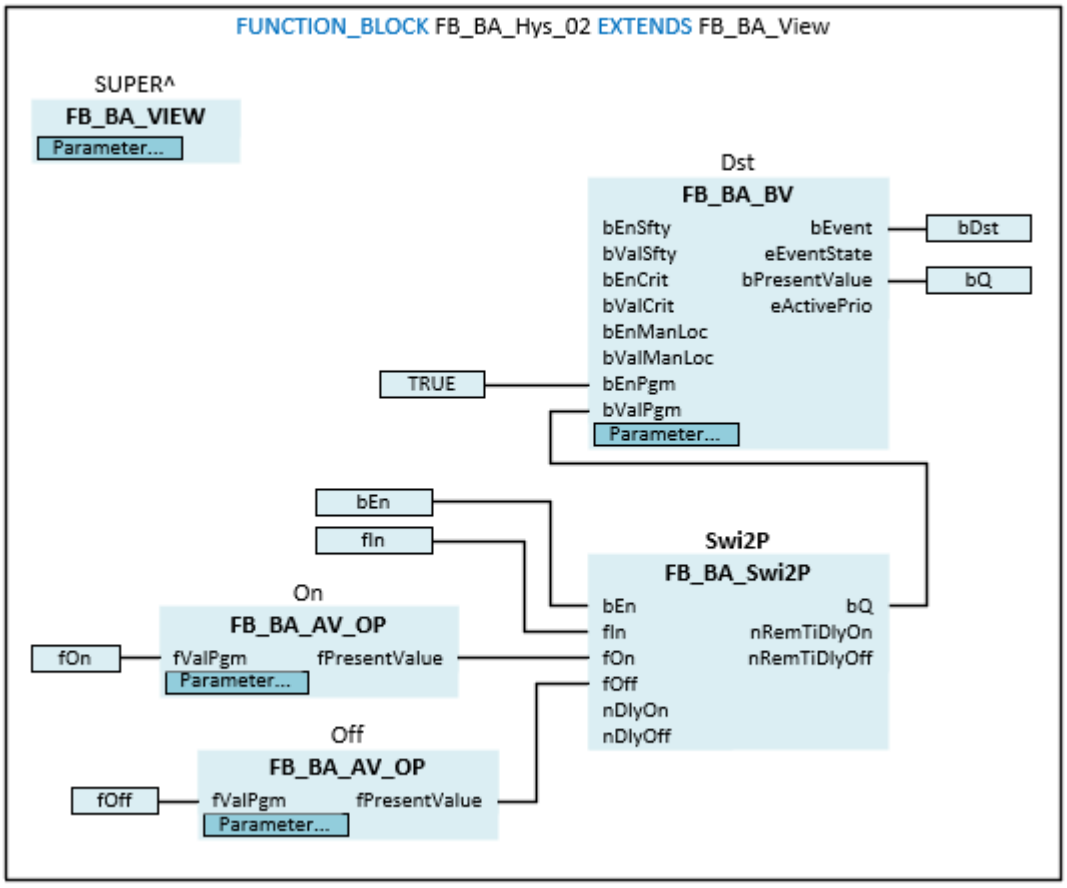
6.1.4.2.2.6.2 FB_BA_Hys_02



The template represents a sliding limit value monitoring with two switching points.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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. *bDst* is only active if the property *bEventDetectionEnable* 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 [▶ 202]	AV object for input of the upper limit of the hysteresis function.
Off	FB_BA_AV_Op [▶ 202]	AV object for input of the lower limit of the hysteresis function.
Q	FB_BA_BV [▶ 213]	The binary object indicates the current state of the hysteresis function.

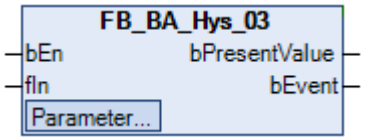
Variables

Name	Type	Description
Swi2P	FB_BA_Swi2P	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

6.1.4.2.2.6.3 FB_BA_Hys_03

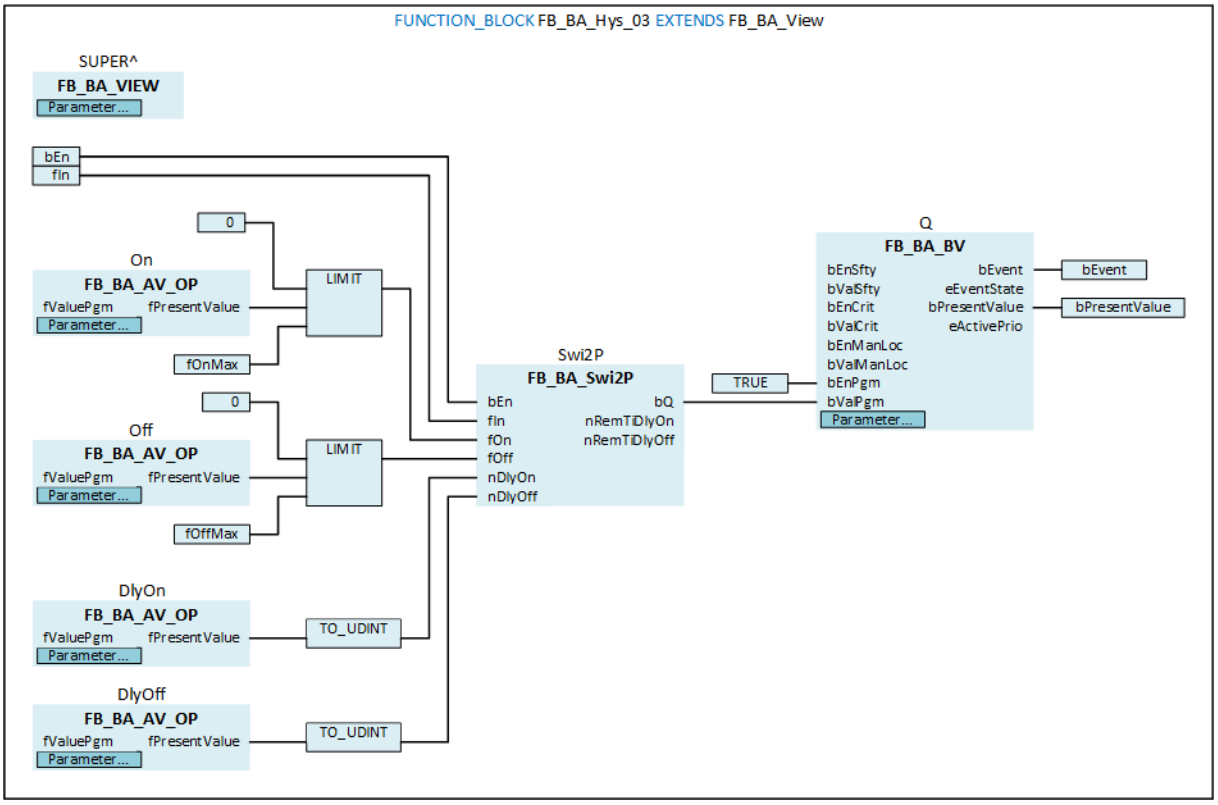


The template represents a sliding limit value monitoring with two switching points which, in contrast to FB_BA_Hys_02 [▶ 1006], are subject to switch-on and switch-off delays.

The switch-on and switch-off thresholds as well as the time delays are entered via analog objects.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Hys_03 EXTENDS FB_BA_View
VAR_INPUT
  bEn                           : BOOL;
  fIn                           : REAL;
END_VAR
VAR_OUTPUT
  bPresentValue                 : BOOL;
  bEvent                        : BOOL;
END_VAR
VAR_INPUT CONSTANT
  On                            : FB_BA_AV_Op;
  Off                           : FB_BA_AV_Op;
  DlyOn                         : FB_BA_AV_Op;
  DlyOff                        : 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	Process value

Outputs

Name	Type	Description
bPresentValue	BOOL	Output of the current state of the hysteresis function.
bEvent	BOOL	Display of a fault or the BV object is active. *bEvent* is only active if the property *bEventDetectionEnable* 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 [▶ 202]	AV object for input of the upper limit of the hysteresis function.
Off	FB_BA_AV_Op [▶ 202]	AV object for input of the lower limit of the hysteresis function.
DlyOn	FB_BA_AV_Op [▶ 202]	AV object for entering the switch-on delay in seconds.
DlyOff	FB_BA_AV_Op [▶ 202]	AV object for entering the switch-off delay in seconds.
Q	FB_BA_BV [▶ 213]	The binary object indicates the current state of the hysteresis function.

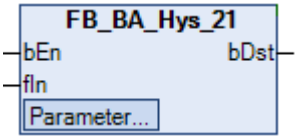
Variables

Name	Type	Description
Swi2P	FB_BA_Swi2P	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

6.1.4.2.2.2.6.4 FB_BA_Hys_21



The template represents two hysteresis functions with fixed switching points.

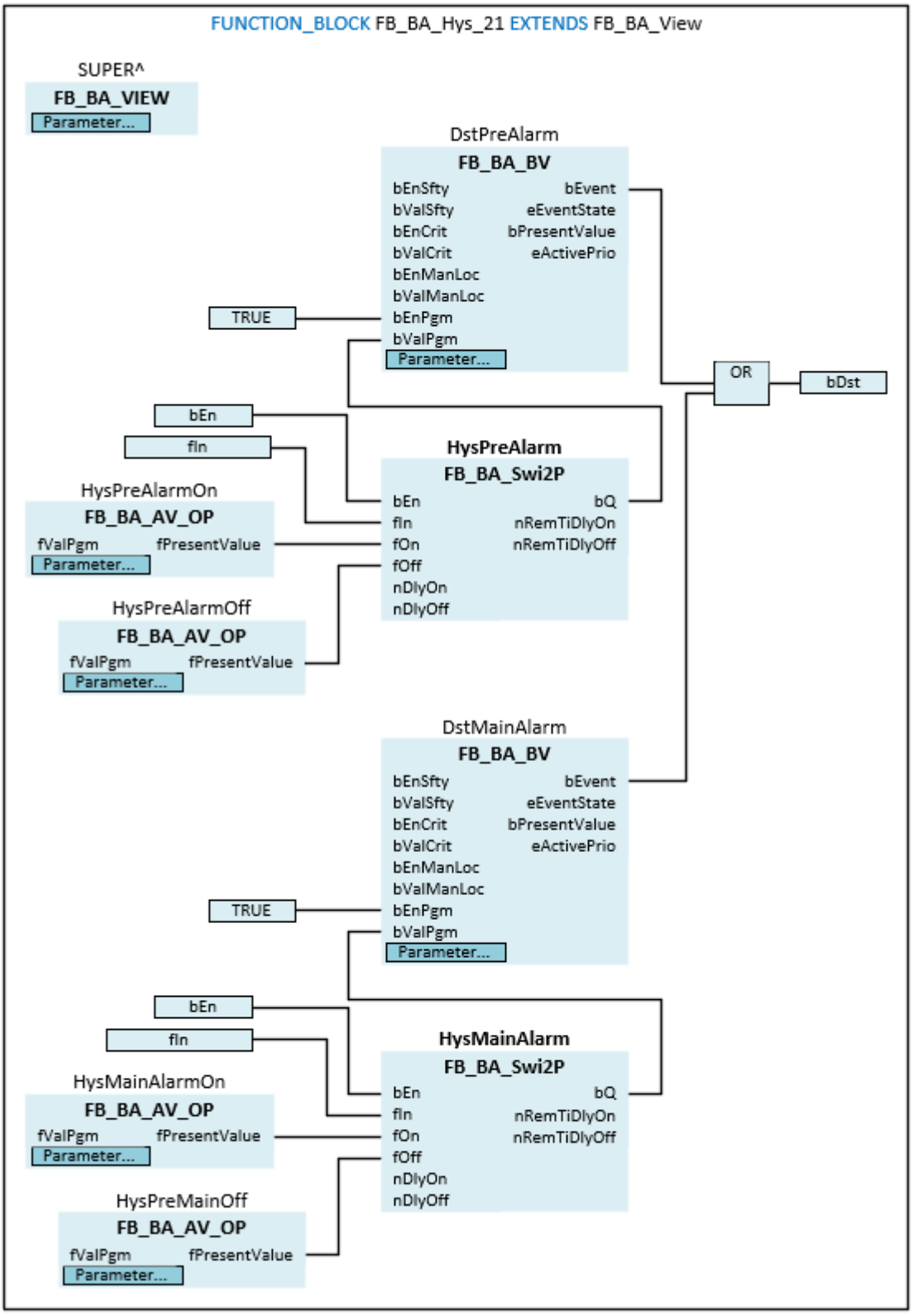
It is used to monitor analog values, such as an air filter.

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.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Hys_21 EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
  fIn                : 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. *bDst* is only active if the property *bEventDetectionEnable* 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 [▶ 202]	AV object for entering the upper limit value of the hysteresis function *HysPreAlarm*.
HysPreAlarmOff	FB_BA_AV_Op [▶ 202]	AV object for entering the lower limit value of the hysteresis function *HysPreAlarm*.
HysMainAlarmOn	FB_BA_AV_Op [▶ 202]	AV object for entering the upper limit value of the hysteresis function *HysMainAlarm*.
HysMainAlarmOff	FB_BA_AV_Op [▶ 202]	AV object for entering the lower limit value of the hysteresis function *HysMainAlarm*.
DstPreAlarm	FB_BA_BV [▶ 213]	The binary object is used to indicate the pre-alarm of the hysteresis function *HysPreAlarm*. A message is triggered via Intrinsic Reporting.
DstMainAlarm	FB_BA_BV [▶ 213]	The binary object is used to indicate the main alarm of the hysteresis function *HysMainAlarm*. A message is triggered via Intrinsic Reporting. A fault that results in plant shutdown must be parameterized in the FB_init method via *eEnPlantLock*.

Variables

Name	Type	Description
HysPreAlarm	FB_BA_Swi2P	The function block is the core of the hysteresis function pre-alarm.
HysMainAlarm	FB_BA_Swi2P	The function block is the core of the hysteresis function main alarm.

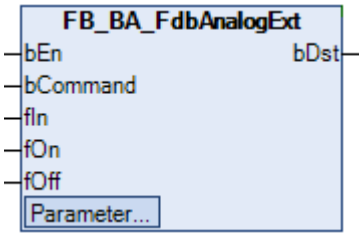
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.7 Monitoring

Presence monitoring.

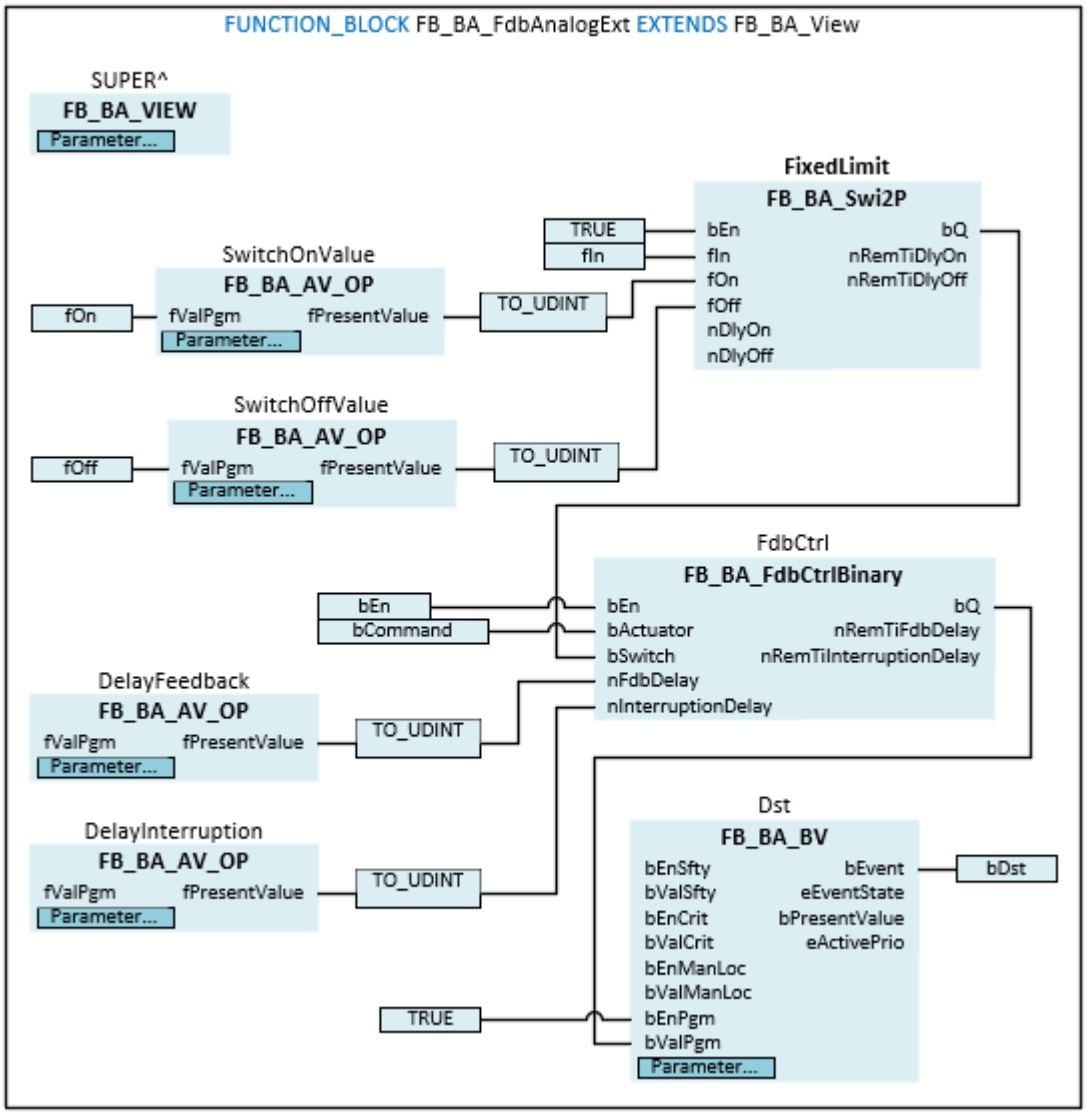
6.1.4.2.2.2.7.1 FB_BA_FdbAnalogExt



The template is used to monitor analog values with dynamic switch values, e.g. for differential pressure monitoring of a fan.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_FdbAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  bCommand              : BOOL;
  fIn                   : REAL;
  fOn                   : REAL := 10.0;
  fOff                  : REAL := 2.0;
END_VAR
VAR_OUTPUT
  bDst                  : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SwitchOnValue         : FB_BA_AV_Op;
  SwitchOffValue        : FB_BA_AV_Op;
  DelayFeedback         : FB_BA_AV_Op;
  DelayInterruption     : FB_BA_AV_Op;
  Dst                   : FB_BA_BV;
END_VAR
VAR
  FixedLimit            : FB_BA_Swi2P;
  FdbCtrl               : FB_BA_FdbCtrlBinary;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
bCommand	BOOL	The switching actuator output of the aggregate to be monitored is connected to the input.
fIn	REAL	The analog value to be monitored is connected to the input, e.g. a differential pressure sensor.
fOn	REAL	The switch-on value for *FixedLimit* is connected to the input. In the case of differential pressure monitoring, a pressure value must be selected here which must not be undershot during the running process.
fOff	REAL	The switch-off value for *FixedLimit* is connected to the input. This value must be just below the switch-on value *fOn*.

Outputs

Name	Type	Description
bDst	BOOL	Binary object for displaying the fault.

Inputs CONSTANT

Name	Type	Description
SwitchOnValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the switch-on value for *fOn*.
SwitchOffValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the switch-off value for *fOff*.
DelayFeedback	FB_BA_AV_Op [▶ 202]	Analog value object for entering the time delay of the "Aggregate ready for operation" information. In the case of differential pressure monitoring, a time delay must be specified here after which it can be assumed that the system has built up the required differential pressure.
DelayInterruption	FB_BA_AV_Op [▶ 202]	Analog value object for entering the delay time to trigger a fault message. The message from the differential pressure sensor can be delayed, in order to buffer pressure fluctuations.
Dst	FB_BA_BV [▶ 213]	Binary object for displaying the fault.

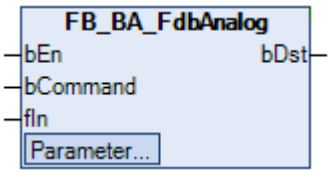
Variables

Name	Type	Description
FixedLimit	FB_BA_Swi2P	Conversion of the analog value *fIn* into a binary switching signal for *FdbCtrl*.
FdbCtrl	FB_BA_FdbCtrlBinary [▶ 467]	Monitoring of the binary feedback.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

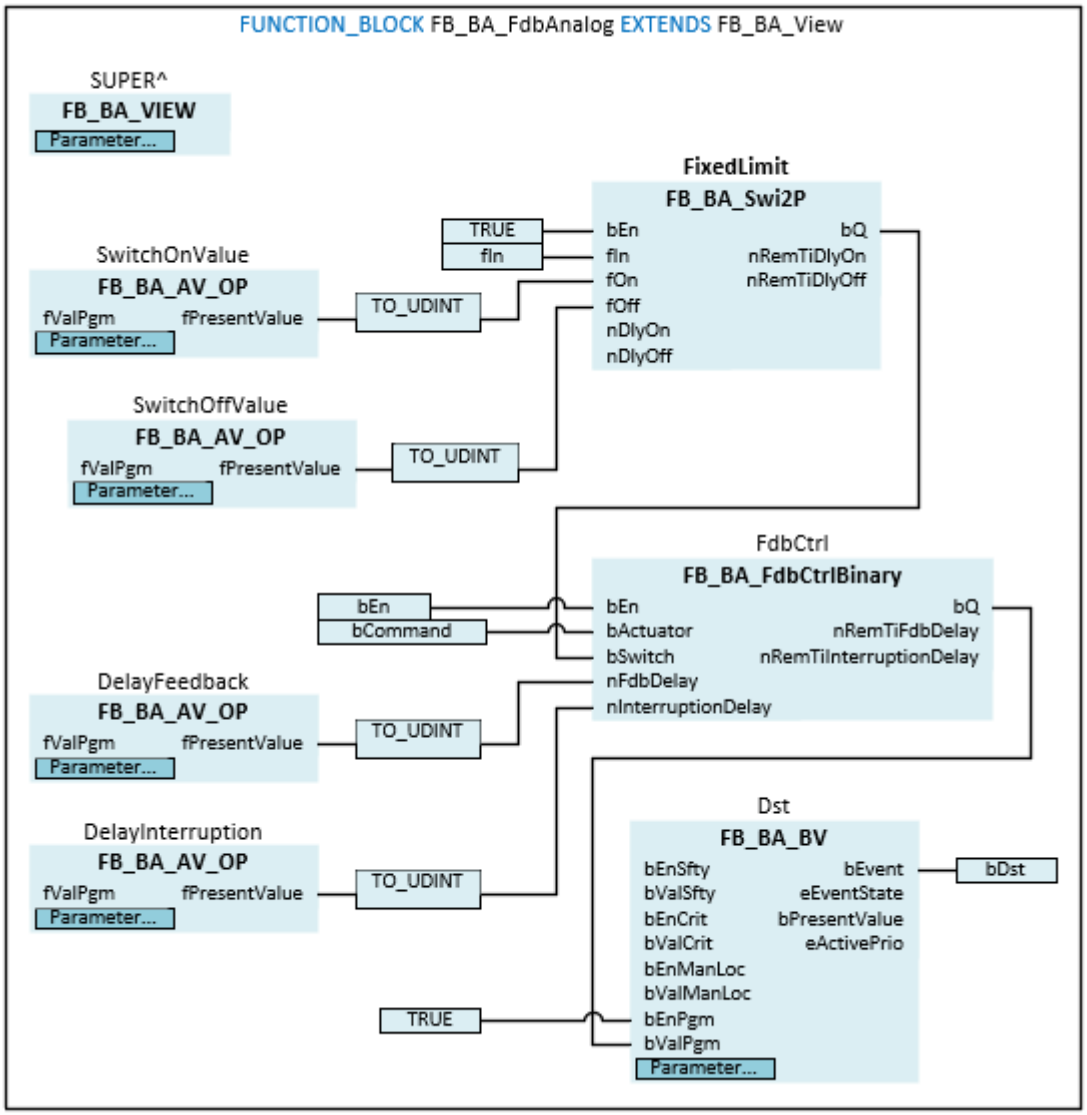
6.1.4.2.2.7.2 FB_BA_FdbAnalog



The template is used to monitor analog values with fixed switch values, such as differential pressure monitoring of a fan.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_FdbAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEn                              : BOOL;
  bCommand                         : BOOL;
  fIn                              : REAL;
END_VAR
VAR_OUTPUT
  bDst                             : BOOL;
END_VAR
```

```
VAR_INPUT CONSTANT
  SwitchOnValue                 : FB_BA_AV_Op;
  SwitchOffValue                : FB_BA_AV_Op;
  DelayFeedback                 : FB_BA_AV_Op;
  DelayInterruption             : FB_BA_AV_Op;
  Dst                           : FB_BA_BV;
END_VAR
VAR
  FixedLimit                    : FB_BA_Swi2P;
  FdbCtrl                       : FB_BA_FdbCtrlBinary;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
bCommand	BOOL	The switching actuator output of the aggregate to be monitored is connected to the input.
fIn	REAL	The analog value to be monitored is connected to the input, e.g. a differential pressure sensor.

Outputs

Name	Type	Description
bDst	BOOL	Binary object for displaying the fault.

Inputs CONSTANT

Name	Type	Description
SwitchOnValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the switch-on value for *FixedLimit*. In the case of differential pressure monitoring, a pressure value must be selected here which must not be undershot during the running process.
SwitchOffValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the switch-off value for *FixedLimit*. This value must be just below the switch-on value *SwitchOnValue*.
DelayFeedback	FB_BA_AV_Op [▶ 202]	Analog value object for entering the time delay of the "Aggregate ready for operation" information. In the case of differential pressure monitoring, a time delay must be specified here after which it can be assumed that the system has built up the required differential pressure.
DelayInterruption	FB_BA_AV_Op [▶ 202]	Analog value object for entering the delay time to trigger a fault message. The message from the differential pressure sensor can be delayed, in order to buffer pressure fluctuations.
Dst	FB_BA_BV [▶ 213]	Binary object for displaying the fault.

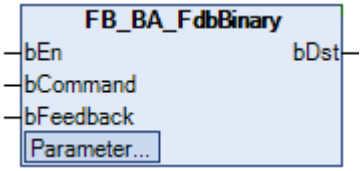
Variables

Name	Type	Description
FixedLimit	FB_BA_Swi2P	Conversion of the analog value *fIn* into a binary switching signal for *FdbCtrl*.
FdbCtrl	FB_BA_FdbCtrlBinary [▶ 467]	Monitoring of the binary feedback.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

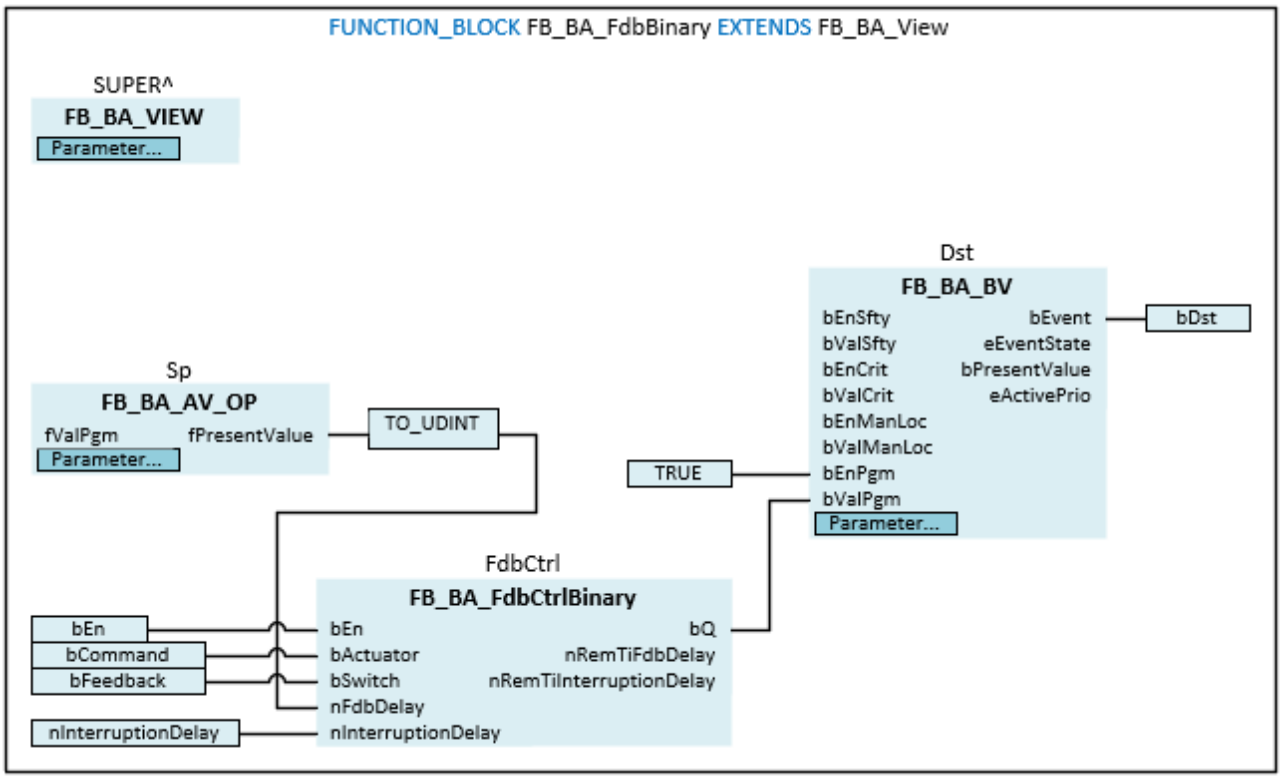
6.1.4.2.2.7.3 FB_BA_FdbBinary



The template is used to monitor binary feedback signals such as the end positions of dampers or valves. However, it can also be used for differential pressure monitoring by means of a differential pressure monitor.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_FdbBinary EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  bCommand              : BOOL;
  bFeedback             : BOOL;
END_VAR
VAR_OUTPUT
  bDst                  : BOOL;
END_VAR
VAR_INPUT CONSTANT
  DelayFeedback         : FB_BA_AV_Op;
  Dst                   : FB_BA_BV;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterUnit':= 's'}
```

```
  nInterruptionDelay    : UDINT := 1;
END_VAR
VAR
  FdbCtrl               : FB_BA_FdbCtrlBinary;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
bCommand	BOOL	The switching actuator output of the aggregate to be monitored is connected to the input.
bFeedback	BOOL	The feedback signal of the aggregate to be monitored is connected to the input, e.g. a differential pressure monitor, flow monitor or limit switch.

Outputs

Name	Type	Description
bDst	BOOL	Binary object for displaying the fault.

Inputs CONSTANT

Name	Type	Description
DelayFeedback	FB_BA_AV_Op [▶ 202]	Analog value object for entering the time delay of the feedback. The travel time of the actuator can be used. In the case of differential pressure monitoring, a time delay must be specified here until the system has to built up the required differential pressure.
Dst	FB_BA_BV [▶ 213]	Binary object for displaying the fault.

Inputs CONSTANT PERSISTENT

Name	Type	Description
nInterruptionDelay	UDINT	Analog value object for entering the switch-on value for *FixedLimit*. In the case of differential pressure monitoring, a pressure value must be selected here which must not be undershot during the running process.

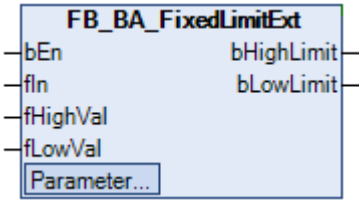
Variables

Name	Type	Description
FdbCtrl	FB_BA_FdbCtrlBinary [▶ 467]	Monitoring the binary feedback of the actuators.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.7.4 FB_BA_FixedLimitExt



The template represents a limit switch with dynamic limit values.

A tolerance range is defined around the value *fIn* to be monitored.

The tolerance range results from a high limit value *HighLimitValue* and a low limit value *LowLimitValue*.

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 *TiDly*.

The binary object *HighLimitOn* is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.

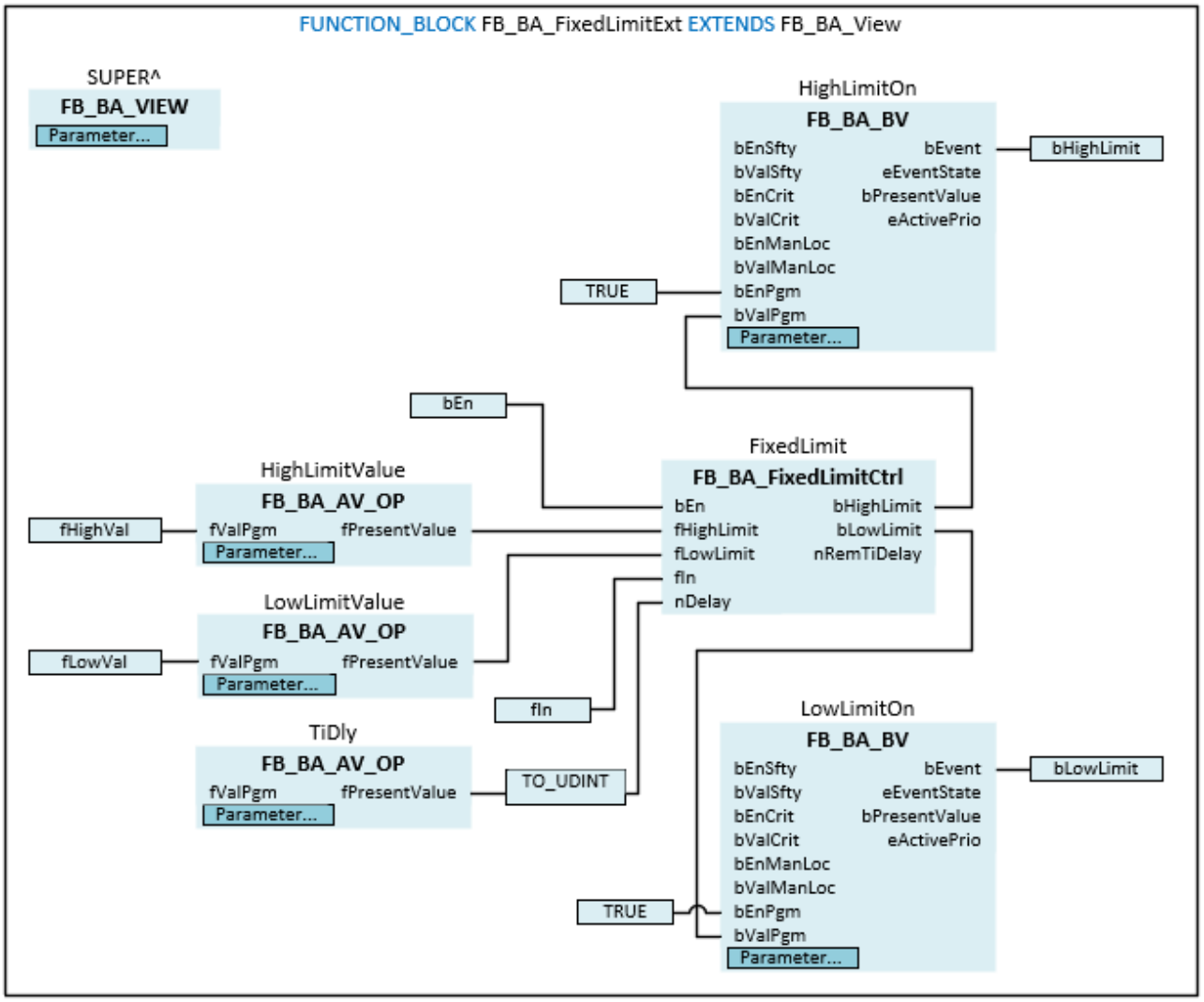
If the value *fIn* falls below the lower limit value of the tolerance range, then the output *bLowLimit* is set.

A response delay of the output *bLowLimit* can be parameterized with the time variable *TiDly*.

The binary object *LowLimitOn* is used to display *bLowLimit* and can be used as a fault message object by changing the parameterization.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_FixedLimitExt EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
  fIn                : REAL;
  fHighVal           : REAL;
  fLowVal            : REAL;
END_VAR
VAR_OUTPUT
  bHighLimit         : BOOL;
  bLowLimit          : BOOL;
END_VAR
VAR_INPUT CONSTANT
  HighLimitValue     : FB_BA_AV_Op;
  LowLimitValue      : FB_BA_AV_Op;
  TiDly              : FB_BA_AV_Op;
  HighLimitOn        : FB_BA_BV;
  LowLimitOn         : FB_BA_BV;
END_VAR
VAR
  FixedLimit         : FB_BA_FixedLimitCtrl;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
fIn	REAL	The analog value to be monitored is connected to the input, e.g. a differential pressure sensor.
fHighVal	REAL	The upper limit value of the tolerance range is connected to the input.
fLowVal	REAL	The lower limit value of the tolerance range is connected to the input.

Outputs

Name	Type	Description
bHighLimit	BOOL	The output indicates that the upper limit value of the tolerance range has been exceeded.
bLowLimit	BOOL	The output indicates that the value has fallen below the lower limit of the tolerance range.

Inputs CONSTANT

Name	Type	Description
HighLimitValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the upper limit value of the tolerance range.
LowLimitValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the lower limit value of the tolerance range.
TiDly	FB_BA_AV_Op [▶ 202]	Analog value object for entering the response delay of the outputs *bHighLimit* and *bLowLimit*.
HighLimitOn	FB_BA_BV [▶ 213]	The binary object is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.
LowLimitOn	FB_BA_BV [▶ 213]	The binary object is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.

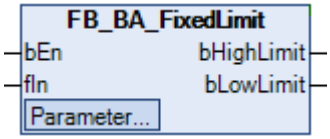
Variables

Name	Type	Description
FixedLimit	FB_BA_FixedLimitCtrl [▶ 468]	Core of the template.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.7.5 FB_BA_FixedLimit



The template represents a limit switch with fixed limits.

A tolerance range is defined around the value *fIn* to be monitored.

The tolerance range results from a high limit value *HighLimitValue* and a low limit value *LowLimitValue*.

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 *TiDly*.

The binary object *HighLimitOn* is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.

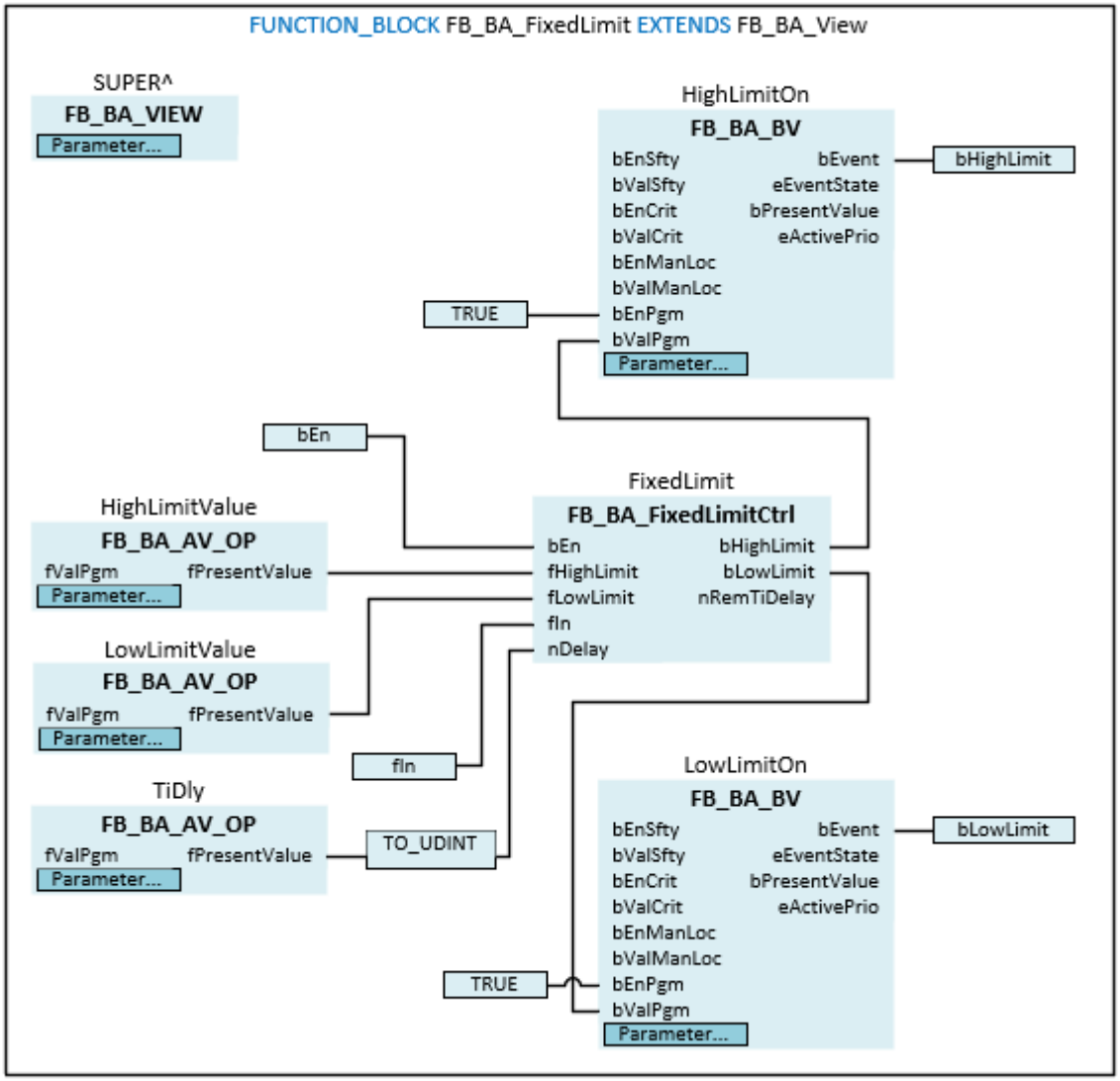
If the value *fIn* falls below the lower limit value of the tolerance range, then the output *bLowLimit* is set.

A response delay of the output *bLowLimit* and is to be parameterized with the time variable *TiDly*.

The binary object *LowLimitOn* is used to display *bLowLimit* and can be used as a fault message object by changing the parameterization.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_FixedLimit EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
```

```
  fIn               : REAL;
END_VAR
VAR_OUTPUT
  bHighLimit        : BOOL;
  bLowLimit         : BOOL;
END_VAR
VAR_INPUT CONSTANT
  HighLimitValue    : FB_BA_AV_Op;
  LowLimitValue     : FB_BA_AV_Op;
  TiDly             : FB_BA_AV_Op;
  HighLimitOn       : FB_BA_BV;
  LowLimitOn        : FB_BA_BV;
END_VAR
VAR
  FixedLimit        : FB_BA_FixedLimitCtrl;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
fIn	REAL	The analog value to be monitored is connected to the input, e.g. a differential pressure sensor.

Outputs

Name	Type	Description
bHighLimit	BOOL	The output indicates that the upper limit value of the tolerance range has been exceeded.
bLowLimit	BOOL	The output indicates that the value has fallen below the lower limit of the tolerance range.

Inputs CONSTANT

Name	Type	Description
HighLimitValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the upper limit value of the tolerance range.
LowLimitValue	FB_BA_AV_Op [▶ 202]	Analog value object for entering the lower limit value of the tolerance range.
TiDly	FB_BA_AV_Op [▶ 202]	Analog value object for entering the response delay of the outputs *bHighLimit* and *bLowLimit*.
HighLimitOn	FB_BA_BV [▶ 213]	The binary object is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.
LowLimitOn	FB_BA_BV [▶ 213]	The binary object is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.

Variables

Name	Type	Description
FixedLimit	FB_BA_FixedLimitCtrl [▶ 468]	Core of the template.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.7.6 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.

Syntax

```
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 *bPresenceState* 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 *bRstSwi*.

Inputs CONSTANT PERSISTENT

Name	Type	Description
nDlyPrc	UDINT	Switch-off delay time [s]. Preset to 3600 in *FB_Init*.

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.

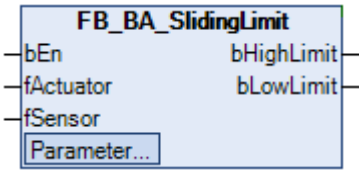
Variables

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

6.1.4.2.2.7.7 FB_BA_SlidingLimit



The template represents a sliding limit value monitoring.

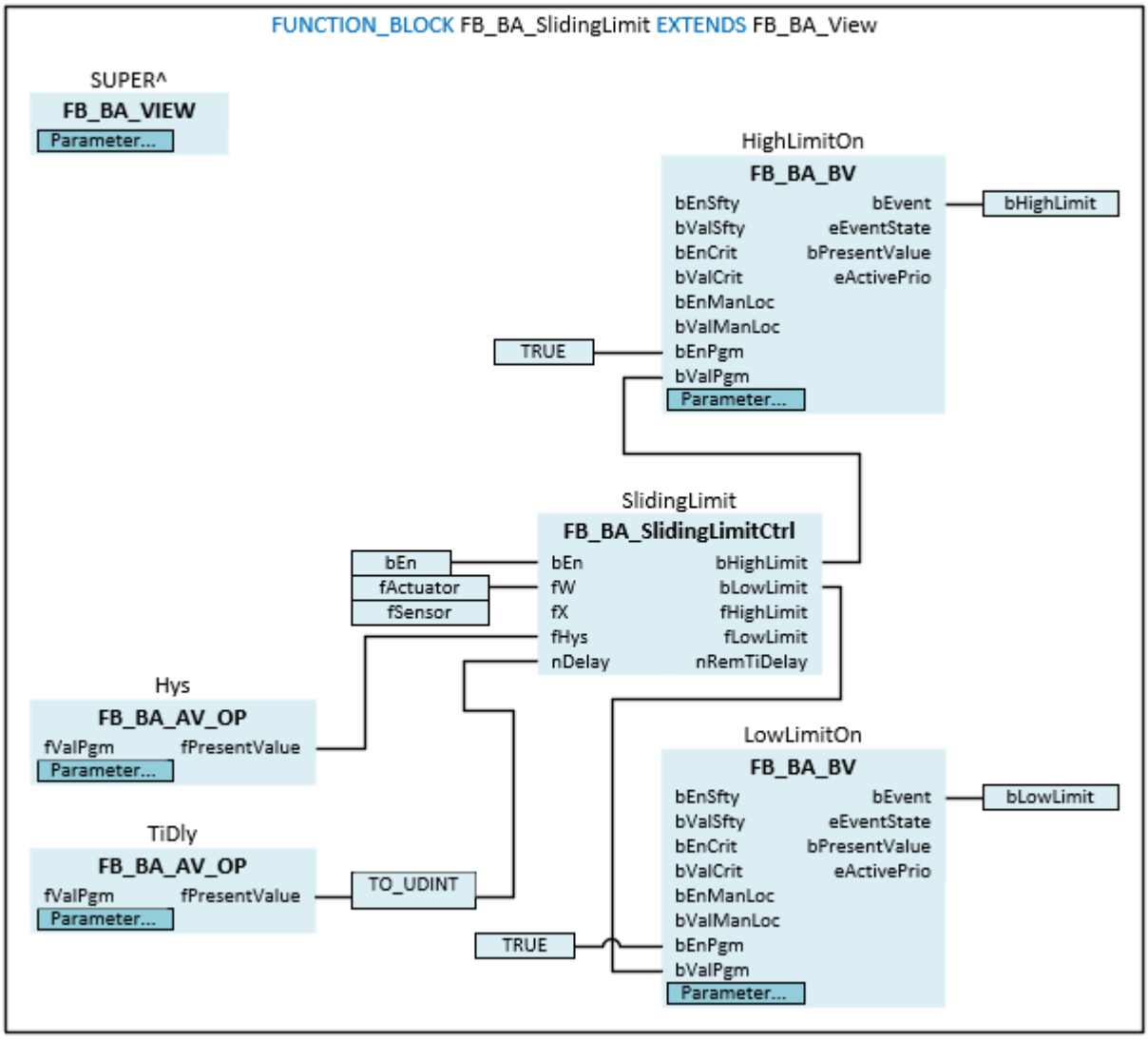
After the start, a check is made whether the actual value *fSensor* of the control is within the tolerance range between the lower limit value *fLowLimit* and the upper limit value *fHighLimit* of the function block *SlidingLimit*. If the actual value is outside this tolerance range and the delay time *TiDly* has expired, one of the variables *bHighLimit* or *bLowLimit* is set depending on whether the tolerance range has been left.

The binary object *HighLimitOn* is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.

The binary object *LowLimitOn* is used to display *bLowLimit* and can be used as a fault message object by changing the parameterization.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_SlidingLimit EXTENDS FB_BA_View
VAR_INPUT
  bEn              : BOOL;
  fActuator        : REAL;
  fSensor          : REAL;
END_VAR
VAR_OUTPUT
  bHighLimit       : BOOL;
  bLowLimit        : BOOL;
END_VAR
VAR_INPUT CONSTANT
  Hys              : FB_BA_AV_Op;
  TiDly            : FB_BA_AV_Op;
  HighLimitOn      : FB_BA_BV;
  LowLimitOn       : FB_BA_BV;
END_VAR
VAR
  SlidingLimit     : FB_BA_SlidingLimitCtrl;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
fActuator	REAL	The base value of the tolerance range is connected to the input. This value can be, for example, a valve position.
fSensor	REAL	The analog value to be monitored is connected to the input. This value could, for example, be the feedback signal of a valve.

Outputs

Name	Type	Description
bHighLimit	BOOL	The output indicates that the upper limit value of the tolerance range has been exceeded.
bLowLimit	BOOL	The output indicates that the value has fallen below the lower limit of the tolerance range.

Inputs CONSTANT

Name	Type	Description
Hys	FB_BA_AV_Op [▶ 202]	Analog value object for entering the hysteresis of the tolerance range. Lower limit of the function block SlidingLimit: *fLowLimit* = *fActuator* - (*Hys* / 2) Upper limit of the function block SlidingLimit: *fHighLimit* = *fActuator* + (*Hys* / 2)
TiDly	FB_BA_AV_Op [▶ 202]	Analog value object for entering the response delay of the outputs *bHighLimit* and *bLowLimit*.
HighLimitOn	FB_BA_BV [▶ 213]	The binary object is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.
LowLimitOn	FB_BA_BV [▶ 213]	The binary object is used to display *bHighLimit* and can be used as a fault message object by changing the parameterization.

Variables

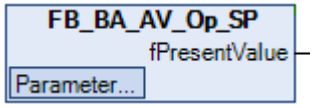
Name	Type	Description
SlidingLimit	FB_BA_SlidingLimitCtrl [▶ 469]	Core of the template.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.8 Object

6.1.4.2.2.2.8.1 FB_BA_AV_Op_SP



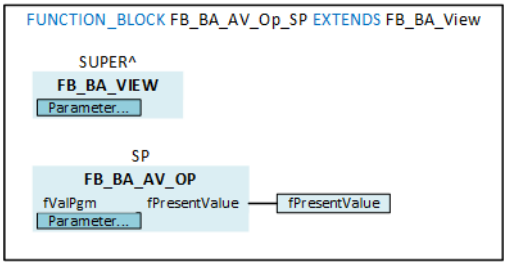
The template FB_BA_AV_Op_SP is used to enter a setpoint within a template at system level.

It is itself a shell that represents the aggregate level and sets the internal analog object at the functional level.

A label is predefined in *FB_Init* that describes the internal analog object [▶ 202] as a "setpoint"; the preselected unit is [°C].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_AV_Op_SP EXTENDS FB_BA_View
VAR_OUTPUT
  fPresentValue    : REAL;
END_VAR
VAR_INPUT CONSTANT
  SP               : FB_BA_AV_Op;
END_VAR
```

Outputs

Name	Type	Description
fPresentValue	REAL	Entered value.

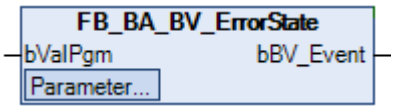
Inputs CONSTANT

Name	Type	Description
SP	FB_BA_AV_Op [▶ 202]	Analog value object for entering the setpoint.

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_BV_ErrorState



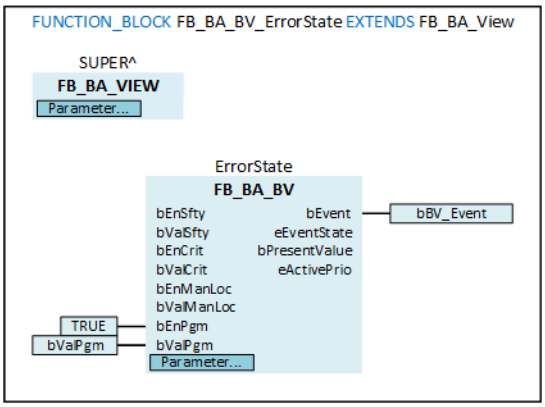
The template is an error state message within a template at system level.

It is itself a shell that represents the aggregate level and sets the internal binary object [▶ 213] at the functional level.

A label is predefined in *FB_Init* that describes the internal binary object as an "error".

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_BV_ErrorState EXTENDS FB_BA_View
VAR_INPUT
  bValPgm       : BOOL;
END_VAR
VAR_INPUT CONSTANT
  ErrorState    : FB_BA_BV;
END_VAR
VAR_OUTPUT
  bBV_Event     : BOOL;
END_VAR
```

VAR_INPUT CONSTANT

Name	Type	Description
bValPgm	BOOL	(Error) input from the program.

VAR_INPUT CONSTANT

Name	Type	Description
ErrorState	FB_BA_BV [▶ 213]	Binary value object for displaying the error message.

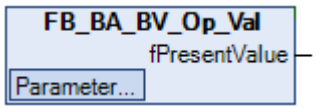
Outputs

Name	Type	Description
bBVEvent	BOOL	Binary state of the message.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.56	TF8040 | TwinCAT Building Automation from V5.8.0.0

6.1.4.2.2.8.3 FB_BA_BV_Op_Val



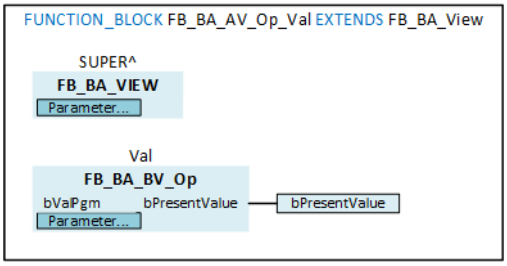
The template FB_BA_BV_OP_Val is a switch within a template at system level.

It is itself a shell that represents the aggregate level and sets the internal binary object [▶ 216] at the functional level.

A label is predefined in *FB_Init* that describes the internal binary object as a "push button", the preselected function is "latching" (E_BA_ToggleMode.eSwitch).

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_BV_OP_Val EXTENDS FB_BA_View
VAR_OUTPUT
  fPresentValue    : BOOL;
END_VAR
VAR_INPUT CONSTANT
  Val              : FB_BA_BV_Op;
END_VAR
```

Outputs

Name	Type	Description
fPresentValue	REAL	Binary state of the switch.

Inputs CONSTANT

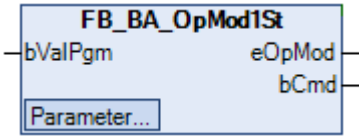
Name	Type	Description
Val	FB_BA_BV_Op [▶ 216]	Binary value object to represent the switch.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.9 OperatingMode

6.1.4.2.2.2.9.1 FB_BA_OpMod1St

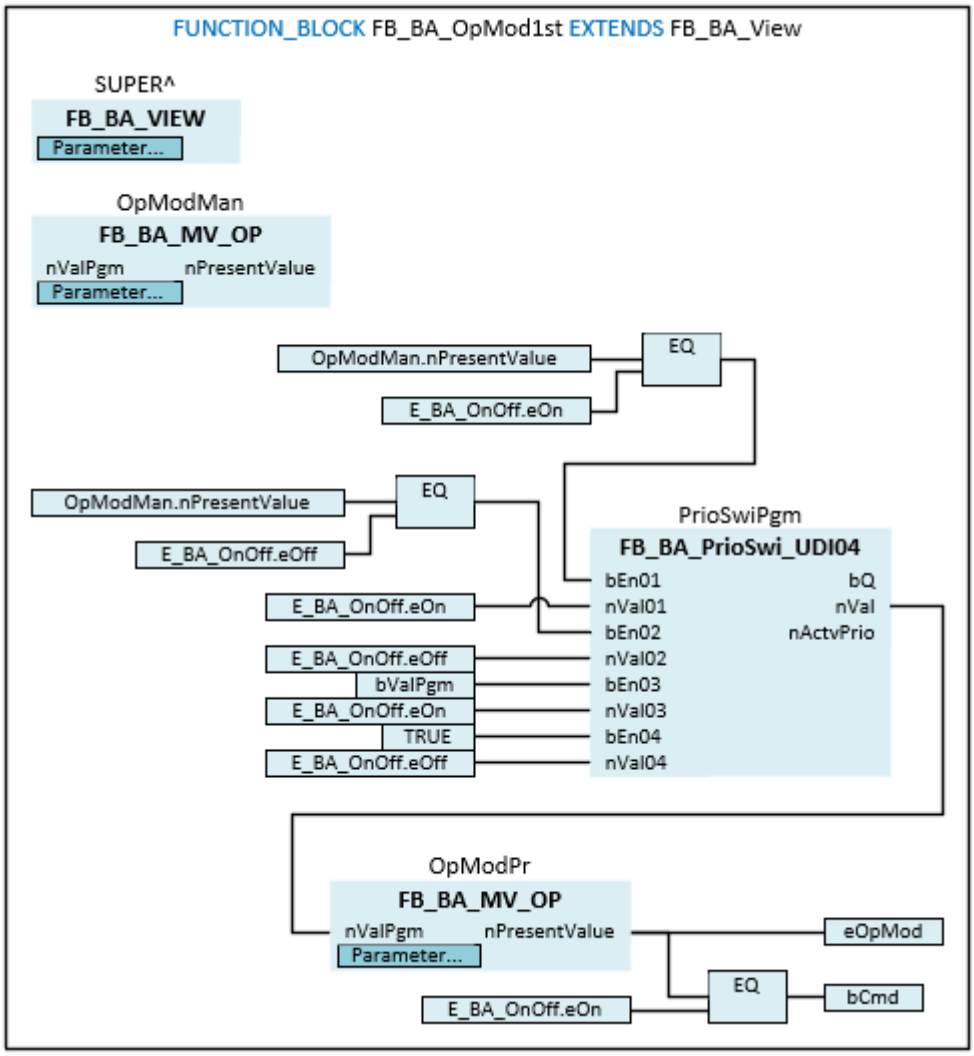


The template maps a single-stage plant selector switch with the operating modes Auto, Manual Off and Manual On.

A plant can be switched on or off via the input *bValPgm* in the operating mode Auto.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_OpMod1St EXTENDS FB_BA_View
VAR_INPUT
  bValPgm      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  OpModMan     : FB_BA_MV_Op;
  OpModPr      : FB_BA_MV_Op;
```

```
END_VAR
VAR_OUTPUT
  eOpMod       : E_BA_OnOff;
  bCmd         : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bValPgm	BOOL	A plant can be switched on or off via the input in the operating mode "Auto".

Inputs CONSTANT

Name	Type	Description
OpModMan	FB_BA_MV_Op [▶ 239]	The Multistate-Value object represents an operating mode switch with the operating modes "Auto", "Manual Off" and "Manual On".
OpModPr	FB_BA_MV_Op [▶ 239]	The Multistate-Value object indicates the state of the currently valid plant operating mode.

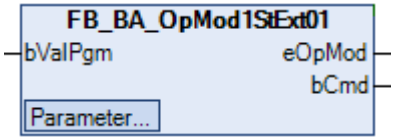
Outputs

Name	Type	Description
eOpMod	E_BA_OnOff [▶ 700]	Displays the state of the currently valid plant operating mode.
bCmd	BOOL	The output shows the state of the currently valid plant operating mode.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.9.2 FB_BA_OpMod1StExt01

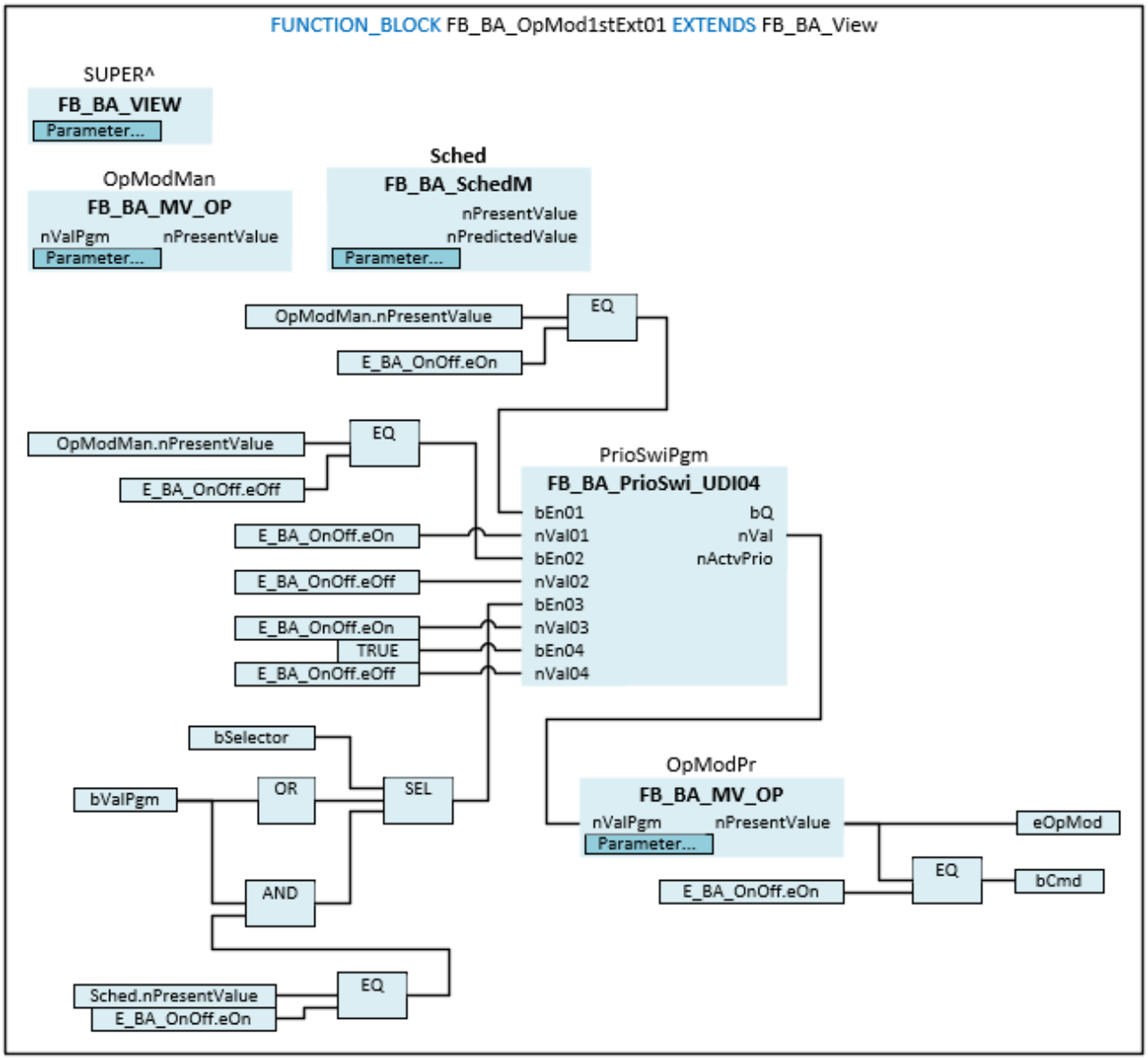


The template maps a single-stage plant selector switch with the operating modes Auto, Manual Off and Manual On.

The variable *bSelector* can be used to parameterize a plant in "Auto" mode so that the plant is switched on or off either via the input *bValPgm* or via the time schedule *Sched*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_OpMod1St EXTENDS FB_BA_View
VAR_INPUT
  bValPgm          : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bSelector        : BOOL;
END_VAR
VAR_INPUT CONSTANT
  OpModMan         : FB_BA_MV_Op;
  Sched            : FB_BA_SchedM;
  OpModPr          : FB_BA_MV_Op;
END_VAR
VAR_OUTPUT
  eOpMod           : E_BA_OnOff;
  bCmd             : BOOL;
END_VAR
VAR
  PrioSwiOpMod     : FB_BA_PrioSwi_UDI04;
END_VAR
```

Inputs

Name	Type	Description
bValPgm	BOOL	A plant can be switched on or off via the input in the operating mode "Auto".

Inputs CONSTANT

Name	Type	Description
bSelector	BOOL	The variable can be used to parameterize a system in the "Auto" operating mode so that the system is switched on or off either via the input *bValPgm* or via the schedule *Sched*.

Inputs CONSTANT

Name	Type	Description
OpModMan	FB_BA_MV_Op [▶ 239]	The Multistate-Value object represents an operating mode switch with the operating modes "Auto", "Manual Off" and "Manual On".
Sched	FB_BA_SchedM [▶ 225]	A plant can be switched on or off by the schedule in the operating mode "Auto" and the parameter *bSelector* = TRUE.
OpModPr	FB_BA_MV_Op [▶ 239]	The Multistate-Value object indicates the state of the currently valid plant operating mode.

Outputs

Name	Type	Description
eOpMod	E_BA_OnOff [▶ 700]	Displays the state of the currently valid plant operating mode.
bCmd	BOOL	The output shows the state of the currently valid plant operating mode.

Variables

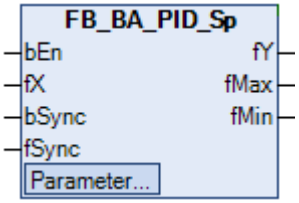
Name	Type	Description
PrioSwiOpMod	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch detects the operating modes, prioritizes them and forwards the result to *OpModPr*.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.10 PID

6.1.4.2.2.2.10.1 FB_BA_PID_Sp



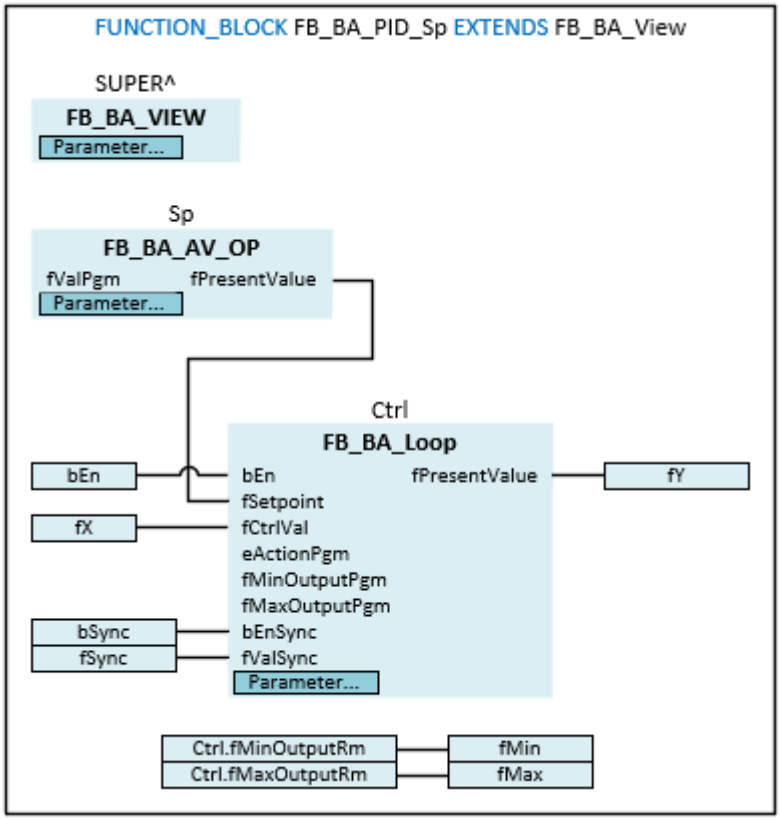
The template is a universal PID controller.

The PID controller is enabled via the input variable *bEn*.

The setpoint is entered via the AV object *Sp*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_PID EXTENDS FB_BA_View
VAR_INPUT
  bEn       : BOOL;
  fX        : REAL;
  bSync     : BOOL;
  fSync     : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
  fMax      : REAL;
  fMin      : REAL;
END_VAR
VAR_INPUT CONSTANT
  Sp        : FB_BA_AV_Op;
  Ctrl      : FB_BA_Loop;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template
fX	REAL	Actual value
bSync	BOOL	A rising edge at this input triggers synchronization of the loop object to the value of *fSync*.
fSync	REAL	Synchronization value

Outputs

Name	Type	Description
fY	REAL	Control value output
fMax	REAL	Maximum controller value
fMin	REAL	Minimum controller value

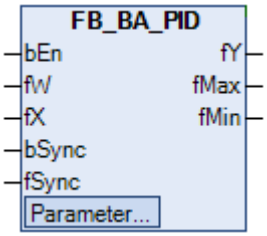
Inputs CONSTANT

Name	Type	Description
Sp	FB_BA_AV_Op [▶ 202]	Input of the setpoint
Ctrl	FB_BA_Loop [▶ 220]	PID controller

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.10.2 FB_BA_PID

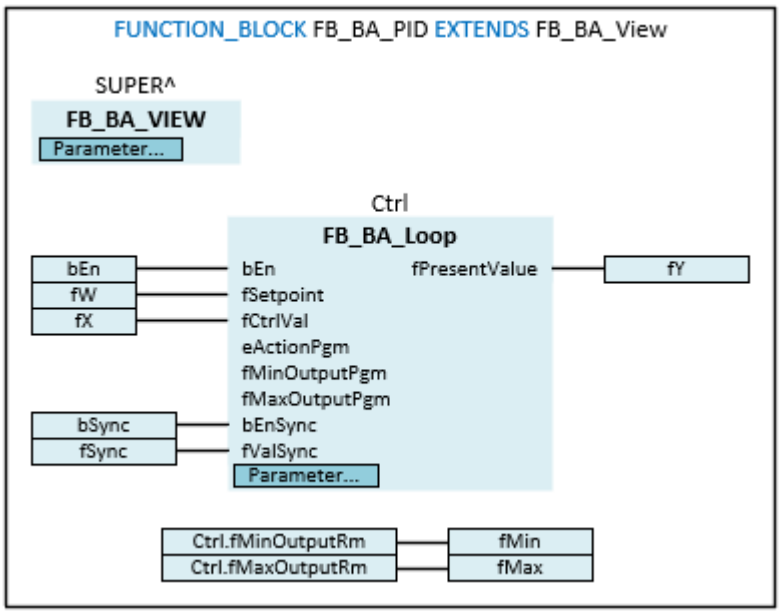


The template is a universal PID controller.

The PID controller is enabled via the input variable *bEn*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_PID EXTENDS FB_BA_View
VAR_INPUT
  bEn       : BOOL;
  fW        : REAL;
  fX        : REAL;
  bSync     : BOOL;
  fSync     : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
  fMax      : REAL;
  fMin      : REAL;
END_VAR
VAR_INPUT CONSTANT
  Ctrl      : FB_BA_Loop;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template
fW	REAL	Setpoint
fX	REAL	Actual value
bSync	BOOL	A rising edge at this input triggers synchronization of the loop object to the value of *fSync*.
fSync	REAL	Synchronization value

Outputs

Name	Type	Description
fY	REAL	Control value output
fMax	REAL	Maximum controller value
fMin	REAL	Minimum controller value

Inputs CONSTANT

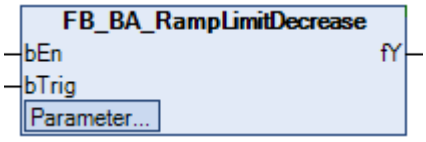
Name	Type	Description
Ctrl	FB_BA_Loop [▶ 220]	PID controller

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.11 Ramp

6.1.4.2.2.2.11.1 FB_BA_RampLimitDecrease

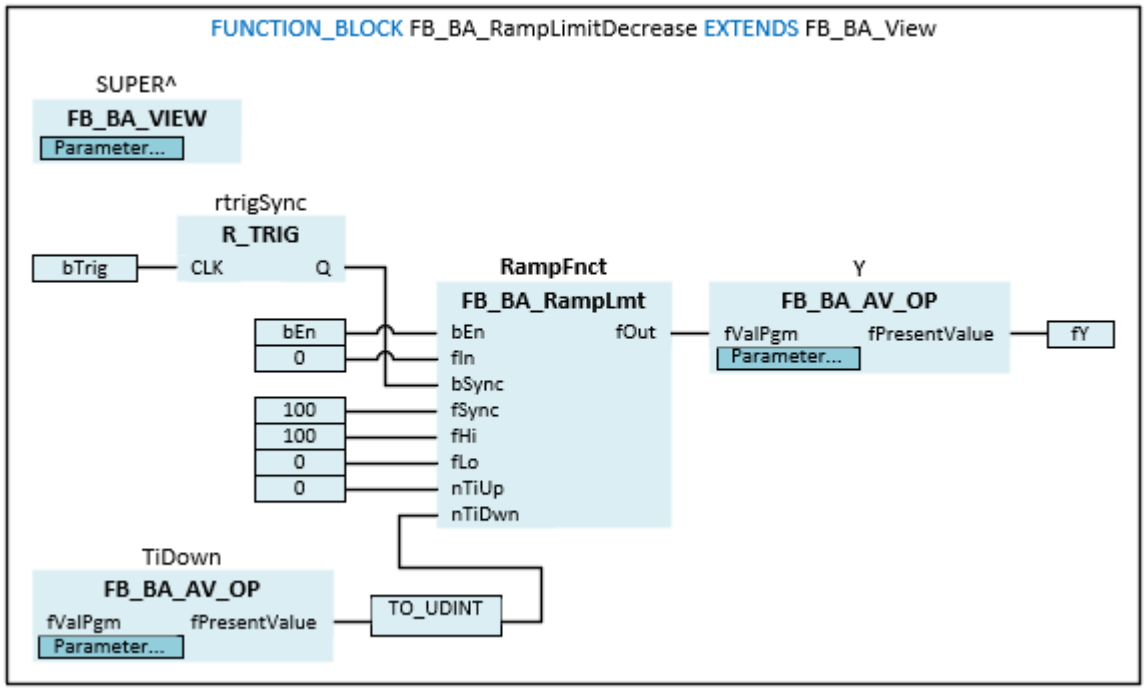


The template implements a falling ramp limitation from 100 to 0 and remains at this value.

Ramp limitation is triggered by a rising edge at the input *bTrig* and specified in terms of time by *TiDown*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_RampLimitDecrease EXTENDS FB_BA_View
VAR_INPUT
  bEn           : BOOL;
  bTrig         : BOOL;
END_VAR
VAR_OUTPUT
  fY            : REAL;
```

```
END_VAR
VAR_INPUT CONSTANT
  TiDown       : FB_BA_AV_Op;
  Y            : FB_BA_AV_Op;
END_VAR
VAR
  RampFnct     : FB_BA_RampLmt;
  rtrigSync    : R_TRIG;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
bTrig	BOOL	The ramp limitation is activated by a rising edge at this input.

Outputs

Name	Type	Description
fY	REAL	Output of the ramp limiting value.

Inputs CONSTANT

Name	Type	Description
TiDown	FB_BA_AV_Op [▶ 202]	AV object for entering the fall time of the ramp limitation.
Y	FB_BA_AV_Op [▶ 202]	AV object for displaying the ramp limitation value.

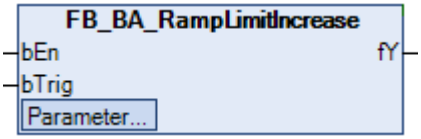
Variables

Name	Type	Description
RampFnct	FB_BA_RampLmt	The function block for the output of a ramp limitation is the core of this template.
rtrigSync	R_TRIG	The function block triggers a rising edge and activates the ramp limitation.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.11.2 FB_BA_RampLimitIncrease

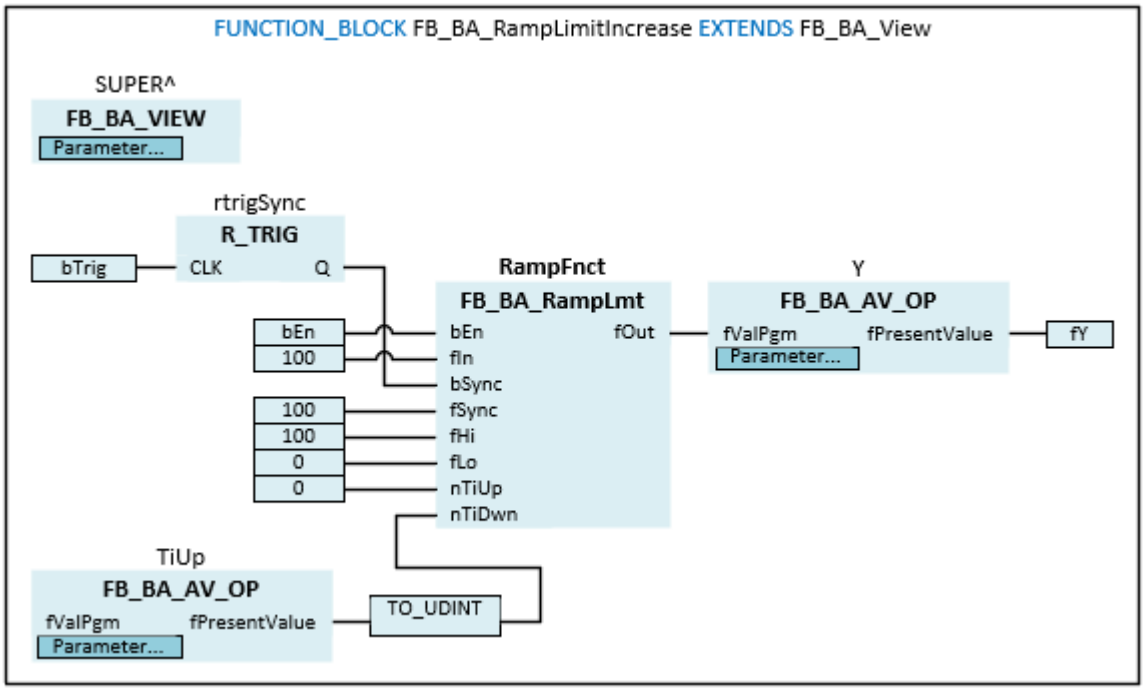


The template implements a rising ramp limitation from 0 to 100 and remains at this value.

The ramp limitation is triggered by a rising edge at the *bTrig* input and is timed by *TiUp*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_RampLimitIncrease EXTENDS FB_BA_View
VAR_INPUT
  bEn          : BOOL;
  bTrig        : BOOL;
END_VAR
VAR_OUTPUT
  fY           : REAL;
END_VAR
VAR_INPUT CONSTANT
  TiUp         : FB_BA_AV_Op;
  Y            : FB_BA_AV_Op;
END_VAR
VAR
  RampFnct     : FB_BA_RampLmt;
  rtrigSync    : R_TRIG;
END_VAR
```

Inputs

Name	Type	Description
bEn	BOOL	General enable of the template.
bTrig	BOOL	The ramp limitation is activated by a rising edge at this input.

Outputs

Name	Type	Description
fY	REAL	Output of the ramp limiting value.

Inputs CONSTANT

Name	Type	Description
TiUp	FB_BA_AV_Op [▶ 202]	AV object for entering the time for the ramp limitation to rise.
Y	FB_BA_AV_Op [▶ 202]	AV object for displaying the ramp limitation value.

Variables

Name	Type	Description
RampFnct	FB_BA_RampLmt	The function block for the output of a ramp limitation is the core of this template.
rtrigSync	R_TRIG	The function block triggers a rising edge and activates the ramp limitation.

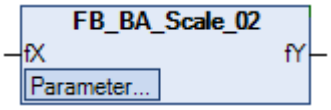
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.12 Scale

Templates for scaling analog values.

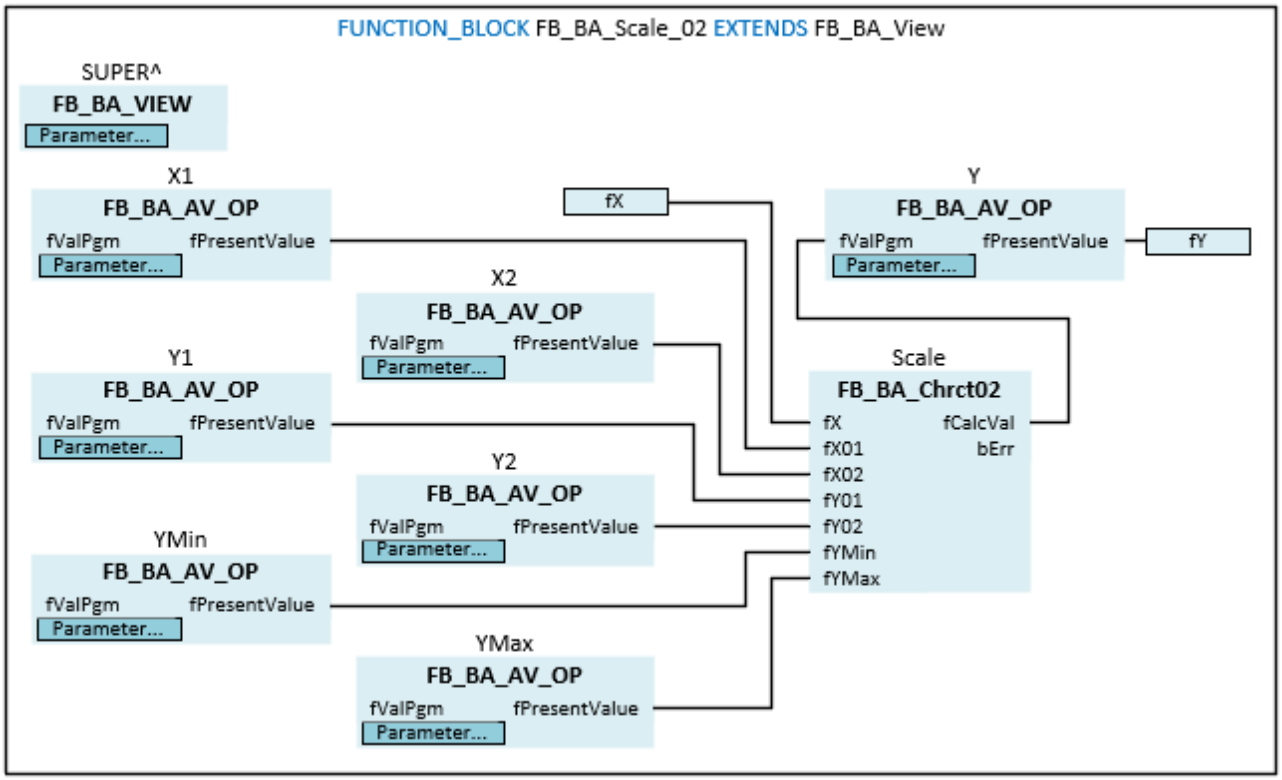
6.1.4.2.2.2.12.1 FB_BA_Scale_02



The template represents a linear interpolation with two interpolation points and can be used to create characteristic curves. 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.

Block diagram



Syntax

```
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_Chrct02;
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 [▶ 202]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X2.
Y1	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y2.
YMin	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the minimum limit of *fY*.
YMax	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the maximum limit of *fY*.
Y	FB_BA_AV_Op [▶ 202]	Output of the calculated output value of the characteristic curve.

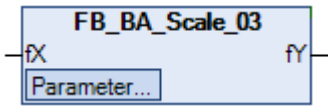
Variables

Name	Type	Description
Scale	FB_BA_Chrct02	The function block represents a linear interpolation with two 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

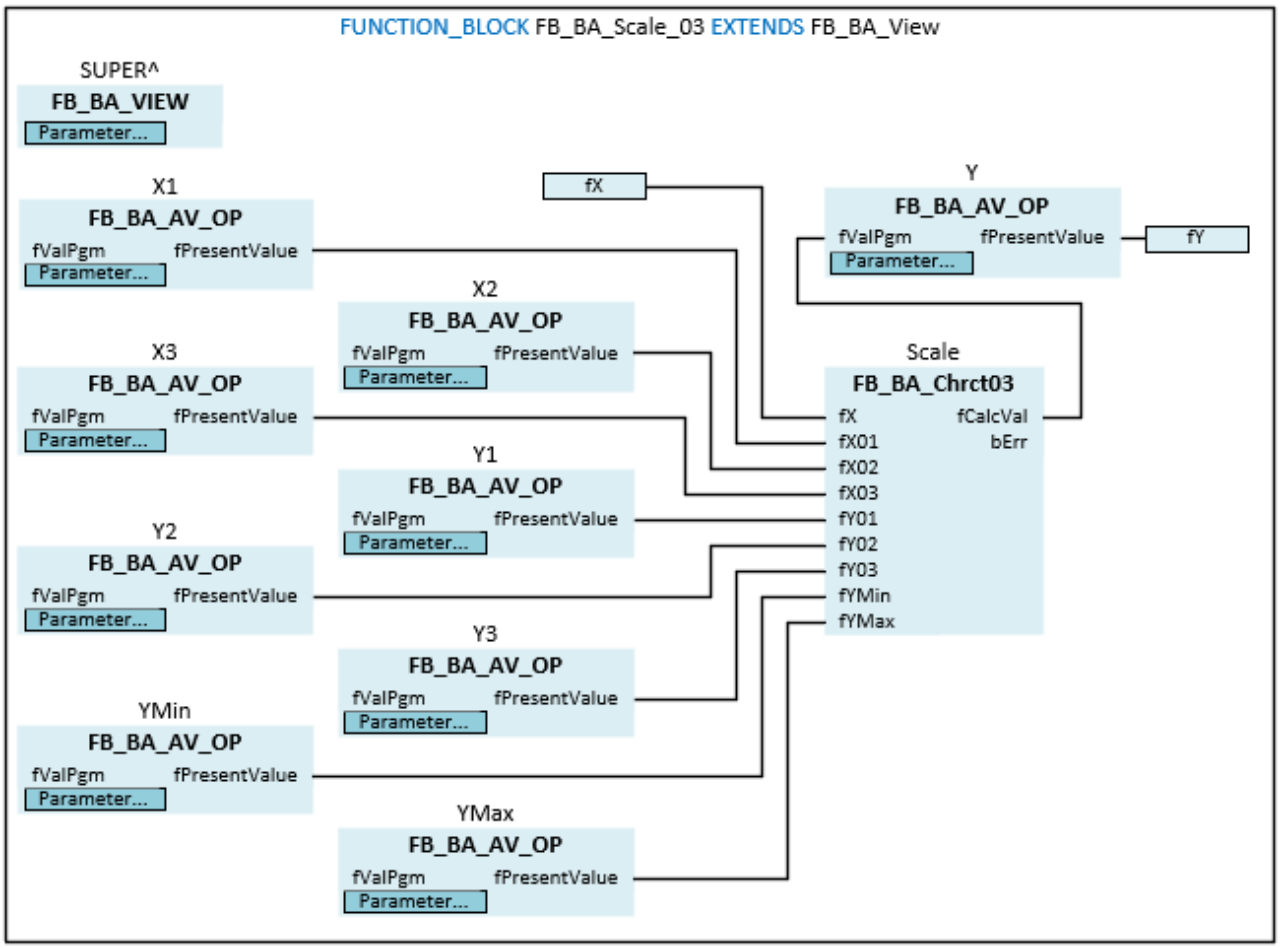
6.1.4.2.2.12.2 FB_BA_Scale_03



The template represents a linear interpolation with three 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.

Block diagram



Syntax

```
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_Chrct03;
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 [▶ 202]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X3.
Y1	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y2.
Y3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y3.
YMin	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the minimum limit of *fY*.
YMax	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the maximum limit of *fY*.
Y	FB_BA_AV_Op [▶ 202]	Output of the calculated output value of the characteristic curve.

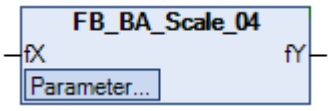
Variables

Name	Type	Description
Scale	FB_BA_Chrct03	The function block represents a linear interpolation with three 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

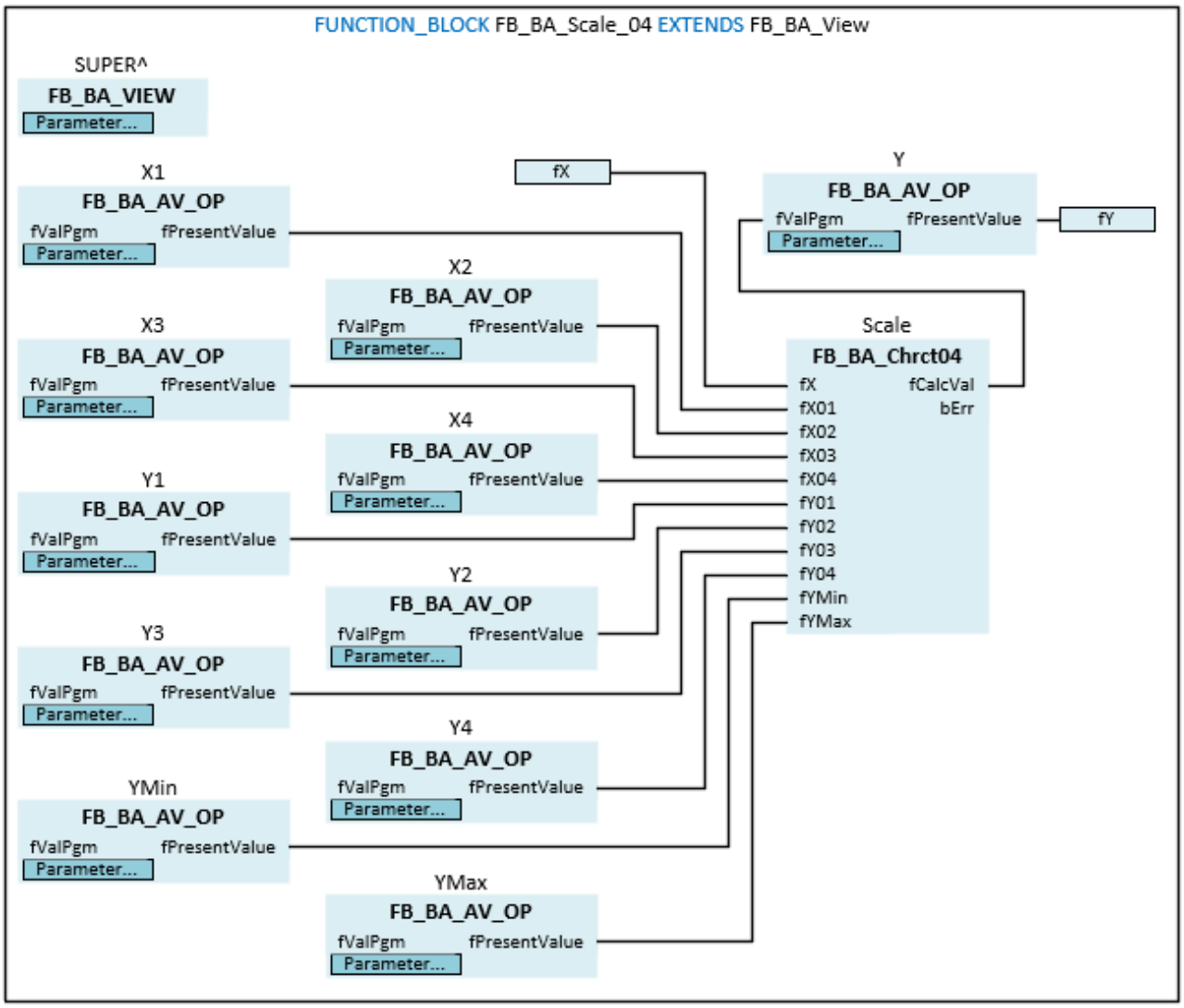
6.1.4.2.2.12.3 FB_BA_Scale_04



The template represents a linear interpolation with four 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.

Block diagram



Syntax

```
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_Chrct04;
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 [▶ 202]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X3.
X4	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X4.
Y1	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y2.
Y3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y3.
Y4	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y4.
YMin	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the minimum limit of *fY*.
YMax	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the maximum limit of *fY*.
Y	FB_BA_AV_Op [▶ 202]	Output of the calculated output value of the characteristic curve.

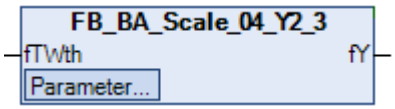
Variables

Name	Type	Description
Scale	FB_BA_Chrc04	The function block represents a linear interpolation with four 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

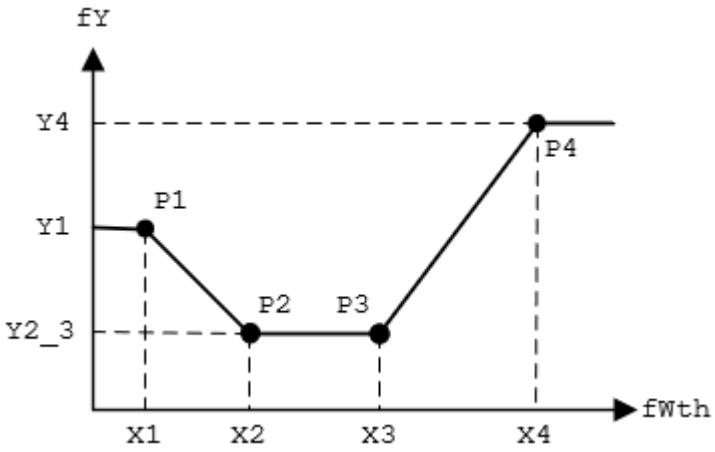
6.1.4.2.2.12.4 FB_BA_Scale_04_Y2_3



The template is a setpoint program for an extract air/supply air cascade with only one room temperature setpoint, including summer and winter compensation.

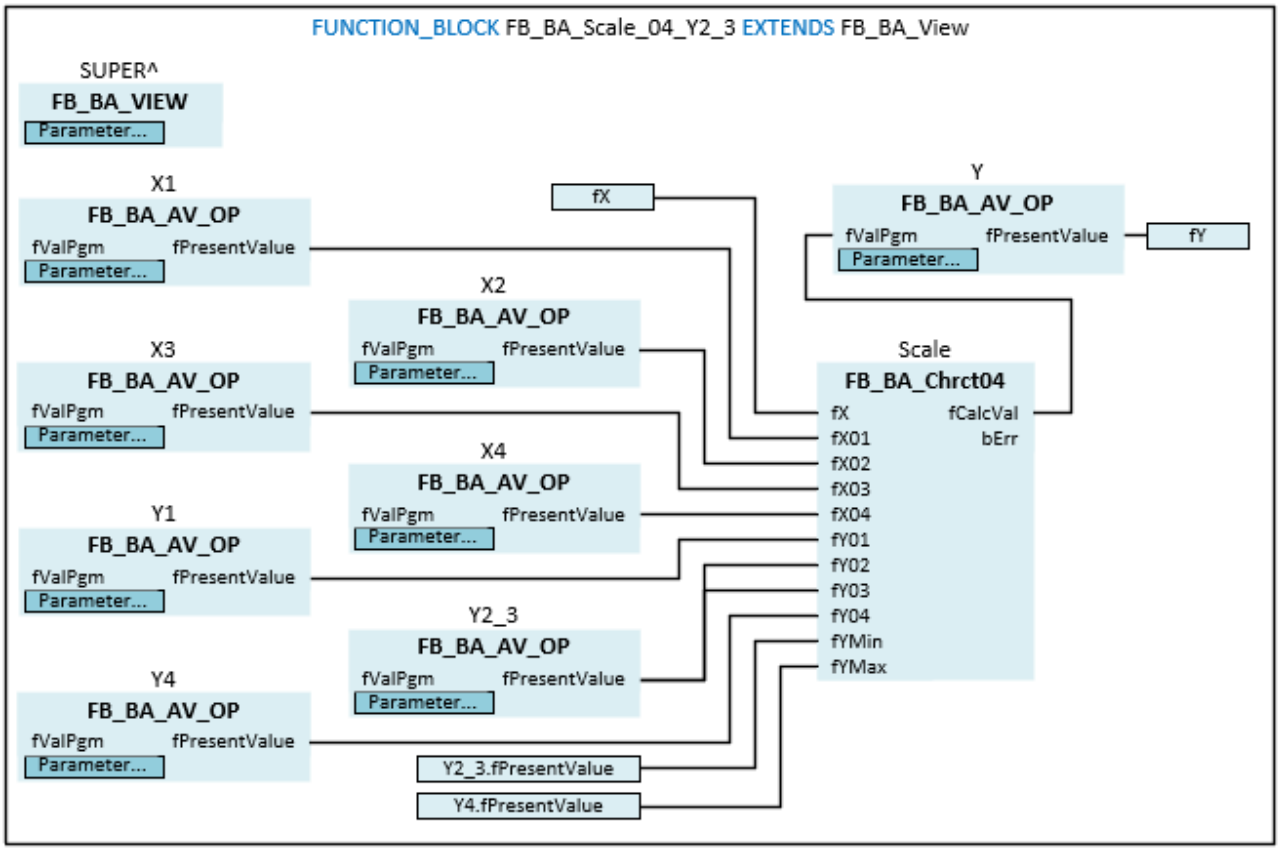
Setpoint program for supply air temperature control with a supply air temperature setpoint, including summer/winter compensation via a characteristic curve.

The supply air temperature setpoint is determined by the function block Scale depending on the outside temperature.



The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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_Chrct04;
END_VAR
```

Inputs

Name	Type	Description
fTWth	REAL	Current 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 [▶ 202]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X3.
X4	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X4.
Y1	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y1.
Y2_3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the interpolation points Y2/Y3.
Y4	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y4.
Y	FB_BA_AV_Op [▶ 202]	Output of the calculated, simple room temperature setpoint. It is output at output *fY*.

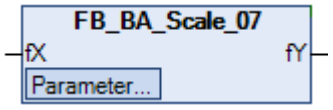
Variables

Name	Type	Description
Scale	FB_BA_Chrct04	The function block represents a linear interpolation with four interpolation points and can be used to generate a characteristic curve. The function block calculates the setpoint 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

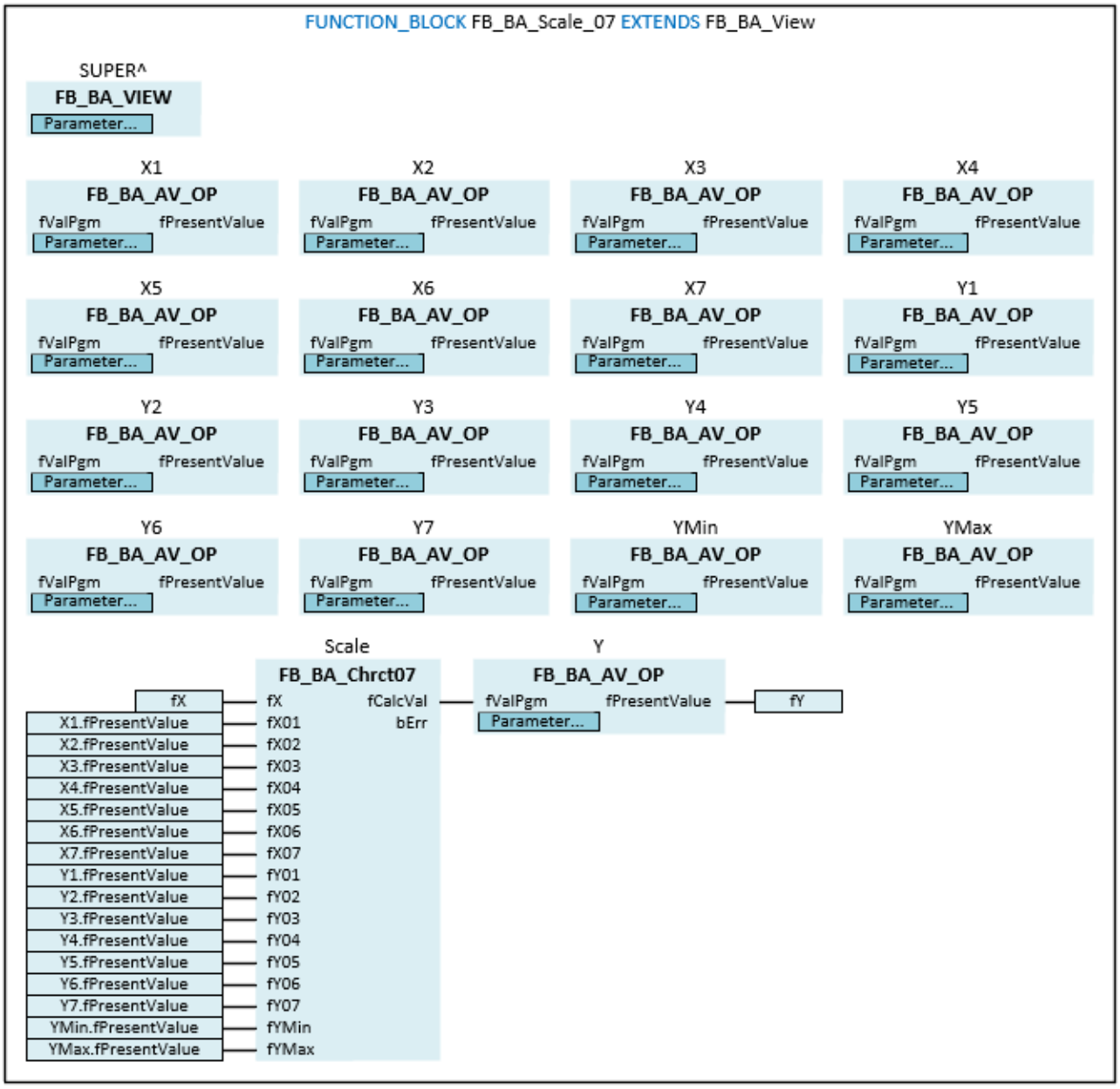
6.1.4.2.2.2.12.5 FB_BA_Scale_07



The template represents a linear interpolation with seven 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.

Block diagram



Syntax

```
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_Chrct07;
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 [▶ 202]	The AV object is used to specify the value for interpolation point X1.
X2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X2.
X3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X3.
X4	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X4.
X5	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X5.
X6	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X6.
X7	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point X7.
Y1	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y1.
Y2	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y2.
Y3	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y3.
Y4	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y4.
Y5	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y5.
Y6	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y6.
Y7	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for interpolation point Y7.
YMin	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the minimum limit of *fY*.
YMax	FB_BA_AV_Op [▶ 202]	The AV object is used to specify the value for the maximum limit of *fY*.
Y	FB_BA_AV_Op [▶ 202]	Output of the calculated output value of the characteristic curve.

Variables

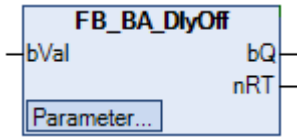
Name	Type	Description
Scale	FB_BA_Chrct07	The function block represents a linear interpolation with seven interpolation points and can be used to generate a characteristic curve.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.2.1.0

6.1.4.2.2.2.13 TimeDelay

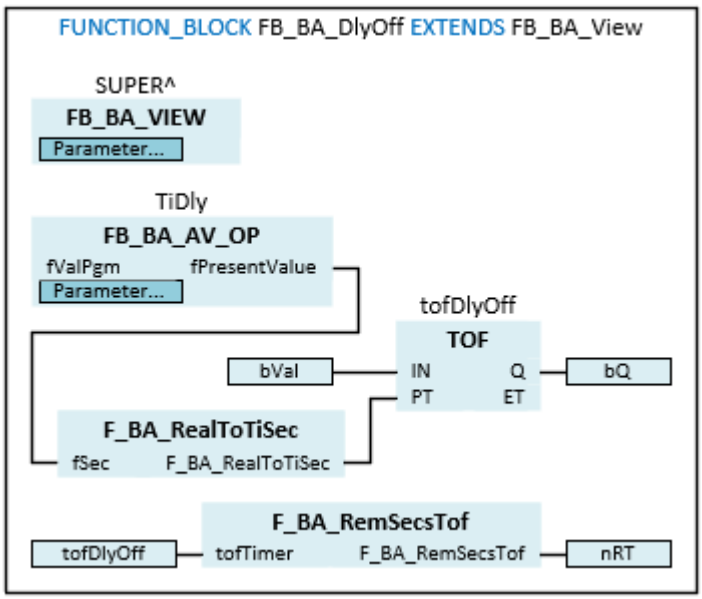
6.1.4.2.2.2.13.1 FB_BA_DlyOff



The template implements a switch-off delay and can be used for the overrun control of pumps.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_DlyOff EXTENDS FB_BA_View
VAR_INPUT
  bVal          : BOOL;
END_VAR
VAR_OUTPUT
  bQ            : BOOL;
  nRT           : UDINT;
END_VAR
VAR_INPUT CONSTANT
  TiDly         : FB_BA_AV_Op;
END_VAR
```

```
VAR
  tofDlyOff    : TOF;
END_VAR
```

Inputs

Name	Type	Description
bVal	BOOL	A falling edge activates the switch-off delay.

Outputs

Name	Type	Description
bQ	BOOL	State of the switch-off delay.
nRT	UDINT	Remaining time of the switch-off delay. After this time has elapsed, *bQ* is set to FALSE.

Inputs CONSTANT

Name	Type	Description
TiDly	FB_BA_AV_Op [▶ 202]	AV object for entering the value of the switch-off delay.

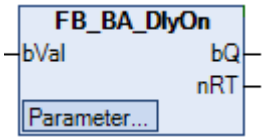
Variables

Name	Type	Description
tofDlyOff	TOF	The timing element is the core of this template for switch-off delay.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

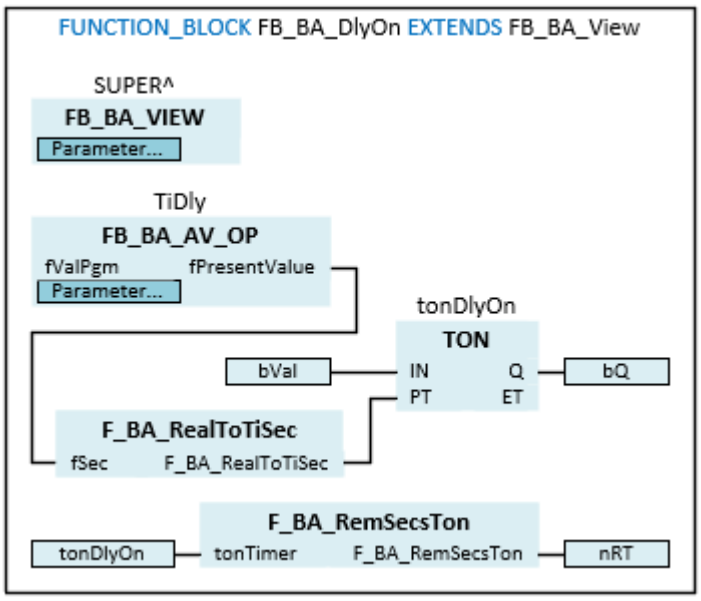
6.1.4.2.2.13.2 FB_BA_DlyOn



The template implements a start-up delay.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_DlyOff EXTENDS FB_BA_View
VAR_INPUT
  bVal         : BOOL;
END_VAR
VAR_OUTPUT
  bQ           : BOOL;
  nRT          : UDINT;
END_VAR
VAR_INPUT CONSTANT
  TiDly        : FB_BA_AV_Op;
END_VAR
VAR
  tonDlyOn     : TON;
END_VAR
```

Inputs

Name	Type	Description
bVal	BOOL	A rising edge activates the start-up delay.

Outputs

Name	Type	Description
bQ	BOOL	State of the start-up delay.
nRT	UDINT	Remaining time of the start-up delay. After this time has elapsed, *bQ* is set to TRUE.

Inputs CONSTANT

Name	Type	Description
TiDly	FB_BA_AV_Op [▶ 202]	AV object for entering the value of the start-up delay.

Variables

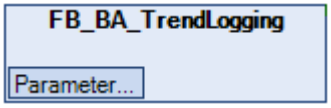
Name	Type	Description
tofDlyOn	TON	The timing element is the core of this template for start-up delay.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.2.14 Trend

6.1.4.2.2.2.14.1 FB_BA_TrendLogging



The template logs the values from a data source and records them in the Trendlog memory. The connection to the Trendlog objects (data sources, e.g. FB_BA_BO_IO [▶ 209], FB_BA_AO_IO [▶ 197]) is made via references.

The variable *TrendCount* defines the size of the field of trendlog objects.

The initialization of the template takes place within the method FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_TrendLogging EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  Trends          : ARRAY[1..TREND_COUNT] OF FB_BA_Trend;
END_VAR
VAR
  i               : UDINT;
END_VAR
VAR CONSTANT
  TREND_COUNT     : UDINT := 30;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
Trends	FB_BA_Trend [▶ 226]	Field with 30 entries for trend log objects.

Variables

Name	Type	Description
i	UDINT	Counter for the FOR loop to call the trendlog field Trends.

Variables

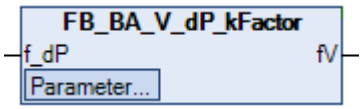
Name	Type	Description
TrendCount	UDINT	Constant to define the set of trendlog objects to be logged. Preset to 30.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.15 VolumeFlowDetermination

6.1.4.2.2.15.1 FB_BA_V_dP_kFactor



The template is used to determine the volume flow by means of differential pressure and k-factor.

The initialization of the template takes place within the method FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_V_dP_kFactor
VAR_INPUT
  {attribute 'parameterUnit':='Pa'}
  f_dP                  : REAL;
END_VAR
VAR_OUTPUT
  {attribute 'parameterUnit':='m³/h'}
    fV                  : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory':='Unit'}
  eConversion_kFactor    : E_BA_Conversion_kFactor := E_BA_Conversion_kFactor.e_Pa;
  {attribute 'parameterCategory':='Config'}
  fK                    : REAL := 381;
END_VAR
VAR
    fConversionF        : REAL;
END_VAR
VAR CONSTANT
    f_Pa                : REAL := 1;
    f_hPa               : REAL := 0.1;
    f_mbar              : REAL := 0.1;
    f_mm_HG             : REAL := 0.086613;
    f_in_HG             : REAL := 0.017185;
    f_mm_WS             : REAL := 0.31933;
    f_psi               : REAL := 0.012043;
    f_inches_H2O        : REAL := 0.063361;
END_VAR
```

Inputs

Name	Type	Description
f_dP	REAL	Detection of the differential pressure [Pa].

Outputs

Name	Type	Description
fV	REAL	Output of the calculated volume flow [m³/h].

Inputs CONSTANT PERSISTENT

Name	Type	Description
eConversion_kFactor	E_BA_Conversion_kFactor [▶ 699]	Conversion factor for the unit of the k-factor.
fK	REAL	k-factor of the fan nozzle vendor (Venturi principle).

Variables

Name	Type	Description
fConversionF	REAL	Indicates which conversion factor has been selected by the enumeration variable *eConversion_kFactor* and is used to calculate the volume flow rate.

Variables

Name	Type	Description
f_Pa	REAL	Conversion factor for pressure [Pa].
f_hPa	REAL	Conversion factor for pressure [hPa].
f_mbar	REAL	Conversion factor for pressure [mbar].
f_mm_HG	REAL	Conversion factor for the unit millimeter mercury column [mmHG, Torr].
f_in_HG	REAL	Conversion factor for the unit inch of mercury [inHG].
f_mm_WS	REAL	Conversion factor for the unit millimeter water column [mmWS, mmH2O].
f_psi	REAL	Conversion factor for pressure [psi].
f_inchesn_H2O	REAL	Conversion factor for the unit inch water column [inH2O].

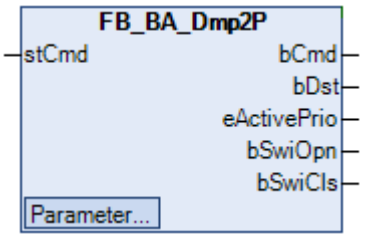
Requirements

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 Damper

6.1.4.2.2.3.1 2P

6.1.4.2.2.3.1.1 FB_BA_Dmp2P

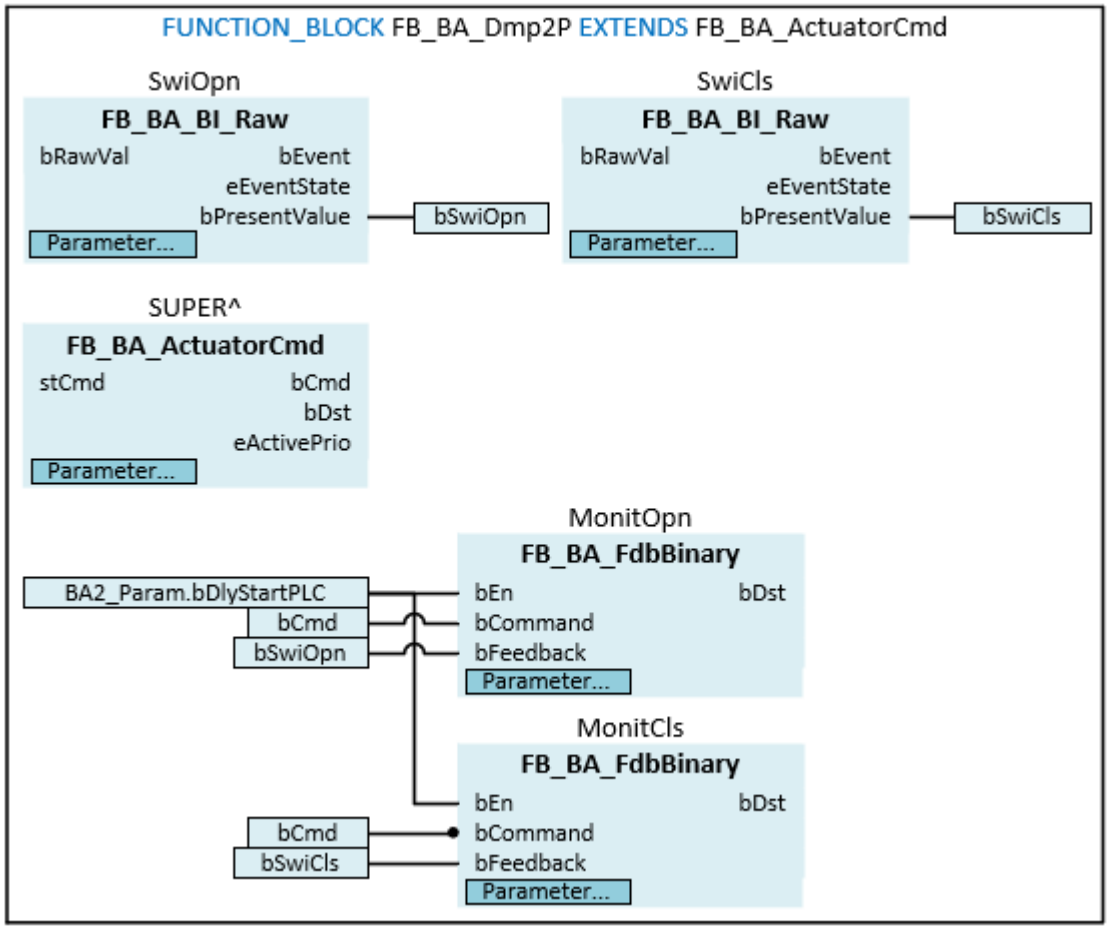


The template is for the control of a two-point damper with integrated monitoring of both end positions.

It essentially consists of the base class FB_BA_ActuatorCmd [▶ 986] 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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 [▶ 206]	Binary input object is used to process the limit switch "Open".
SwiCls	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the limit switch "Closed".
MonitOpen	FB_BA_FdbBinary [▶ 1018]	The template monitors the "Open" end position.
MonitClose	FB_BA_FdbBinary [▶ 1018]	The template monitors the "Closed" end position.

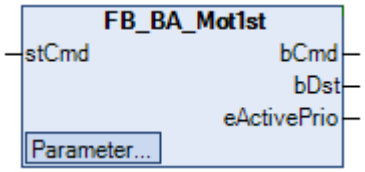
Requirements

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

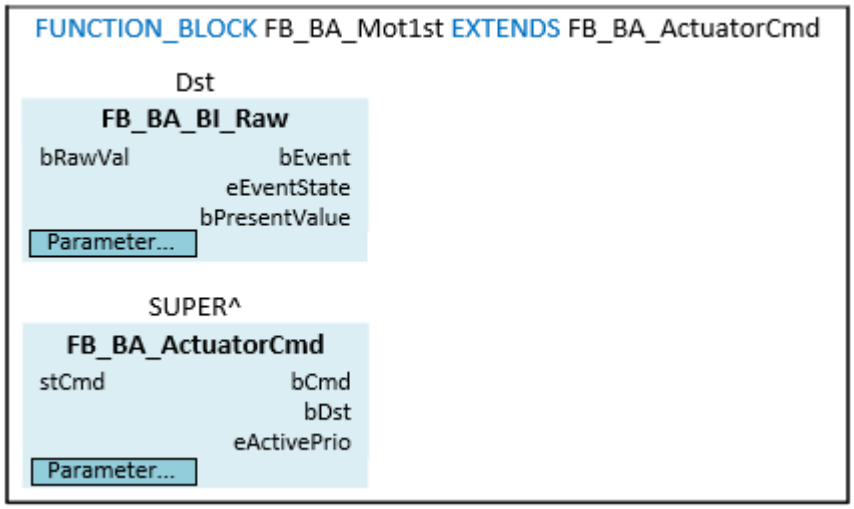


The template is used for the control of a single-step motor with fault signal, e.g. a fan.

It essentially consists of the base class FB_BA_ActuatorCmd [▶ 986] for controlling a binary aggregate and collecting all safety-related faults. The BI object *Dst* indicates a motor fault.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Mot1st EXTENDS FB_BA_ActuatorCmd
VAR_INPUT CONSTANT
  Dst    : FB_BA_BI_Raw;
END_VAR
VAR
END_VAR
```

Inputs CONSTANT

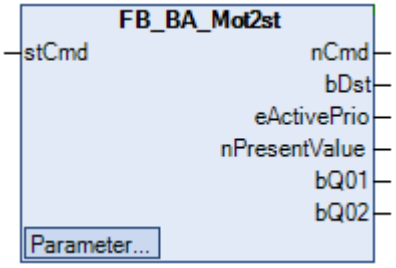
Name	Type	Description
Dst	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process a fault.

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.

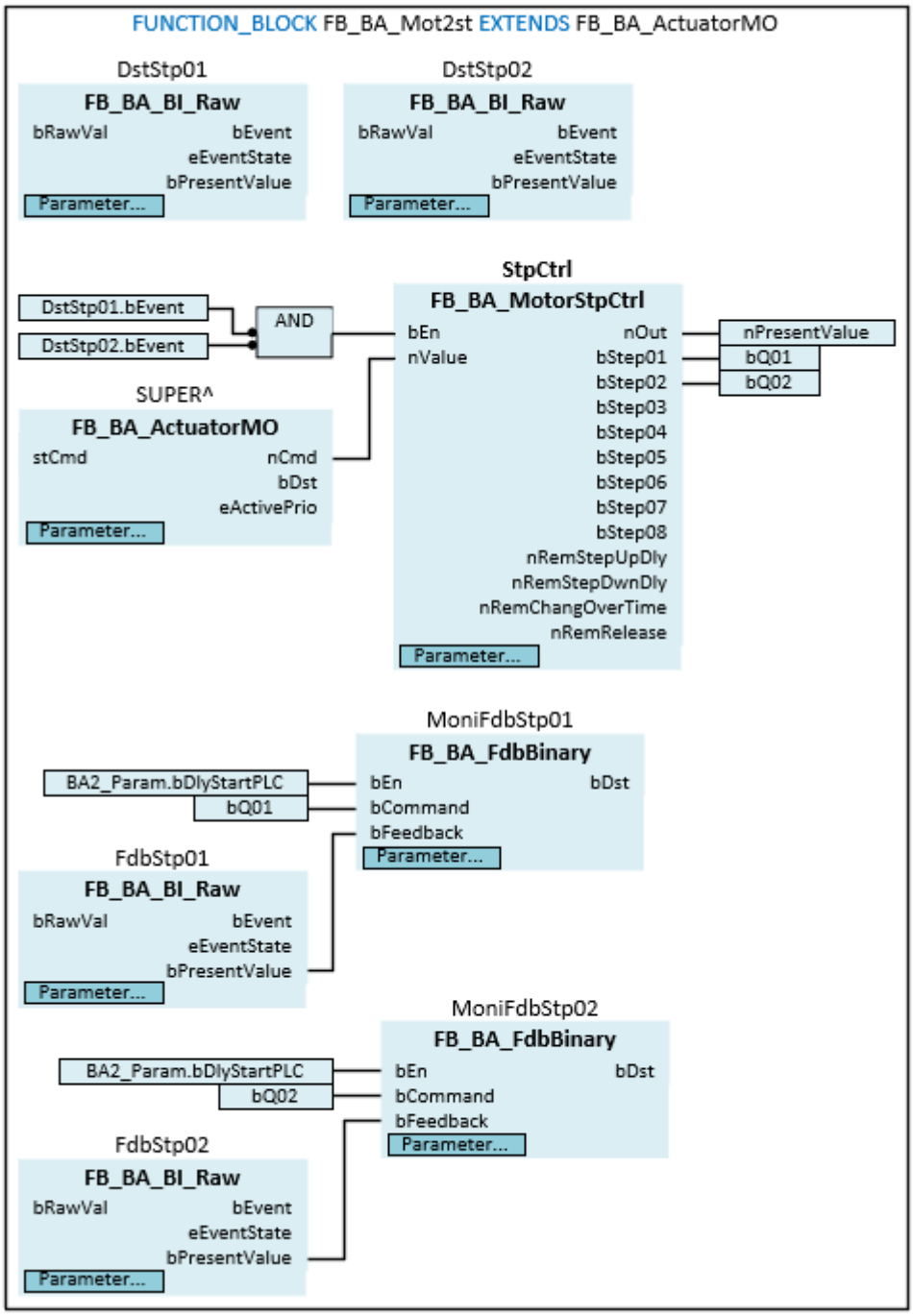
It mainly consists of the function block *StpCtrl* and the base class FB_BA_ActuatorMO [▶ 991] for the control of a multi-stage aggregate and the collection of all safety-relevant faults.

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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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

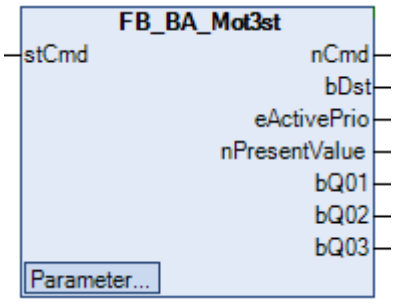
Name	Type	Description
DstStp01	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the step 1 motor fault.
DstStp02	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the step 2 motor fault.
FdbStp01	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the feedback of step 1 of the motor.
FdbStp02	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the feedback of step 2 of the motor.
MoniFdbStp01	FB_BA_FdbBinary [▶ 1018]	Template that monitors the feedback of step 1 of the motor.
MoniFdbStp02	FB_BA_FdbBinary [▶ 1018]	Template that monitors the feedback of step 2 of the motor.
StpCtrl	FB_BA_MotorStpCtrl [▶ 413]	The function block *StpCtrl* receives the numerical switch value from the base class FB_BA_ActuatorMO [▶ 991] 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.

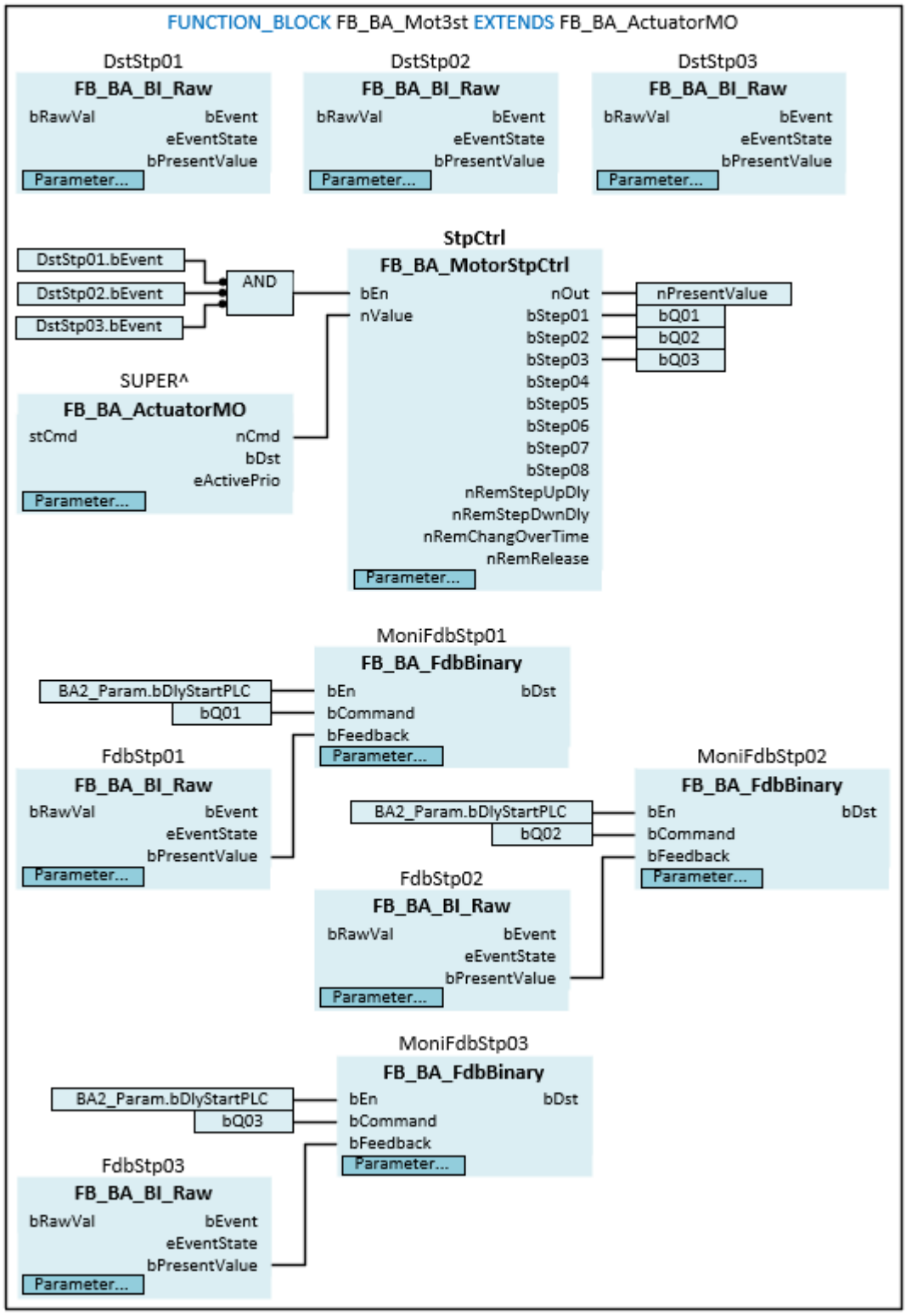
It mainly consists of the function block *StpCtrl* and the base class FB_BA_ActuatorMO [▶ 991] for the control of a multi-stage aggregate and the collection of all safety-relevant faults.

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.

Block diagram



Syntax

```
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

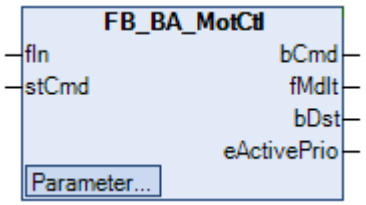
Name	Type	Description
DstStp01	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the step 1 motor fault.
DstStp02	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the step 2 motor fault.
DstStp03	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the step 3 motor fault.
FdbStp01	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the feedback of step 1 of the motor.
FdbStp02	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the feedback of step 2 of the motor.
FdbStp03	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process the feedback of step 3 of the motor.
MoniFdbStp01	FB_BA_FdbBinary [▶ 1018]	Template that monitors the feedback of step 1 of the motor.
MoniFdbStp02	FB_BA_FdbBinary [▶ 1018]	Template that monitors the feedback of step 2 of the motor.
MoniFdbStp03	FB_BA_FdbBinary [▶ 1018]	Template that monitors the feedback of step 3 of the motor.
StpCtrl	FB_BA_MotorStpCtrl [▶ 413]	The function block *StpCtrl* receives the numerical switch value from the base class FB_BA_ActuatorMO [▶ 991] 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

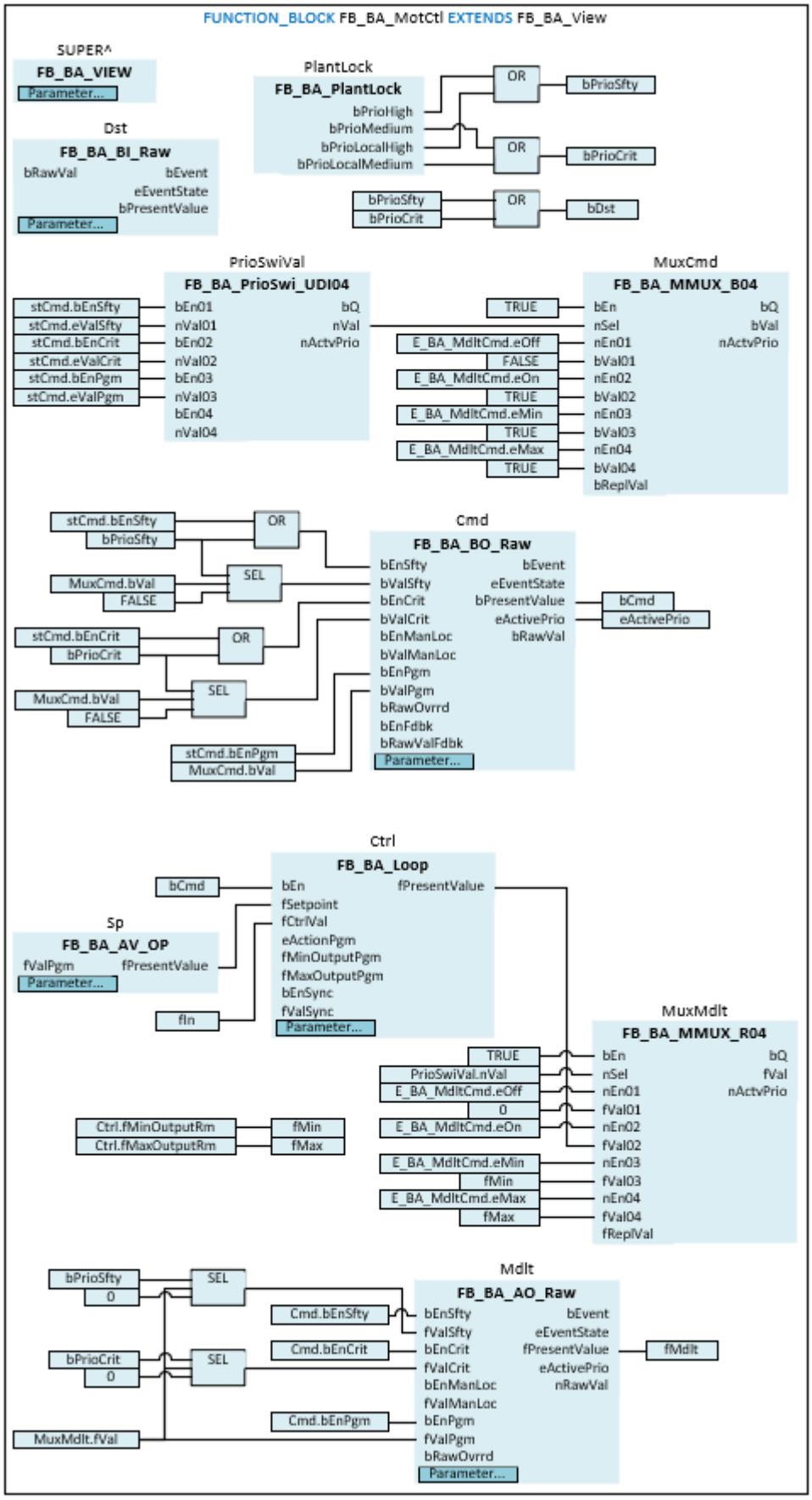


The template is used to control and regulate a speed-controlled motor.

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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtl EXTENDS FB_BA_View
VAR_INPUT
  fIn             : 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
fIn	REAL	The actual value for the controller *Ctrl* is connected to the input.
stCmd	ST_BA_Mdlt [▶ 276]	The enables and modulation commands are transmitted to the template via the command structure *stCmd*. The command with the highest priority determines the switching command at the binary output object *Cmd* and the control command at the analog output object *Mdlt*.

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 [▶ 110]	Display of the active priority.

Inputs CONSTANT

Name	Type	Description
Dst	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process a fault.
Cmd	FB_BA_BO_Raw [▶ 211]	The binary output object is used to output a switching command and transmit it to the I/O level.
Mdlt	FB_BA_AO_Raw [▶ 199]	Current value of the analog output object.
Sp	FB_BA_AV_Op [▶ 202]	Entering the setpoint for the PID controller *Ctrl*.
Ctrl	FB_BA_Loop [▶ 220]	PID controller.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* collects all safety-relevant faults at this level of the project structure and triggers switching commands in the template accordingly.

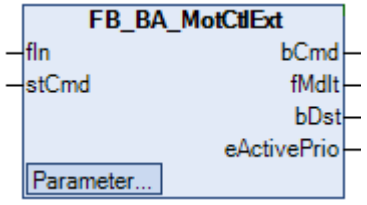
Variables

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 [▶ 433]	The priority switch PrioSwiVal [▶ 433] uses the command structure *stCmd* to determine the modulation command for the multiplexers *MuxCmd* and *MuxMdlt*.
MuxCmd	FB_BA_MMUX_B04 [▶ 428]	The multiplexer MuxCmd [▶ 428] determines the current switch value from the command of the priority switch *PrioSwiVal*. The resulting output is sent to the binary output object *Cmd*.
MuxMdlt	FB_BA_MMUX_R04 [▶ 428]	The multiplexer MuxMdlt [▶ 428] determines the current control value from the modulation values *Ctrl.fPresentValue*, *fMin, fMax* and the modulation command of the priority switch *PrioSwiVal*. The resulting output is sent to the analog output object *Mdlt*.
fMax	REAL	Maximum value of the controller, which is output when the modulation command *E_BA_MdltCmd.eMax* is pending (see E_BA_Mdlt [▶ 270]).
fMin	REAL	Minimum value of the controller, which is output when the modulation command *E_BA_MdltCmd.eMin* is pending (see E_BA_Mdlt [▶ 270]).

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

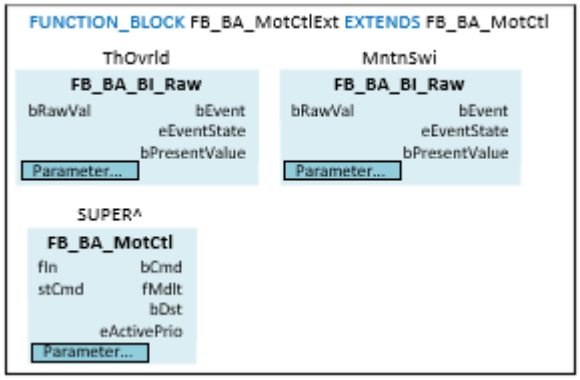


The template is used to control and regulate a speed-controlled motor. It consists essentially of the base class FB_BA_MotCtl [▶ 1066].

The difference to the template FB_BA_MotCtl [▶ 1066] are the additional BI objects *ThOvrld* and *MntnSwi*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtlExt EXTENDS FB_BA_MotCtl
VAR_INPUT CONSTANT
  ThOvrld    : FB_BA_BI_Raw;
  MntnSwi    : FB_BA_BI_Raw;
END_VAR
```

Inputs CONSTANT

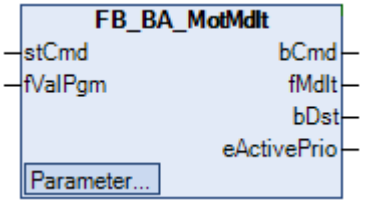
Name	Type	Description
ThOvrld	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a maintenance switch.

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_MotMdlt

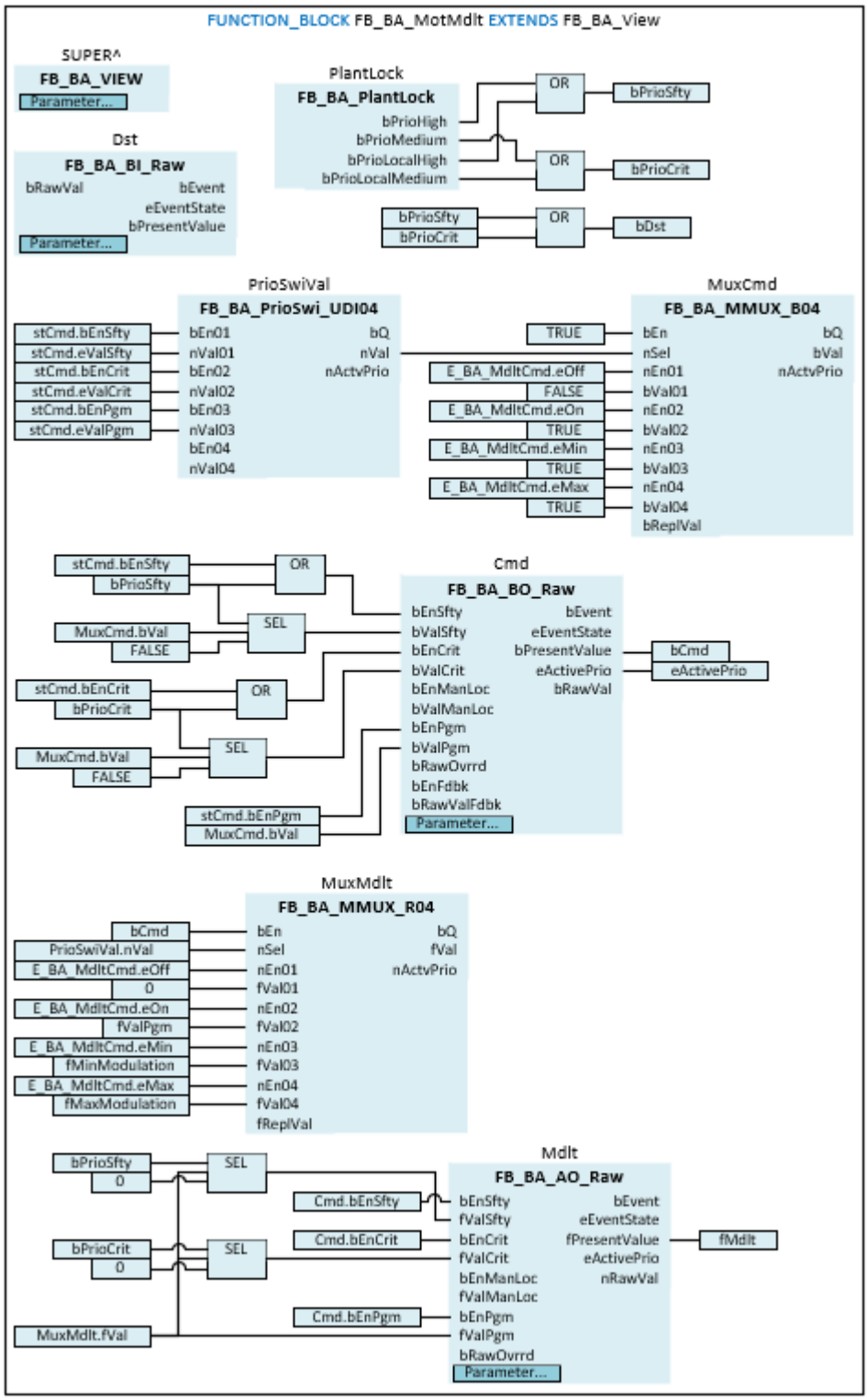


The template is used to control a speed-controlled motor.

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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtl EXTENDS FB_BA_View
VAR_INPUT
  stCmd             : ST_BA_Mdlt;
  fValPgm           : REAL;
END_VAR
```

```
VAR_INPUT CONSTANT
  Dst               : FB_BA_BI_Raw;
  Cmd               : FB_BA_BO_Raw;
  Mdlt              : 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 [▶ 276]	The enables and modulation commands are transmitted to the template via the command structure *stCmd*. The command with the highest priority determines the switching command at the binary output object *Cmd* and the control command at the analog output object *Mdlt*.
fValPgm	REAL	Control value for the control of the motor.

Inputs CONSTANT

Name	Type	Description
Dst	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process a fault.
Cmd	FB_BA_MO [▶ 232]	The multistate output object is used to output the current switch value.
Mdlt	FB_BA_AO_Raw [▶ 199]	Current value of the analog output object.
PlantLock	FB_BA_PlantLock [▶ 135]	The function block *PlantLock* 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 *E_BA_MdltCmd.eMin* is pending (see E_BA_Mdlt [▶ 270]).
fMaxModulation	REAL	Constant maximum value which is output if the modulation command *E_BA_MdltCmd.eMax* is pending (see E_BA_Mdlt [▶ 270]).

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 [▶ 110]	Display of the active priority.

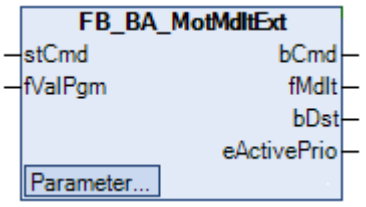
Variables

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 [▶ 433]	The priority switch PrioSwiVal [▶ 433] uses the command structure *stCmd* to determine the modulation command for the multiplexers *MuxCmd* and *MuxMdlt*.
MuxCmd	FB_BA_MMUX_B04 [▶ 428]	The multiplexer MuxCmd [▶ 428] determines the current switch value from the command of the priority switch *PrioSwiVal*. The resulting output is sent to the binary output object *Cmd*.
MuxMdlt	FB_BA_MMUX_R04 [▶ 428]	The multiplexer MuxMdlt [▶ 428] determines the current control value from the modulation values *Ctrl.fPresentValue*, *fMin, fMax* and the modulation command of the priority switch *PrioSwiVal*. The resulting output is sent to the analog output object *Mdlt*.

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_MotMdltExt

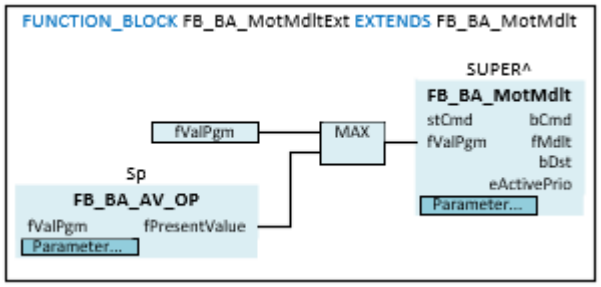


The template is used to control a speed-controlled motor. It consists essentially of the base class FB_BA_MotMdlt [▶ 1070].

The difference to the template FB_BA_MotMdlt [▶ 1070] 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.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtlExt EXTENDS FB_BA_MotMdlt
VAR_INPUT CONSTANT
  Sp     : FB_BA_AV_Op;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
Sp	FB_BA_AV_Op [▶ 202]	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 *fValPgm* of the base class FB_BA_MotMdlt [▶ 1070].

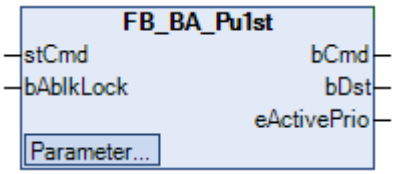
Requirements

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

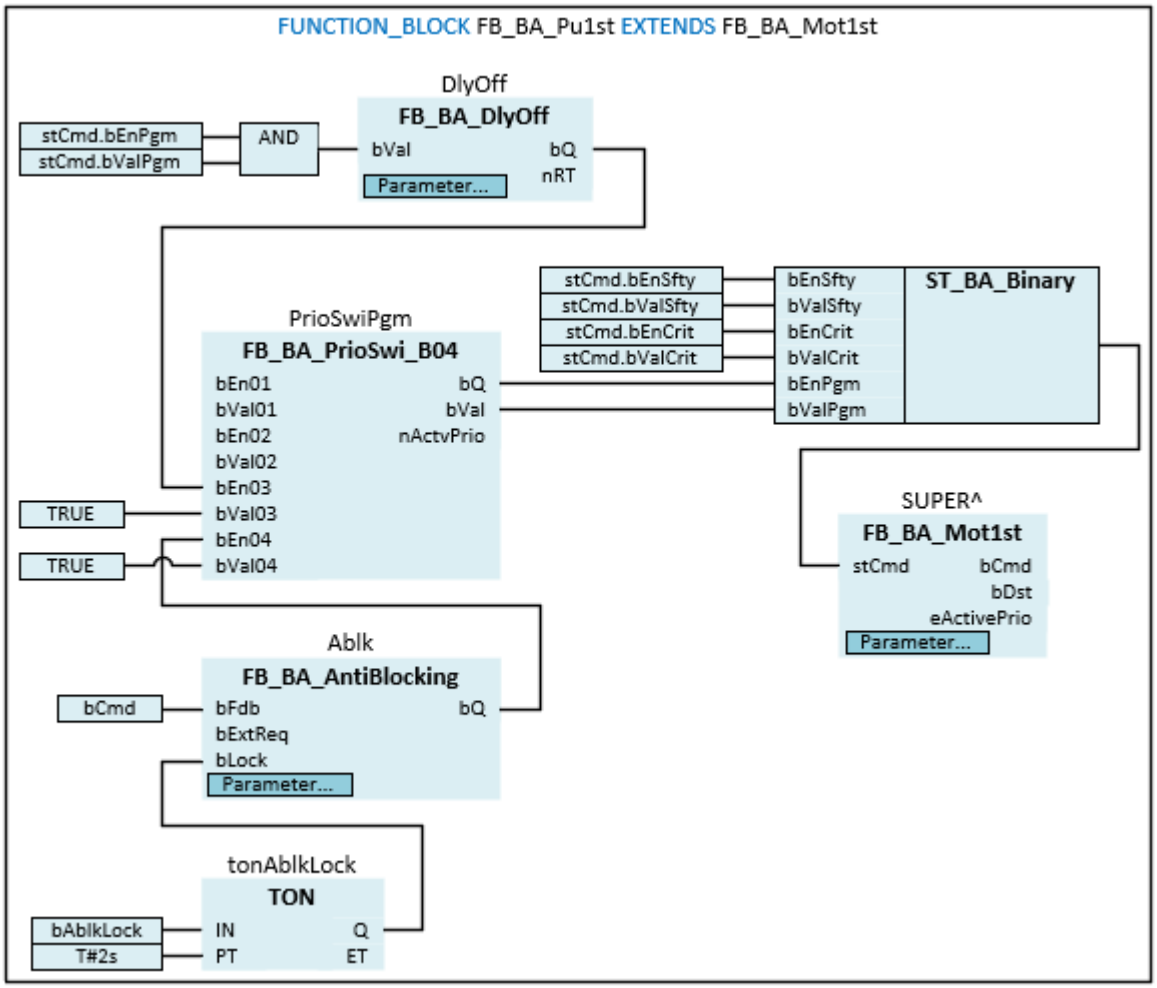


The template is used to control a single-stage pump with binary inputs and outputs. It mainly consists of the base class FB_BA_Mot1st [▶ 1060], the switch-off delay DlyOff [▶ 1053] and the anti-blocking protection function Ablk [▶ 997].

The pump is switched on externally by the priorities of the command structure *stCmd* of the base class FB_BA_Mot1st [▶ 1060] or internally by the anti-blocking protection function Ablk [▶ 997]. The external request via the priority "program" is switched off with a delay by the template *DlyOff*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 anti-blocking protection function Ablk [▶ 997]. It should be prevented that pumps and valves get an anti-blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
DlyOff	FB_BA_DlyOff [▶ 1053]	The template serves as a delayed pump shutdown.
Ablk	FB_BA_AntiBlocking [▶ 997]	Anti-blocking protection.

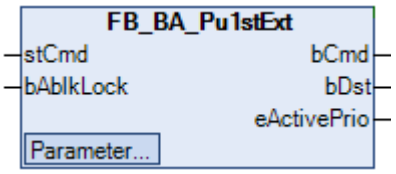
Variables

Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_B04 [▶ 433]	The priority switch *PrioSwiPgm* uses the switch-off delay DlyOff, the anti-blocking protection function Ablk [▶ 997] and the priority program of the command structure *stCmd* of the base class FB_BA_Mot1st [▶ 1060] to determine the current switch value for the base class FB_BA_Mot1st [▶ 1060].
tonAblkLock	TON	Switch-off delay of the anti-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

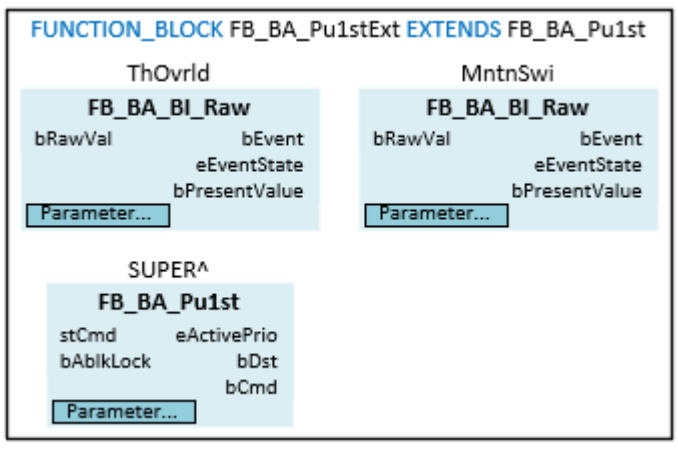


The template is used to control a single-stage pump with binary inputs and outputs. It consists essentially of the base class FB_BA_Pu1st [▶ 1074].

The difference to the template FB_BA_Pu1st [▶ 1074] are the additional BI objects *ThOvrld* and *MntnSwi*.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Pu1stExt EXTENDS FB_BA_Pu1st
VAR_INPUT CONSTANT
  ThOvrld    : FB_BA_BI_Raw;
  MntnSwi    : FB_BA_BI_Raw;
END_VAR
```

Inputs CONSTANT

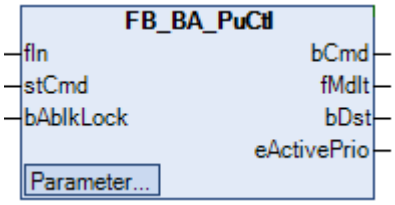
Name	Type	Description
ThOvrld	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a maintenance switch.

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

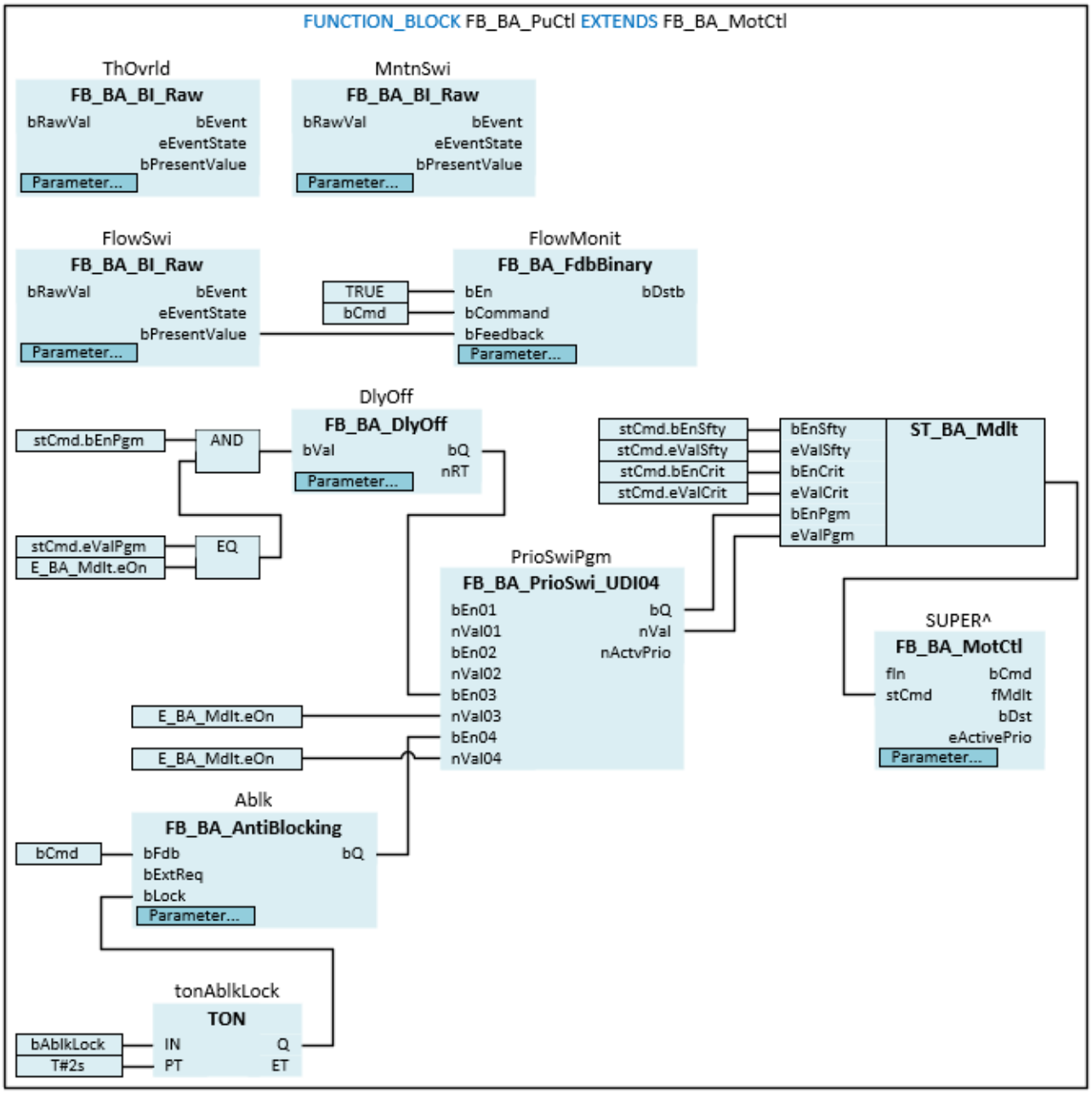


The template is used to control and regulate a speed-controlled pump with analog and binary inputs and outputs. It mainly consists of the base classFB_BA_MotCtl [▶ 1066], the switch-off delay DlyOff [▶ 1053] and the anti-blocking protection function Ablk [▶ 997]. The template *FlowMonit* stands for the 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 [▶ 1066] or internally by the anti-blocking protection function Ablk [▶ 997]. The external request via the priority "Program" is switched off with a delay by the template DlyOff [▶ 1053].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_PuCtl EXTENDS FB_BA_MotCtl
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_UDI04;
  tonAblkLock    : TON;
END_VAR
```

Inputs

Name	Type	Description
bAblkLock	BOOL	A TRUE at this input variable interrupts the anti-blocking protection function Ablk [▶ 997]. It should be prevented that pumps and valves get an anti-blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
ThOvrld	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a maintenance switch.
FlowSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a flow monitor.
FlowMonit	FB_BA_FdbBinary [▶ 1018]	Template for monitoring the flow monitor.
DlyOff	FB_BA_DlyOff [▶ 1053]	The template serves as a delayed pump shutdown.
Ablk	FB_BA_AntiBlocking [▶ 997]	Anti-blocking protection.

Variables

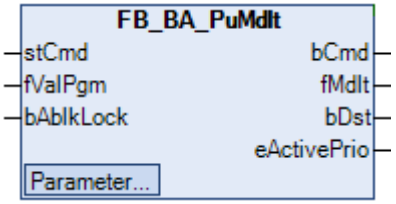
Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch *PrioSwiPgm* uses the switch-off delay DlyOff, the anti-lock protection function Ablk [▶ 997] and the priority program of the command structure *stCmd* of the base class FB_BA_MotCtl [▶ 1066] to determine the current modulation value for the base class FB_BA_MotCtl [▶ 1066].
tonAblkLock	TON	Switch-off delay of the anti-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_PuMdlt

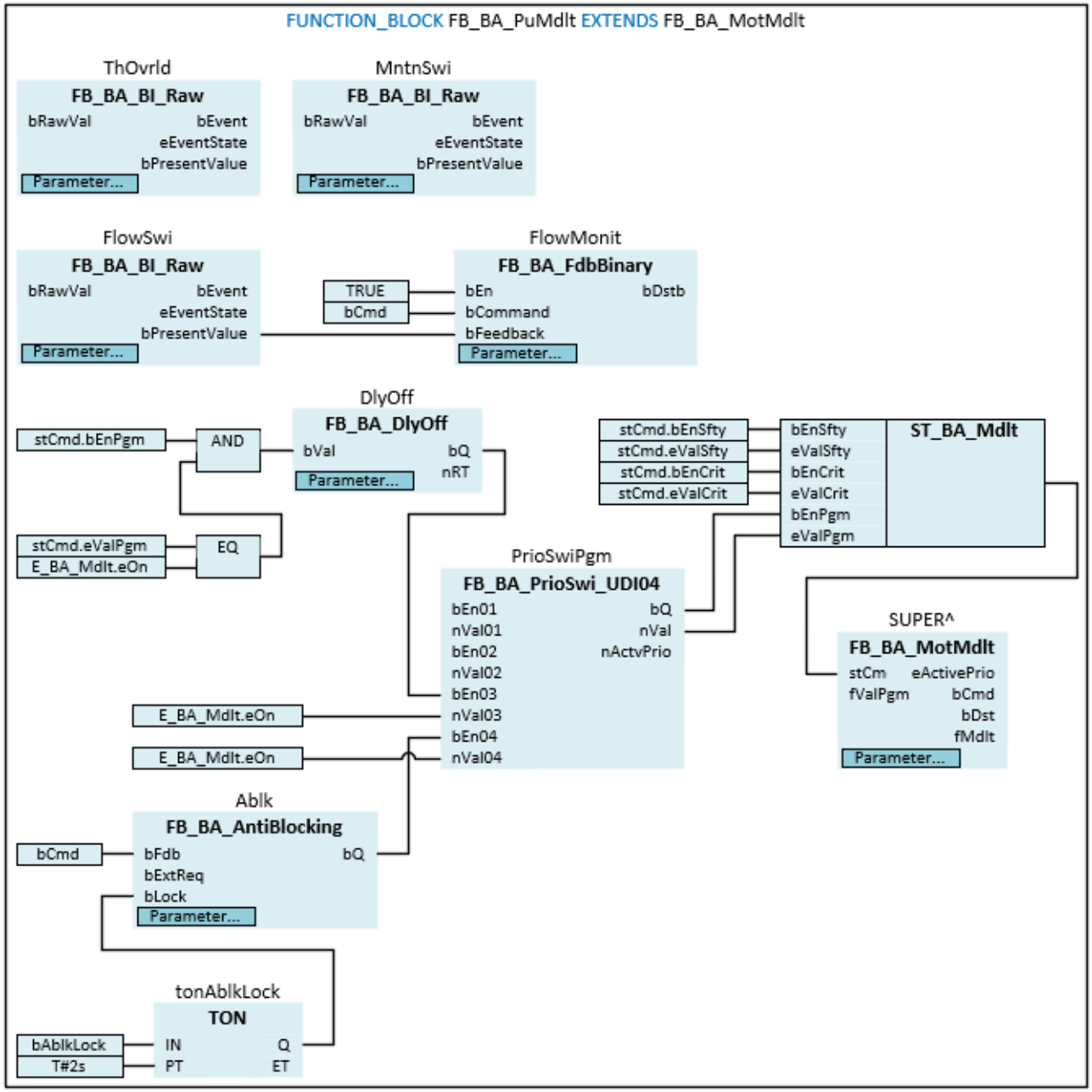


The template is used to control a speed-controlled pump with analog and binary inputs and outputs. It mainly consists of the base classFB_BA_MotMdlt [▶ 1070], the switch-off delay DlyOff [▶ 1053] and the anti-blocking protection function Ablk [▶ 997]. The template *FlowMonit* stands for the monitoring 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_MotMdlt [▶ 1070] or internally by the anti-blocking protection function Ablk [▶ 997]. The external request via the priority "program" is switched off with a delay by the template DlyOff [▶ 1053].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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_UDI04;
  tonAblkLock    : TON;
END_VAR
```

Inputs

Name	Type	Description
bAblkLock	BOOL	A TRUE at this input variable interrupts the anti-blocking protection function Ablk [▶ 997]. It should be prevented that pumps and valves get an anti-blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
ThOvrld	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a maintenance switch.
FlowSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a flow monitor.
FlowMonit	FB_BA_FdbBinary [▶ 1018]	Template for monitoring the flow monitor.
DlyOff	FB_BA_DlyOff [▶ 1053]	The template serves as a delayed pump shutdown.
Ablk	FB_BA_AntiBlocking [▶ 997]	Anti-blocking protection.

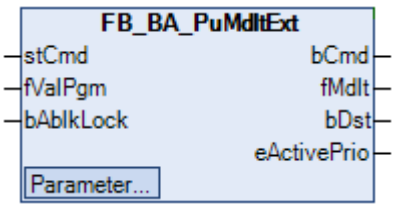
Variables

Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch *PrioSwiPgm* uses the switch-off delay DlyOff, the anti-blocking protection function Ablk [▶ 997] and the priority program of the command structure *stCmd* of the base class FB_BA_Mot1st [▶ 1060] to determine the current switch value for the base class FB_BA_Mot1st [▶ 1060].
tonAblkLock	TON	Switch-off delay of the anti-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_PuMdltExt

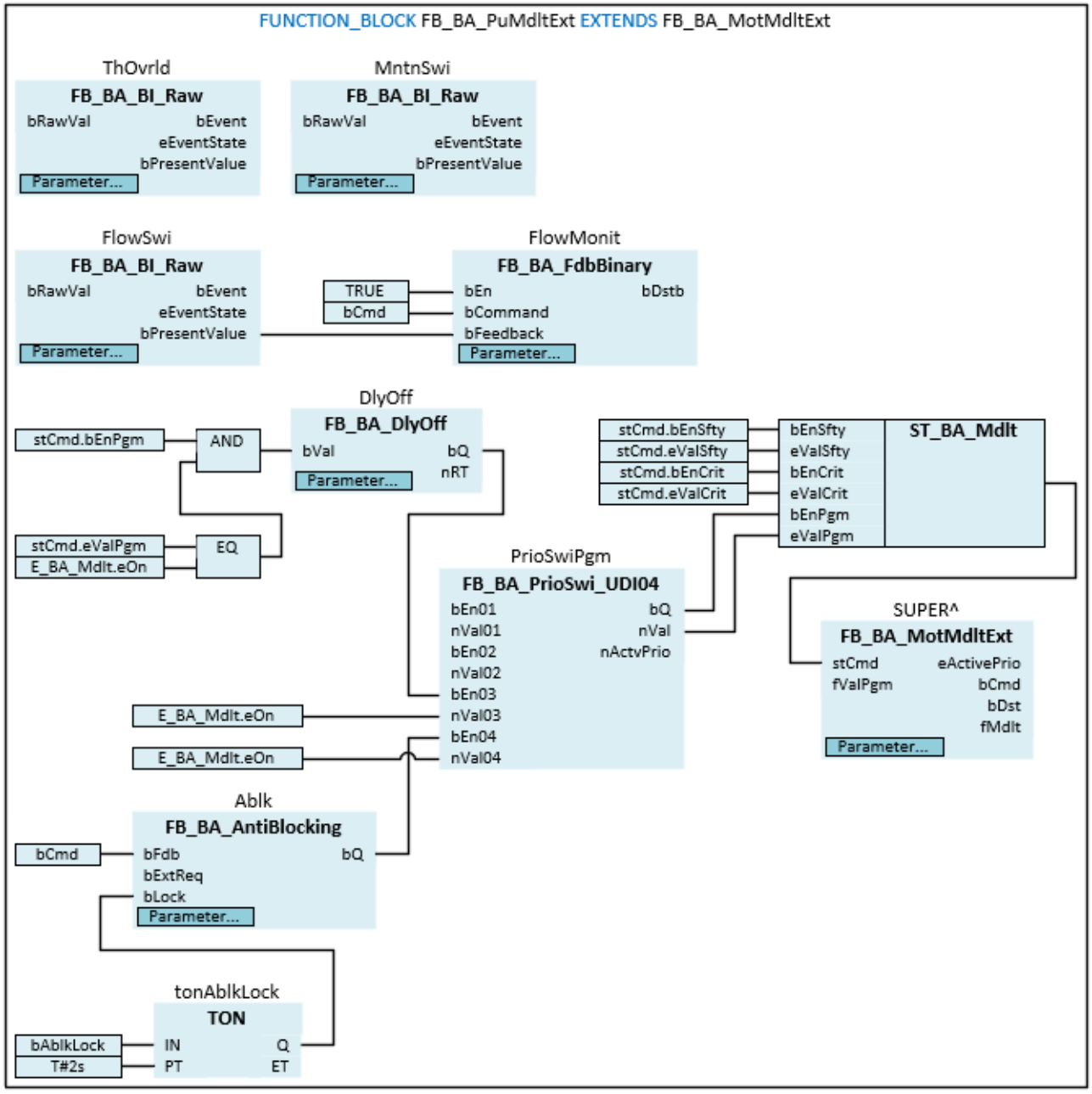


The template is used to control a speed-controlled pump with analog and binary inputs and outputs. It mainly consists of the base class FB_BA_MotMdltExt [▶ 1073], the switch-off delay DlyOff [▶ 1053] and the anti-blocking protection function Ablk [▶ 997]. The template *FlowMonit* stands for the monitoring 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_MotMdltExt [▶ 1073] or internally by the anti-blocking protection function Ablk [▶ 997]. The external request via the priority "program" is switched off with a delay by the template DlyOff [▶ 1053].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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_UDI04;
  tonAblkLock     : TON;
END_VAR
```

Inputs

Name	Type	Description
bAblkLock	BOOL	A TRUE at this input variable interrupts the anti-blocking protection function Ablk [▶ 997]. It should be prevented that pumps and valves get an anti-blocking protection pulse at the same time.

Inputs CONSTANT

Name	Type	Description
ThOvrld	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process the "Thermal overload" fault.
MntnSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a maintenance switch.
FlowSwi	FB_BA_BI_Raw [▶ 206]	The binary input object is used to process a flow monitor.
FlowMonit	FB_BA_FdbBinary [▶ 1018]	Template for monitoring the flow monitor.
DlyOff	FB_BA_DlyOff [▶ 1053]	The template serves as a delayed pump shutdown.
Ablk	FB_BA_AntiBlocking [▶ 997]	Anti-blocking protection.

Variables

Name	Type	Description
PrioSwiPgm	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch *PrioSwiPgm* uses the switch-off delay DlyOff, the anti-blocking protection function Ablk [▶ 997] and the priority program of the command structure *stCmd* of the base class FB_BA_Mot1st [▶ 1060] to determine the current switch value for the base class FB_BA_Mot1st [▶ 1060].
tonAblkLock	TON	Switch-off delay of the anti-blocking protection pulse.

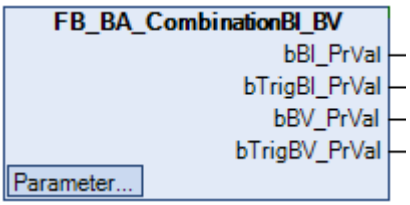
Requirements

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_CombinationBI_BV



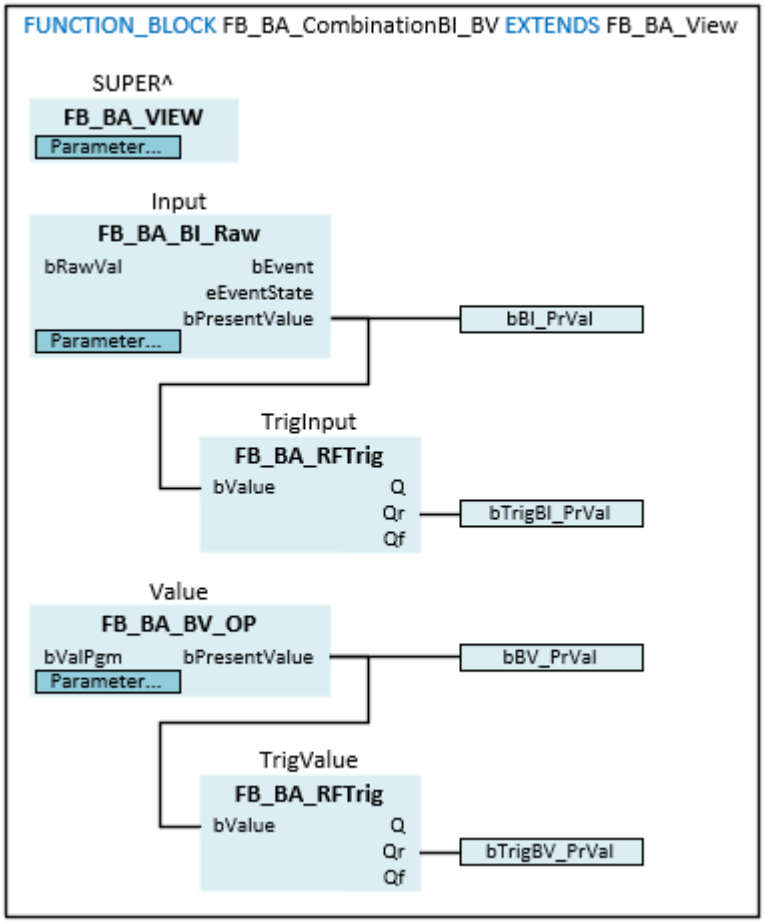
The template is a combination of a BI and BV object.

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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 [▶ 206]	Binary input object for displaying a process value.
Value	FB_BA_BV_Op [▶ 216]	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 *Input*.
bTrigBI_PrVal	BOOL	Current value of the output *Qr* of the function block *TrigInput*.
bBV_PrVal	BOOL	Current value of the binary value object *Value*
bTrigBV_PrVal	BOOL	Current value of the output *Qr* of the function block *TrigValue*.

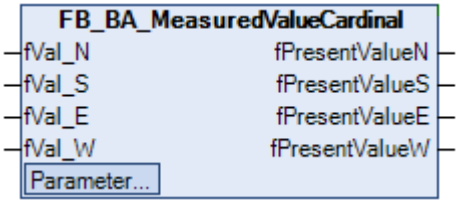
Variables

Name	Type	Description
TrigInput	FB_BA_RFTrig	The function block generates a rising edge from the output signal of the binary input object *Input*.
TrigValue	FB_BA_RFTrig	The function block generates a rising edge from the output signal of the binary value object *Value*.

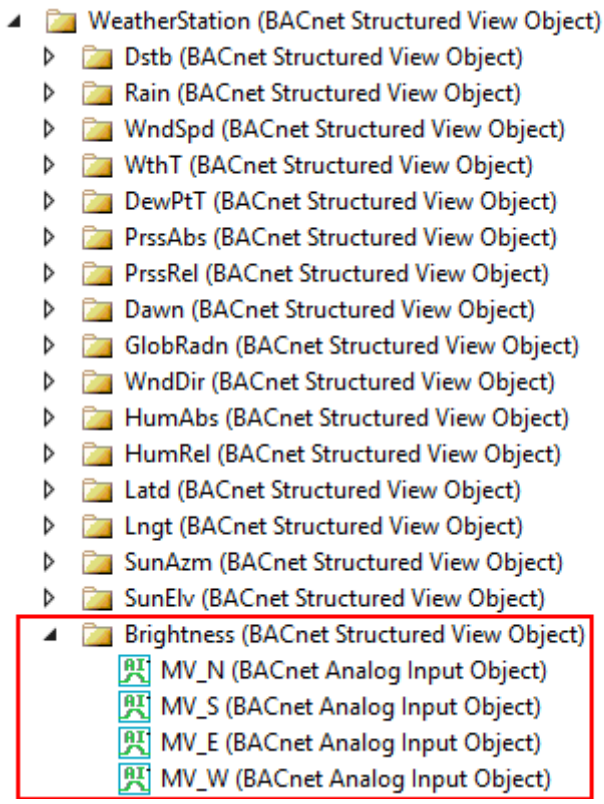
Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.6.2 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.

Syntax

```
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 [▶ 191]	Analog input object for the "North" value.
MV_S	FB_BA_AI [▶ 191]	Analog input object for the "South" value.
MV_E	FB_BA_AI [▶ 191]	Analog input object for the "East" value.
MV_W	FB_BA_AI [▶ 191]	Analog input object for the "West" value.

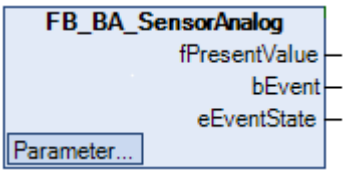
Outputs

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

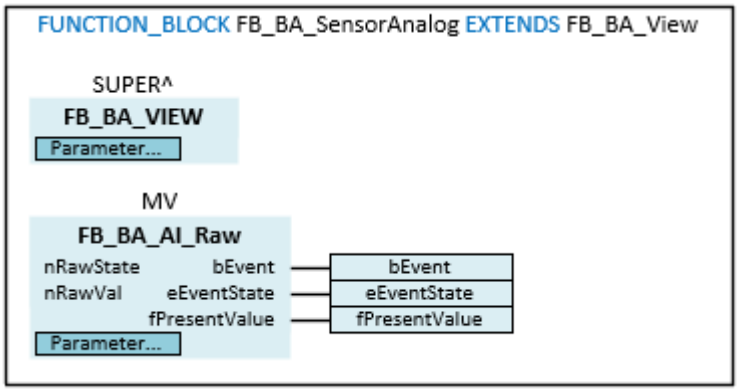
6.1.4.2.2.6.3 FB_BA_SensorAnalog



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 initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_SensorAnalog EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  MV                : FB_BA_AI_Raw;
END_VAR
```

```
VAR_OUTPUT
  fPresentValue     : REAL;
  bEvent            : BOOL;
  eEventState       : E_BA_EventState;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
MV	FB_BA_AI_Raw [▶ 195]	Analog input object for displaying a process value.

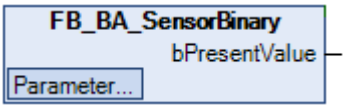
Outputs

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

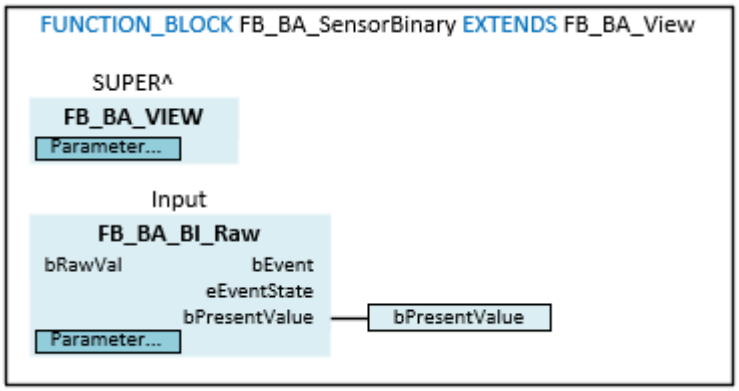
6.1.4.2.2.6.4 FB_BA_SensorBinary



The function block Input logs a binary input value from an I/O bus terminal and outputs it as a boolean process value.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_SensorBinary EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  Input             : FB_BA_BI_Raw;
END_VAR
```

```
VAR_OUTPUT
  bPresentValue    : BOOL;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
Input	FB_BA_BI_Raw [▶ 206]	Binary input object for displaying a process value.

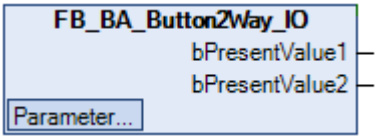
Outputs

Name	Type	Description
bPresentValue	BOOL	Current value of the binary input object *Input* 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

6.1.4.2.2.6.5 FB_BA_Button2Way



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.

The initialization of the template takes place within the method FB_Init.

Inheritance hierarchy

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 264]

FB_BA_View [▶ 241]

Syntax

```
FUNCTION_BLOCK FB_BA_Button2Way EXTENDS FB_BA_VIEW
VAR_INPUT
END_VAR
VAR_INPUT CONSTANT
  Swi1              : FB_BA_BI_Raw;
  Swi2              : FB_BA_BI_Raw;
END_VAR
VAR_OUTPUT
  bPresentValue1    : BOOL;
  bPresentValue2    : BOOL;
END_VAR
```

Inputs CONSTANT

Name	Type	Description
Swi1	FB_BA_BI_Raw [▶ 206]	Binary input object for switch1.
Swi2	FB_BA_BI_Raw [▶ 206]	Binary input object for switch2.

Outputs

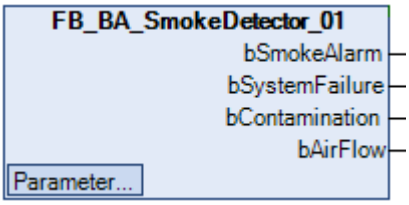
Name	Type	Description
bPresentValue1	BOOL	Output: state switch 1.
bPresentValue2	BOOL	Output: state switch 2.

Requirements

Development environment	Necessary function
TwinCAT from v3.1.4024.35	TF8040 | TwinCAT Building Automation from V5.0.0.0

6.1.4.2.2.7 SmokeDetector

6.1.4.2.2.7.1 FB_BA_SmokeDetector_01

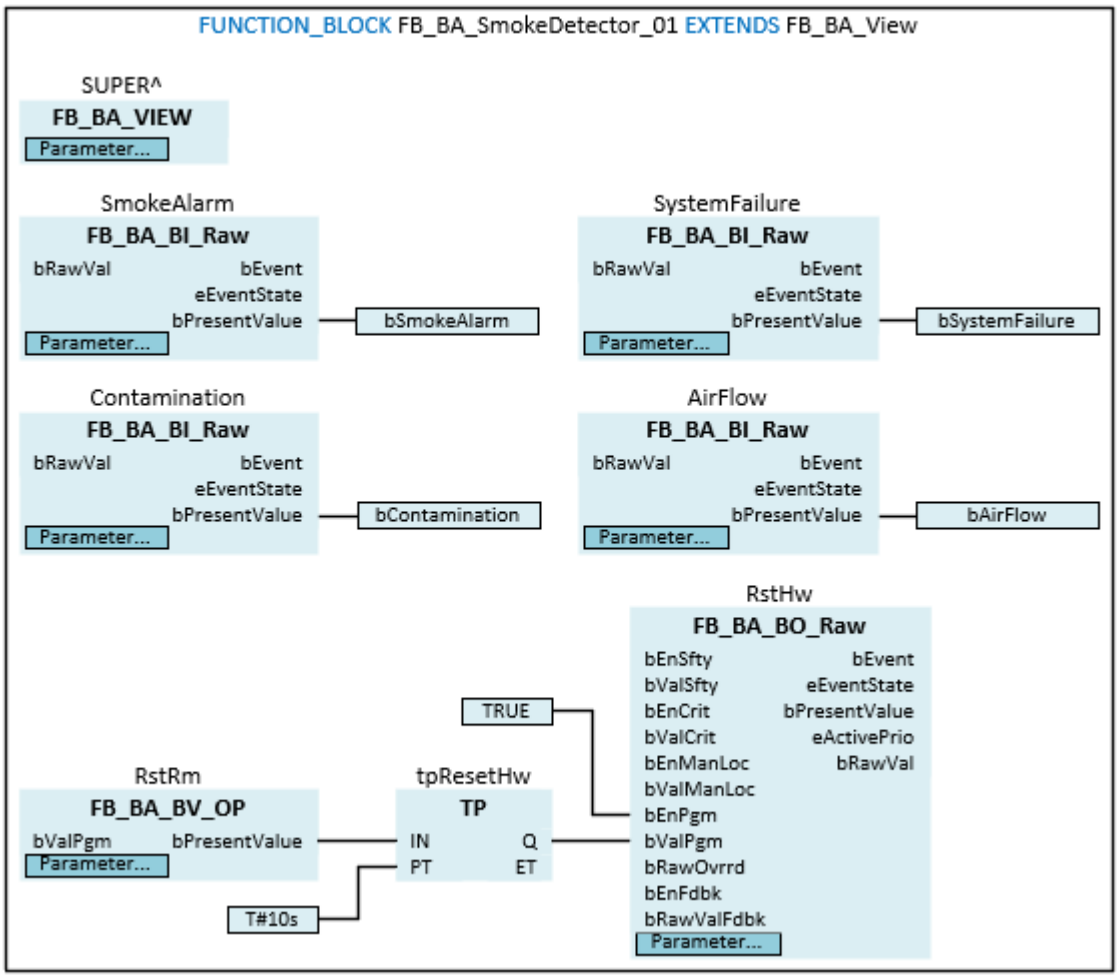


The template is used to display faults and warnings of a duct smoke detector.

The binary output object *RstHw* can be used to reset these messages by remote release.

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 *SmokeAlarm*.
bSystemFailure	BOOL	Current value of the binary input object *SystemFailure*.
bContamination	BOOL	Current value of the binary input object *Contamination*.
bAirFlow	BOOL	Current value of the binary input object *AirFlow*.

Inputs CONSTANT

Name	Type	Description
SmokeAlarm	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process a smoke alarm.
SystemFailure	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process a system fault of the duct smoke detector.
Contamination	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process a pollution of the duct smoke detector.
AirFlow	FB_BA_BI_Raw [▶ 206]	Binary input object is used to process insufficient air flow.
RstHw	FB_BA_BO_Raw [▶ 211]	Binary output object which triggers a remote release on the smoke detector.
RstRm	FB_BA_BV_Op [▶ 216]	The binary value object *RstRm* triggers a remote release on the smoke detector from the management level. The *RstRm* object is initialized as a button object by the additional parameter eToggleMode.

Variables

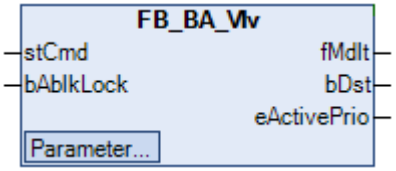
Name	Type	Description
tpResetHw	TP	The timing element extends the acknowledgement pulse for interfacing relay circuits (e.g. frost protection relay).

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

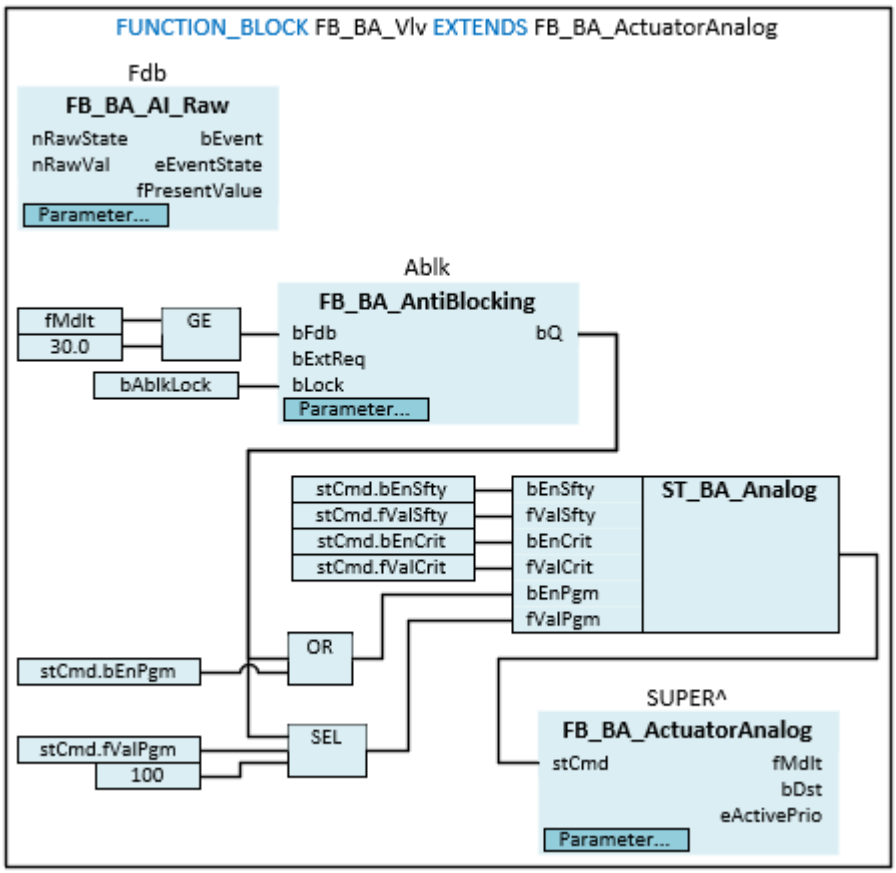


The template is used to control a continuous valve with analog inputs and outputs. It mainly consists of the base class FB_BA_ActuatorAnalog [▶ 982], the anti-blocking protection function Ablk [▶ 997] and the AI object *Fdb* for recording the position feedback from the valve.

The valve is switched on externally by the priorities of the command structure *stCmd* of the base class FB_BA_ActuatorAnalog [▶ 982] or internally by the anti-blocking protection function Ablk [▶ 997].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
FUNCTION_BLOCK FB_BA_Vlv EXTENDS FB_BA_ActuatorAnalog
VAR_INPUT
  bAblkLock    : BOOL;
END_VAR
VAR_INPUT CONSTANT
  Ablk         : FB_BA_AntiBlocking;
  Fdb          : FB_BA_AI_Raw;
END_VAR
```

Inputs

Name	Type	Description
bAblkLock	BOOL	A TRUE at this input variable interrupts the anti-blocking protection function Ablk [▶ 997]. It should be prevented that pumps and valves get an anti-blocking protection pulse at the same time.

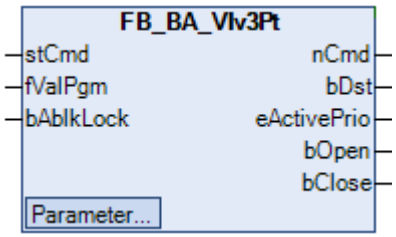
Inputs CONSTANT

Name	Type	Description
Ablk	FB_BA_AntiBlocking [▶ 997]	Anti-blocking protection.
Fdb	FB_BA_AI_Raw [▶ 195]	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

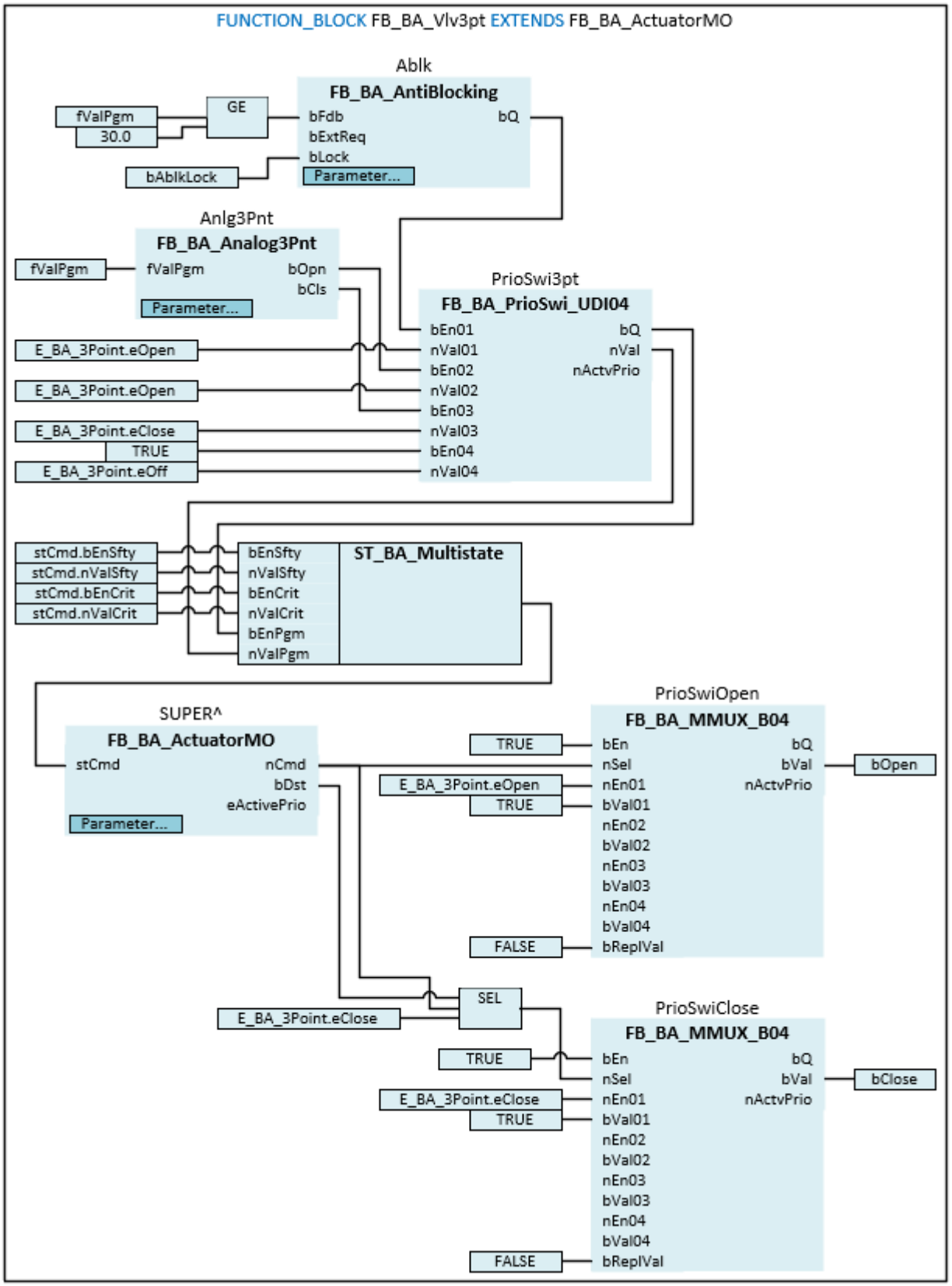


The template is used to control a three-point valve. It mainly consists of the base class FB_BA_ActuatorMO [▶ 991], the function block Anlg3Pnt [▶ 407] for the analog conversion of the input signal *fValPgm* into a three-point signal and the anti-blocking protection function Ablk [▶ 997].

The three-point valve is switched on externally by the priorities of the command structure *stCmd* of the base class FB_BA_ActuatorMO [▶ 991] or internally by the anti-blocking protection function Ablk [▶ 997].

The initialization of the template takes place within the method FB_Init.

Block diagram



Syntax

```
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 anti-blocking protection function Ablk [▶ 997]. It should be prevented that pumps and valves get an anti-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 [▶ 995]	The template *Anlg3Pnt* converts the analog input signal *fValPgm* into a three-point signal.
Ablk	FB_BA_AntiBlocking [▶ 997]	Anti-blocking protection.

Variables

Name	Type	Description
PrioSwi3pt	FB_BA_PrioSwi_UDI04 [▶ 433]	The priority switch *PrioSwi3pt* determines the current switching value for the priority "Program" of the command structure *stCmd* on the basis of the analog 3-point converter *Anlg3Pnt* and the anti-blocking protection function Ablk [▶ 997].
PrioSwiOpen	FB_BA_MMUX_B04 [▶ 428]	The multiplexer *PrioSwiOpen* receives the numeric switch value *nCmd* from the base class FB_BA_ActuatorMO [▶ 991] and converts the switch value into the 3-point signal *bOpen*.
PrioSwiClose	FB_BA_MMUX_B04 [▶ 428]	The multiplexer *PrioSwiClose* receives the numeric switch value *nCmd* from the base class FB_BA_ActuatorMO [▶ 991] and converts the switch value into the 3-point signal *bClose*.

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 WeatherStation

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:

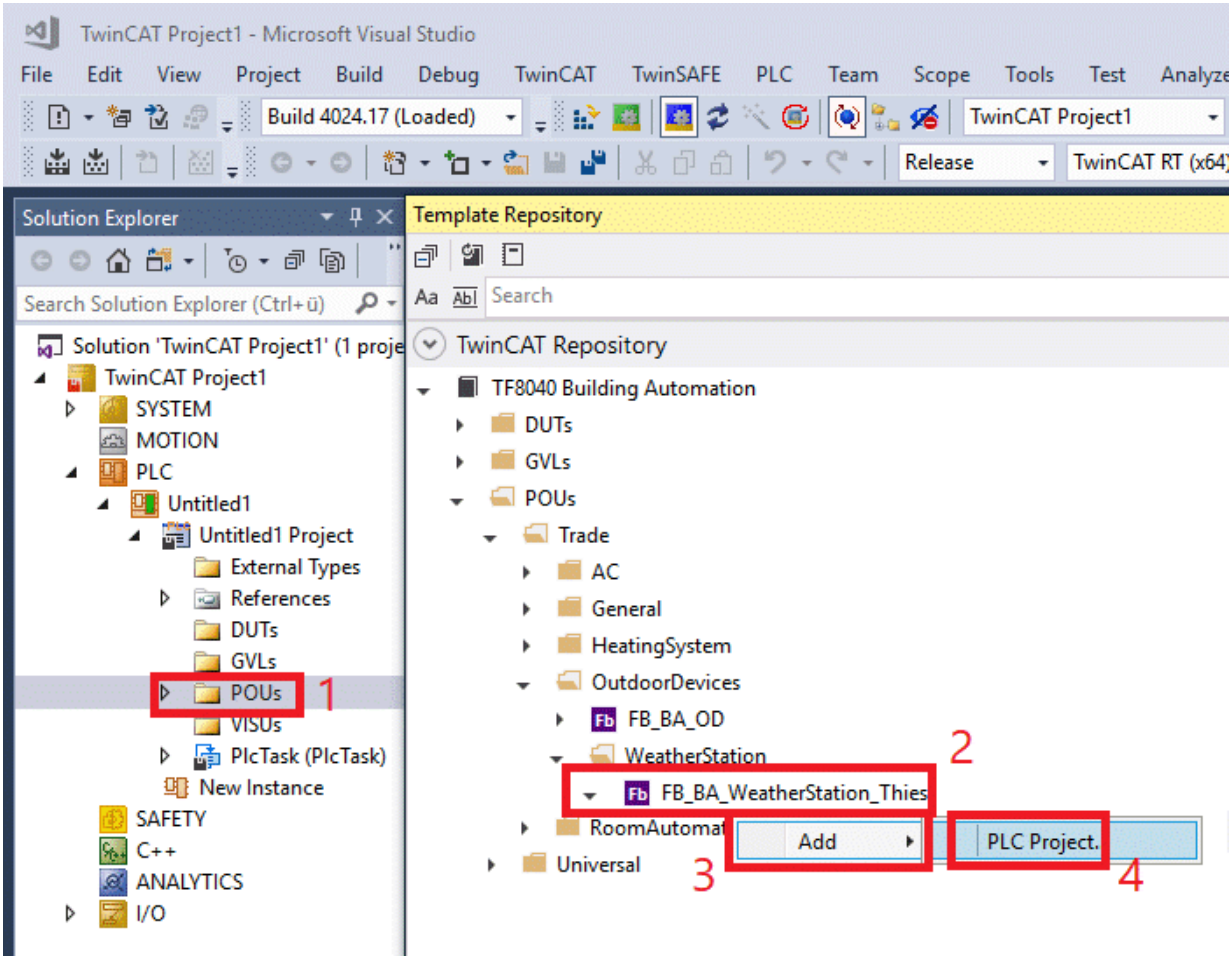
- Tc2_SerialCom

Adding the template:

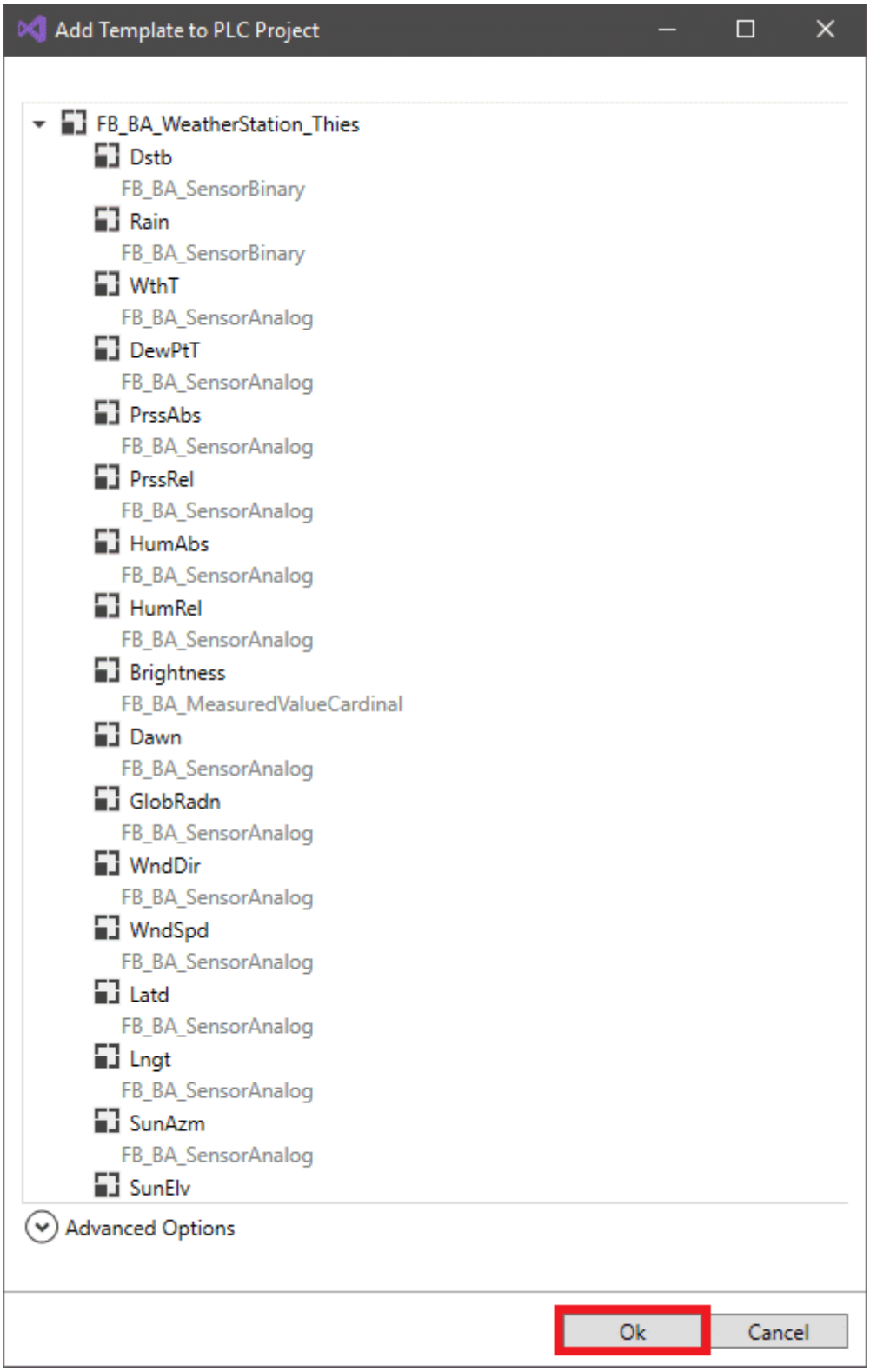
✓ Click on the desired PLC project (1).

1. In the template repository, right-click on the template **FB_BA_WeatherStation_Thies** (2)

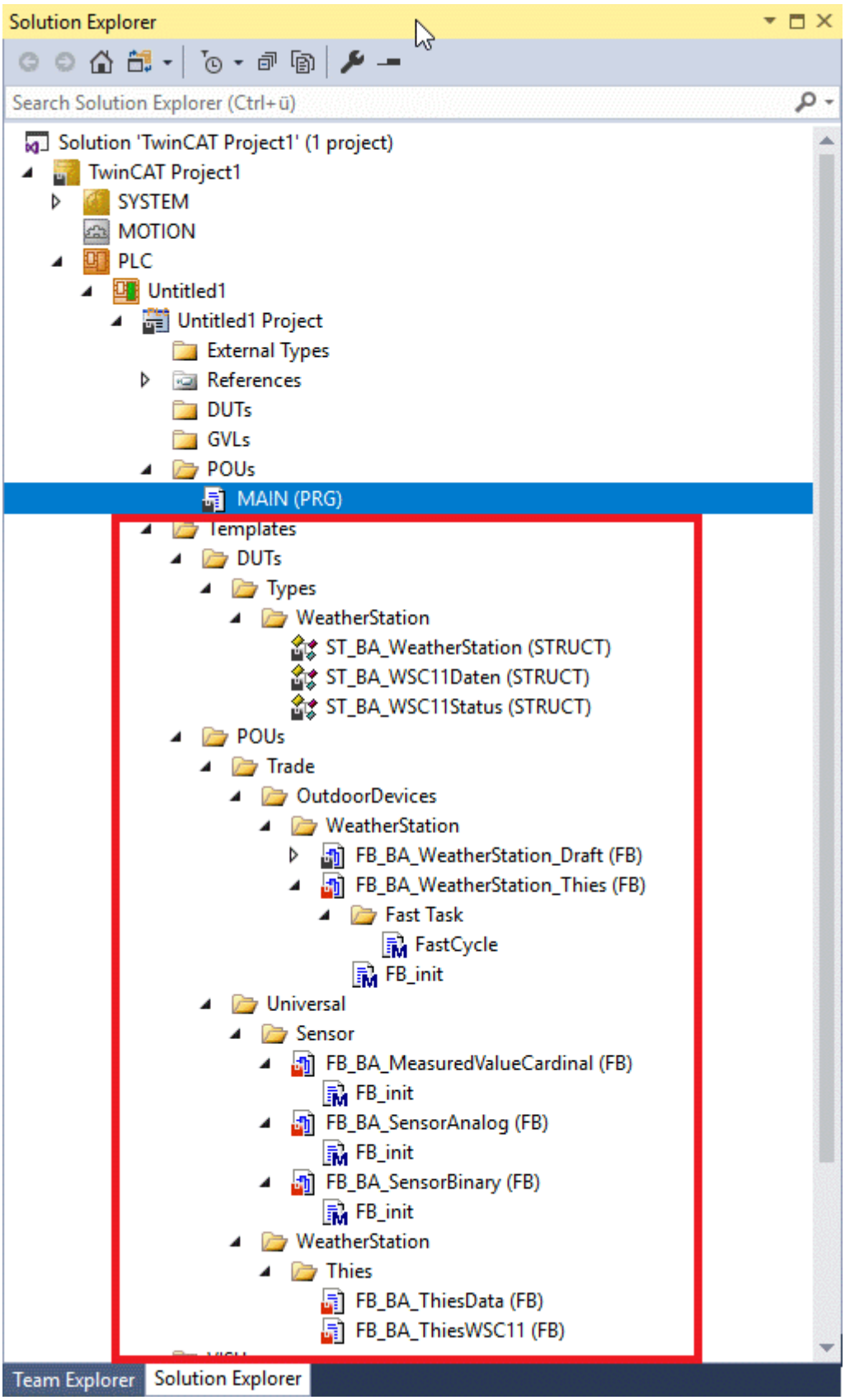
2. Add the template (3) and (4).



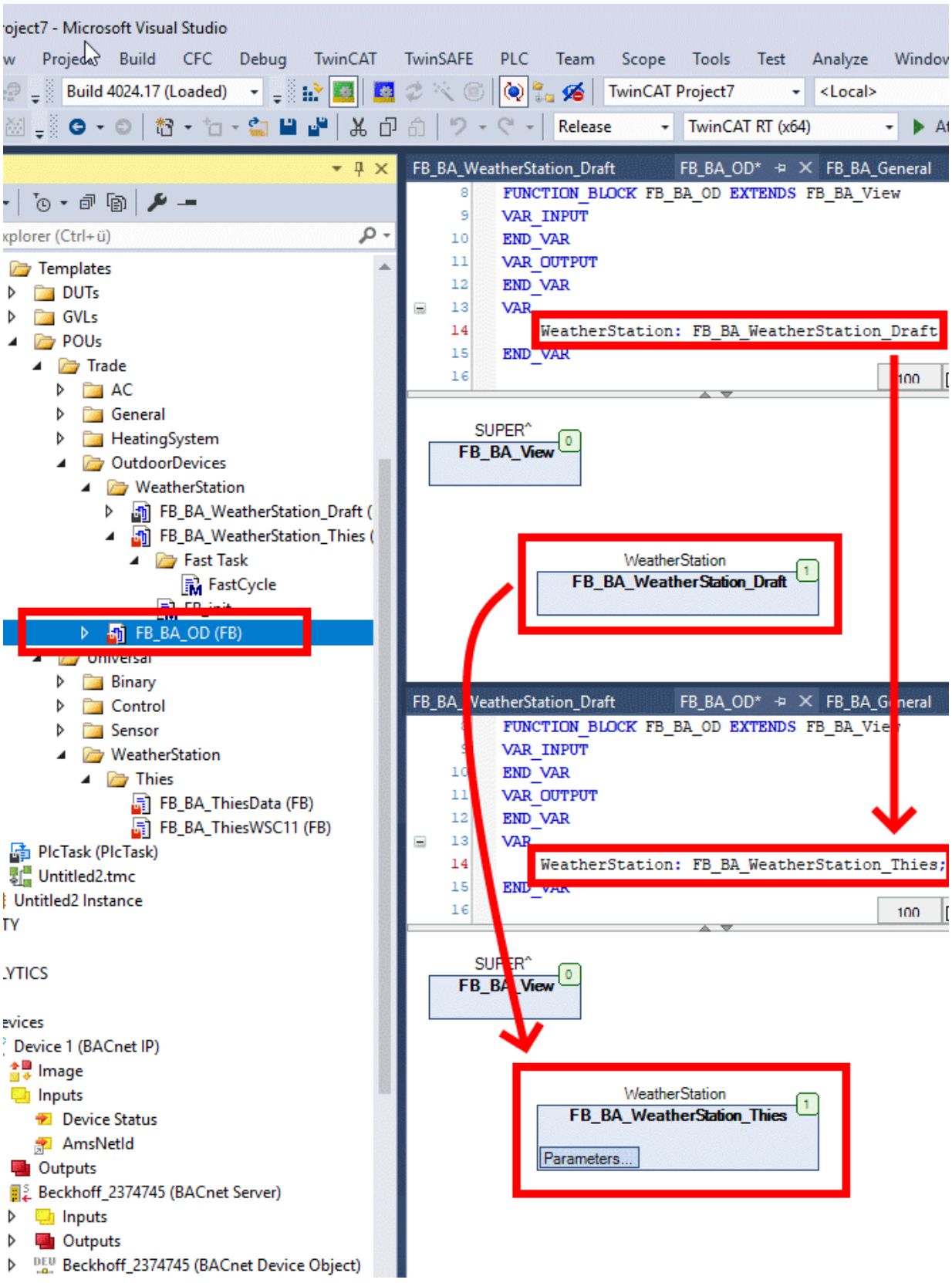
3. The following dialog with the list of additionally implemented function blocks opens. Close the dialog with **OK**.



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

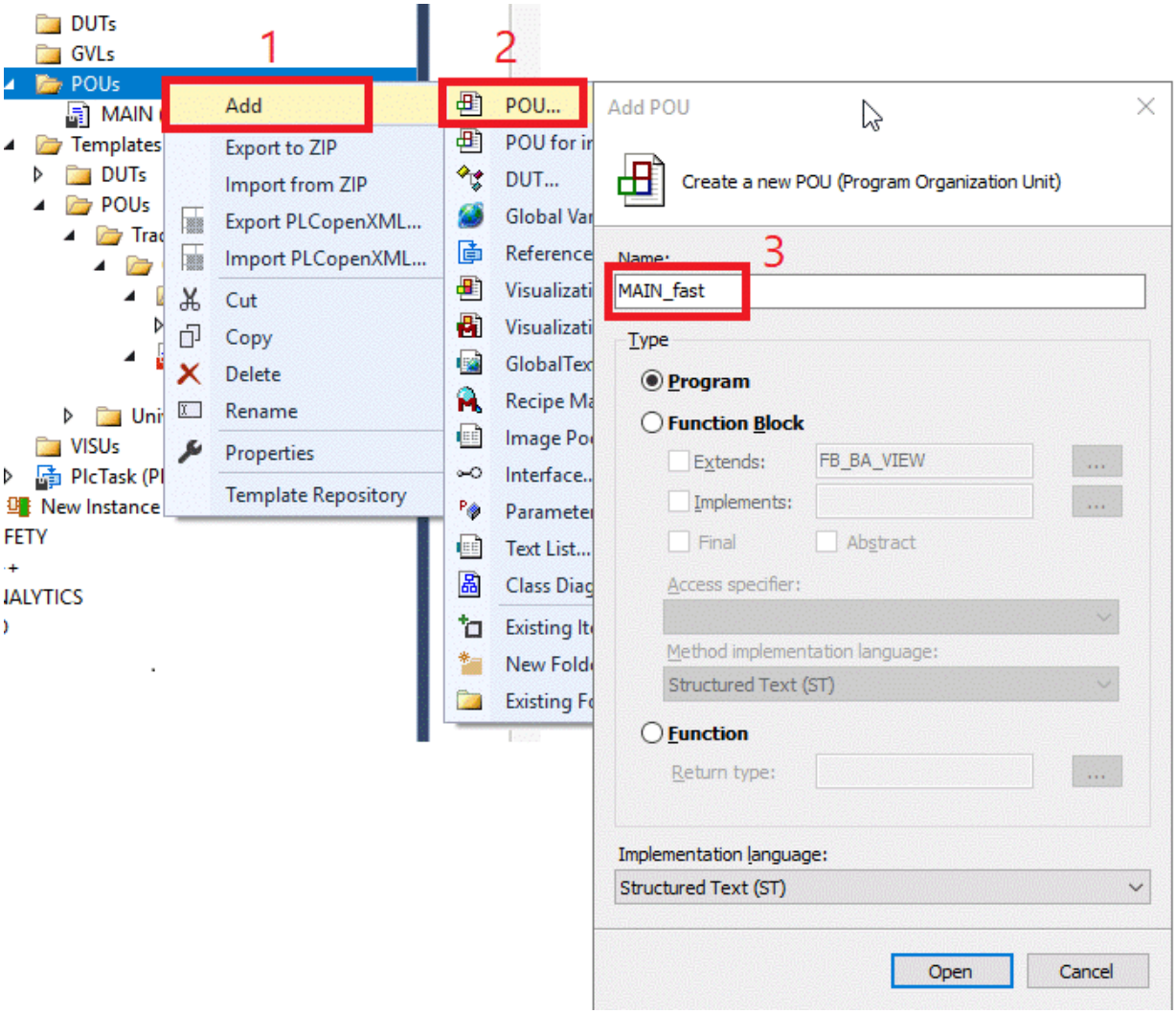
- Data bits: 8
- Parity: none
- Stop bits: 1
- Handshake: RS485 HALFDUPLEX

The process image must be set to 22 bytes beforehand.

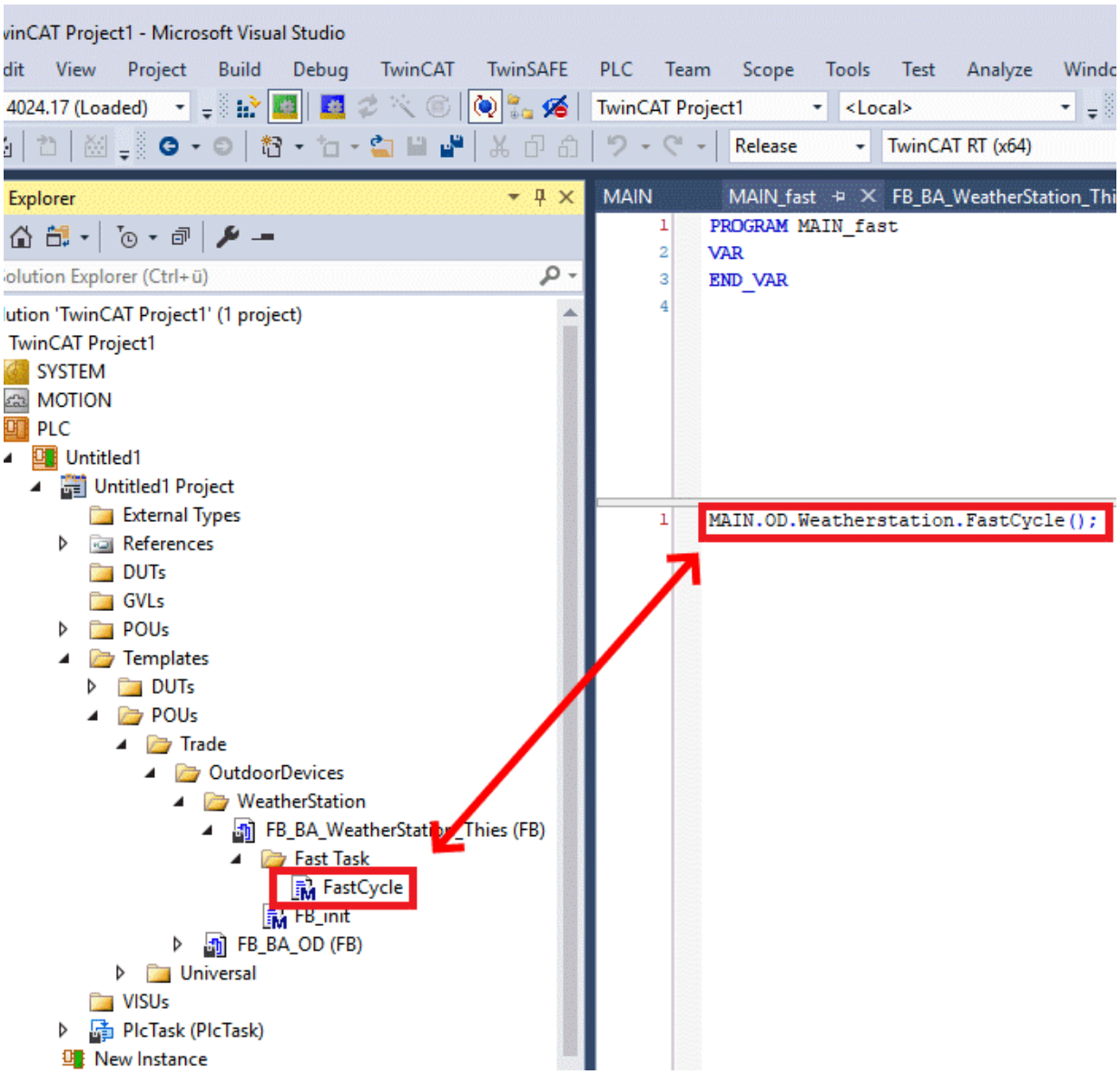
After that the actual communication between terminal and weather station starts.

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)**.**

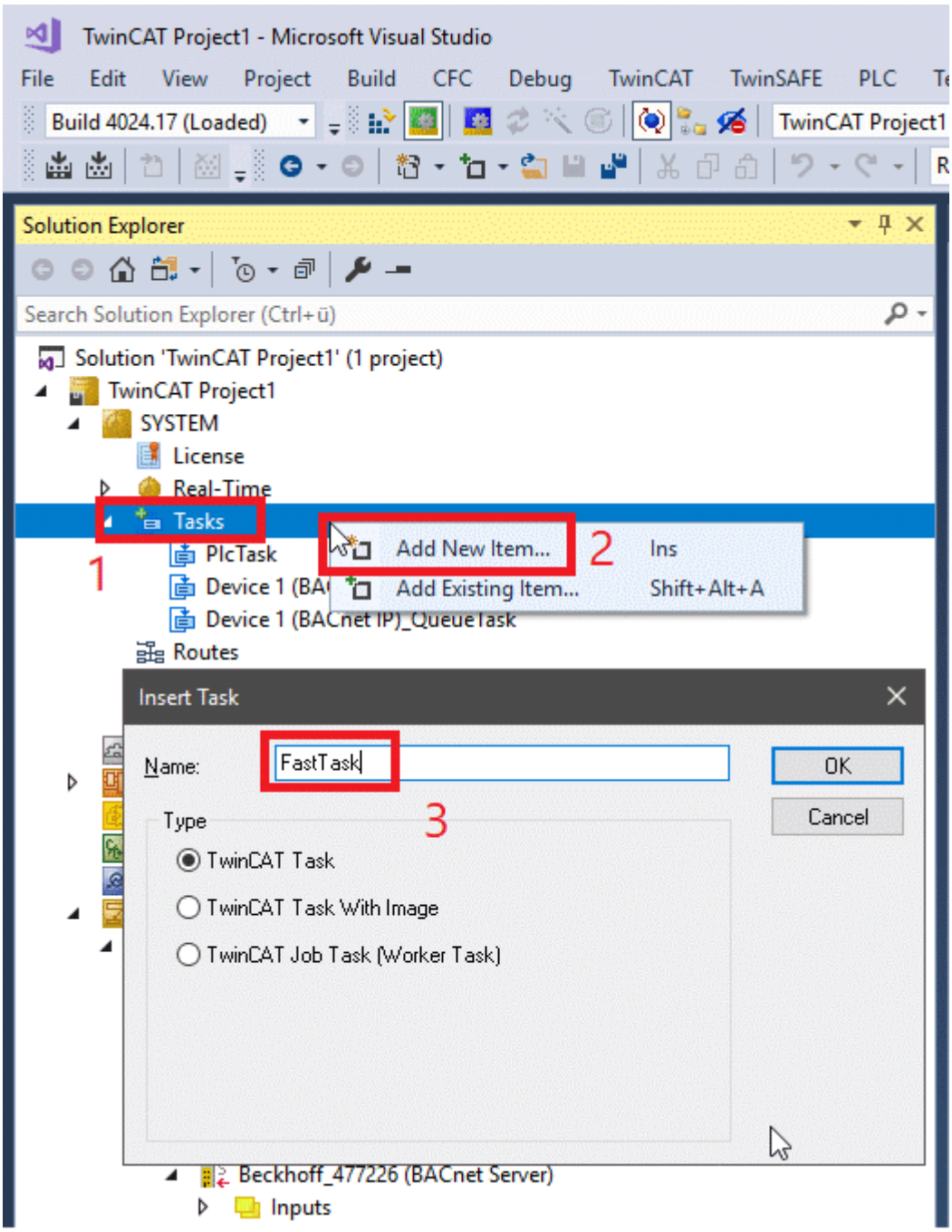


2. A window opens. Enter the name there (example **MAIN_fast**) (3). The POU type is a program in structured text (ST).

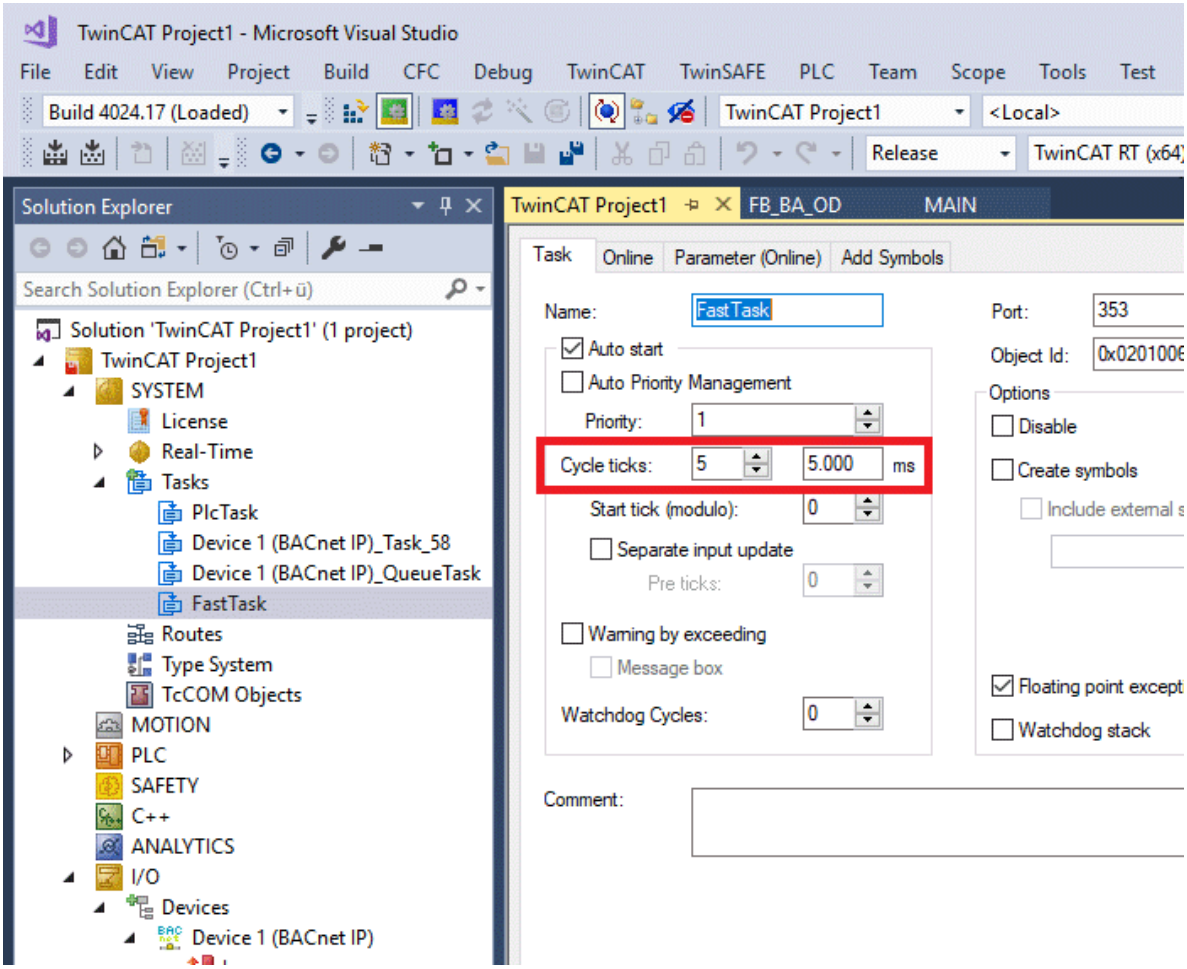


⇨ In this program, the method **FastCycle** is now called.

3. Add a new item (2) by right-clicking on **Tasks** (1). Name it with a meaningful name, for example **FastTask** (3).

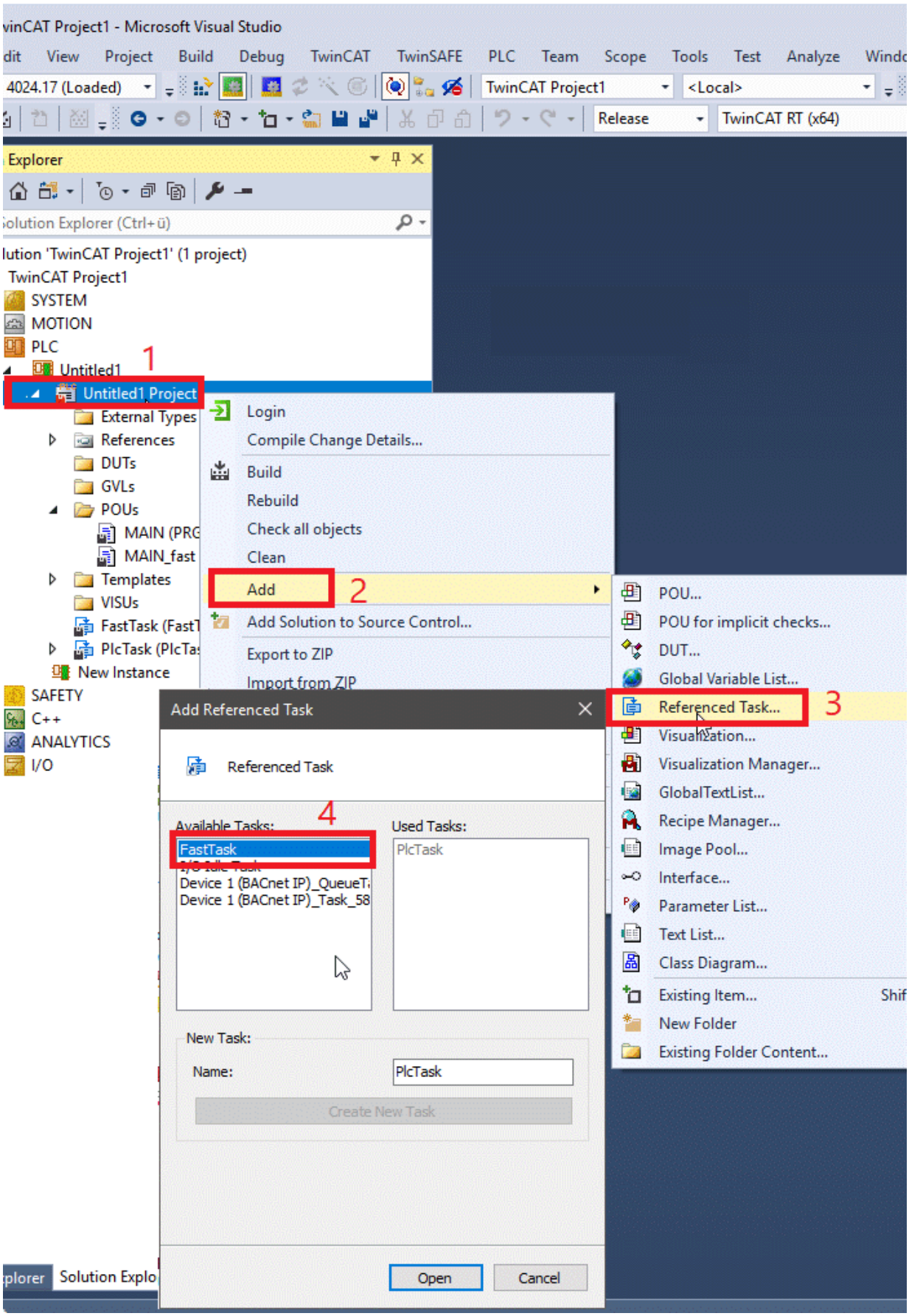


⇨ This new task must now be set to a small cycle time. The 5 ms shown here are a recommendation



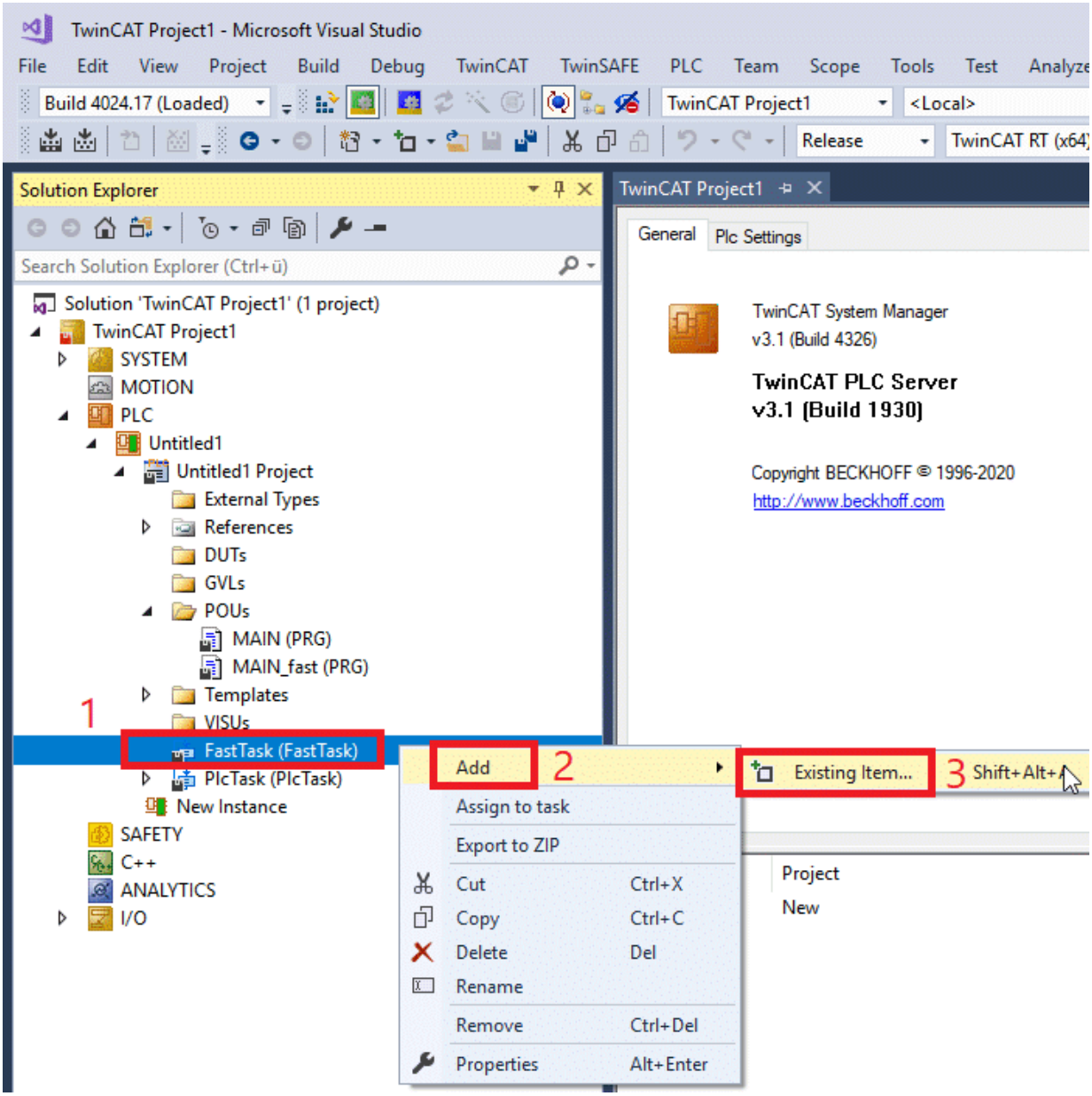
⇨ A task reference must be created so that this task is available to the program part:

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).

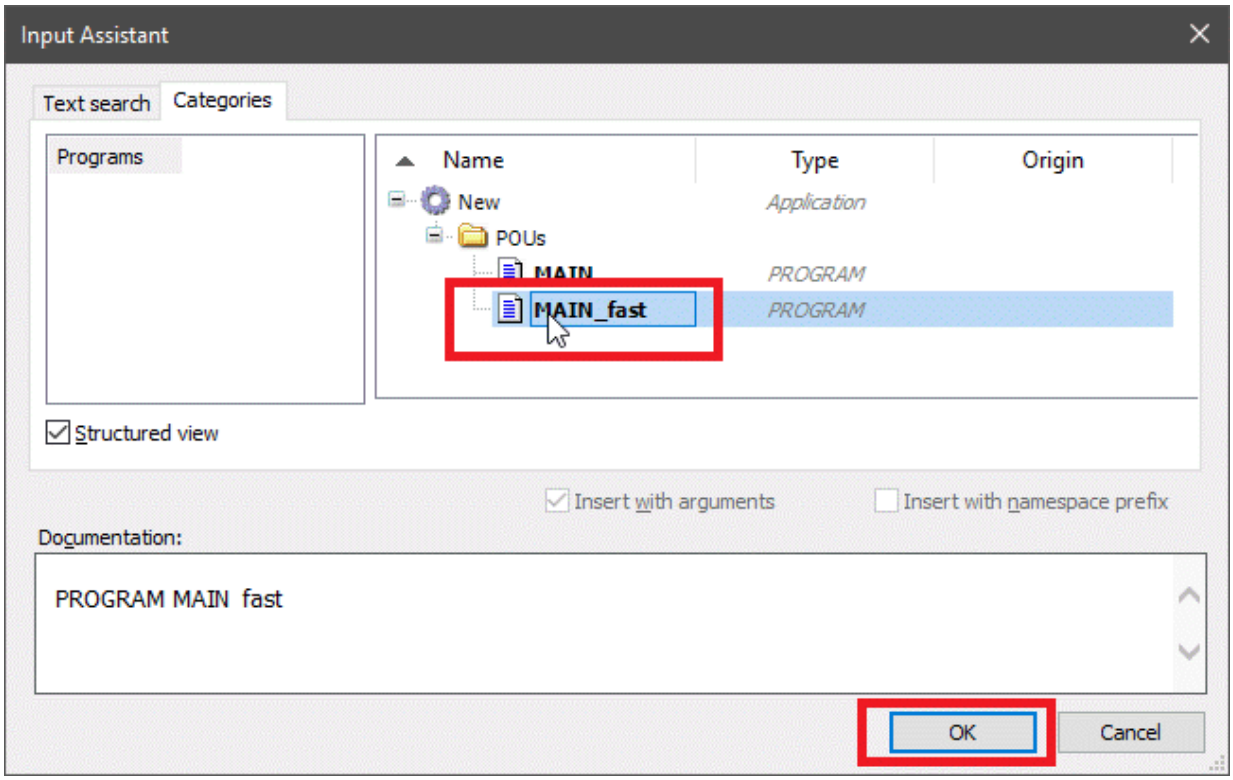


⇨ The task reference now appears in the PLC part below. The function block that is to be called in this task can now be assigned to it.

5. Open a dialog by right-clicking on the task **Fast Task**(1), select **add** (2) and then **existing item** (3).

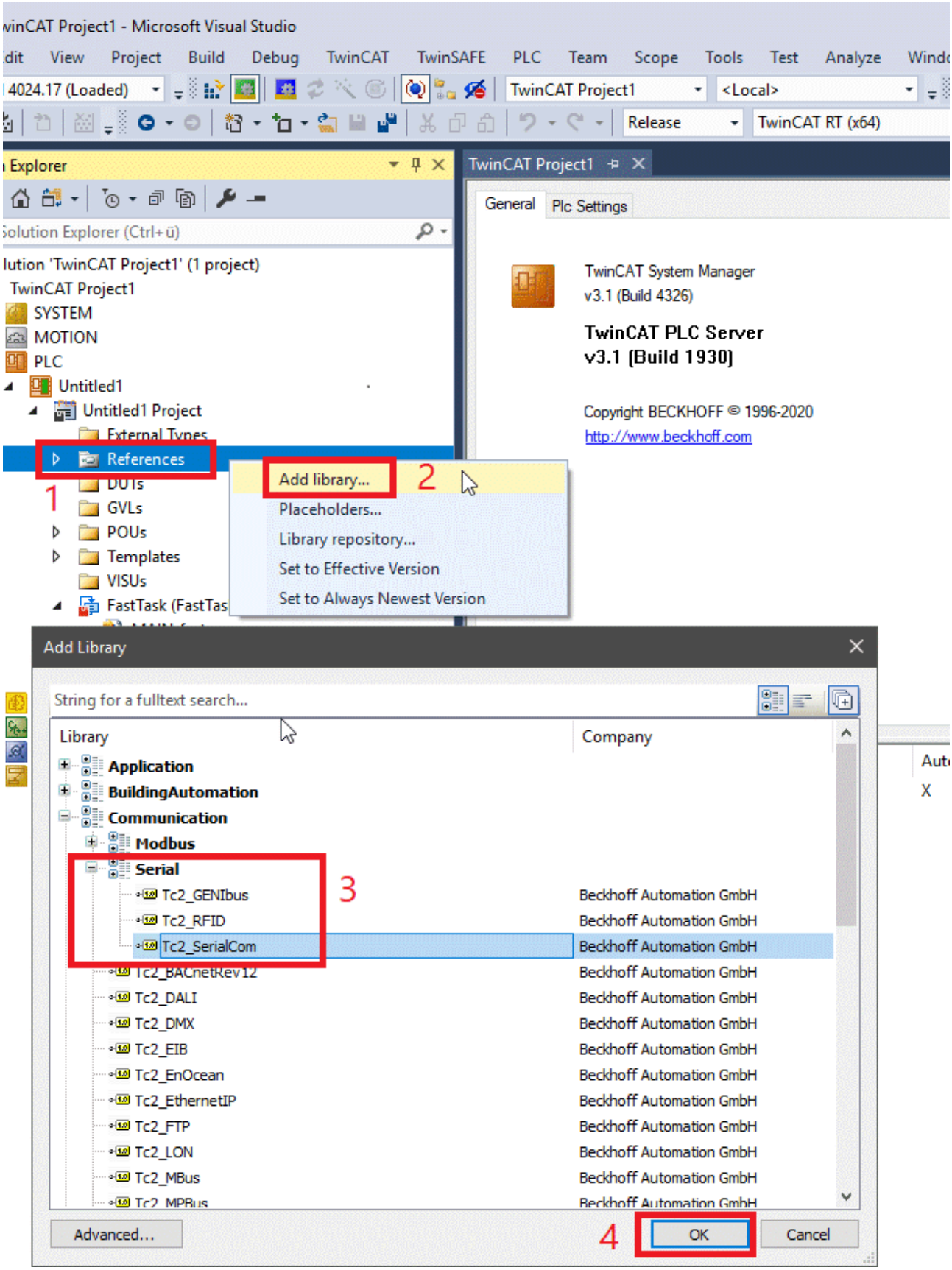


⇨ A window will open where you can select the call program.



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:

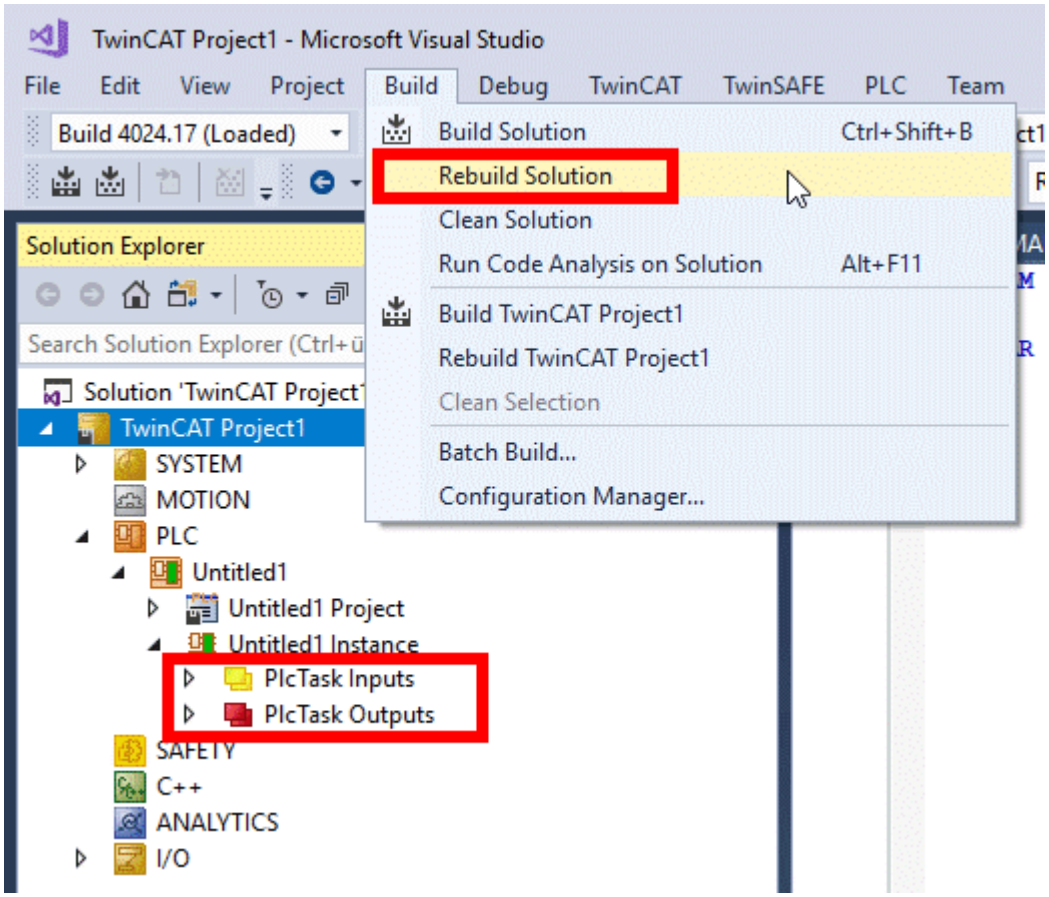


Right-click on "References" (1) and select "Add Library" (2). In the window that opens, select the "Tc2_SerialCom" library (3) and confirm with "OK" (4).

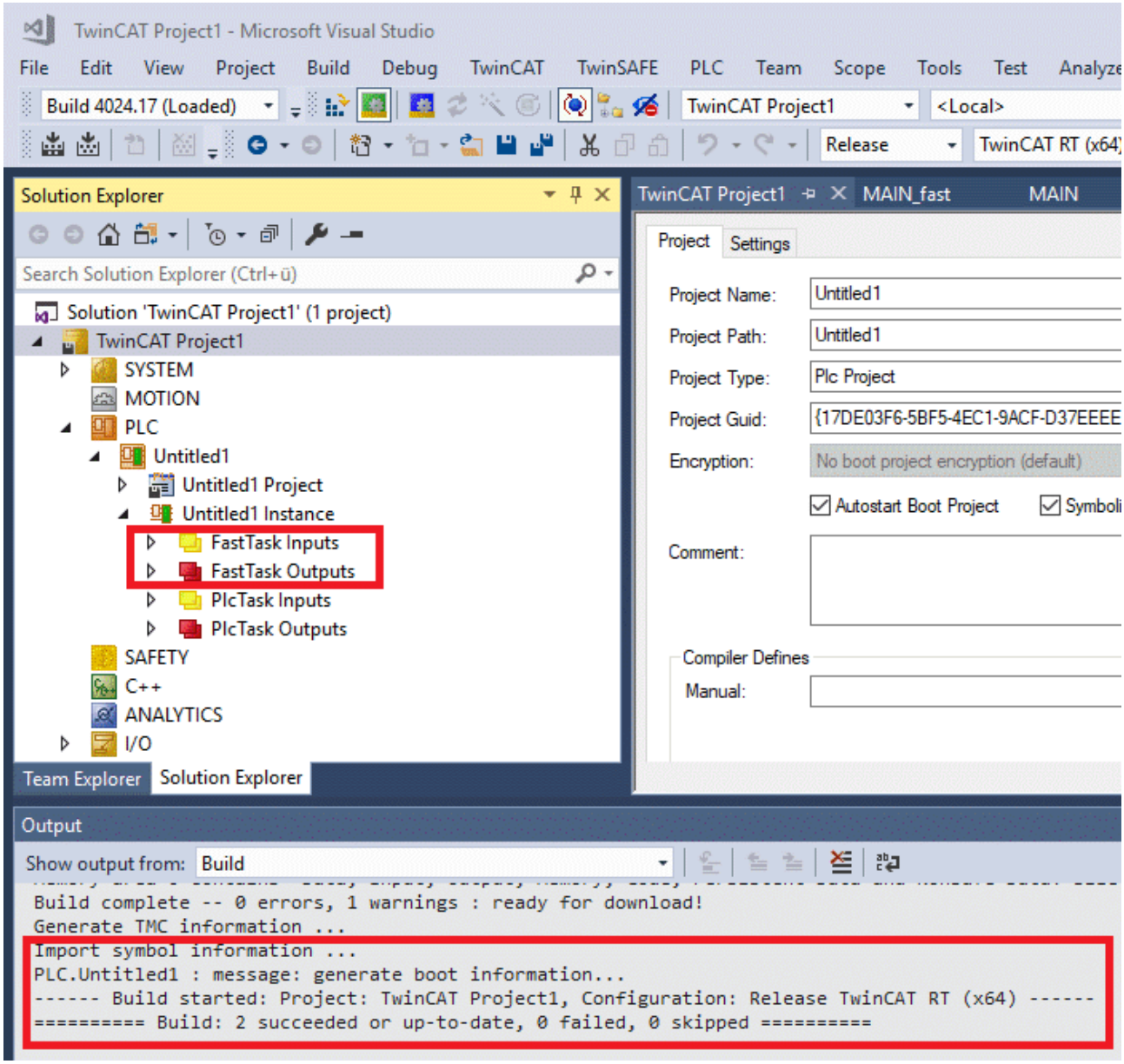
Creating program links

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.

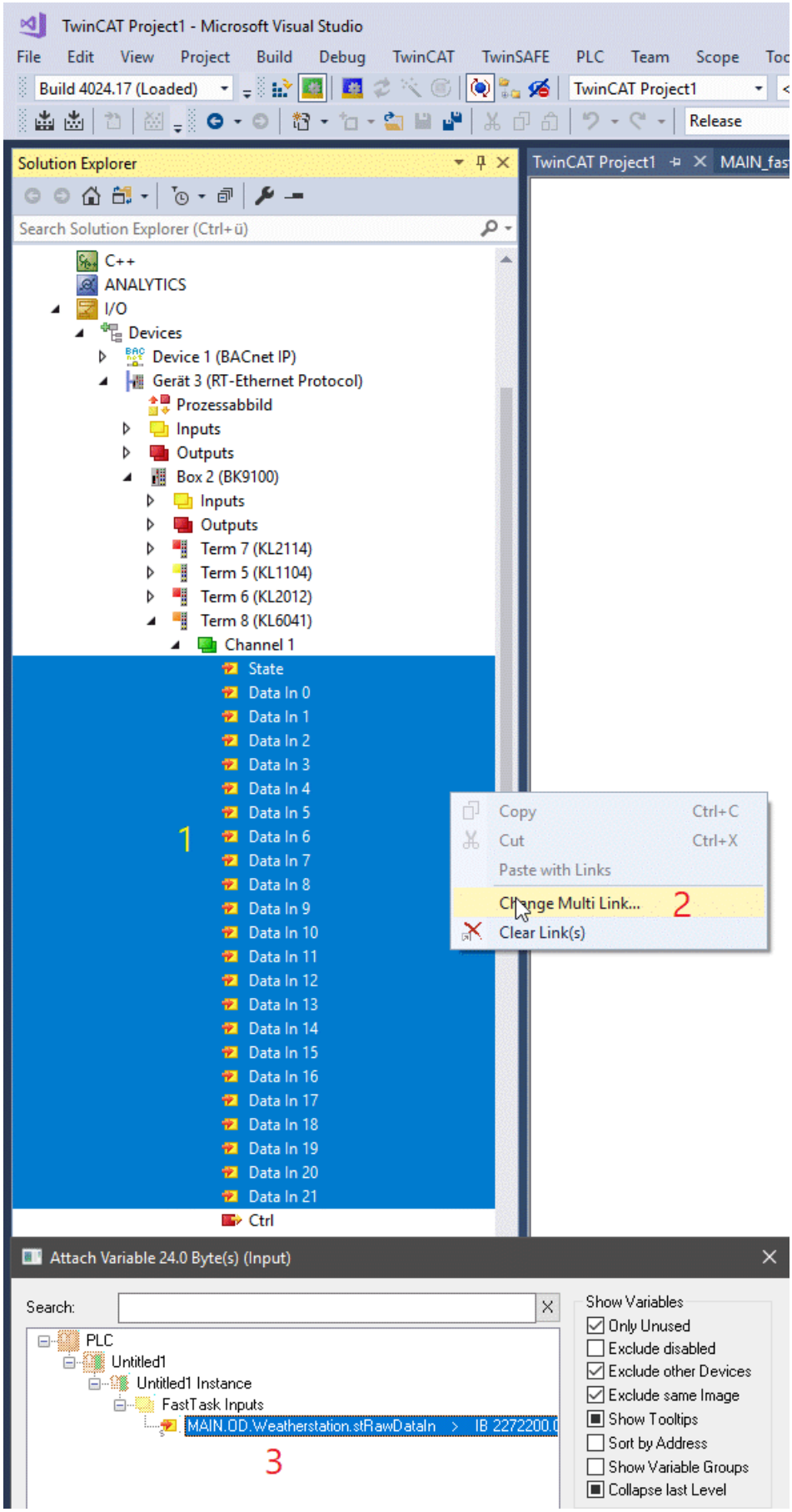
It is important at this point that no other compilation errors occur in the program.



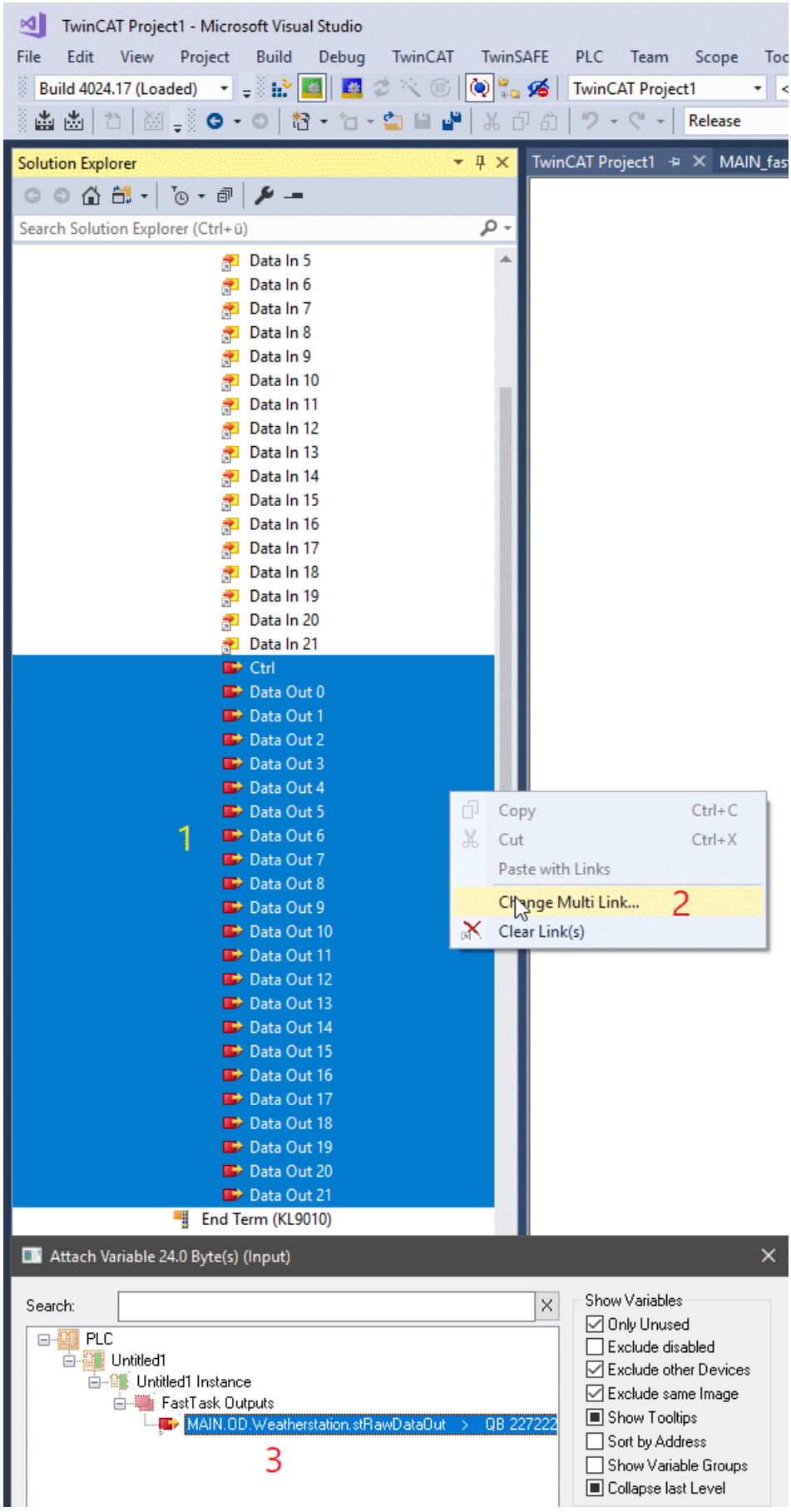
Under "Build", select "Rebuild Solution". At this point, only the already existing PLC task has a process image.



After an error-free creation, the link variables of the template are available and the Fast Task also holds a process image area.

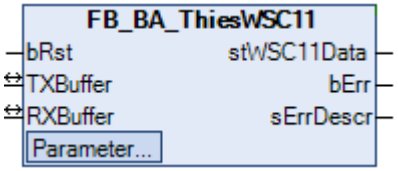


First, all input variables of the terminal process image are marked (click on status and then press the arrow down key while holding down the shift key) (1), then right-click and select "Multi-link" (2). Link to "*stRawDataIn*" of the template.



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.

Syntax

```
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.
bWndSpdLEDOn	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 [▶ 706]	Output of the data
bErr	BOOL	Error
sErrDescr	T_MaxString	Textual error description

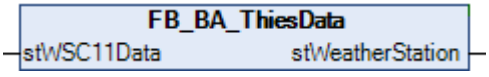
Inputs/outputs

Name	Type	Description
TXBuffer / RXBuffer	ComBuffer	Serial data exchange with the reading function block SerialLineControl 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.

Syntax

```
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 [▶ 706]	Data from the Thies weather station.

Outputs

Name	Type	Description
stWeatherStation	ST_BA_WeatherStation [▶ 705]	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 and FB_BA_AdsComServer.

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 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 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 [▶ 139] and FB_BA_RawSubscriber [▶ 158].

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.eP1ID;
  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	Default value for the parameter Time delay of transitions to abnormal states.
nDefTimeDelay_ToNormal	UDINT	Default value for the parameter Time delay of transitions to normal states.
fDefCOVIncrement	REAL	Default value for the COV increment parameter.
fDefLimitDeadband	REAL	Default value for the Dead band limit parameter.
nLoop_DefOpMode	E_BA_PIDMode	Default value for the Operation mode parameter.
stTrend_DefStartTime	ST_BA_DateTime	Default value for the Start time parameter.
stTrend_DefStopTime	ST_BA_DateTime	Default value for the Stop time parameter.
nTrend_BufferSize	UDINT	Number of entries in a trend buffer.
bTrend_DefLogEnable	BOOL	Enabling trend logging.
bTrend_DefStopOnFull	BOOL	FALSE, Ring buffer; TRUE, Fixed memory that does not store anything when the buffer is full.
nTrend_DefLogInterval	UDINT	Default value for the Logging interval parameter.
nTrend_DefNotificationThreshold	UDINT	Default value for the Notification threshold parameter.
eTrend_DefLoggingType	E_BA_LoggingType	Default value for the Logging type parameter.

VAR_GLOBAL

Name	Type	Description
bBlink	BOOL	Global blink pulse. The blink pulse is generated in the template FB_BA_ControlCabinetBasic [▶ 825].
bDlyStartPLC	BOOL	The variable is set to TRUE delayed after a restart of the PLC. In the template FB_BA_Device [▶ 830] this delay is implemented by the timing element *tonDlyStartPLC*.

6.1.4.3.5 Priority

The global variable list uses *XBA_BACnetParam* from the Tc3_XBA library to determine the BACnet plant priorities.

The chapter Commanding [▶ 34] contains further information on the subject of priorities.

Syntax

```
VAR_GLOBAL
  nProgram             : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[2],16));
  nManualRemote        : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[3],16));
  nManualLocal         : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[4],16));
  nCritical            : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[5],16));
  nLifeSafety          : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[6],16));
END_VAR
```

Name	Type	Description
nProgram	UDINT	Priority "Program", program control.
nManualRemote	UDINT	Priority "Manual Remote", remote manual override (MBE).
nManualLocal	UDINT	Priority "Manual Local", local manual override (LVB).
nCritical	UDINT	Priority "Critical", plant safety.
nLifeSafety	UDINT	Priority "Life Safety", personal safety.

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 management and control level (MCL).

Note that TcHmiBa is **not** a MCL.

Contents

The easy creation of HMIs for building automation is enabled by TcHmiBa by providing various controls [▶ 1122] and icons [▶ 1231].

Generic HMI

For integrators using the complete TF8040 solution, there is also the option of accessing generic functions [▶ 54] 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 Introduction

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 management and control level (MCL). However, it should be noted that *TcHmiBa* is **not** an MBE.

Contents

The easy creation of HMIs for building automation projects is enabled by *TcHmiBa* by providing various controls [▶ 1122] and icons [▶ 1231].

Generic HMI

For integrators using the complete TF8040 solution, there is also the option of accessing generic functions [▶ 54] from *TcHmiBa*, which further simplify and accelerate engineering.

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*
  - BaSite-Extension: *§ProjektPfad§/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*
  - BaSite-Extension: *C:\ProgramData\Beckhoff\TF2000 TwinCAT 3 HMI Server\service\TcHmiProject\BaSite\Legal*

6.2.1.1.1 System requirements

For the development system:

- TwinCAT HMI Engineering

For the target system

- TwinCAT HMI Server

The NuGet packages included are divided into two areas.

Client packages

- Beckhoff.TwinCAT.HMI.BA.Icons
- Beckhoff.TwinCAT.HMI.BA.Framework
- Beckhoff.TwinCAT.HMI.BA.Controls

The following system requirements apply to the Client packages:

- Chrome engine (e.g. Microsoft Edge®, Google Chrome®, Chromium®)

When using web browsers with a different engine, proper function and display is not guaranteed.

Server packages

- Beckhoff.TwinCAT.HMI.BA.BaSite

The following system requirements apply to the Server packages:

- at least Windows 10 on the development and target system
- .NET Desktop Runtime > v6.0

6.2.1.2 Controls

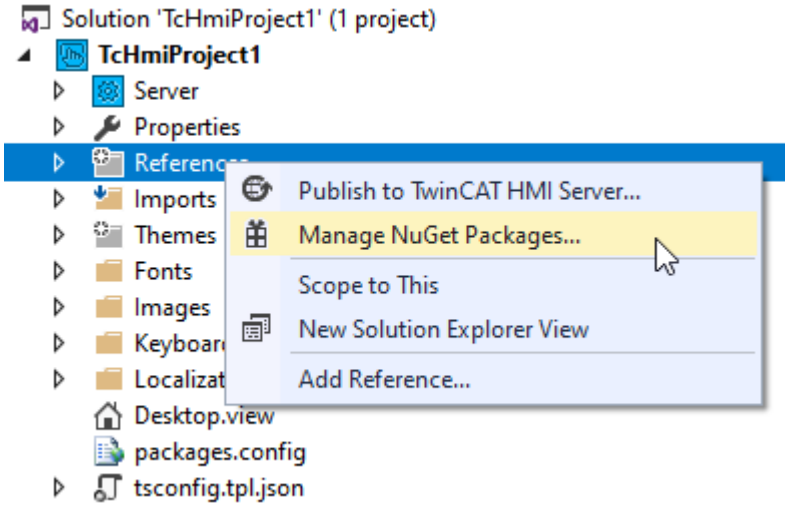
The following pages describe the content and functions of the NuGet package **Beckhoff.TwinCAT.HMI.BA.Controls**. The main focus is on the attributes and functions of the controls that can be used in the Designer.

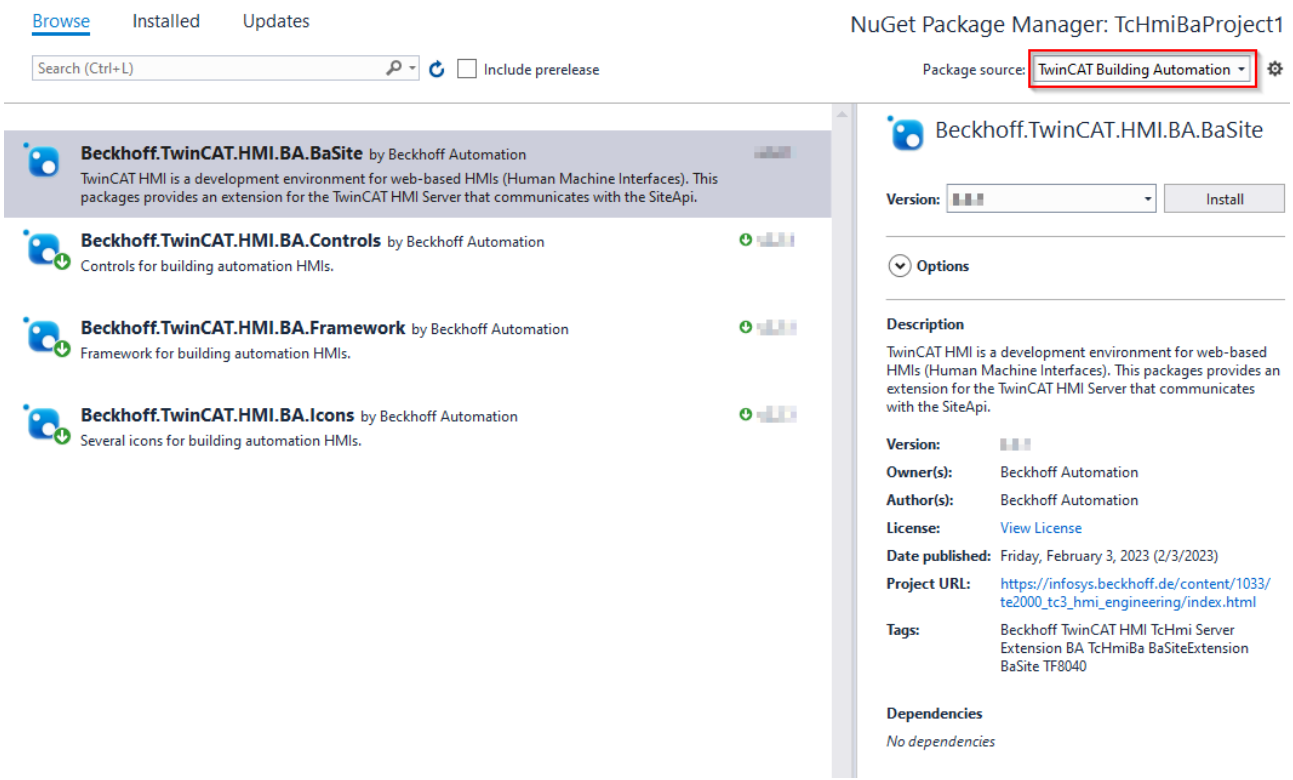
A TcHmiBa control is implemented as a TcHmi-FrameworkControl and provides various functionalities of TF8040 in graphical form.

Although the creation involves more effort, the result is a more performant and convenient control for the user.

Installation

The NuGet package **Beckhoff.TwinCAT.HMI.BA.Controls** must be installed in order to use the controls and functions.





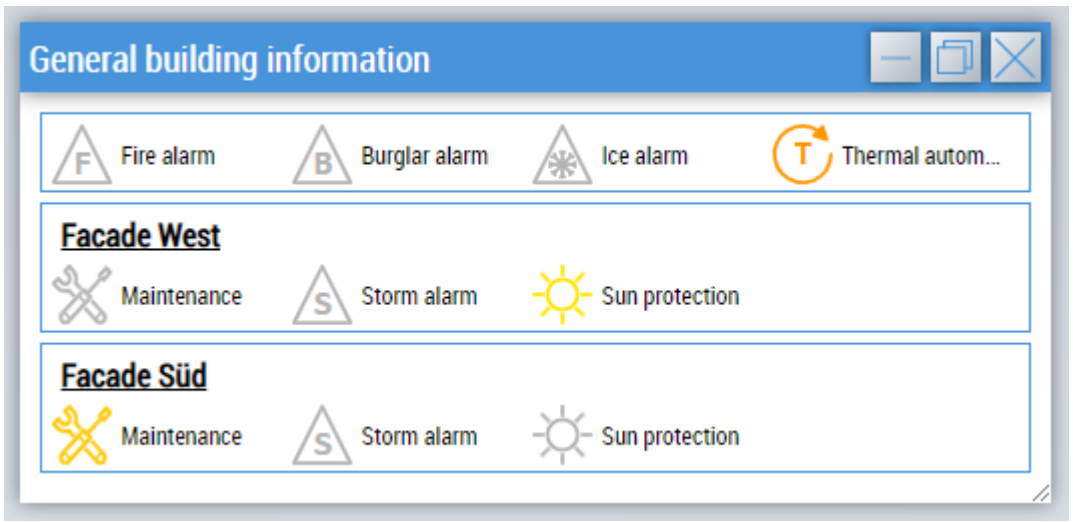
The package depends on the following packages and therefore installs them as well:

- Beckhoff.TwinCAT.HMI.Framework
- Beckhoff.TwinCAT.HMI.Controls
- Beckhoff.TwinCAT.HMI.BA.Framework [▶ 1242]
- Beckhoff.TwinCAT.HMI.BA.Icons [▶ 1231]

6.2.1.2.1 BuildingGeneral

6.2.1.2.1.1 BuildingInformation

The **BuildingInformation** control initially appears like a normal button [▶ 1130]. 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 [▶ 1210]
- Window [▶ 1214]

Attributes

The control inherits from the button [▶ 1130] 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.

BurglarAlarm

```
tchmi:general#/definitions/Boolean
```

If TRUE, the burglar alarm is displayed.

IceAlarm

```
tchmi:general#/definitions/Boolean
```

If TRUE, the ice alarm is displayed.

ThermalAutomatic

```
tchmi:general#/definitions/Boolean
```

If TRUE, the thermal automatic is displayed.

Facades

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingInformation.Facades
```

Create different facades that can then be referenced by corresponding controls (see above).

6.2.1.2.1.2 Legend

The **Legend** control displays all icons on the active web page with their description.

Use

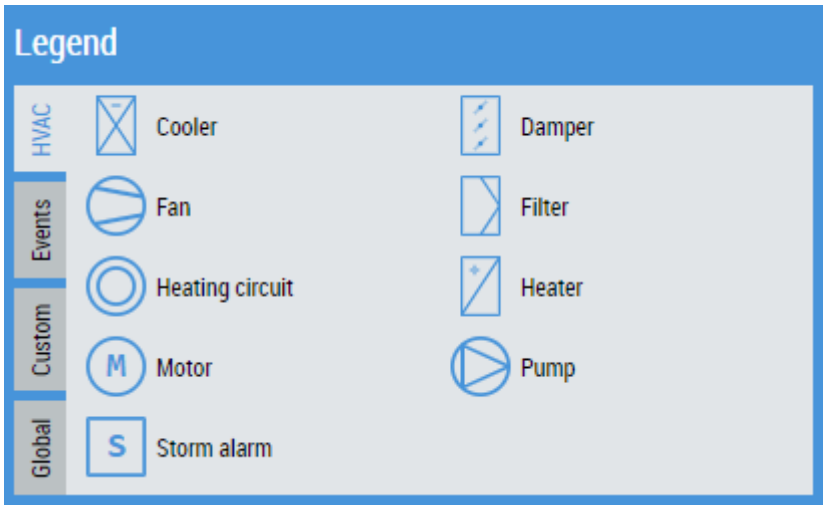
Use on any page where you want to display an explanation of the icons.

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 [▶ 1256] function.



Additional icons

Local

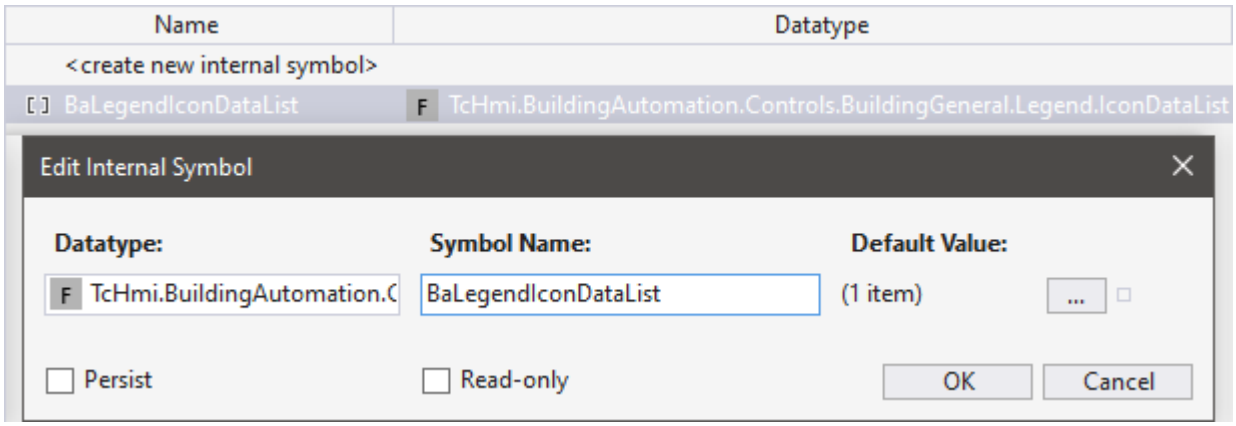
Additional icons can be added to the respective instance of the legend via the IconDataCustom [▶ 1126] attribute. When the function is called, the attribute is in the parameter list.

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 [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

Common

ShowHeadline

```
tchmi:general#/definitions/Boolean
```

Sets the visibility of the title.

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.

IconDataSource

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataSource
```

Selection of entries to be displayed.

IconDataCustom

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList
```

List with additional entries.

6.2.1.2.1.3 WeatherStation

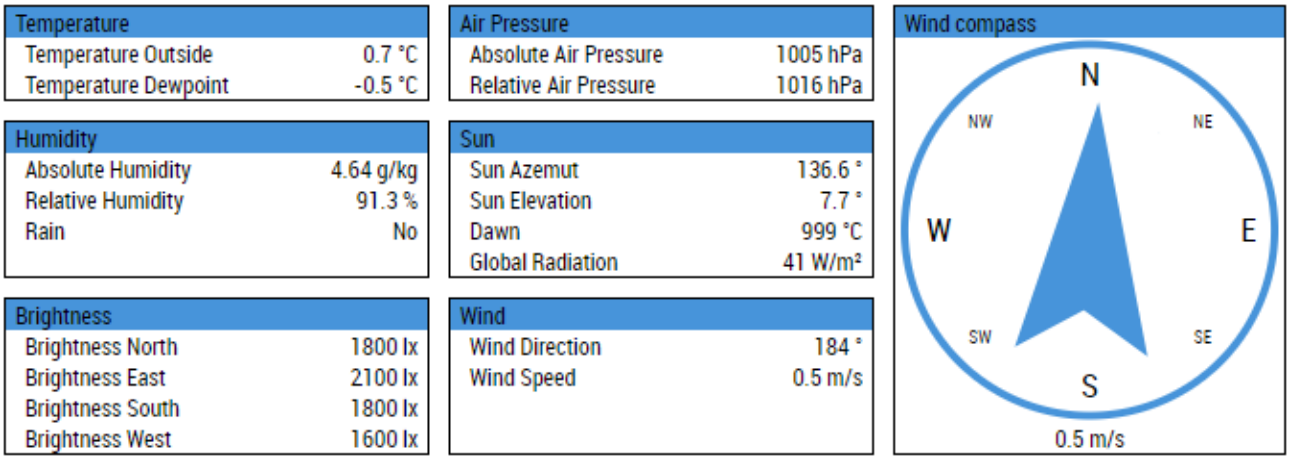
The **WeatherStation** template displays the weather data.

Use

Use on any page where a template of the **WeatherStation** type is to be displayed.

Compatibility

The BaTemplateDescription supports the following BaObjects.



Subelements:

Symbol name	PLC template	Description
Dstb	FB_BA_SensorBinary [▶ 1088]	Fault
Rain	FB_BA_SensorBinary [▶ 1088]	Rain
WthT	FB_BA_SensorAnalog [▶ 1087]	Outside temperature
DewPtT	FB_BA_SensorAnalog [▶ 1087]	Dew point
PrssAbs	FB_BA_SensorAnalog [▶ 1087]	Air pressure (absolute)
PrssRel	FB_BA_SensorAnalog [▶ 1087]	Air pressure (relative)
HumAbs	FB_BA_SensorAnalog [▶ 1087]	Air humidity (absolute)
HumRel	FB_BA_SensorAnalog [▶ 1087]	Air humidity (relative)
Brightness	FB_BA_MeasuredValueCardinal [▶ 1085]	Brightness
Dawn	FB_BA_SensorBinary [▶ 1088]	Dawn
GlobRadn	FB_BA_SensorAnalog [▶ 1087]	Solar radiation
WndDir	FB_BA_SensorAnalog [▶ 1087]	Wind direction
WndSpd	FB_BA_SensorAnalog [▶ 1087]	Wind speed
Latd	FB_BA_SensorAnalog [▶ 1087]	Latitude
Lngt	FB_BA_SensorAnalog [▶ 1087]	Longitude
SunAzm	FB_BA_SensorAnalog [▶ 1087]	Sun direction
SunElv	FB_BA_SensorAnalog [▶ 1087]	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 [▶ 853]
- FB_BA_WeatherStation_Thies [▶ 855]

Attributes

The control inherits from BaseTemplate and thus has the same attributes. In addition, there are the following attributes.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.BuildingGeneral.WeatherStation.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Common

ShowPosition

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the position are visible.

ShowTemperature

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the temperature are visible.

ShowAirPressure

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the air pressure are visible.

ShowHumidity

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the air humidity are visible.

ShowSun

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the sun are visible.

ShowBrightness

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the brightness are visible.

ShowWind

```
tchmi:general#/definitions/Boolean
```

Determines whether the values for the wind are visible.

ShowWindCompass

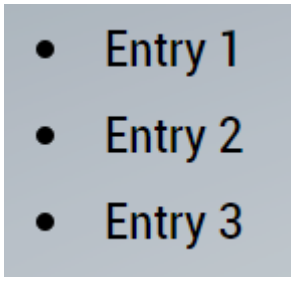
```
tchmi:general#/definitions/Boolean
```

Determines whether the wind compass is visible.

6.2.1.2.2 Common

6.2.1.2.2.1 BulletPointList

The **BulletPointList** lists various texts in bullet point form.

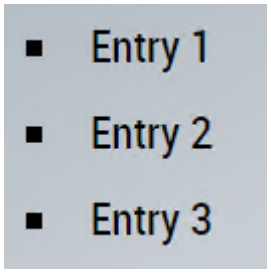


Use

Use on any page where a bullet point list is to be displayed.

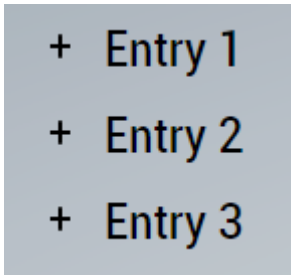
Features

Change of the bullet by the attribute *ListStyleType*.



Numbering of the entries through the corresponding selection of the *ListStyleType* attribute.

Change of the bullet by the attribute *ListStyleImage*.



Attributes

The control inherits from TextControl [▶ 1197] 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.

ListStyleType

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.BulletPointList.ListStyleTypes
```

Type of the bullet. If *String* is selected, the text entered is used as a bullet without interpretation.

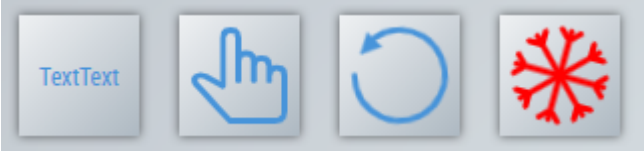
ListStyleImage

```
tchmi:framework#/definitions/Path
```

Path to the image of the bullet.

6.2.1.2.2.2 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 TcHmiBa.Icons [▶ 1231].

Attributes

The control inherits from TextControl [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Icon

```
tchmi:framework#/definitions/Path
```

Path to the icon.

IconWidth

```
tchmi:general#/definitions/Number
```

Width of the icon.

IconWidthUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Unit of the width of the icon.

IconHeight

```
tchmi:general#/definitions/Number
```

Height of the icon.

IconHeightUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Unit of the height of the icon.

IconHorizontalAlignment

```
tchmi:general#/definitions/HorizontalAlignment
```

Definition of the horizontal alignment of the icon within the button.

IconVerticalAlignment

```
tchmi:general#/definitions/VerticalAlignment
```

Definition of the vertical alignment of the icon within the button.

IconRotation

```
tchmi:general#/definitions/Number
```

Determines by how many degrees the icon should be rotated.

IconRotationSpeed

```
tchmi:general#/definitions/Number
```

Determines the speed at which the icon should rotate.

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 clockwise.

IconOverlays

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Icon.OverlayList
```

Defines overlays of the icon.

Legend

ShowInLegend

```
tchmi:general#/definitions/Boolean
```

Specifies whether the icon is visible in the legend.

Category

```
tchmi:general#/definitions/String
```

Specifies the category of the icon in the legend.

Description

```
tchmi:general#/definitions/String
```

Specifies the description text of the icon in the legend.

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 [▶ 1269].

6.2.1.2.2.3 Calendar

The control **Calendar** is used to display and manage exceptions to a schedule and select a date.

Use

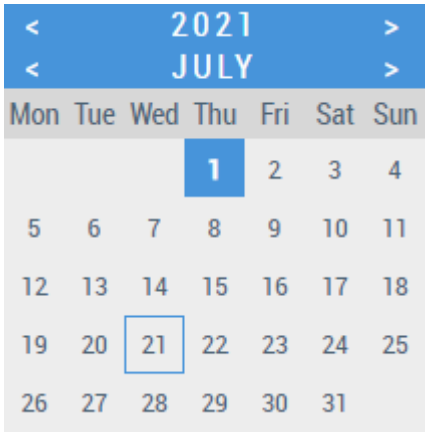
Use on any page where a date is to be selected.
If a Schedule object is passed to the BaObject [▶ 1135] attribute in the *EventCalendar* display mode, the exceptions of a time schedule can also be displayed or edited.

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.

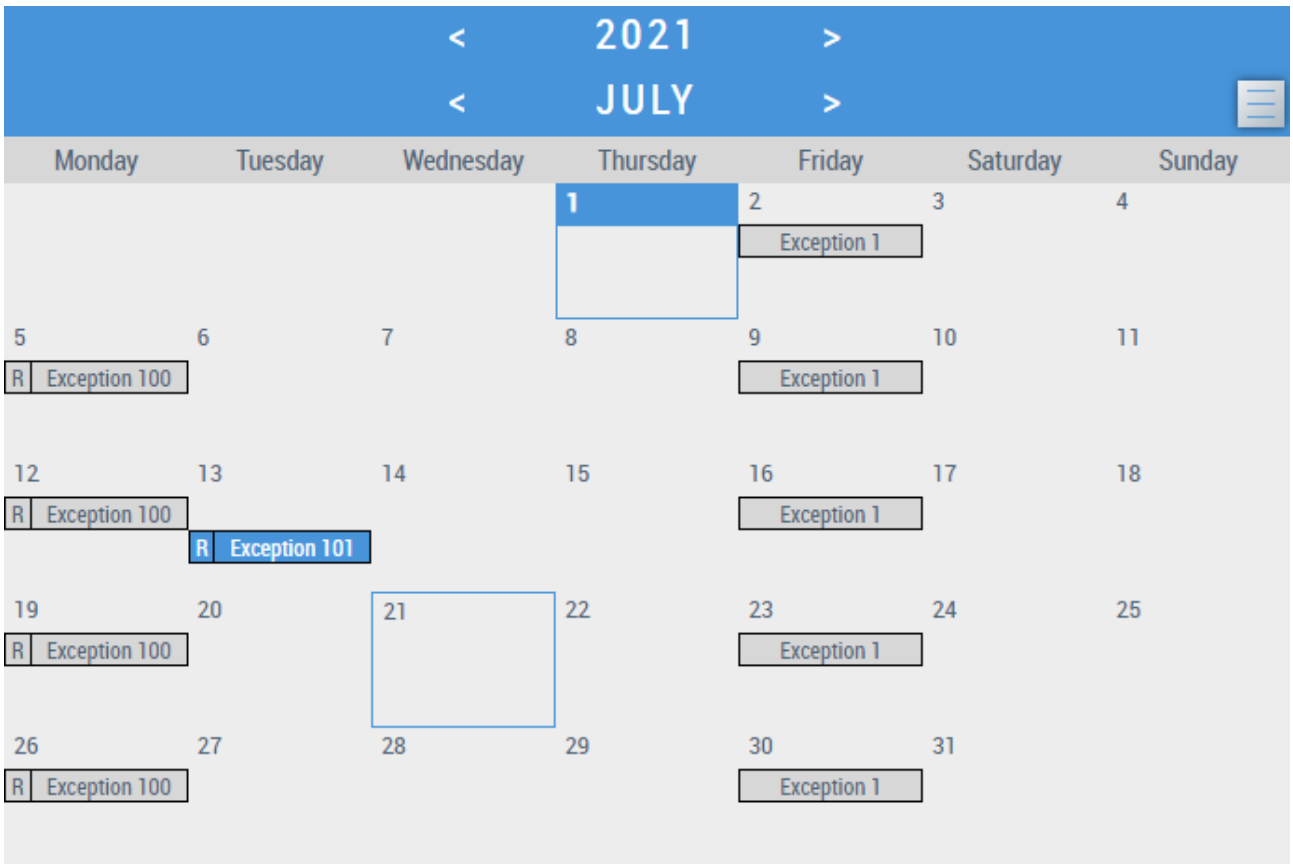


EventCalendar

The EventCalendar is available in the month view and in the year view.

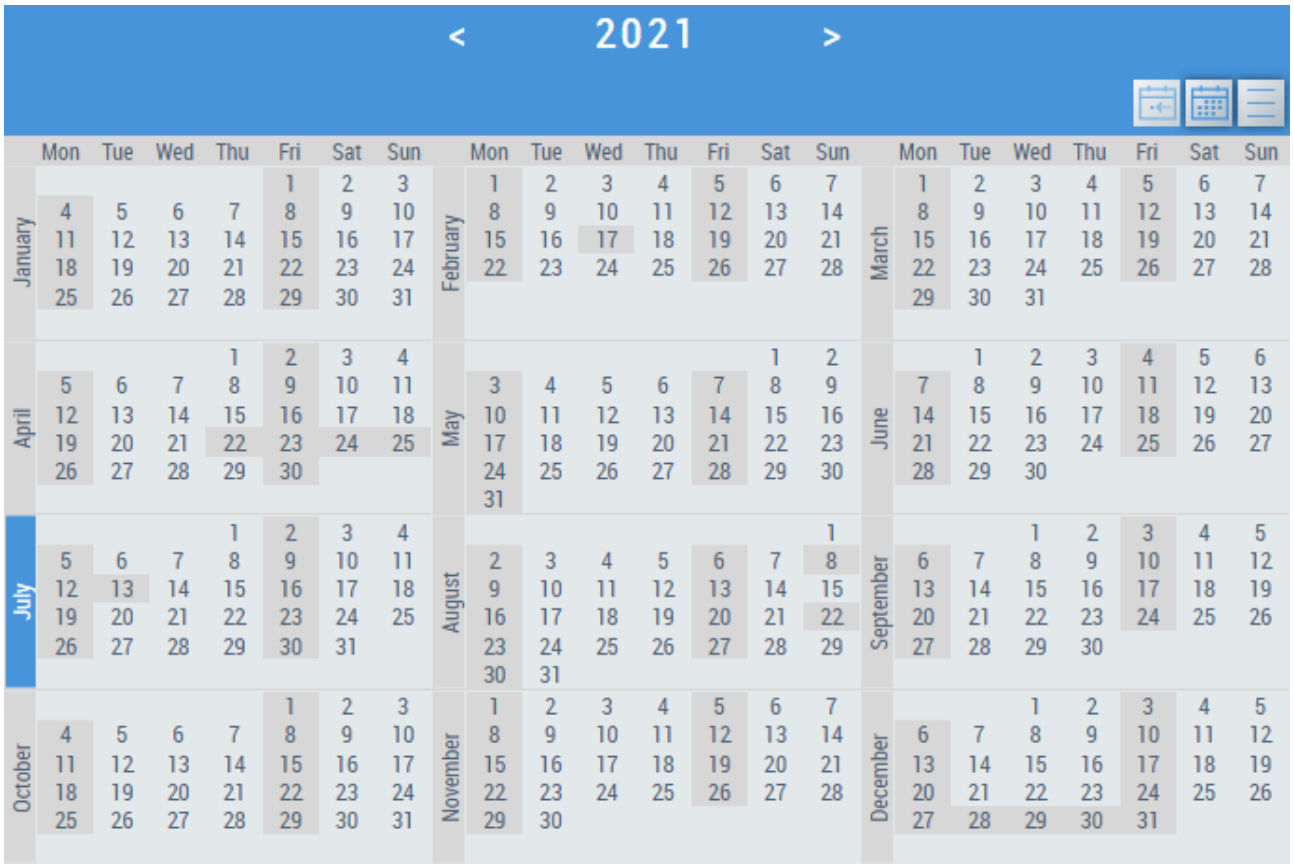
Month view

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.



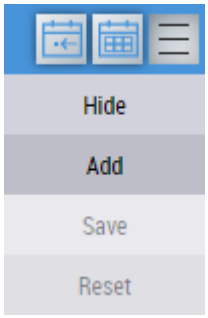
Year view

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.



Menu

In the upper right corner there are buttons for quick access and other actions.



Quick access:

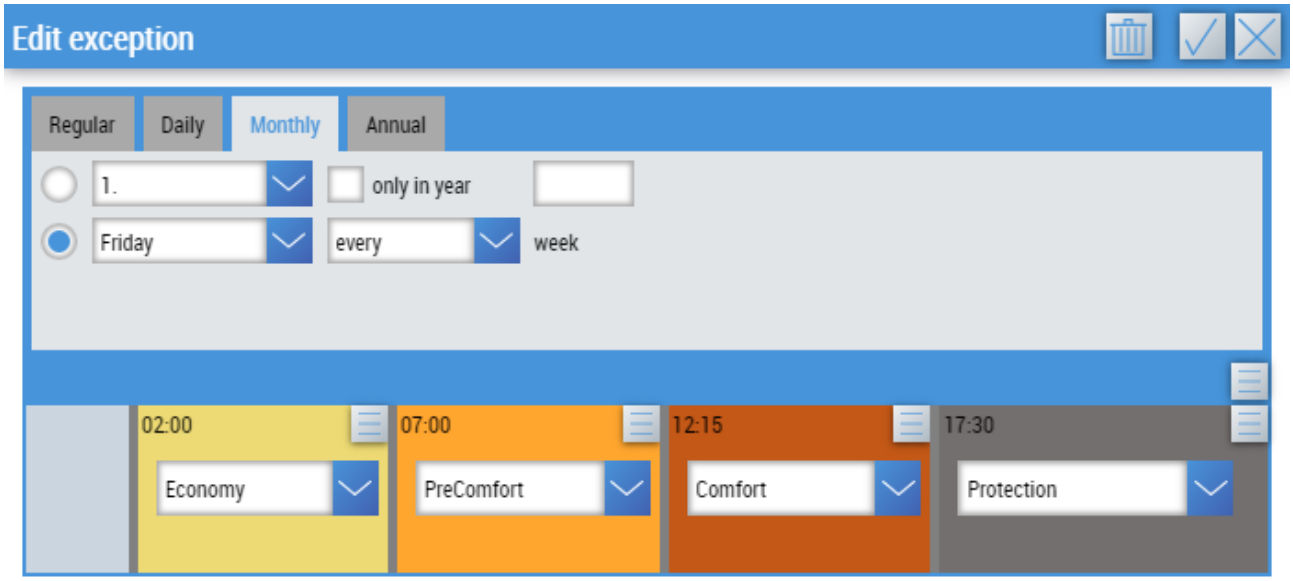
- Today: Jumps to the current date.
- View: Switches between the month and year view.

Other actions:

- Show/Hide: Shows or hides exceptions.
- Add: Adds local exceptions.
- Save: Writes all changes to the PLC.
- Reset: Discards all unconfirmed changes.

Local exceptions

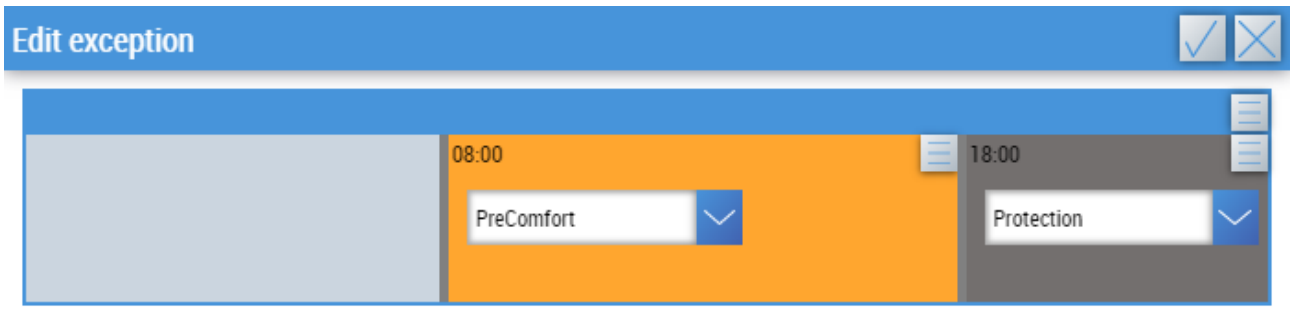
Entries in the *aException* collection of a Schedule object (e.g. FB_BA_SchedM [▶ 225]) are regarded as local exceptions.



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 [▶ 225]) are regarded as global exceptions.



For global exceptions, only the time periods and applicable values can be defined. The date or the repetition type must be configured in the referenced Calendar object (e.g. FB_BA_Cal [▶ 217]).
The automatic numbering of the global exceptions starts at 100.

Attributes

The control inherits from BaseControl [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

DisplayMode

Determines the display mode of the calendar.

DisplayView

Determines the view of the calendar in the *EventCalendar* display mode.

ShowMenu

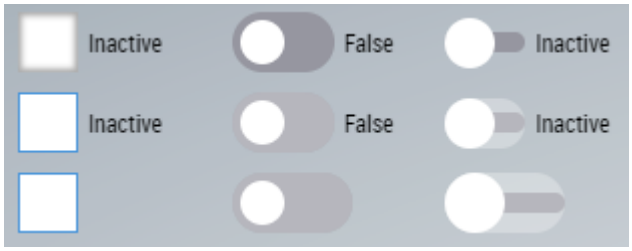
Sets the visibility of the menu.

Events

Event	Description
onDateChanged	Returns the selected date.

6.2.1.2.2.4 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 [▶ 1136] attribute (see image above).

Possibility to link a BaObject [▶ 1136] 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 [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

ActiveText

```
tchmi:general#/definitions/String
```

Specifies the text that is displayed if *State* is TRUE.

InactiveText

```
tchmi:general#/definitions/String
```

Specifies the text that is displayed if *State* is FALSE.

Appearance

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Checkbox.Appearance
```

Determines how the checkbox appears at the top.

Colors

CheckBackgroundColor

```
tchmi:general#/definitions/SolidColor
```

Background color of the checkbox if *State* is TRUE.

This attribute has no effect if *Appearance* is set to *ToggleSlider*.

CheckmarkColor

```
tchmi:general#/definitions/SolidColor
```

Color of the check mark or toggle.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol that fulfills the BaInterface [▶ 62] of the control.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.Checkbox.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

State

```
tchmi:general#/definitions/Boolean
```

State of the checkbox.

StateFeedback

```
tchmi:general#/definitions/Boolean
```

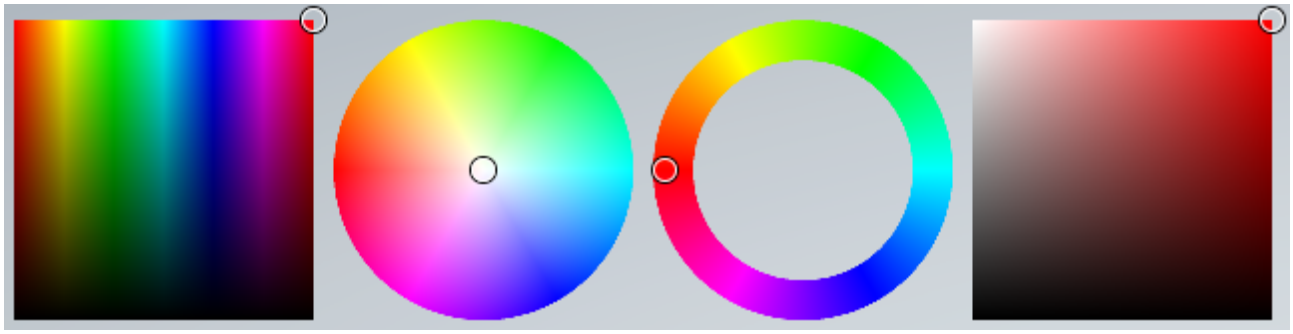
Feedback for the state of the checkbox.

Events

Event	Description
onStateChanged	Triggered when the value of State has changed.
onUserInteractionFinished	Triggered when the user interaction with the checkbox has finished.

6.2.1.2.2.5 ColorPicker

The **ColorPicker** can be used to select a color from various color palettes.



Use

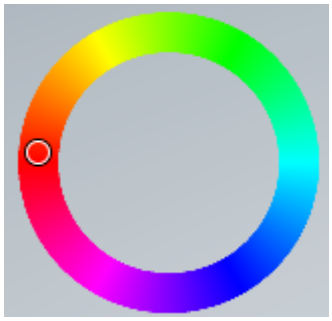
Can be used on any page where it is necessary to select a color.

Features

Color selection can be done using different color spaces.

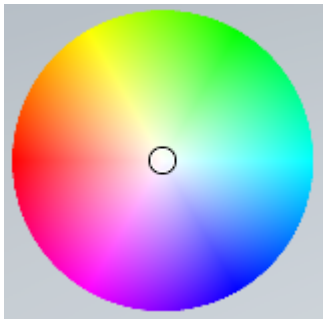
HSL Ring

All colors are selected without shading.



HSV Circle

Selection of all colors up to white.

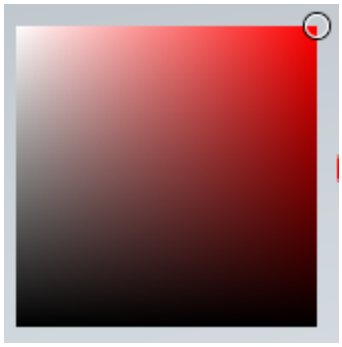


HSL Rect All color

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 TcHmiControl and thus has the same attributes. In addition, there are the following attributes.

ColorPlateType

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.ColorPlateType
```

Defines the color palette used.

BackgroundColor

```
tchmi:framework#/definitions/SolidColor
```

Determines the background color when the HslRect1 color palette is selected.

SelectedSolidColor (read-only)

```
tchmi:framework#/definitions/SolidColor
```

Selected color.

SelectedRgbaColor (read-only)

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.RGBAColor
```

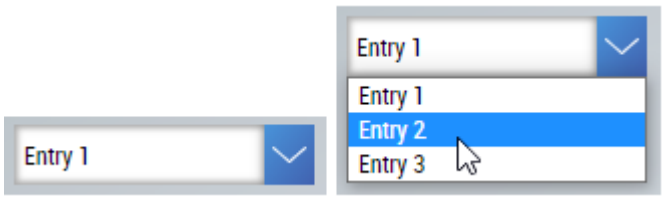
Selected color in RGBA format.

Events

Event	Description
onSelectedColorChanged	Triggered when the selected color has changed.

6.2.1.2.2.6 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 [▶ 1140] 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 [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

Data

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Combobox.ComboboxItems
```

Data for the combo box.

Colors

ButtonColor

```
tchmi:framework#/definitions/SolidColor
```

Color of the button that opens the dropdown list.

ButtonArrowColor

```
tchmi:framework#/definitions/SolidColor
```

Color of the arrow in the button.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol that fulfills the BaInterface [▶ 62] of the control.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.Combobox.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

SelectedData (read-only)

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Combobox.ComboboxItem
```

Currently selected data.

SelectedValue

```
tchmi:general#/definitions/Number
```

Value of the currently selected data.

UseSelectedValueFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether or not the attribute SelectedValueFeedback is used.

SelectedValueFeedback

```
tchmi:general#/definitions/Number
```

Feedback for the selected value.

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.

6.2.1.2.2.7 DateTimeField

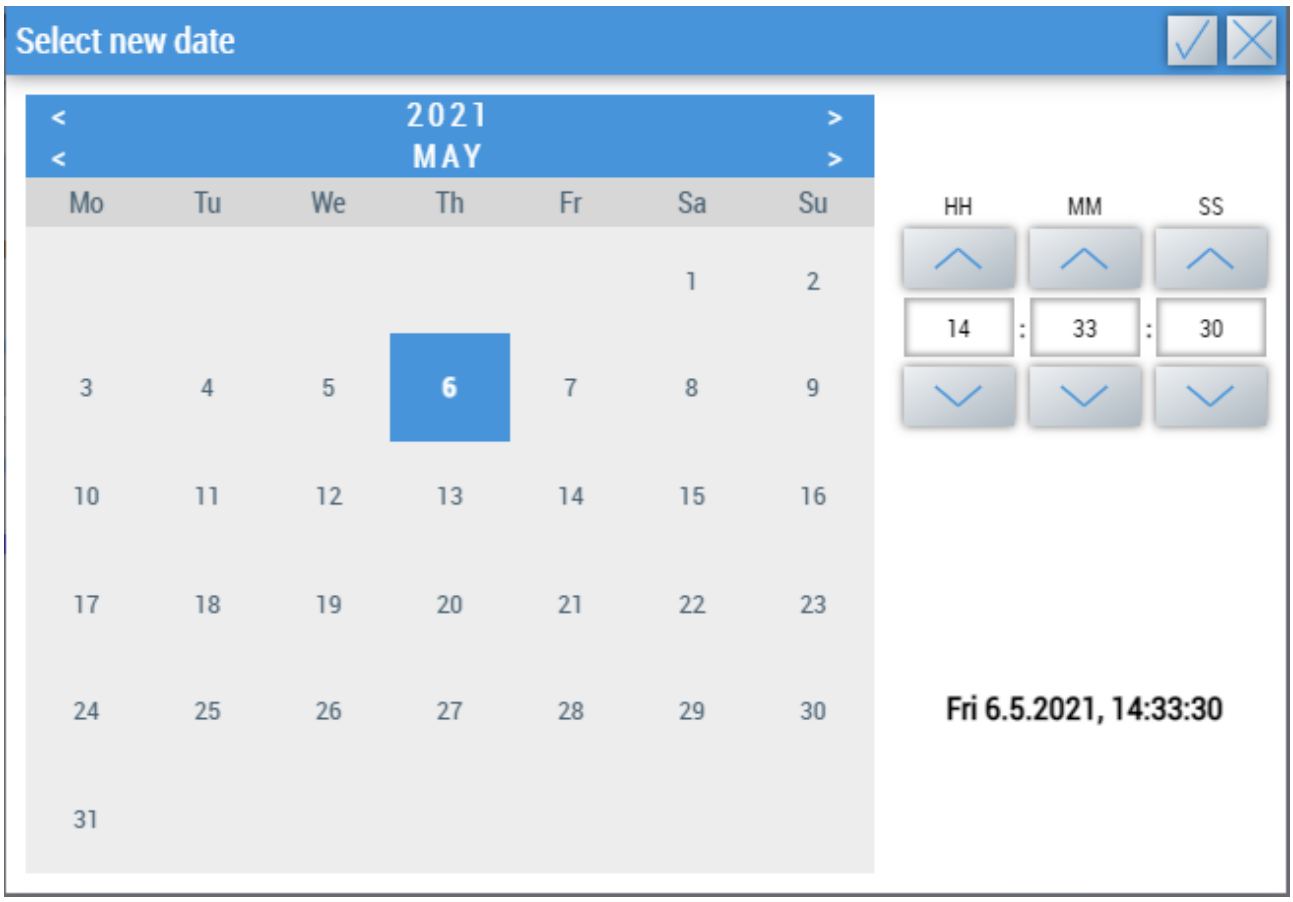
The **DateTimeField** can be used to select or display a date and time.

Use

Use on any page where date values are to be displayed or edited.

Features

If the attribute ReadOnly [▶ 1194] is FALSE, the **DateTimePicker** can be opened via the button to select a new date or time.



Attributes

The control inherits from TextControl [▶ 1197] 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.

Events

Event	Description
onDateChanged	Triggered when the value of the date or time has changed.
onUserInteractionFinished	Triggered when the user interaction with the DateTimeField has finished.

6.2.1.2.2.8 Icon

The icon essentially corresponds to the TcHmiImage. The only difference is extended functionalities for the icon, because the options of the icon package [▶ 1233] can be used here.

Use

Use wherever an icon is required.

Features

Provides advanced functionality for icons from the NuGet package TcHmiBaIcons [▶ 1231].

Attributes

Icon

```
tchmi:framework#/definitions/Path
```

Path to the icon.

IconWidth

```
tchmi:general#/definitions/Number
```

Width of the icon.

IconWidthUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Unit of the width of the icon.

IconHeight

```
tchmi:general#/definitions/Number
```

Height of the icon.

IconHeightUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Unit of the height of the icon.

IconHorizontalAlignment

```
tchmi:general#/definitions/HorizontalAlignment
```

Definition of the horizontal alignment of the icon within the button.

IconVerticalAlignment

```
tchmi:general#/definitions/VerticalAlignment
```

Definition of the vertical alignment of the icon within the button.

IconRotation

```
tchmi:general#/definitions/Number
```

Determines by how many degrees the icon should be rotated.

IconRotationSpeed

```
tchmi:general#/definitions/Number
```

Determines the speed at which the icon should rotate.

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 clockwise.

IconOverlays

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Icon.OverlayList
```

Defines overlays of the icon.

Icon

ShowInLegend

```
tchmi:general#/definitions/Boolean
```

Specifies whether the icon is visible in the legend.

Category

```
tchmi:general#/definitions/String
```

Specifies the category of the icon in the legend.

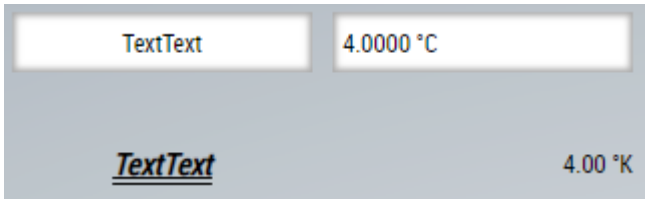
Description

```
tchmi:general#/definitions/String
```

Specifies the description text of the icon in the legend.

6.2.1.2.2.9 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 [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

DataType

```
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. If *number* is selected *Value* does **not** contain the unit.

Text (read-only)

```
tchmi:general#/definitions/String
```

The displayed text.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol that fulfills the BaInterface [▶ 62] of the control.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.InputBox.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

Value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Current Value. Depending on the selected **DataType** the value is numerical or textual.

ValueFeedback

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Feedback for the value.

UseValueFeedback

```
tchmi:general#/definitions/Boolean
```

If TRUE, then the *ValueFeedback* attribute is used.

Number

MinValue

```
tchmi:general#/definitions/Number
```

Lowest permissible input value (if *DataType* is equal to *number*).

MaxValue

```
tchmi:general#/definitions/Number
```

Largest permissible input value (if *DataType* is equal to *number*).

Unit

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Determines the unit after *Value* (if *DataType* is equal to *number*). Possible values:

- textual (e.g. "°C")
- numerical (enumeration value of E_BA_Unit)

Digits

```
tchmi:general#/definitions/Number
```

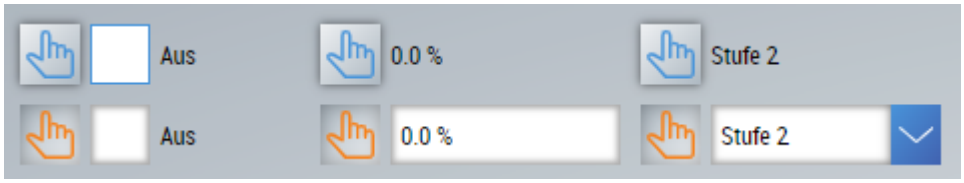
Number of decimal places (if *DataType* is equal to *number*).

Events

Event	Description
onValueChanged	Triggered when the value has changed.
onUserInteractionFinished	Triggered when the user exits the input. This means: • Enter key is pressed • InputBox loses focus

6.2.1.2.2.10 ManualOverride

The *ManualOverride* is used to manually override a binary, analog or multi-level value.



Use

Use on any page where a value is to be manually overridden.

The control is used in the ProjectNavigationTextual [▶ 1167].

Features

Simply override the current value with a manual value. If manual override is active, this is indicated visually.

Attributes

The control inherits from BaseControl [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol that fulfills the BaInterface [▶ 62] of the control.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.ManualOverride.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

ManualEnable

```
tchmi:general#/definitions/Boolean
```

Defines whether manual override is enabled or not.

ManualValue

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

The current manual value.

AutoValue

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

The current automatic value.

6.2.1.2.2.11 ManualOverrideBinary

ManualOverrideBinary is used to manually override a binary value.

Use

Use on any page where a binary value is to be overridden manually.

Features

Simply override the current value with a manual value. If manual override is active, this is indicated visually.

Attributes

The control inherits from BaseControl [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol that fulfills the BaInterface [▶ 62] of the control.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.ManualOverrideBinary.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

ManualEnable

```
tchmi:general#/definitions/Boolean
```

Defines whether manual override is enabled or not.

ManualValue

```
tchmi:general#/definitions/Boolean
```

The current manual value.

AutoValue

```
tchmi:general#/definitions/Boolean
```

The current automatic value.

6.2.1.2.2.12 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

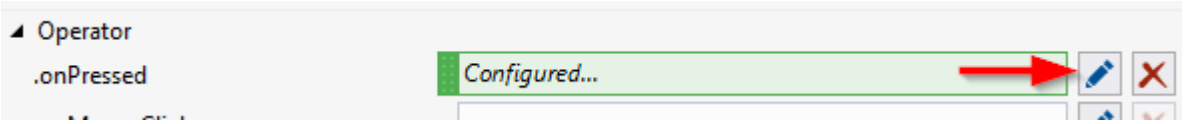
Devices with touch screen

Switch between configured content pages with a swipe gesture to the left or right.

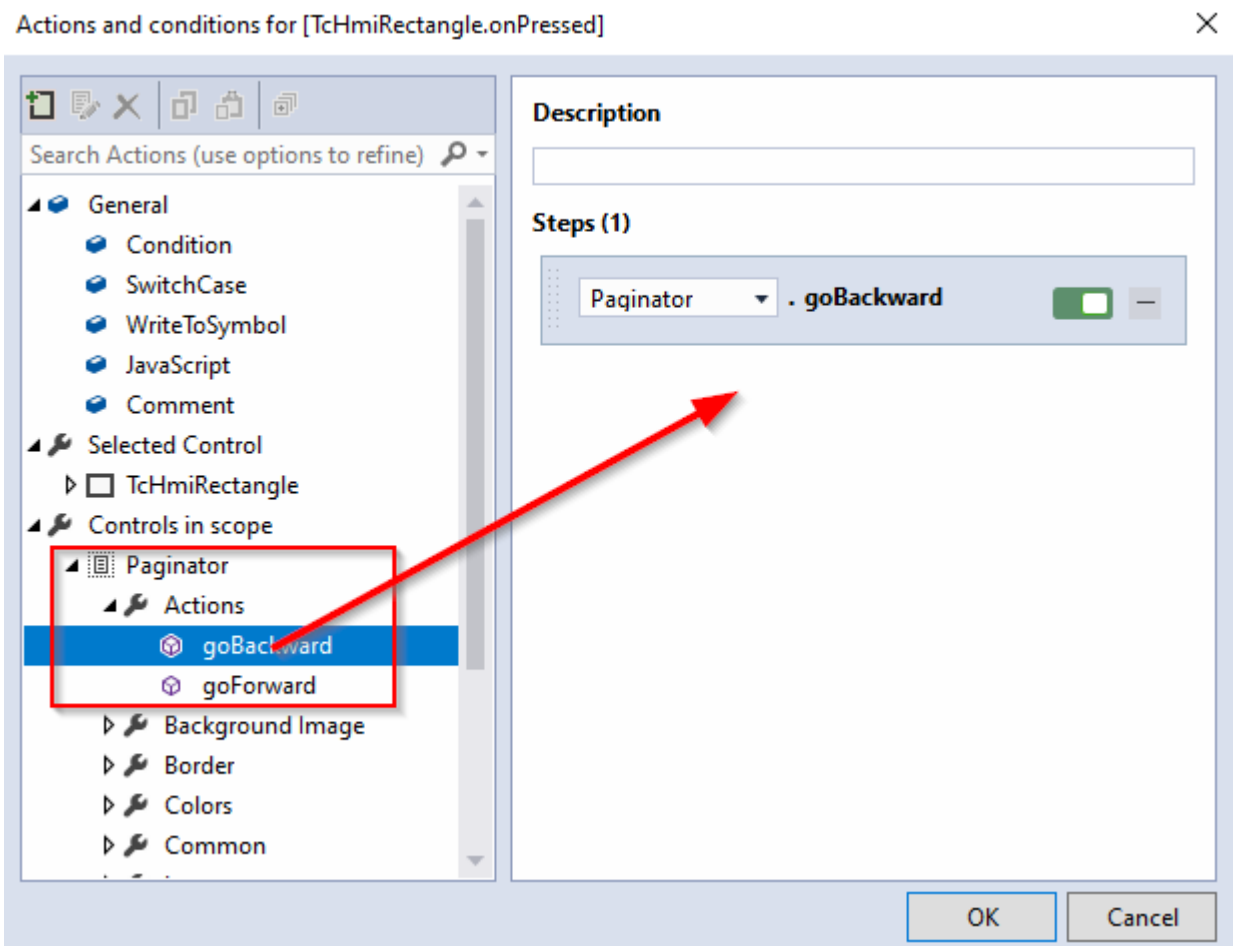
Devices without touch screen

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.

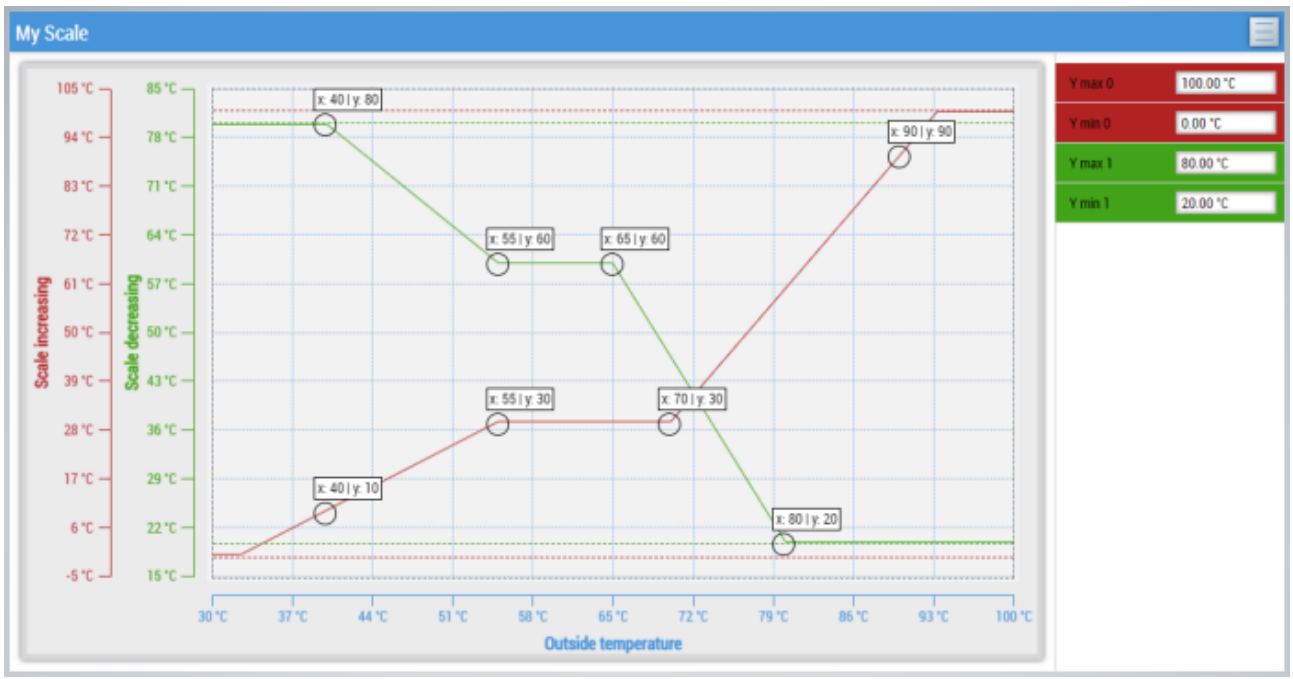
GoBackward

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.

6.2.1.2.2.13 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 [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

Data

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Scale.Scales
```

Determines the data for the different scales.

XAxisExtension

```
tchmi:general#/definitions/Number
```

Determines how much longer the X-axis is displayed, depending on XMin and XMax.

Title

```
tchmi:general#/definitions/String
```

Determines the title that will be displayed in the header of the control.

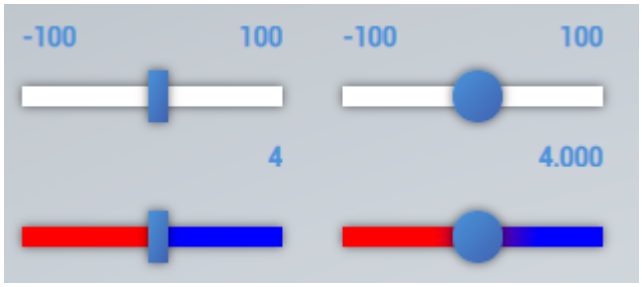
ShowHeader

```
tchmi:general#/definitions/Boolean
```

Determines whether the header of the control is displayed or not.

6.2.1.2.2.14 Slider

The **slider** can be used to display and edit numerical values.

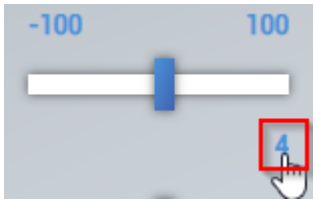


Use

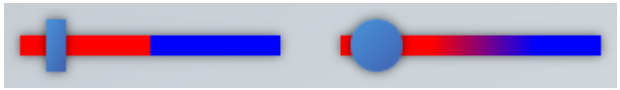
Can be dragged to any page where numerical values are to be edited.

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 [▶ 1152] 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 [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

ShowValue

```
tchmi:general#/definitions/Boolean
```

Determines whether the current value is displayed.

ShowScale

```
tchmi:general#/definitions/Boolean
```

Determines whether the MinValue [▶ 1156] and MaxValue [▶ 1156] are displayed.

Orientation

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Orientation
```

Determines the orientation of the slider (horizontal or vertical).

SwitchMinMax

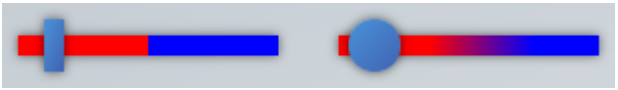
```
tchmi:general#/definitions/Boolean
```

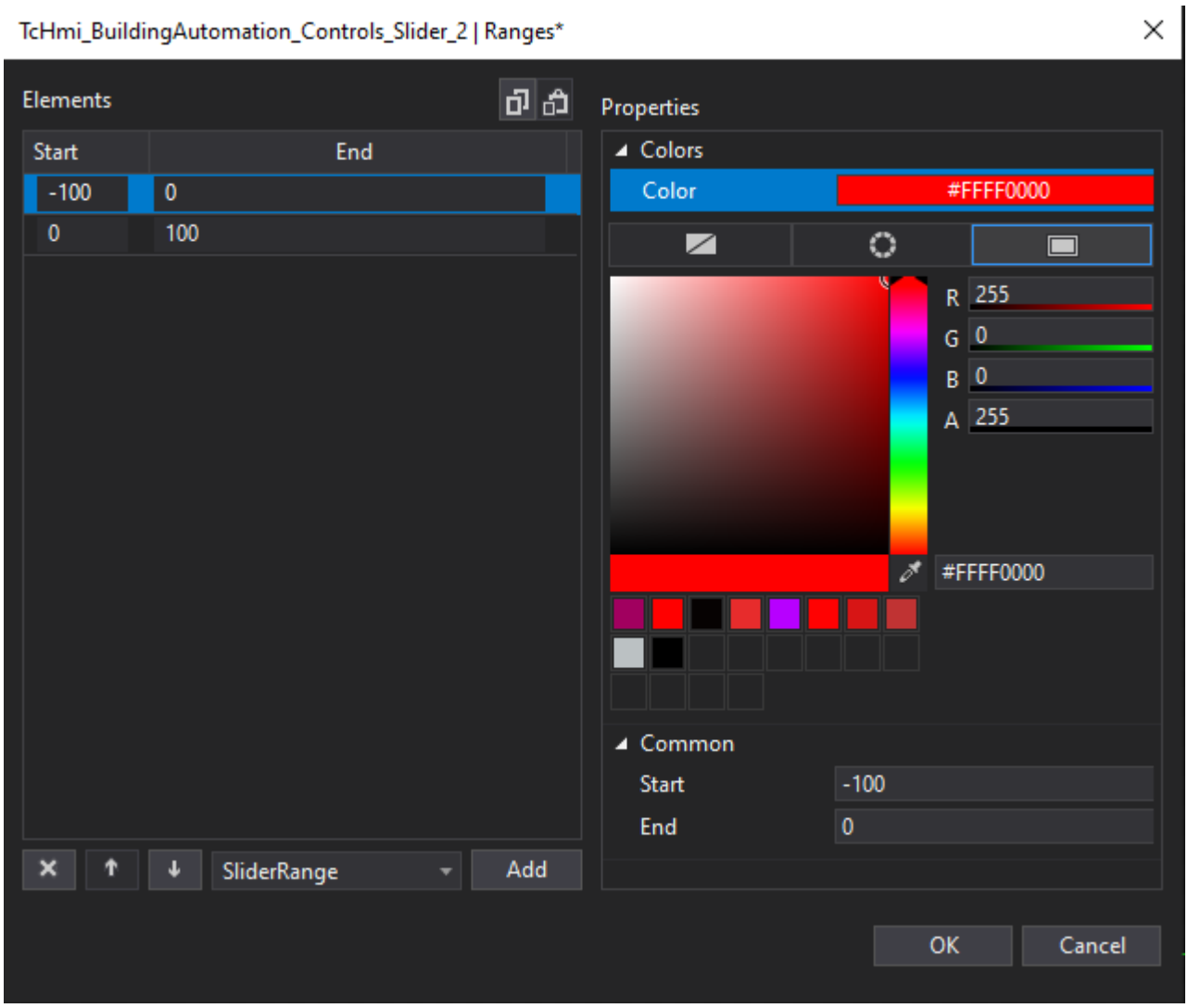
If active, the positions of *MinValue* and *MaxValue* are swapped.

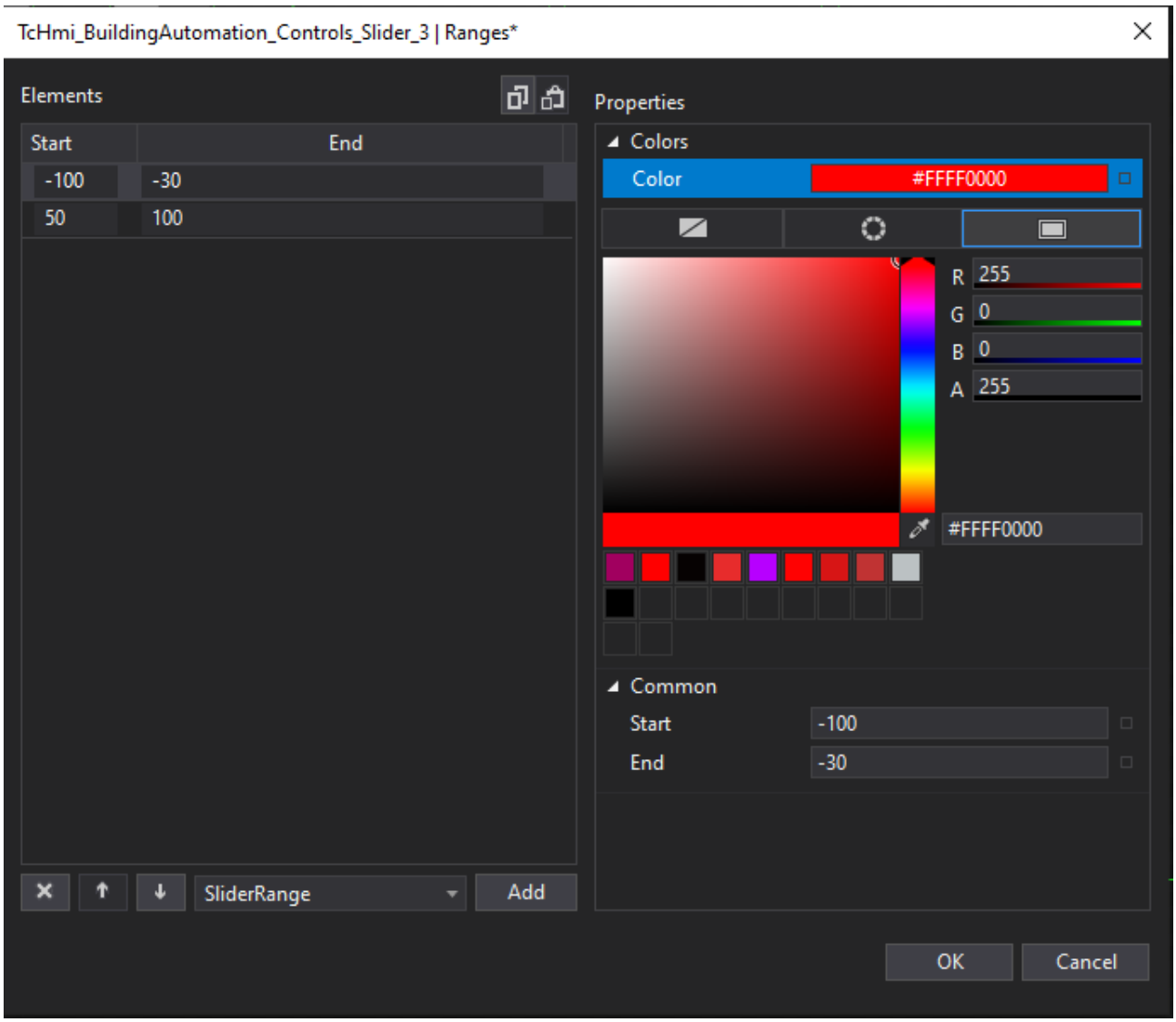
Ranges

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Slider.SliderRanges
```

Specifies different color areas or color gradients to be displayed in the slider.




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 display of the slider knob.

BaData

BaInterface

`tchmi:framework#/definitions/Symbol`

Symbol that fulfills the BaInterface [▶ 62] of the control.

BaInterfaceSymbolNames

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Slider.BaInterfaceSymbolNames`

Edit the BaInterfaceSymbolNames [▶ 1246].

Value

`tchmi:general#/definitions/Number`

The current value of the slider.

ValueFeedback

`tchmi:general#/definitions/Number`

The feedback for the value of the slider.

Number

Unit

```
tchmi:general#/definitions/String
```

Determines the unit that is displayed after Value [▶ 1155].

MinValue

```
tchmi:general#/definitions/Number
```

The minimum value of the slider.

MaxValue

```
tchmi:general#/definitions/Number
```

The maximum value of the slider.

Step

```
tchmi:general#/definitions/Number
```

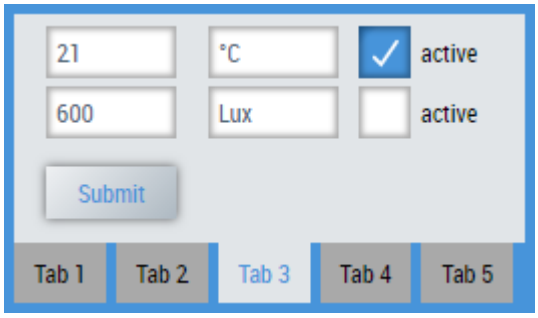
Determines the accuracy with which the value can be set with the slider (e.g. 0.01).

Events

Event	Description
OnValueChanged	The event is triggered every time the value of the slider changes, for example, when the slider is moved.
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.

6.2.1.2.2.15 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 [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

Common

Data

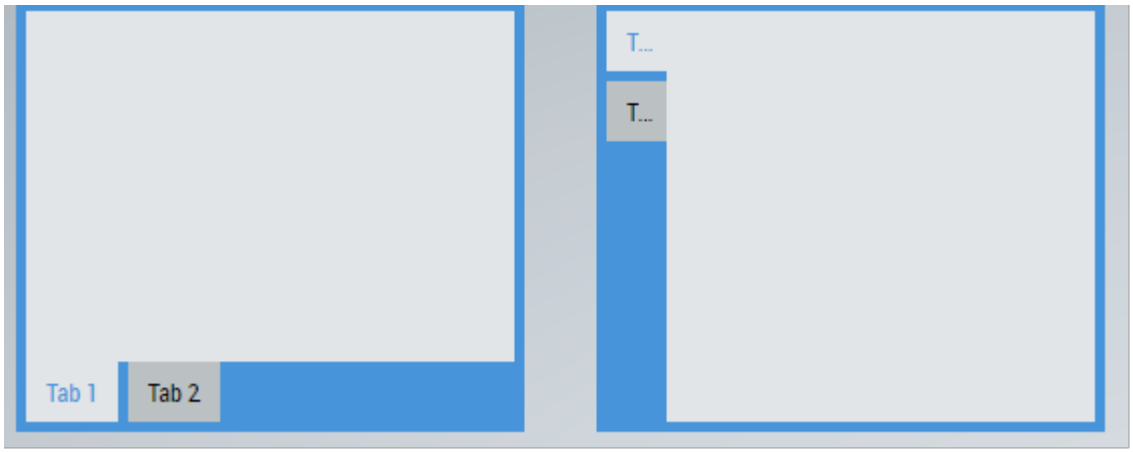
```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TabWindow.TabWindowData
```

Data for the different tabs.

TabPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position
```

Position of the tabs.



TabDistance

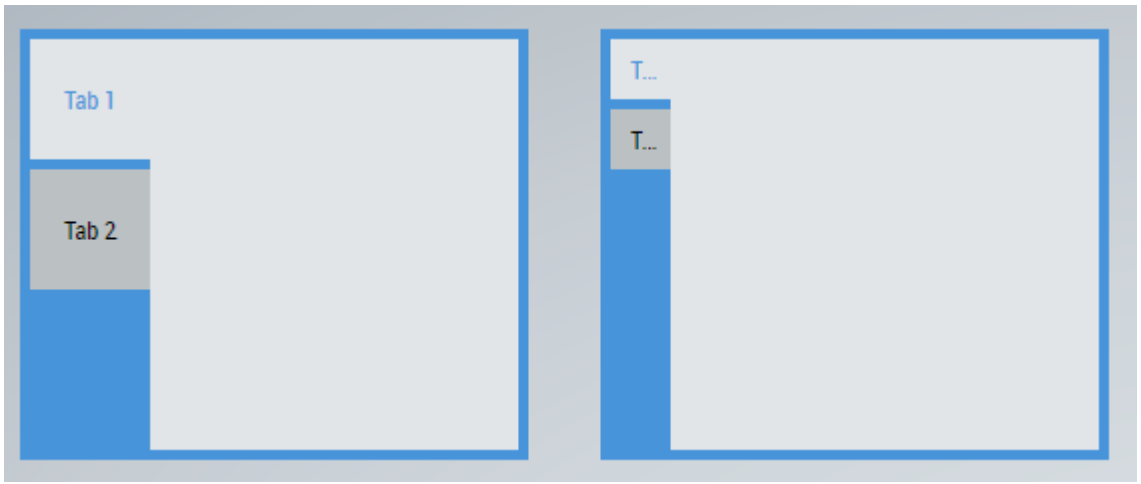
```
Tchmi:framework#/definitions/Number
```

Distance between the tabs.

TabContainerDistance

```
tchmi:general#/definitions/Number
```

The size of the tab container.



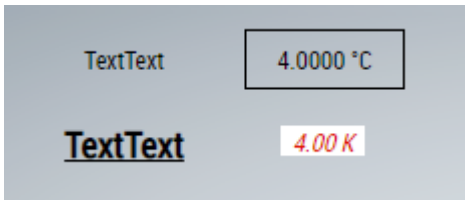
TabSizeAuto

```
tchmi:general#/definitions/Boolean
```

Determines whether the tabs should occupy the complete width of the tab container or not.

6.2.1.2.2.16 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 [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Digits

```
tchmi:general#/definitions/Number
```

Number of decimal places if an analog BaObject [▶ 60] is linked.

Common

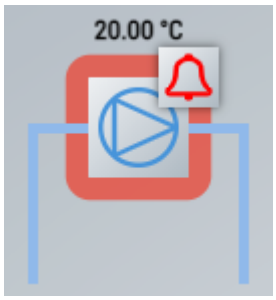
Text

```
tchmi:general#/definitions/String
```

Text for the text block.

6.2.1.2.2.17 UiIcon

The **UiIcon** control can be used to display events and values. It looks like a normal button [▶ 1130] and can be filled with different icons.



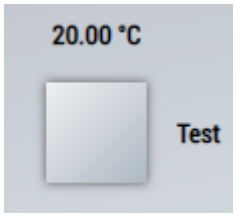
Use

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 UiIcon with a main line, for example.

Features

Value displays

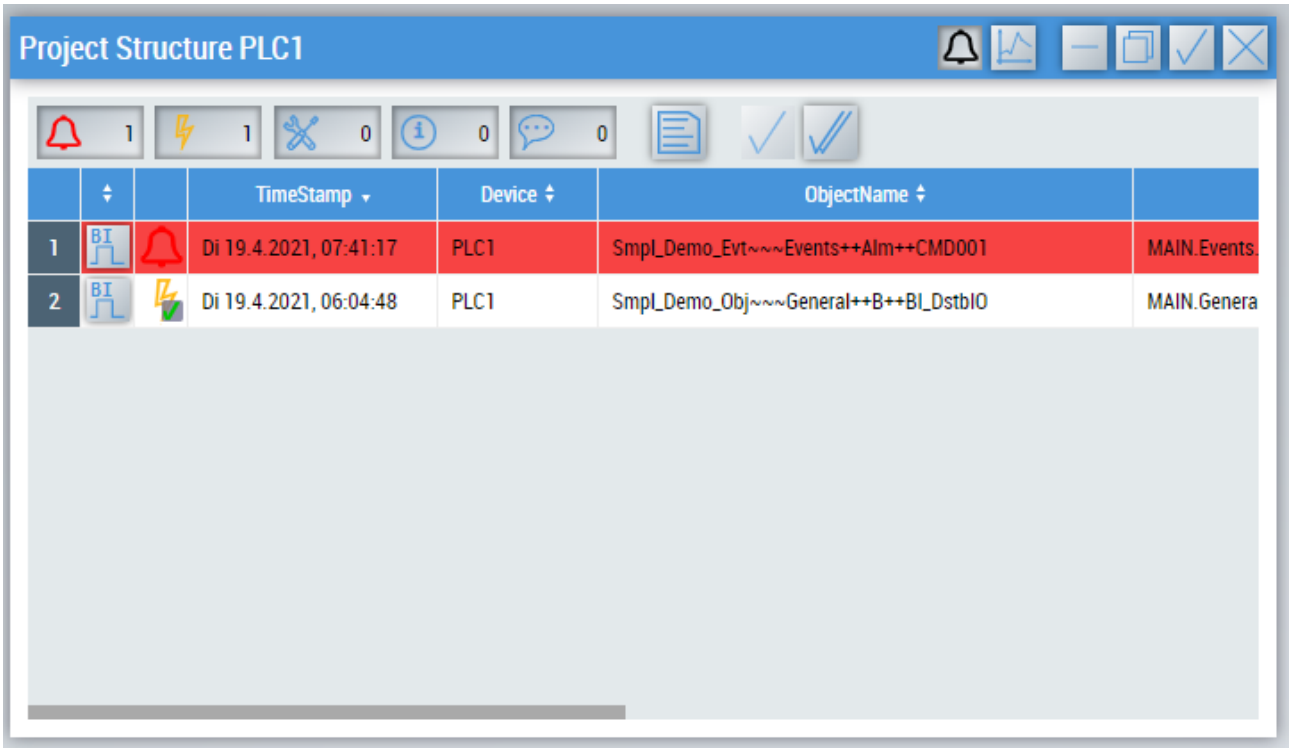
Various displays can be added to the **UiIcon** via the attribute *DisplaysData*.



Event displays

The attribute *EventsData* can be used to display various events around the **UiIcon**.

If the generic approach [▶ 77] of TcHmiBa is used and a BaObject / BaView is linked to the control, active events are displayed automatically. When the **UiIcon** is actuated, the project navigation [▶ 1167] of the linked object opens and, in the case of an event, the parameter window [▶ 1170] with the event view opens accordingly.



Attributes

The control inherits from the button [▶ 1130] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

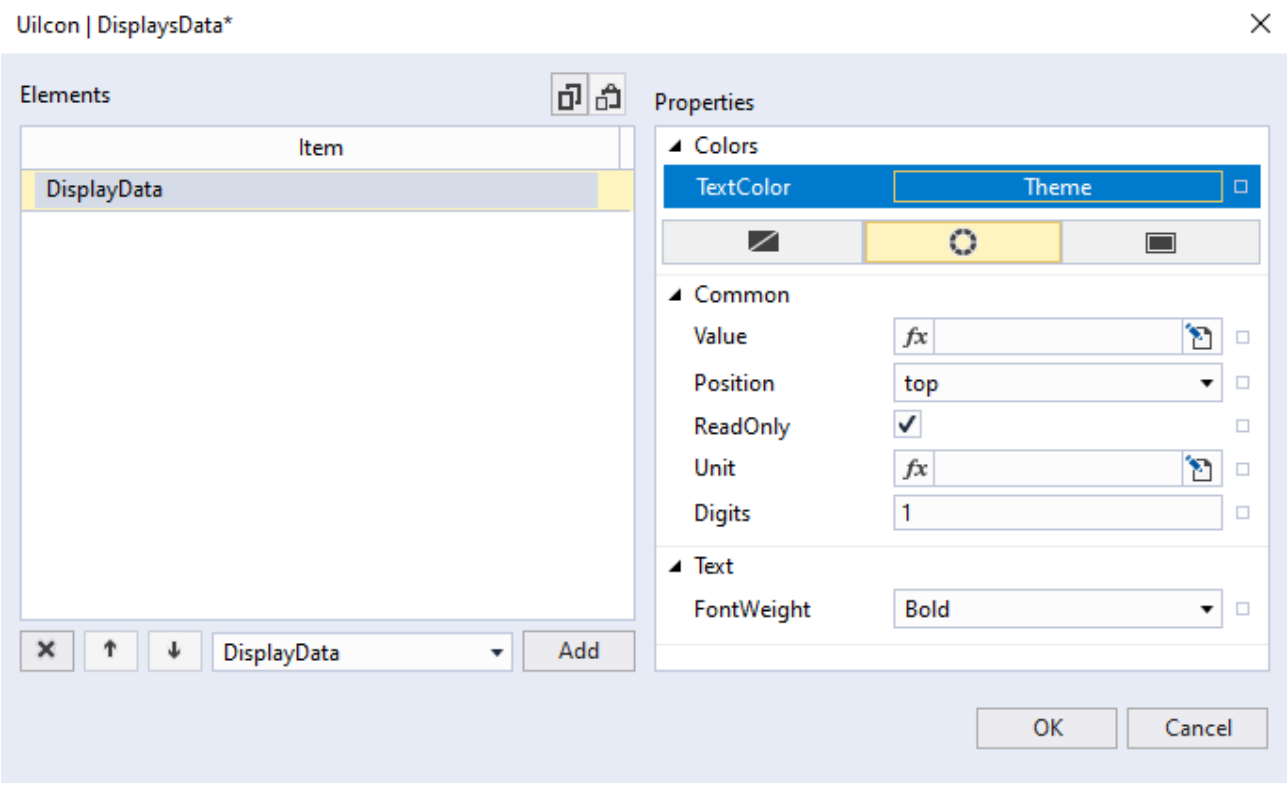
The attribute is not applicable to all controls.

Common

DisplaysData

```
TcHmi.BuildingAutomation.Controls.UiIcon.DisplaysData
```

The attribute makes it possible to create different displays via an editor.



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.

DisplayedDigits

```
tchmi:general#/definitions/Boolean
```

The number of decimal places when a number is shown in a display.

IconStatus

```
TcHmi.BuildingAutomation.Controls.Common.UiIcon.Status
```

Colors the icon according to the set status.

Automatic coloring is set via the global variable AutoActivateIconStatus [▶ 1271].

Status	Display
Alarm	
Fault	
Maintenance	
Notification	
Others	
Active	
Inactive	

Change the colors used via the following CSS variables in the theme CSS file:

```
:root {
    --tchmi-ba-global-event-color-alarm: rgb(255, 0, 0);
      --tchmi-ba-global-event-color-disturb: rgb(255, 255, 0);
      --tchmi-ba-global-event-color-maintenance: rgb(255, 255, 0);
      --tchmi-ba-global-event-color-notification: rgb(255, 255, 55);
      --tchmi-ba-global-event-color-other: rgb(255, 255, 255);
      --tchmi-ba-global-color-active: rgb(0, 255, 0);
      --tchmi-ba-global-color-inactive: rgb(255, 255, 255);
}
```

It should be noted that changes to these colors apply per visualization, which can make it difficult to maintain different systems due to different appearances.

BorderStatus

```
TcHmi.BuildingAutomation.Controls.Common.UiIcon.Status
```

Colors the icon according to the set status.

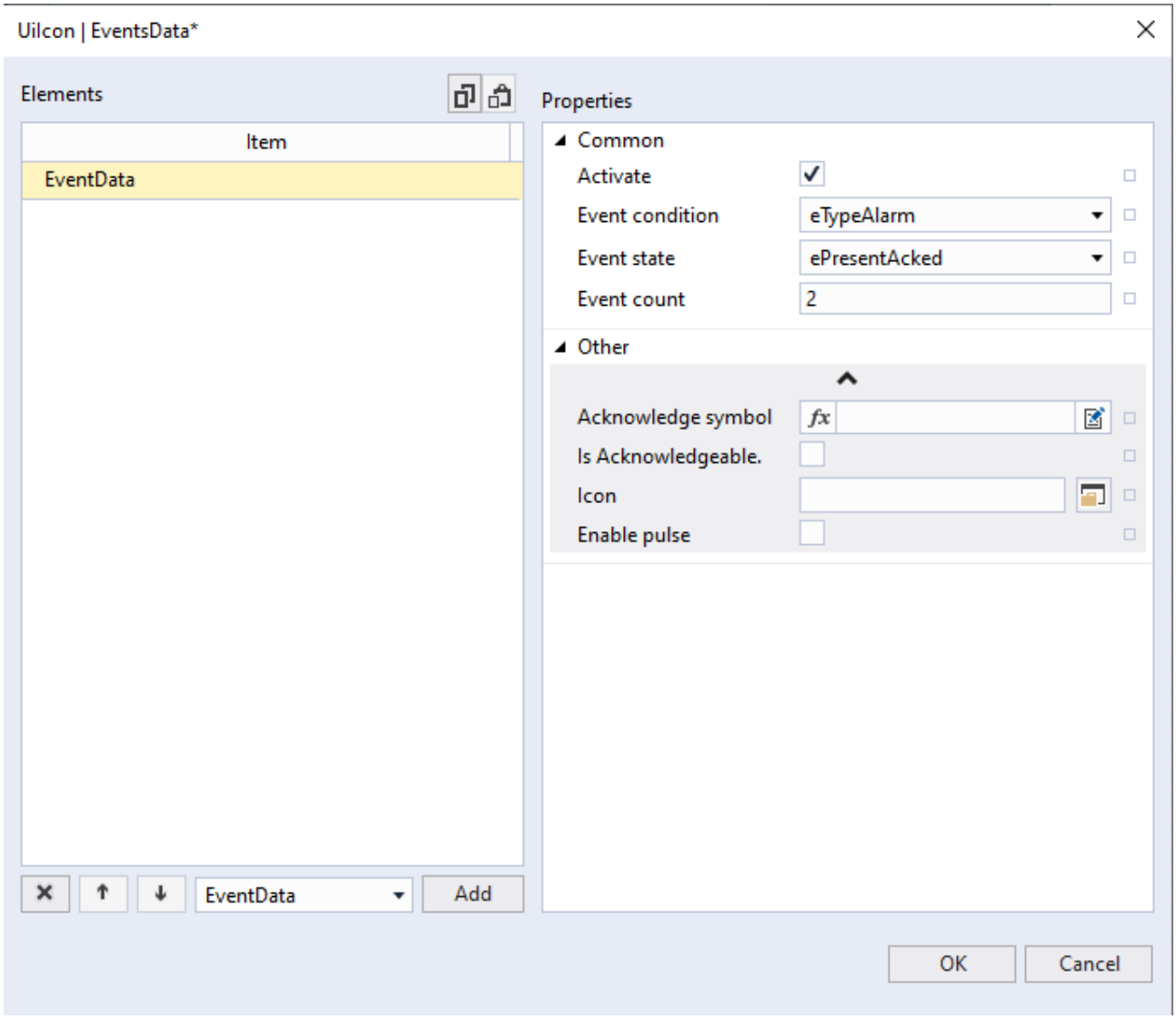
Status	Display
Alarm	
Fault	
Maintenance	
Notification	
Others	
Active	
Inactive	

For the colors used, see the attribute IconStatus.

EventsData

`TcHmi.BuildingAutomation.Controls.UiIcon.EventsData`

The attribute makes it possible to create different events via an editor.



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 *event condition* is selected to allow user specific icons.
Enable pulse	Evaluation is done only if *Event condition* and *Event state* are not used. When activated, a red pulse is displayed around the **UiIcon**.

ShowDisplays

```
tchmi:general#/definitions/Boolean
```

Determines whether the displays defined in the attribute *DisplayData* are shown or not.

PopUp

ShowFaceplate

```
tchmi:general#/definitions/Boolean
```

Determines whether a pop-up is opened when the UiIcon is clicked.

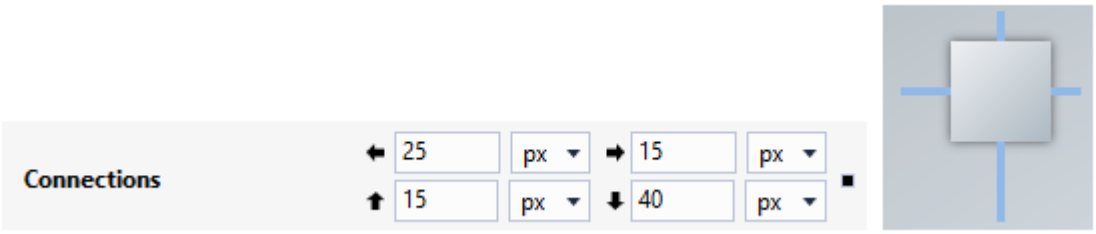
If the attribute is not activated, the UiIcon can, for example, be used for navigation to another content but still display events of the linked BaObject.

Connections

Connections can be used to represent connections to other lines in a P&I diagram.

```
tchmi:framework#/definitions/Padding
```

Connections can be created here that extend vertically or horizontally away from the UiIcon.



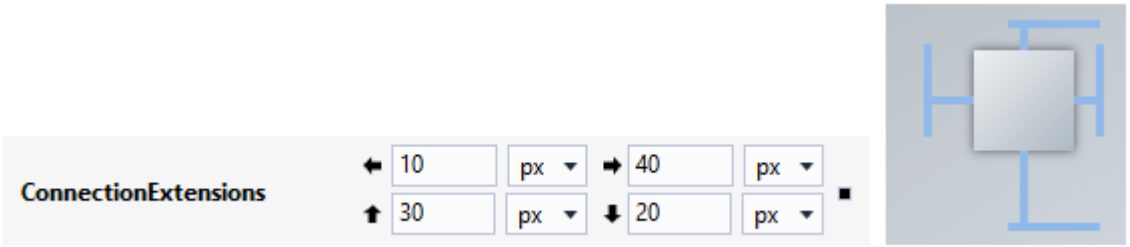
The length of the connection must be specified in each case.

The unit pixel is always used. Percent is **not** supported at this point.

ConnectionExtensions

```
tchmi:framework#/definitions/Padding
```

Here extensions can be created for the connections created above.



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.

ConnectionsColor

```
tchmi:framework#/definitions/SolidColor
```

Specification of the color for the connections.

ConnectionsColorPerSide

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.FourSidedColor
```

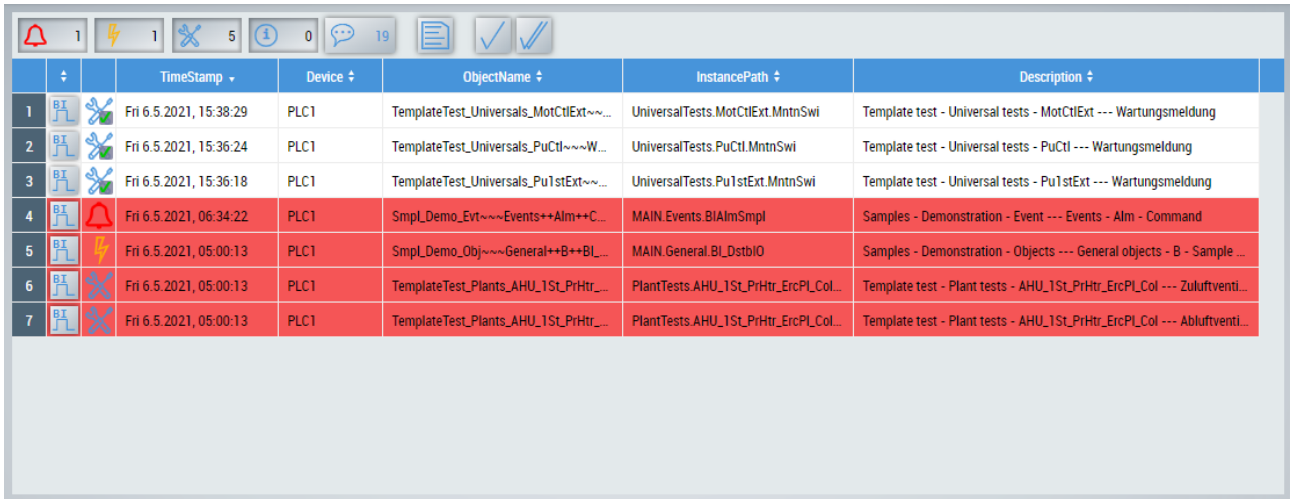
Defines the color for different connections. The *ConnectionsColor* attribute must be set to *NULL* or *NONE*.

6.2.1.2.3 Management

6.2.1.2.3.1 EventList

The **EventList** displays events in list form.

To use the control, the generic functionalities [▶ 77] of TcHmiBa must be used.



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 [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

IsGlobalEventList

```
tchmi:general#/definitions/Boolean
```

Determines whether the events of all connected controllers should be displayed.

ActiveEventsCount (read-only)

```
tchmi:general#/definitions/Number
```

Number of active events that the user can acknowledge.

Columns

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Management.EventList.ColumnList
```

Specifies the order and settings of the columns.

The default setting can be changed in the global settings [▶ 1266].

If the column width is specified in the unit "*factor*", the columns automatically use the remaining free space.

ColumnSorting

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Management.EventList.ColumnSorting
```

Has been replaced by the attribute *Columns*.

Specifies the column sorting.

The default attribute setting can also be overwritten globally for all EventList controls in CodeBehind.

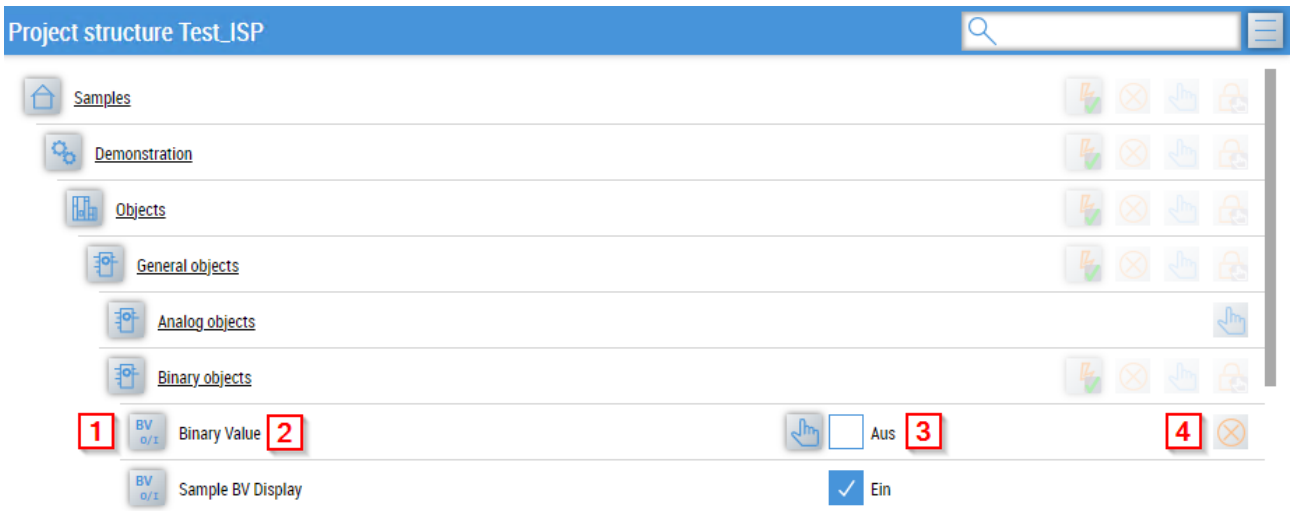
Events

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.

6.2.1.2.3.2 ProjectNavigationTextual

The **ProjectNavigationTextual** is one of the generic controls and is used to navigate through the project structure of a device. It provides information about the type (e.g. Analog Input or Structured View) [1], the description [2], the value [3] and the events (if available) [4] of the objects.

Here you will find more detailed information about the generic possibilities [▶ 77] of TcHmiBa and how they can be used.



Use

To navigate, the BaObject [▶ 1171] attribute must be linked. If a BaView is linked, you can navigate through its children. If only a single BaObject (no BaView) is linked, then only one entry is displayed with this BaObject.

Features

Generic navigation

The navigation is built generically based on the structure of the linked BaObject or BaView, which means that all BaObjects of a BaView can be reached with just one binding.

Manual override

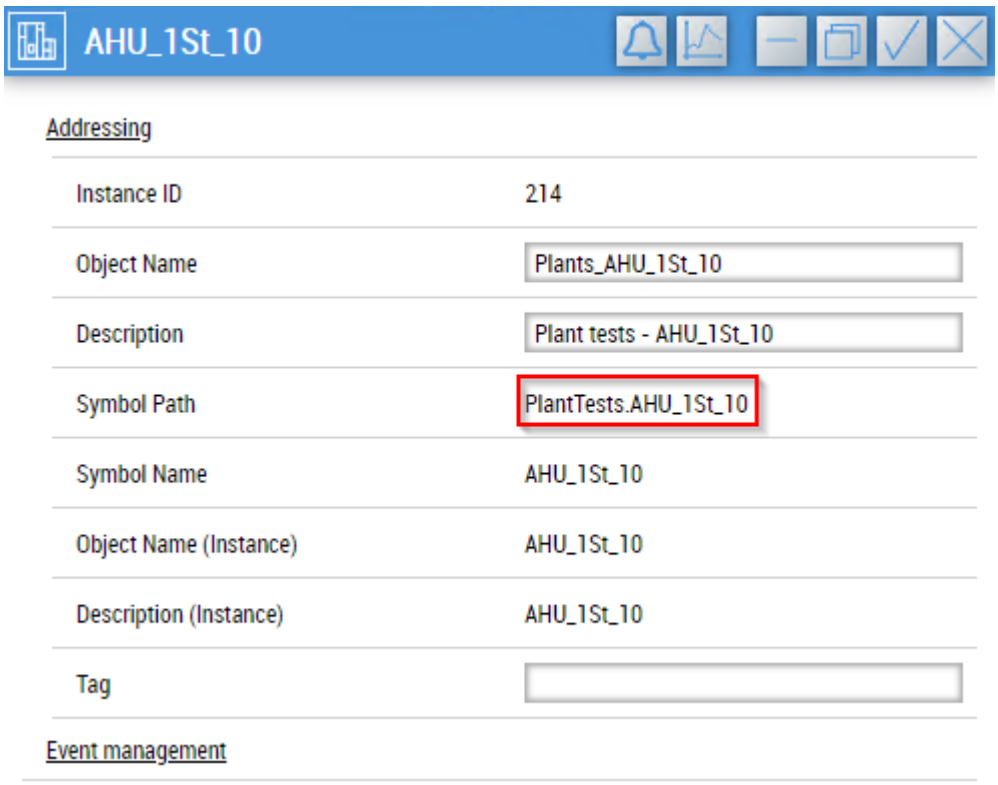
The value display and manual override is realized with the ManualOverride [▶ 1146] control.

Link content page

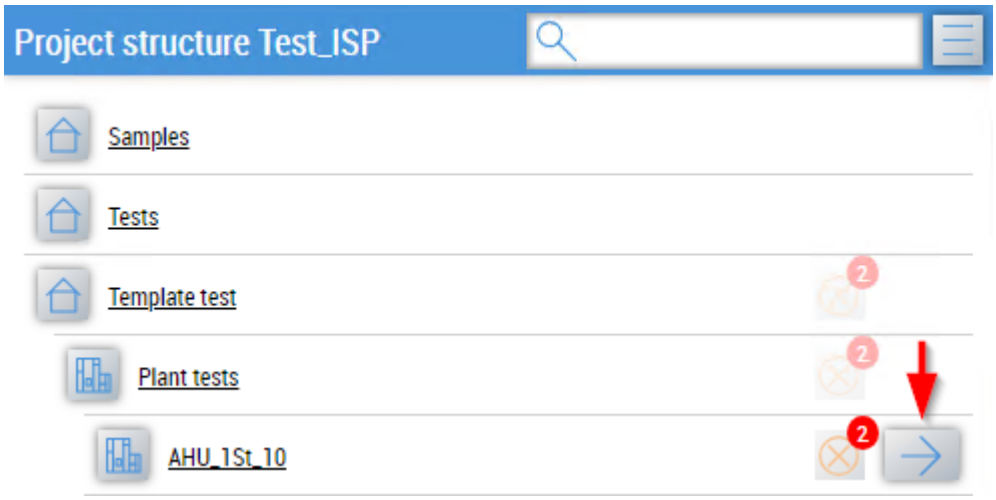
The function for linking content pages is an additional navigation aid.

A content page must exist in the TcHmi project that has the **Symbol Path** of a view as its name.

The name of the content page is case-sensitive.



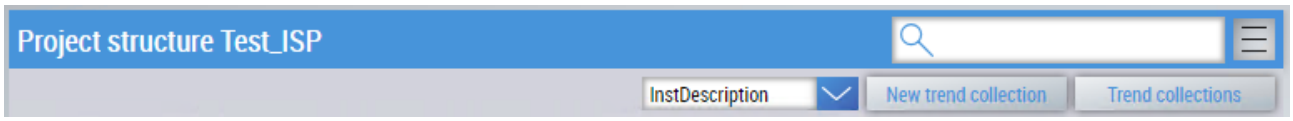
A button then appears behind the relevant entry in the project navigation.



In addition, a TcHmiRegion with the name **TargetRegion** must exist in the TcHmi project in order to display the linked content page.

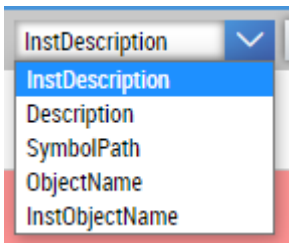
Header

The header contains an input field for searching the listing and a burger menu for displaying additional actions.



Actions:

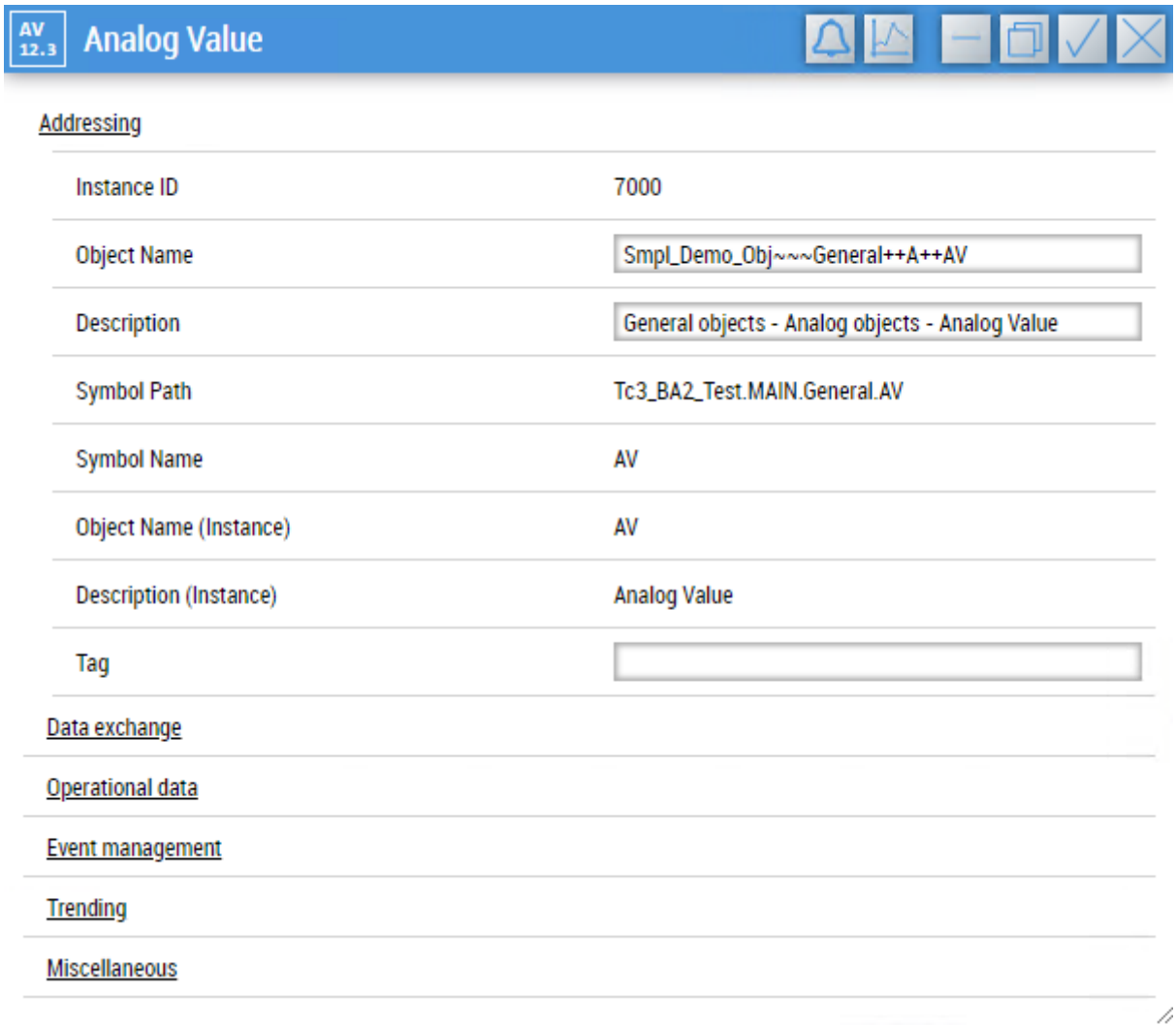
- Selection of labels to be displayed for the entries



- Opening the trend configurator [▶ 55]
- Opening generated trend configurations

Parameter dialog

After clicking on the type icon, the parameter dialog of the selected BaObject appears.



The parameters listed are divided into categories. Visibility and permitted read/write accesses depend on the role [▶ 15] of the logged-in user.

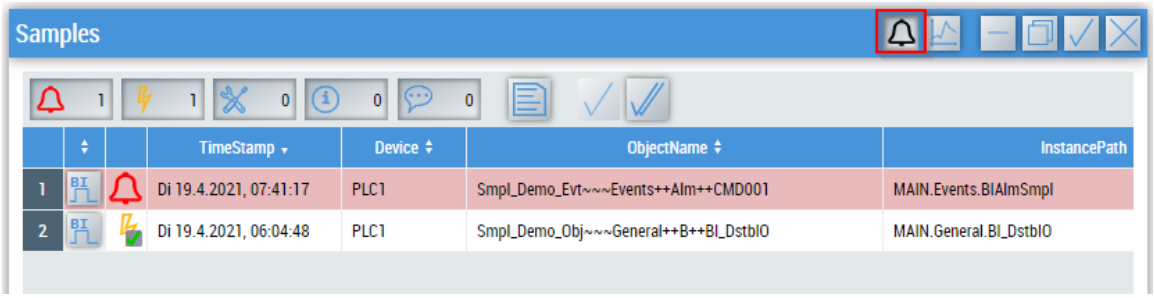
Changes to parameters are written to the PLC with the tick icon and discarded with the cross icon.

The category assignment and visibility of the parameters can be set globally (see Global settings [▶ 1266]).

Depending on the BaObject selected, additional buttons may appear in the header.

Buttons:

- Event bell
  - available for every event-capable object
  - replaces the content of the window with the event list [▶ 1166] of the object when clicked



- Trend functionalities
  - only available for objects that support them or views that contain objects with trend functions.

Attributes

The control inherits from BaseControl [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

BaUsedTitle

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.ProjectNavigationTextual.BaUsedTitle
```

Determines which parameter is used for the description in an entry. The setting can be customized in the client.

Show Header

```
tchmi:general#/definitions/Boolean
```

Determines whether the header is displayed or not.

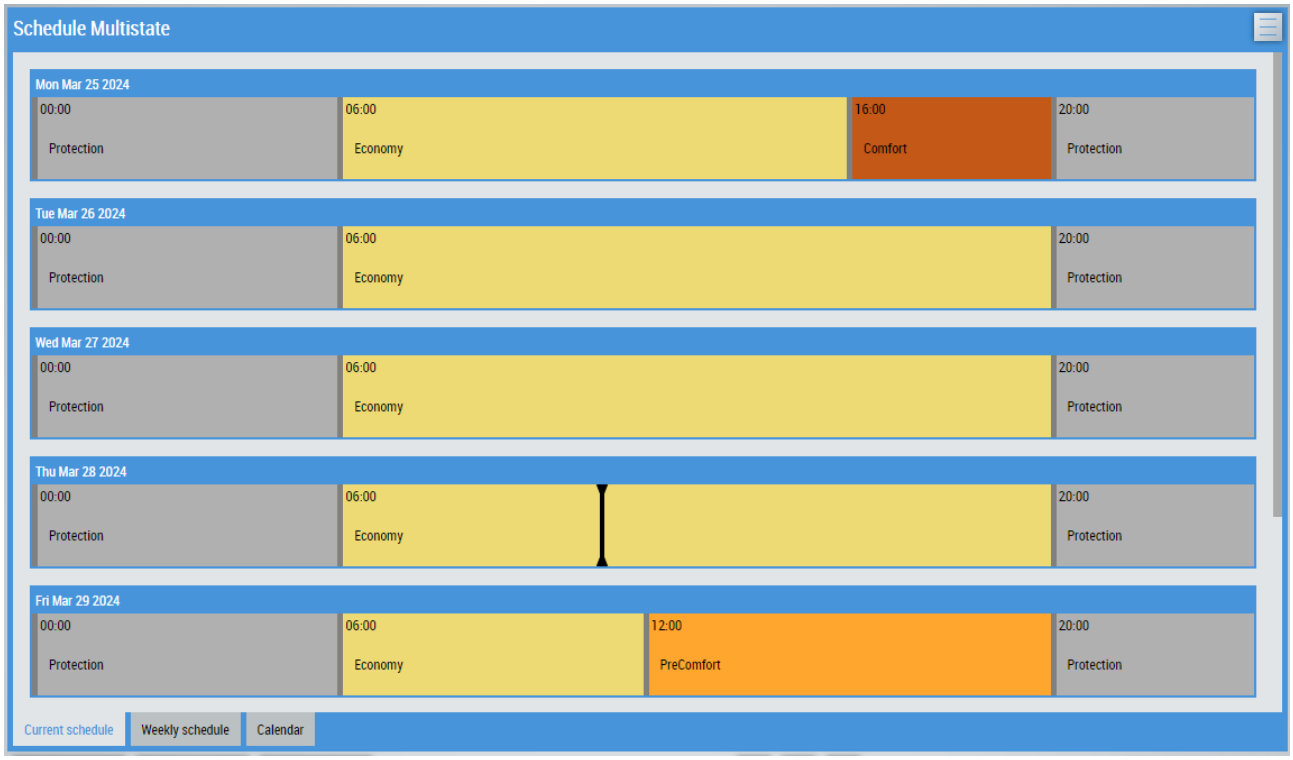
AutoCollapse

```
tchmi:general#/definitions/Boolean
```

Determines whether list entries are automatically closed when another list entry is selected.

6.2.1.2.3.3 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 [▶ 1174] attribute, the generic functions [▶ 77] can be used.

Features

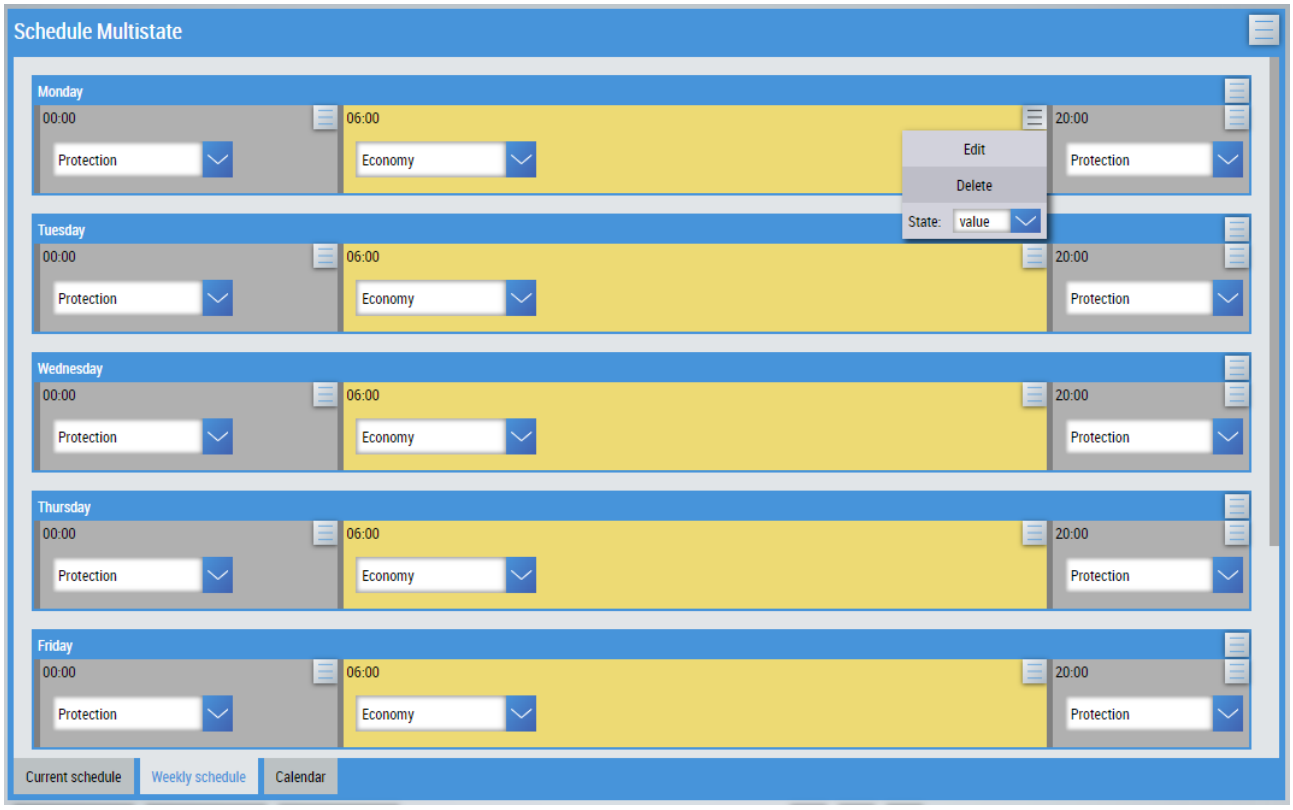
Resulting schedule

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

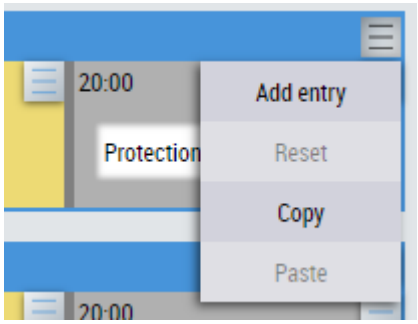
On the **Weekly schedule** tab it is possible to edit the weekly schedule without taking into account exceptions that have already been defined.



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.

By selecting the copy function, the day's entries are copied and can be inserted on other days.
Copying is also possible to other schedules or exceptions.



Managing exceptions

The exceptions are managed on the **Calendar** tab. For more information on how to use it, see the Calendar [▶ 1131] 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 [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

Orientation

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Orientation
```

Determines the orientation of the weekly schedule.

SnapPeriod

```
tchmi:general#/definitions/Number
```

Determines how precisely schedule entries can be set. If *SnapPeriode* is set to 15, for example, entries can be set to the nearest quarter of an hour.

DisplayTimeCursor

```
tchmi:general#/definitions/Boolean
```

Specifies whether a cursor for the current time is displayed in the resulting schedule.

CurrentDateTime

```
tchmi:general#/definitions/DateTime
```

Specifies the time that the cursor displays for the current time.

If a BaObject is used, the current time from the corresponding device is displayed.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol that fulfills the BaInterface [▶ 62] of the control.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Conrols.Management.Schedule.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

ActiveText

```
tchmi:general#/definitions/String
```

Specifies the text that is displayed in the entries if a binary schedule has been linked and the value is TRUE.

InactiveText

```
tchmi:general#/definitions/String
```

Specifies the text that is displayed in the entries if a binary schedule has been linked and the value is FALSE.

StateTexts

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Combobox.ComboboxItems
```

Specifies the texts that are displayed in the entries when a multistate schedule has been linked.

Unit

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Specifies the unit in the entries if an analog schedule has been linked. Possible values:

- textual (e.g. "°C")
- numerical (enumeration value of E_BA_Unit)

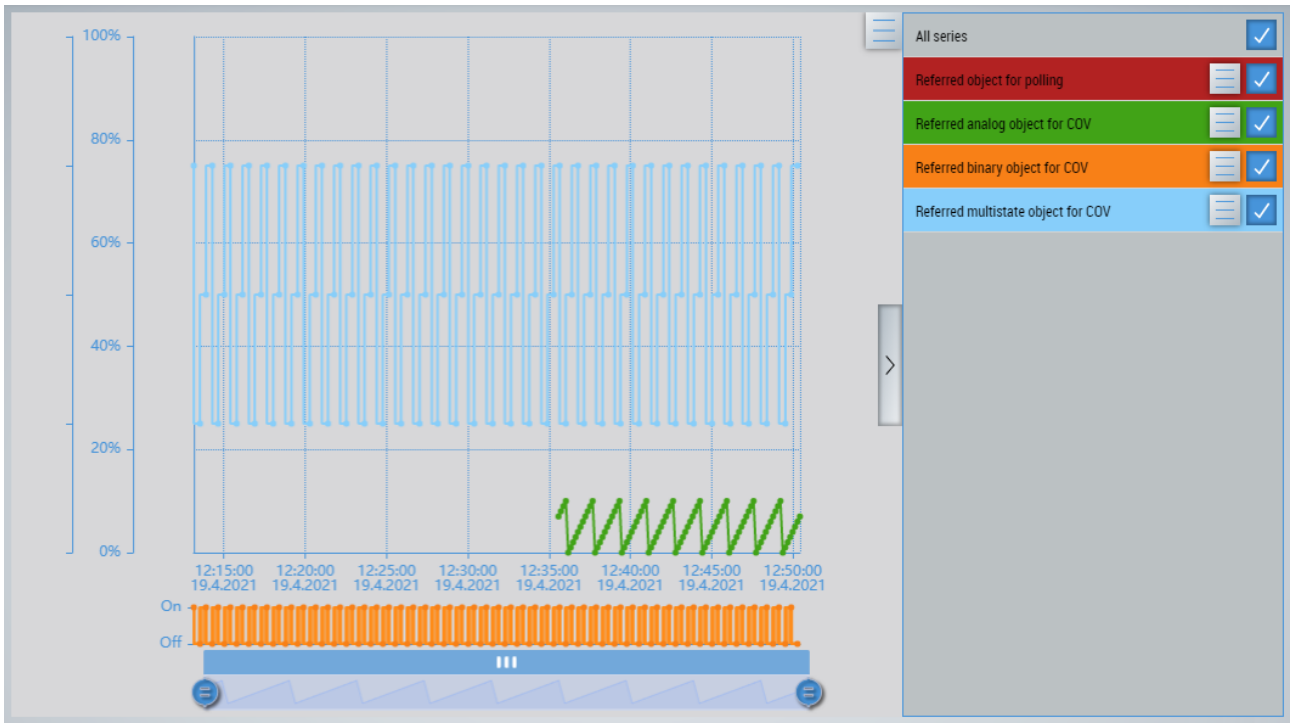
Digits

```
tchmi:general#/definitions/Number
```

Defines the number of decimal places if an analog schedule has been linked.

6.2.1.2.3.4 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 [▶ 1178] of type Trend object or View.

For more information, see the documentation on Trending [▶ 55].

Features

Multiple trend curves

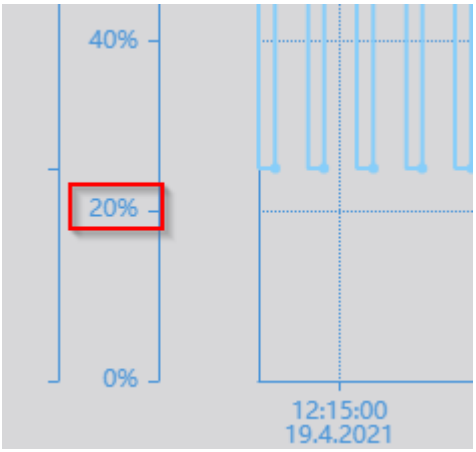
If the BaObject is a **trend object**, then only the associated trend curve is displayed. It is not possible to select from different trend curves.

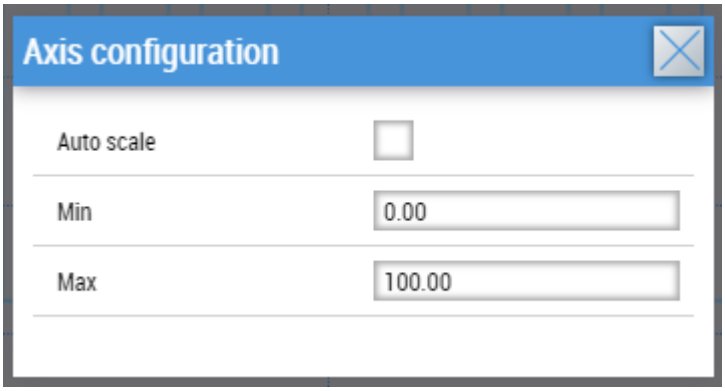
If the BaObject is a **View**, it is searched for trend objects and existing trend curves are displayed accordingly. It is possible to select from different trend curves if more than two trend objects are found.

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 [▶ 1170] of the respective trend object.

Axis parameterization

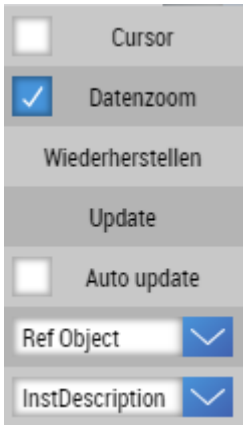
The settings of a y-axis can be opened by selecting the respective scale values.





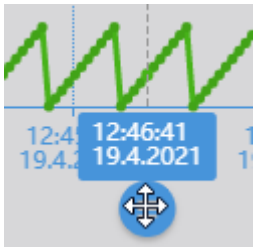
Menu

The **menu** allows further settings for the trend.



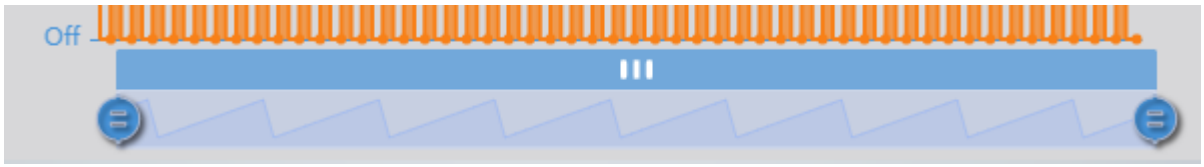
Cursor

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

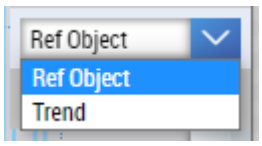
The trend curves can be updated once.

Auto Update

If the checkbox is checked, the trend curves are automatically updated as soon as new trend entries are available.

Displayed objects

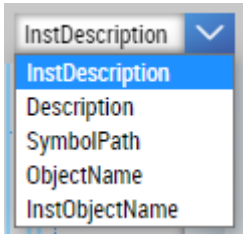
Determines the objects to be displayed in the listing.



- 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 [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

ShowDataZoom

```
tchmi:general#/definitions/Boolean
```

Defines whether the data zoom is displayed.

XAxesRange

```
tchmi:general#/definitions/Number
```

The time period in minutes displayed on the X-axis. Entering 24 * 60 = 1440 shows the last 24 hours.

SeriesSelectorWidth

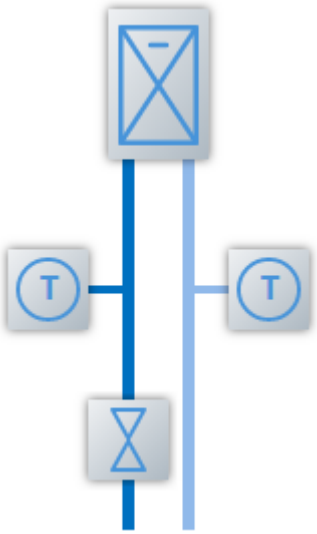
```
tchmi:general#/definitions/Number
```

The width of the selection of displayed trends in % if several trends are displayed.

6.2.1.2.4 Plants

6.2.1.2.4.1 Cooler

The **Cooler** template represents a cooler.



Use

Use on any page where a template of the **FB_BA_ColT_02** type is to be displayed.

Compatibility

The BaTemplateDescription [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
TFl	FB_BA_SensorAnalog_IO	Flow temperature sensor
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv [▶ 1092]	Valve

Hierarchy:

- BaObject
  - TFl
  - TRt
  - Vlv

Corresponds to the PLC template:

- FB_BA_AC_ColT_02

Attributes

This template inherits from the BaseTemplate [▶ 1195] control. In addition, there are the following attributes.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Cooler.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Flow Temperature Sensor

Setpoint

```
tchmi:general#/definitions/Boolean
```

Object for the setpoint.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

Return Temperature Sensor

Setpoint

```
tchmi:general#/definitions/Boolean
```

Object for the setpoint.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

Valve

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Valve.DisplayMode
```

Determines the icon to be displayed.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

Colors

FlowPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Color for the flow pipe.

ReturnPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Color for the return pipe.

6.2.1.2.4.2 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 [▶ 61] supports the following *BaObjects*.

Two-point

Subelements:

Symbol name	PLC template	Required	Description
SwiCls	FB_BA_BI_IO [▶ 205]	x	Switch close
SwiOpn	FB_BA_BI_IO [▶ 205]	x	Switch open

Hierarchy:

- BaObject
  - SwiCls
  - SwiOpn

Corresponds to the PLC template:

- FB_BA_Dmp2P [▶ 1058]

Analog

Subelements:

Symbol name	PLC template	Required	Description
MdIt	FB_BA_AO_IO [▶ 197]	x	Feedback

Hierarchy:

- BaObject
  - MdIt

Corresponds to the PLC template

- FB_BA_ActuatorAnalog [▶ 982]

Attributes

This template inherits from the UiIconFdbStp [▶ 1199] 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.

If "Custom" is selected, the icon [▶ 1130] that was set in the icon attribute is displayed.

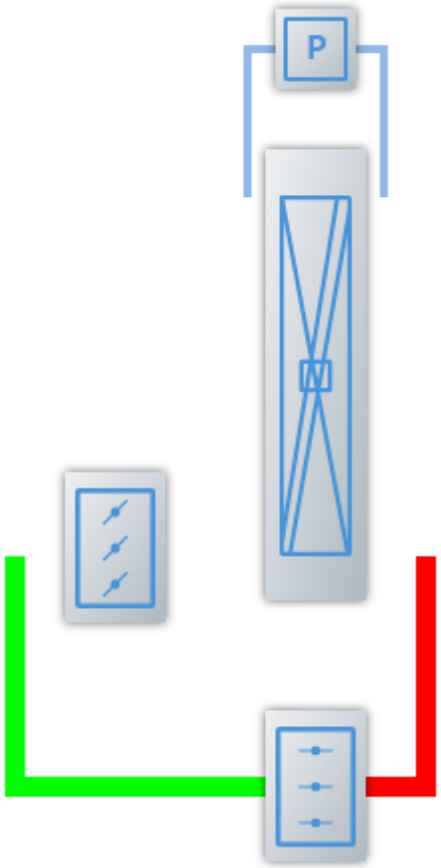
FlapPosition

```
tchmi:general#/definitions/Number
```

Position of the flaps in percent (0 is closed).

6.2.1.2.4.3 ErcPlate

The **ErcPlate** template is an energy recovery system with a plate heat exchanger.



Use

Use on any page where a template of the **ErcPI_02** type is to be displayed.

Compatibility

The BaTemplateDescription [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
ByDmp	FB_BA_ActuatorAnalog [▶ 982]	Bypass damper
DiffPrssSwi	FB_BA_SensorBinary_IO	Differential pressure
Dmp	FB_BA_ActuatorAnalog [▶ 982]	Damper

Hierarchy:

- BaObject
    - ByDmp

- DiffPrssSwi

- Dmp

Corresponds to the PLC template:

- FB_BA_AC_ErcPl_02 [▶ 733]

Attributes

This template inherits from the BaseTemplate [▶ 1195] control.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.ErcPlate.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Damper

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

Damper Bypass

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

Colors

FlowPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Color for the flow pipe.

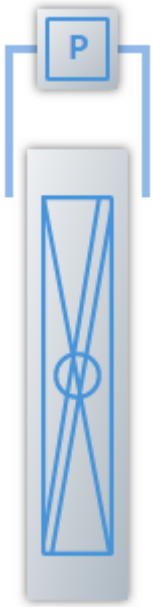
ReturnPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Color for the return pipe.

6.2.1.2.4.4 ErcRotation

The **ErcRotation** template is an energy recovery system with a rotary heat exchanger.



Use

Use on any page where a template of the **ErcRot_01** type is to be displayed.

Compatibility

The BaTemplateDescription [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
DiffPrssSwi	FB_BA_SensorBinary_IO	Differential pressure
MdIt	FB_BA_AO_IO [▶ 197]	Motor feedback
Motor	FB_BA_MotMdlt [▶ 1070]	Motor

Hierarchy:

- BaObject
    - DiffPrssSwi
    - Motor
        - MdIt

Corresponds to the PLC template:

- FB_BA_AC_ErcRot_01 [▶ 738]

Attributes

This template inherits from the BaseTemplate [▶ 1195] control.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.ErcRotation.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Common

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

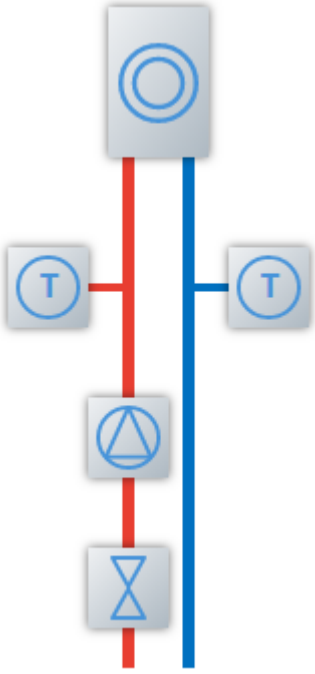
DisplayPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.UiIcon.DisplayPosition
```

Position of the setpoint and actual value.

6.2.1.2.4.5 HeatingCircuit

The **HeatingCircuit** template represents a heating circuit.



Use

Use on any page where a template of the **HtgCir01** type is to be displayed.

Compatibility

The BaTemplateDescription [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
Pu	FB_BA_Pu1st [▶ 1074]	Pump
Sp	FB_BA_H_HtgCir_Sp	Setpoint
SpFlWT	FB_BA_AV_Op [▶ 202]	Setpoint flow temperature sensor
TFl	FB_BA_SensorAnalog_IO	Flow temperature sensor
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv [▶ 1092]	Valve

Hierarchy:

- BaObject
    - Pu
    - Sp
        - SpFlWT
    - TFl
    - TRt
    - Vlv

Corresponds to the PLC template:

- FB_BA_H_HtgCir01 [▶ 975]

Attributes

This template inherits from the template FB_BA_AC_ColT_02 [▶ 1178].

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.HeatingCircuit.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Flow Temperature Sensor

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

Pump

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

6.2.1.2.4.6 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 [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
Cmd	FB_BA_BO_IO [▶ 209]	x	Command
MdIt	FB_BA_BO_IO [▶ 209]		Feedback

Hierarchy:

- BaObject
  - Cmd
  - MdIt

Corresponds to the PLC templates

- FB_BA_MotCtl [▶ 1066]
- FB_BA_MotCtlExt [▶ 1069]
- FB_BA_Pu1st [▶ 1074]
- FB_BA_Pu1stExt [▶ 1076]
- FB_BA_PuCtl [▶ 1077]

Attributes

This template inherits from the UiIconFdbStp [▶ 1199] control. In addition, there are the following attributes.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Motor.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Common

DisplayMode

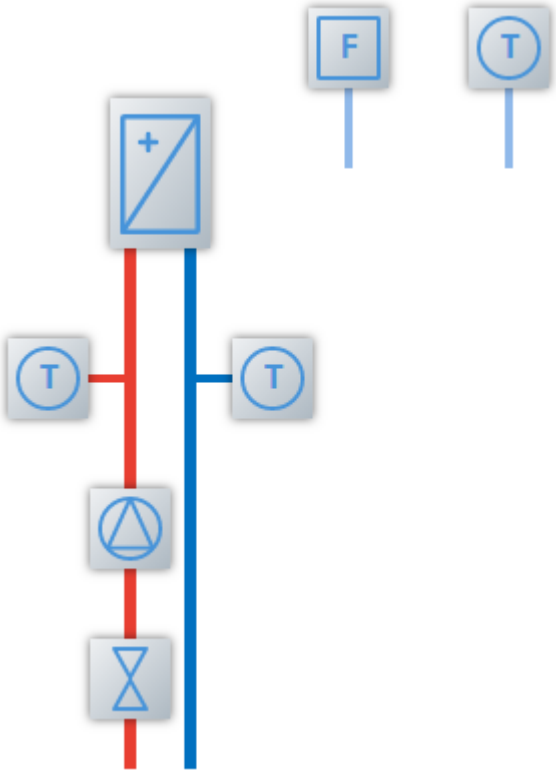
```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Motor.DisplayMode
```

Determines the icon to be displayed. If "Custom" is selected, the icon that was set in the icon [▶ 1130] attribute is displayed.

If "Custom" is selected, the icon [▶ 1130] that was set in the icon attribute is displayed.

6.2.1.2.4.7 PreHeater

The **PreHeater** template is a preheater.



Use

Use on any page where a template of the **PreHtr** type is to be displayed.

Compatibility

The BaTemplateDescription [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Description
FrostThermostat	FB_BA_SensorBinary_IO	Frost protection thermostat
Pu	FB_BA_Pu1st [▶ 1074]	Pump
Sp	FB_BA_H_HtgCir_Sp	Setpoint
SpFlWT	FB_BA_AV_Op [▶ 202]	Setpoint flow temperature sensor
TFl	FB_BA_SensorAnalog_IO	Flow temperature sensor
TFrost	FB_BA_AO_IO [▶ 197]	Temperature frost
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv [▶ 1092]	Valve

Hierarchy:

- BaObject
    - FrostThermostat
    - Pu
    - Sp
        - SpFlWT

- TFl
- TFrost
- TRt
- Vlv

Corresponds to the PLC template:

- FB_BA_AC_PreHtr [▶ 748]

Attributes

This template inherits from the template HeatingCircuit [▶ 1185].

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.PreHeater.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Frost Temperature Sensor

Setpoint

```
tchmi:general#/definitions/Boolean
```

Object for the setpoint.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

6.2.1.2.4.8 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 [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
Cmd	FB_BA_BO_IO [▶ 209]	x	Command
Mdlt	FB_BA_BO_IO [▶ 209]		Feedback

Hierarchy:

- BaObject
    - Cmd
    - MdIt

Corresponds to the PLC template

- FB_BA_MotCtl [▶ 1066]
- FB_BA_MotCtlExt [▶ 1069]
- FB_BA_Pu1st [▶ 1074]
- FB_BA_Pu1stExt [▶ 1076]
- FB_BA_PuCtl [▶ 1077]

Attributes

This template inherits from the UiIconFdbStp [▶ 1199] control. In addition, there are the following attributes.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Pump.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Common

DisplayMode

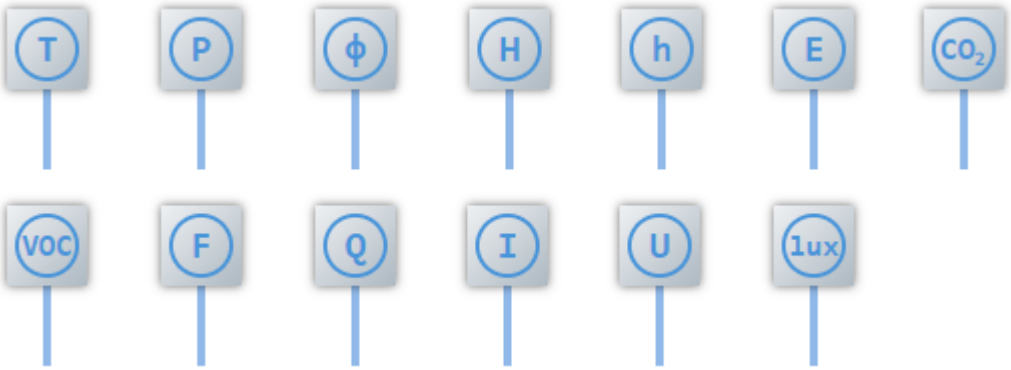
```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Pump.DisplayMode
```

Determines the icon to be displayed.

If "Custom" is selected, the icon [▶ 1130] that was set in the icon attribute is displayed.

6.2.1.2.4.9 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 [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
MV	FB_BA_AI_Raw [▶ 195]	x	Feedback

Hierarchy:

- BaObject

  - MV

Corresponds to the PLC template

- FB_BA_SensorAnalog [▶ 1087]

Attributes

This template inherits from the control Sensor [▶ 1196]. In addition, there are the following attributes.

BA

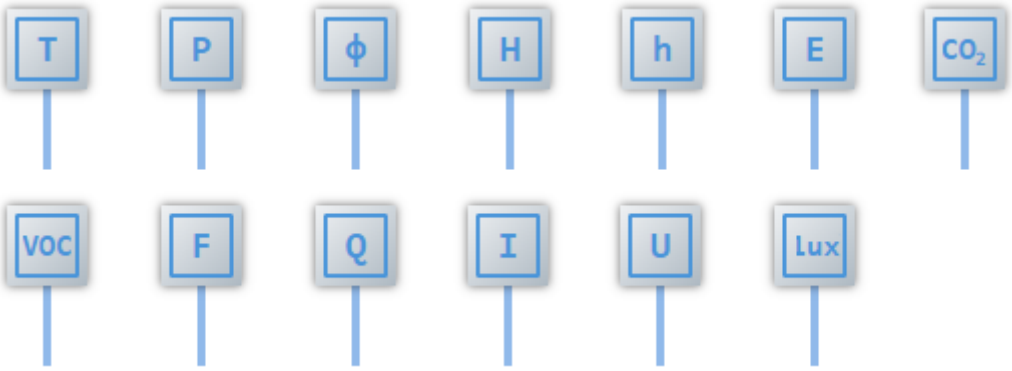
BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

6.2.1.2.4.10 SensorBinary

The **SensorBinary** template displays the feedback of a binary value and, if linked, a setpoint value.



Use

Use on any page where a template of the **SensorBinary** type is to be displayed.

Compatibility

The BaTemplateDescription [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
Input	FB_BA_BI_Raw [▶ 206]	x	Value

Hierarchy:

- BaObject

    - Input

Corresponds to the PLC template:

- FB_BA_SensorBinary [▶ 1088]

Attributes

This template inherits from the control Sensor [▶ 1196]. In addition, there are the following attributes.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorBinary.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

6.2.1.2.4.11 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 [▶ 61] supports the following *BaObjects*.

Three-point

Subelements:

Symbol name	PLC template	Required	Description
Anlg3Pnt	FB_BA_Analog3Pnt [▶ 995]	x	
Pos	FB_BA_AV_Op [▶ 202]	x	Feedback

Hierarchy:

- BaObject

    - Anlg3Pnt

        - Pos

Corresponds to the PLC template

- FB_BA_Vlv3pt [▶ 1094]

Analog value

Subelements:

Symbol name	PLC template	Required	Description
Fdb	FB_BA_AI_IO [▶ 192]	x	Feedback
MdIt	FB_BA_AO_IO [▶ 197]		Setpoint

Hierarchy:

- BaObject
  - Fdb
  - MdIt

Corresponds to the PLC template

- FB_BA_Vlv [▶ 1092]

Attributes

This template inherits from the UiIconFdbStp [▶ 1199] control. In addition, there are the following attributes.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Valve.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Common

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Valve.DisplayMode
```

Determines the icon to be displayed.

If "Custom" is selected, the icon [▶ 1130] that was set in the icon attribute is displayed.

6.2.1.2.4.12 VAV

The **VAV** template shows the setpoint of a volume flow controller and, if available, the setpoint.

Use

Use on any page where a template of the **VAV** type is to be displayed.

Compatibility

The BaTemplateDescription [▶ 61] supports the following *BaObjects*.

Subelements:

Symbol name	PLC template	Required	Description
MdIt	FB_BA_AO_IO [▶ 197]		Setpoint
Fdb	FB_BA_AI_IO [▶ 192]	x	Feedback

Hierarchy:

- BaObject

- Mdlt
- Fdb

Corresponds to the PLC template:

- FB_BA_ActuatorAnalog [▶ 982]

Attributes

This template inherits from the UiIconFdbStp [▶ 1199] control. In addition, there are the following attributes.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.VAV.BaTemplateDescription
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Common

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.VAV.DisplayMode
```

Determines the icon to be displayed.

If "Custom" is selected, the icon [▶ 1130] that was set in the icon attribute is displayed.

6.2.1.2.5 System

6.2.1.2.5.1 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 access.

The attribute is not applicable to all controls.

Layout

ContentPadding

```
tchmi:framework#/definitions/Padding
```

Specifies the padding for the content of the control.

The attribute is not applicable to all controls.

6.2.1.2.5.2 BaseTemplate

The **BaseTemplate** is the basis for all more sophisticated template controls (e.g. PreHtr, HtgCir), which are more than just a UiIcon [▶ 1159]. 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 [▶ 61] 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
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

BaTemplateDescription

```
tchmi:general#/definitions/Object
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

ShowTags

```
tchmi:general#/definitions/Boolean
```

Determines whether the tags are displayed or not.

6.2.1.2.5.3 Sensor

The sensor is a specialized form of UiIconFdbStp [▶ 1199] and serves as the basis for sensor controls (e.g. 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 various display options of sensors.

Attributes

The control inherits from UiIconFdbStp [▶ 1199] and thus has the same attributes. In addition, there are the following attributes.

Common

PinLayout

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.System.Sensor.PinLayout
```

Specifies the layout of the pins.

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.System.Sensor.DisplayMode
```

Specifies the icon to be displayed.

If "Custom" is selected, the icon [▶ 1130] that was set in the icon attribute is displayed.

PinPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position
```

Specifies the orientation of the pins.

PinConnection

```
tchmi:general#/definitions/Number
```

Specifies the line length of the first pin.

PinExtension

```
tchmi:general#/definitions/Number
```

Specifies the line length from the first pin.

PinWidth

```
tchmi:general#/definitions/Number
```

Specifies the line width of the pins.

CustomLetter

```
tchmi:general#/definitions/String
```

Specifies the letter to be displayed in the icon if the DisplayMode [▶ 1130] attribute has the value "Custom".

ShowValue

```
tchmi:general#/definitions/Boolean
```

Specifies whether the value is visible.

6.2.1.2.5.4 BaseRoomControl

BaseRoomControl is the basis for all room controls (e.g. Light [▶ 1203], Sunblind [▶ 1210], HeatingCooling [▶ 1200], Window [▶ 1214]). 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 [▶ 61] for all inheriting controls.

Attributes

The control inherits from BaseControl [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

BA

BaInterfaceSymbolNames

```
tchmi:general#/definitions/Object
```

Edit the BaInterfaceSymbolNames [▶ 1246].

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol that fulfills the BaInterface [▶ 62] of the control.

Error

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.RoomAutomation.HeatingCooling.EnergyLevel
```

If TRUE, an error message is displayed.

ShowPriority

```
tchmi:general#/definitions/Boolean
```

Defines whether the priority is displayed.

6.2.1.2.5.5 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 [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

Colors

TextColor

```
tchmi:framework#/definitions/SolidColor
```

Color of texts.

TextDecorationColor

```
tchmi:framework#/definitions/SolidColor
```

Color of text decorations.

Text

TextVerticalAlignment

```
tchmi:framework#/definitions/VerticalAlignment
```

Vertical alignment of texts.

TextHorizontalAlignment

```
tchmi:framework#/definitions/HorizontalAlignment
```

Horizontal alignment of texts.

TextFontSize

```
tchmi:framework#/definitions/MeasurementValue
```

Font size of texts. Percentages are relative to the font size of the parent element.

TextFontSizeUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit for the font size of texts. Can be absolute (px) or relative (%).

TextFontFamily

```
tchmi:framework#/definitions/FontFamily
```

Font of texts.

TextFontStyle

```
tchmi:framework#/definitions/FontStyle
```

Font style of texts.

TextFontWeight

```
tchmi:framework#/definitions/FontWeight
```

Font weight of texts.

TextDecorationLine

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationLine
```

Position of the text decoration.

TextDecorationStyle

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationStyle
```

Style of text decoration.

UserSelect

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.UserSelect
```

Behavior when selecting the text of a user.

TextOverflow

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextOverflow
```

Defines how to display text that is wider than the control.

6.2.1.2.5.6 UiIconFdbStp

The UiIconFdbStp is a specialized form of the UiIcon [▶ 1159] 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 UiIcon [▶ 1159] and thus has the same attributes. In addition, there are the following attributes.

BA

```
tchmi:general#/definitions/Object
```

Edit the *BaTemplateDescription* of the BaTemplate [▶ 61].

Common

DisplayMode

```
tchmi:general#/definitions/Number
```

Determines the icon to be displayed.

If "Custom" is selected, the icon [▶ 1130] that was set in the icon attribute is displayed.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Determines whether the feedback is visible.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint is visible.

ShowDisplays

```
tchmi:general#/definitions/Boolean
```

Determines whether the setpoint and actual value is visible.

DisplayPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.UiIcon.DisplayPosition
```

Position of the setpoint and actual value.

6.2.1.2.6 RoomAutomation

6.2.1.2.6.1 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

The following table shows the possible operation modes.

Operation mode	Symbol
Heating	
Cooling	
Inactive	

If the plant is inactive, the control is in the neutral zone.

Displays

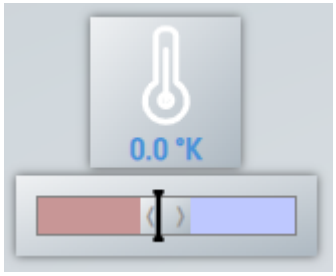
The user level determines the information available in the display.
For users with *Advanced* level or higher, the current values for temperature and setpoint are visible.

For users up to Basic level, only the current setpoint adjustment is displayed.

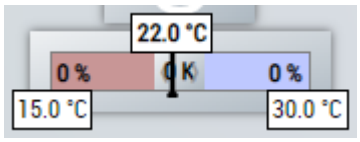
Operation

The **HeatingCooling** control displays the current state of the plant and opens the activity display for setting the room temperature with a click.

The temperature setpoints are adjusted by moving the slider. The cursor in the slider represents the room temperature by its position.

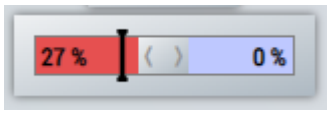


The room temperature and limits are visible from the "Advanced" user level.

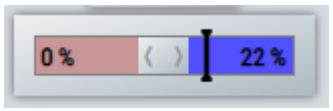


The activity display also shows the operation mode. It can be binary or in percent.

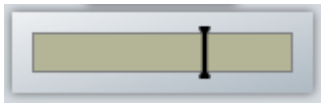
Heating:



Cooling:



The control is only active if the active energy level corresponds to "Comfort".



Attributes

The control inherits from BaseRoomControl [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

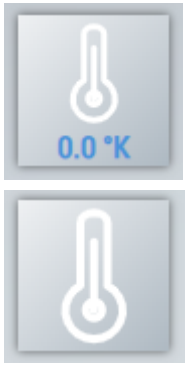
The attribute is not applicable to all controls.

Common

ShowTemperatures

```
tchmi:general#/definitions/Boolean
```

If TRUE, the temperatures are displayed.



BaData

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.RoomAutomation.HeatingCooling.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

Temperature

RoomTemp

```
tchmi:general#/definitions/Number
```

Current measured room temperature.

RoomTempAdjust

```
tchmi:general#/definitions/Number
```

Current room temperature adjustment.

RoomTempAdjustFeedback

```
tchmi:general#/definitions/Number
```

Feedback for the current room temperature adjustment.

RoomTempAdjustRange

```
tchmi:general#/definitions/Number
```

Specifies the range of the room temperature adjustment.

HeatingSetpoint

```
tchmi:general#/definitions/Number
```

Current setpoint for heating mode.

CoolingSetpoint

```
tchmi:general#/definitions/Number
```

Current setpoint for cooling mode.

HeatingActive

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

If TRUE or greater than 0, then heating is displayed.

CoolingActive

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

If TRUE or greater than 0, then cooling is displayed.

Unit

```
tchmi:general#/definitions/String
```

Unit for displaying the temperatures.

Events

Event	Description
onTempAdjustChanged	Triggered when the user changes the temperature adjustment.

6.2.1.2.6.2 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

Priorities

Display of different priorities via the attribute *Priority*.

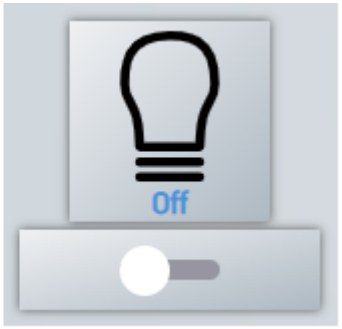
Brightness specification

The brightness can be adjusted using an analog (dimmable) or binary (switchable) value.
Buttons with predefined brightness values are available for dimmable lamps.

The display of these buttons can be set globally [▶ 1267]:

Operation

Clicking on the **Light** control changes the visibility of the menu for adjusting the brightness.



Attributes

The control inherits from BaseRoomControl [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

`tchmi:framework#/definitions/Symbol`

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

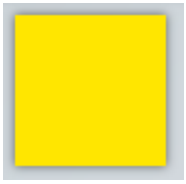
The attribute is not applicable to all controls.

Common

DisplayMode

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.DisplayMode`

Determines the display mode of the lamp.

Name	Presentation
lightBulb	
lightBulbFilled	
filles	

ShowValue

`tchmi:general#/definitions/Boolean`

If TRUE, the brightness value is displayed.

BaData

Priority

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority`

The current priority displayed on the control. High priorities (low value in priority enumeration) deactivate the control's operating options.

Priority	Presentation
fire	
communicationError	
burglaryAlarm	
maintenance	
cleaning	
nightWatch	
simple	
manual	
automaticLight	
scene1…scene3	

The icons used for the various priorities can be set globally [▶ 1267].

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Light.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

Brightness

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Current brightness value (0-100%).

BrightnessFeedback

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Feedback for the brightness value (0-100%).

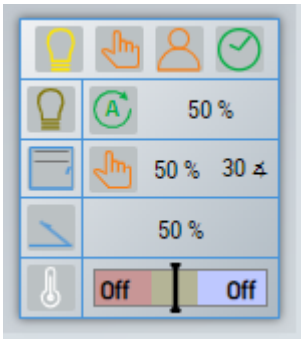
Events

Event	Description
onBrightnessChanged	Triggered when the user changes the brightness.
onHandModeReset	Triggered when the button for resetting from "hand" to automatic mode is pressed.

6.2.1.2.6.3 RoomControl

The **RoomControl** can combine the various controls of the room automation. Available components are:

- HeatingCooling [▶ 1200]
- Light [▶ 1203]
- Sunblind [▶ 1210]
- Window [▶ 1214]



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

Operating options and settings correspond to those of the individual controls.

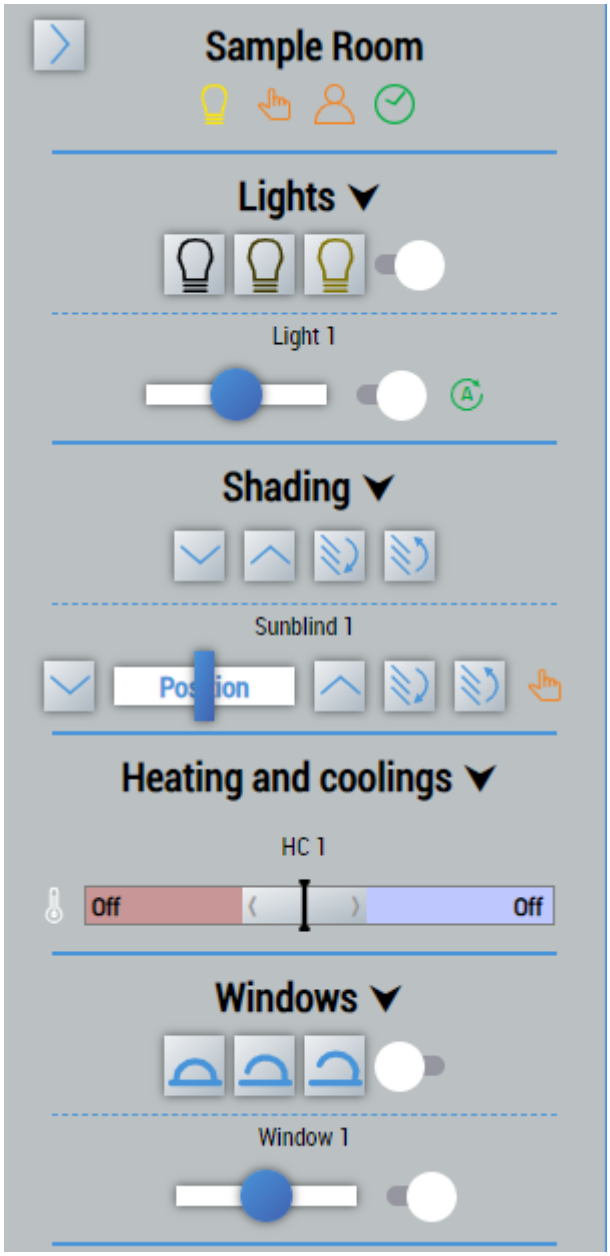
Room status

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 a side menu containing the controls for the individual areas.



Central control

Each area contains a header with quick settings for uniform control of all controls.

Attributes

The control inherits from BaseControl [▶ 1194] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

ControlUnits

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomControl.ControlList
```

Specifies which components (HeatingCooling [▶ 1200], Light [▶ 1203], Sunblind [▶ 1210], Window [▶ 1214]) are to be added.

Name

```
tchmi:general#/definitions/String
```

The room name.

HideRoomStatus

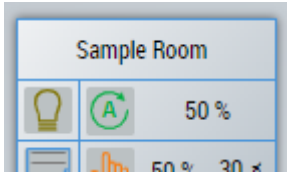
```
tchmi:general#/definitions/Boolean
```

If TRUE, then the room information (header) is not visible.

ShowRoomName

```
tchmi:general#/definitions/Boolean
```

If TRUE, the room name [▶ 1208] is displayed instead of the room information.



BaData

Presence

```
tchmi:general#/definitions/Boolean
```

If TRUE, presence was detected in the room.

- Presence active

- Presence inactive

SwitchOffDelayActive

```
tchmi:general#/definitions/Boolean
```

If TRUE, the delay for the automatic system is active.

- Delay active

- Delay 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.

Further information on the attributes can be found in the documentation for the Light [▶ 1203] control.

Sunblinds

ShowSunblinds

```
tchmi:general#/definitions/Boolean
```

If TRUE, the sunblinds are displayed in the control. The components are always visible in the side menu.

Further information on the attributes can be found in the Sunblind [▶ 1210] control documentation.

HeatingCooling

ShowHeatingCooling

```
tchmi:general#/definitions/Boolean
```

If TRUE, the HeatingCooling applications are displayed in the control. The components are always visible in the side menu.

Further information on the attributes can be found in the documentation of the HeatingCooling [▶ 1200] control

Windows

ShowWindows

```
tchmi:general#/definitions/Boolean
```

If TRUE, the windows are displayed in the control. The components are always visible in the side menu.

Further information on the attributes can be found in the documentation of the Window [▶ 1214] control.

6.2.1.2.6.4 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

Priorities

Display of different priorities via the attribute *Priority*.

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 [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

The attribute is not applicable to all controls.

Common

Controls

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Controls`

Specifies the type of controls to be used for position and angle settings.

- sliderHorizontal

- buttons

ShowValue

`tchmi:general#/definitions/Boolean`

If TRUE, the values for angle and position are displayed.

BaData

Priority

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority`

The current priority displayed on the control. High priorities (low value in priority enumeration) deactivate the control's operating options.

Name	Presentation
fire	
storm	
ice	
communicationError	
burglaryAlarm	
maintenance	
referencing	
manualActuator	
manualGroup	
allDown	
allUp	
scene1…scene3	

Name	Presentation
facadeThermoAutomatic	
facadeTwilightAutomatic	
parkPosition	
sunProtection	
groupThermoAutomatic	
groupTwilightAutomatic	

The icons used for the various priorities can be set globally [▶ 1268].

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Sunblind.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

Position

```
tchmi:general#/definitions/Number
```

Current position (0-100%).

PositionFeedback

```
tchmi:general#/definitions/Number
```

Feedback for the position (0-100%).

Angle

UseAngle

```
tchmi:general#/definitions/Boolean
```

If TRUE, the controls for controlling the angle are displayed.

Angle

```
tchmi:general#/definitions/Number
```

Current angle (0-100%).

AngleFeedback

```
tchmi:general#/definitions/Number
```

Feedback for the angle (0-100%).

AngleStep

```
tchmi:general#/definitions/Number
```

Determines the step size with which the angle is adjusted via the angle buttons.

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.
onManualReset	Triggered when the button for resetting from "hand" to automatic mode is pressed.

6.2.1.2.6.5 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

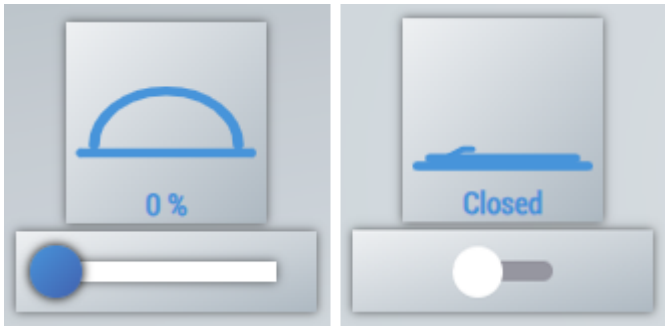
Position specification

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 globally [▶ 1269].

Operation

Clicking on the **Window** control changes the visibility of the menu for setting the position.



Attributes

The control inherits from BaseRoomControl [▶ 1197] and thus has the same attributes. In addition, there are the following attributes.

Feedback concept

The control can use the feedback concept [▶ 60].

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol for BaObject [▶ 60] for using the generic functionalities [▶ 77] of TcHmiBa. Links a single object or a complete view (including children) to the control.

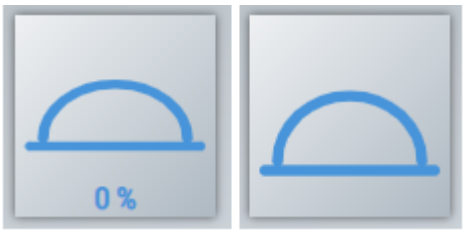
The attribute is not applicable to all controls.

Common

ShowValue

```
tchmi:general#/definitions/Boolean
```

If TRUE, the value for position is displayed.



DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Window.DisplayMode
```

Determines the display mode of the window.

Name	Presentation
roofDome	
window	

IconRotation

```
tchmi:general#/definitions/Number
```

Rotates the icon. Angle in degrees.

BaData

Priority

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Window.Priority
```

The current priority displayed on the control. High priorities (low value in priority enumeration) deactivate the control's operating options.

Name	Presentation
fire	
storm	
ice	
communicationError	
burglaryAlarm	
maintanence	
referencing	
manualActuator	
manualGroup	
allDown	
allUp	
scene1…scene3	

Name	Presentation
facadeThermoAutomatic	
facadeTwilightAutomatic	
parkPosition	
sunProtection	
groupThermoAutomatic	
groupTwilightAutomatic	

The icons used for the various priorities can be set globally [▶ 1269].

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Window.BaInterfaceSymbolNames
```

Edit the BaInterfaceSymbolNames [▶ 1246].

Position

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Current position (0-100%).

PositionFeedback

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Feedback for the position (0-100%).

Events

Event	Description
onPositionChanged	Triggered when the user changes the position of the window or a roof dome.
onManualReset	Triggered when the button for resetting from "hand" to automatic mode is pressed.

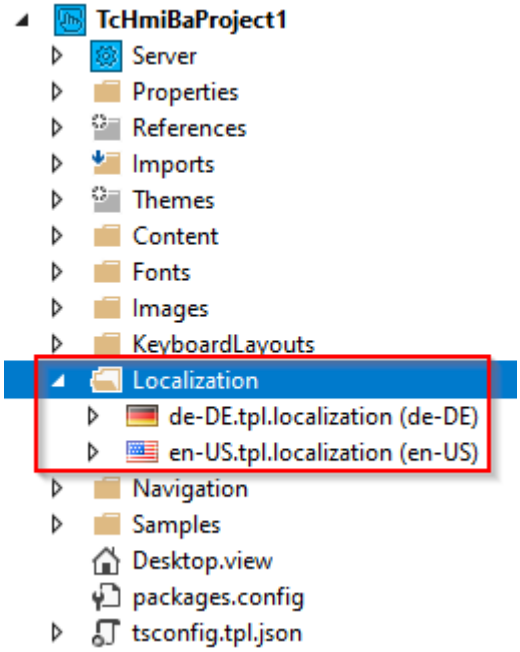
6.2.1.3 Guidlines

6.2.1.3.1 Localization

Localization in the TcHmiBa controls is managed by the TwinCAT HMI localization system.

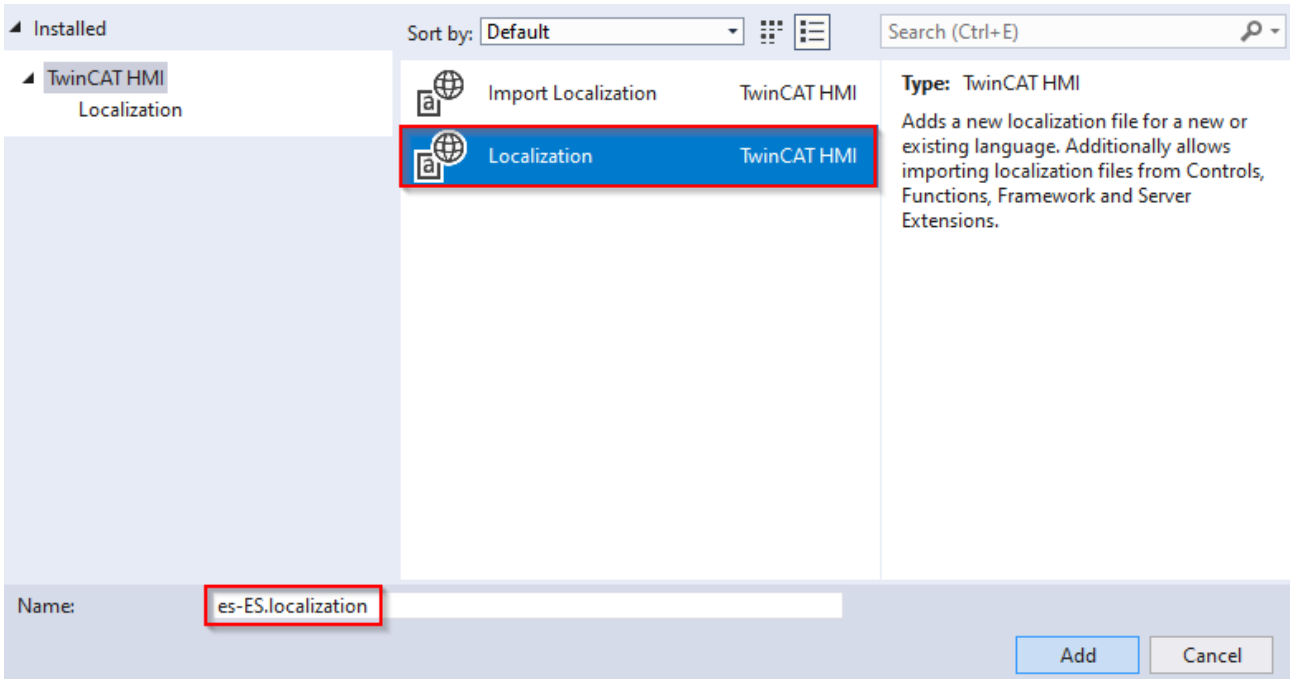
Localization in the project

A TcHmi project created with the TcHmiBa project template already contains localizations for German and English. They can be modified, supplemented and extended to include other languages.

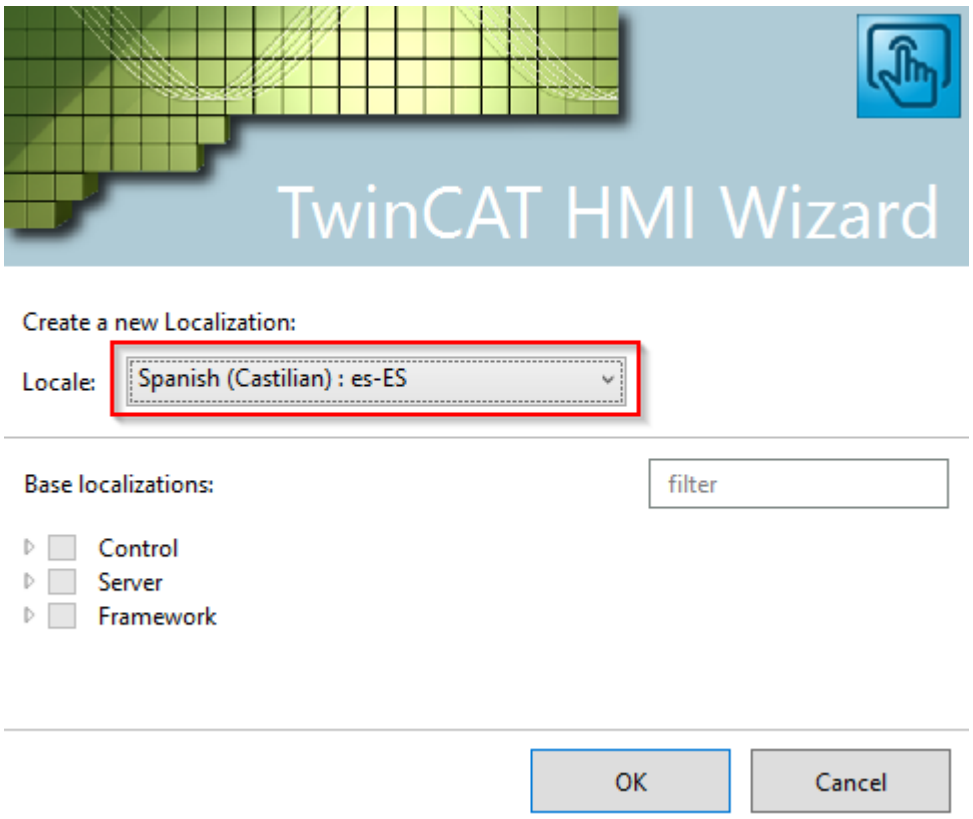


Add localization

Each new language requires a new localization file in the localization folder.

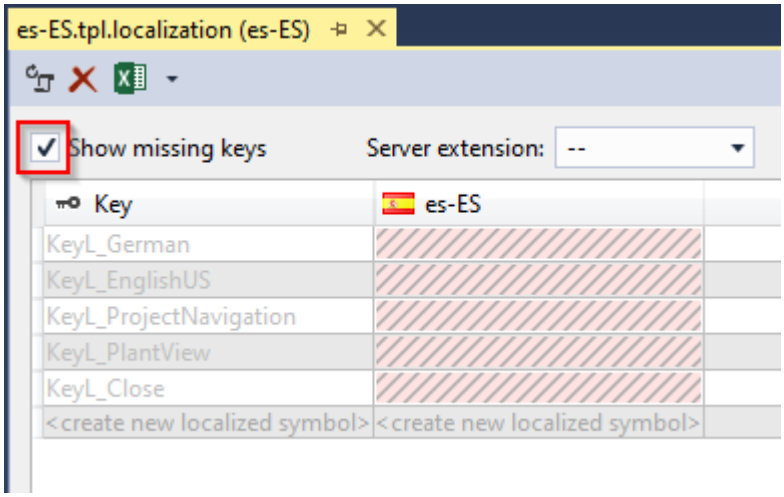


The appropriate combination of language and area must then be selected.



A newly created localization file is automatically opened and initially appears empty.

Existing language entries from other languages can be shown by activating the **Show missing keys** option.

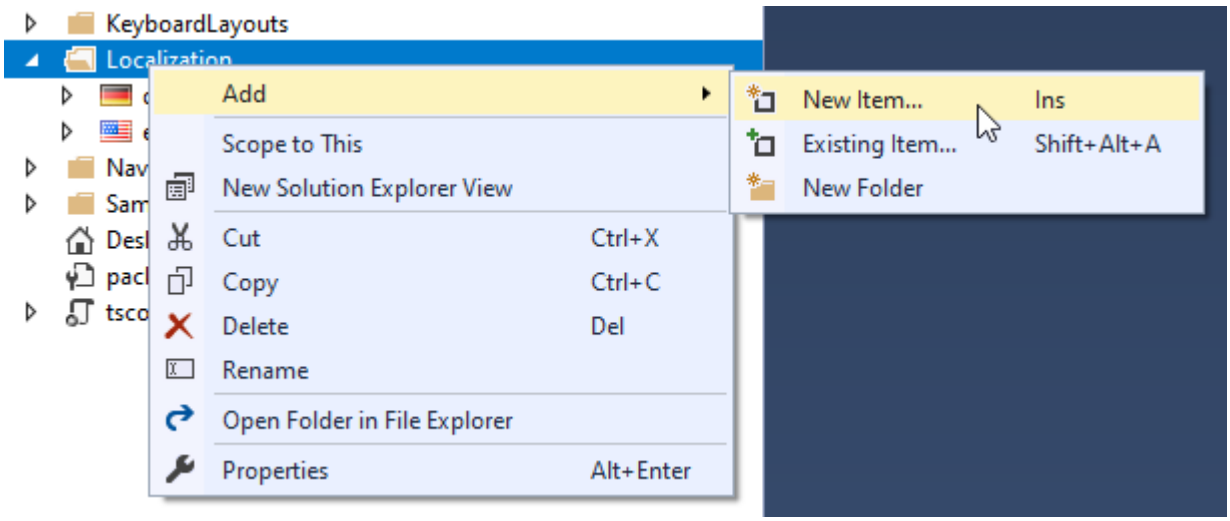


Localization of controls

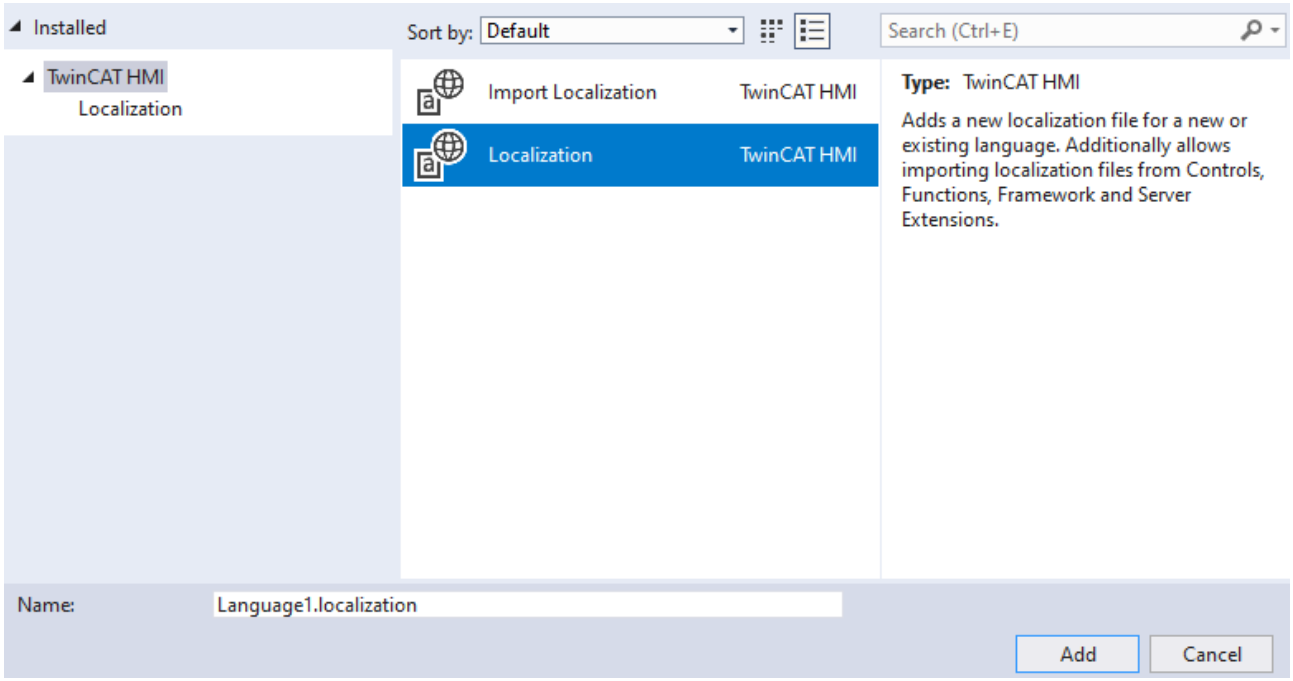
Localizations of controls from NuGet packages can be overridden and supplemented with additional languages.

Add localization

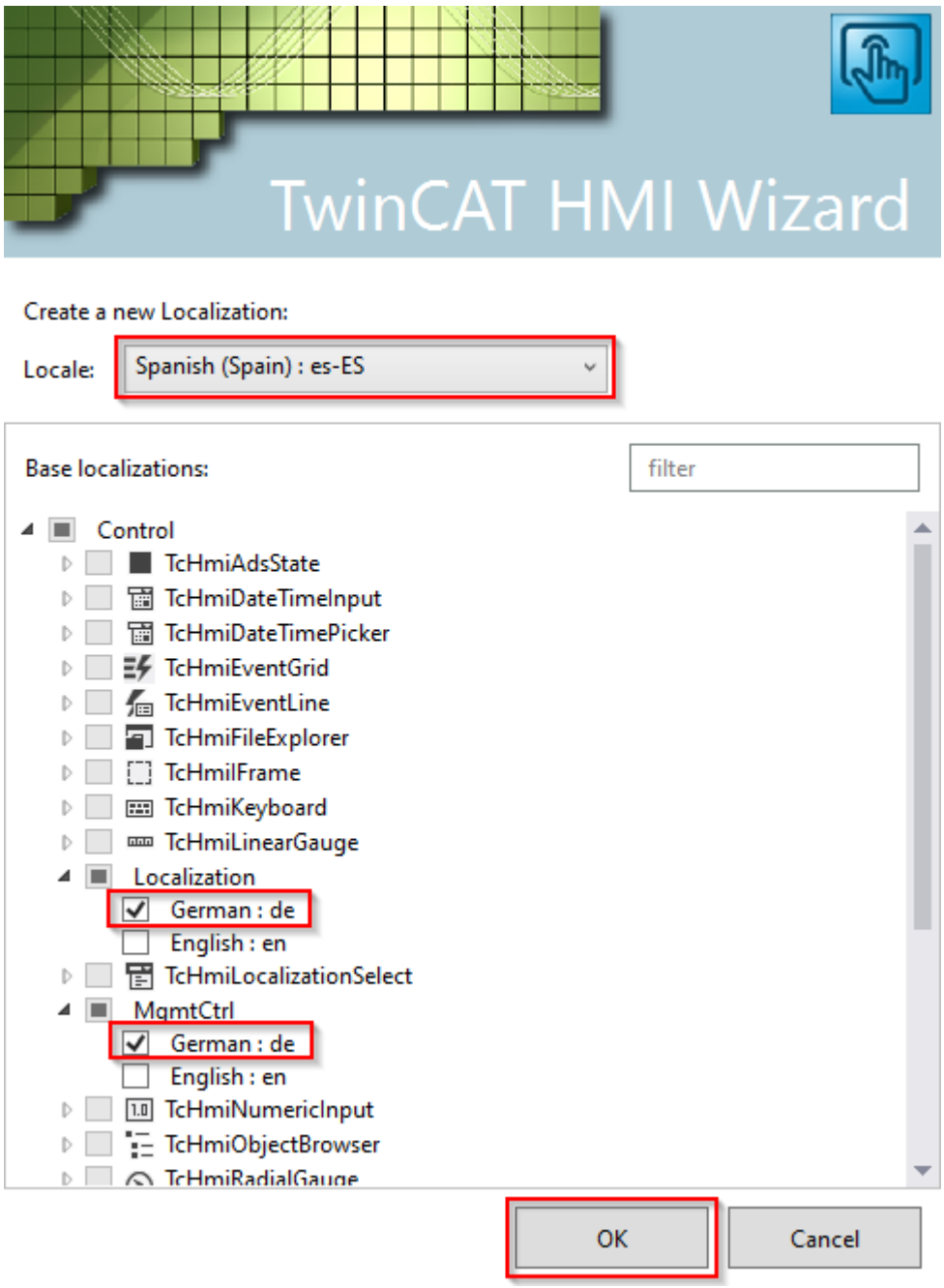
Each new language requires a new localization file in the localization folder.



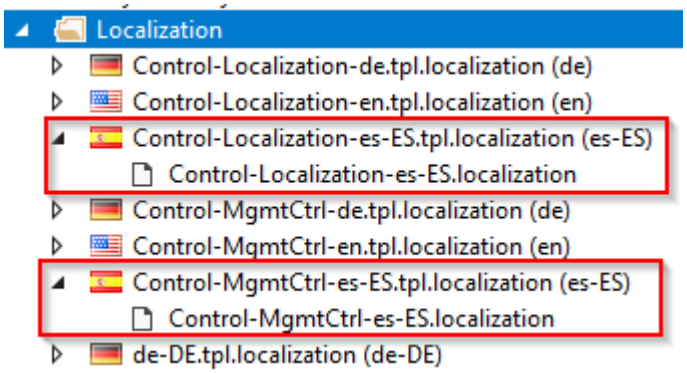
The name of the file is generated automatically and is therefore irrelevant.



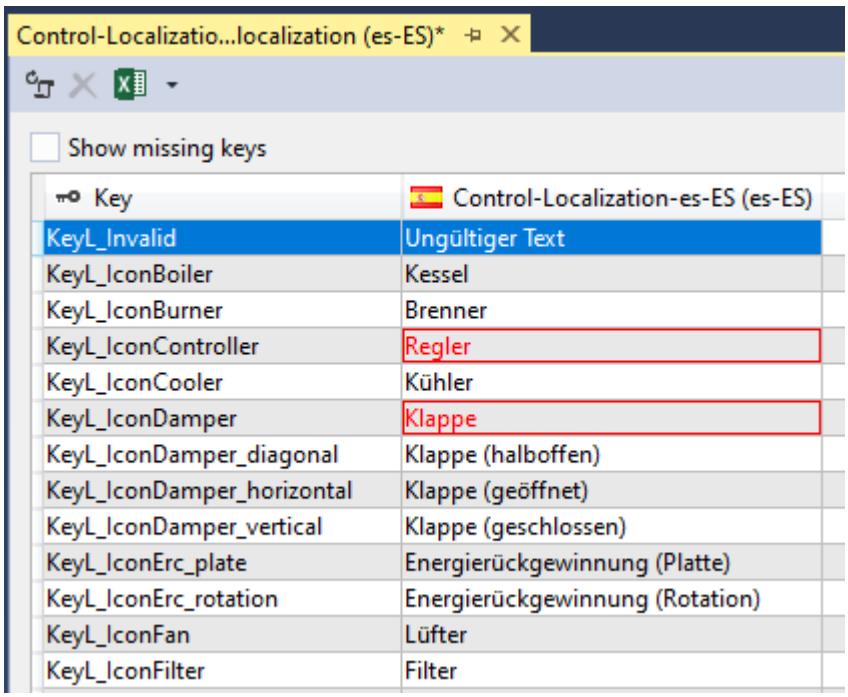
For each control, select any existing localization in order to transfer the language entries.



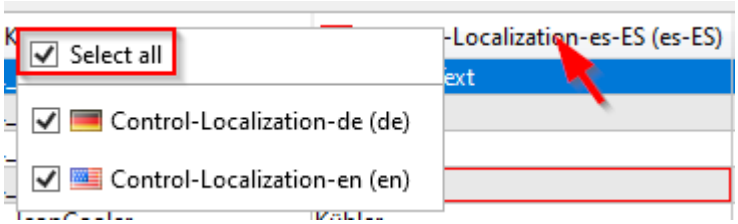
After confirming the selection, the localization files are available in the project.



Finally, the language entries must be replaced in the new localization.

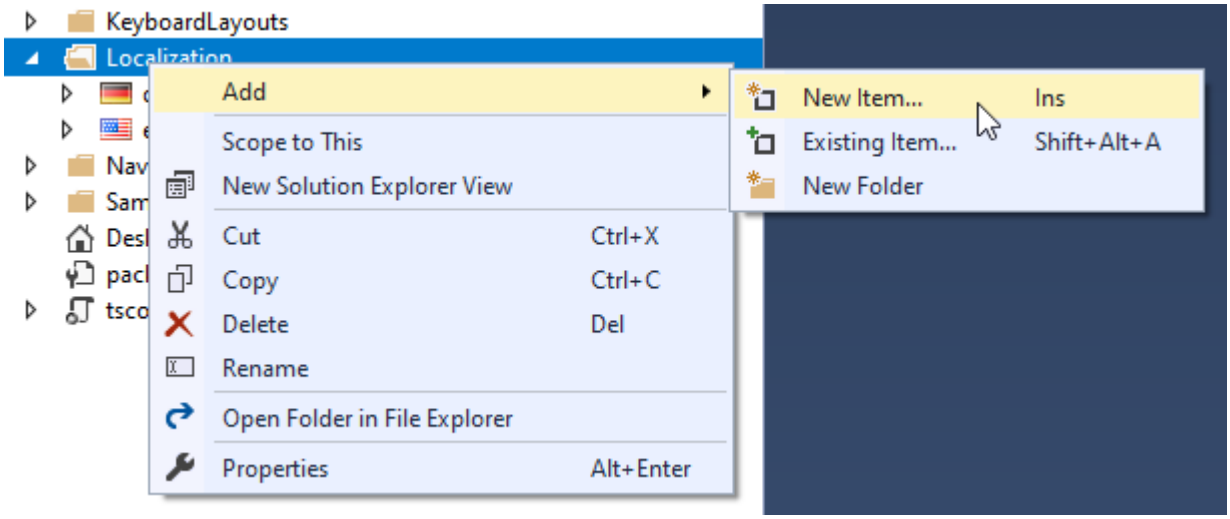


The available localizations can be shown and hidden via the right-click menu of a column.

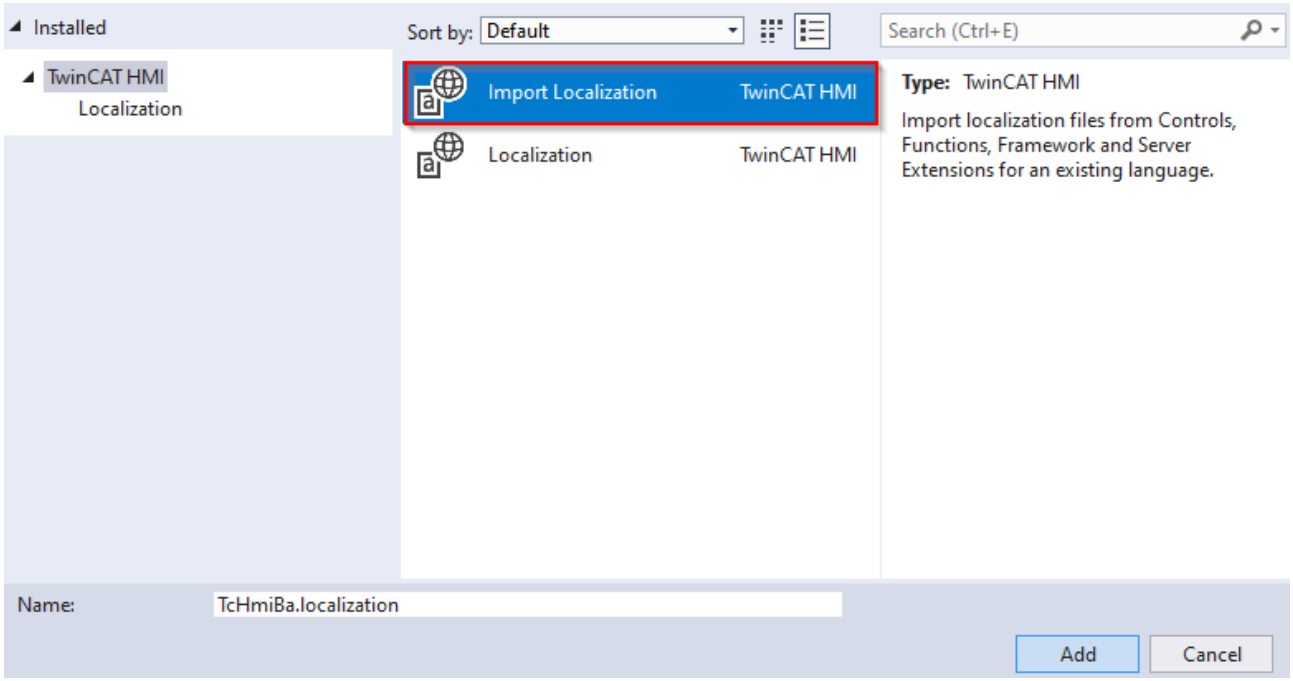


Override localization

The localization of the control must first be imported into the folder for localizations.



The name of the file is generated automatically and is therefore irrelevant.

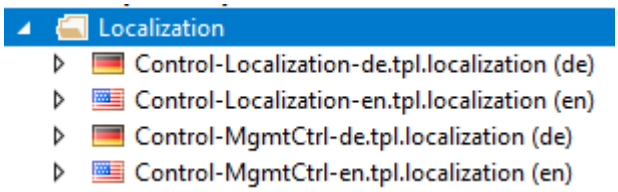


The localizations to be overridden must then be selected.

MgmtCtrl contains all localizations of the TcHmiBaFramework [▶ 1242] package and TcHmiBaControls [▶ 1122] package.

Localization contains all localizations from the TcHmiBaIcons [▶ 1231] package.

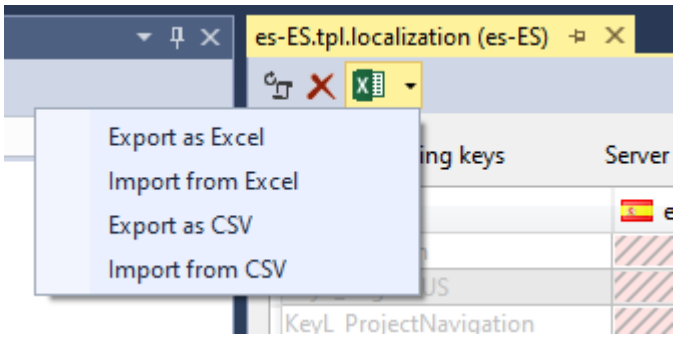
After confirming the selection, the localization files are available in the project.



Export/import localization

Excel

The localization file menu allows you to export localizations. The exported localizations can be edited in Excel and then re-imported.



6.2.1.3.2 BaObject handling

Almost all existing TcHmiBa controls [▶ 1122] can process a BaObject [▶ 60]. The handling of symbols in the BaSite-Extension is basically identical to that of the ADS-Extension.

Handling

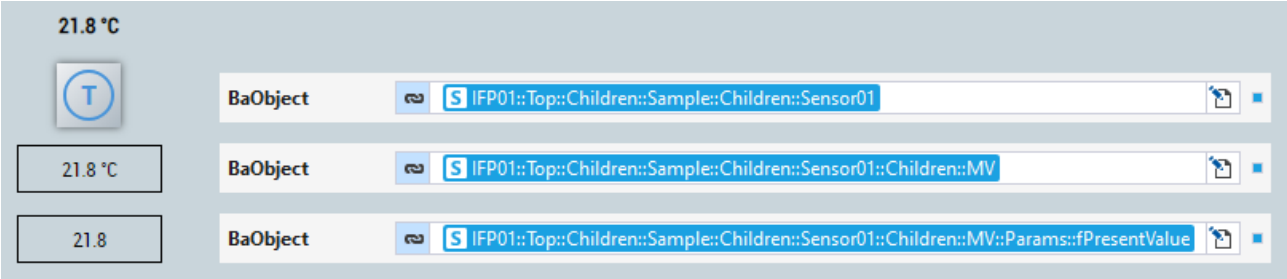
A simple example should give you an idea of the possible applications inside and outside a UserControl. For this purpose, the TcHmiBa controls *SensorAnalog* and *Textblock* are linked to a greatly reduced variant of an analog sensor in a TcHmi project.

The structure of the PLC is as follows:

Name	Datatype
ADS	object
BaSite	object
EventHistory	object
Events	object
IFP01	BA.IFP01
Top	BA.IFP01.ProjectStructure
Children	object
Sample	BA.IFP01.Tc3_XBA.FB_BA_View
Children	object
Sensor01	BA.IFP01.FB_BA_SensorAnalog
Children	object
MV	BA.IFP01.Tc3_XBA.FB_BA_AI_Raw_34321
Params	object
fPresentValue	BaVar.System.Single
Params	object
Params	object

Symbol mapping direct

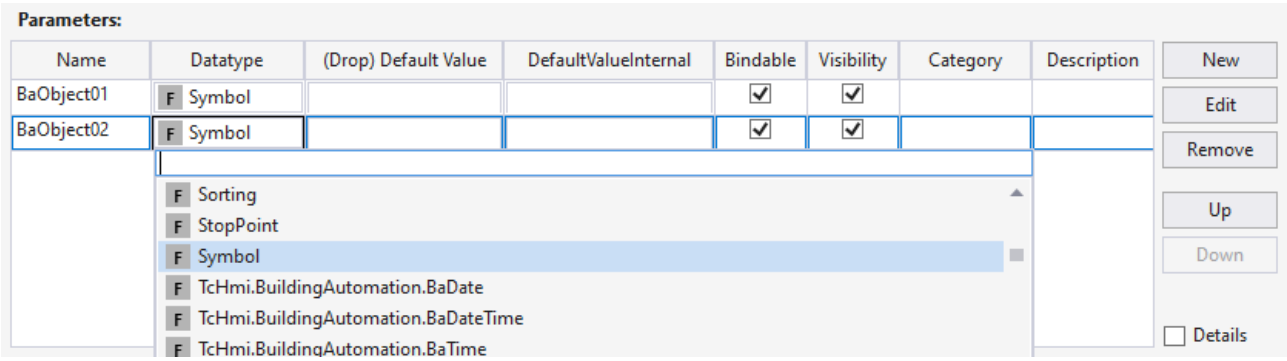
Linking the symbol from the BaSite-Extension directly to the attribute BaObject of a TcHmiBa control.



Symbol mapping via UserControl parameters

The UserControl in this example has the parameters BaObject01 and BaObject02 of type Symbol to pass through the symbols from the BaSite-Extension.

The *Symbol* data type is more performant for the TcHmiBa controls.



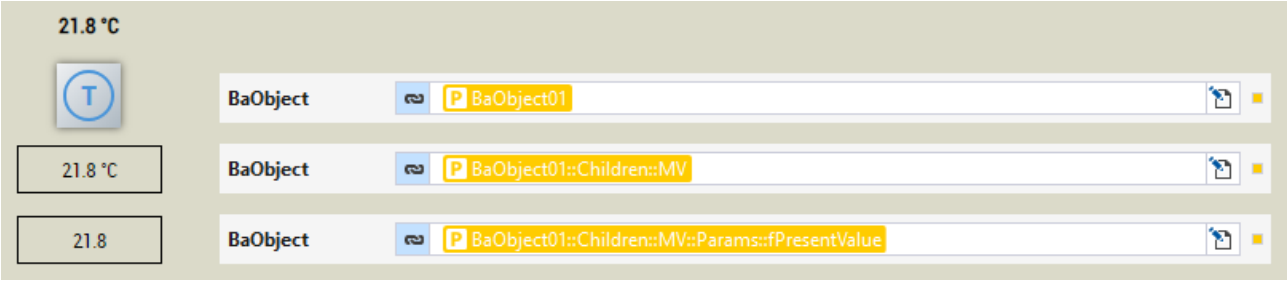
Due to the data type, it is no longer possible to navigate in the object when creating the binding.

However, deeper objects can still be reached by manually extending the symbol.

One parameter

Symbol for parameter BaObject01:

- `IFP01::Top::Children::Sample::Children::Sensor01`



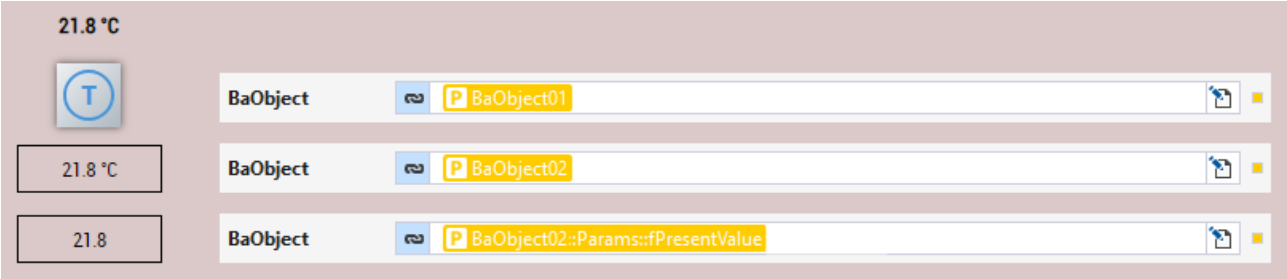
Several parameters

Symbol for parameter *BaObject01*:

- `IFP01::Top::Children::Sample::Children::Sensor01`

Symbol for parameter *BaObject02*:

- `IFP01::Top::Children::Sample::Children::Sensor01::Children::MV`



6.2.1.3.3 BaTemplate handling

Template handling

Explanation of the functionality using the example of the standard TF8040 PLC template *FB_BA_AC_ColT_02* for a cooler.

The matching TcHmiBa template *FB_BA_AC_ColT_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
Mdlt	FB_BA_AO_IO	Setpoint
MV	FB_BA_AI_IO	Feedback
TFl	FB_BA_SensorAnalog_IO	Flow temperature sensor
TRt	FB_BA_SensorAnalog_IO	Return temperature sensor
Vlv	FB_BA_Vlv	Valve

Hierarchy:

- BAObject
  - TFl
    - MV
  - TRt
    - MV
  - Vlv
    - Fdb
    - Mdlt

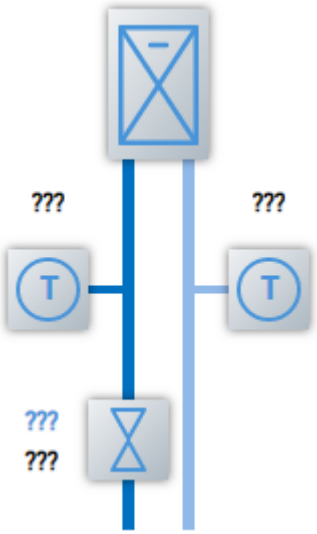
States

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

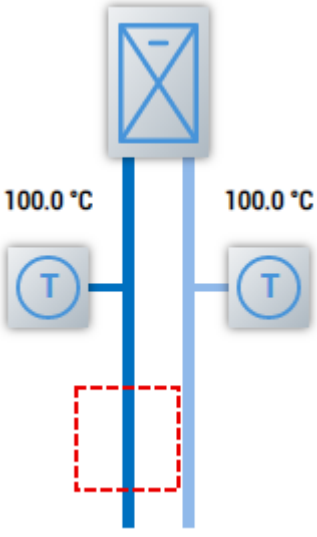
The cooler has been assigned a *BaObject* in which the valve (Vlv) is missing.

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

Hierarchy:

- BAObject
  - TFl
    - MV
  - TRt
    - MV

Presentation:



Symbol name renamed

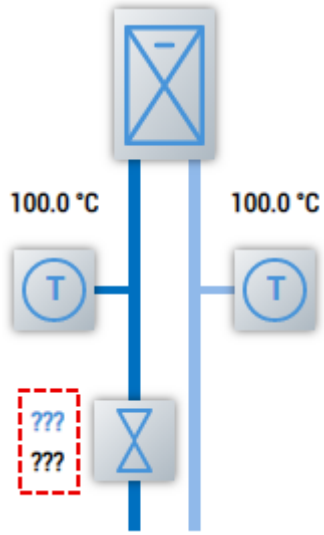
In this *BaObject* the top level subelements are correctly present, but the symbol name for Fdb from the Vlv has been renamed in the PLC template.

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

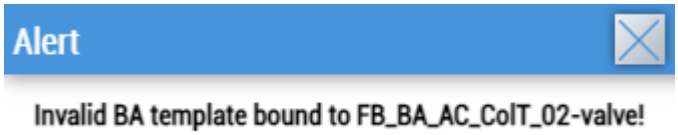
Hierarchy:

- BAObject
  - TFl
    - MV
  - TRt
    - MV
  - Vlv
    - Fdb_Test
    - Mdlt

Presentation:

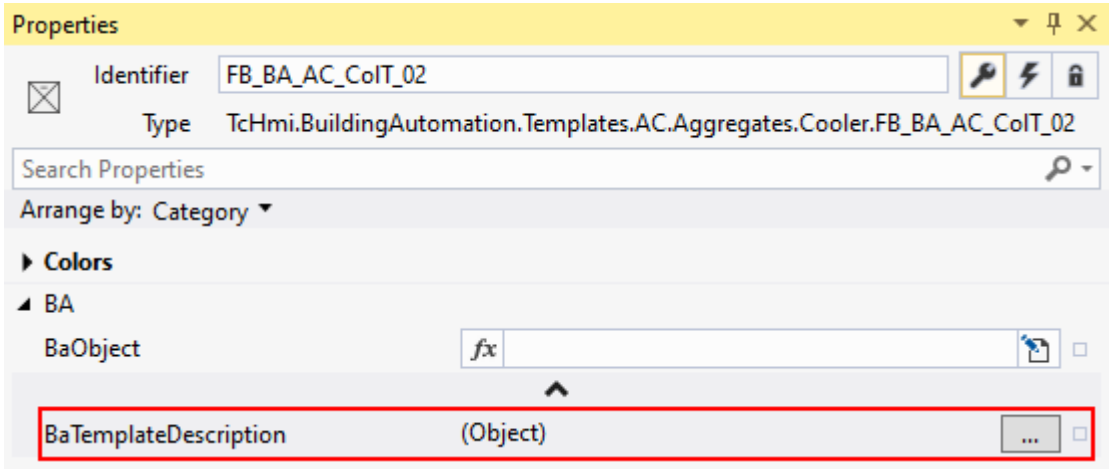


In addition, an error message indicates the subelement.



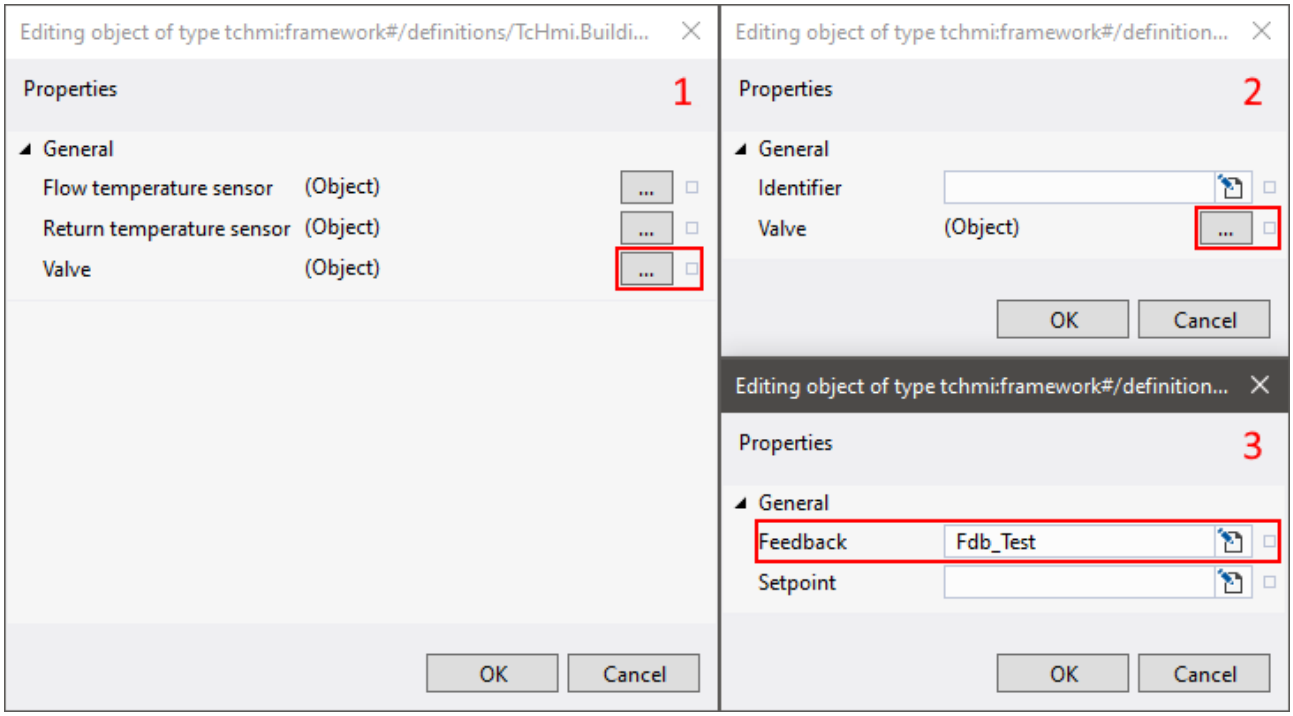
Customize symbol names

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



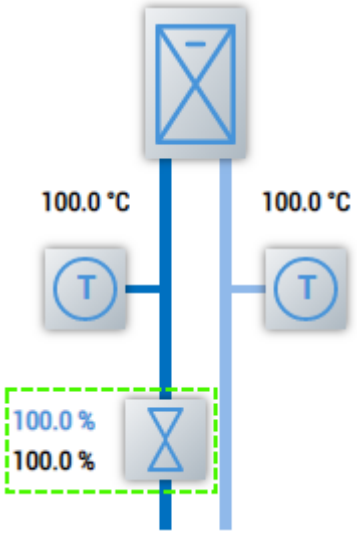
A dialog lists all subelements and allows navigation through the further levels [1]. The **identifier** refers to the symbol name of the current level [2]. If this specification is no longer possible, then the last level of the subelement is reached [3].

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.

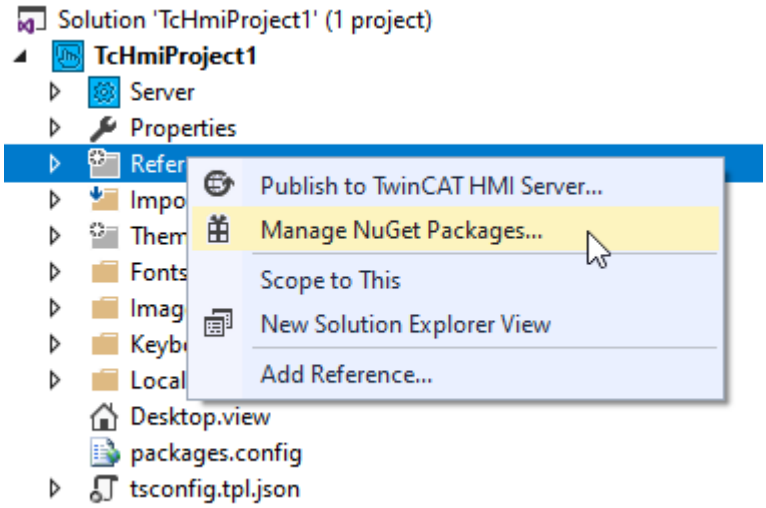
```
MyView::Motor::Cmd
```

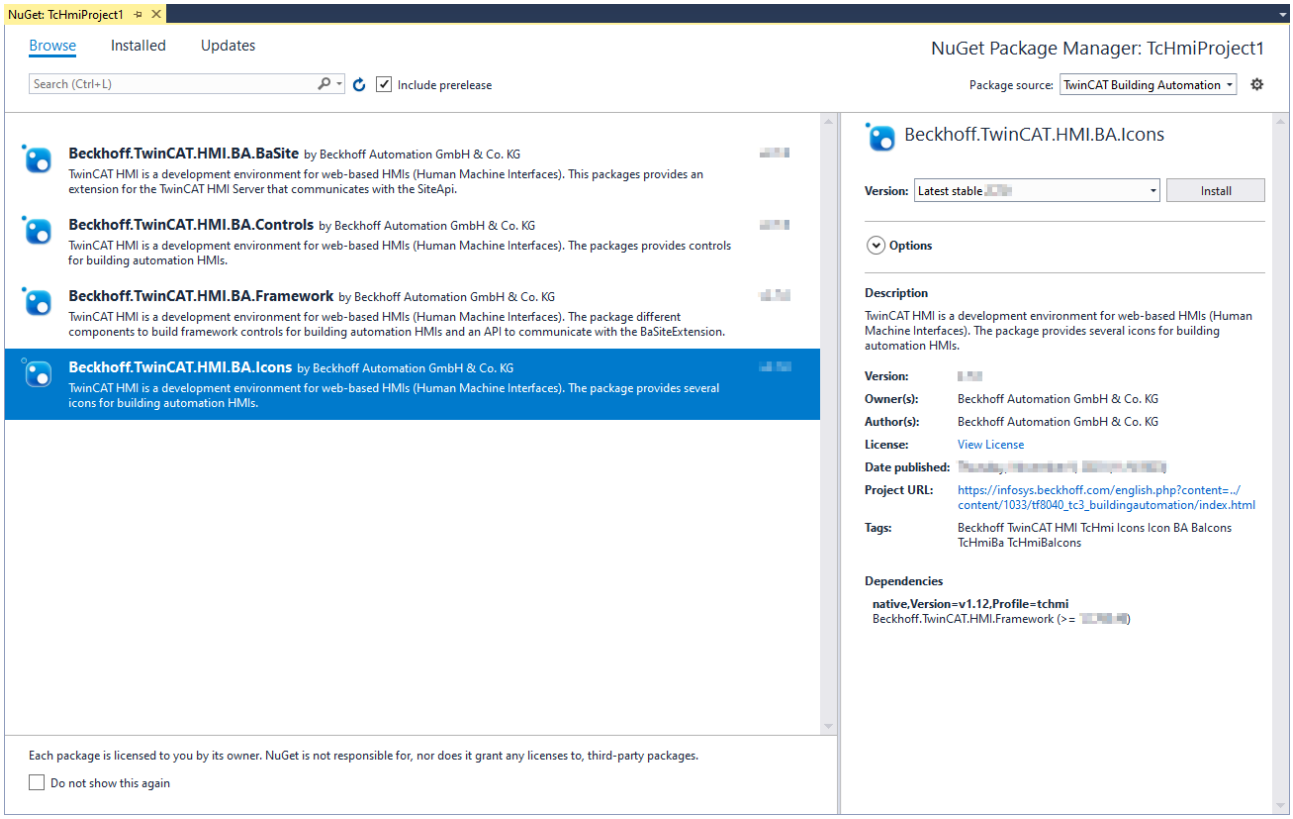
6.2.1.4 Icons

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.





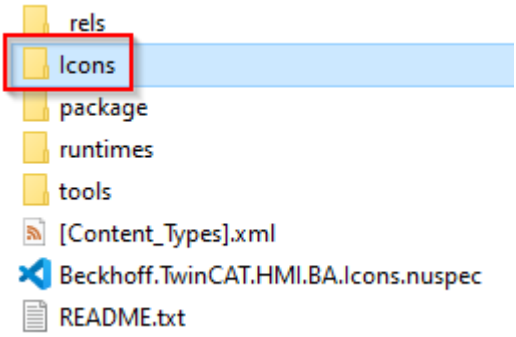
Since there is a dependency on the Beckhoff.TwinCAT.HMI.Framework, this is also installed.

Use

There are three different application options.

ZIP archive

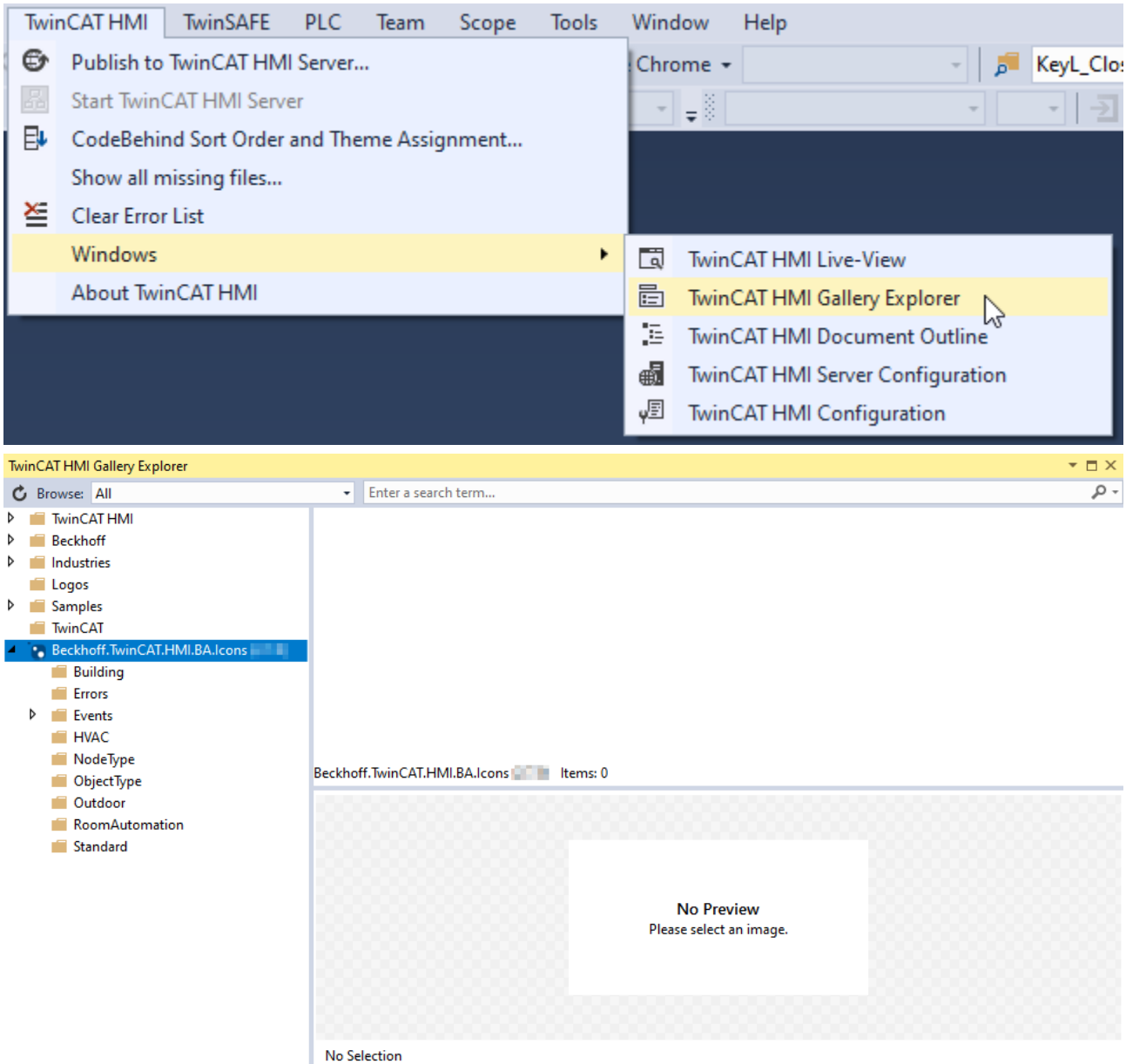
A NuGet package is basically a ZIP archive that allows direct use of the icons after unpacking. The unpacked content looks like this:



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

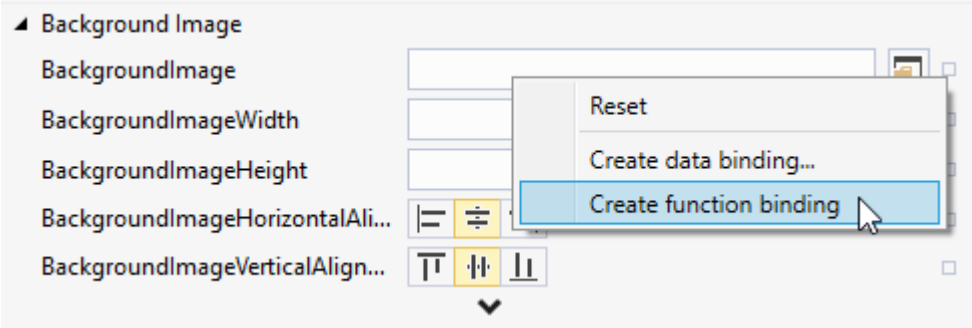
With the function

```
TcHmi.BuildingAutomation.Functions.GetBaIconPath()
```

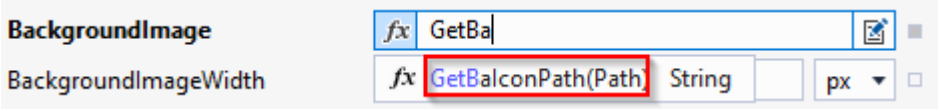
the icons can be used as a reference in a TcHmi project.

The following steps are required:

1. Drag a control (e.g. a button) to the content/view.
2. Create the **Function Binding**.



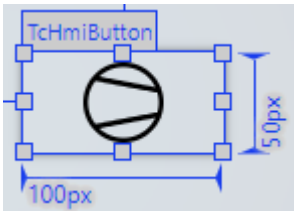
3. In the field **Backgroundimage** enter **GetBaIconPath** (function is suggested).



4. Path specification to the icon, e.g. *"HVAC/fan"* (quotation marks must be observed).
5. Adjust size and position.

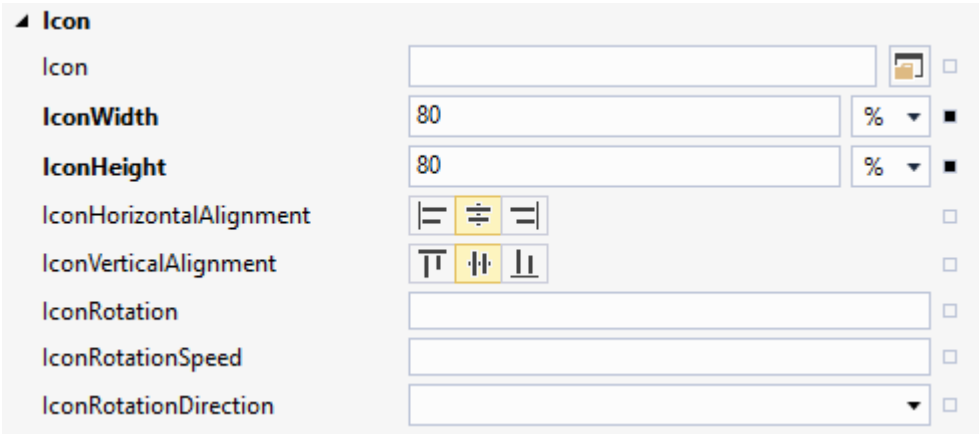
The path information can be taken from the structure of the *GalleryExplorer*.

⇨ 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 | Common** is used below.



The Icon attribute can be set again using the

```
TcHmi.BuildingAutomation.Functions.GetBaIconPath()
```

function.

This form of embedding allows the icon to change 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. The namespace

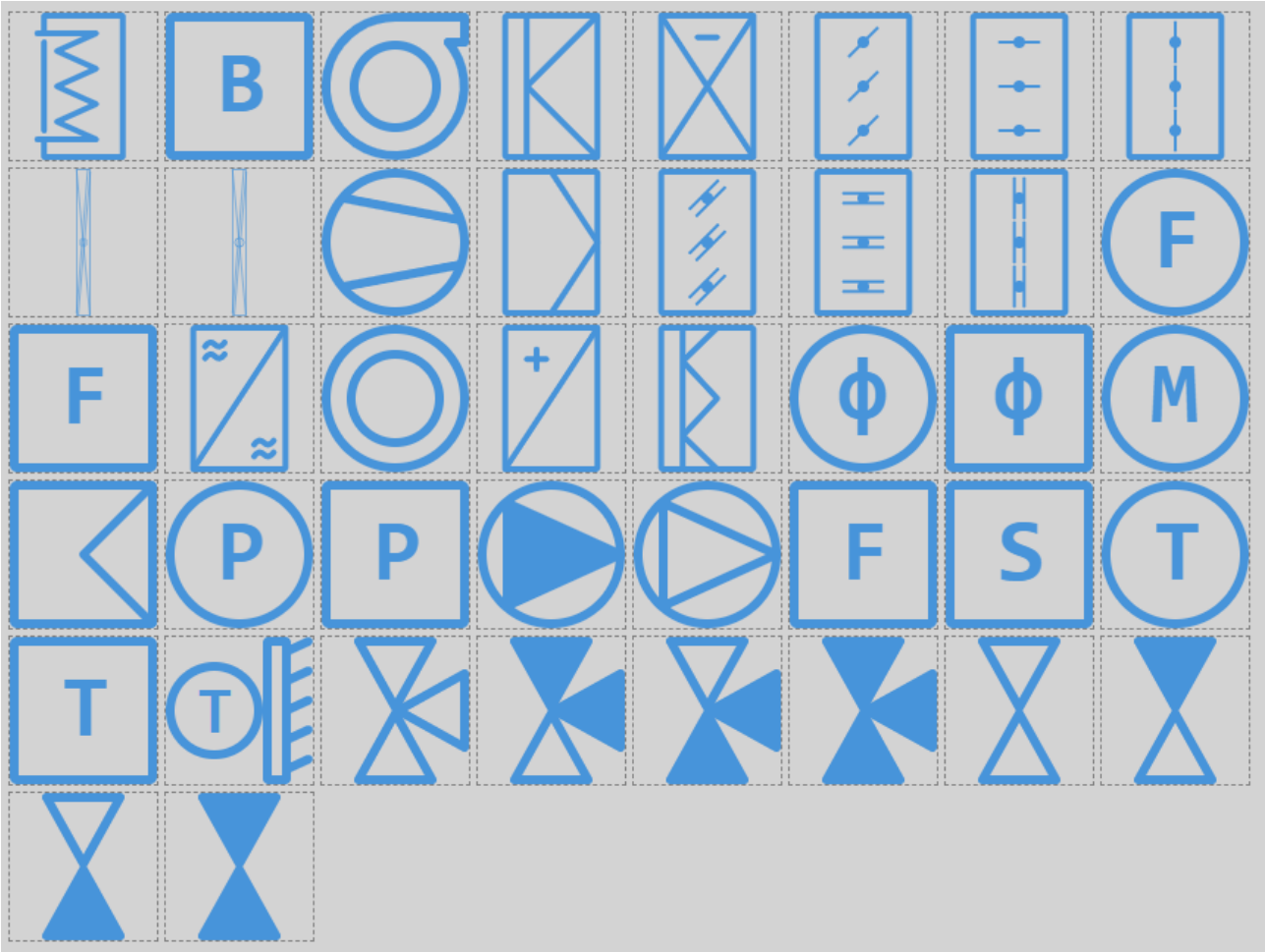
```
TcHmi.BuildingAutomation.Icons
```

contains constants that point to the respective icons in the NuGet package (e.g. *TcHmi.BuildingAutomation.Icons.HVAC.Fan.path*).

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 [▶ 32] uses the following icons.

Icon	Name
	Alarm
	Fault
	Maintenance
	Notification
	Miscellaneous

The events [▶ 32] 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

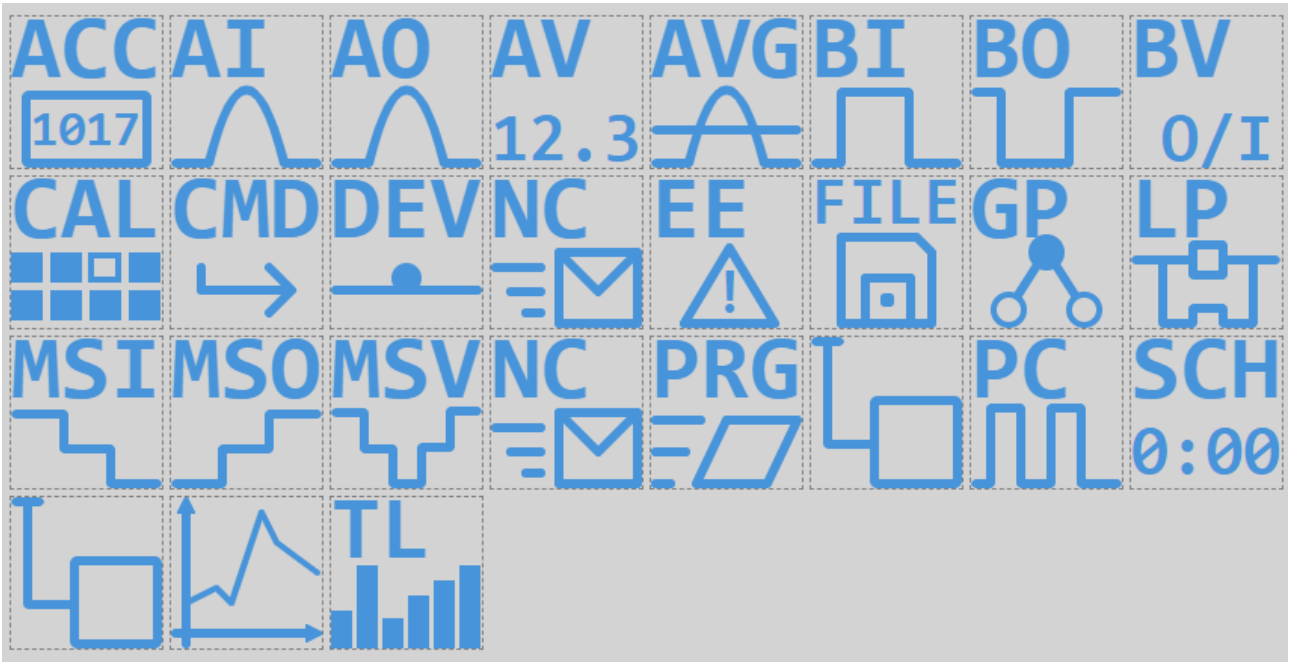
NodeType

Icon	Name
	Aggregate
	Buildings
	Building element
	Component
	Control cabinet
	Floor
	Information focus
	Property
	Plant

Icon	Name
	Room
	Technical system

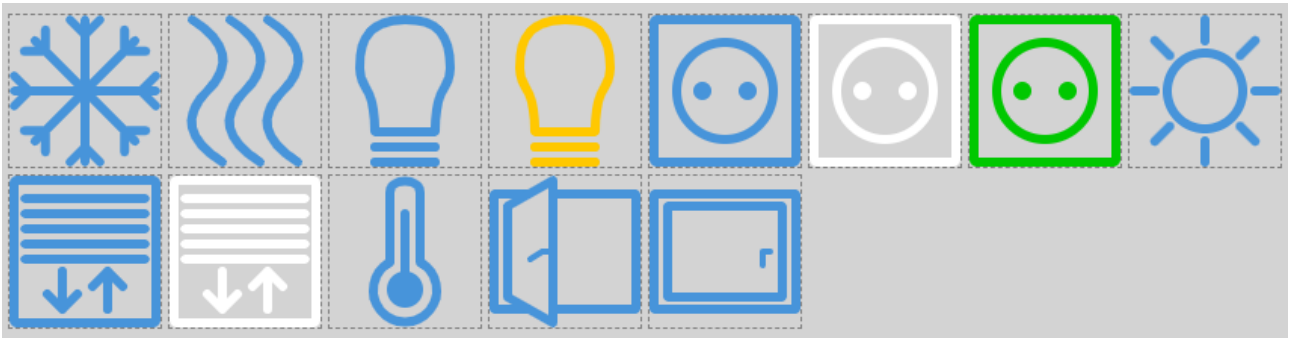
ObjectType

Symbols for the different object types.



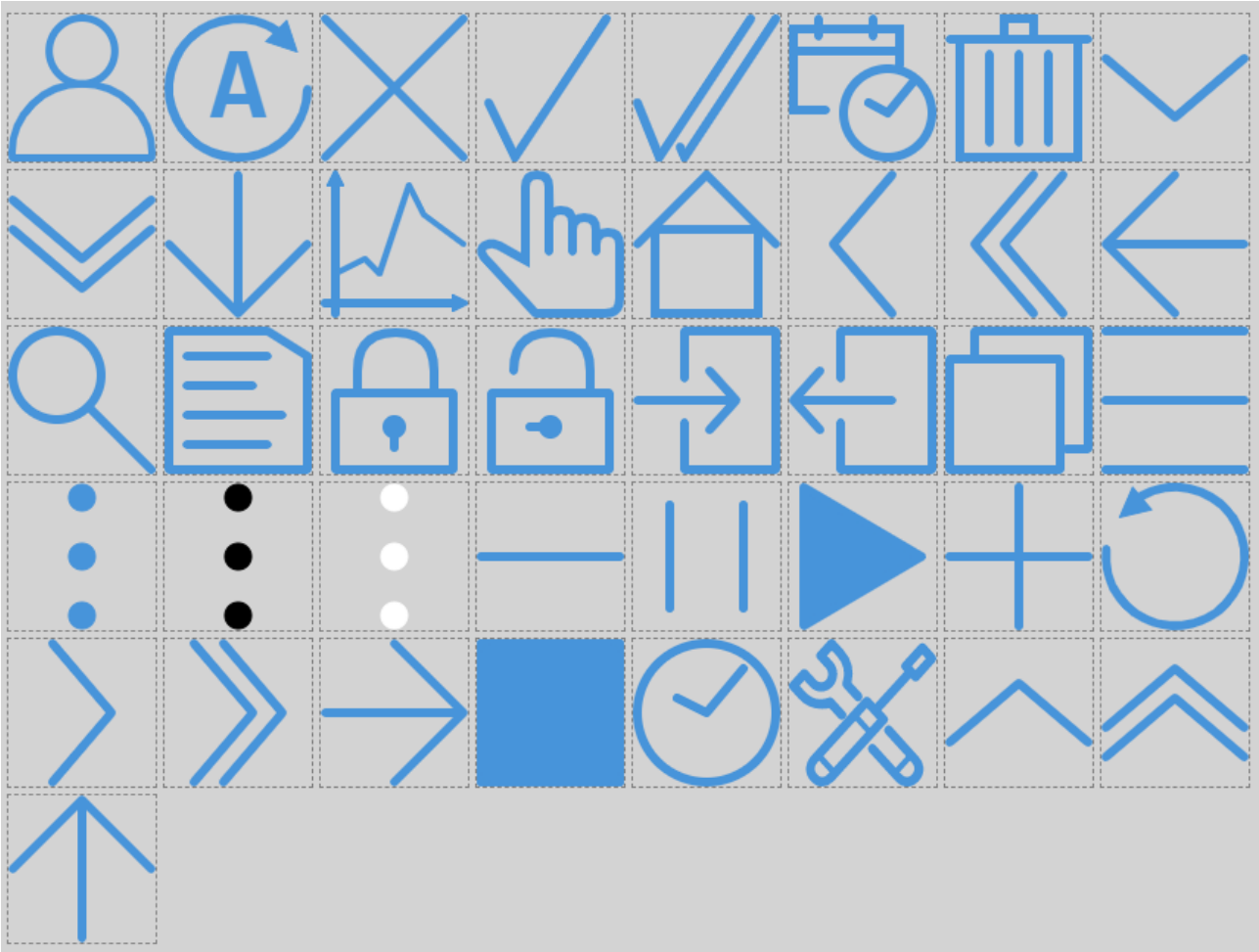
Room Automation

Symbols for room automation.



Standard

Standard symbols for visualizations.



6.2.1.5 Framework

The framework contains various classes, enumerations, interfaces and types that are used to create the controls [▶ 1122]. 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.

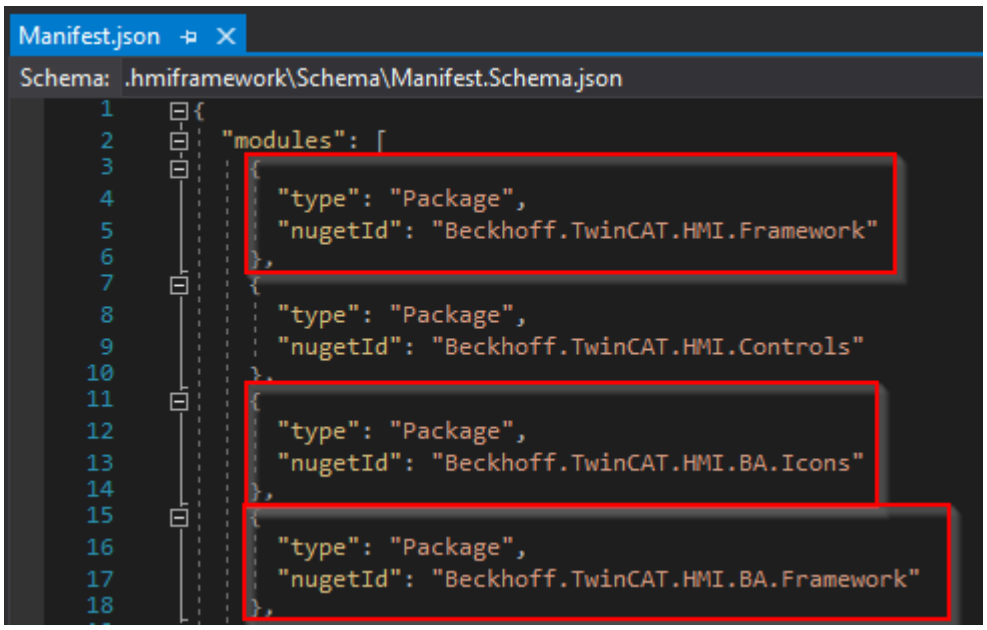
If the complete solution TF8040 [▶ 8] is used, the framework also takes over many management functions to realize the Generic [▶ 54].

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 [▶ 1231]
- 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"
]
```

6.2.1.5.1 BA

6.2.1.5.1.1 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 *BaObject* (as BaView).
loadTexts	Determines whether all texts are loaded when setting the *BaObject* (as BaView).
enableParentBaObjectProcessor	Determines whether the *BaObject* processor of the parent control is called.
baObject	Returns the *BaObject*.
baObjectSymbolExpression	Returns the *SymbolExpression* from the *BaObject*.
isLoadingBaObject	Checks if the *BaObject* is set but still loading.

Methods

Name	Description
setBaObject	Sets the *BaObject*.
resolveBaObject	Resolves the passed information to create a *BaObject*.
readBaObject	Reads the BaObject from the server.
watchBaVariable	Adds a watch to the passed BaVariable. The Watch is also destroyed when the class is destroyed.
tryWatchBaVariable	Attempts to monitor a BaVariable.
tryWatchChildrenBaVariable	Attempts to monitor a BaVariable of a child element of a BaView.
watchValueRange	Monitors the value range of a specific variable.
checkBaObjectAccess	Checks the OperationType and the write access of the BaVariable *ValueRm*.

Events

Name	Description
onBaObjectChanged	Triggered when the *BaObject* has changed.

6.2.1.5.2 Helper

6.2.1.5.2.1 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>.

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.

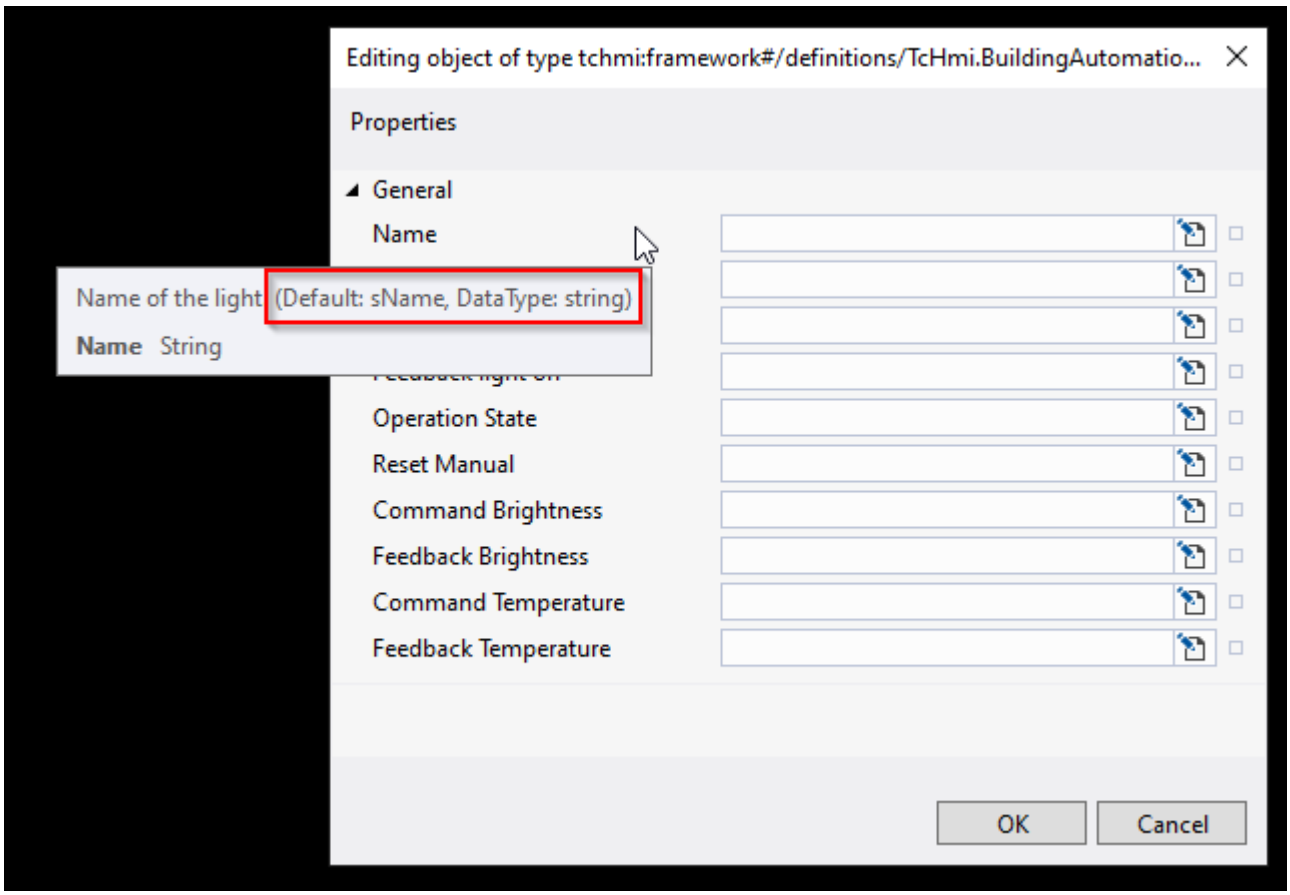
```
export const BaInterfaceDef: BaInterfaceDefinition<BaInterface> = {
    state: {
        type: 'boolean'
    },
    stateFeedback: {
        type: 'boolean',
        optional: true
    },
    activeText: {
        type: 'string',
        optional: true
    },
    inactiveText: {
        type: 'string',
        optional: true
    }
};
```

It is possible to specify multiple data types.

```
command: {
    type: ['number', 'boolean']
}
```

BaInterfaceSymbolNames

The default values of BaInterfaceSymbolNames, as well as the expected data types can be found in the tooltip of the dialog for setting 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.

Attribute BaInterfaceSymbolNames

Since the setters and getters for the BaInterfaceSymbolNames are already defined in the interface IUsesBaInterface<T>, the following attribute can be defined in the Description.json so that the symbol names can be changed in the designer:

```
{
  "name": "data-tchmi-ba-interface-symbol-names",
  "displayName": "BaInterfaceSymbolNames",
  "propertyName": "BaInterfaceSymbolNames",
  "propertySetterName": "setBaInterfaceSymbolNames",
```

```
  "propertyGetterName": "getBaInterfaceSymbolNames",
  "visible": true,
  "themeable": "None",
  "displayPriority": 61,
  "type": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Common.Checkbox.BaInterfaceSymbolNames",
  "category": "BaData",
  "description": "Symbol names for the interface symbol.",
  "requiredOnCompile": false,
  "readOnly": false,
  "bindable": true,
  "heritable": true,
  "defaultValue": null,
  "defaultValueInternal": null
}
```

The

```
type
```

is to be defined in the Types.Schema.json.

Overwrite in onInitialized

If the symbol names for all controls are to be overwritten, it is not practical to edit the BaInterfaceSymbolNames attribute of each control separately.

The default symbol names can be globally overwritten in the onInitialized event. For this purpose, a CodeBehind function is created with the following content:

```
module TcHmi {
    var destroyOnInitialized = TcHmi.EventProvider.register('onInitialized', function (e, data) {
        e.destroy();

        // overwrite BaInterfaceSymbolNames of the Checkbox
        BuildingAutomation.Controls.Checkbox.BaInterfaceSymbolNames = {
            state: {
                symbolName: 'bState'
            },
            stateFeedback: {
                symbolName: 'bStatFdb'
            },
            activeText: {
                 symbolName: 'sActiveText'
            },
            inactiveText: {
                symbolName: 'sInactiveText'
            }
        }
    });
}
```

It is not necessary to set the symbol names of all elements. Only the elements that have not been marked as optional are to be set.

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
hasSubSymbol	Checks if the BaInterface has a specific subelement. This method must be used if optional elements are to be read or written.
writeSubSymbol	Writes the value of a subelement.
watchSubSymbol	Creates a subscription for a subelement.
updateSymbolNames	Updates the symbol names of the BaInterface subelements.
convertToBaInterfaceSymbolNames	Converts the data type `BaInterfaceSymbolNamesDesigner` to `BaInterfaceSymbolNames`. If the data type `BaInterfaceSymbolNames` is already passed to the method, no conversion is performed and the object is returned directly.

6.2.1.5.3 BusyHandler

The BusyHandler is a class of TcHmiBaFramework [▶ 1242]. 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;
}
```

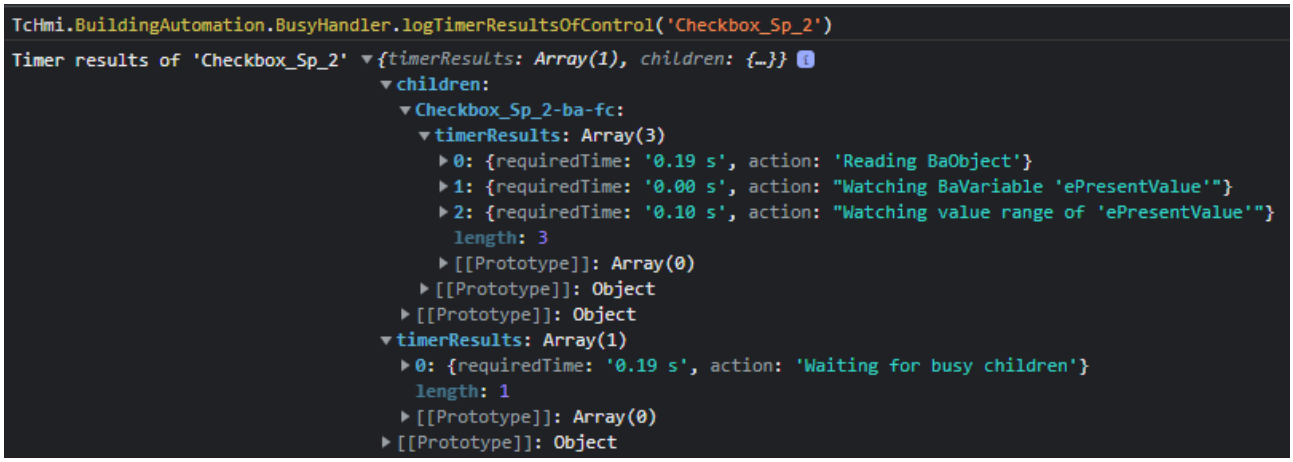
Use

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:



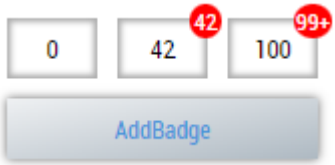
It is recognizable what the control was busy with.

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.

6.2.1.6 Functions

6.2.1.6.1 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.

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.

6.2.1.6.2 ConvertHexToRgbaColor

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 be converted.

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA color.

6.2.1.6.3 ConvertHslToRgbaColor

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.

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA color.

6.2.1.6.4 ConvertRgbaToHexColor

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.

Return value

```
tchmi:general#/definitions/String
```

Hex color.

6.2.1.6.5 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.

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.HSLColor
```

HSL color.

6.2.1.6.6 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.

Return value

```
tchmi:framework#/definitions/SolidColor
```

TcHmi Solid color.

6.2.1.6.7 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.

Return value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA color.

6.2.1.6.8 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 icon.

Return value

```
tchmi:general#/definitions/String
```

Unit.

6.2.1.6.9 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 icon.

Return value

```
tchmi:general#/definitions/String
```

Full path from the icon.

6.2.1.6.10 GetCurrentUserName

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.

6.2.1.6.11 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.

StartValue

```
tchmi:framework#/definitions/Number
```

Start value of the cross-fade.

EndColor

```
tchmi:framework#/definitions/SolidColor
```

End color of the cross-fade.

EndValue

```
tchmi:framework#/definitions/SolidColor
```

End value of the cross-fade.

Value

```
tchmi:framework#/definitions/Number
```

Value used to calculate the color.

Return value

```
tchmi:general#/definitions/SolidColor
```

Calculated color.

6.2.1.6.12 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.

UserContents

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.UserContents
```

Collection of users with their associated content.

StoreLastContent

```
tchmi:general#/definitions/Boolean
```

Determines whether the last content will be reloaded on the user's next visit.

6.2.1.6.13 OpenBaTrend

Opens a dialog in which the trend from a linked BaObject or the trends from a BaView are displayed.

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/ContentPath
```

The BaObject / BaView whose trends are displayed.

6.2.1.6.14 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.

Modal

```
tchmi:general#/definitions/Boolean
```

Determines whether the dialog is opened modally or not.

Scrolling

```
tchmi:framework#/definitions/ScrollMode
```

Determines whether the content can be scrolled.

Headline

```
tchmi:general#/definitions/String
```

Title of the dialog.

Width

```
tchmi:general#/definitions/Number
```

Width of the dialog.

WidthUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit of the width of the dialog.

Height

```
tchmi:general#/definitions/Number
```

Height of the dialog.

HeightUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Unit of the height of the dialog.

6.2.1.6.15 OpenLegendDialog

Opens an instance of the Legend [▶ 1124] 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.

Buttons

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.DialogWindowButtons
```

Buttons for closing or confirming the dialog.

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.

IconDataSource

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataSource
```

Selection of entries to be displayed.

IconDataCustom

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList
```

List with additional entries.

6.2.1.6.16 OpenLightZoneDialog

Opens an instance of the LightZone 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.

6.2.1.6.17 OpenTrendCollectionView

Opens a dialog for Monitoring trend collections [▶ 57].

Use

The function is called event-driven and can be found in the Actions and Conditions editor under *Functions* > *BuildingAutomation*.

6.2.1.6.18 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*.

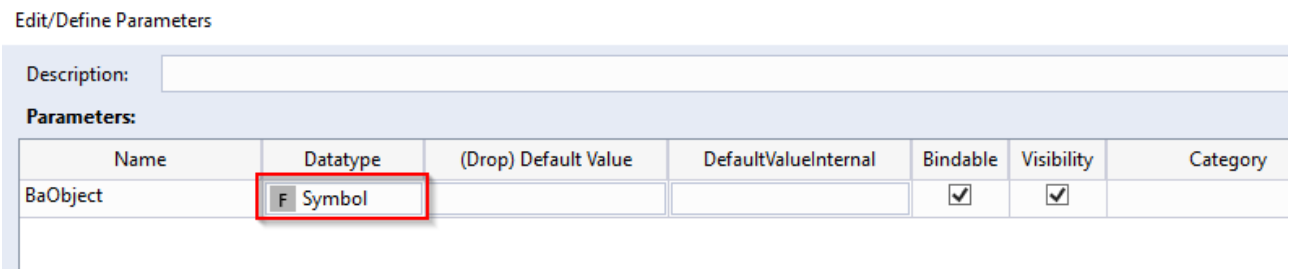
6.2.1.6.19 UseBaObjectsInUserControl

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.

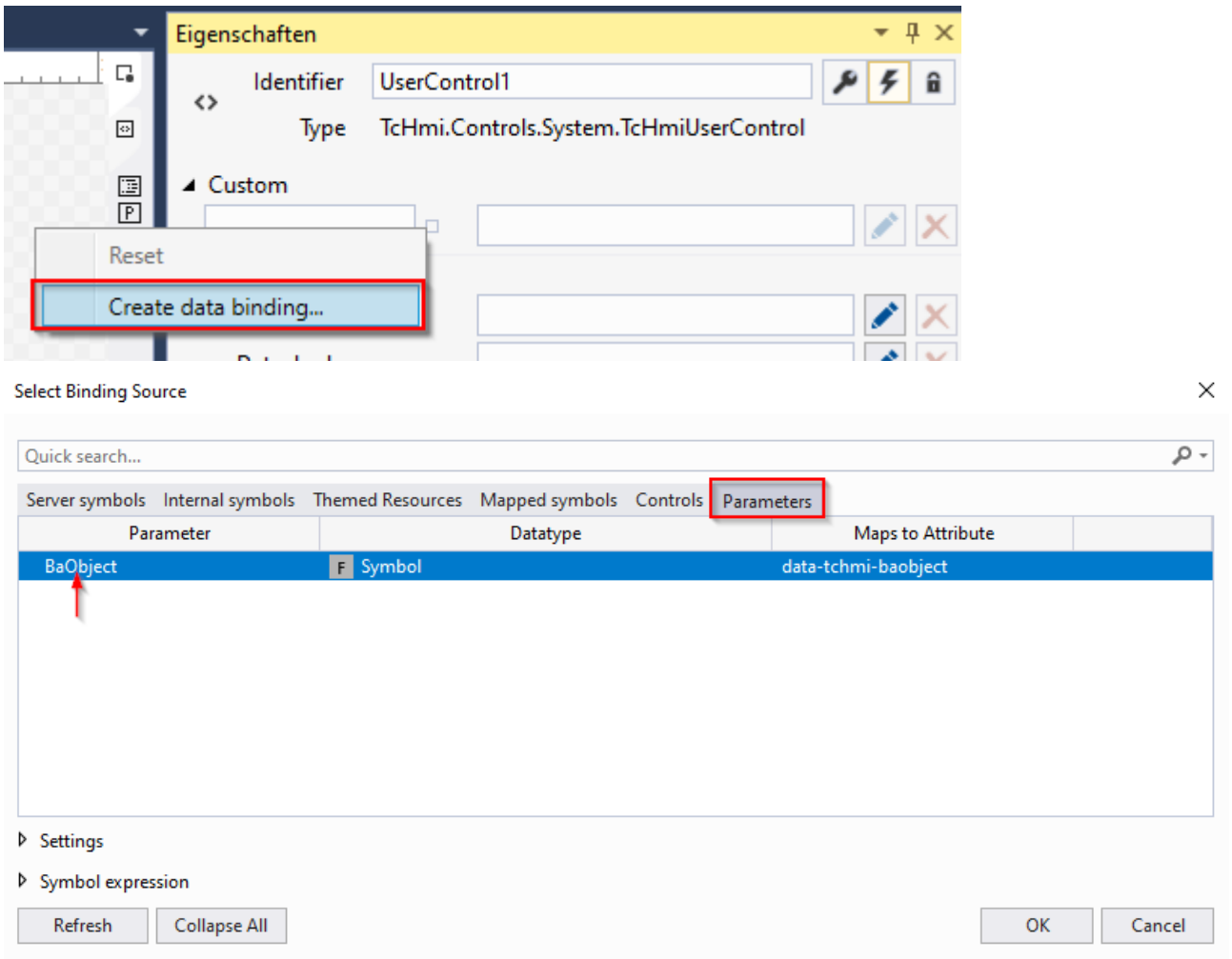
Use

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*!

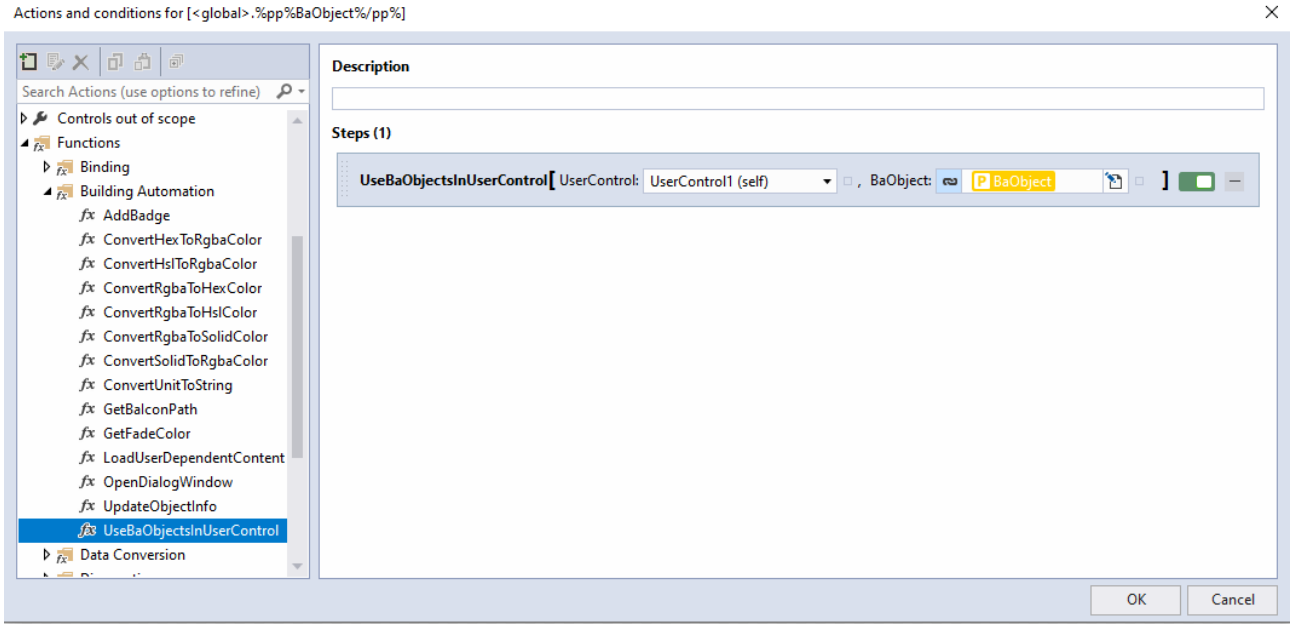


In this way, access to the underlying symbols of the BaObject is no longer possible.

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"*.



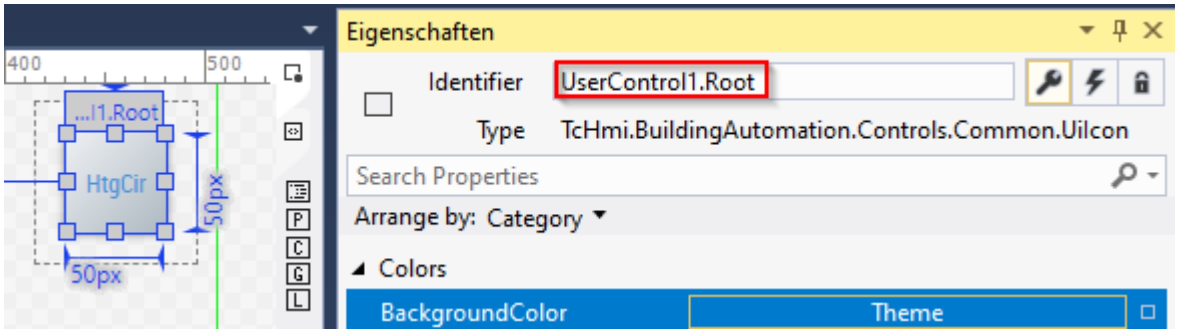
The controls in the UserControl are then linked to the BaObject or its subelements via the identifiers of the controls.

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*.

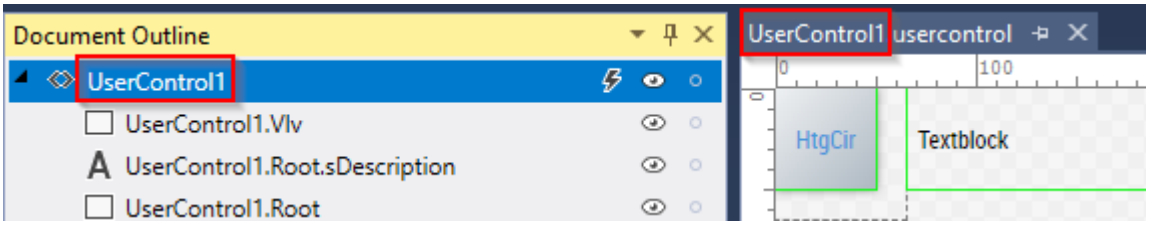
Linking

BaObject

To use the BaObject directly, the control must have the identifier *$UserControlName$.Root*.
$UserControlName$ is replaced with the name of the UserControl.



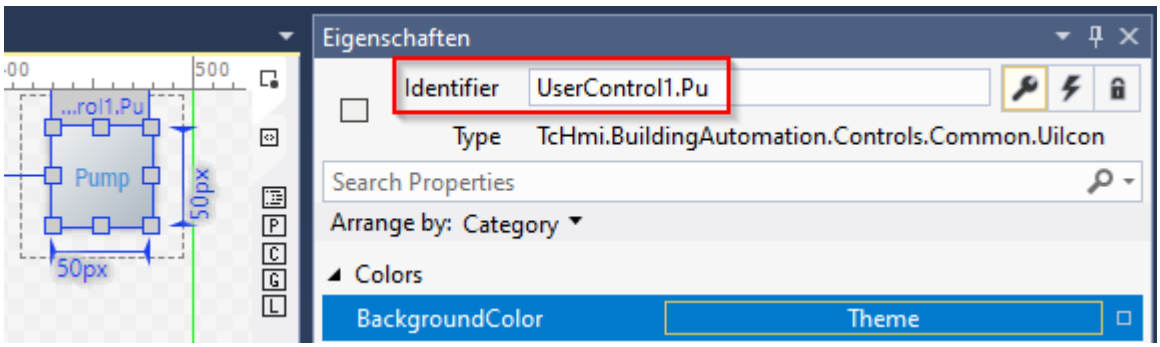
Make sure that the root element of the UserControl has the same name as the file of the UserControl.



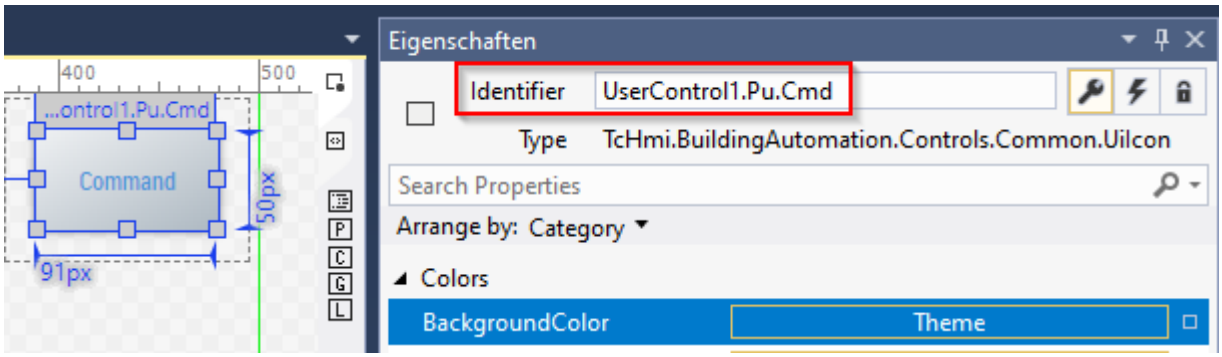
Subelements of BaObject

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 [▶ 1135], ComboBox [▶ 1139] or InputBox [▶ 1144], 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.

6.2.1.7 Server extensions

6.2.1.7.1 BaSiteExtension

Description

The BaSite-Extension serves as an interface between a TF8040 PLC (TF8040 Getting Started [▶ 64]) and a TcHmi client. The extension makes it possible to offer generic functions that significantly simplify and accelerate the engineering of the HMI.

Links between the PLC and the HMI server are still possible via the ADS extension, even without the extension. In this case the benefits referred to above no longer apply.

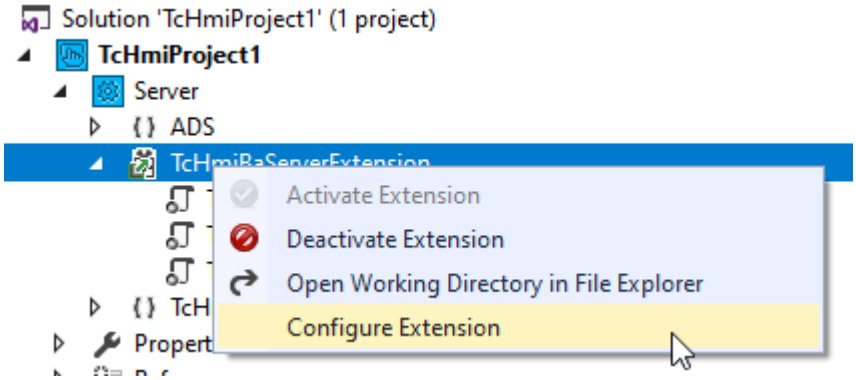
Use

To be able to use the generic functionalities in a project, the extension [▶ 77] must be installed.

Configuration

The extension can be configured in TcHmi engineering or via the configuration page of the HMI server.

Configuration from TcHmi engineering:



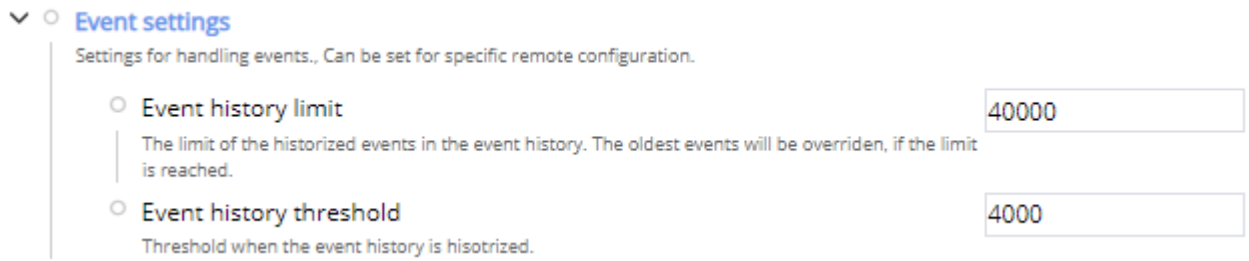
General

Interval

This time determines how often variables are read (updated) in the interface.

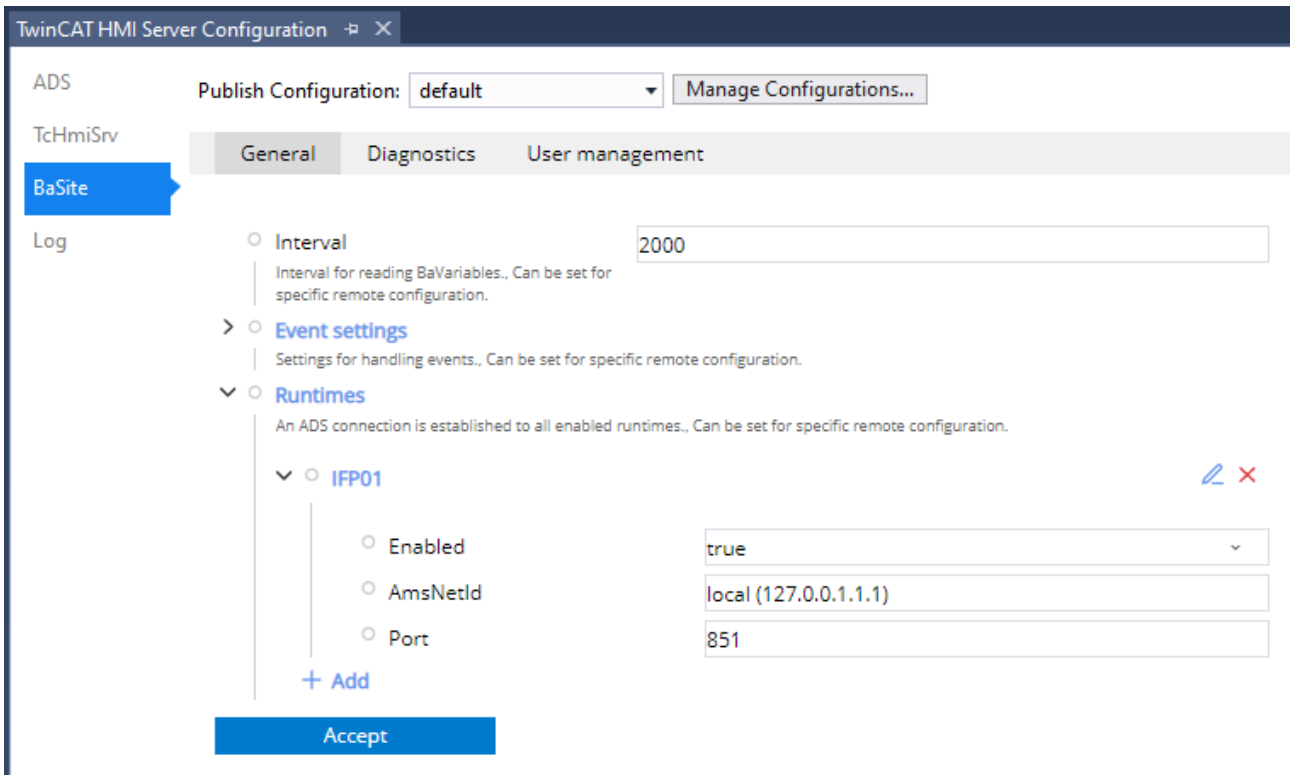
Event setting

These settings influence the behavior of the event history. You can set how many events are saved in the event history and from which number the events are historized.



Devices

The extension must be notified of the devices to which the HMI is to connect. The devices can be added or removed via the configuration mask.



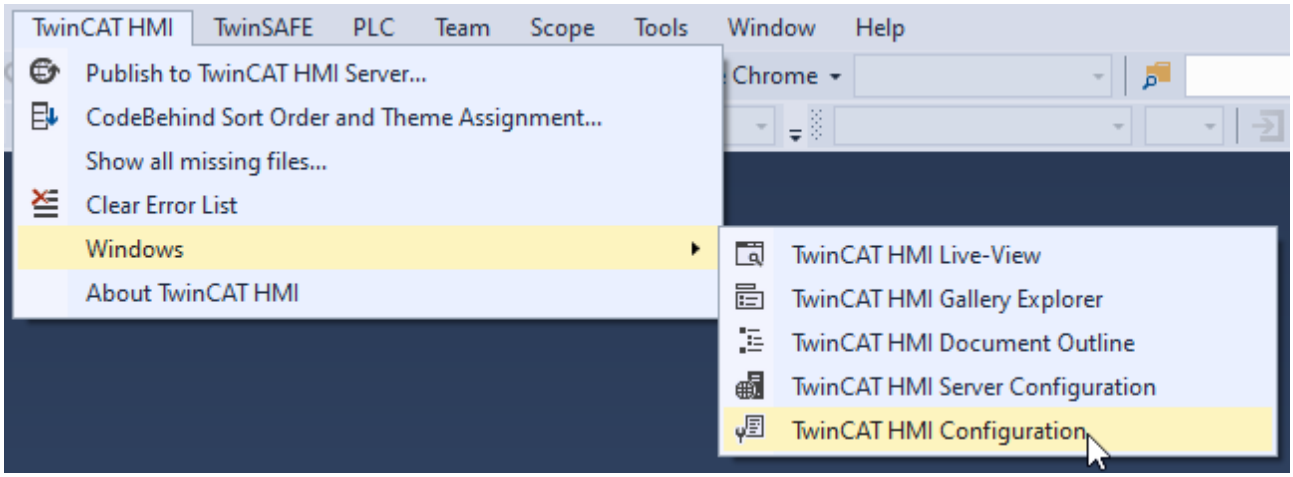
Settings required for each device:

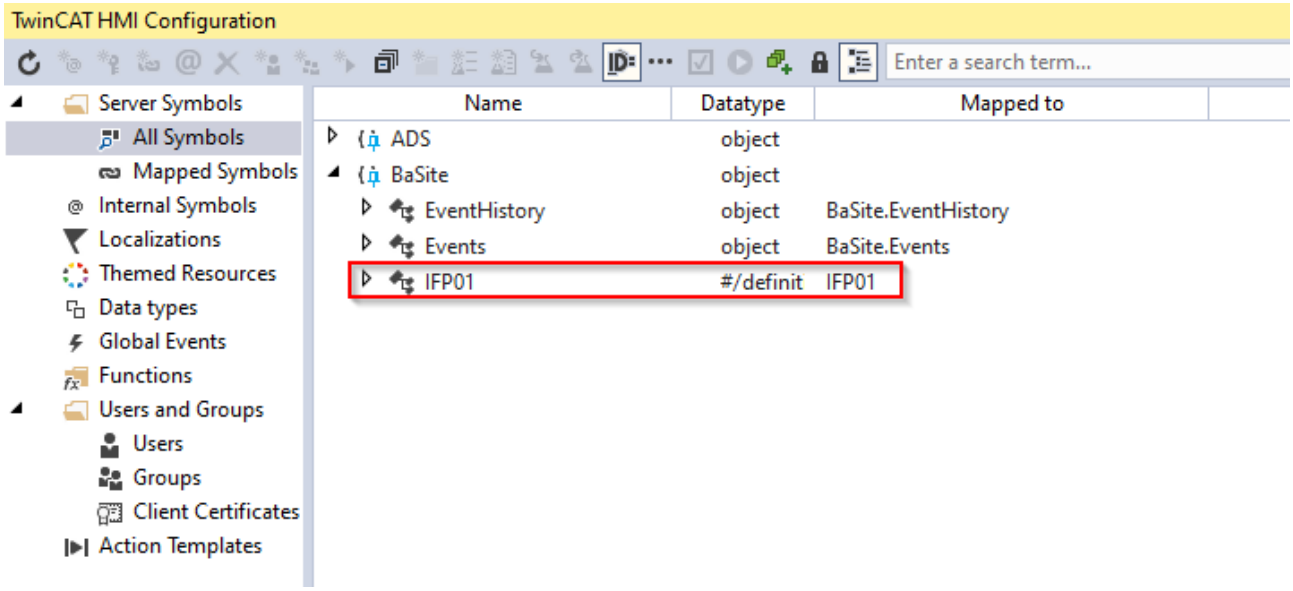
- Determination whether enabled/disabled
- AmsNetId
- PLC port

The configuration is activated by clicking on **Accept**.

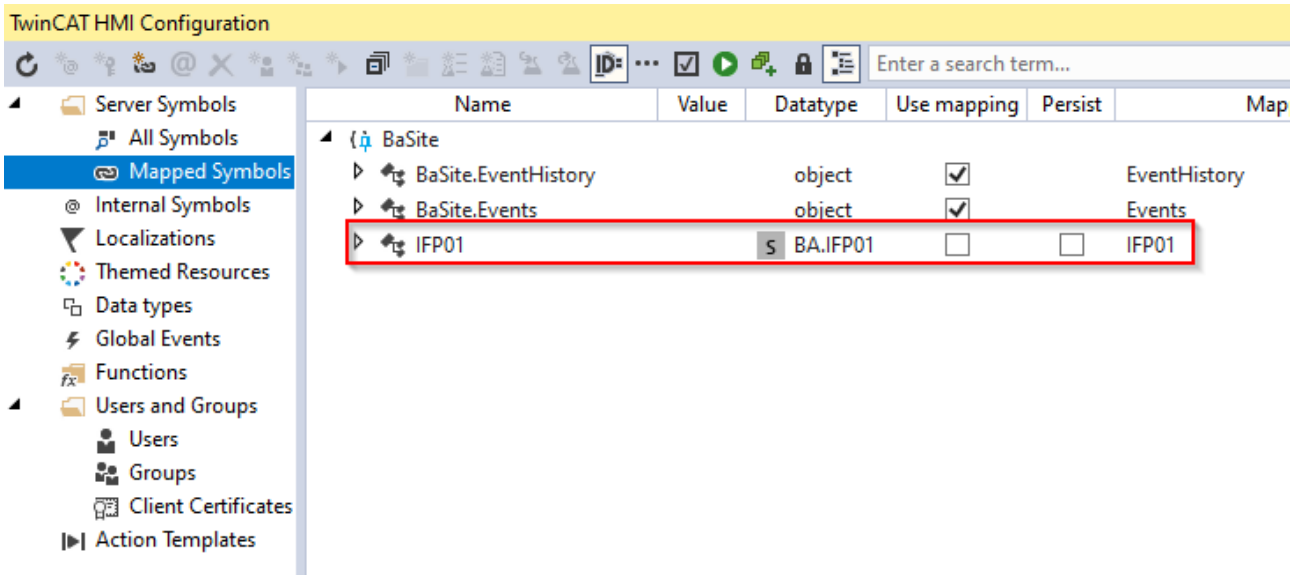
Checking the configuration

In the **TwinCAT HMI Configuration** window, all devices should now be listed under **All symbols**.





A mapping for the device was also created.



Required mappings

The mapping automatically created for each device with **the same** name as the device itself is mandatory and must **not** be renamed or deleted.

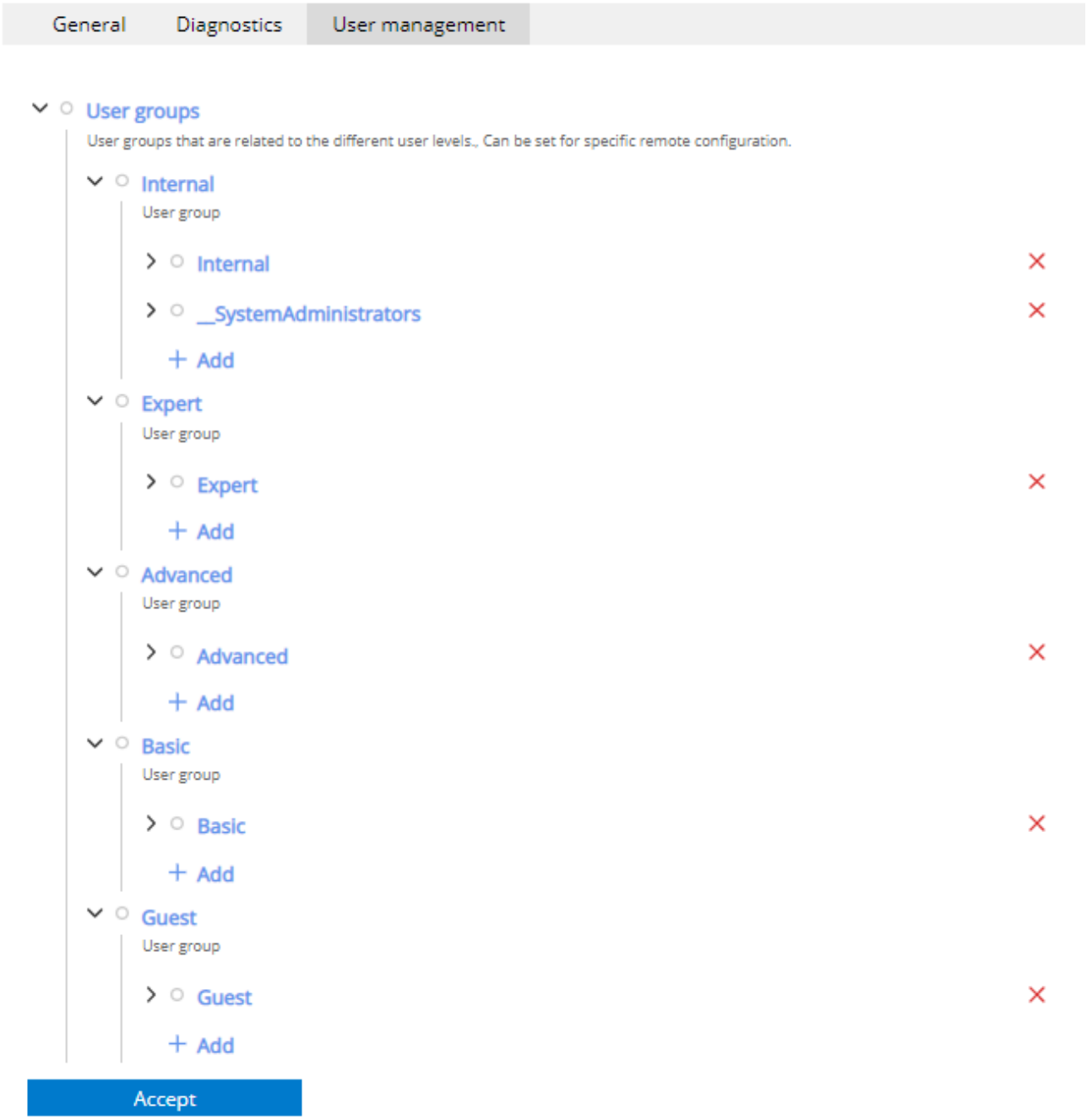
Activating the configuration automatically adds a device of the same name and with the same settings in the ADS extension. This device in the ADS extension is also mandatory and must **not** be renamed or deleted.

User management

User groups

Here, the user groups *_Guest_*, *_Basic_*, *_Advanced_*, *_Expert_* and *_Internal_* can each be assigned user groups from the HMI.

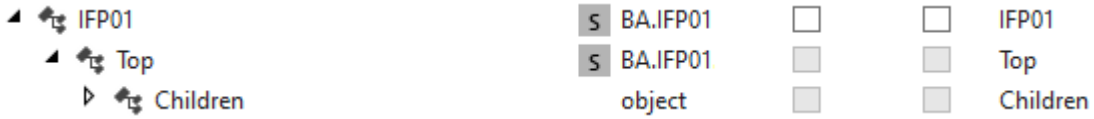
It can be configured that different user groups receive the same user level.



Symbols

Device symbols

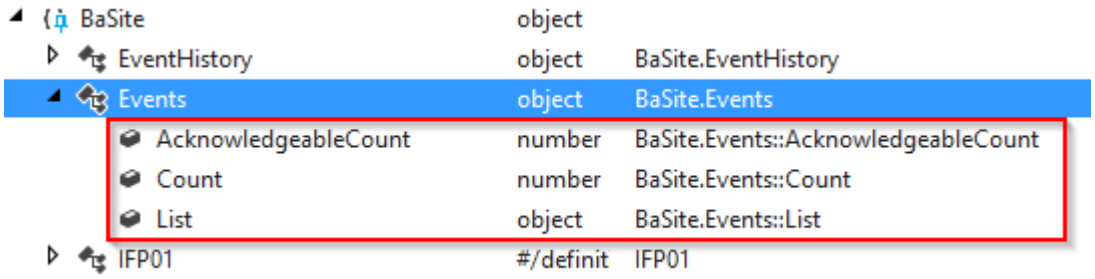
A dynamic symbol with the same name as the device is created for each device. The structure of the first level of a device symbol is always the same.



The first level contains the top node of the device, where the children are located. These Children are thus the first actual elements in the project structure.

Events

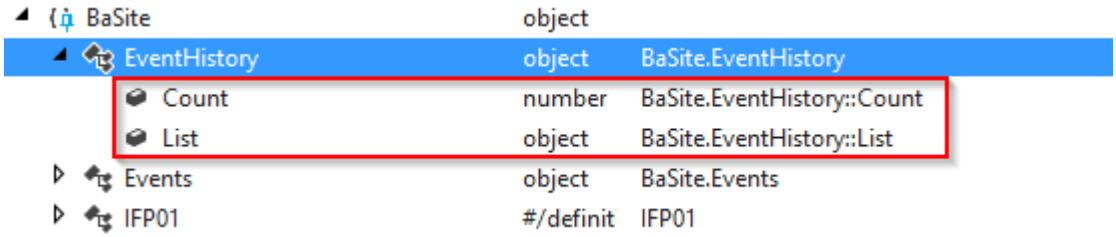
The symbol offers various subsymbols that refer to the current events of **all**.



- AcknowledgeableCount: Active events that require an interaction from the user (acknowledge)
- Count: Number of all events
- List: The list of events
- This makes the symbol suitable for a cross-device event list [▶ 1166].

EventHistory

The symbol offers various subsymbols that refer to the event history of **all** devices.



- Count: Number of all events in the history.
- List: The list of events in the history.
  This makes the symbol suitable for a cross-device event history [▶ 1166].

6.2.1.8 Global settings

The global settings of the various TcHmiBa NuGet packages can be overwritten. The entry point is the callback from the `TcHmiBaEvents.onOverrideSettings` event that is to be created in the CodeBehind.

```
(function (TcHmi) {
  TcHmi.EventProvider.register(TcHmi.BuildingAutomation.TcHmiBaEvents.onOverrideSettings, function (
e, data) {
    e.destroy();
    // Todo
  });
})(TcHmi);
```

The value is then changed in the `ToDo` area, as described below for the respective property.

TcHmiBaControls

EventList

DefaultColumns

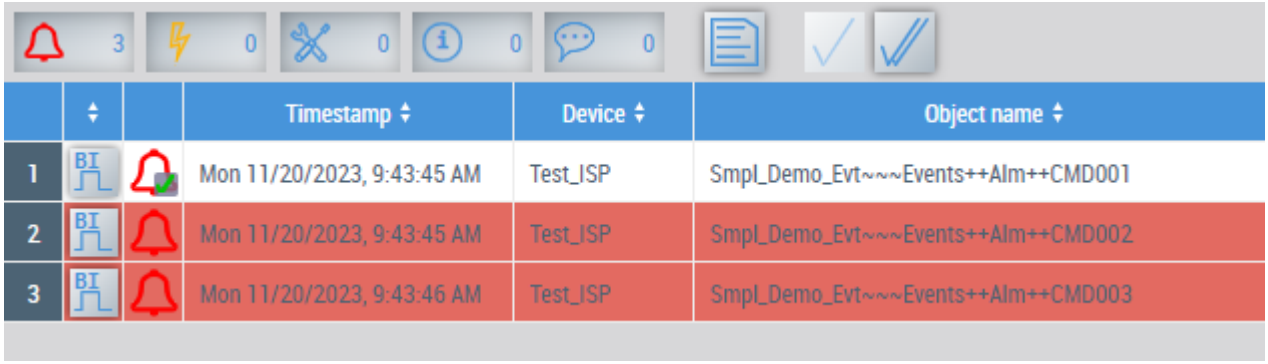
Defines the default column settings for all EventList [▶ 1166] controls.

```
TcHmi.BuildingAutomation.Controls.Management.EventList.DefaultColumns = [
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.baIdentifier, width: 30,
widthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.event, width: 30, widthUn
it: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.timestamp, width: 140, wi
dthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.device, width: 70, widthU
nit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.objectName, width: 300, w
idthUnit: 'px' },
```

```
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.instancePath, width: 300,
 widthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.description, width: 300,
widthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.eventClass, width: 30, wi
dthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.eventClassInstanceDescrip
tion, width: 200, widthUnit: 'px' }
];
```

MaximumEventTypePulse

Defines for each user level whether a pulse is displayed if an active event with this or a higher priority [▶ 103] is pending.



Setting a value.

```
TcHmi.BuildingAutomation.Controls.Management.EventList.MaximumEventTypePulse.set(TcHmi.BuildingAutom
ation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb);
```

Setting all values.

```
TcHmi.BuildingAutomation.Controls.Management.EventList.MaximumEventTypePulse = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventType.eDisturb]
]);
```

ProjectNavigationTextual

DefaultContentViewDialogWidth

Defines the default size of the dialog that displays the content of a BaObject.

```
TcHmi.BuildingAutomation.Controls.Management.ProjectNavigationTextual.DefaultContentViewDialogWidth
= 1000;
```

RoomControl

DefaultBaObjectListDialogWidth

Defines the default size of the dialog that displays the BaObject list.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.RoomControl.DefaultBaObjectListDialogWidth = 1000;
```

Light

ShowQuickLinks

Defines whether the quick settings are displayed for Light-Control [▶ 1203].

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.ShowQuickLinks = false;
```

PriorityIcons

Defines the icons that are displayed on the Light-Control [▶ 1203] for a certain priority.

If no icon is defined for a priority, nothing is displayed.

Setting a value.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.PriorityIcons.set(TcHmi.BuildingAutomation.Co
ntrols.RoomAutomation.Light.Priority.fire,
TcHmi.BuildingAutomation.Icons.convertIIconDataToIIconAttributes({ ...
TcHmi.BuildingAutomation.Icons.Building.FireAlarm, color:
TcHmi.BuildingAutomation.Color.RGBAColor.Red }));
```

Setting all values.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.PriorityIcons = new Map([
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.fire, TcHmi.BuildingAutomation.
Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.FireAlarm, colo
r: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.communicationError, TcHmi.Build
ingAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Events.Fla
g.Fault, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.burglary, TcHmi.BuildingAutomat
ion.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.BurglarAlar
m, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.maintenance, TcHmi.BuildingAuto
mation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.Maintena
nce, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.cleaning, TcHmi.BuildingAutomat
ion.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.Cleaning, c
olor: TcHmi.BuildingAutomation.Color.RGBAColor.Yellow })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.nightWatch, TcHmi.BuildingAutom
ation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.NightWatc
h, color: TcHmi.BuildingAutomation.Color.RGBAColor.Blue })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.manual, TcHmi.BuildingAutomatio
n.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomation.Manual,
 color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.automaticLight, TcHmi.BuildingA
utomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomation
.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })]
]);
```

Sunblind

PriorityIcons

Defines the icons that are displayed on the Sunblind control [▶ 1210] for a certain priority.

If no icon is defined for a priority, nothing is displayed.

Setting a value.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.PriorityIcons.set(TcHmi.BuildingAutomation
.Controls.RoomAutomation.Sunblind.Priority.fire, TcHmi.BuildingAutomation.Icons.convertIIconDataToII
conAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.FireAlarm, color:
TcHmi.BuildingAutomation.Color.RGBAColor.Red }));
```

Setting all values.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.PriorityIcons = new Map([
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.fire, TcHmi.BuildingAutomati
on.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.FireAlarm, c
olor: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.storm, TcHmi.BuildingAutomat
ion.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.StormAlarm,
 color: TcHmi.BuildingAutomation.Color.RGBAColor.Blue })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.ice, TcHmi.BuildingAutomatio
n.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.IceAlarm, col
or: TcHmi.BuildingAutomation.Color.RGBAColor.Blue })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.commError, TcHmi.BuildingAut
omation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Events.Flag.Faul
t, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.burglary, TcHmi.BuildingAuto
mation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.BurglarA
larm, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.maintenance, TcHmi.BuildingA
```

```
utomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.Maint
enance, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.manualActuator, TcHmi.Buildi
ngAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomat
ion.Manual, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.manualGroup, TcHmi.BuildingA
utomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomation
.Manual, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.facadeThermoAutomatic, TcHmi
.BuildingAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Room
Automation.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.facadeTwilightAutomatic, TcH
mi.BuildingAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Ro
omAutomation.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.sunProtection, TcHmi.Buildin
gAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomati
on.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })]
]);
```

Window

ShowQuickLinks

Defines whether the quick settings are displayed for the Window [▶ 1214] control.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Window.ShowQuickLinks = false;
```

TcHmiBaFramework

BusyHandler

RecordTimerResults

Defines whether the timer results for various processes are recorded.

```
TcHmi.BuildingAutomation.BusyHandler.RecordTimerResults = false;
```

Button

DoublePressDuration

Defines the time that may elapse between two clicks until the two clicks are recognized as a double click.

```
TcHmi.BuildingAutomation.Components.Button.DoublePressDuration = 200;
```

Calendar

DefaultEventDialogSize

Defines the default size of the event dialog.

```
TcHmi.BuildingAutomation.Components.Calendar.DefaultEventDialogSize = {
    width: 700,
    height: 350
};
```

CalendarList

DefaultCalendarListDialogSize

Defines the default size of the CalendarList dialog.

```
TcHmi.BuildingAutomation.Components.CalendarList.DefaultCalendarListDialogSize = {
    width: 750,
    height: 400
};
```

ContentWindow

HideHeaderIconBorderDefault

Defines whether the border of the header icon is hidden by default. Visibility can still be set separately for each instance.

```
TcHmi.BuildingAutomation.Components.ContentWindow.HideHeaderIconBorderDefault = false;
```

DateTimePicker

DefaultDateTimePickerDialogSize

Defines the default size of the DateTimePicker dialog.

```
TcHmi.BuildingAutomation.Components.DateTimePicker.DefaultDateTimePickerDialogSize = {
    width: 650,
    height: 450
};
```

DefaultDatePickerDialogSize

Defines the default size of the DatePicker dialog.

```
TcHmi.BuildingAutomation.Components.DateTimePicker.DefaultDatePickerDialogSize = {
    width: 300,
    height: 300
};
```

DialogWindow

MoveLimitation

Limitation for positioning outside the browser window.

```
TcHmi.BuildingAutomation.Components.DialogWindow.MoveLimitation = 100;
```

TimePicker

DefaultTimePickerDialogSize

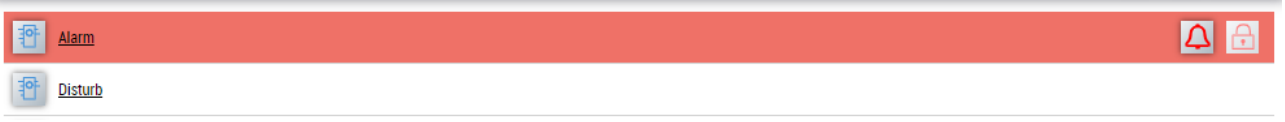
Defines the default size of the TimePicker dialog.

```
TcHmi.BuildingAutomation.Components.TimePicker.DefaultTimePickerDialogSize = {
    width: 350,
    height: 200
};
```

ProjectNavigationList

MaximumEventTypePulse

Defines for each user level whether a pulse is displayed if an active event with this or a higher priority [▶ 103] is pending.



Setting a value.

```
TcHmi.BuildingAutomation.Navigation.ProjectNavigationList.MaximumEventTypePulse.set(TcHmi.BuildingAu
tomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb);
```

Setting all values.

```
TcHmi.BuildingAutomation.Navigation.ProjectNavigationList.MaximumEventTypePulse = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
```

```
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventType.eDisturb]
]);
```

MaximumEventConditionDisplayed

Defines the maximum EventCondition [▶ 102] for each user level, which is displayed in the lines of the ProjectNavigation [▶ 1167].

Setting a value.

```
TcHmi.BuildingAutomation.Navigation.ProjectNavigationList.MaximumEventConditionDisplayed.set(TcHmi.B
uildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther);
```

Setting all values.

```
Navigation.ProjectNavigationList.MaximumEventConditionDisplayed = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventCondition.eEventIcon
Displayed],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed]
]);
```

UiIcon

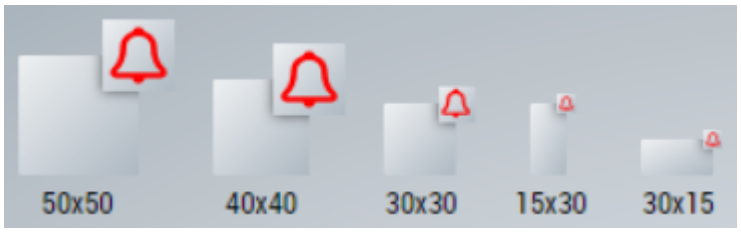
AutoActivateIconStatus

Defines whether the IconStatus [▶ 1162] attribute is set automatically when events are configured. With this setting, the icon is colored in the defined event color when an event is active.

```
TcHmi.BuildingAutomation.Components.UiIcon.AutoActivateIconStatus = false;
```

AutoScaleEventIconThreshold

Defines the threshold value that the height or width must fall below before the automatic scaling of the event icon becomes active.




```
TcHmi.BuildingAutomation.Components.UiIcon.AutoScaleEventIconThreshold = 40;
```

EnableEventCountBadge

Defines whether the number of events of the same event type are displayed.


```
TcHmi.BuildingAutomation.Components.UiIcon.EnableEventCountBadge = true;
```

EventIconSize

Defines the initial height and width of the EventIcon.

```
TcHmi.BuildingAutomation.Components.UiIcon.EventIconSize = 30;
```

MaximumEventTypePulse

Defines for each user level whether a pulse is displayed if an active event with this or a higher priority [▶ 103] is pending.

Setting a value.

```
TcHmi.BuildingAutomation.Components.UiIcon.MaximumEventTypePulse.set(TcHmi.BuildingAutomation.BA.Rol
e.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb);
```

Setting all values.

```
Components.UiIcon.MaximumEventTypePulse = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventType.eDisturb]
]);
```

MaximumEventConditionDisplayed

Defines the maximum EventCondition [▶ 102] for each user level, which is displayed at UiIcon [▶ 1159].

Setting a value.

```
TcHmi.BuildingAutomation.Components.UiIcon.MaximumEventConditionDisplayed.set(TcHmi.BuildingAutomati
on.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther);
```

Setting all values.

```
TcHmi.BuildingAutomation.Components.UiIcon.MaximumEventConditionDisplayed = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventCondition.eEventIcon
Displayed],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed]
]);
```

Charting

Axis

DefaultAxisOptionsDialogSize

Defines the default size of the dialog for the axis options.

```
TcHmi.BuildingAutomation.Charting.Axis.DefaultAxisOptionsDialogSize = {
    height: 180,
    width: 350
};
```

Trend

DefaultDisplayedObjects

Defines the default object in the trend display.

```
TcHmi.BuildingAutomation.Charting.Trend.DefaultDisplayedObjects =
TcHmi.BuildingAutomation.Trend.DisplayedObjects.trendableObjects;
```

DefaultDisplayedDescription

Defines the default description in the trend display.

```
TcHmi.BuildingAutomation.Charting.Trend.DefaultDisplayedDescription =
TcHmi.BuildingAutomation.BA.BaParameterId.eInstDescription;
```

CollectionConfigurator

DefaultConfiguratorDialogWidth

Defines the default width of the trend collection configurator dialog.

```
TcHmi.BuildingAutomation.Charting.Trend.CollectionConfigurator.DefaultConfiguratorDialogWidth = 1000
;
```

DefaultTrendCollectionSelectionDialogWidth

Defines the default width of the dialog for the selection of trend collections to be displayed.

```
TcHmi.BuildingAutomation.Charting.Trend.CollectionConfigurator.DefaultTrendCollectionSelectionDialog
Width = 400;
```

Storage

UserData

LastContentSorageLocation

Defines the location where a user's last opened content is saved.

```
TcHmi.BuildingAutomation.Storage.UserData.LastContentStorageLocation = TcHmi.BuildingAutomation.Stor
age.Location.baSite;
```

LastThemeSorageLocation

Defines the location where a user's last used theme is saved.

```
TcHmi.BuildingAutomation.Storage.UserData.LastThemeStorageLocation = TcHmi.BuildingAutomation.Storag
e.Location.baSite;
```

TrendSettingsStorageLocation

Defines the location where a user's trend settings are saved.

```
TcHmi.BuildingAutomation.Storage.UserData.TrendSettingsStorageLocation = TcHmi.BuildingAutomation.St
orage.Location.baSite;
```

TrendCollectionStorageLocation

Defines the location where a user's trend collections are saved.

```
TcHmi.BuildingAutomation.Storage.UserData.TrendCollectionStorageLocation = TcHmi.BuildingAutomation.
Storage.Location.baSite;
```

TrendCollectionSelectionStorageLocation

Defines the location where a user's displayed trend collections are saved.

```
TcHmi.BuildingAutomation.Storage.UserData.TrendCollectionSelectionStorageLocation = TcHmi.BuildingAu
tomation.Storage.Location.baSite;
```

Server

BaSite-Extension

DefaultDiagnosticsDialogSize

Defines the default size of the diagnostics dialog.

```
TcHmi.BuildingAutomation.Server.BaSite.DefaultDiagnosticsDialogSize = {
    height: 500,
    width: 500
};
```

EventHelper

MaximumEventCondition

Defines the maximum EventCondition [▶ 102] for each user level, that is displayed in the HMI .

Default setting displays all events [▶ 1235], flags [▶ 1236], priorities [▶ 1238] and locks [▶ 1238].

Setting a value.

```
TcHmi.BuildingAutomation.BA.EventHelper.MaximumEventCondition.set(TcHmi.BuildingAutomation.BA.Role.e
Guest, TcHmi.BuildingAutomation.BA.EventCondition.eEventIconDisplayed);
```

Setting all values.

```
TcHmi.BuildingAutomation.BA.EventHelper.MaximumEventCondition = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eEventIconD
isplayed],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventCondition.eEventIconD
isplayed],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventCondition.eEventIcon
Displayed],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed]
]);
```

BA

HiddenBaParameterIds

Defines the BaParameterIds that are not displayed in the generic parameter dialog.

```
TcHmi.BuildingAutomation.BA.BaDevice.HiddenBaParameterIds = [];
```

BaParameterCategories

Organizes the different BaParameterIds into categories. The order in which they are added is taken into account when they are displayed in the parameter dialog.

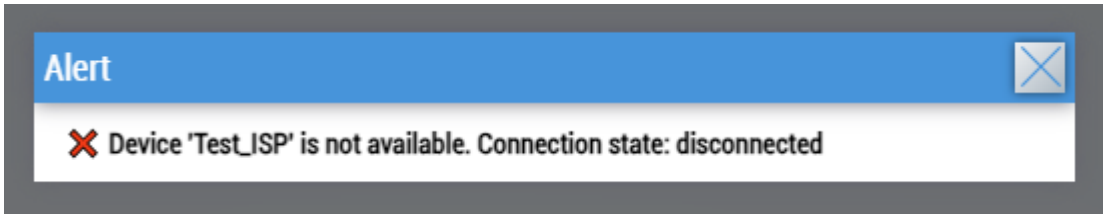
Setting a value.

```
TcHmi.BuildingAutomation.BA.BaParameterCategories.set(TcHmi.BuildingAutomation.BA.BaParameterCategor
y.configuration, [
    TcHmi.BuildingAutomation.BA.BaParameterId.eConfigurate,
    TcHmi.BuildingAutomation.BA.BaParameterId.eToggleMode,
    TcHmi.BuildingAutomation.BA.BaParameterId.eStepDelay,
    TcHmi.BuildingAutomation.BA.BaParameterId.eMinOffTime,
    TcHmi.BuildingAutomation.BA.BaParameterId.eMinOnTime
]);
```

BaDevice

DialogConnectionAutoCloseStateChanged

Defines whether the notification is automatically closed when a device loses the connection after the connection is restored.



```
TcHmi.BuildingAutomation.BA.BaDevice.DialogConnectionAutoCloseStateChanged = true;
```

DialogConnectionAutoReloadOnReconnect

Defines whether the HMI is automatically reloaded after the connection to a device is restored.

```
TcHmi.BuildingAutomation.BA.BaDevice.DialogConnectionAutoReloadOnReconnect = true;
```

DialogConnectionAutoReloadOnReconnectTime

Defines the delay time until the HMI is automatically reloaded after the connection to a device is restored.

```
TcHmi.BuildingAutomation.BA.BaDevice.DialogConnectionAutoReloadOnReconnectTime = 30;
```

BaBasicObject

DisableParameterDialogHeaderIcon

Defines whether the object/node icon is not displayed in the header of the parameter dialog.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DisableParameterDialogHeaderIcon = false;
```

DefaultParameterDialogWidth

Defines the default width of the parameter window.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DefaultParameterDialogWidth = 800;
```

DefaultOnlineTrendDialogSize

Defines the standard size of the window of the online trend.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DefaultOnlineTrendDialogSize = {
    height: 250,
    width: 450
}
```

DefaultNavigationDialogWidth

Defines the default width of the dialog that displays the ProjectNavgiationList of a BaObject.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DefaultNavigationDialogWidth = 1000;
```

AutoCollapseNavigationDialogEntries

Specifies whether only one entry or multiple entries may be open in the dialog that displays the ProjectNavigationList.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.AutoCollapseNavigationDialogEntries = true;
```

BaView

DisableNodeTypeIcons

DisableNodeTypeIcons

Defines whether the NodeTypeIcons [▶ 1239] are used in the HMI.

```
TcHmi.BuildingAutomation.BA.BaView.DisableNodeTypeIcons = false;
```

6.2.1.9 Project templates

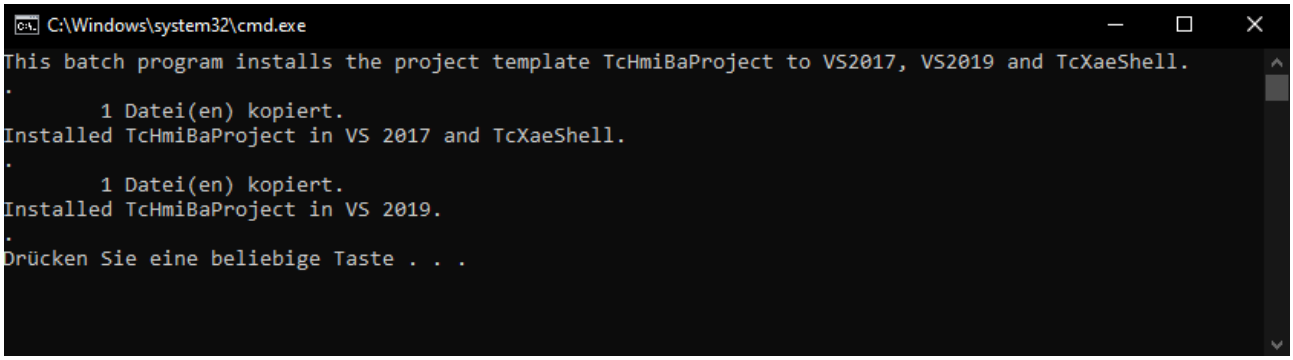
Project templates are intended to make it easier to get started with a *TcHmiBa* project by pre-installing required dependencies and providing additional elements (e.g. navigation).

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.



The project templates currently have some limitations. The following section describes these in more detail and how they can be remedied.

User management

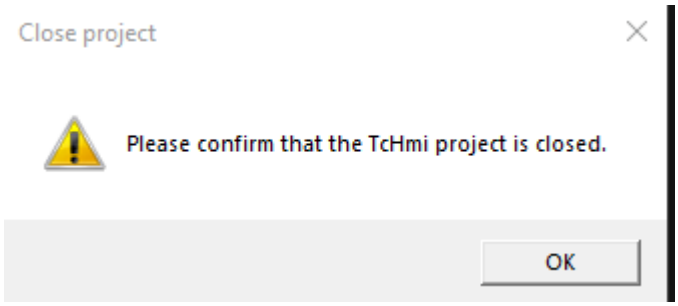
In order to use the integrated user management of TF8040, it is necessary that certain user groups and matching users exist in the TcHmi project. The project template does **not** include these users.
The program *CreateDefaultUserManagement.exe* can be used to create the required settings for a project.

More detailed information on the various user groups [▶ 15] can be found here.

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.

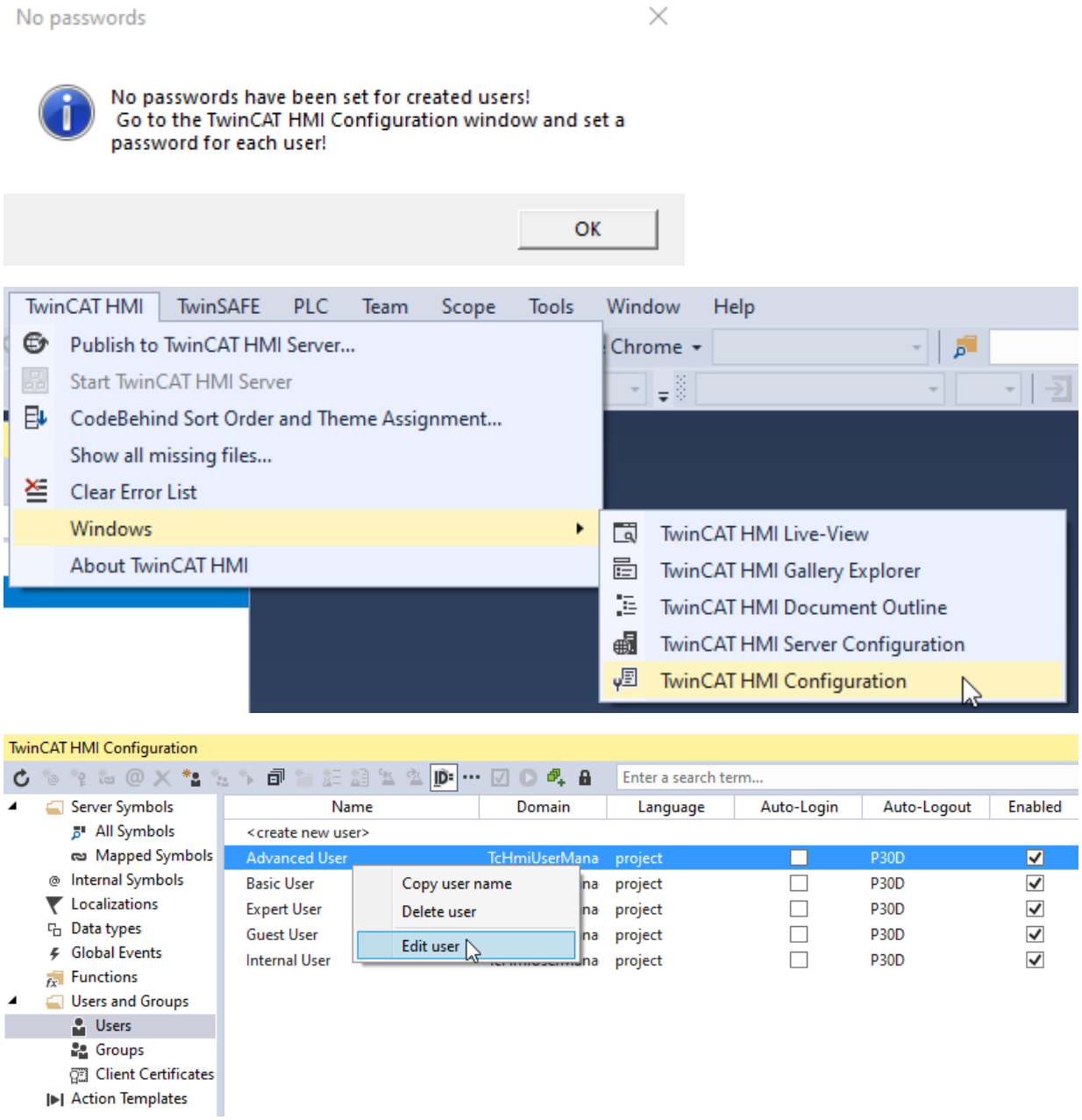
1. First select the HMI project.
2. Close the project (if not already closed).



3. Check output

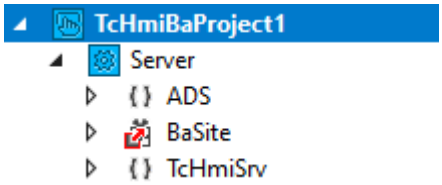
```
Please select TcHmi project file:
Selected project file: 'C:\temp\TcHmiBaProject1\TcHmiBaProject1.hmiproj'

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.
```

4. Open the project and create passwords for the created users.

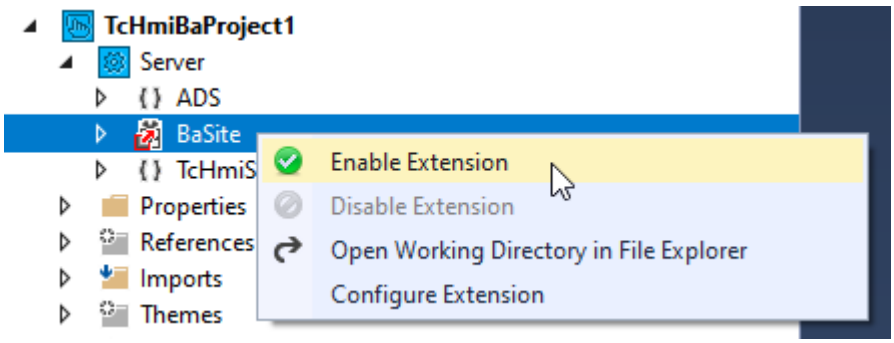


BaSite-Extension

The server extension will start inactive because the configuration files are not loaded with the project template.

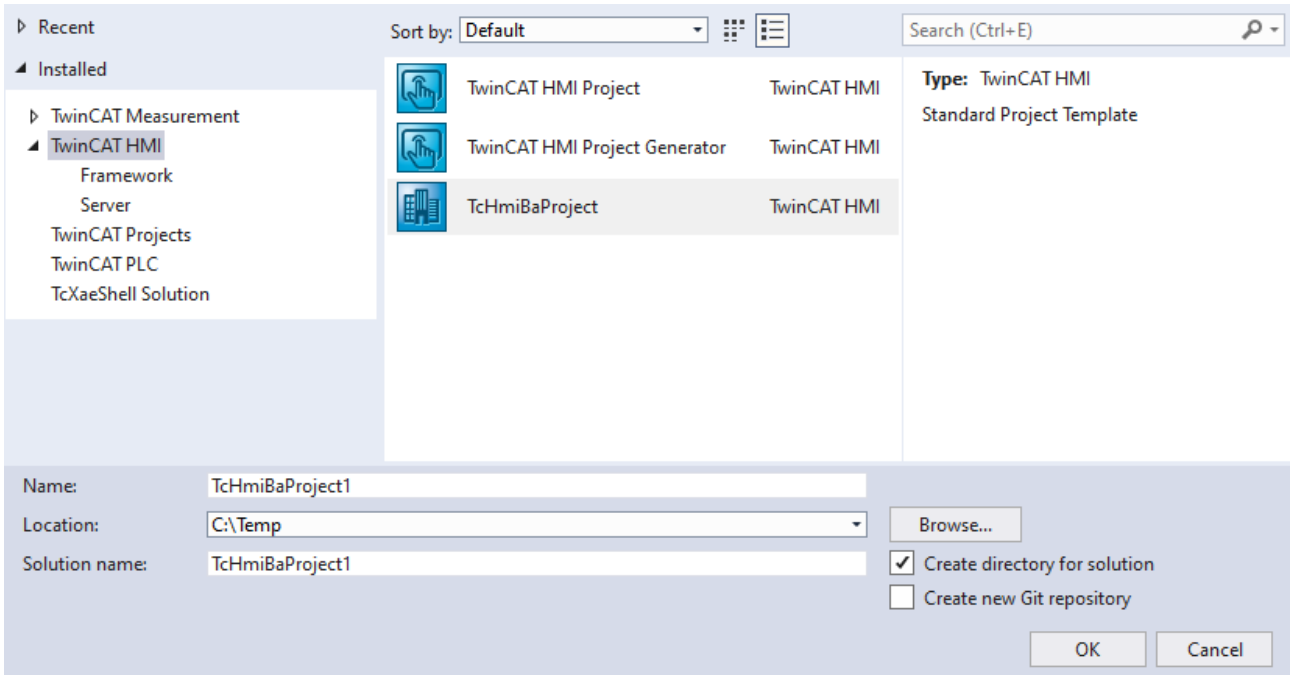


Therefore a manual start is necessary.



6.2.1.9.1 TcHmiBaProject

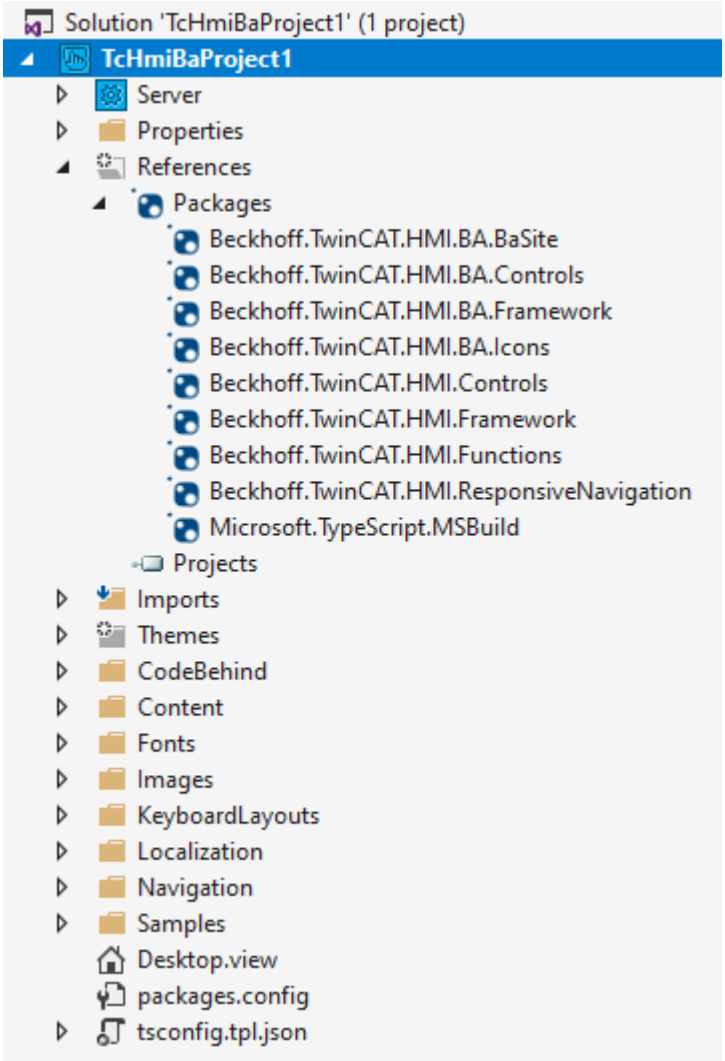
The project template is used to create a TcHmi project quickly and easily.



The notes [▶ 1276] for project templates must be observed.

Contents

Preview



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
- Microsoft.TypeScript.MSBuild

CodeBehind

Listing of all global settings for TcHmiBa.

- TcHmiBaSettings.js

Content

Pages

To make it easier to get started, the project template already contains some content pages, such as navigation in the header.

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 [▶ 1283] of the header must be updated (see Navigation).

Navigation

Configurable menu bar.

- HeaderMenu.usercontrol

Samples

The "Samples" folder contains samples of UserControls for creating user-specific plants.

6.2.1.9.1.1 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).

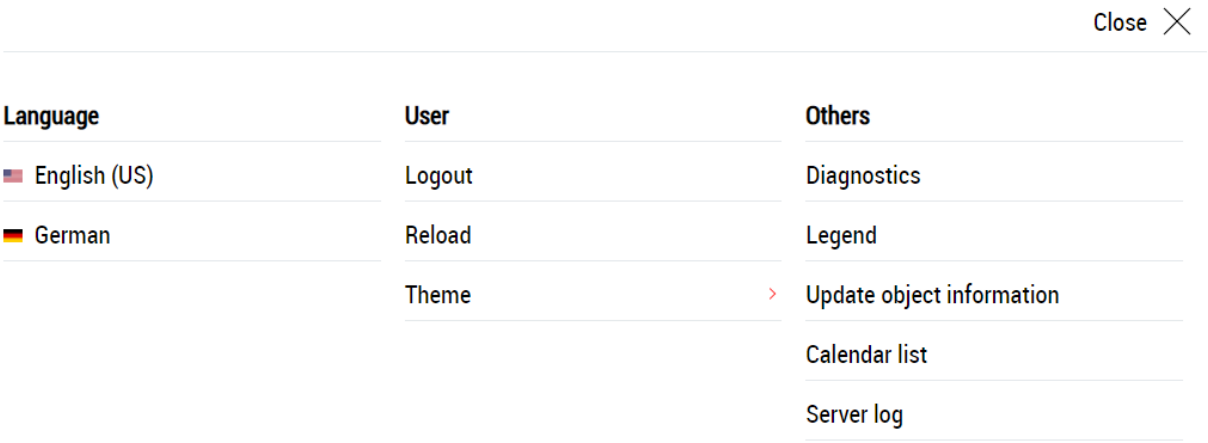
- Logo (1)
- Responsive navigation (2)
- User settings and further information (3)
- Event list (4)
- Building information (5)
- Outdoor temperature (6)
- Date and time (7)

User settings and further information

In this menu, the user can make various settings and display dialogs:

- Setting the language
- Setting the theme
- Display diagnostic data of the BaSite-Extension

- Show dialog with icon legend
- Updating the general object information that is not updated cyclically (e.g. description, active text, status texts)
- List for displaying and editing the calendar objects in the project
- Show server log



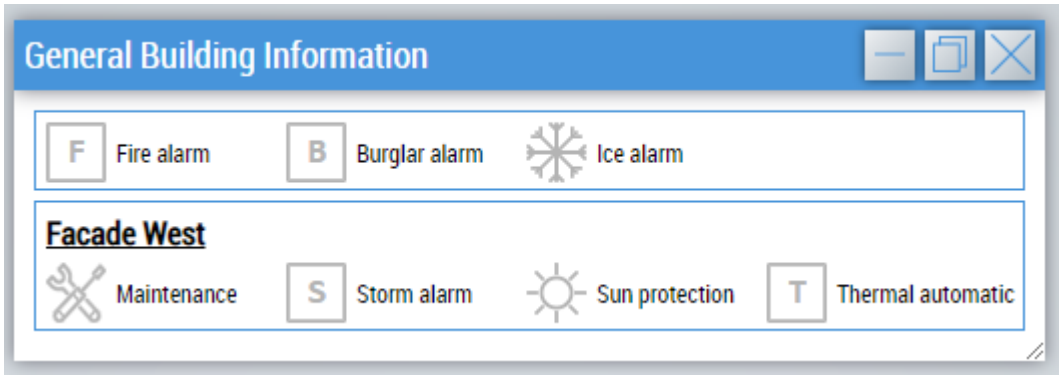
Event list

The event list [▶ 1166] can be accessed via the button with the bell symbol. The button also shows the number of active events if the number was linked via the *EventCount* attribute.

Building information

The button with the info symbol is the control BuildingInformation [▶ 1123].

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

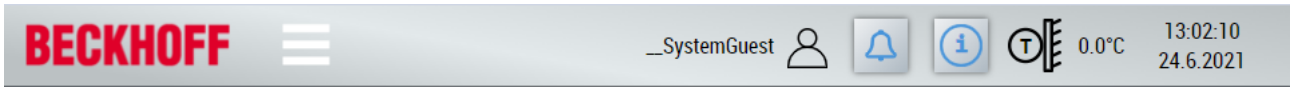
```
tchmi:framework#/definitions/TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationBar.MenuItemList
```

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 outside 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:framework#/definitions/TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent
```

Content to display in the responsive navigation.

UserContent

```
tchmi:framework#/definitions/TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent
```

Content for display in the user menu.

TargetRegion

```
tchmi:framework#/definitions/TcHmiRegion
```

The *TcHmiRegion* must be linked here, 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 Building Automation Site Explorer

The Building Automation Site Explorer maps all objects clearly in the project structure.

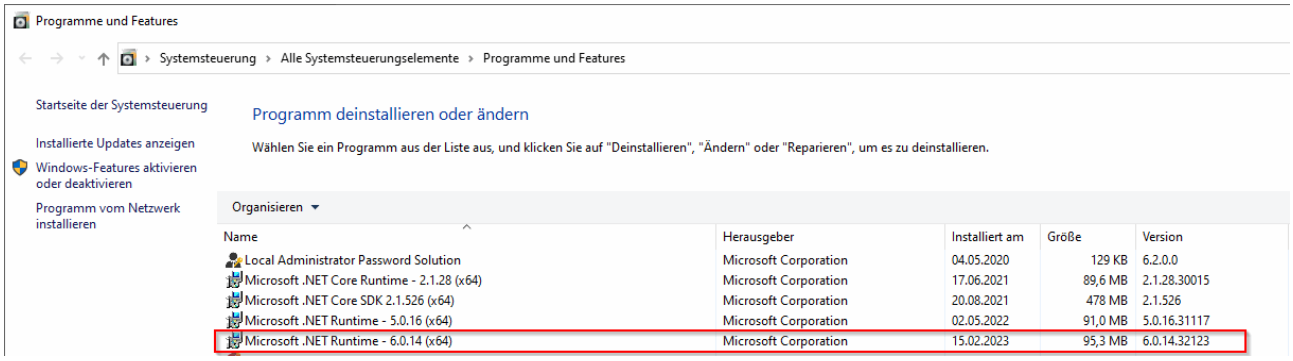
He assists in the commissioning and adjustment of plants.

System requirements

Microsoft:

- Windows 10 or higher
- .NET Desktop Runtime > v6.0

The current version of the .NET Runtime can be checked in the Control Panel under **Programs and Features**.



Beckhoff:

One of the following components must be installed to use the Building Automation Site Explorers:

- TC1000 | TC3.1 ADS
- TE1000 | TC3 Engineering

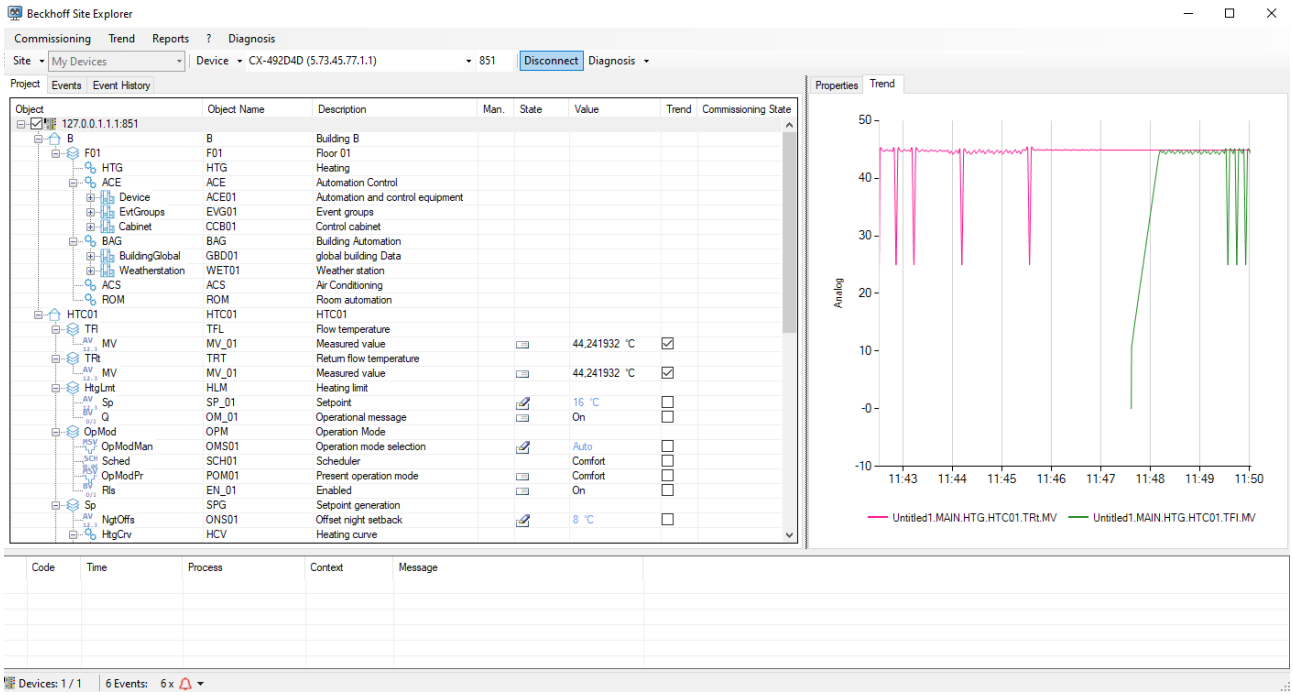
Application

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



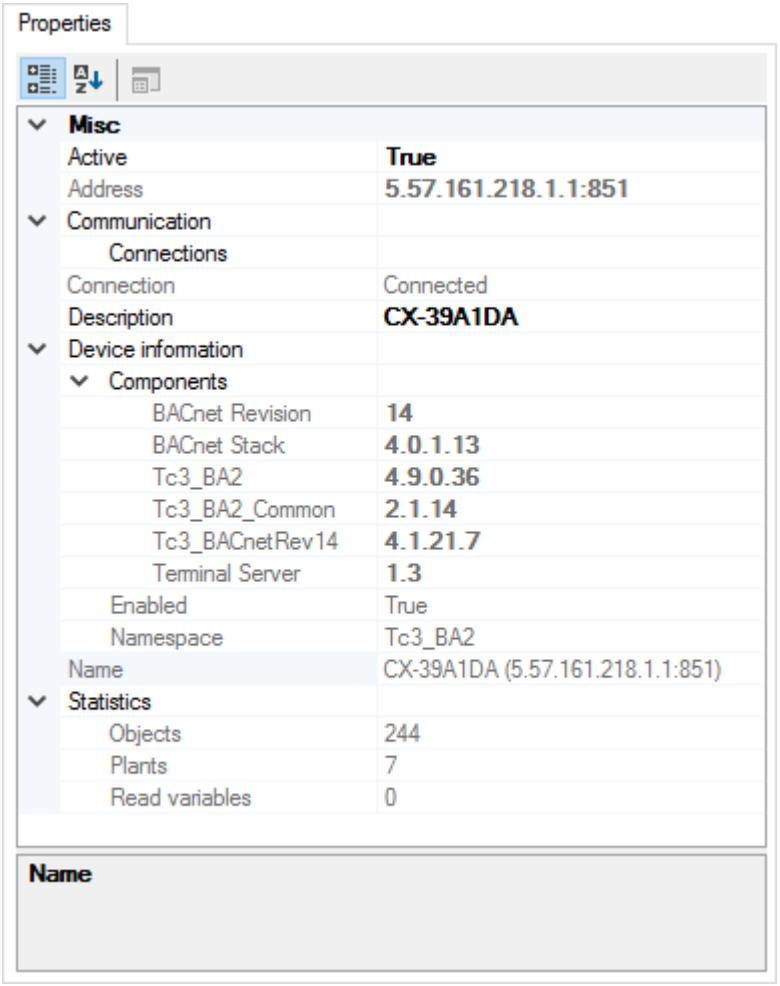
Columns

- Object [▶ 32]
- Object name
- Description
- State
  The following states are displayed:
  - Value source
    Indicates a setpoint or display value.
  - Active event [▶ 32]
  - Overridden
  - Out of service
  - Active priority
- Manual override
- Current value
- Trend
- Commissioning condition

Properties

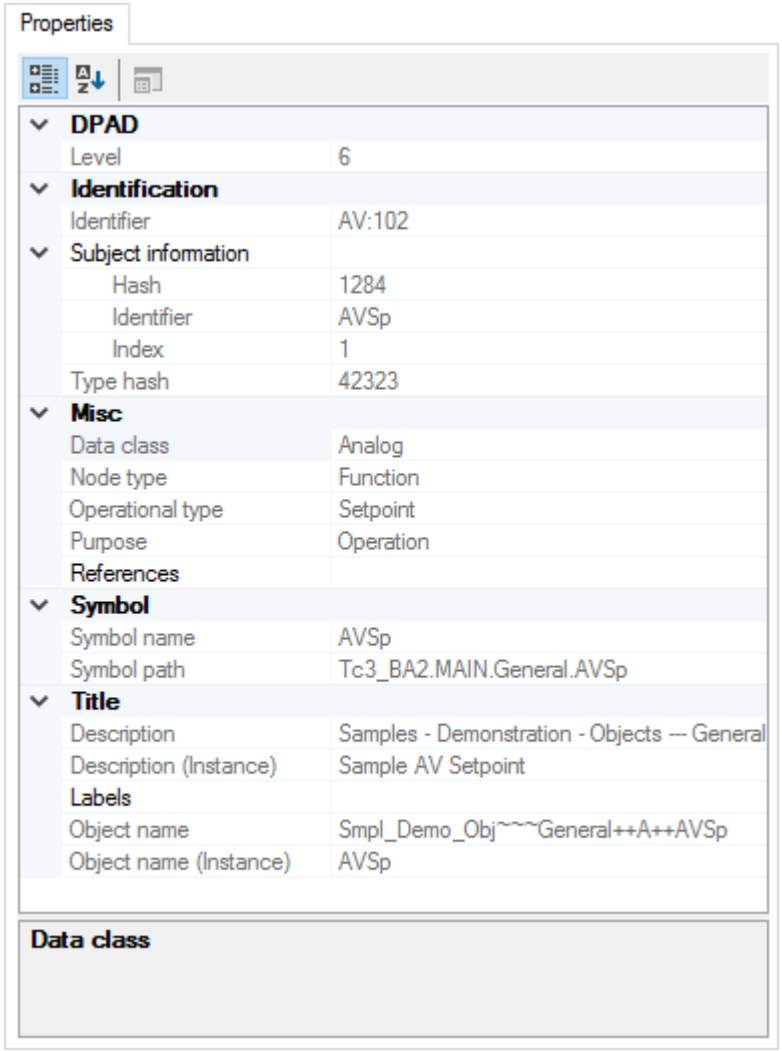
Displays the properties of the selected entry.

- **Control**
  Among other things, the properties of a control list components (such as services or supplements) that are executed at runtime.



Example: Properties of a control

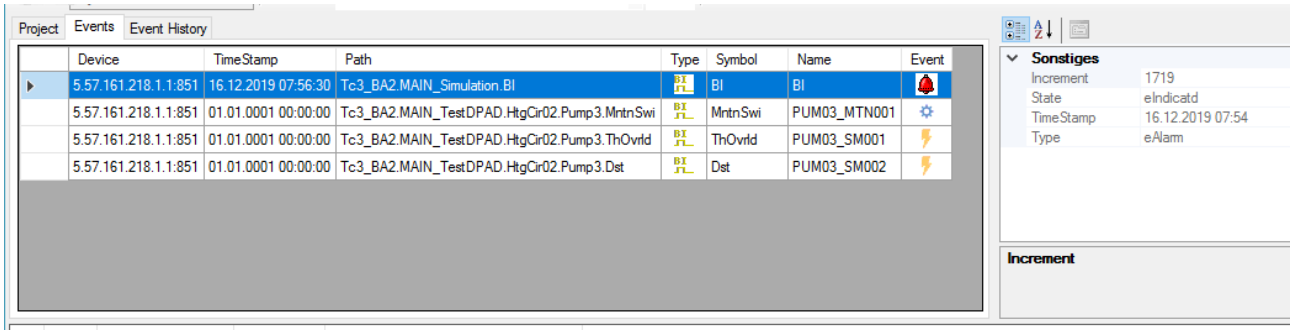
- **Objects**



Example: Properties of an analog object.

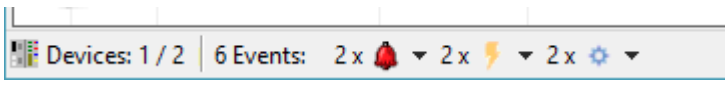
Events

Active events [▶ 32] are listed in the event overview:

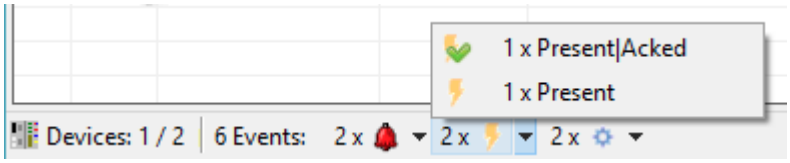


In addition, a summary of all active events is displayed in the status bar at the bottom.

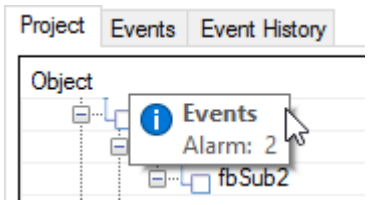
For each event type, the highest priority event symbol is indicated:



Individual states of the combined events [▶ 32] can be viewed via the drop-down menu:

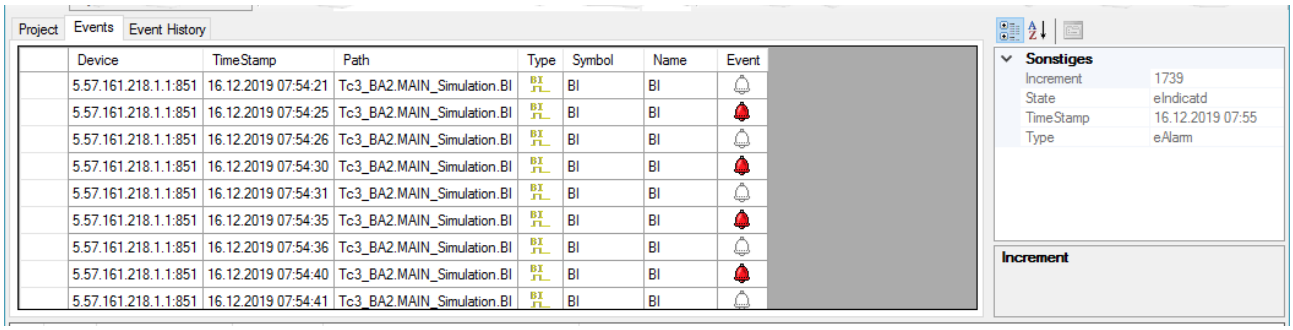


In the project view, a summary of all active events [▶ 32] of a view can also be displayed by moving the mouse over the entry:



Event history

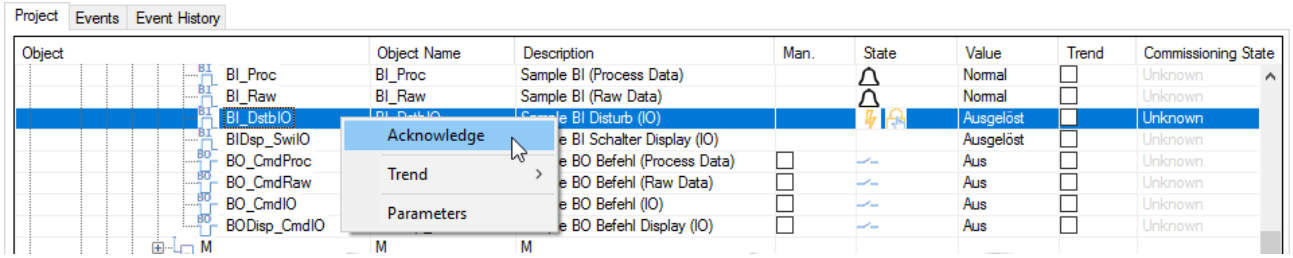
All occurred events [▶ 32] of a view are listed in the event history:



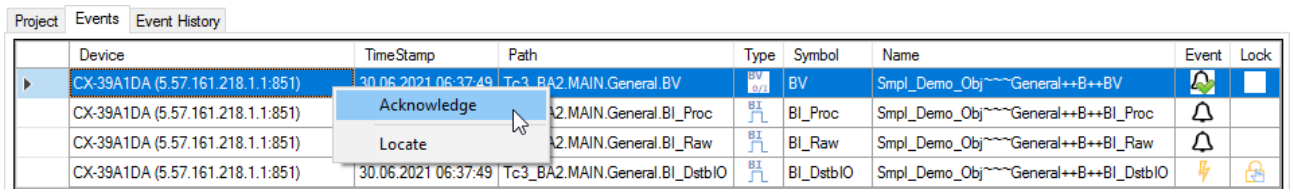
Functions

Acknowledge

Events [▶ 32] can be acknowledged from different views via the context menu:



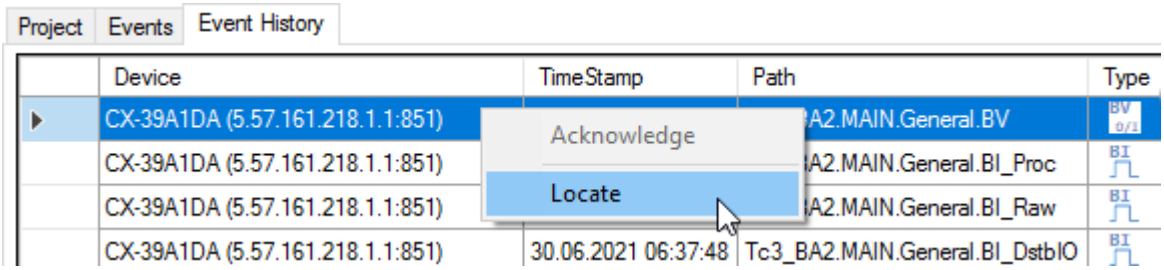
Example: Acknowledge fault in the project view.



Example: Acknowledge alarm in the event overview.

Navigate to an object

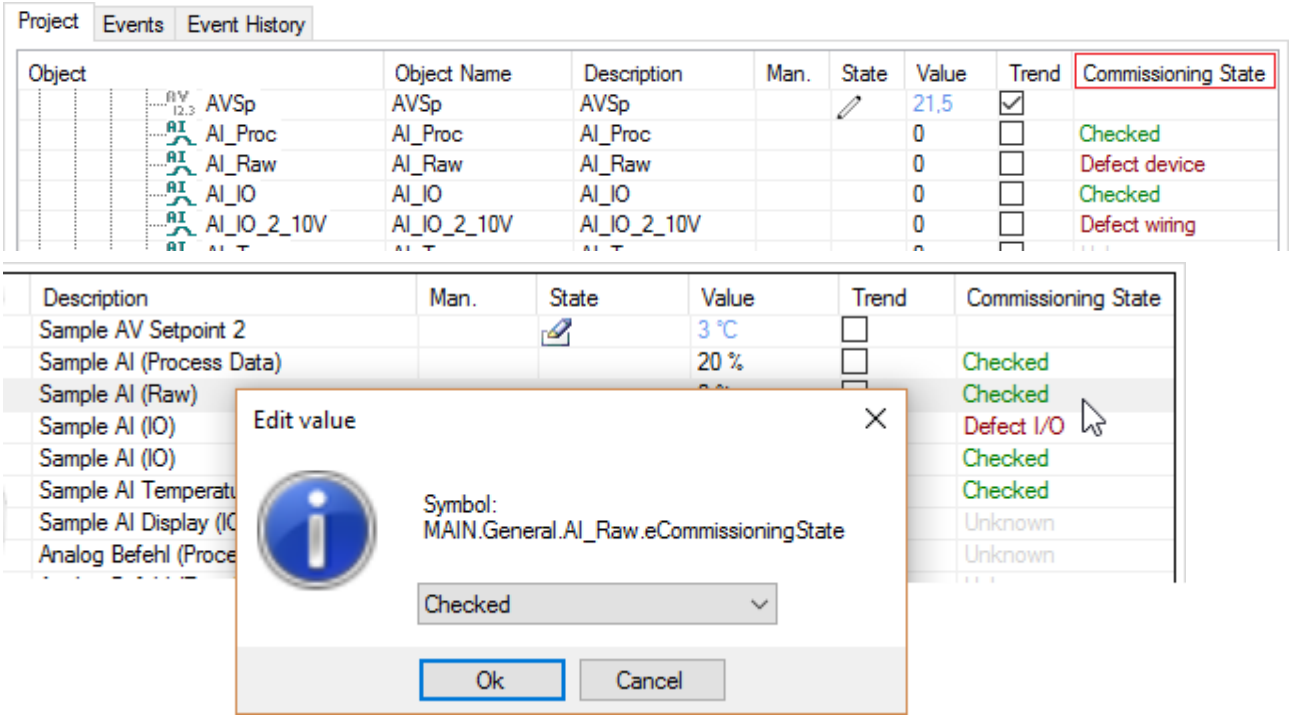
The context menu can be used to navigate directly to the selected object [▶ 32] (in the project view [▶ 1285]):



Example: Navigation to the selected object from the event history.

Commissioning:

Commissioning states of individual objects can be adjusted in the corresponding column:

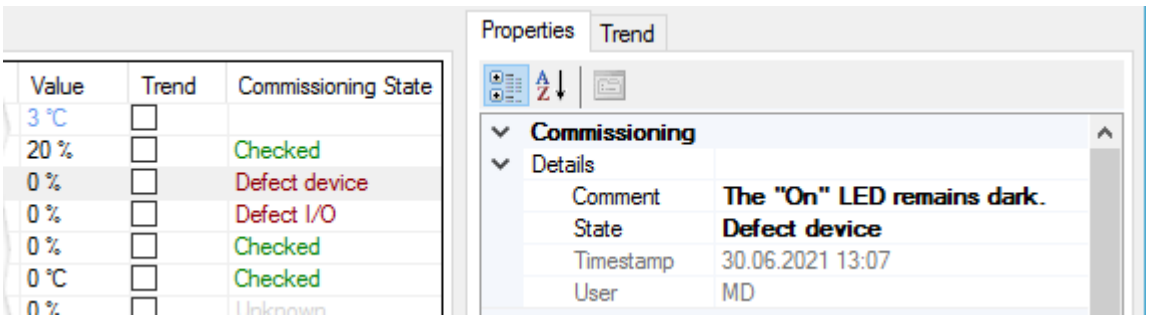


Example: Interface for entering the commissioning state of an object.

Details

Commissioning details of selected objects are displayed in the properties.

Subsequent editing is possible for some details.



Example: Commissioning details of a defective field device.

- **Online**
  The following details are stored online (per object [▶ 32]) in the controller.

- State: Current commissioning state.

- **Offline**
  The following details are stored offline (per object [▶ 32]) in the site settings.
  - Timestamp: Time of the last change to the commissioning state (if the change was made using Building Automation Site Explorer).
  - Comment: Optional comment.

Menu

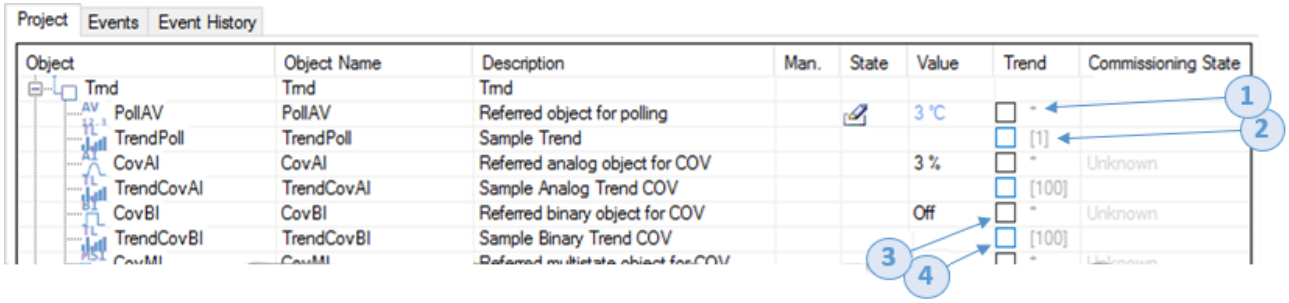
Save: Saves the commissioning details of all connected devices.

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:

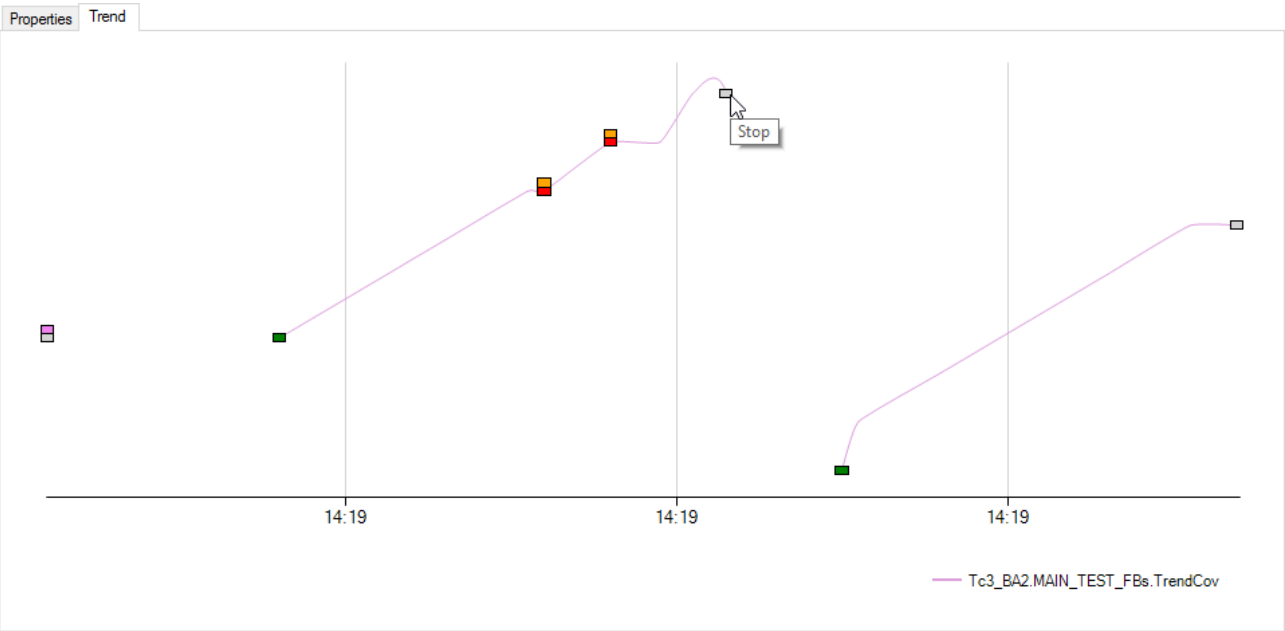
1. Indication of a referencing Trend object [▶ 226].
2. Current number of data sets of the referencing Trend object [▶ 226].
3. Start or stop recording the current value of an object [▶ 32] (**Online trend**).
4. Display or remove the log buffer of a Trend object [▶ 226] (**Offline trend**).



Example: Representation of different trends.

Offline trend

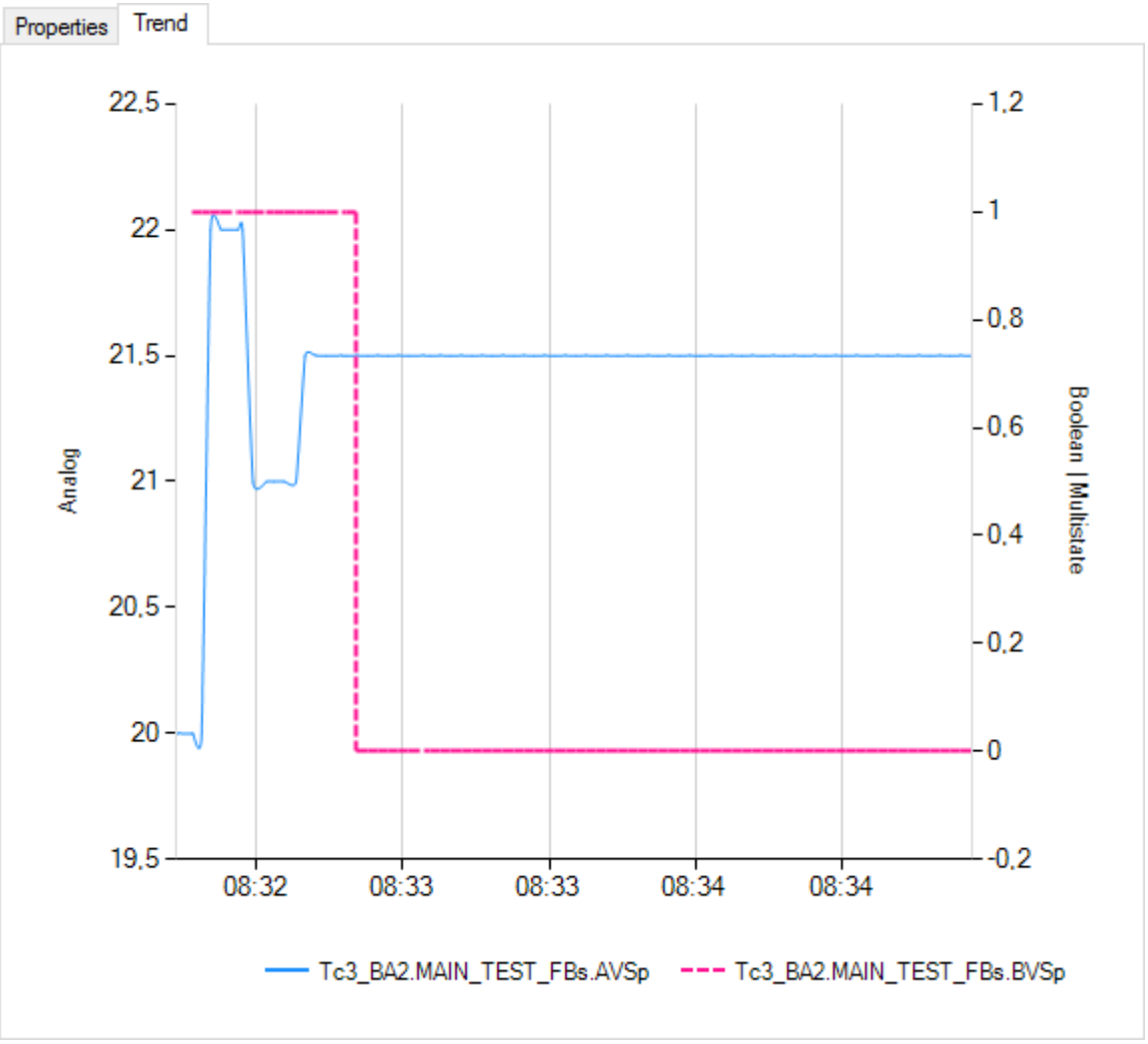
The **offline trend** reads the log buffer of a Trend object [▶ 226] and displays it in the **Trend view**.



Entries are not updated automatically!

Online trend

The **online trend** records the current value of an object [▶ 32] in the **Trend view**.

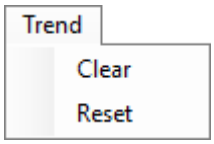


NOTICE

Loss of data

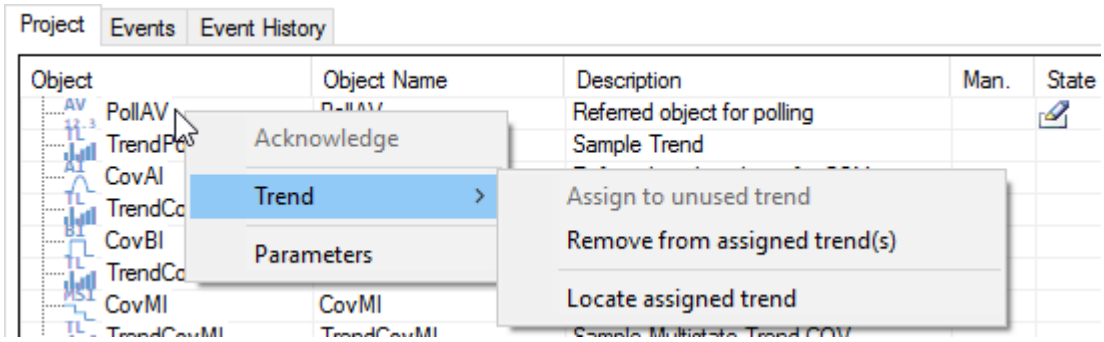
Recorded values are not saved after ending the recording or closing the trend view!

Menu



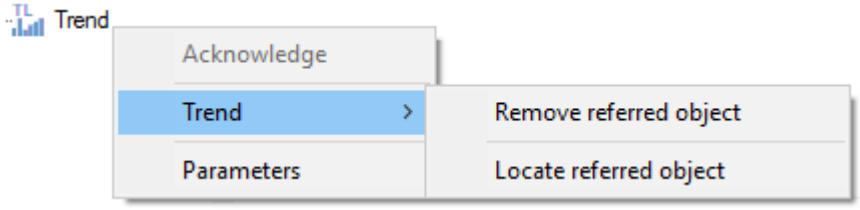
Name	Description
Cleaning	Removes all entries from the trend view (equivalent to restarting the current recordings).
Reset	Not only removes all entries from the Trend view, but also stops recording all objects [▶ 32].

Referencing



Name	Description
Assign free trend	Assigns an object [▶ 32] to the next, unused Trend object [▶ 226].
Remove reference from assigned trend object(s)	Removes the Objects [▶ 32] reference from all referencing Trend object(s) [▶ 226].
Navigate to assigned trend	Navigates to the assigned Trend object [▶ 226] (in the project view).

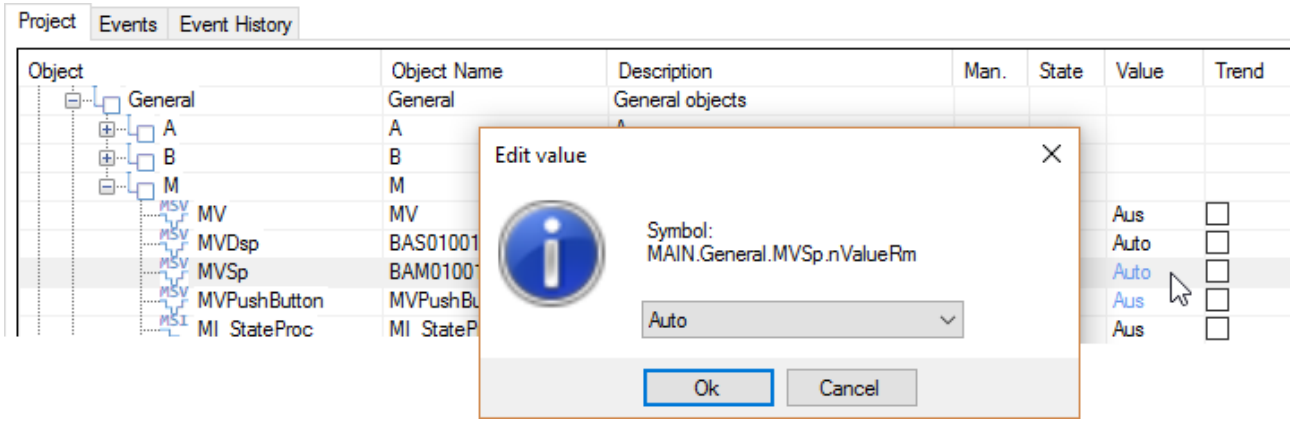
Referencing trend objects



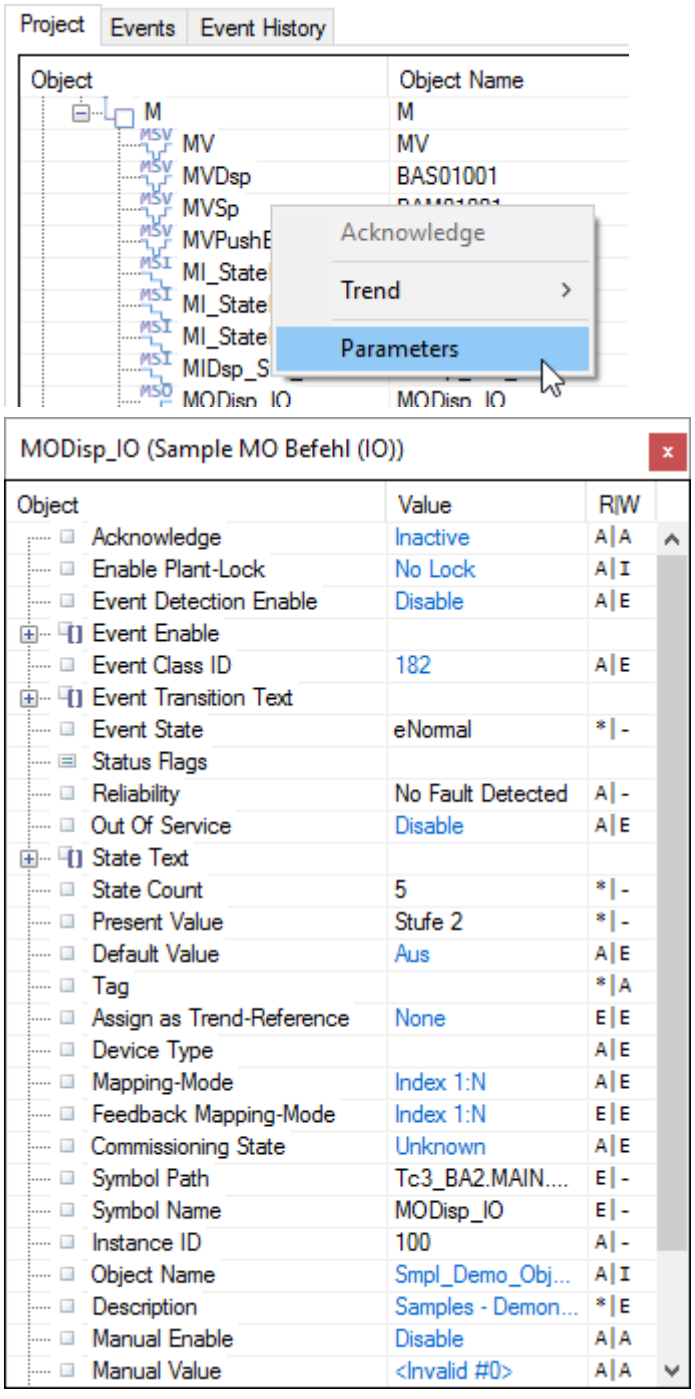
Name	Description
Remove referred object	Removes the current object [▶ 32] reference.
Locate referred object	Navigates to the assigned bject [▶ 32] (in the project view).

Editing values

Current values (if writable) can be edited directly in the project view.



Editing parameters



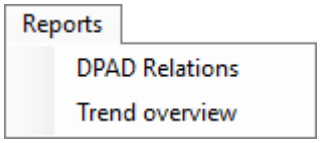
Access rights: The column **R|W** (Read, Write) lists the required access rights per parameter.

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

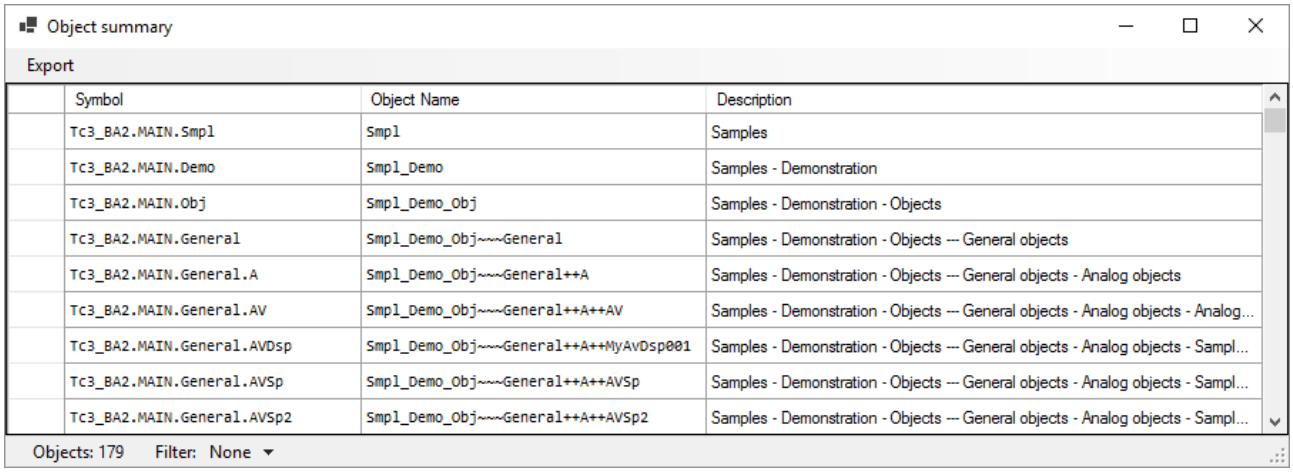


All reports can be exported to the following formats:

- *.csv

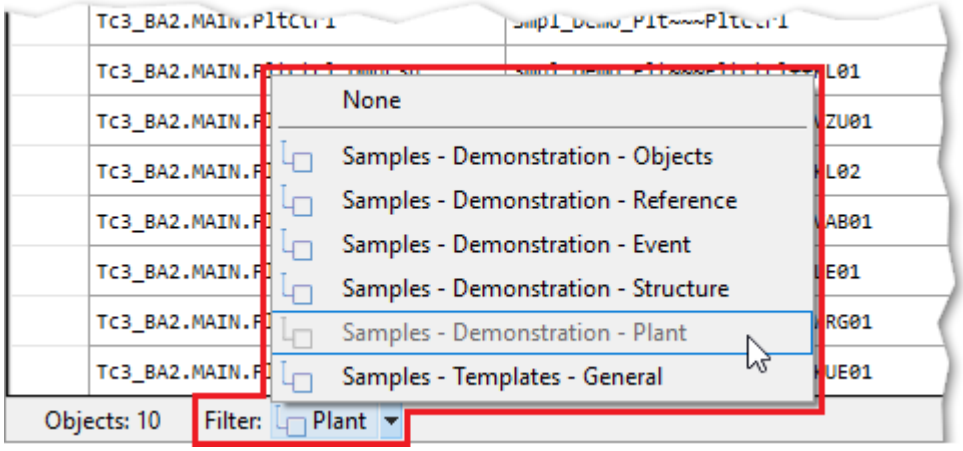
Object summary

Overview of all objects [▶ 32] located in the connected site.



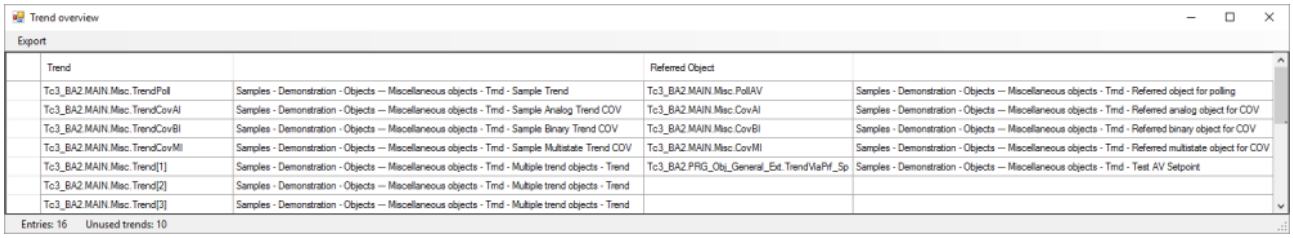
Filter

To increase the overview, the view can be filtered according to plants:



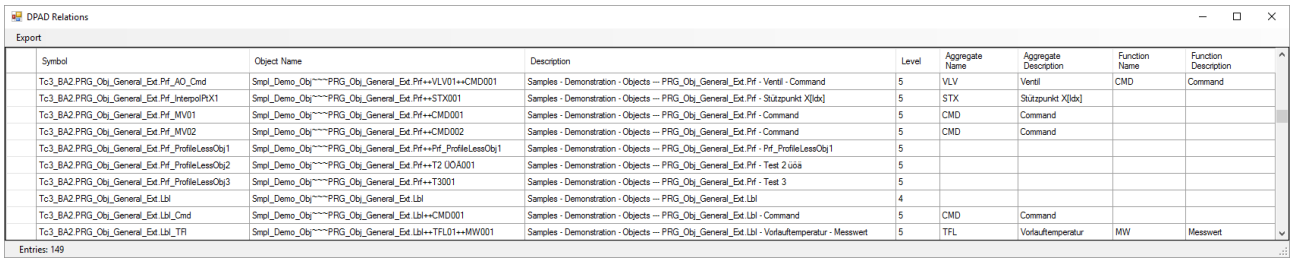
Trend overview

Overview of available trends [▶ 226] and recorded or referred objects [▶ 32].



DPAD Relations

Create a comparison to show DPAD [▶ 42] relations.



Included information per object [▶ 32]:

Symbol path

Properties

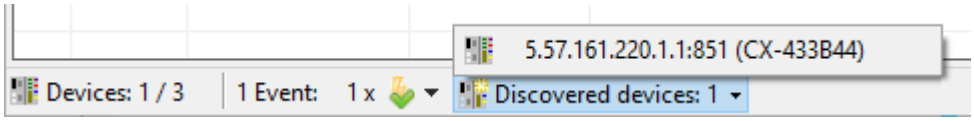
- ObjectName
- Description

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:

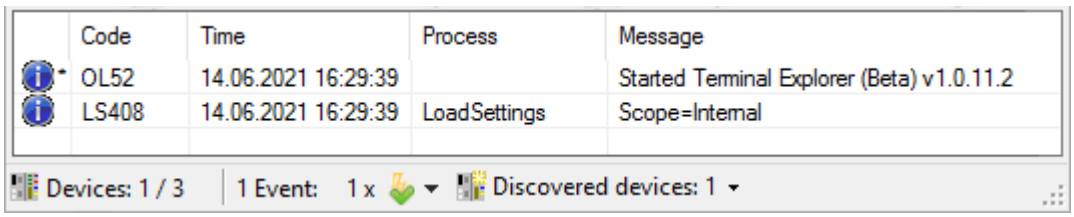


To add a discovered device to the Site configuration, just click on the corresponding menu item.

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:



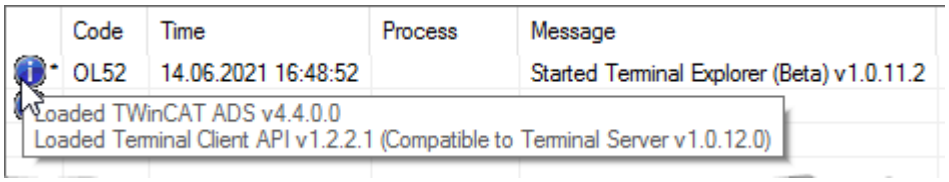
Log entries may contain additional information ( ).

These can be called by moving the mouse pointer over the *.
The following functions can be called 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:



- **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!

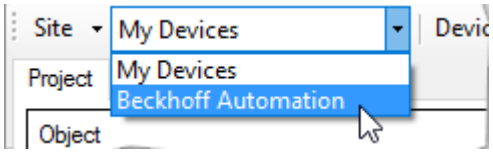
- **TwinCAT ADS**

```
Loaded TwinCAT ADS v4.4.0.0
```

Toolbar

Sites

Selection of configured Sites.



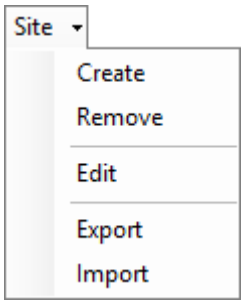
Use cases

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

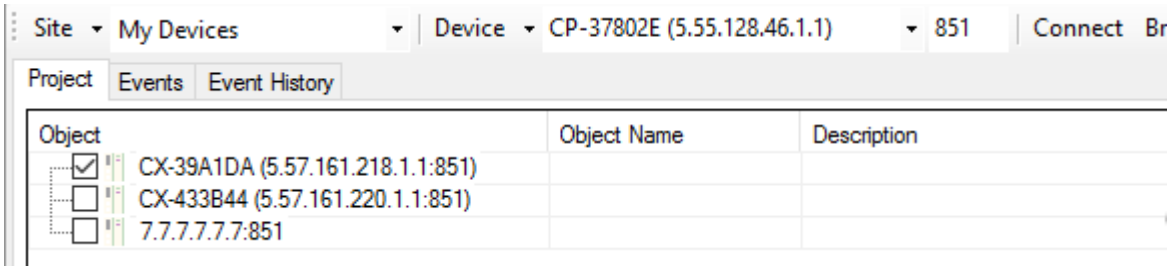
Site management



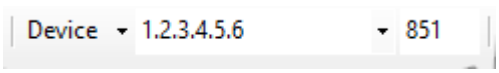
Name	Description
Create	Creates a new Site.
Remove	Removes an existing Site.
Edit	Displays the Site properties for editing.
Export	Exports a Site to a specific directory for referencing.
Import	Imports a Site for use as a reference.

Connection

When the Site connects, communication is established to all selected devices:

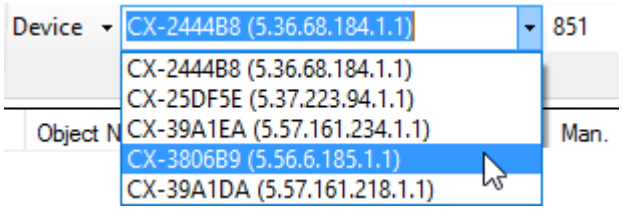


Manage devices

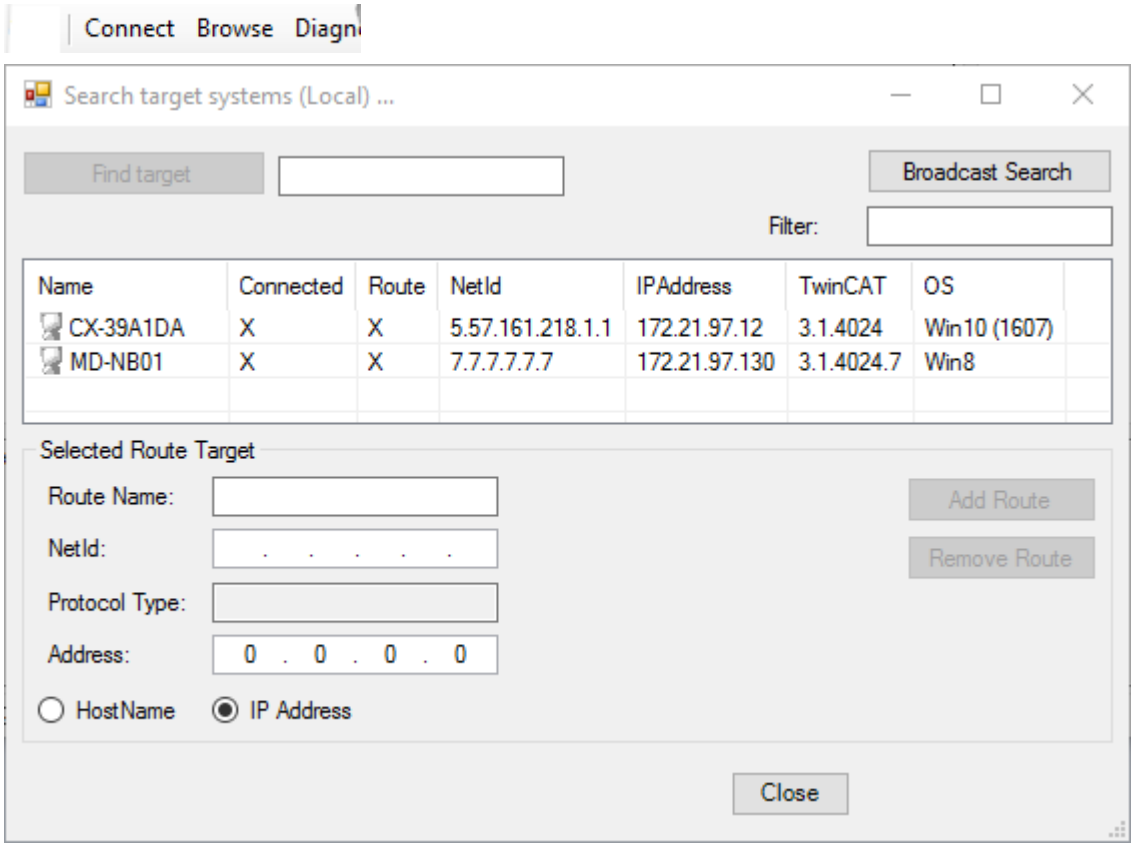
Name	Description
Add	Adds a new device to the active Site. Adding devices by manually entering the *AMS NetID* is possible. All prerequisites (such as setting up the ADS route) must be met before a connection can be established. 
Remove	Removes a selected device from the active Site configuration.

Manage routes

- **Select route:** The selection box can be used to select routes that have already been created:



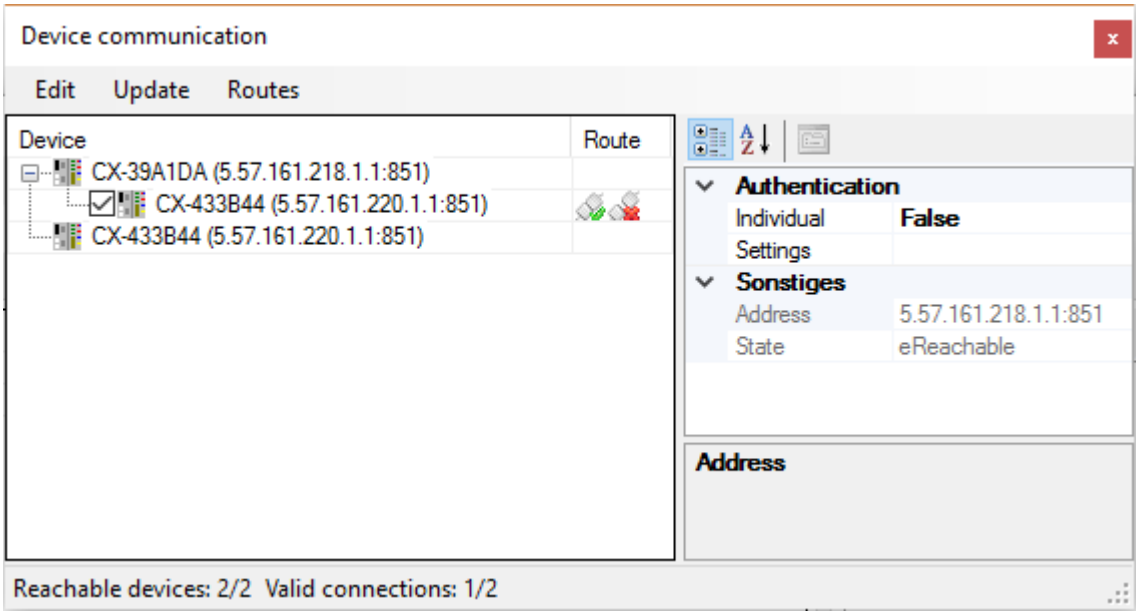
- **Create new route:** The dialog box for searching devices can be opened via the Browse button:



Diagnose

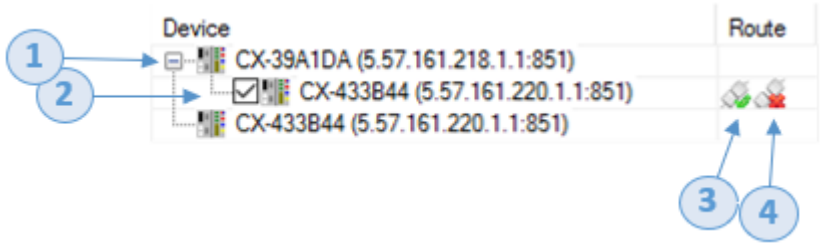
Diagnostic functions are accessible for configured devices of the active Site via the menu:

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).

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.

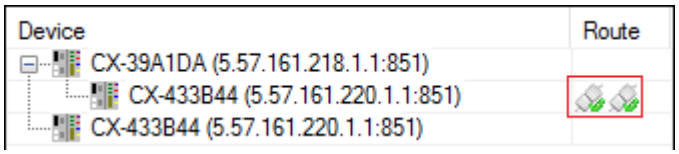
- **Routes**

Graphic	Description
	Route unknown if the state could not be determined (e.g. if the device is unreachable).
	Route missing.
	Route valid.

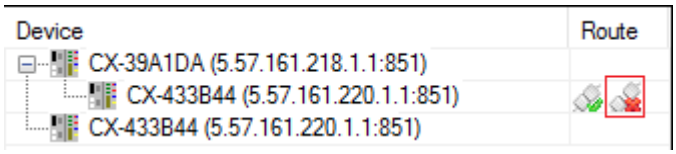
Create route

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:



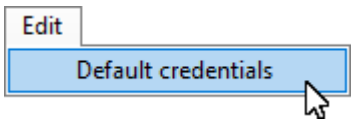
Example 2: Route valid on source device but not on target device:



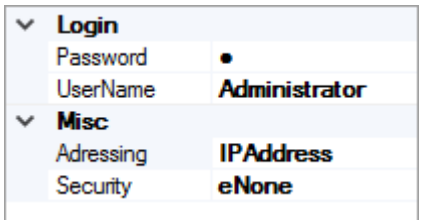
Missing routes (both of all and for selected devices) can be configured at once:

✓ **Provide credentials:** In most cases, uniform credentials are used across all devices**.**

1. Open the properties for default credentials:

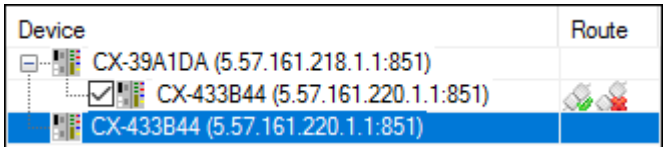


2. Make adjustments via properties:

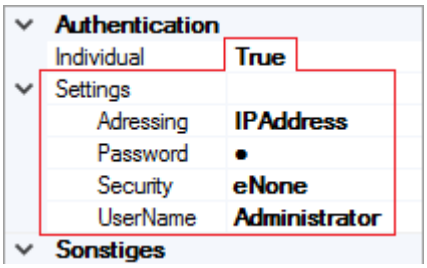


⇨ If individual credentials must be stored for different devices, the default credentials can be overwritten as follows:

3. Select device:

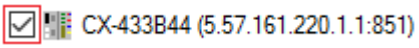


4. Activate and edit the individual credentials:

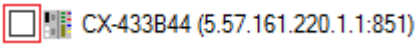


⇨ **Select devices for route configuration**

5. Devices with invalid route settings are automatically selected for configuration at the beginning:

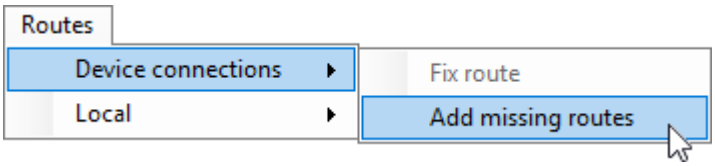


6. If devices should not be provided for configuring the route, they can be deselected:

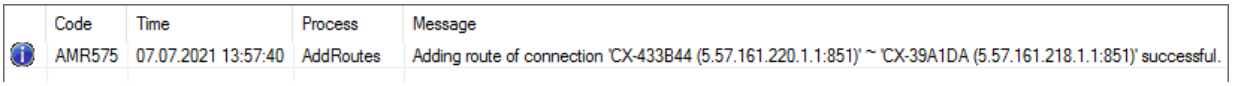


⇨ **Apply configuration:**

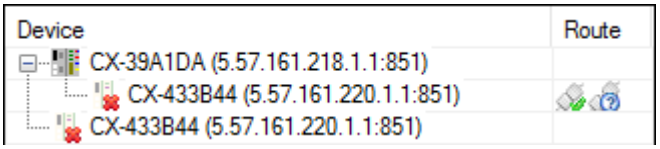
7. The route configuration of the selected devices can be rolled out via the menu:



⇨ All configuration operations are logged in the log:



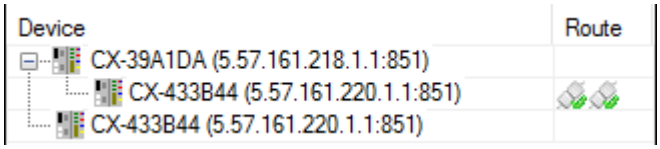
⇨ For unavailable devices (e.g. not reachable or route not configured) neither route states can be displayed nor route configurations can be adopted:



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).

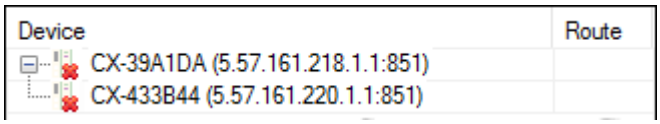


Example: Display of a valid connection due to routes set up on both sides.

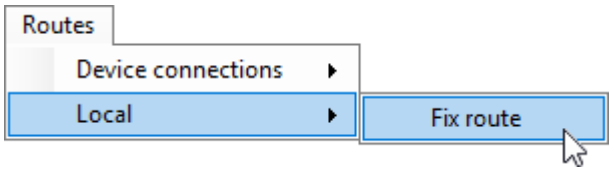
A (highlighted) connection can be repaired via the menu.
This reconfigures the routes on both devices.

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).



Routes to unreachable devices can be fixed via the menu:



7.2 Symbol Explorer

7.2.1 Introduction

With the Symbol Explorer you can access controllers online via the ADS communication interface:

- You can read the variables declared as persistent.
- 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.

7.2.2 Definitions

7.2.2.1 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, sub-variables 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.

7.2.2.2 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.

7.2.3 User interface

The interface is divided into various windows with different outputs for using the Symbol Explorer.

7.2.3.1 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.

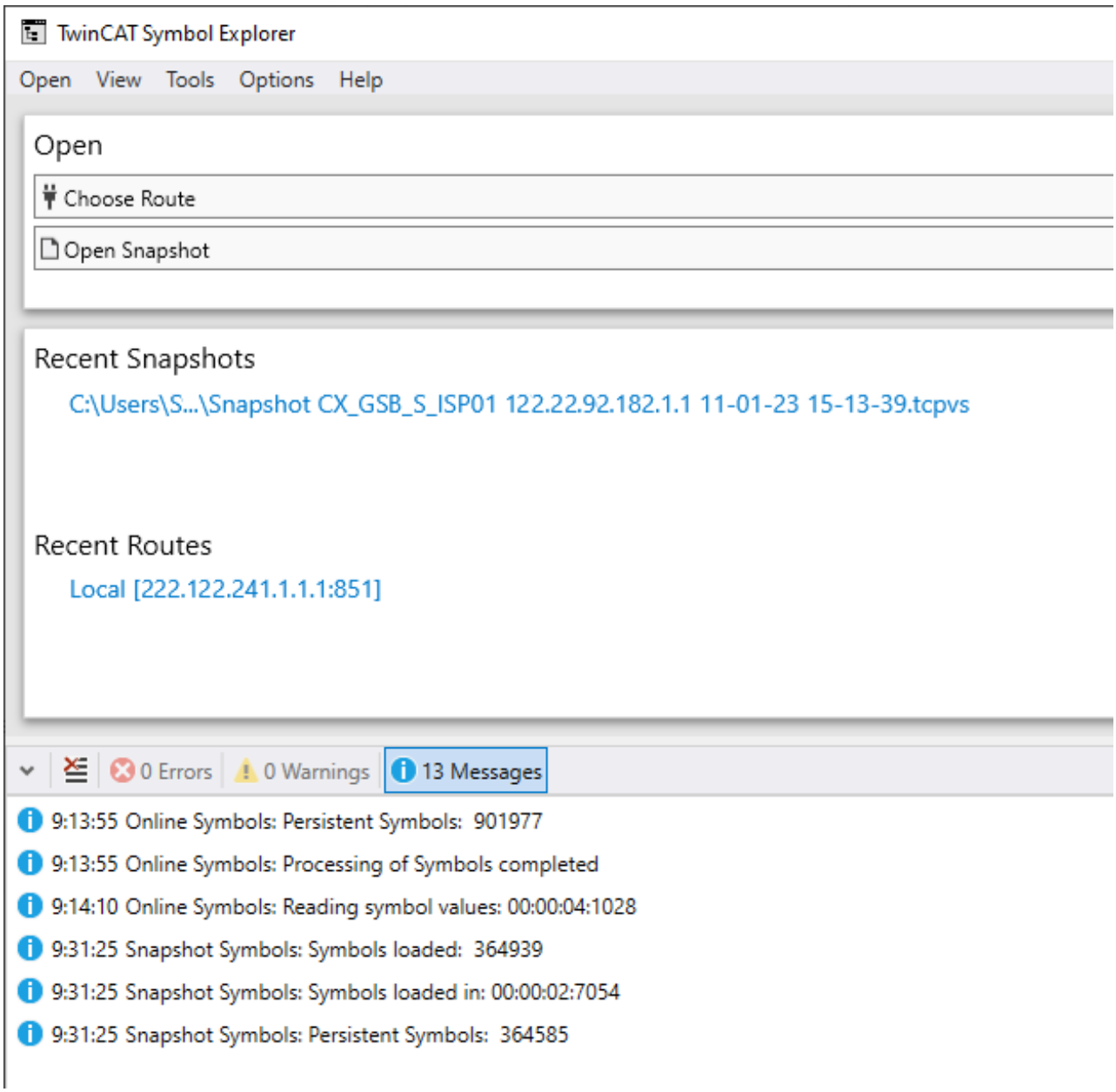
Help

Command	Description
About TwinCAT Symbol Explorer	Version specification

7.2.3.2 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

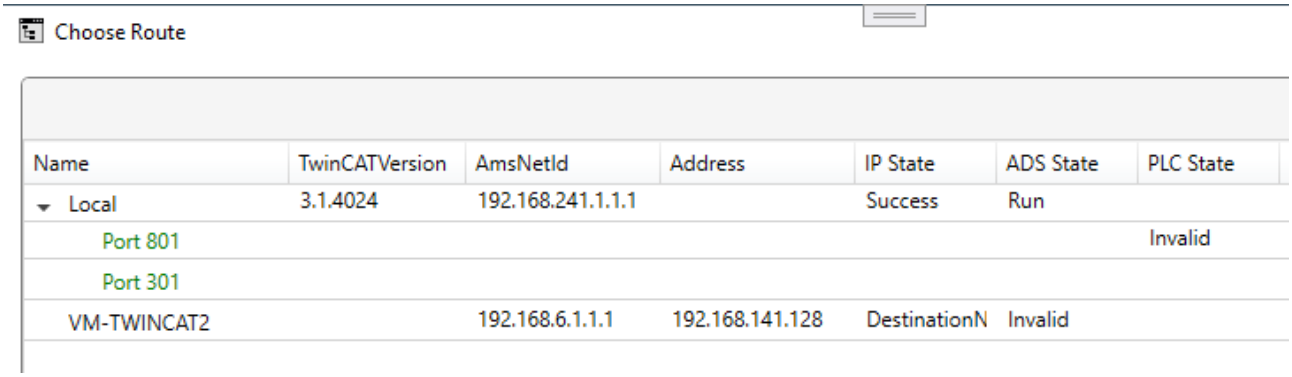
Command	Description
Choose Route	Connect to a route.
Open snapshot	Open a snapshot

Recent

Command	Description
Recent snapshots	List of frequently used snapshots.
Recent routes	List of frequently used routes.

7.2.3.3 Choose Route

The "Choose Route" dialog provides a quick and easy view of the routes that are registered on the system. The ports that can be reached by the Symbol Explorer are displayed for each route. These are the pre-set ports for accessing the PLC (TwinCAT2 / TwinCAT3) or specially configured ports for accessing the EtherCAT master under the Process Image, for example. The ports can be configured via the Options dialog.



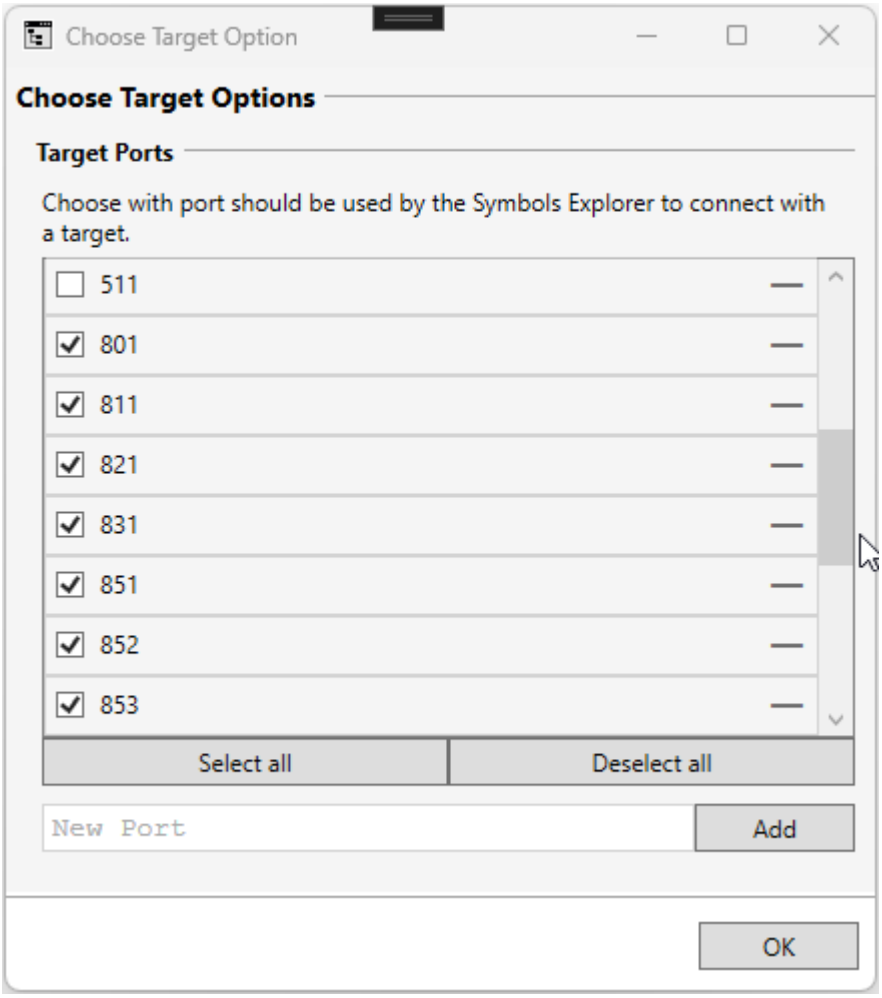
Toolbar

Symbol	Command	Description
	Options	Open Choose Route dialog Options
	Refresh Route	Refresh the status of the routes

Options dialog

Here you can choose which ports should be used by the Symbol Explorer to connect to a Traget.

You can choose from the default ports or add a specific port.



Default port numbers

Description	Ports
MOTION	500, 501, 511
Tc2 PLC	801, 811, 821, 831
Tc3 PLC	851, 852, 853, 854, 855
IO IMAGE	27905, 27906, 27907, 27908, 27909, 27910
TC3 TASK	350, 351, 352, 353, 354, 355

7.2.3.4 Symbol Entry Point

The Symbol Entry Point describes the entry point(s) that the Symbol Explorer should take in the symbolic structure of a target. The required symbols can be selected. Various filters are also available that can reduce the number of symbols to be loaded.

Symbol Type Selection

This is a coarse filter that is applied to a symbol area (e.g. 'Persistent'). If the filter is selected, only the symbol and the parent symbol marked in the PLC with the respective scope (e.g. 'Persistent') are displayed.

Symbol Datatypes

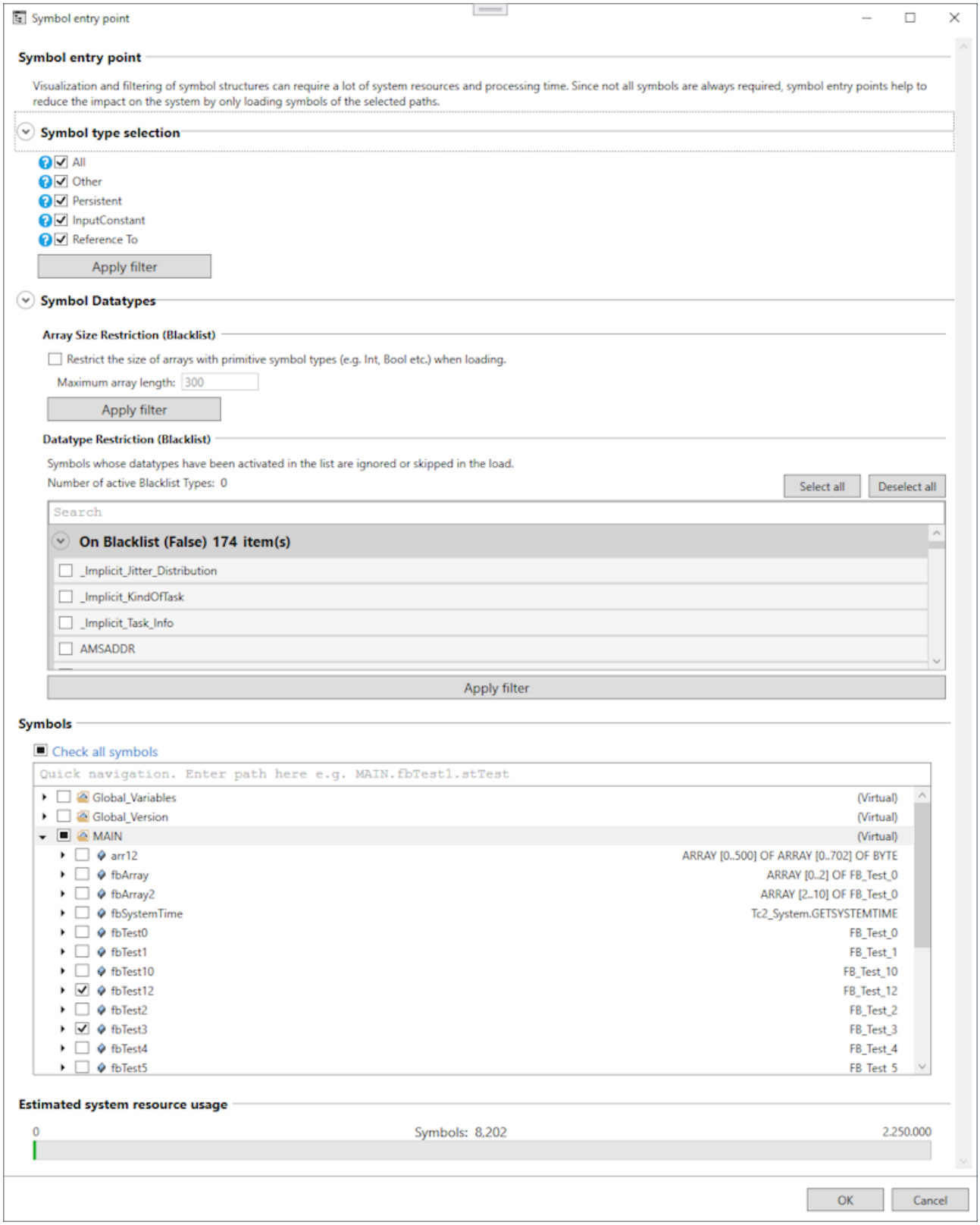
A filter that is applied to the data type or a special feature of the data type.

Array size limitation

If the filter is active, all arrays with a primitive data type that are greater than the value for 'Maximum array length' are excluded.

Datatype Blacklist

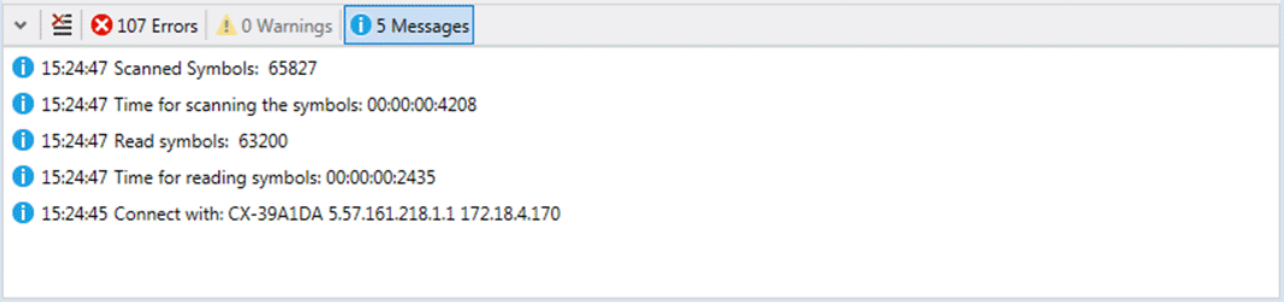
Symbols whose data type is in the blacklist are excluded.



7.2.3.5 Output window

The Output window displays information, errors and warnings at runtime.

The functions of the toolbar are described in the following table.

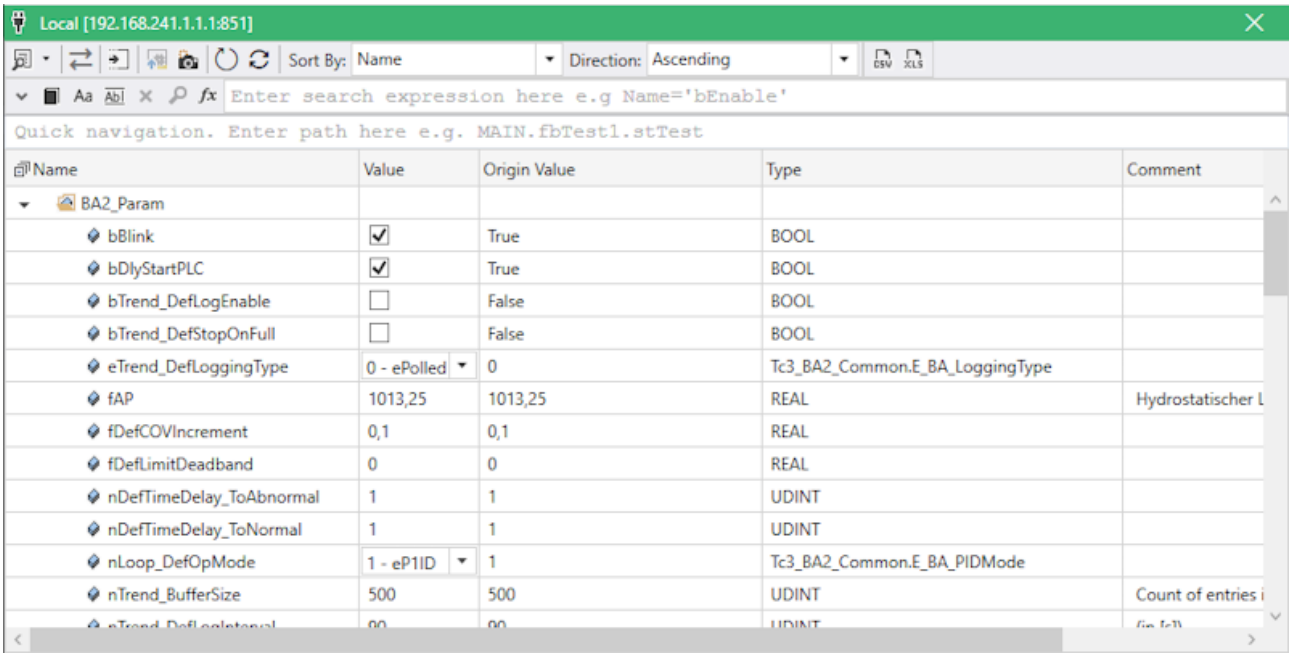


Toolbar

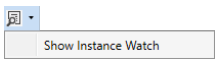
Symbol	Command	Description
	Expand / Collapse	Expands and collapses the Output window
	Clear messages	Clears the messages
107 Errors	Show / Hide errors	Shows and hides the error messages
0 Warnings	Show / Hide warnings	Shows and hides the warnings
5 Messages	Show / Hide messages	Shows and hides the messages

7.2.3.6 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	Symbol instances view.
	Synchronize symbols	Show dialog for synchronizing symbols.
	Sort by	Sorting of symbols by e.g. name or size.
	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
	Export / Import CSV	Export or import symbols as CSV file
	Export symbols as Excel	Export symbols as an Excel spreadsheet.
	Close window	Close window

Exclusions when refreshing

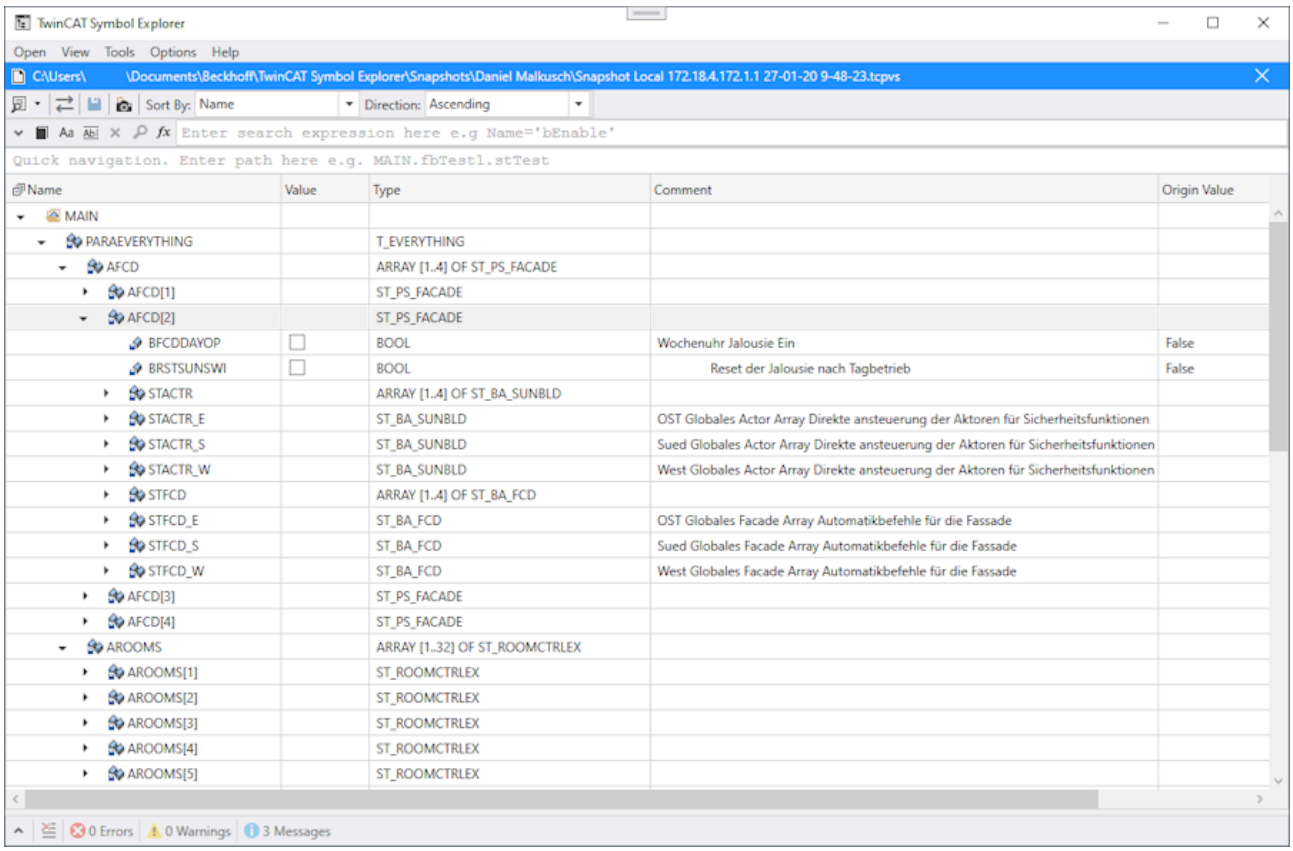
Symbols that have been edited via the Symbol Explorer and symbols that are currently being edited are excluded from refreshing.

Notes on ADS communication

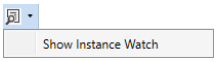
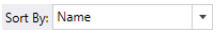
The Symbol Explorer uses the ADS protocol for the online communication. Note that ADS is only a transport layer; however, side effects can occur. Read these requirements and note the restrictions: ADS itself is only the transport layer, the requested ADS device must support the ADS command. When the 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.

7.2.3.7 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.

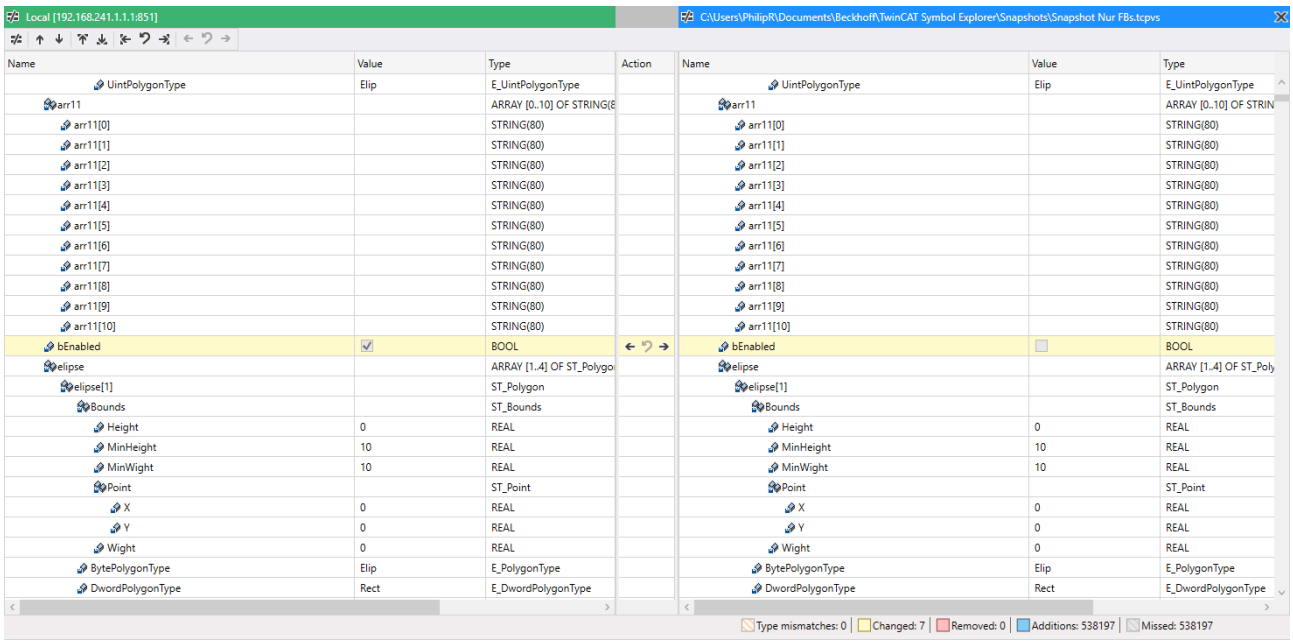


Toolbar

Symbol	Command	Description
	Show symbols as list	Shows symbols as a flat list.
	Drop down box	Extended symbol lists.
	Show instance watch	Symbol instances view.
	Synchronize symbols	Show dialog for synchronizing symbols.
	Sort by	Sorting of symbols by e.g. name or size.
	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

7.2.3.8 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.



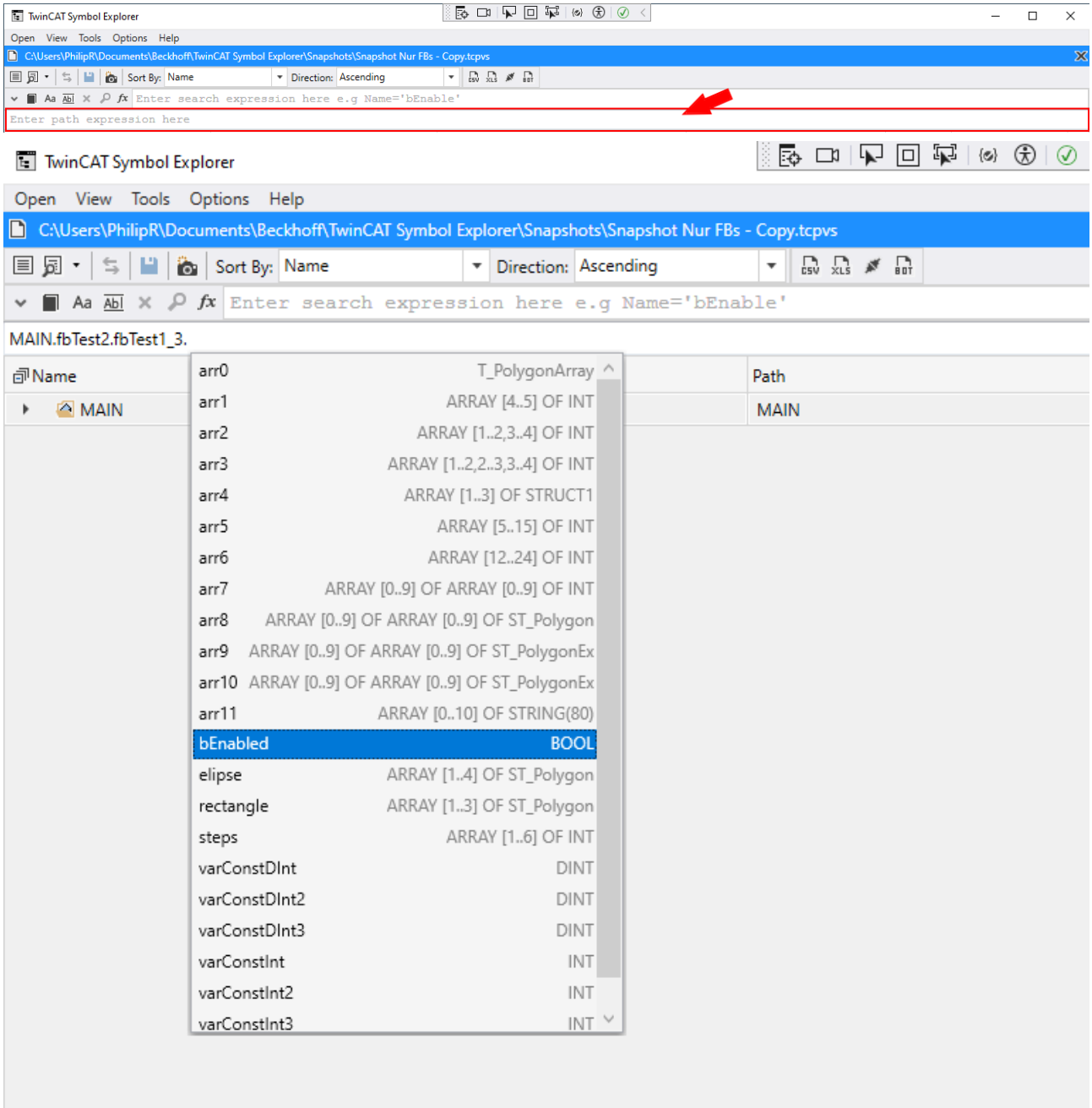
The Symbol Compare window displays the following differences:

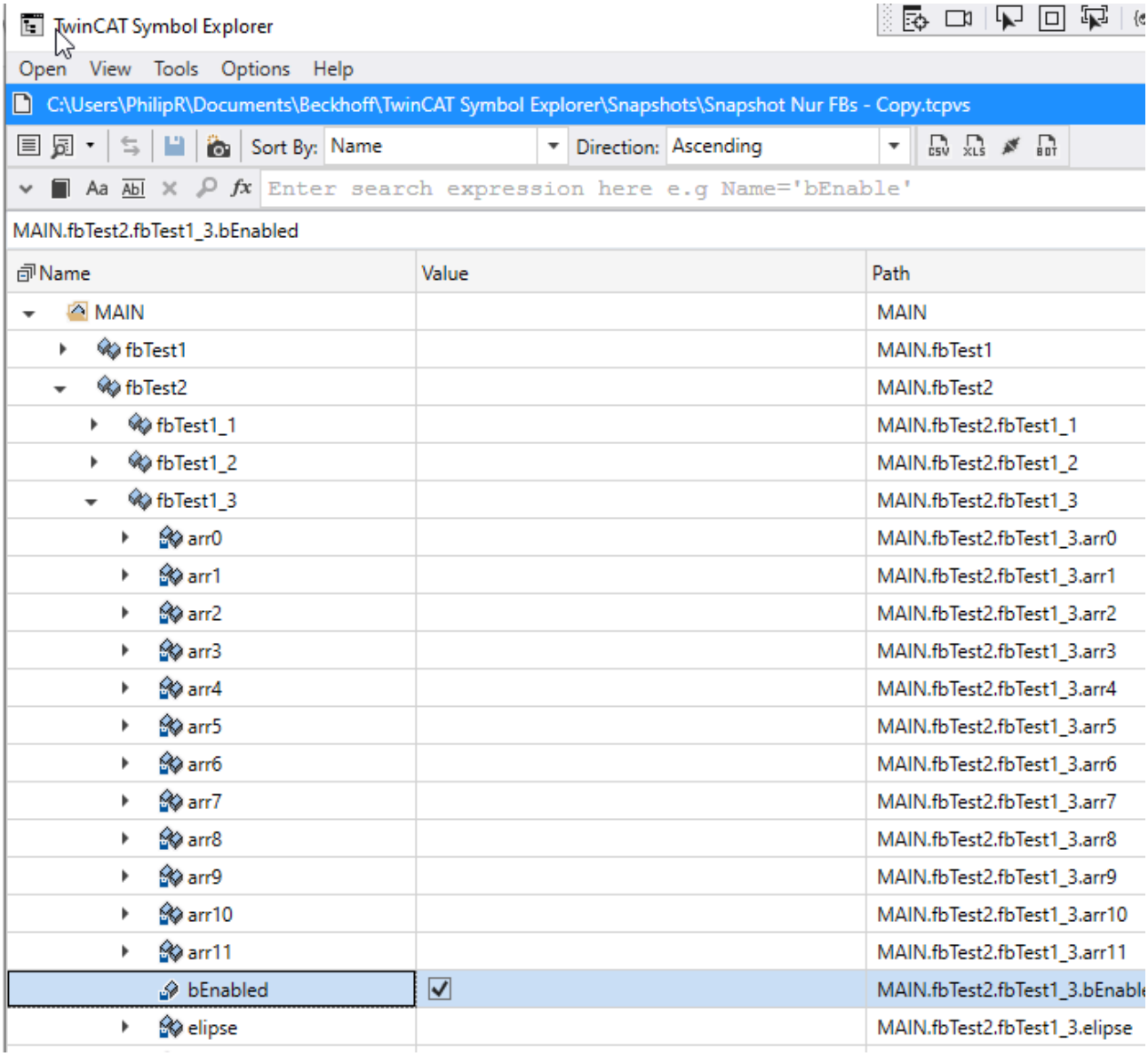
- Differences from symbol to symbol (value)
- Removed symbols
- Added symbols
- Missing symbols
- Types differences

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

7.2.3.9 Symbol Quick Navigation Bar





7.2.3.10 Symbol list

Copying and pasting in the Symbol Explorer is list cell dependent!
This means that if a cell is selected and copied in the Symbol Explorer, the content of the cell is 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

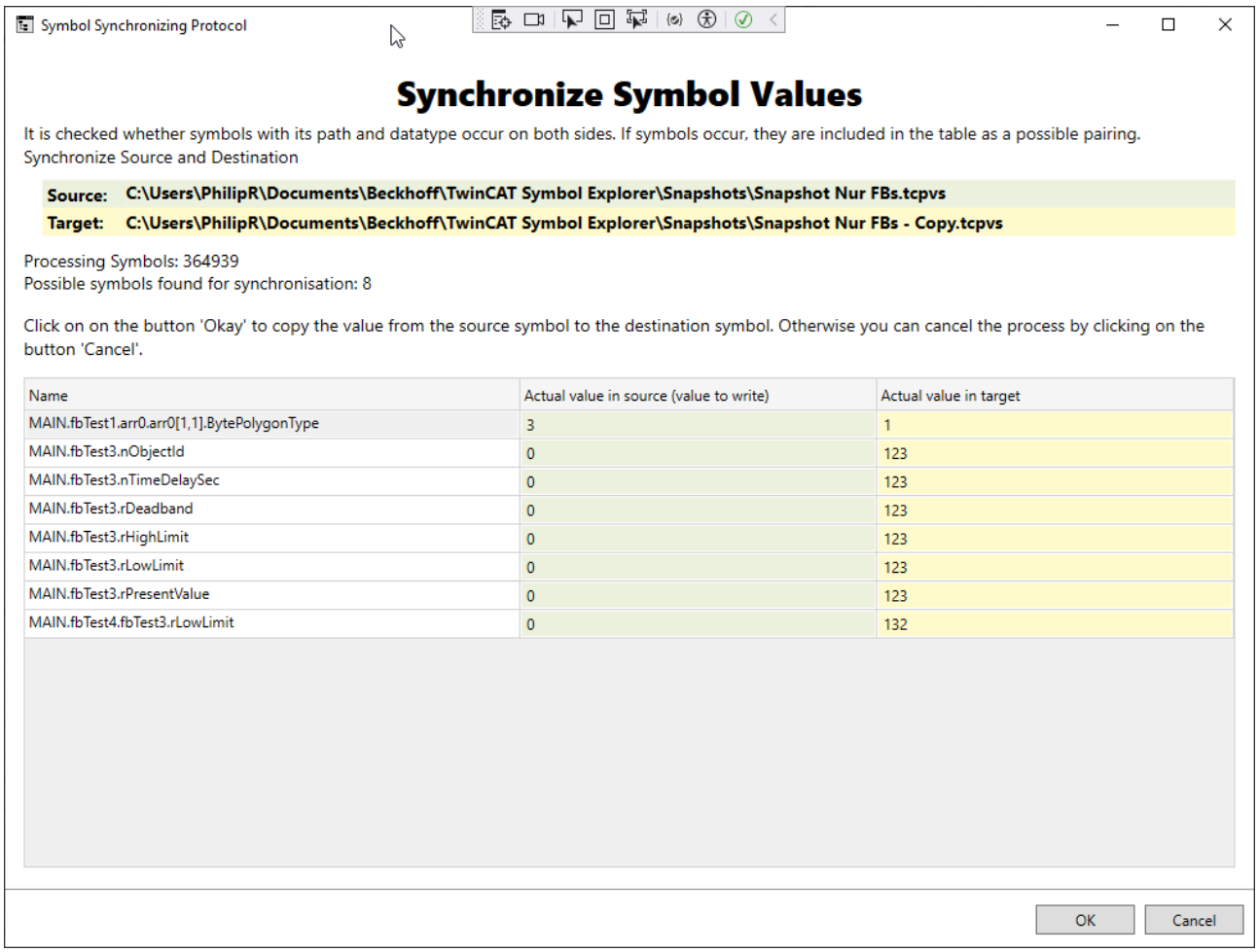
Shortcut	Description
Ctrl + left-click 	Expands or collapses the symbol and its sub-symbols.
Ctrl + left-click	Selective multi-selection on a row.
Ctrl + C	Copies the selected symbol (and all sub-symbols).
Ctrl + V	Pastes the copied symbol values into the selected symbol (and the sub-symbols).
Ctrl + 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

7.2.3.11 Synchronize Symbol window

This window can be used to compare different targets (Online <-> Online or Snapshort <-> Online) or to load a snapshot (recipe or value snapshot) back onto a target. The symbols of the targets are compared with each other. If a symbol exists on both sides, its value is compared. If there is a difference at the point, the symbol is noted as a difference in the list (see below in the image). At the end, the different symbols can be synchronized between the targets.



7.2.4 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.

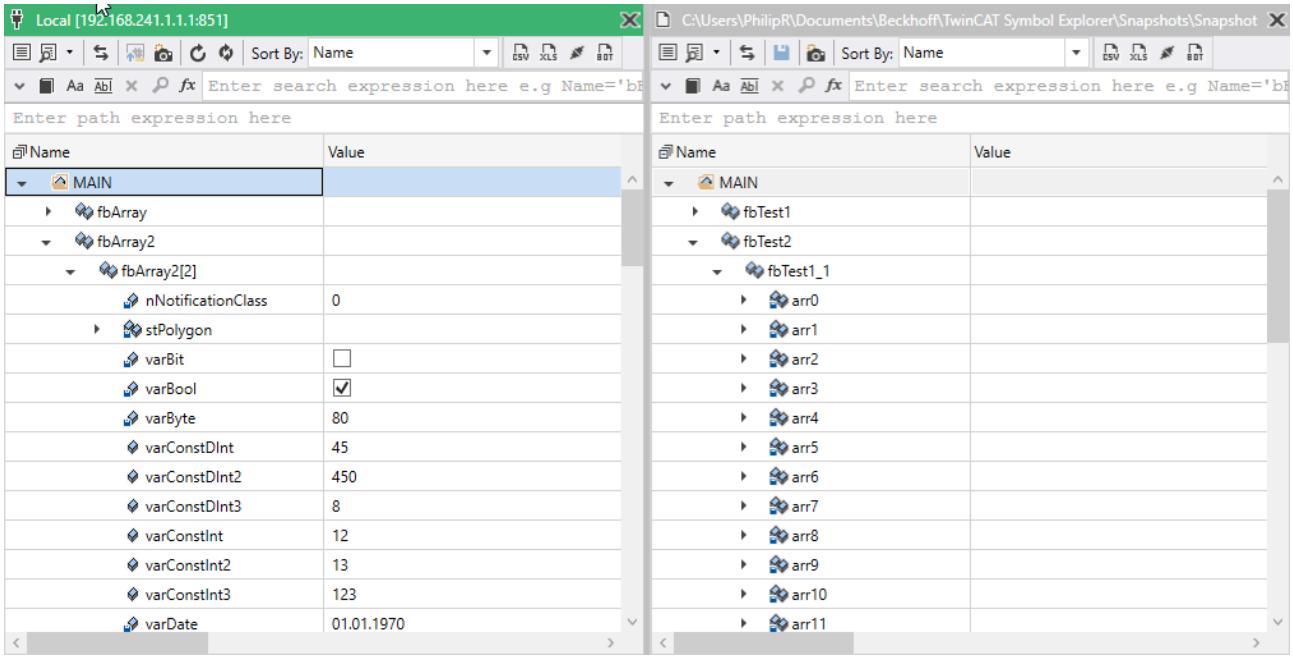
Loading a snapshot

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

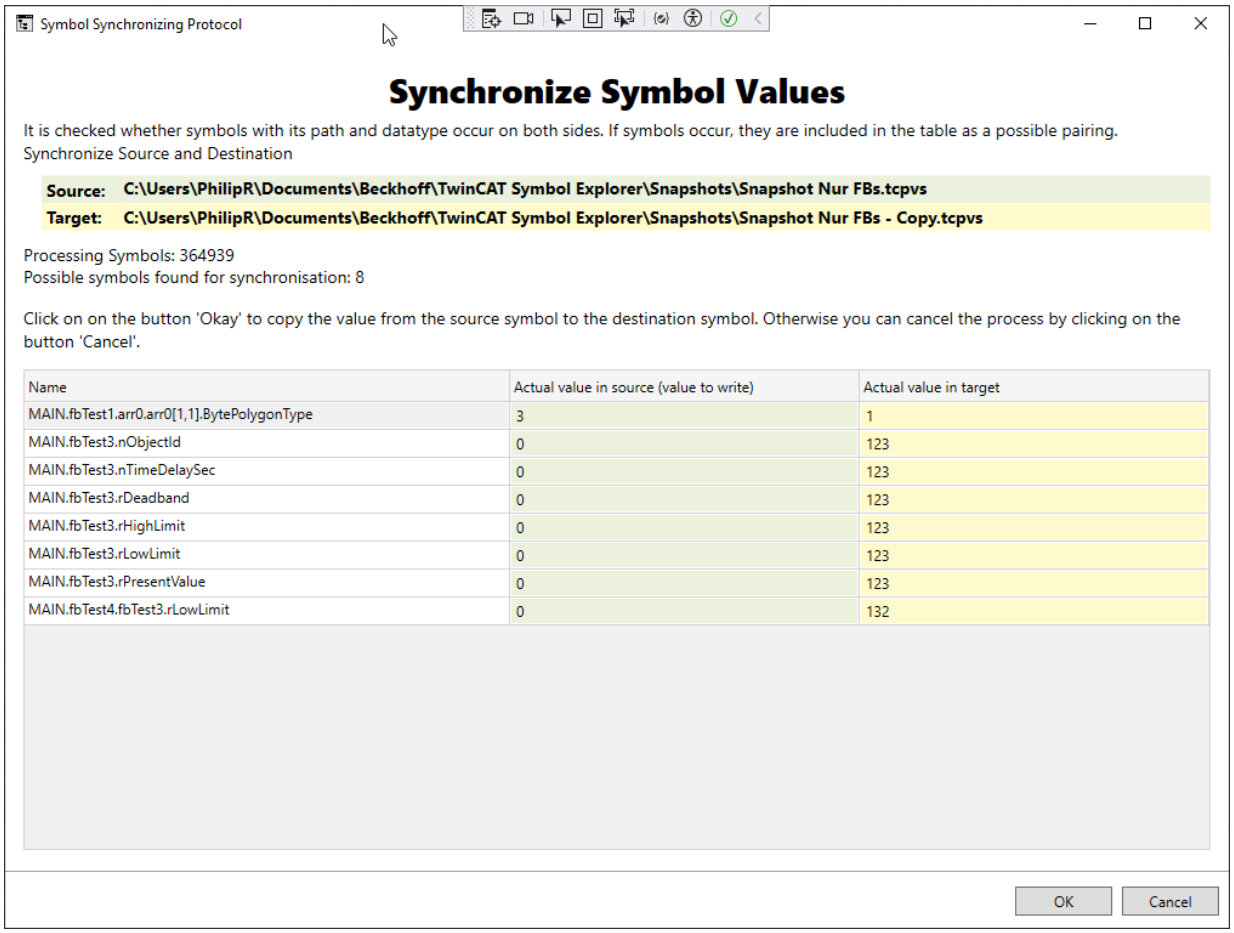
The Symbol Explorer can display two symbol lists in parallel. It is thus possible, for example, to place the current symbols (online) and symbols from a snapshot next to one another and to edit symbol values between these two lists. Simply connect to a controller and then open a snapshot. A different order of opening/connecting is also possible.



Copying a snapshot to a controller

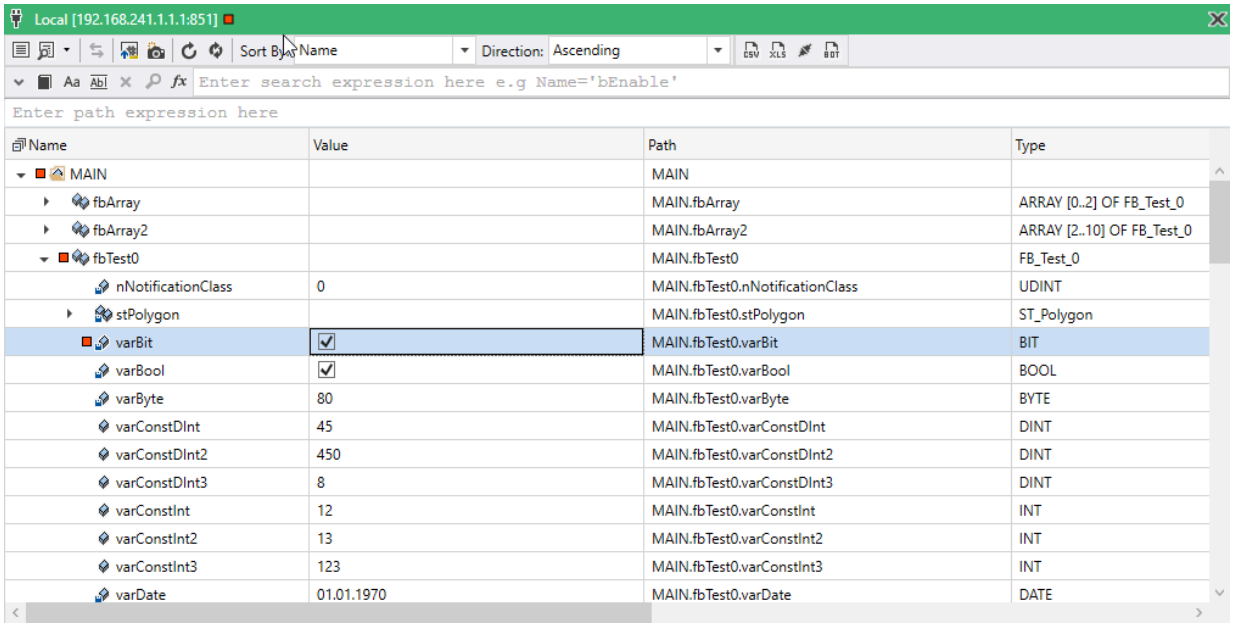
1. Open and connect the Symbol Explorer to a controller.
2. Then open a snapshot. You can now see the Online and Snapshot windows side by side.

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.

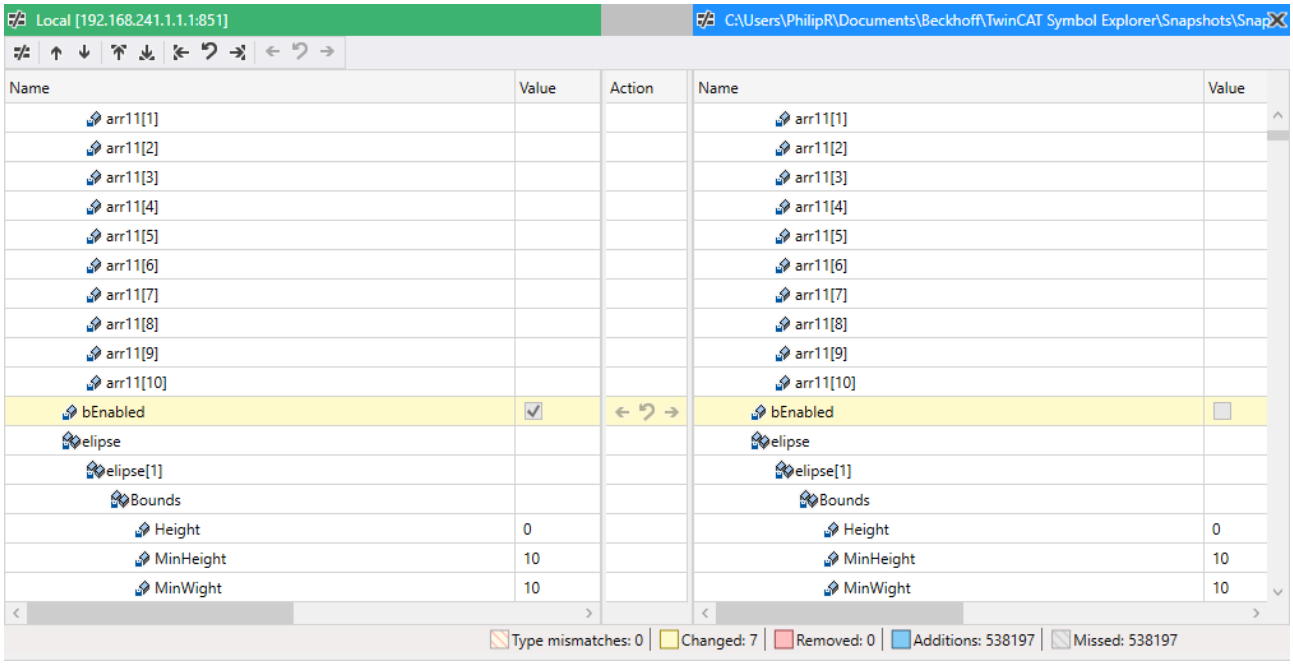
⇨ The comparison window then opens.

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

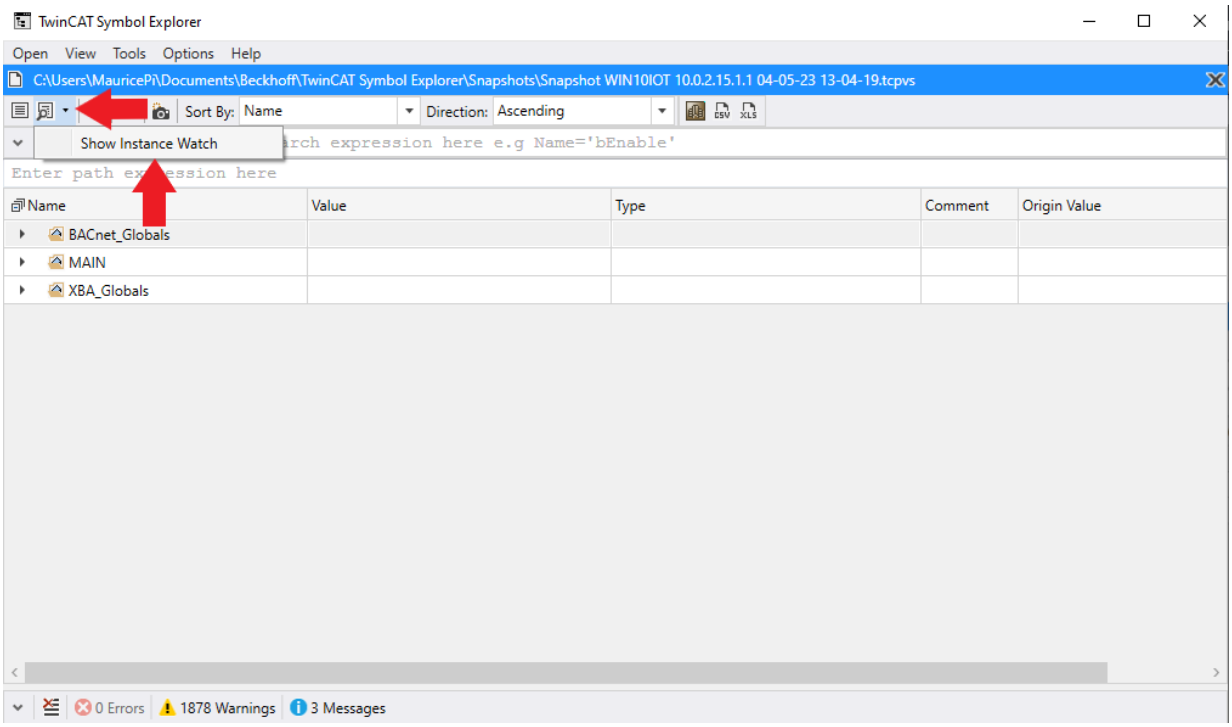
The synchronization functions can be applied implicitly to differences on the basis of a row selection. For example, if you click a row with a difference and then the right arrow button on the central toolbar, the symbol value is copied from the left into the symbol on the right. This allows the easy merging of many small differences.



7.2.4.1 Open Instance Watch Template

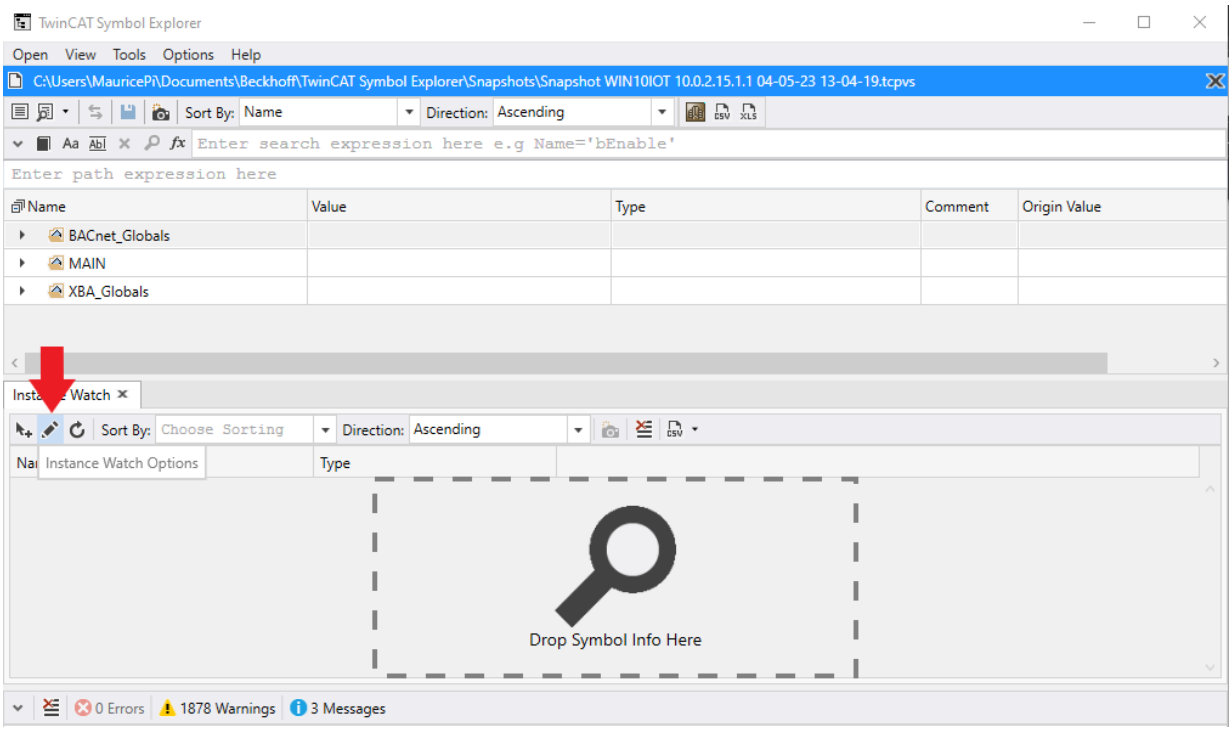
✓ To load an Instance Watch template, the Symbol Explorer must first be started.

1. Open the Instance Watch

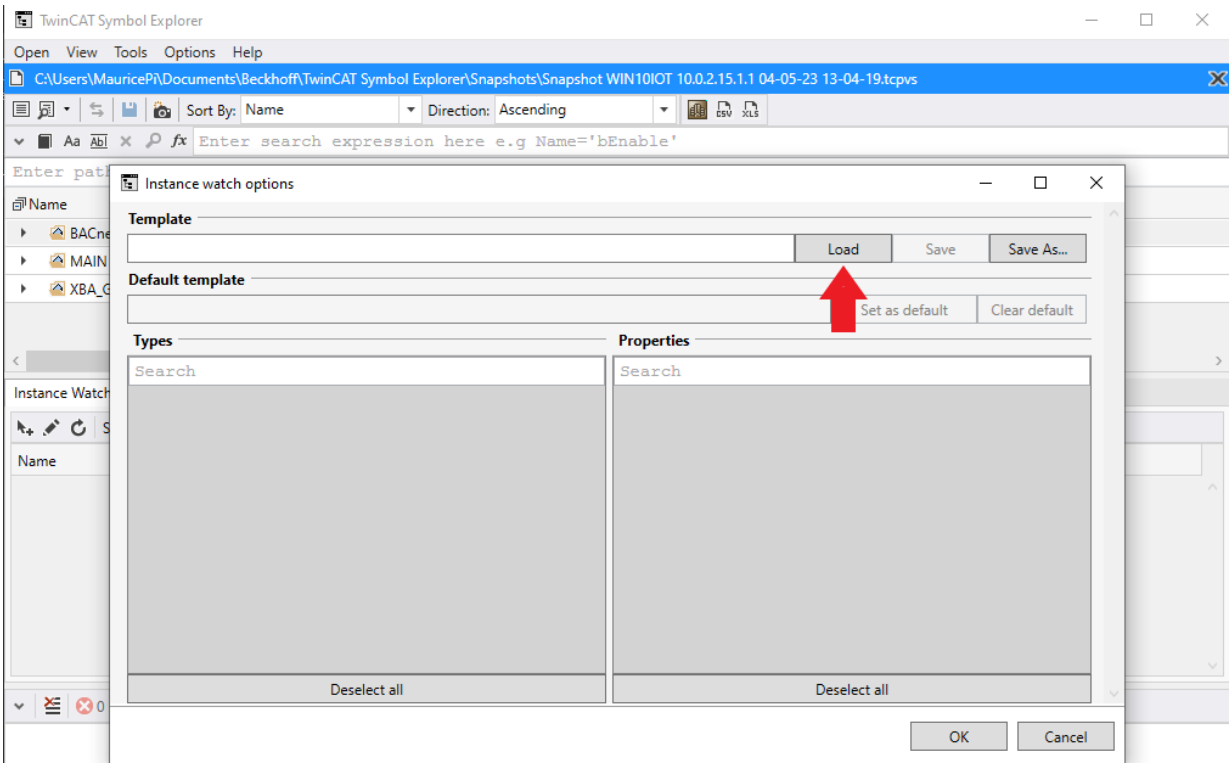


⇨ If no default template has been defined, the Instance Watch is empty.

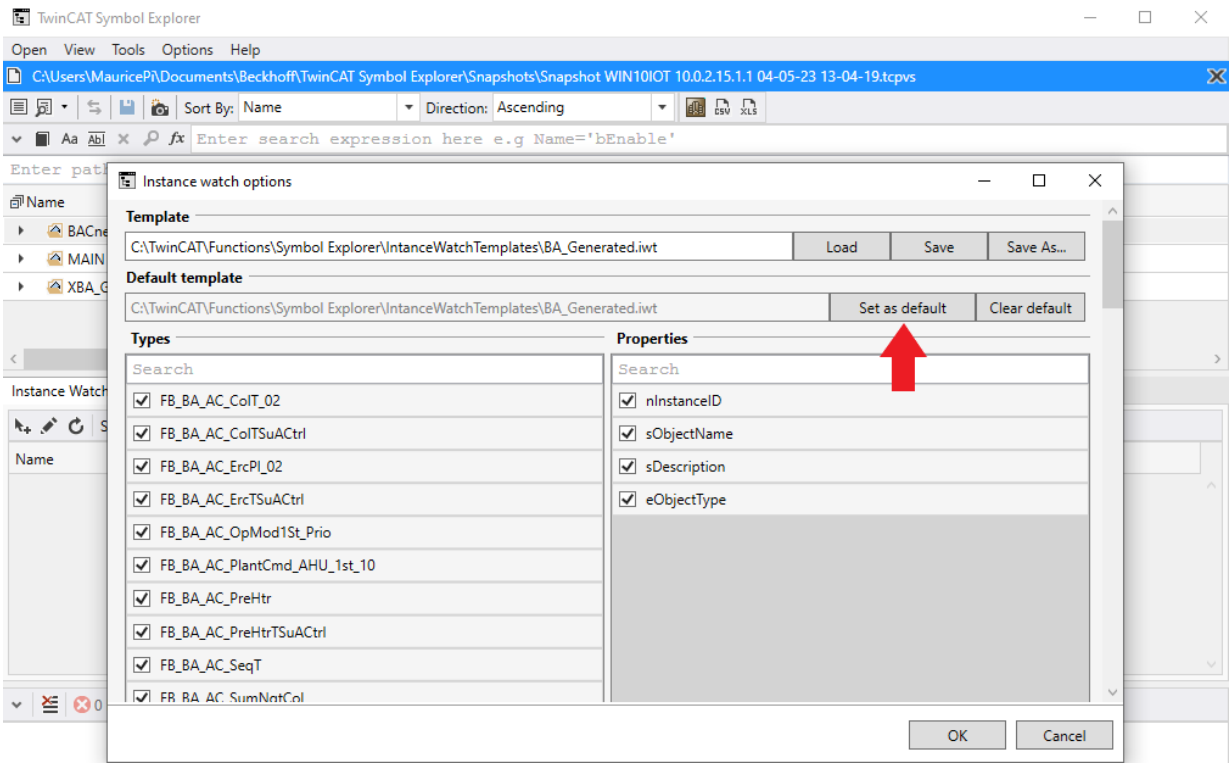
2. To load a template press the button **Instance Watch Options (pen).**



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**.



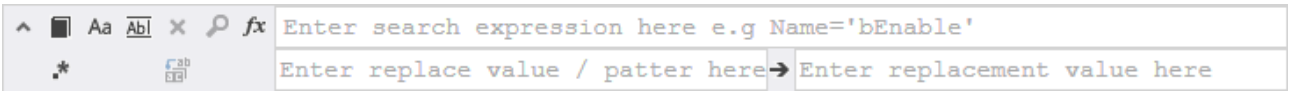
5. Confirm the **Instance watch options** dialog with **OK.**

⇨ The instance watch is loaded with the template.

7.2.5 Filter, search and replace

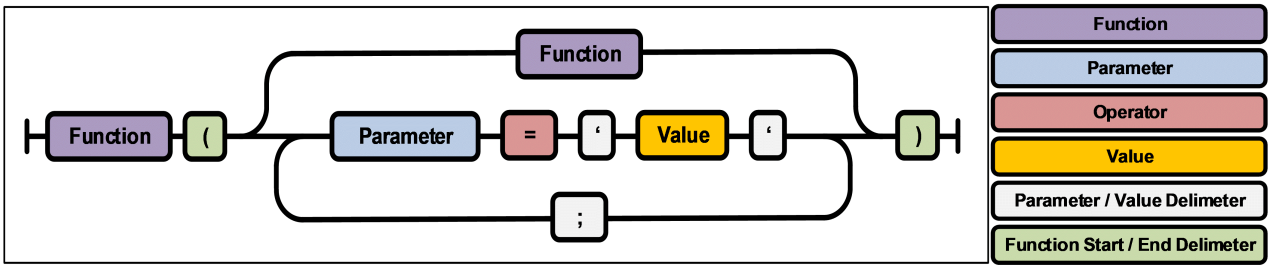
By means of the individual filtering of the symbols, it is possible to find the desired symbols faster. You can filter for the name, type or comment of a symbol.

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
	Enter search example	Enter search expression
	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
	Search for value to replace	Search for value to replace
	Replacement value	Replacement value

7.2.5.1 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 '*sSobjectName*' 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.

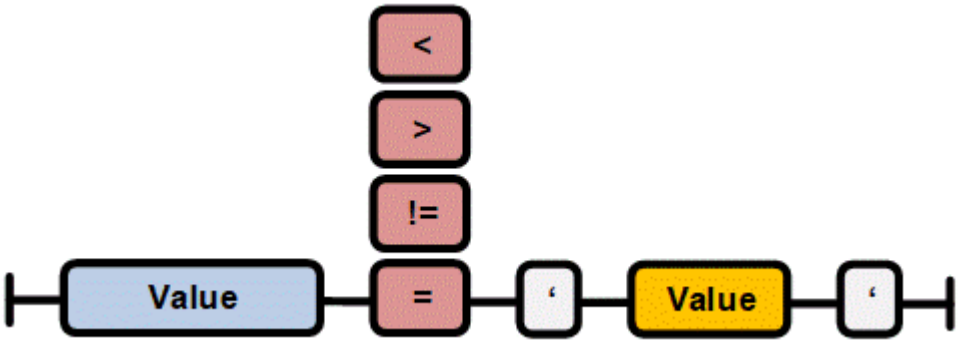
The CHILD function can only be used in an AND function or in an OR function.

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.



Samples:

```
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'
```

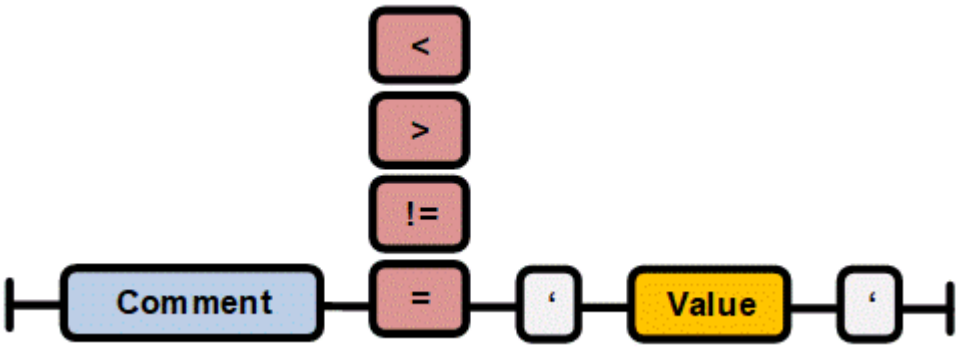
Search for symbols whose value is not '100'.

```
Value = ''
```

Search for symbols whose value is empty ' '.

Comment Parameter

The function searches for symbols whose value satisfies a condition with a certain comment.



Samples:

```
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'
```

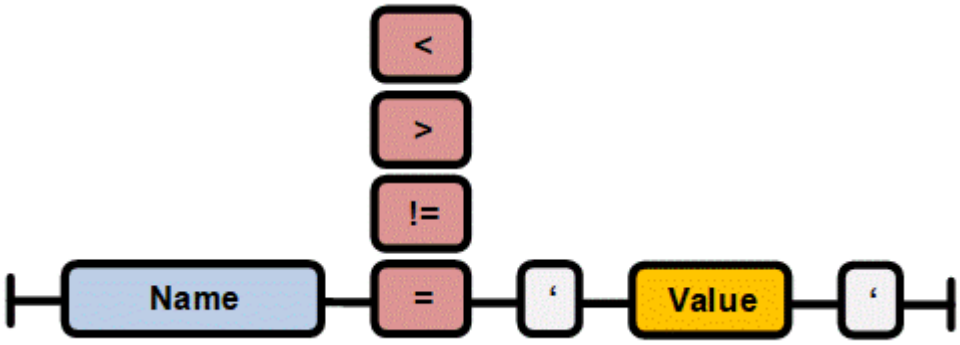
Search for symbols whose comment begins with 'Signal'.

```
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.



Samples:

```
Name = 'bEnable'
```

Search for symbols whose name is 'bEnable'.

```
Name != 'bEnable'
```

Search for symbols whose name is not 'bEnable'.

```
Name > 'bEn'
```

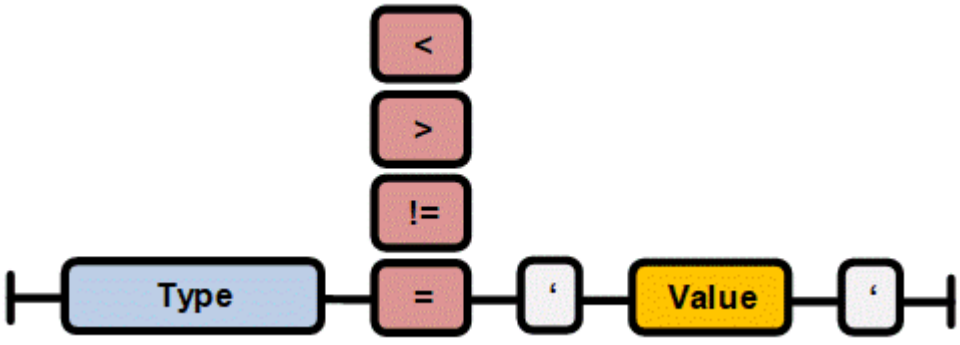
Search for symbols whose name begins with 'bEn'.

```
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.



Samples:

```
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'
```

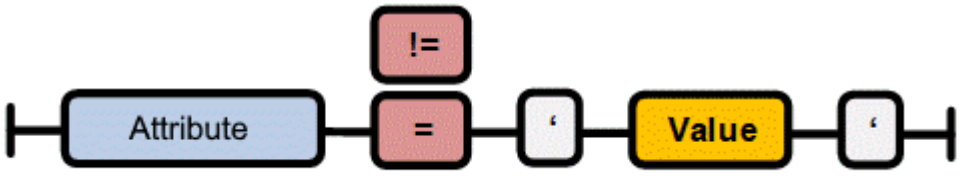
Search for symbols whose name begins with 'FB_BACnet'.

```
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'.

7.2.6 Command Line Interface

The Symbol Explorer can be called up via the command line using the command line interface (CLI).

The following commands are available:

```
--SnapShotFromPlc     Take a snapshot from PLC target.
--SyncPlcToSnapShot   Synchronize PLC target with a Snapshot.
--SyncFileToPlc       Synchronize Snapshot with a PLC target.
```

--SnapShotFromPlc

Create snapshots of a PLC.

Option	Description
<NetId>	AMS Net Id of the target
<Port>	AMS port of the target
<SnapShotPath>	SnapShot file path

```
  --SnapShotFromPlc <NetId> <Port> <SnapShotPath>
```

--SyncPlcToSnapShot

Synchronize PLC symbol values with a snapshot.
The current PLC symbol values are transferred to a snapshot.

Option	Description
<NetId>	AMS Net Id of the target
<Port>	AMS port of the target
<SnapShotPath>	SnapShot file path

```
  --SyncPlcToSnapShot <NetId> <Port> <SnapShotPath>
```

--SyncSnapShotToPlc

Synchronize snapshot symbol values with the PLC.
The snapshot symbol values are transferred to the PLC.

Option	Description
<NetId>	AMS Net Id of the target
<Port>	AMS port of the target
<SnapShotPath>	SnapShot file path

```
  --SyncSnapShotToPlc <NetId> <Port> <SnapShotPath>
```

Symbol Entry Point

A symbol entry point is required for the CLI as well as via the UI to access symbols of a target. The symbol entry point can be specified for the CLI in the form of a configuration file. To create the file, the target must be accessed once with the UI and the Symbol Entry Point dialog is started. In the dialog, you can select which symbols are of interest, just as for the UI.

Call sample

```
...\TwinCAT\Functions\SymbolExplorer.exe --SnapShotFromPlc <192.168.10.1.1> <851> <"c:
\Temp\Snapshots\File.tcpvs>
```

Sample for starting the Symbol Explorer from the PLC

TwinCAT 3 PLC Library: Tc2_Utilities -> NT_StartProcess Sample

NT_StartProcess

Sample for configuring the input variables.

PATHSTR: "...\TwinCAT\Functions\SymbolExplorer.exe"

DIRNAME: "...\TwinCAT\Functions"

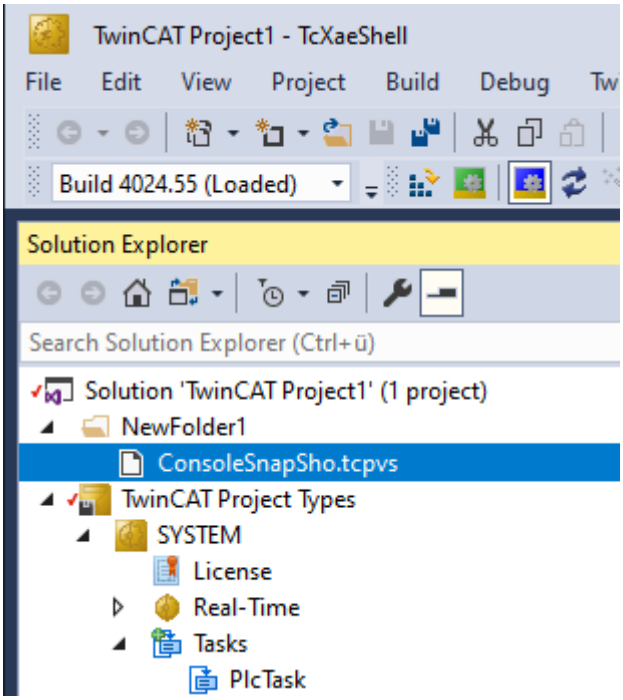
COMNDLINE: "--SnapShotFromPlc 192.168.10.1.1 851 "c:\Temp\Snapshots\File.tcpvs".

7.2.7 Workflows

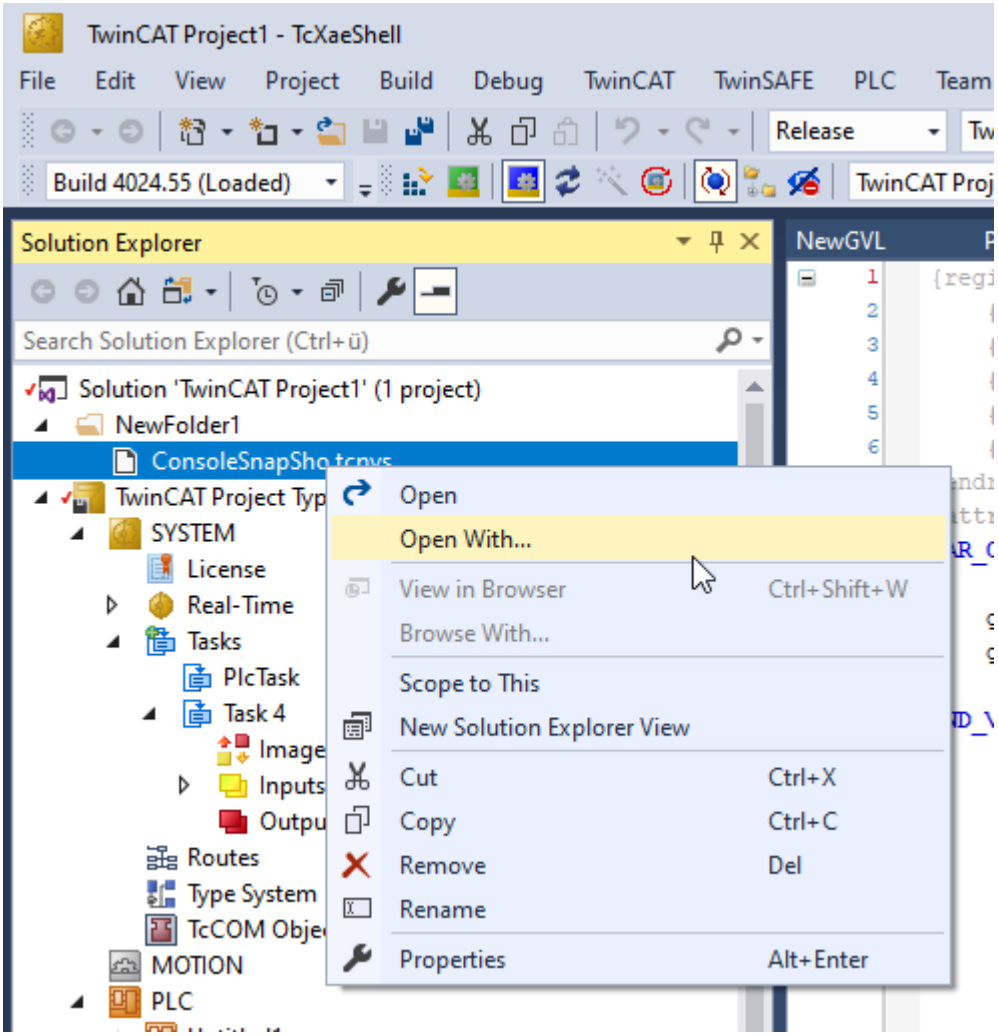
The following are some examples of how to use the Symbol Explorer.

7.2.7.1 Opening snapshots from Visual Studio and the XAE Shell

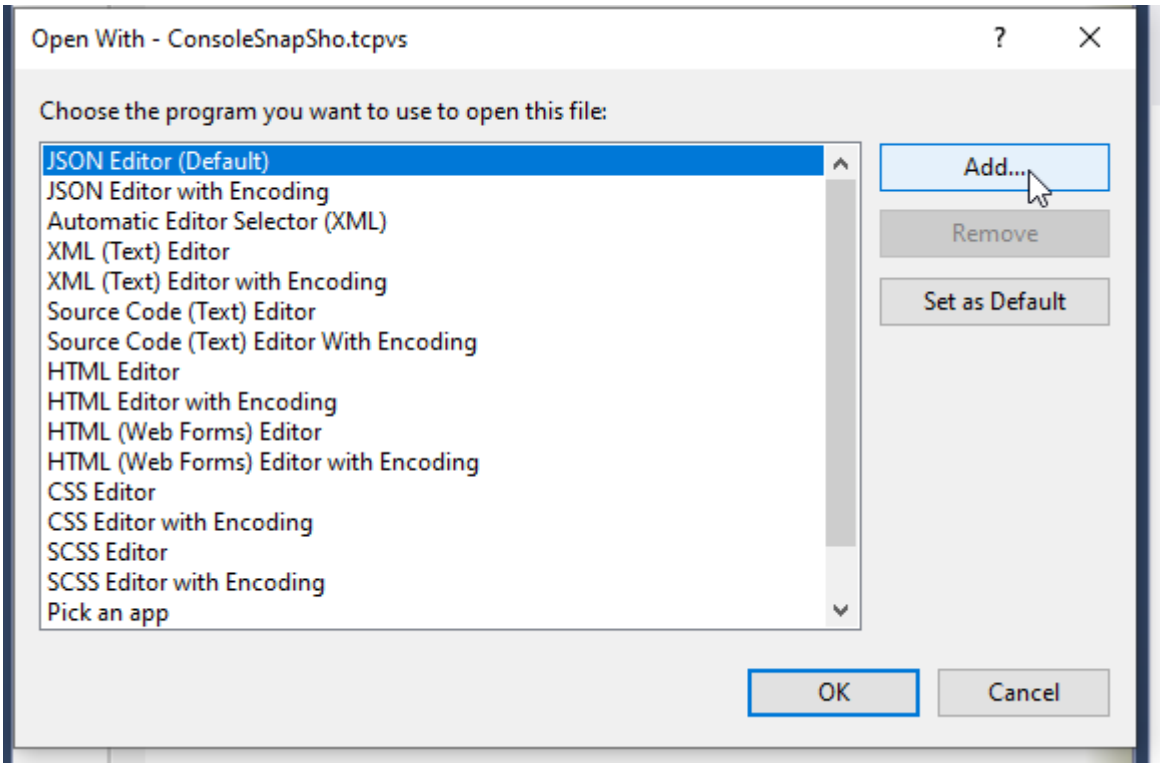
1. Add the Symbol Explorer snapshot to the Visual Studio / XAE.



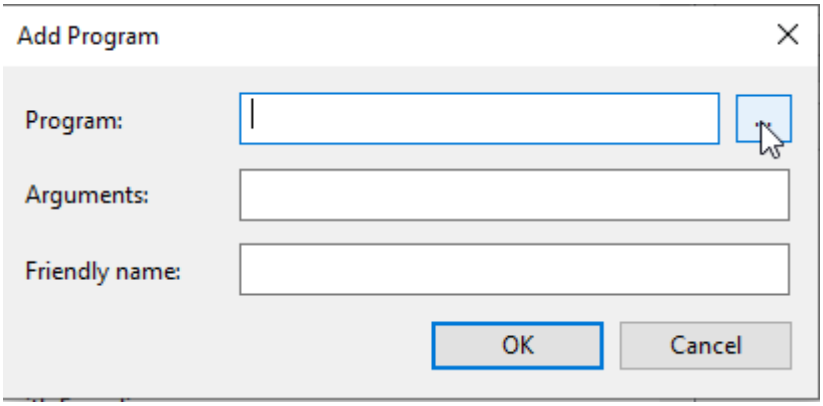
2. Right-click on the snapshot and select **Open With...** from the context menu



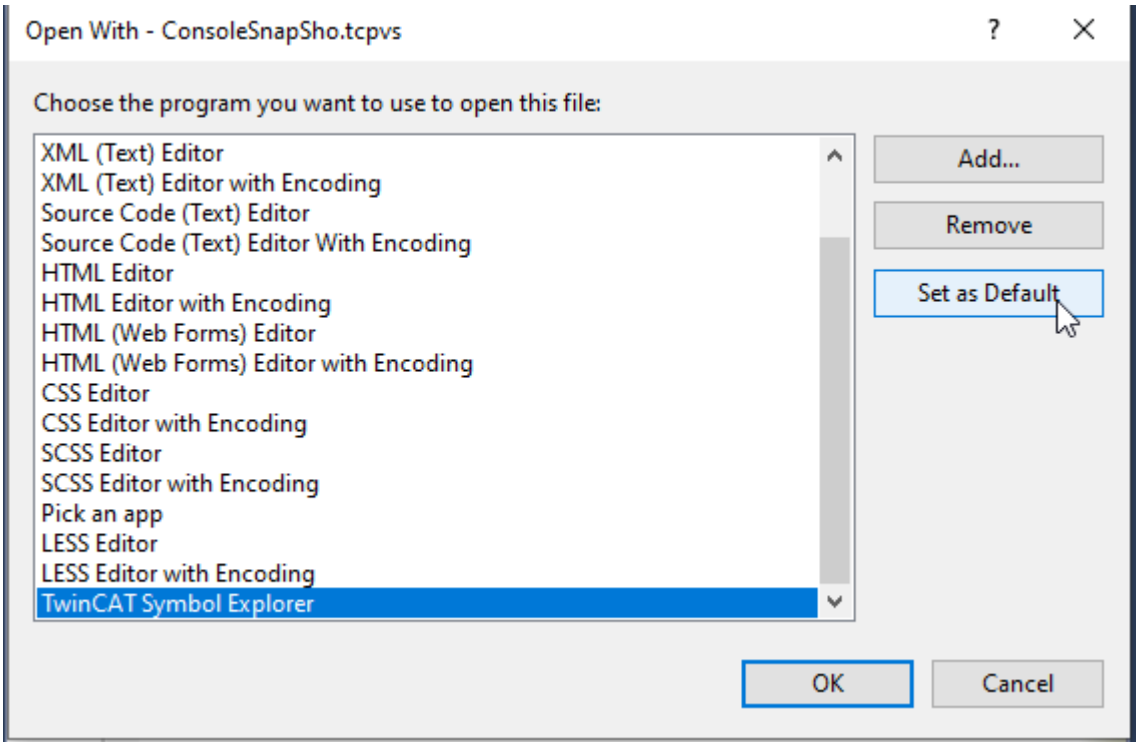
3. Click on the **Add** button.



4. Click on the button with the dots.
   Search for the *SymbolExplorer.exe* file and confirm with **OK**.



5. Select **TwinCAT Symbol Explorer** from the list of programs and click on the **Set as Default** button. Confirm the dialog with **OK**.



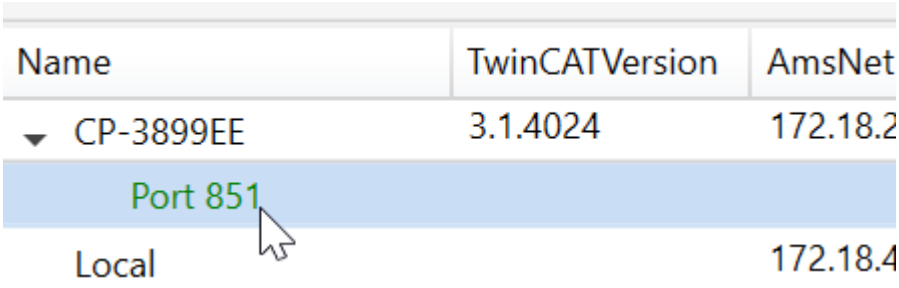
⇨ The Symbol Explorer has now been linked as a program for opening snapshots.

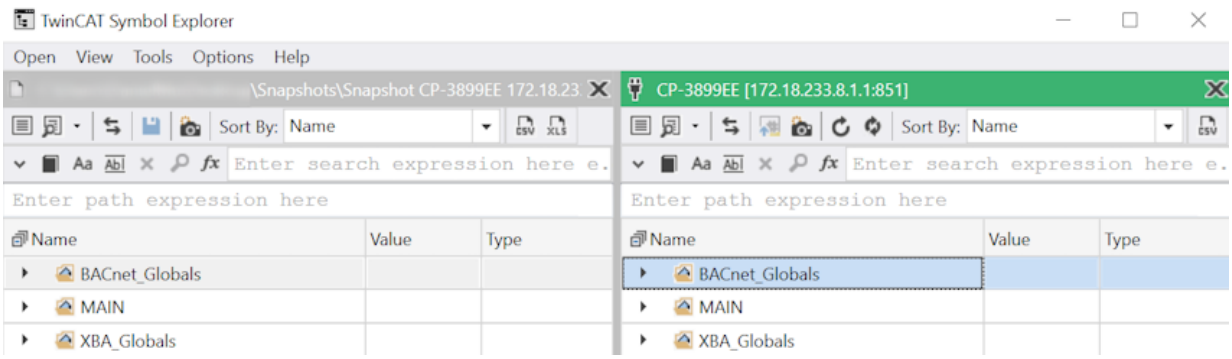
7.2.7.2 Instance view for parameterization of AI and AO objects

The following example shows the creation of an instance watch to carry out typical parameterization tasks on analog objects. The view can be used, for example, to log the parameterization of a sensor calibration.

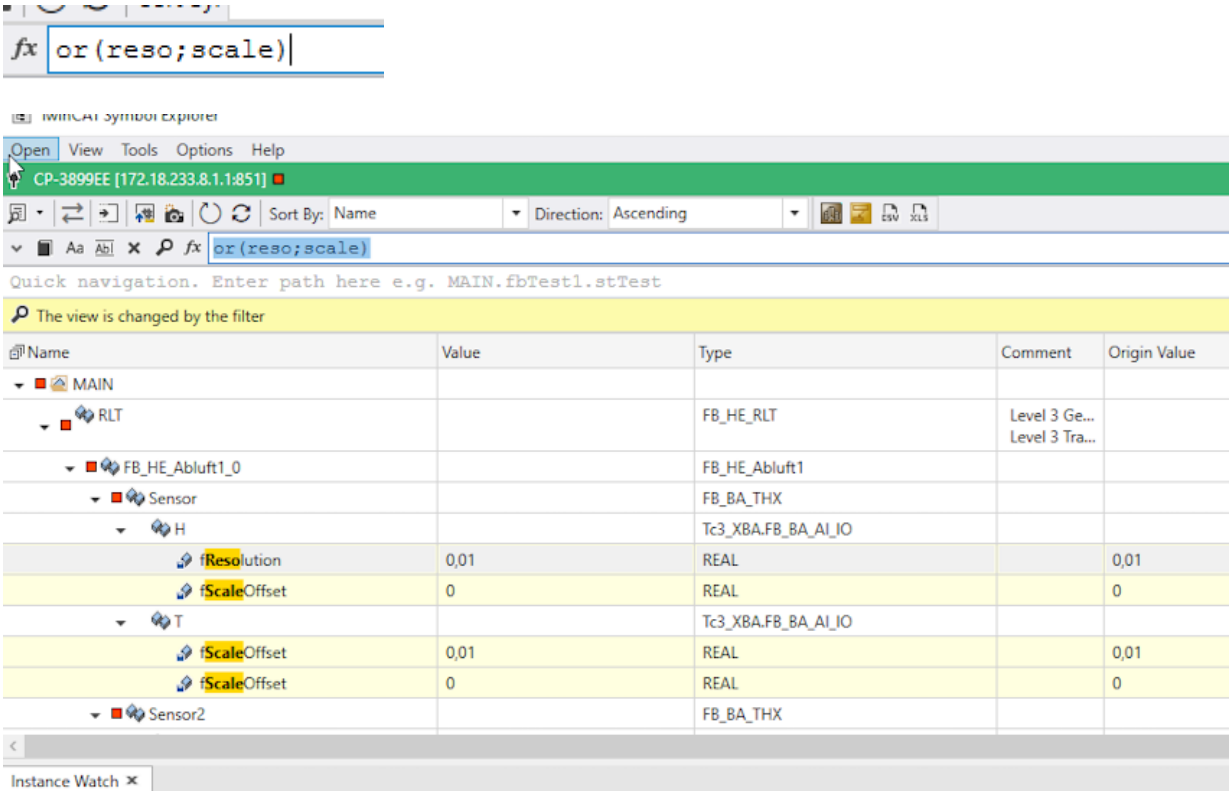
Connecting to a target system

1. Log in to the target system in the Symbol Explorer.

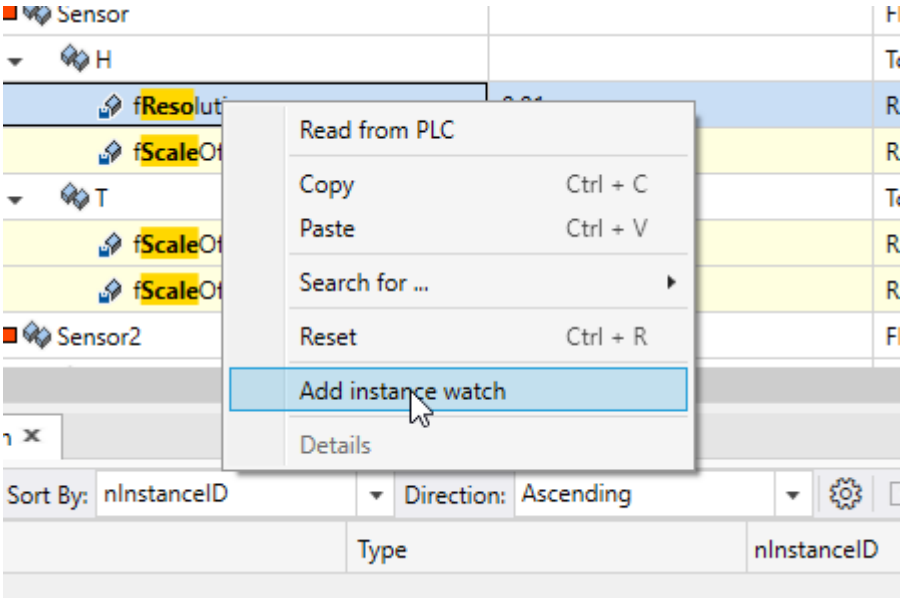




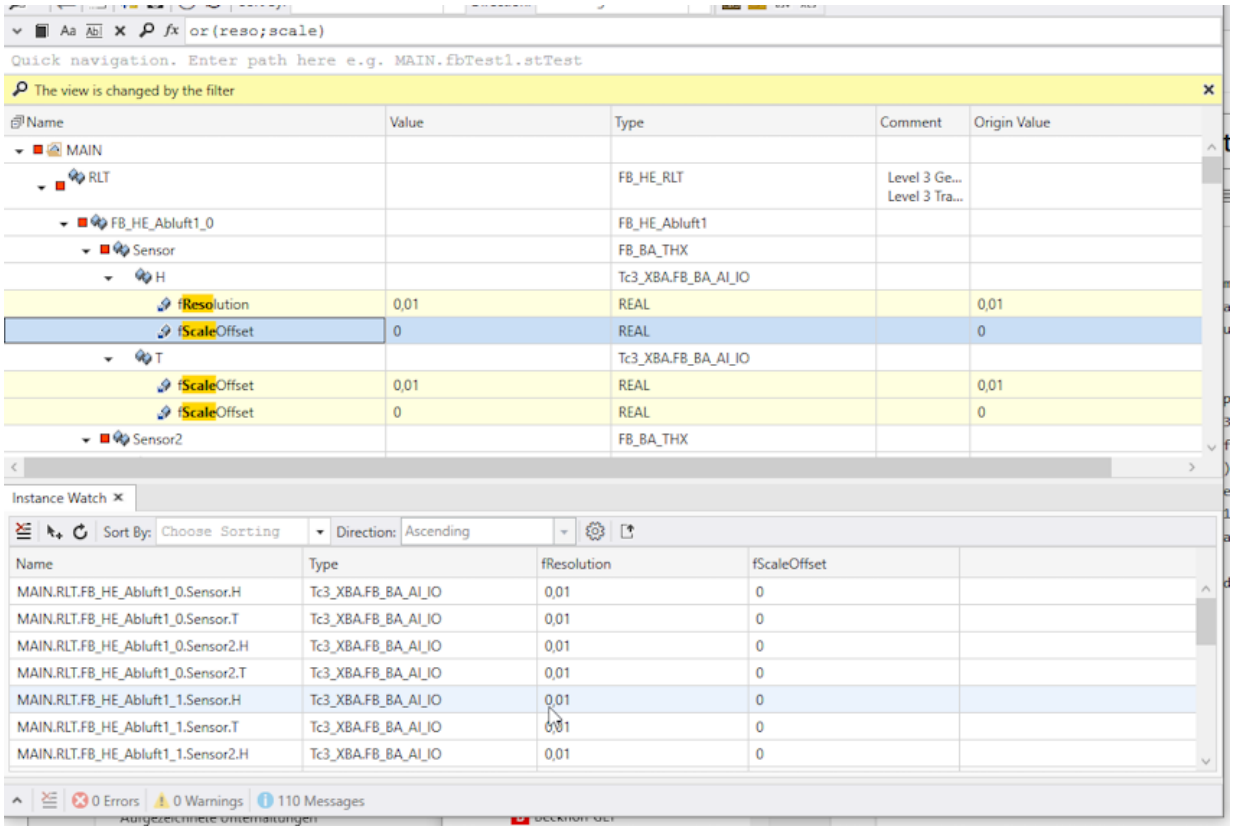
2. Set a filter expression.



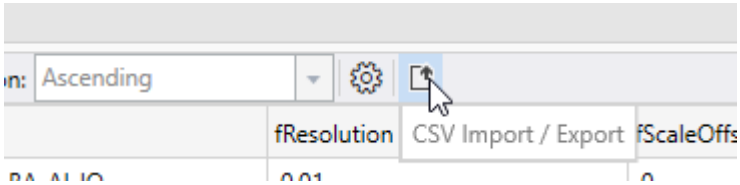
3. Open the InstanceWatch window.



4. Drag and drop the **ScaleOffset** and **Resolution** symbols into the Instance window.



5. Generate a filter from the Instance Watch.



⇨ This function creates a filter for the tree view with the following expression:

```
And(Or(Name='fResolution';Name='fScaleOffset');Parent(Or(Type='Tc3_XBA.FB_BA_AI_IO')))
```

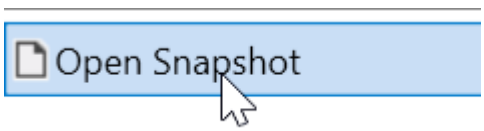
7.2.7.3 Synchronizing filtered data, example with Exclude

In the following example, data from a snapshot is synchronized:

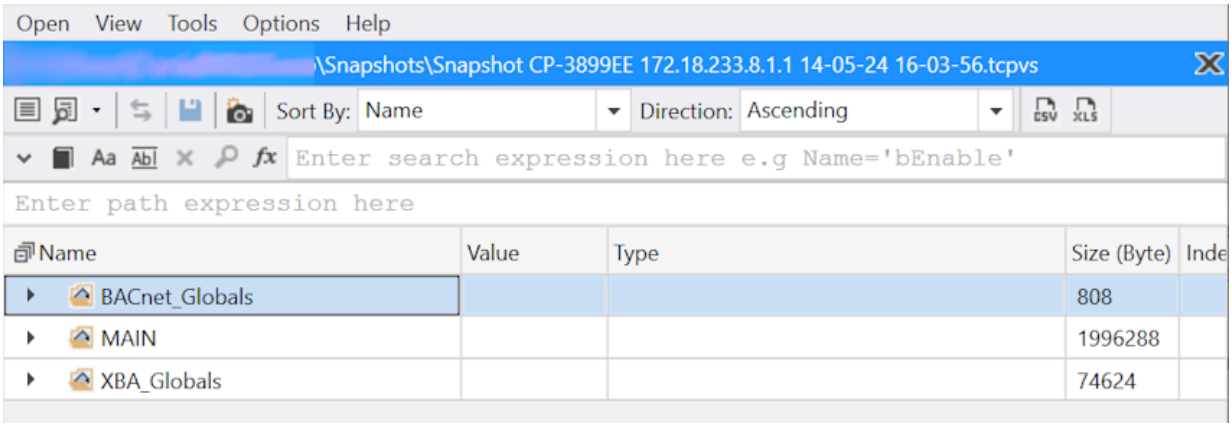
All changes from the snapshot should be transferred to the controller, except for symbols with operating data with the name *nActiveTimeElapsed* and *nStateChangeCount*.

Connecting to the target system and loading a snapshot

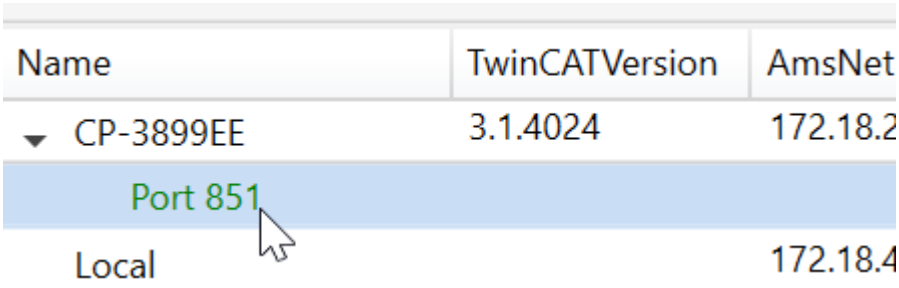
1. Open the snapshot [▶ 1303] with data backup of the affected target system (PLC).

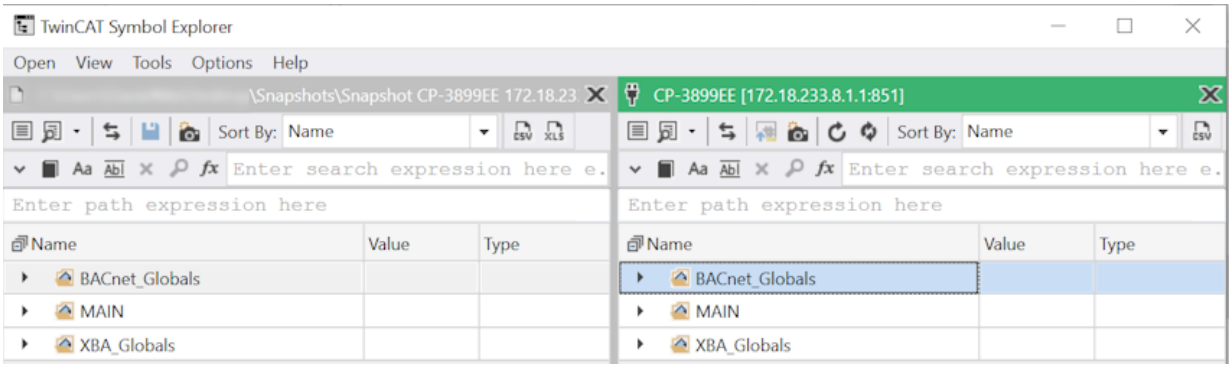


2. Select **BACnet_Globals**.



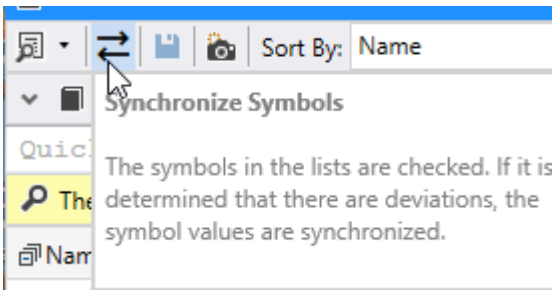
3. Log in to the target system in the Symbol Explorer.





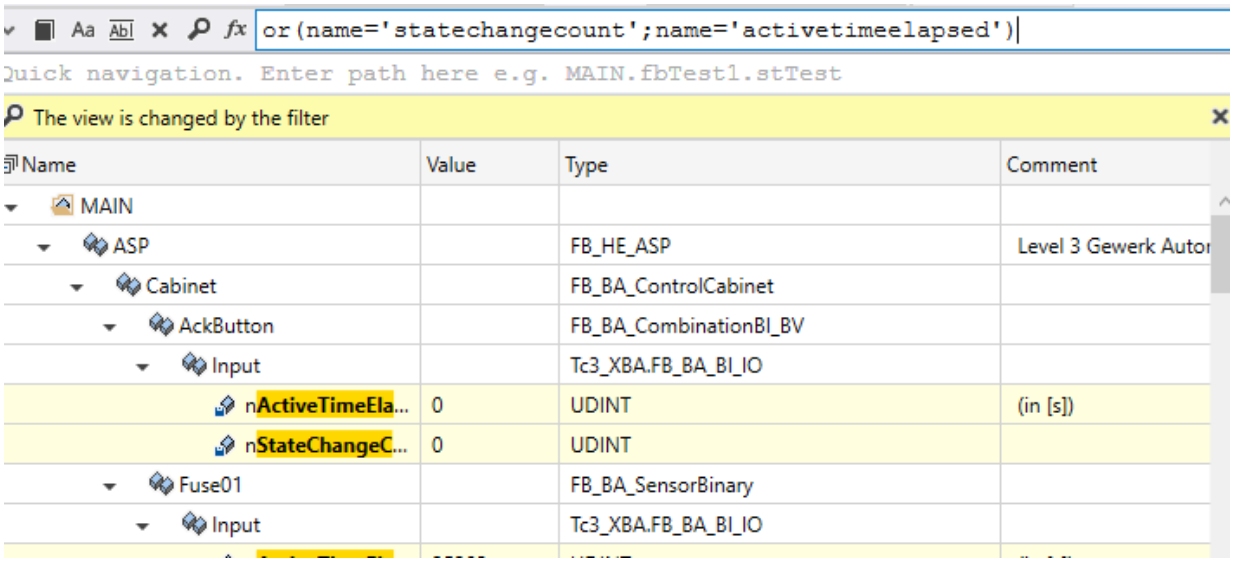
Synchronize and snapshot -> target system with filtered symbols

1. Synchronize the data from the snapshot to the target system by clicking the Sync button in the main window of the snapshot.

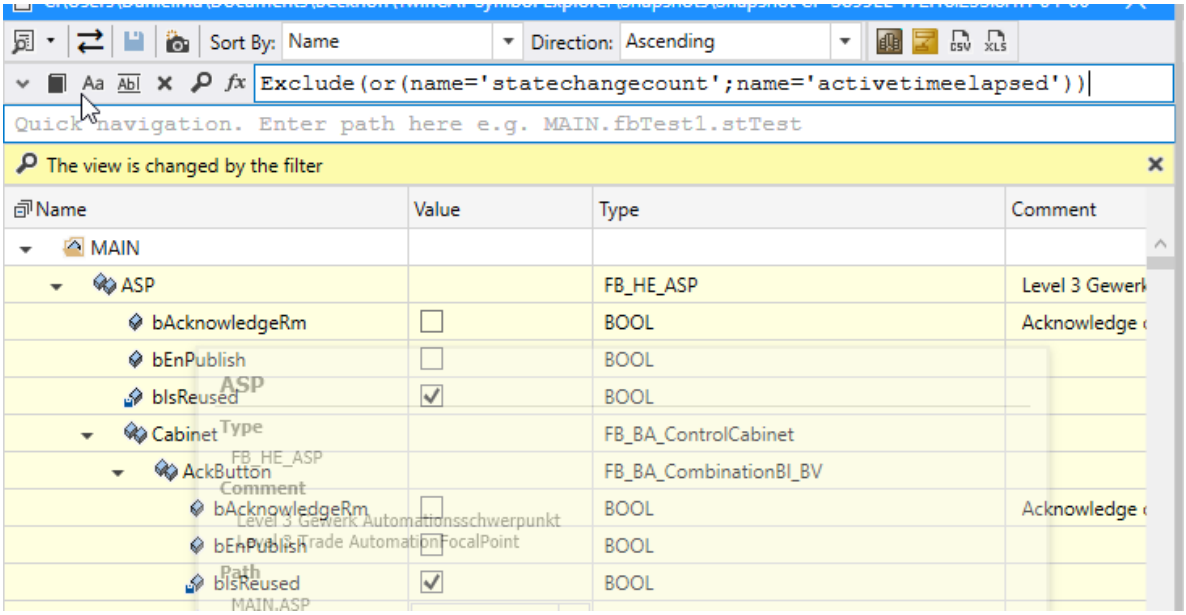


⇨ Suggestion for synchronization: 43 symbols are found for synchronization.

2. Set filters on the symbols **nStateChangeCount** and **nActiveTimeElapsed** in the snapshot.
   Filter:
   or(name='statechangecount';name='activetimeelapsed')

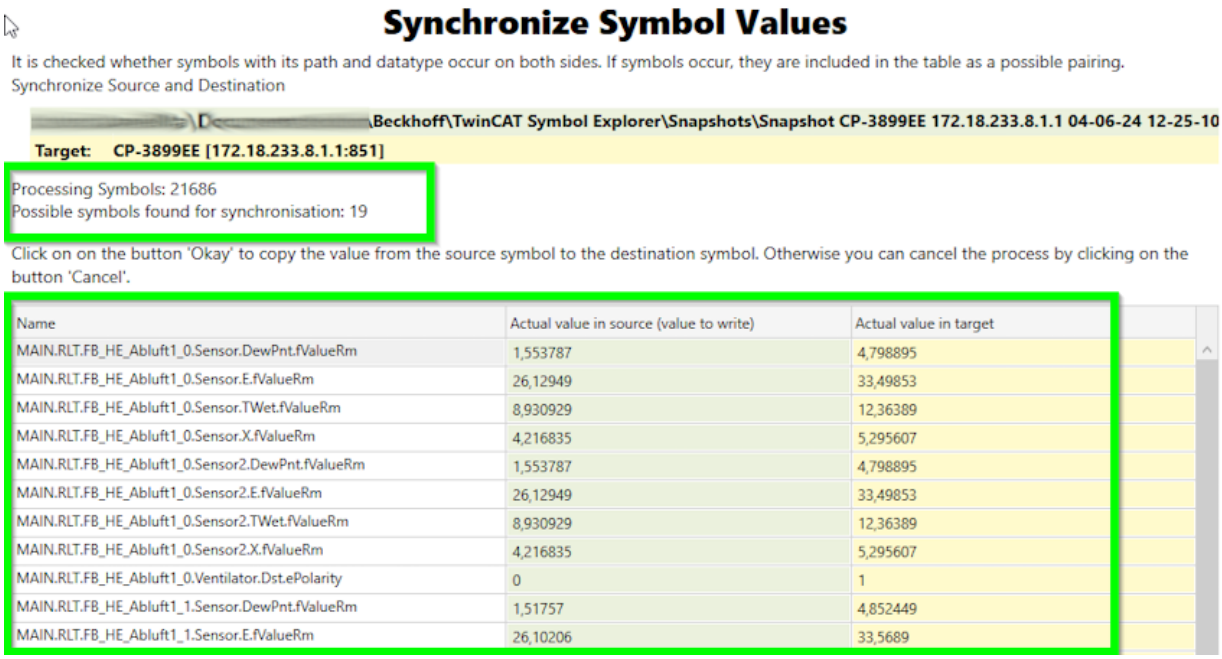


3. Negate the selection using the Exclude function.
   ⇨ All symbols except **nStateChangeCount** and **nActiveTimeElapsed** are displayed.



4. Click on the **Sync** button again.

⇨ Now only 19 symbols are available for synchronization. Symbols that receive the operating data have been successfully filtered out



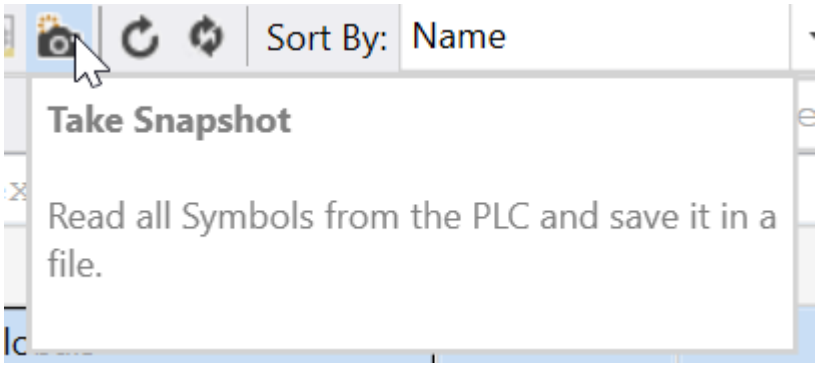
7.2.7.4 Back up and restore BACnet ObjectIDs with snapshot

These instructions serve as a workflow for writing BACnet ObjectIDs from a SymbolExplorer snapshot back into a PLC.

Restoring object instance IDs is necessary if, for example, a controller that is already communicating with a BACnet MBE is replaced.

Back up IDs:

1. Login with the SymbolExplorer.
2. Create snapshot



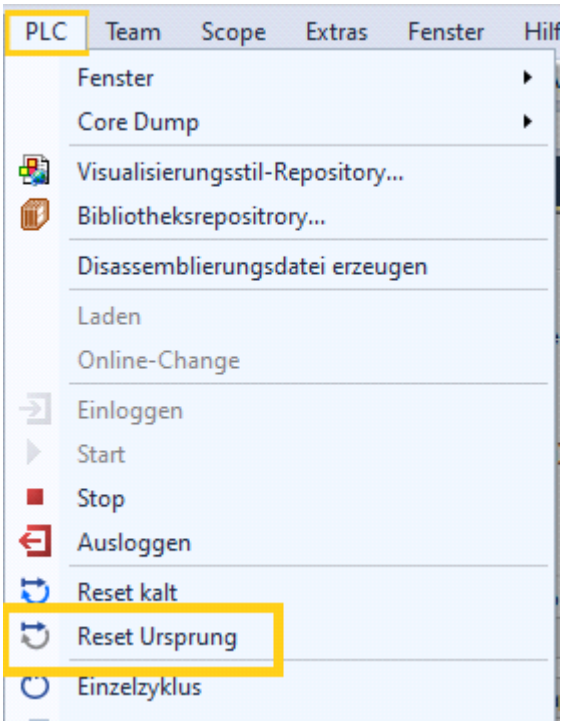
Start the SymbolExplorer and connect it to a controller. Click on the Take Snapshot button in the Online window to create a snapshot.
Sekect the storage location for the snapshot file.

⇨ Once you have selected the folder and confirmed with OK, all values of the symbols in the list are synchronized with the PLC and written to the file.

Restore IDs from a snapshot in the target system:

Start in XAE

1. PLC Reset origin

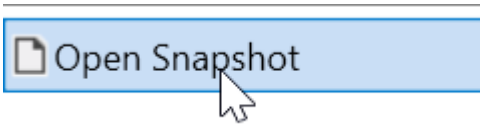


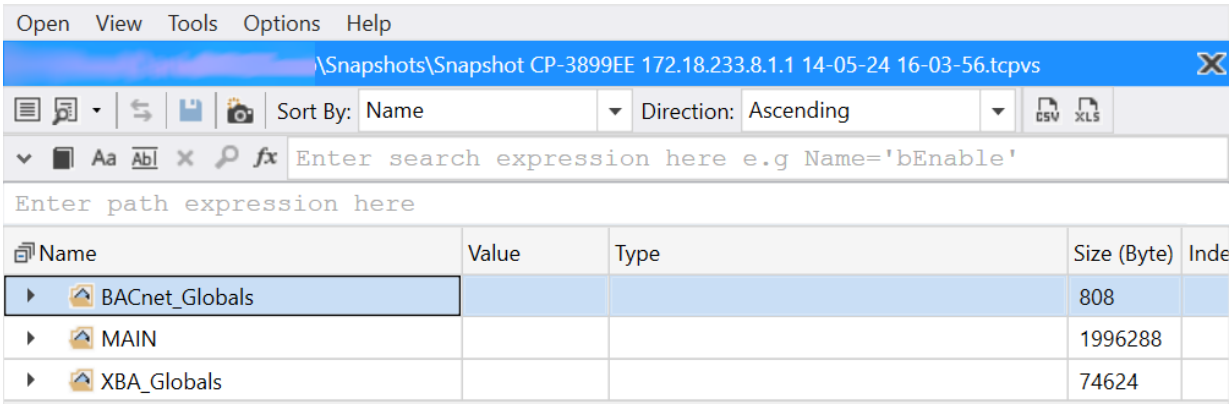
2. Log in PLC

3. **Do not** start the PLC!

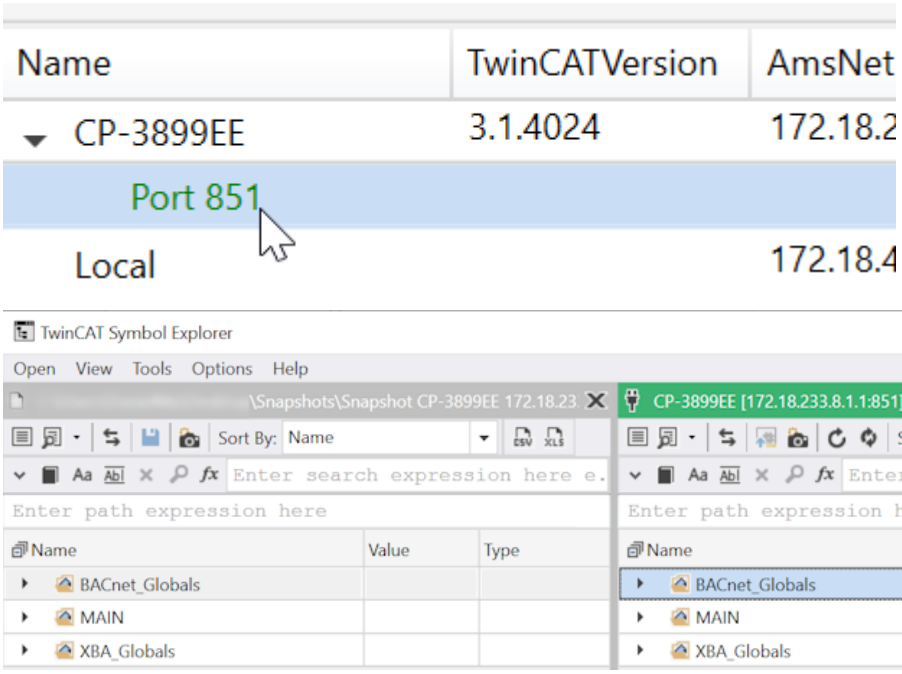
SymbolExplorer

1. Open the snapshot [▶ 1303] with the data backup of the affected target system (PLC).

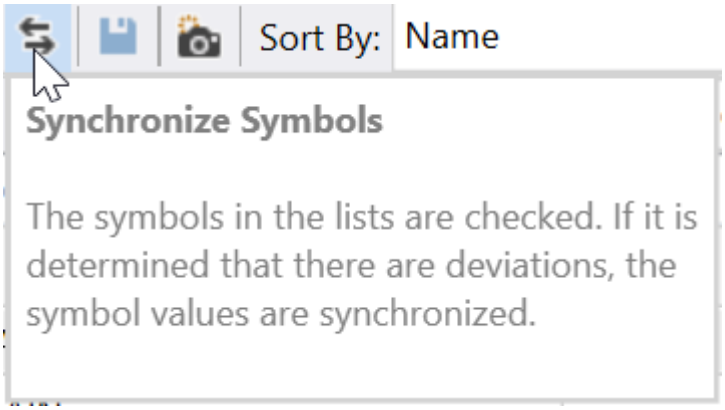




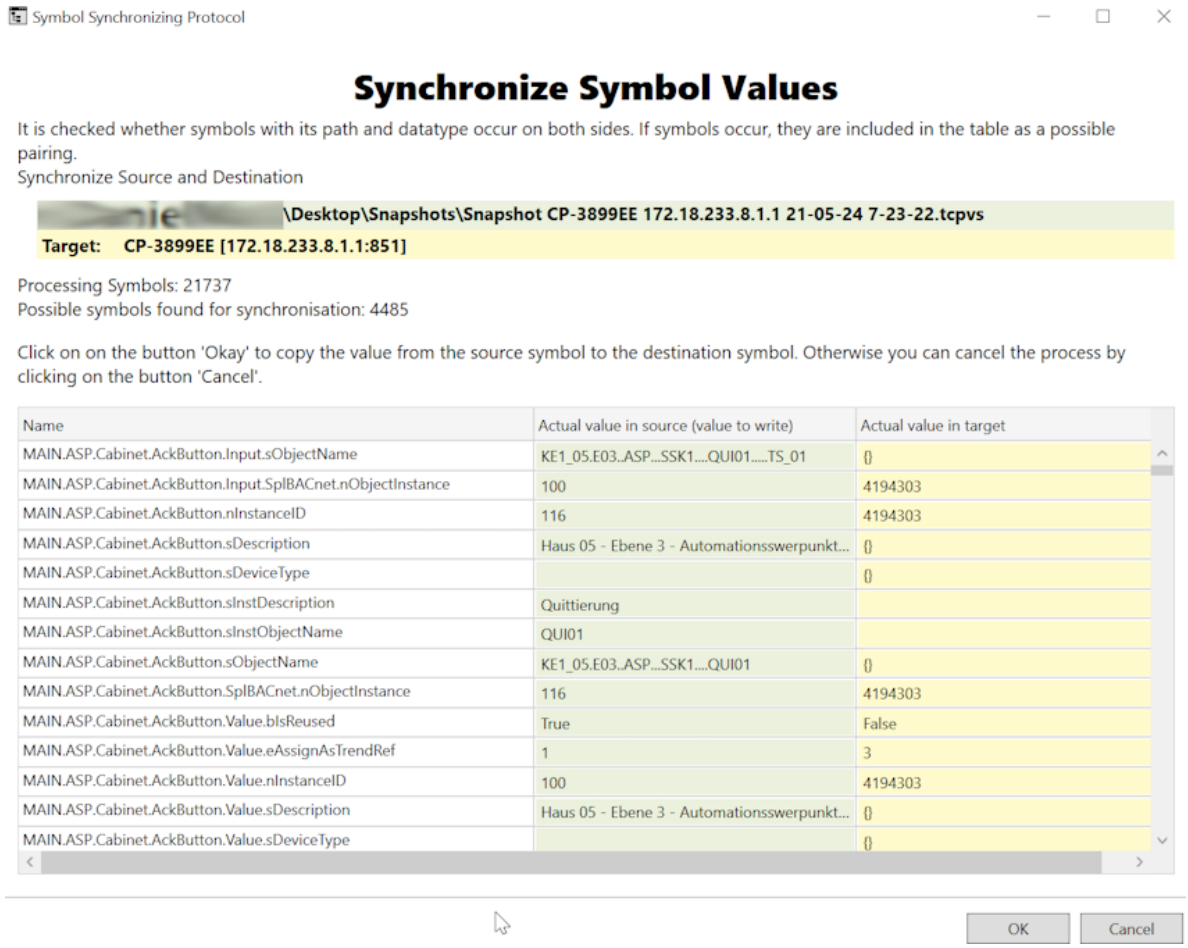
2. Log in to the target system in the Symbol Explorer



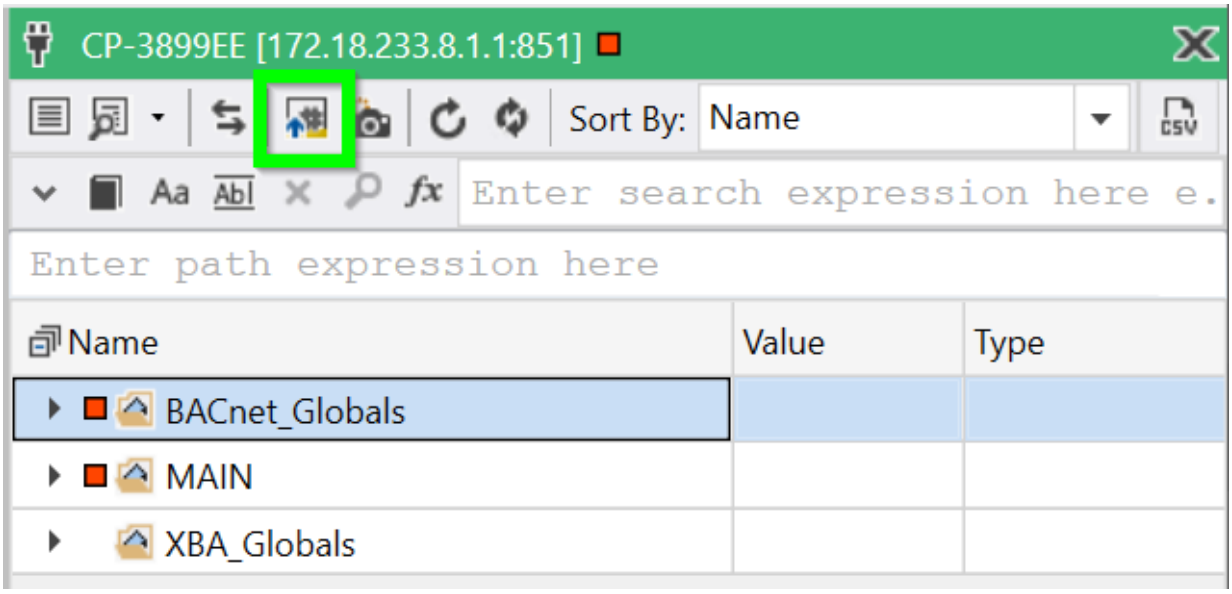
3. Synchronize the data from the snapshot to the target system



⇨ The symbols for synchronization are displayed in the dialog box and prepared for transfer to the target system.

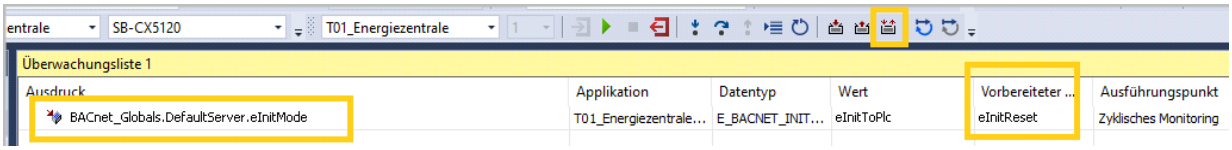


4. Write the symbols from the snapshot to the PLC using the SymbolExplorer.



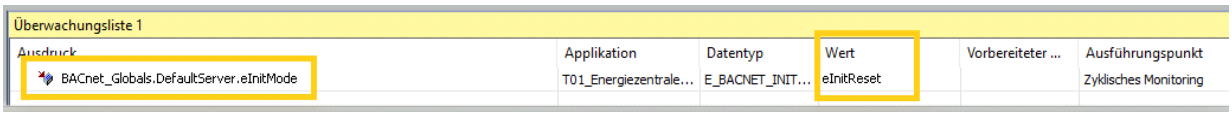
Checking in the XAE

1. Write the Init mode from the BACnet server in the PLC to *eInitReset*.



2. Start the PLC.

⇨ The BACnet server starts in *eInitMode* and recreates the objects.



7.2.8 Extensions

The functional scope of the Symbol Explorer has been expanded to include industry-specific features:

- Building Automation I/O Export [▶ 1339]:
  This extension is used to export I/O variables of type *Tc3_XBA.FB_BA_**_Raw*.
- Building Automation Instance Watch [▶ 1341]:
  This extension is a more comprehensive Instance Watch that is adapted to the needs of building automation.

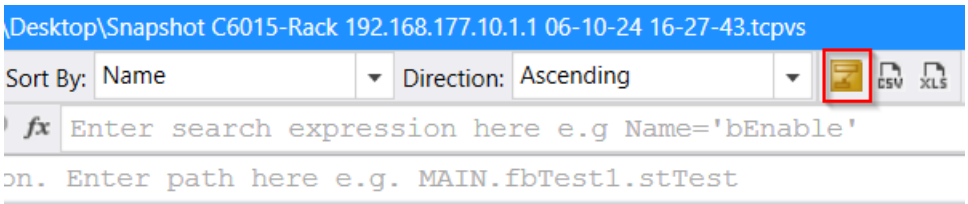
7.2.8.1 Building Automation I/O Export

This plugin creates a list of all symbols of the types "Tc3_XBA.FB_BA_**_Raw" that are present in the opened route or snapshot. The symbols found can be saved as CSV or TwinCAT Function Block.

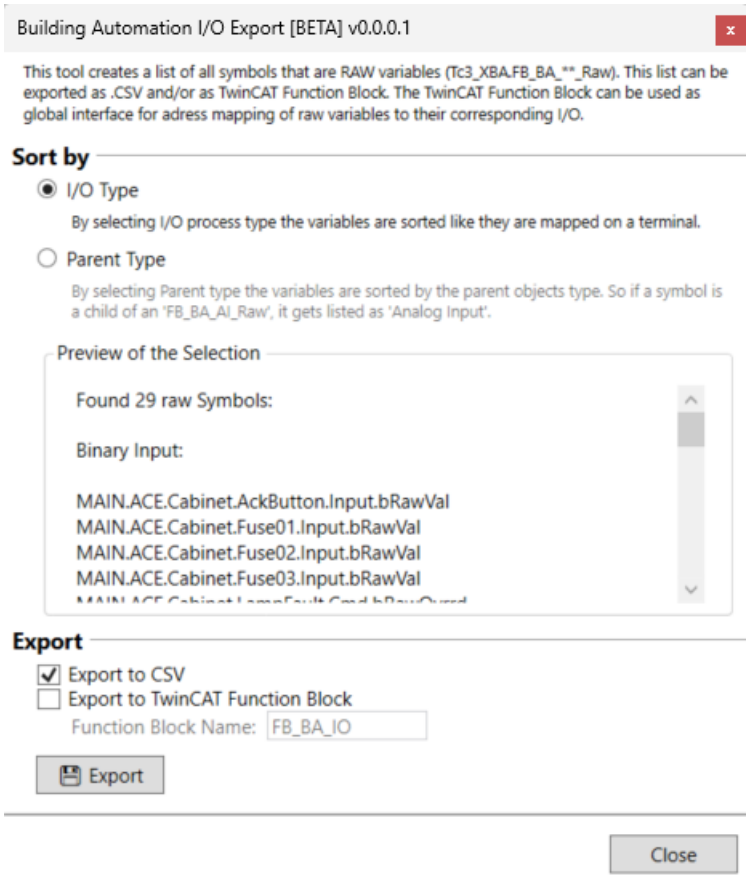
Starting the I/O export

The I/O export is available in an open route or snapshot.

It can be found as a button in the top toolbar:



A window opens in which the RAW objects can first be sorted according to their own I/O process types or according to the type of the parent element:

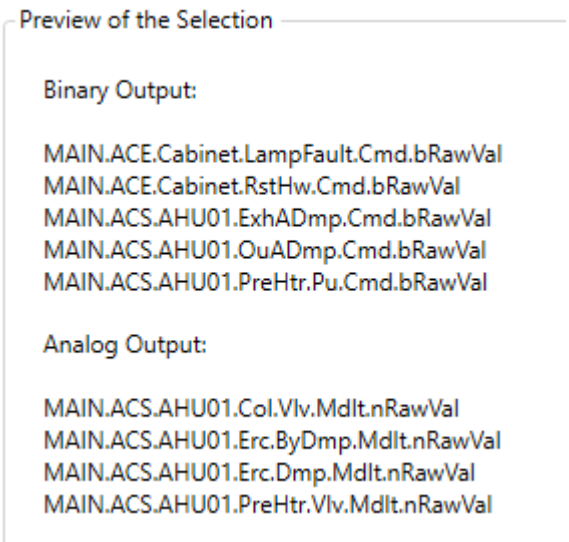


If sorted according to the I/O process type(**I/O Type**), the object is sorted according to its own type.

When sorting according to the type of the parent element (**Parent Type**), the type of the parent element is decisive.

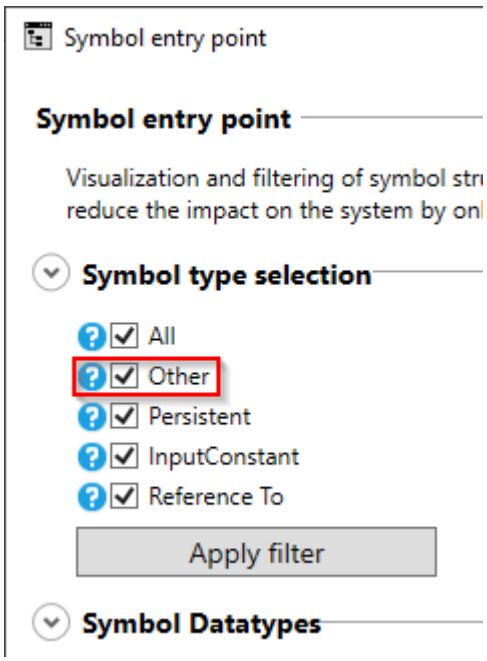
The preview (**Preview of the Selection**) is updated as a result of the upper selection.

The objects are sorted into the categories **Binary Input**, **Binary Output**, **Analog Input** and **Analog Output**:



If no symbols were found, a check should be made to see whether the Solution contains symbols of the types *Tc3_XBA.FB_BA_**_Raw*.

In addition, the Symbol Explorer must be set so that the entry **Other** is selected in the Symbol Entry Point dialog under **Symbol Type Selection**.



For a snapshot, this must be set in the **Symbol Entry Point** dialog of a route before it is created. Otherwise these symbols will not be saved in a snapshot.

The RAW variables found can be exported using the '□ Export' button.

A CSV file and/or a TwinCAT Function Block is available as an export option.

The name of the function block can be entered in the text field **Function Block Name**.

The TwinCAT Function Block can be used as a global interface for address mappings.

For use, all placeholders ('YOUR_IO_**') must be replaced with the corresponding I/O variables.

7.2.8.2 Instance Watch BA

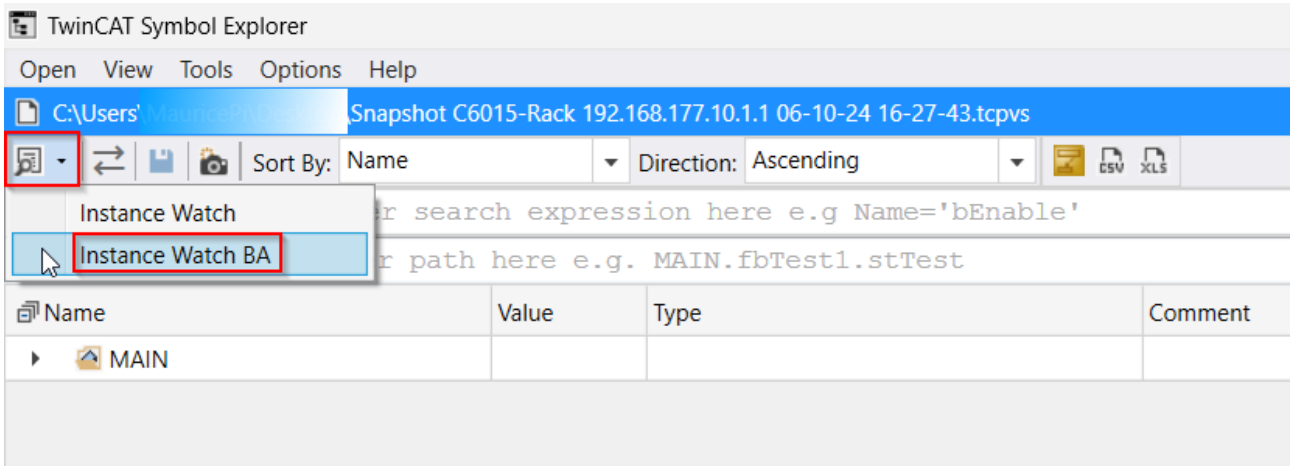
The Instance Watch BA is an extension of the Instance Watch integrated in the Symbol Explorer.

The extensions concern a customized Instance Watch BA template, which is loaded when the Instance Watch BA is started, and an extended view of sophisticated BA-specific data types.

Start of the Instance Watch BA

First, start the Symbol Explorer and open a snapshot or route.

The **Instance Watch BA** can then be selected from the drop-down menu in the top left-hand corner:



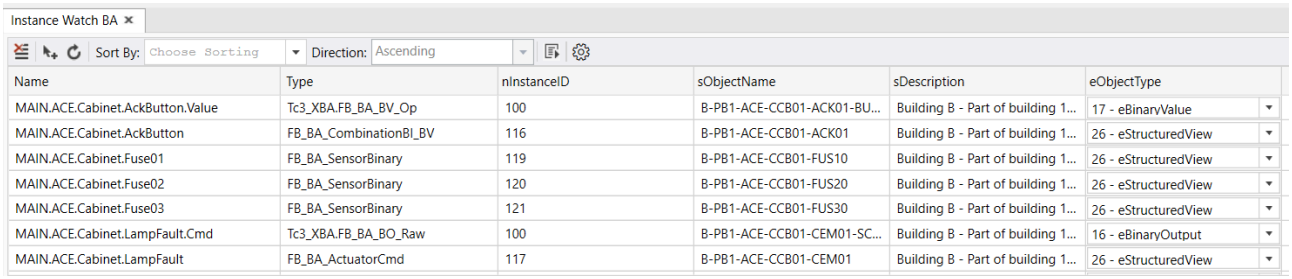
An Instance Watch with the title **Instance Watch BA** and an already loaded Instance Watch template for Building Automation will open.

BA-specific Instance Watch Template

This template contains every object that has implemented the interface "Tc3_XBA.I_BA_View" or "Tc3_XBA.I_BA_Object".

The properties *nInstanceId*, *sObjectName*, *sDescription* and *eObjectType* are also displayed for these objects.

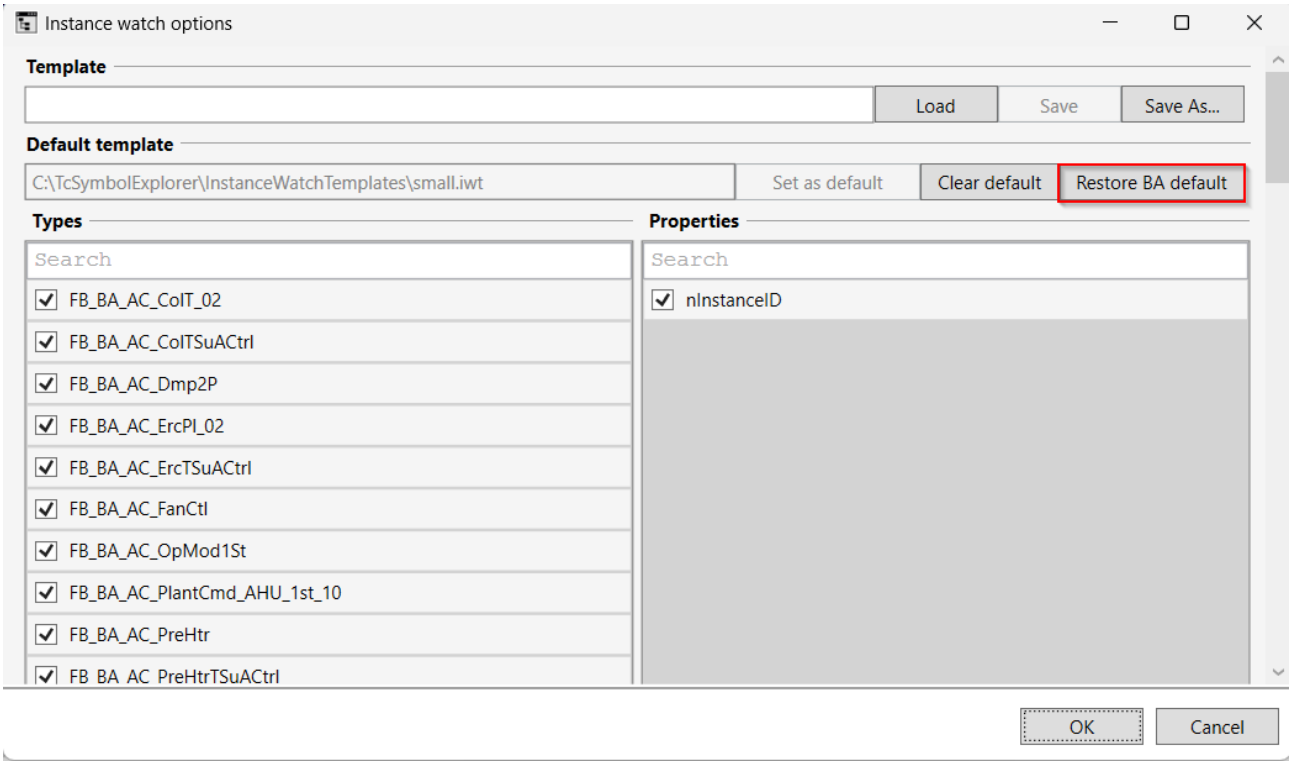
This template is loaded every time the Instance Watch BA is started.



Resetting the Instance Watch Template

If changes have been made to the Instance Watch Template (e.g. properties added/removed), the BA-specific Instance Watch Template can be restored. To do this, the **Instance Watch Options** must be opened. To reset the template, press the **Restore BA default** button.

Then confirm with **OK**.



Extended view of BA-specific sophisticated types

With the Instance Watch BA, some sophisticated types can be clearly displayed and edited.

This currently includes these types:

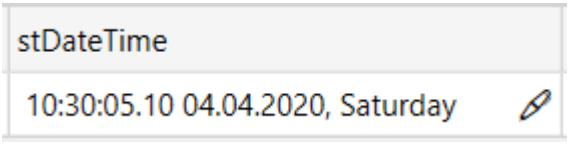
- Tc3_XBA.T_BA_EventTransitionText
- Tc3_XBA.T_BA_EventTransitions

- Tc3_XBA.ST_BA_LimitParam
- Tc3_XBA.ST_BA_TimeDelayParam
- Tc3_BA2_Common.ST_BA_StatusFlags
- Tc3_BA2_Common.ST_BA_Date
- Tc3_BA2_Common.ST_BA_Time
- Tc3_BA2_Common.ST_BA_DateTime

View

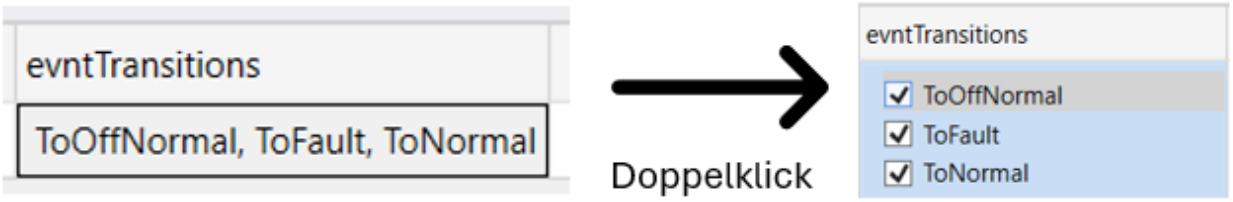
To see the simplified view of the above symbols, these symbols must be added to the Instance Watch BA as usual.

In the example of Tc3_BA2_Common.ST_BA_DateTime, the symbol looks like this:



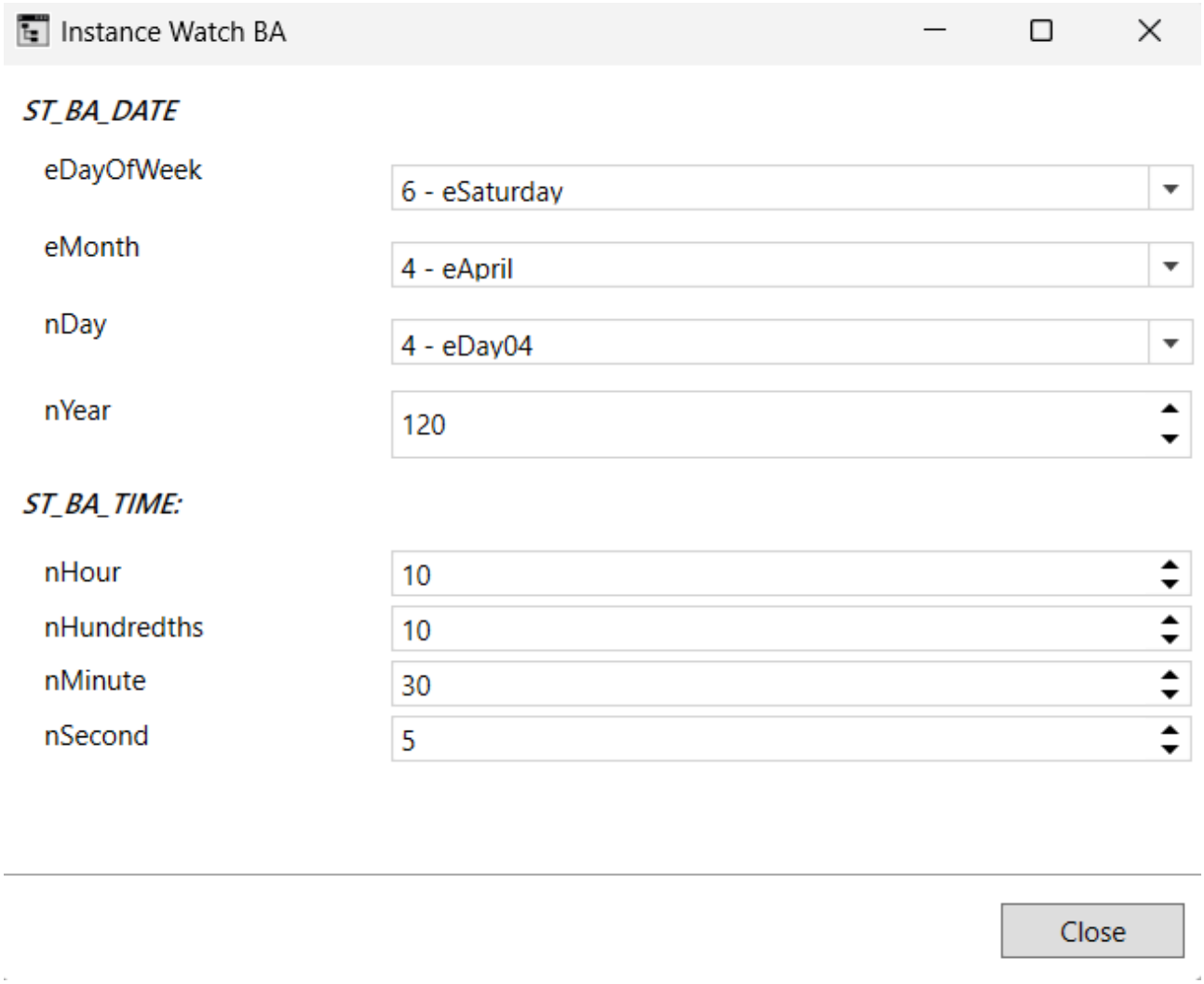
Processing

Symbols with a maximum of four sub-symbols can be edited by double-clicking on the symbol cell:



For symbols that have more than four sub-symbols, the symbol is edited using the pencil icon. This only appears for these symbols.

After clicking, a dialog opens:



Changes are applied immediately in both cases, but are not yet synchronized with the PLC or saved in the snapshot. Therefore, the dialog can be closed after editing the symbol.

7.2.9 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

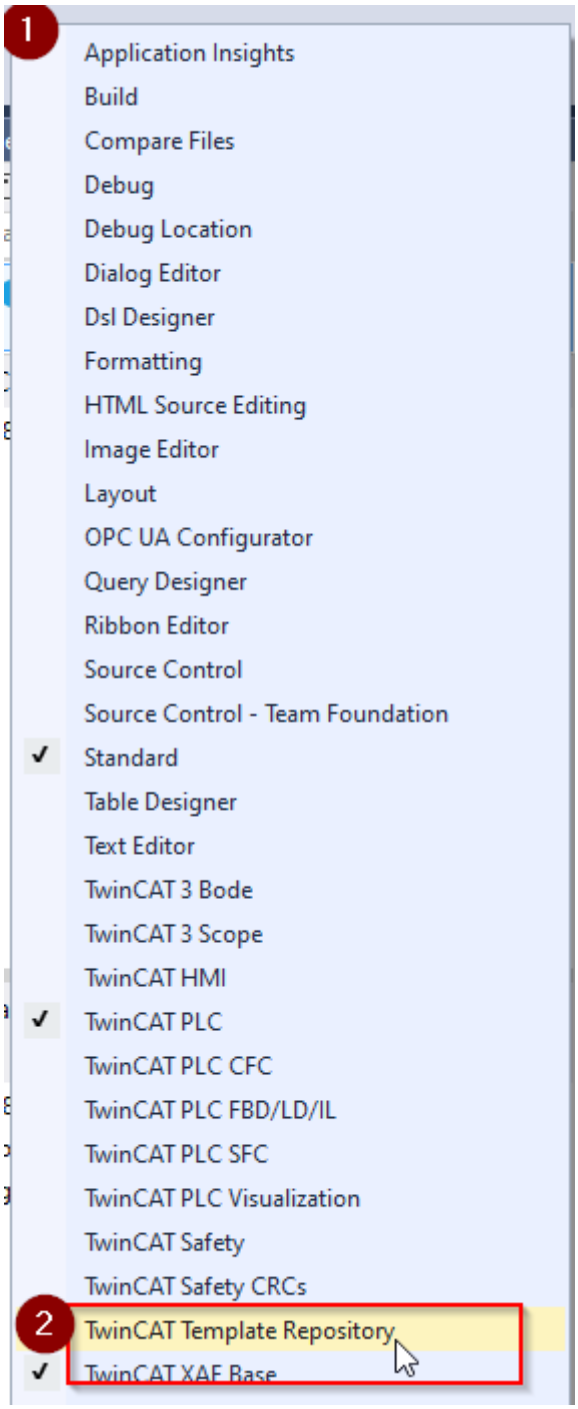
The Template Repository is an app for managing templates.

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).

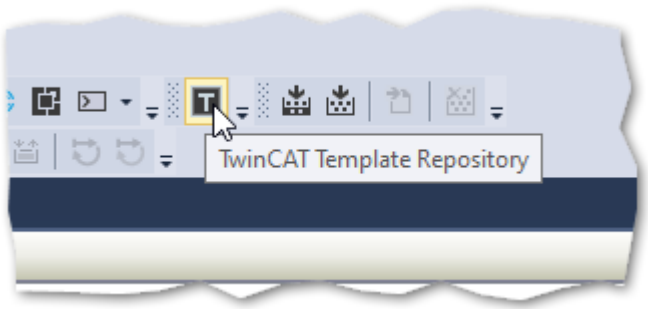
1. Right-click on a free field in the toolbar to open a context menu.

2. Select the app **Template Repository** from the context menu.



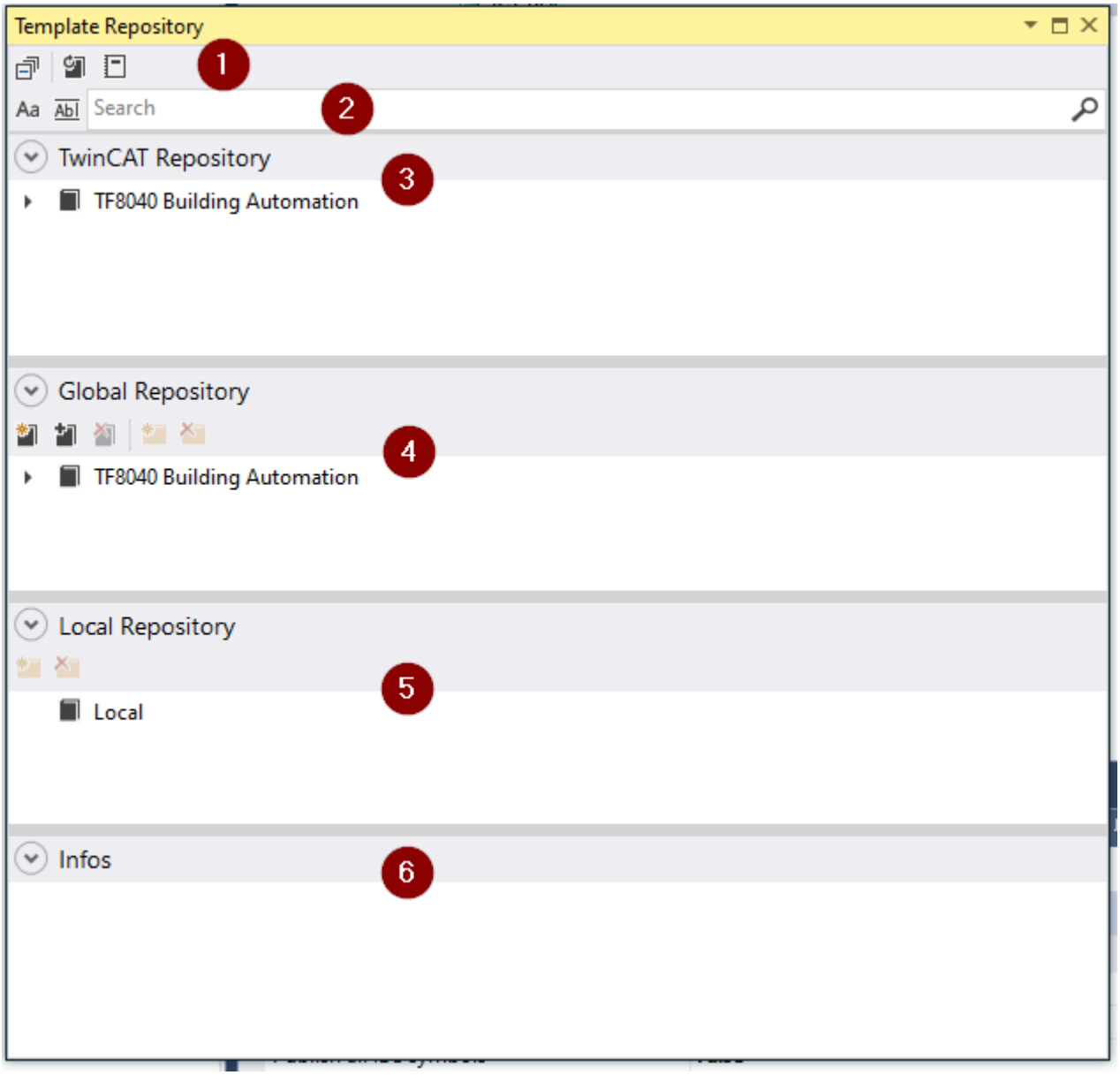
3. Start the app by clicking on the icon **TwinCAT Template Repository**.

⇨ The main window opens.



Main Window

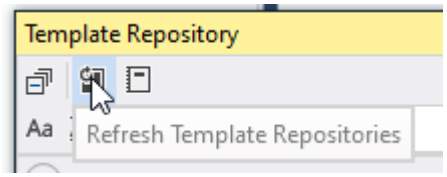
The main window is divided into 6 different areas:



1 Menu bar

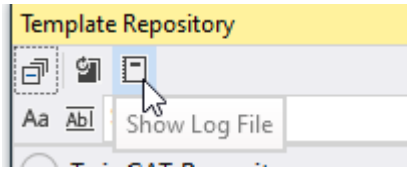
The following functions are available in the menu bar:

- **Refresh Template Repositories** updates all repositories



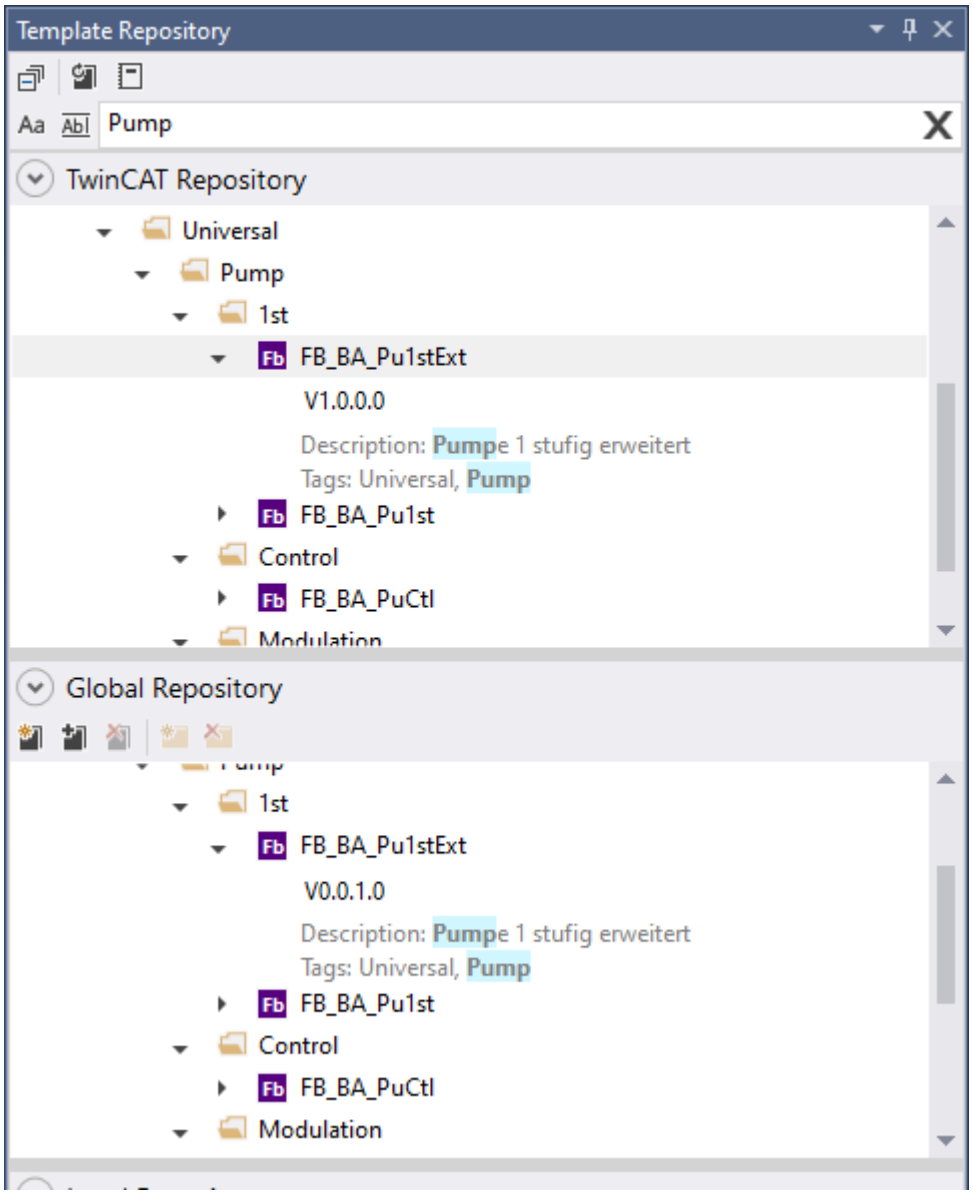
- **Show LogFile** opens the log file dialog

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 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.

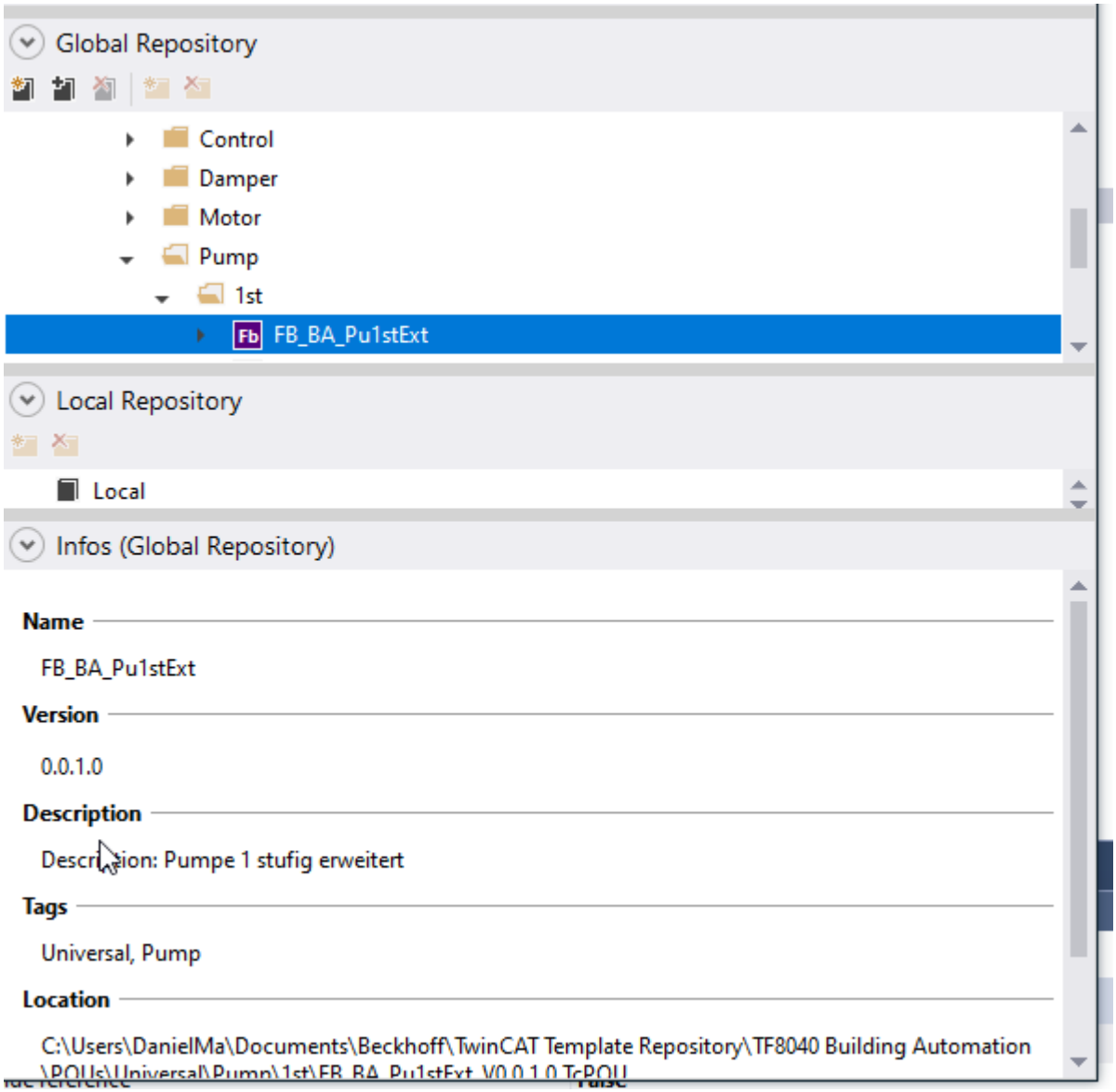
5 Local Repository area

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.

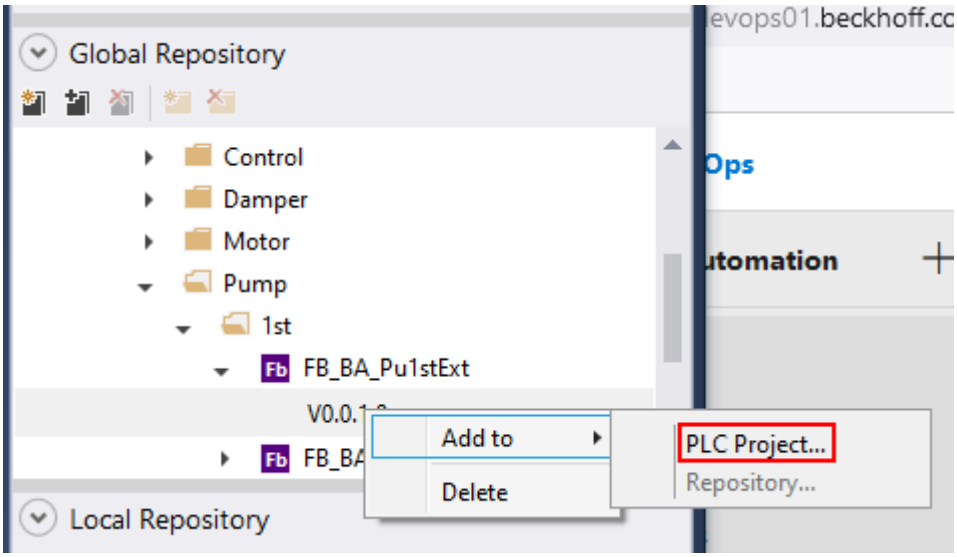


Adding templates to a PLC project

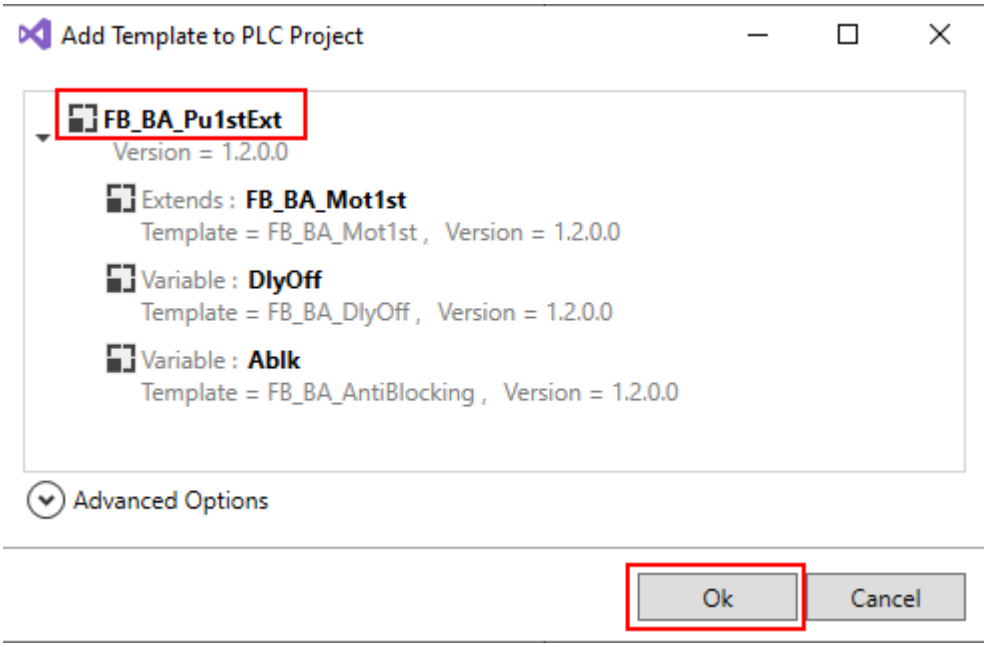
Templates can be integrated into the current solution in two ways:

✓ **Double click** on the desired template

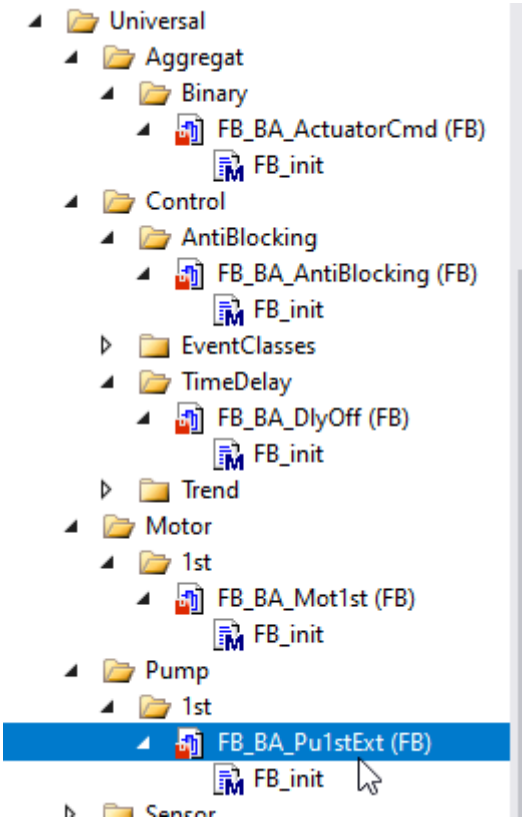
✓ **Right click** on the desired template **> Add to > PLC Project**



1. The dialog window **Add Template to PLC Project** *opens*. Select the template and confirm this with **OK**.



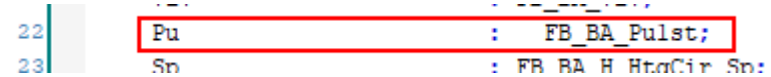
⇨ The desired template including all dependent subtemplates is now in the PLC project.



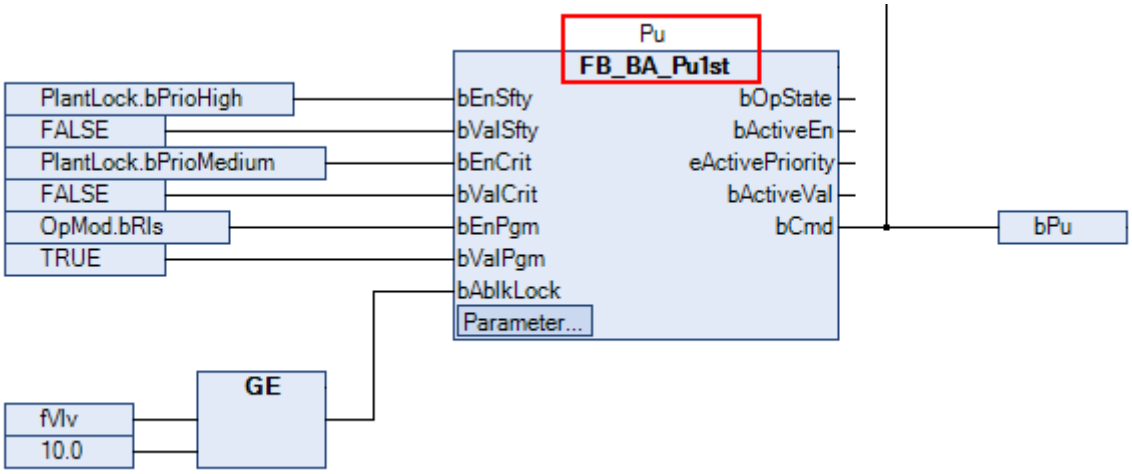
Call and instantiation of a template

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:



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

Trademark statements

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.

Third-party trademark statements

Chrome, Chromium and Google are trademarks of Google LLC.

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

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