BECKHOFF New Automation Technology

Manual | EN

# TS8040

TwinCAT 2

## Supplement | Building Automation

2024-04-08 | Version: 1.4

# 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.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

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

### Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
No claims to modify products that have already been supplied may be made on the basis of the data, diagrams, and descriptions in this documentation.

### Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

### Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

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

### Copyright



## 1.2 For your safety

### Safety regulations

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

### Exclusion of liability

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

### Personnel qualification

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

### Signal words

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

### Personal injury warnings

**⚠ DANGER**

Hazard with high risk of death or serious injury.

**⚠ WARNING**

Hazard with medium risk of death or serious injury.

**⚠ CAUTION**

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

### Warning of damage to property or environment

***NOTICE***

The environment, equipment, or data may be damaged.

### Information on handling the product

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

## 1.3 Notes on information security

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

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

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

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

# 2 Introduction

The basic prerequisite for meeting the high demands placed on building automation, such as comfort, the saving of energy, low investment and running costs and a fast return on investment, is an integrated, co-ordinated control system for the automation of all building technical systems. In TwinCAT Building Automation Beckhoff has developed a software product that reduces the engineering time and integrates all essential functions for each system of a modern building automation. Extensive software libraries and supplements continue the concept of the modular Beckhoff automation construction kit at software level as well. The new software suite essentially encompasses three basic functions:

1. TwinCAT BA PLC Libraries: basic functions for all building systems.
2. TwinCAT BA PLC Templates: function templates for all building systems.
3. TwinCAT BA Project Builder: configuration tool that links templates, hardware and BACnet objects with one another.

# 3 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.

# 4 Setup

### System requirements

- Beckhoff TwinCAT PLC V2.11 Build 2254 or higher
- Microsoft .NET Framework 4.5.2 or higher
- Windows 7 SP1, Windows 10 or higher

### Installation

1. Execute the *.exe file of the TwinCAT Supplement.
2. Click on *Continue*, read the license agreement carefully, accept it and click on *Continue*.
3. Enter user name, company name and serial number.
4. Click on *Install* to start the installation.

### Uninstallation

1. Open the Control Panel and select *Programs and Features*.
2. Select *TwinCAT Building Automation* and click on *Uninstall*.
3. Confirm with *Yes* to start the uninstallation.

# 5 Requirement profile

The user requires basic knowledge of the following.

- TwinCAT PLC-Control
- TwinCAT System Manager
- 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

# 6 Hardware requirements

The software is usable on all PC-based hardware platforms. The ideal target platforms for heating, ventilation, air conditioning and sanitary applications are the Embedded PCs from the CX series.

# 7 Basic principles and definitions

## 7.1 Sequence controller

In heating, ventilation, and air-conditioning systems, it is often the case that several actuators, working in a so-called controller sequence, are used in order to achieve a control variable.
In the air conditioning system shown below, three actuators are involved in the regulation of the supply air temperature. In the BA library, a dedicated sequence controller is instantiated for each of these actuators. During active control only one of these sequence controllers is active. The other, non-active controllers fix their control signal so that it is energetically optimal for the tempering of the supply air temperature. Depending on the control direction of the individual controller, this means either the maximum or the minimum for the control variable *lrY*.

If the effect of the active actuator (controller) is insufficient when reaching an end position, the active controller switches to the adjacent controller to the left or right.
This then takes over control. The previously active controller remains at the limit position of *lrYMax* or *rYMin*, depending on the direction of control. This is repeated with the remaining actuators until the set value or the left or right end of the sequence is reached.

In the sequence of the illustrated air conditioning system, all actuators that influence the control variable are shown from left to right. At the far left is the actuator, which enables the highest increase of the inlet air temperature, at the far right is the actuator, which enables the greatest decrease of inlet air temperature.

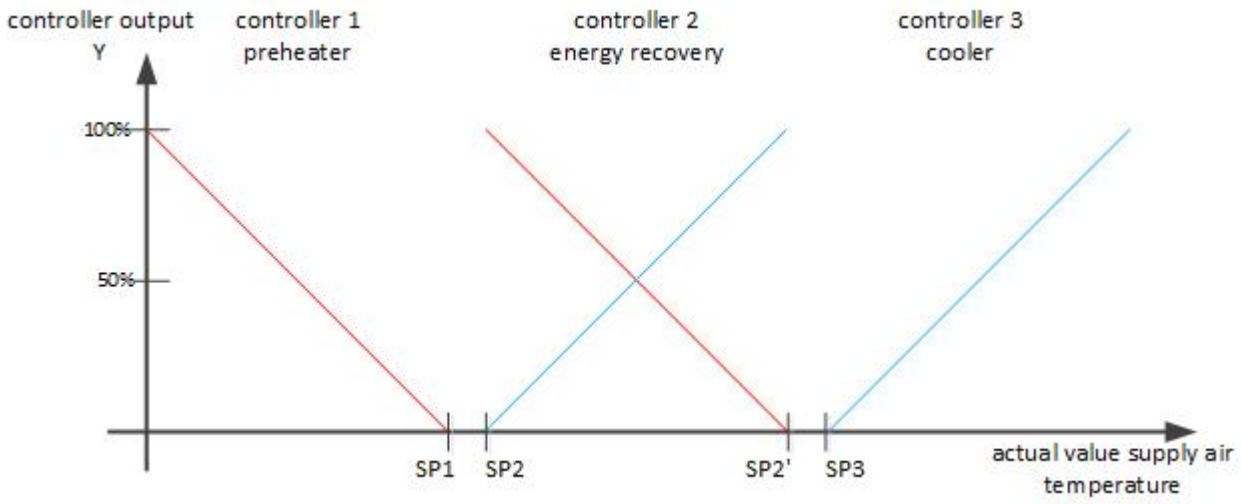
Some actuators, such as a recirculating air flap or a heat recovery unit change their direction of action during operation. (indirect = heating, direct = cooling)

Actuators with varying direction of action, such as outside air flap, recirculating air flap or heat recovery unit, are only listed once.

- Controller 1: Pre-heater
- Controller 2: Mixed air
- Controller 3: Cooler

**Schematic diagram**

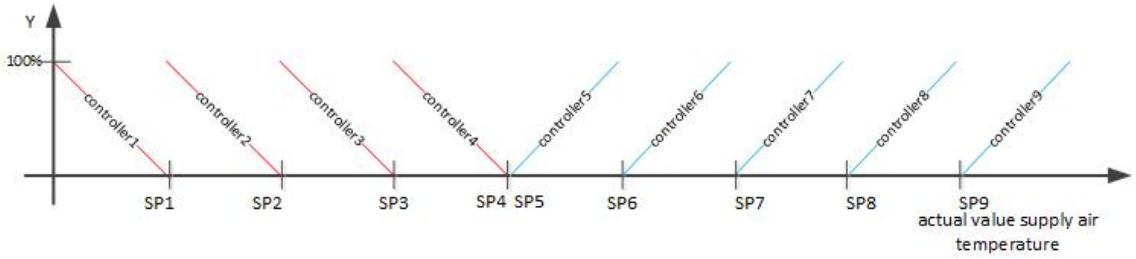
This plant is schematically represented as follows:



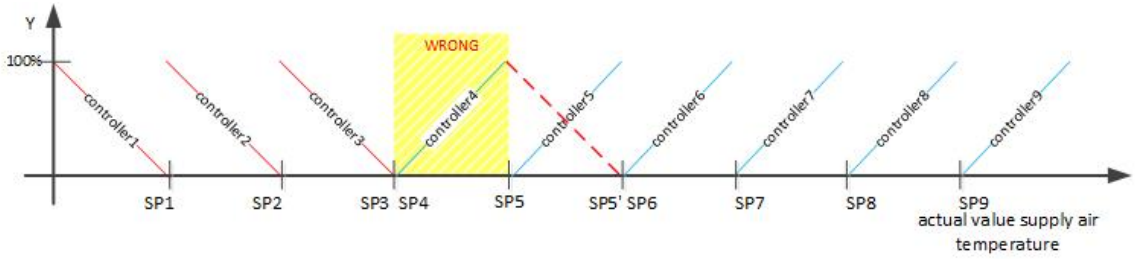

**Rules for creating a sequence**

The following rules must be followed for creating the sequences; inlet air control is used as reference:

1. The sequence controllers are numbered starting with the heating sequences with low ordinal numbers to the cooling sequences with high ordinal numbers.

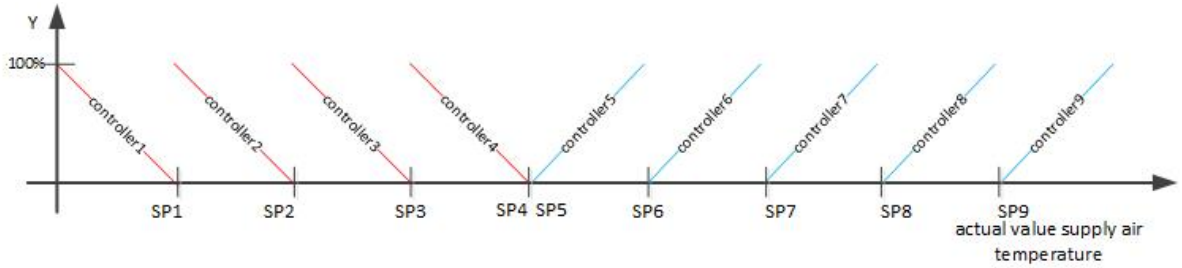



2. A series of heating sequences should not include a cooling sequence. Conversely, a series of cooling sequences should not contain a heating sequence. Sequences with reversal of direction of action for a mixed air system or heat recovery should be positioned between the heating and cooling sequences.




In this diagram controller 4 would be placed incorrectly, if controller 5 changed to heating mode. Or: Controller 4 is correct, but controller 5 would have to be a pure cooling controller. In both cases there would be two switches from heating to cooling.

3. The setpoints within the sequence must be monotonically increasing. This requirement is a result of the switching behaviour explained above: If the setpoint of a controller with a lower number is higher than the next higher one, the result could be continuous switching between the two controllers. As mentioned above, controllers with the same direction of action usually have the same setpoint.




SP1 ≤ SP2 ≤ SP3 ≤ SP4 ≤ SP5 ≤ SP6 ≤ SP7 ≤ SP8 ≤ SP9

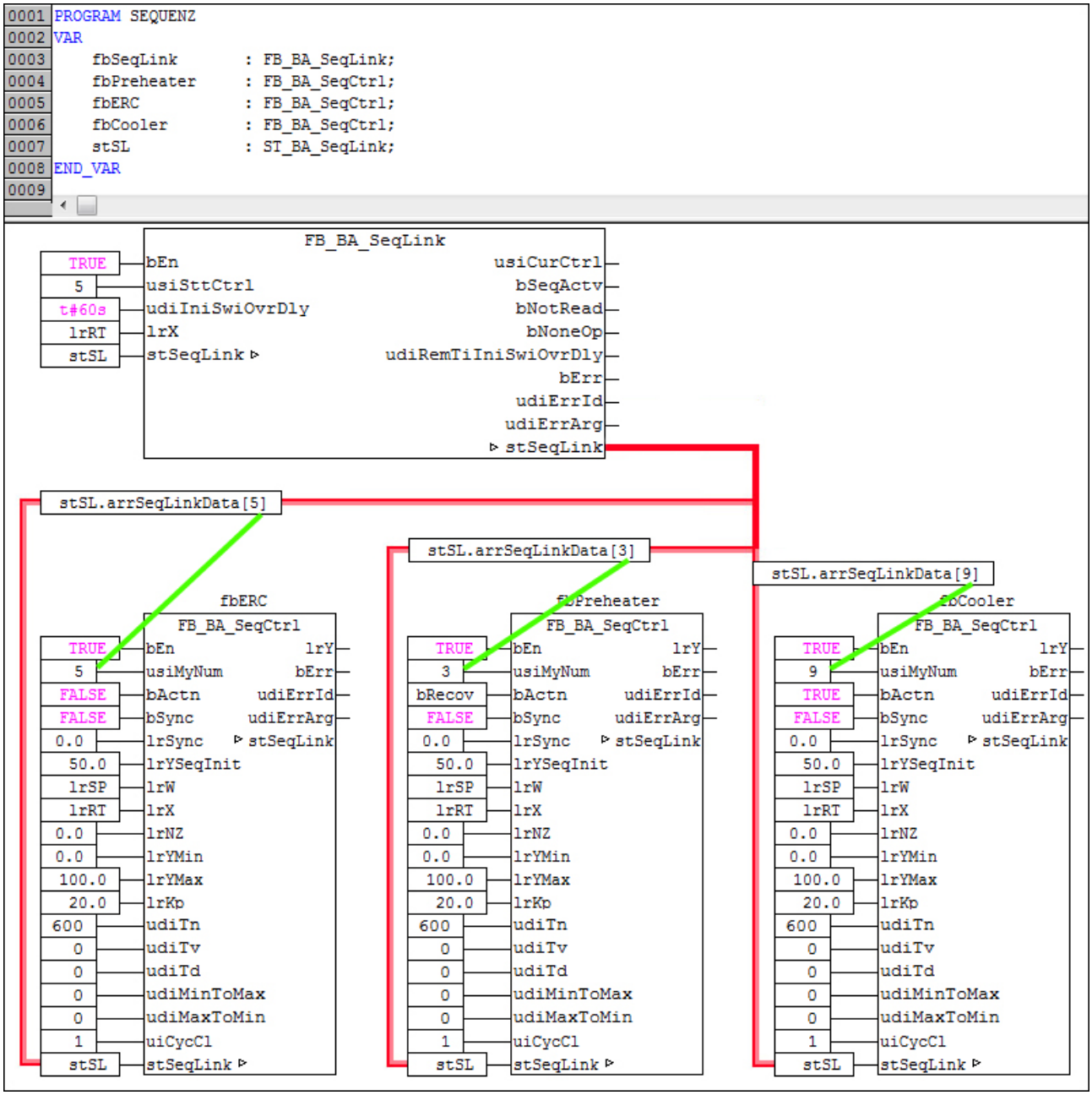
**Sequence controllers in the PLC**

The BA library provides two function blocks for realising a sequence control in the PLC program:

- The function block FB_BA_SeqCtrl [▶ 165]. This function block provides an individual controller as part of a sequence of up to 16 controllers.
- The function block FB_BA_SeqLink [▶ 168]. This function block is the control function block of the sequence and therefore only exists once per sequence. It decides which controller of the sequence is currently active and checks the sequence for certain error states, such as duplicate allocation of ordinal number at the controllers.

The structure variable ST_BA_SeqLink [▶ 327] is used to link the sequence controllers with the sequence linker FB_BA_SeqLink [▶ 168].
This structure variable has to be declared once per sequence control.

The sequence control is enabled at input *bEn* of the function block FB_BA_SeqLink [▶ 168]. The variable *usiStartCtrl* is used to determine which controller is used as the first one after the start of automatic control mode. In the example, the sequence controller with the no. 5 is assigned as the start controller. Switching from controller 5 to another controller in the sequence is locked for 60 seconds after the restart of the control system based on the value of the input variable *tIniSwiOvrDly*.




## 7.2 Shading correction

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:

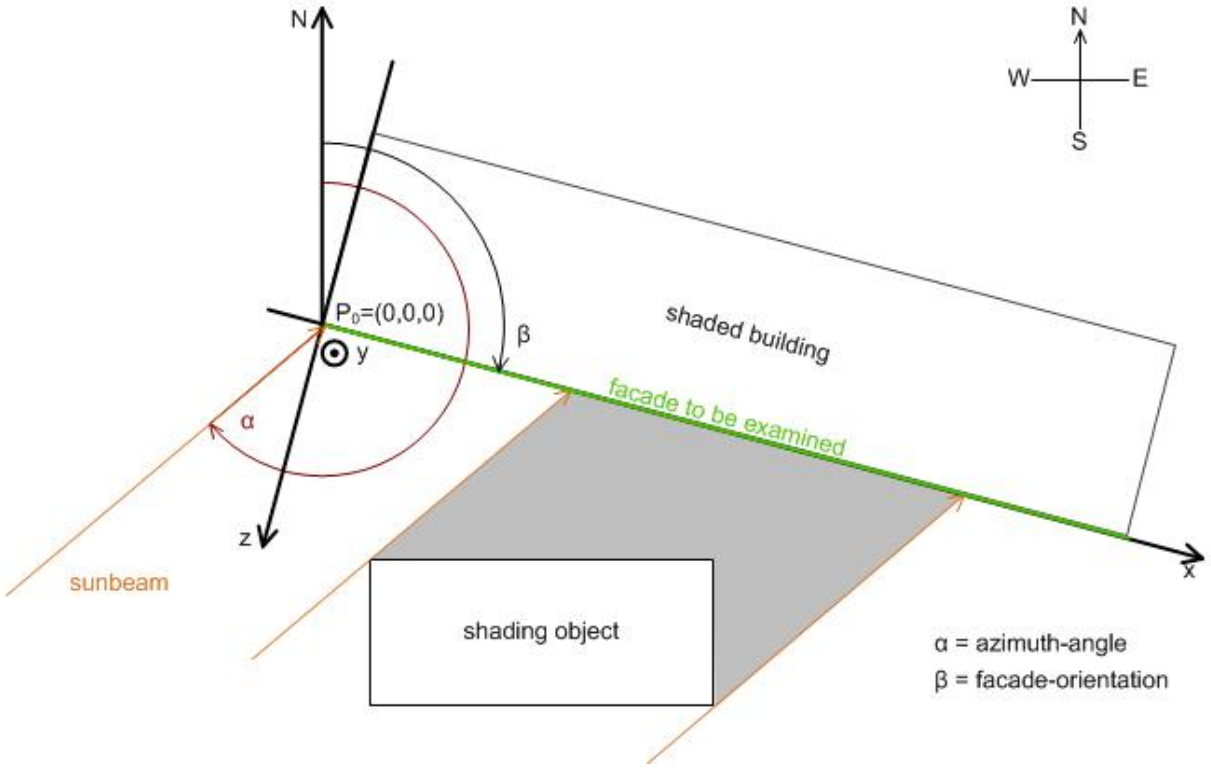
- the orientation of the facade
- the position of the windows

- 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



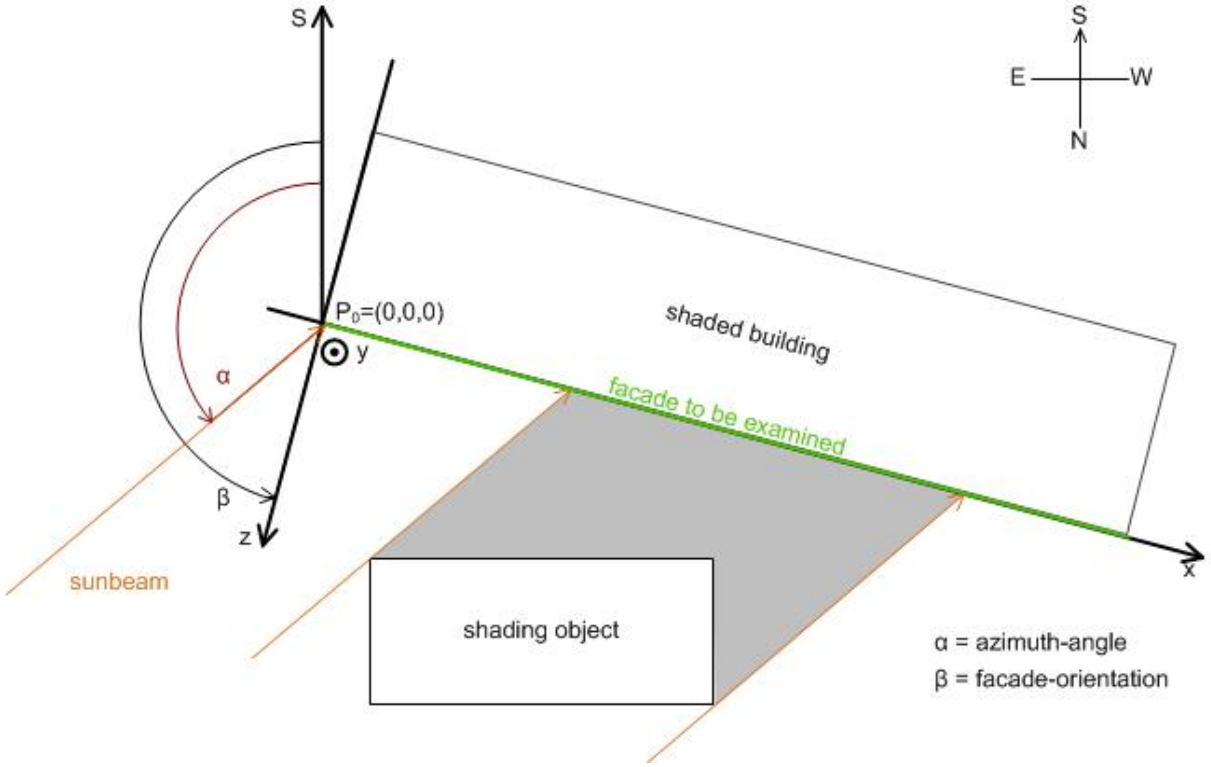

For the pure observation of the shadow thrown on the facade, a two-dimensional coordinate system is ultimately required, therefore the x and y axis were placed on the facade. The zero point is thereby at the bottom left on the base, as if one were regarding the facade from the front. For the calculation 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:

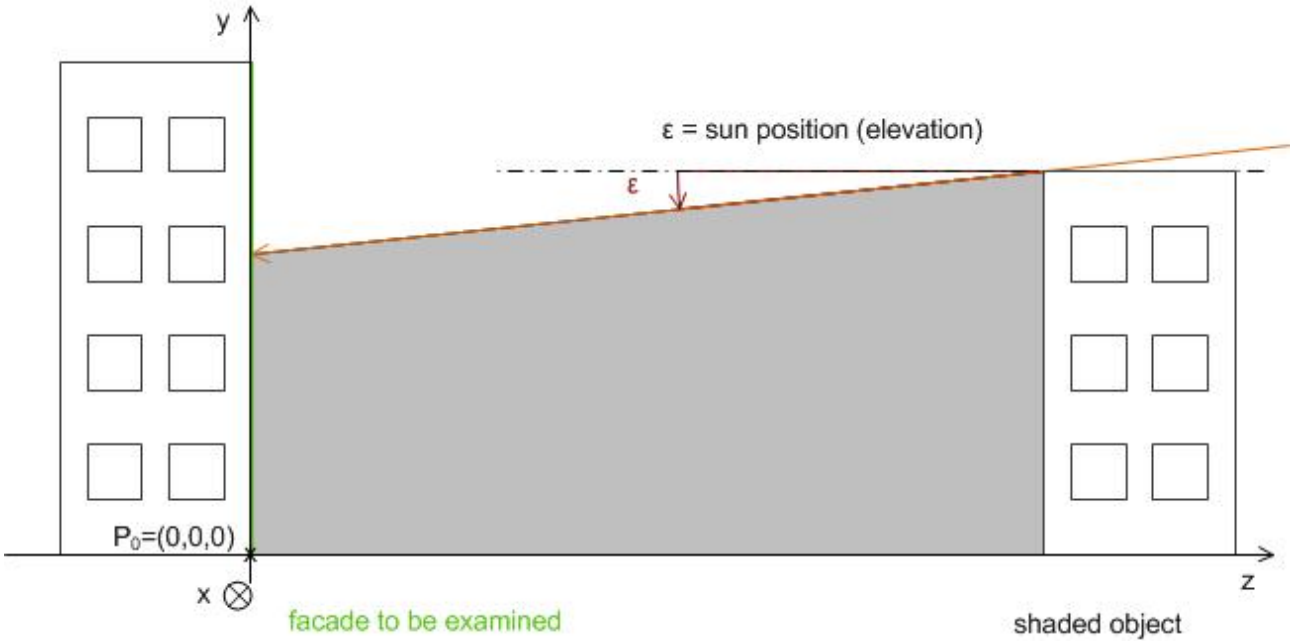
| Line of sight | Facade orientation |
|---|---|
| South | β=0° |
| East | β=90° |
| North | β=180° |
| West | β=270° |




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 [▶ 278] (shading correction) is parameterised they are activated through a boolean input, *bSouth*

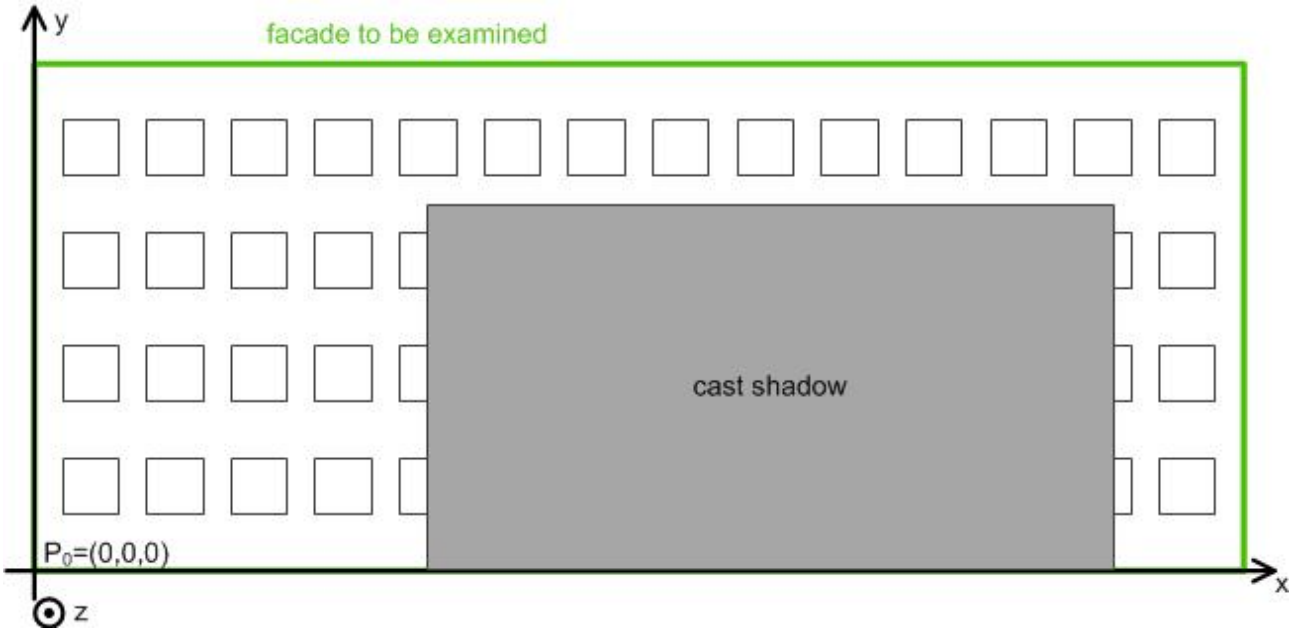
The following two illustrations are intended to further clarify the position of the point of origin $P_0$ as well as the orientation of the coordinate system:

**Observation from the side**




The angle of elevation (height of the sun) can be represented using this illustration: 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.

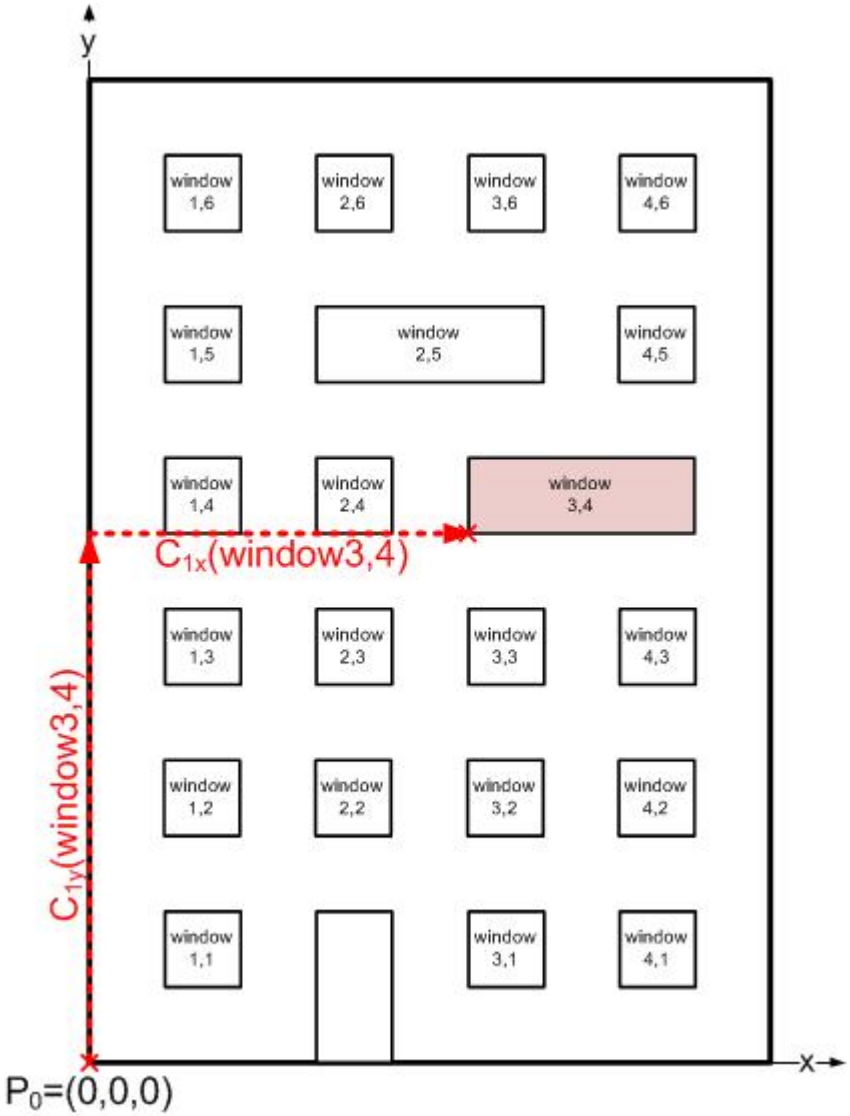
**Observation from the front**



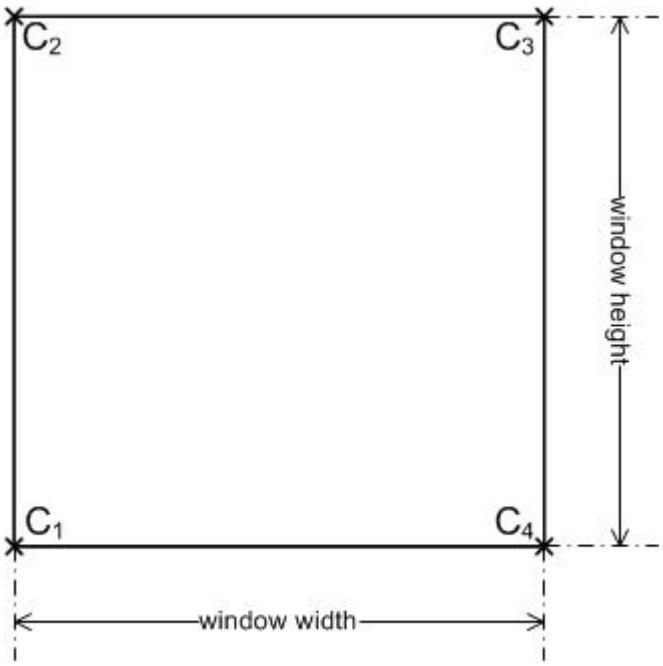

Here, the position of the point of origin, $P_0$, at the bottom left base point of the facade is once more very clear. Beyond that the X-Y orientation is illustrated, which is important later for the entry of the window elements.

**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.




The width and height must additionally be specified.



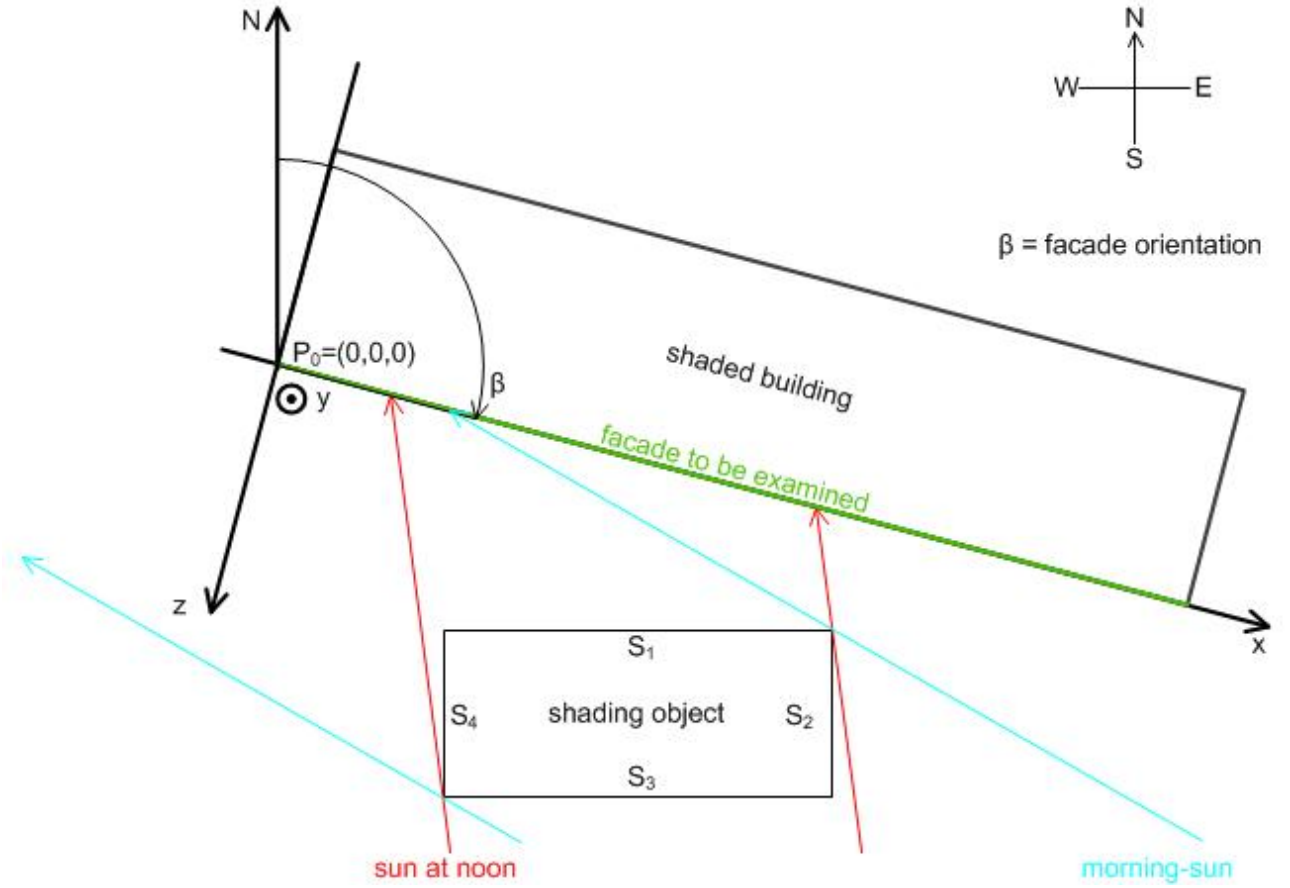

$C_n$=corner n

The position of each window corner on the facade is determined internally from the values entered. A window is in the shade if all corners lie in the shade.

**Positioning of the shading objects**

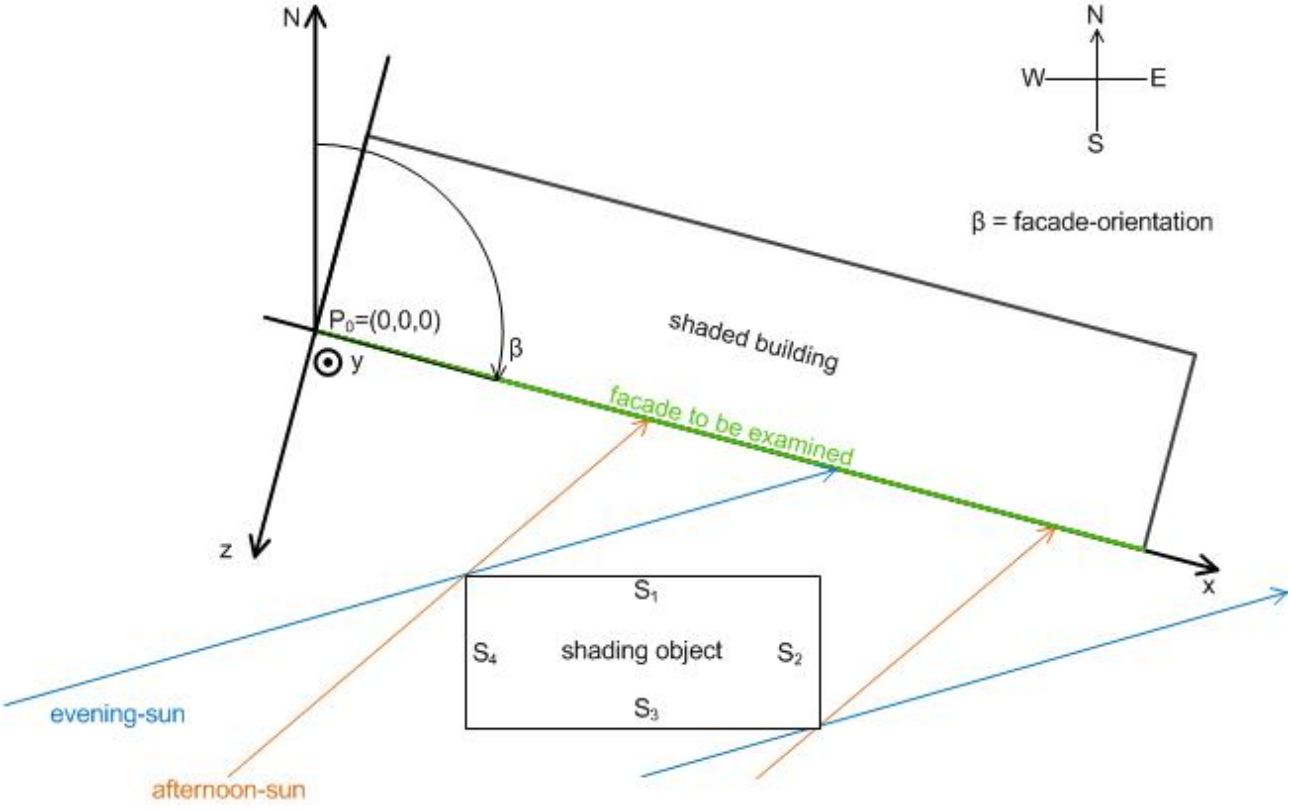
When describing the shading objects, distinction is made between angular objects (building, column) and objects that are approximately spherical (e.g. trees). Angular objects can be categorised according to the shadow they cast into square, shadow-casting facades, wherein one must consider which ones cast the main shadows over the course of the day:

**Morning/midday**

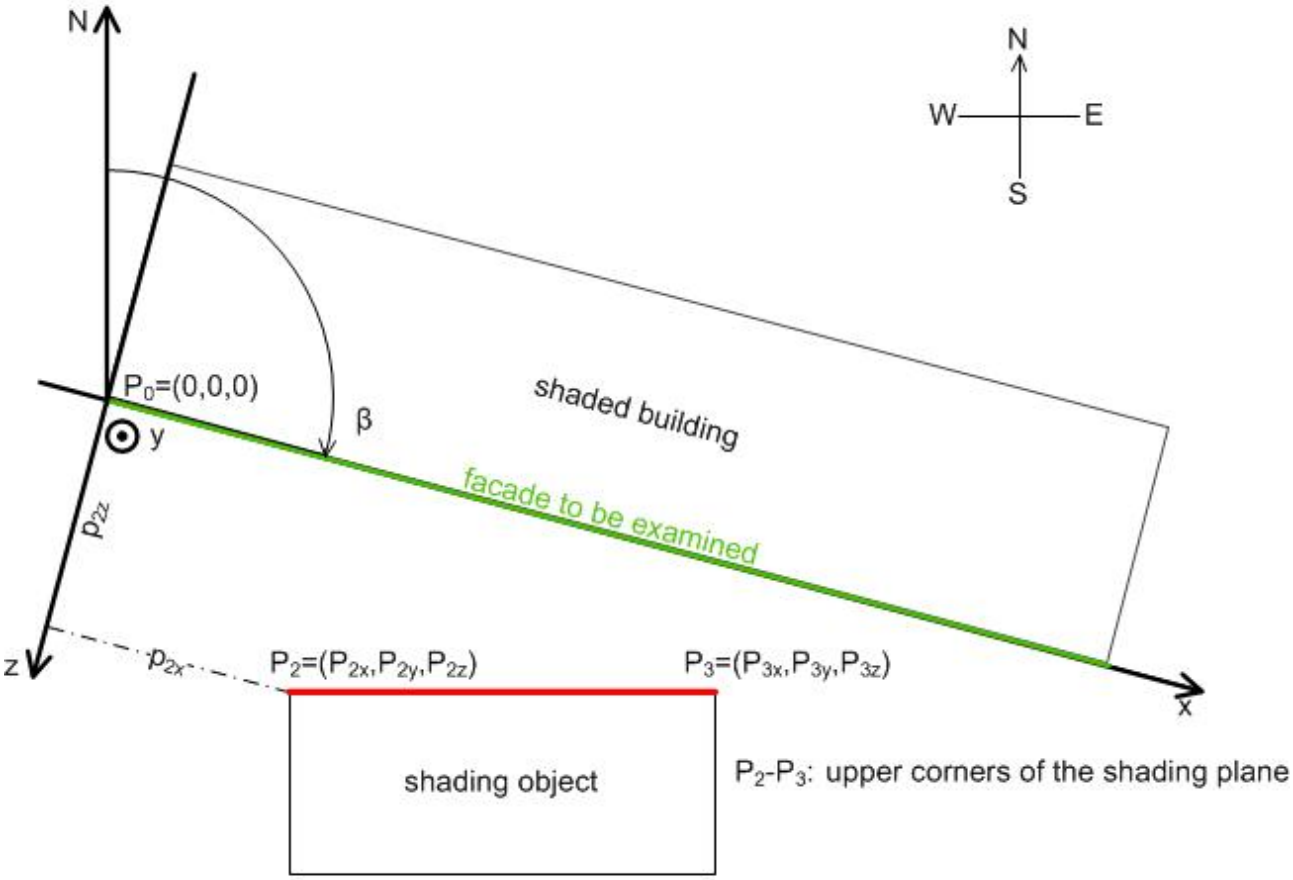



In the morning and at midday, the shadows are mainly cast by the sides $S_1$ and $S_4$; $S_2$ and $S_3$ need not be considered if they are not higher.

**Afternoon / evening**



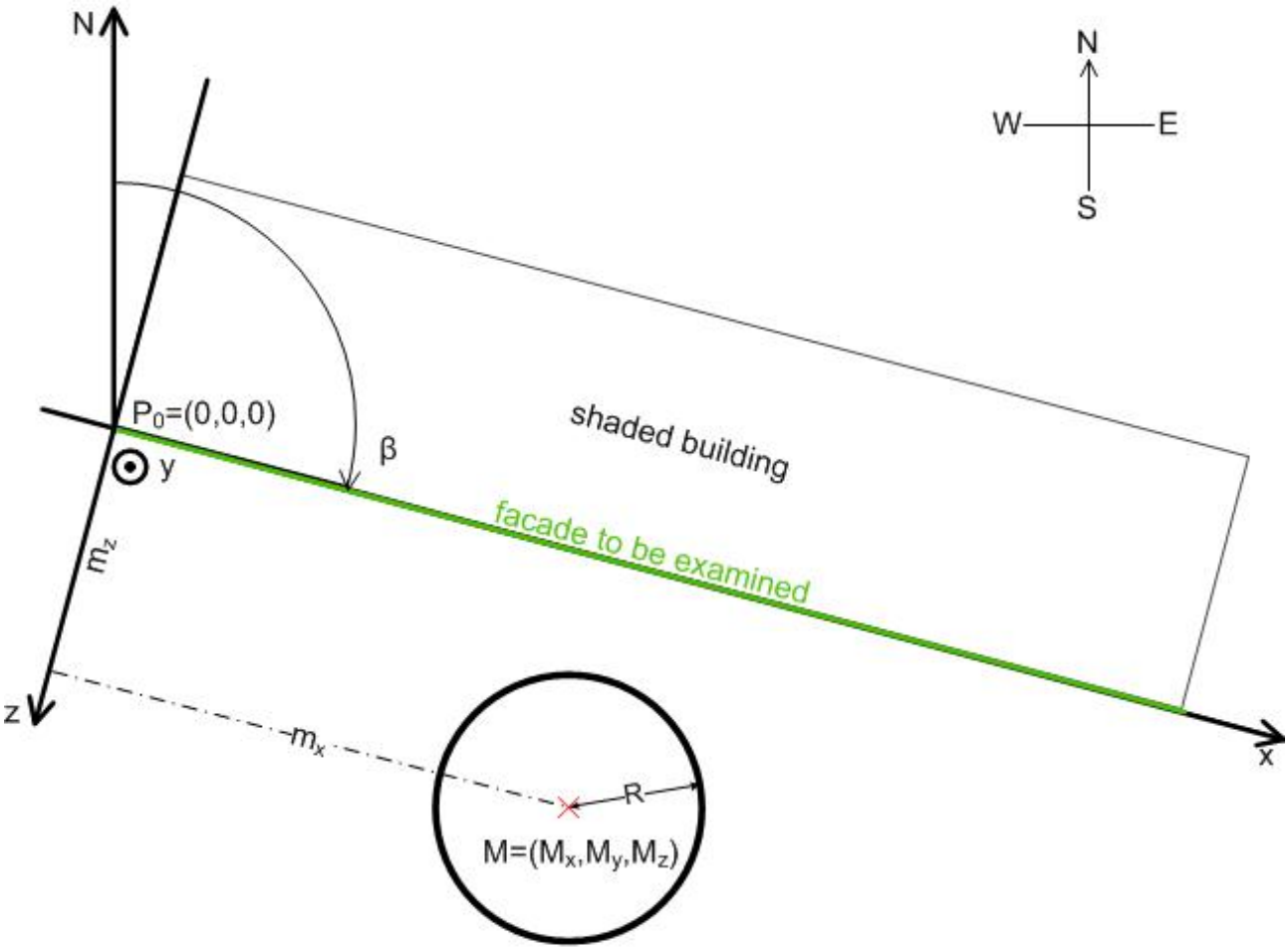
In the afternoon and evening, the total shade can be determined solely through $S_1$ and $S_2$. In this case it is therefore sufficient to specify $S_1$, $S_2$ and $S_4$ as shadow casters. The entry is made based on the four corners or their coordinates in relation to the zero point of the facade:




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.

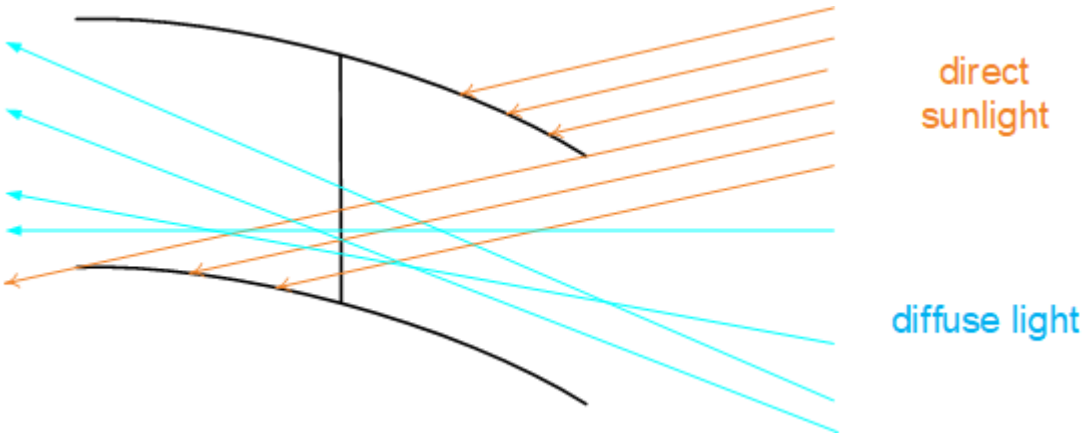
## 7.3 Sun protection: Basic principles and definitions

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.

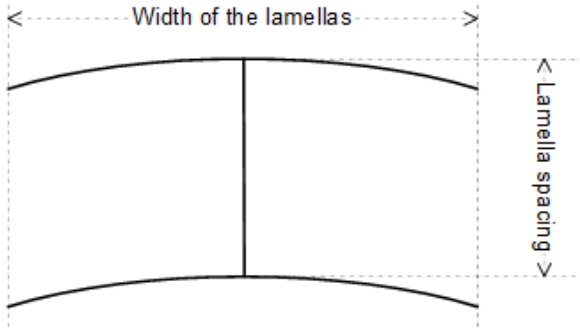
This relationship can easily be illustrated with a set square positioned upright on the table: Viewed directly from the side you can see a triangle with two 45° angles and one 90° angle. If the triangle is rotated, the side on the table appears to become shorter and the two original 45° angles change. 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.

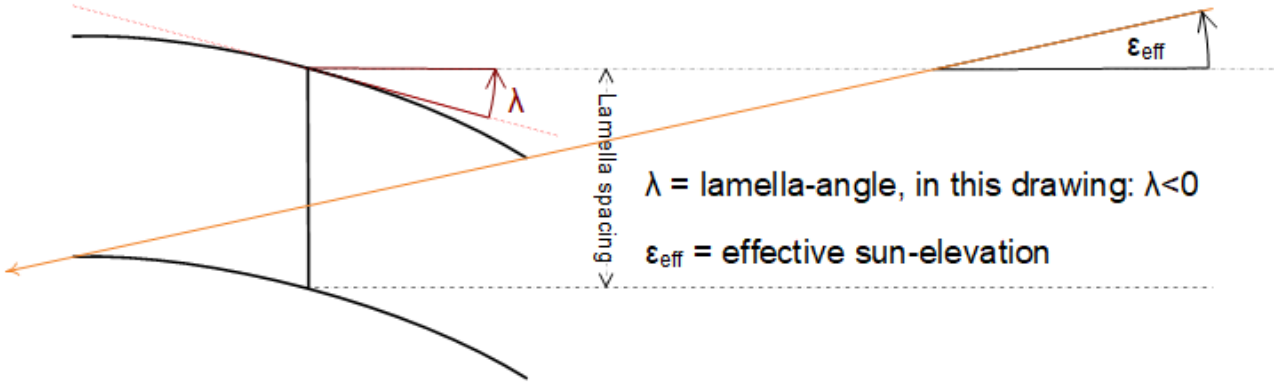
The following three images illustrate the relationship between the effective elevation angle and the blind dimensions, and how the resulting lamella angle λ changes during the day:

## Lamella-angle

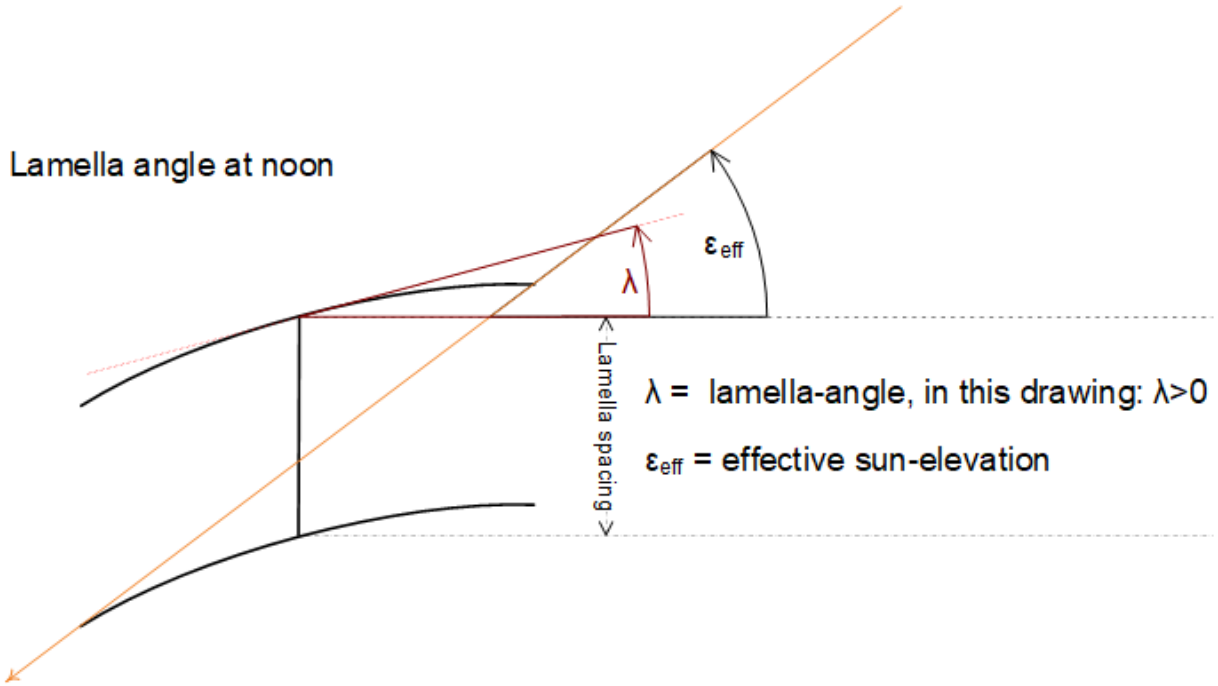
### Lamella at an angle of λ=0




### Lamella-angle in the morning and in the evening



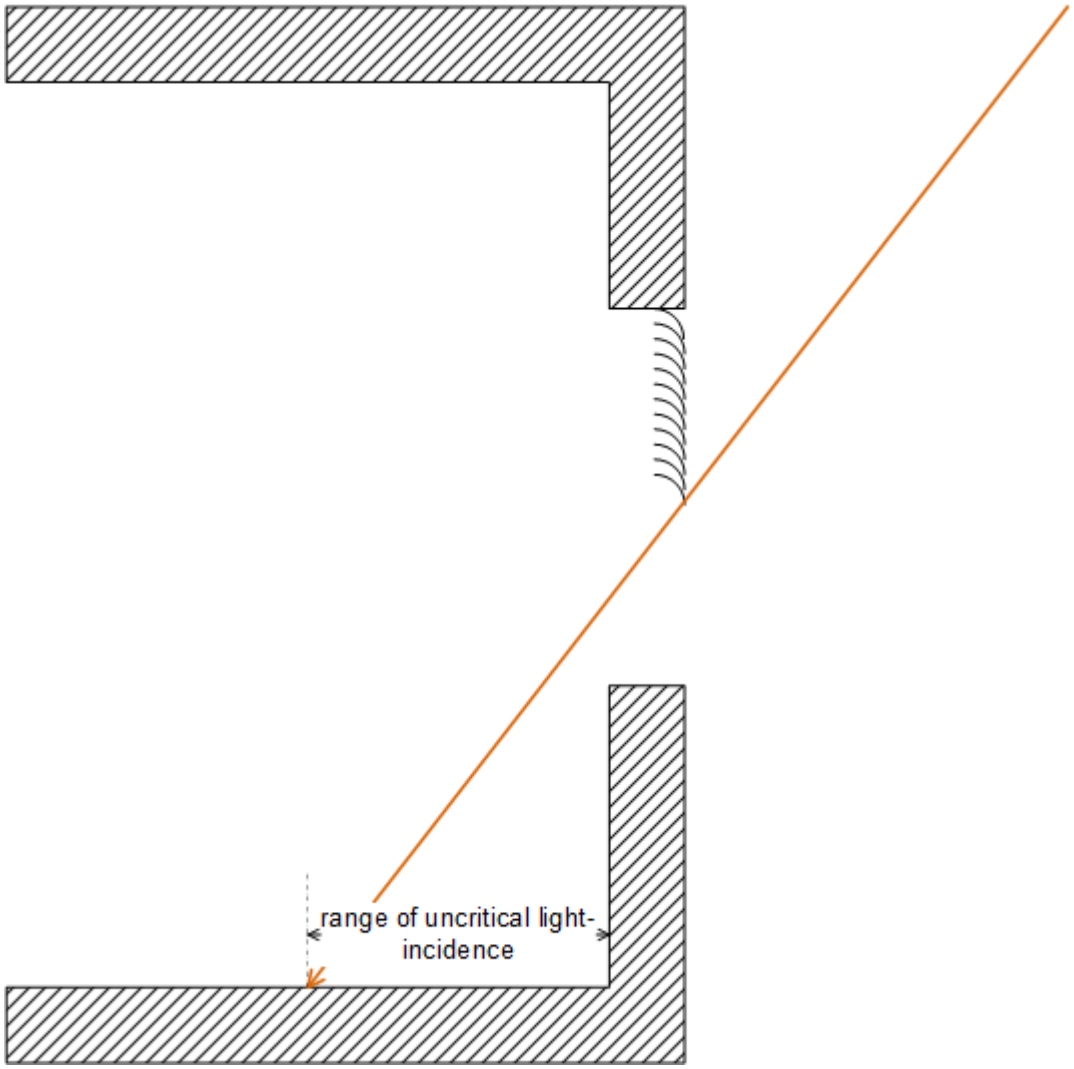

### Lamella angle at noon




More detailed information on this topic can be found in the chapter Effective elevation angle [▶ 35].

### 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.

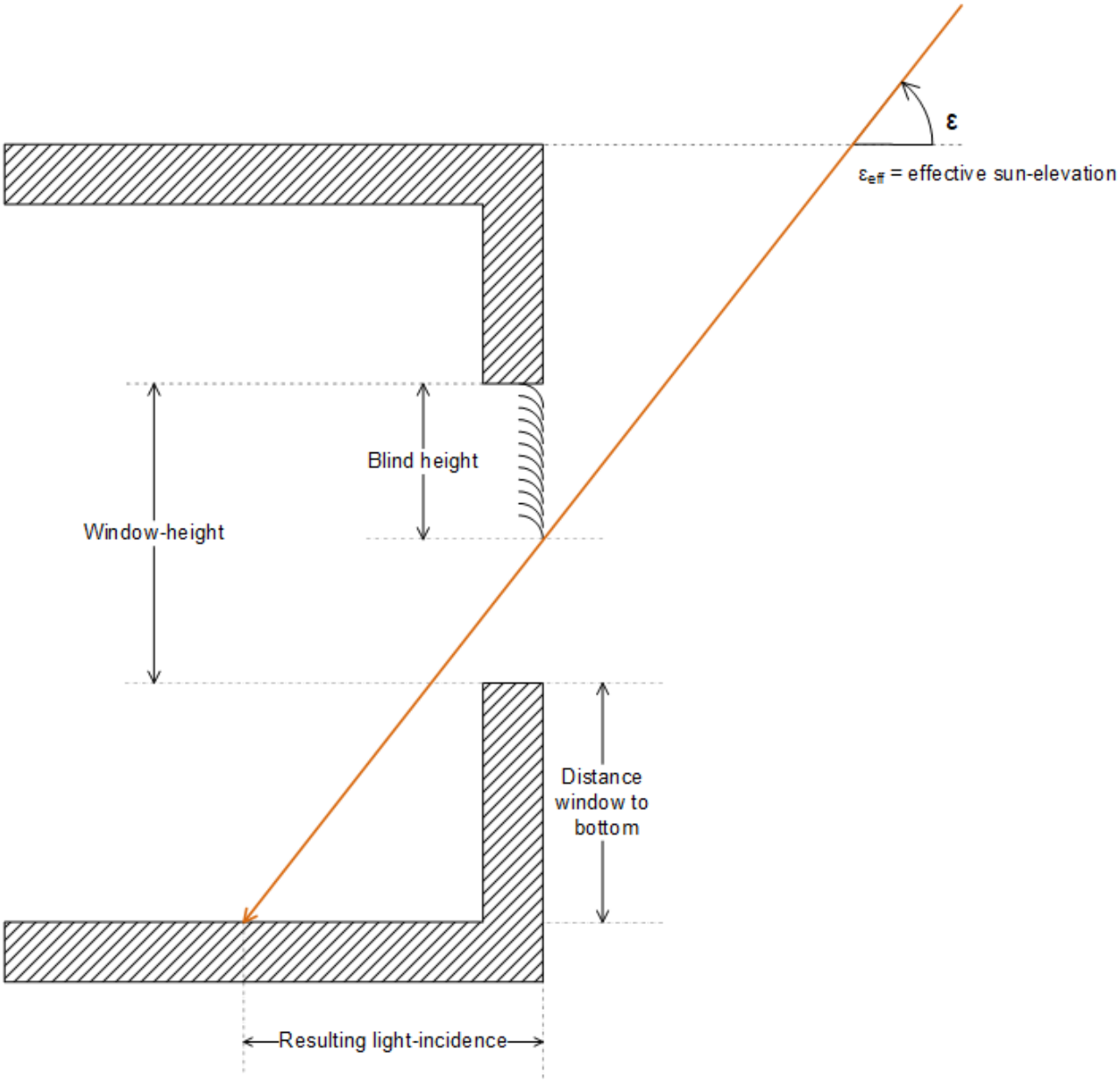



In order to be able to calculate at any time the appropriate blind height that guarantees that the incidence of sunlight does not exceed a certain value, the following values are necessary.

Required for the calculation of the respective blind height:

- Height of the sun (elevation)
- Window height
- Distance between the window and the floor

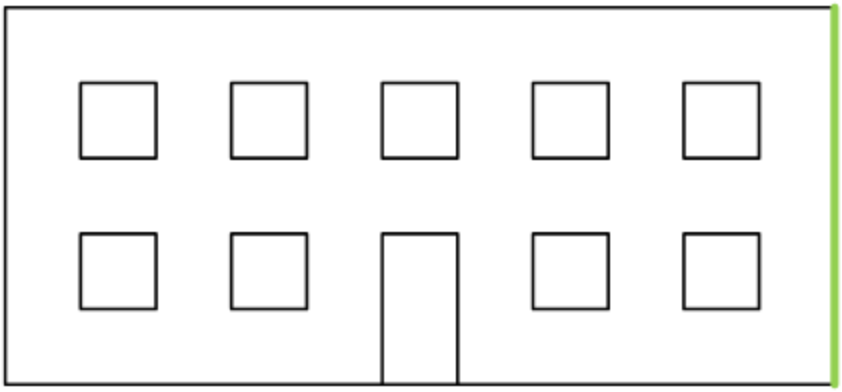
The following illustration shows where these parameters are to be classified:



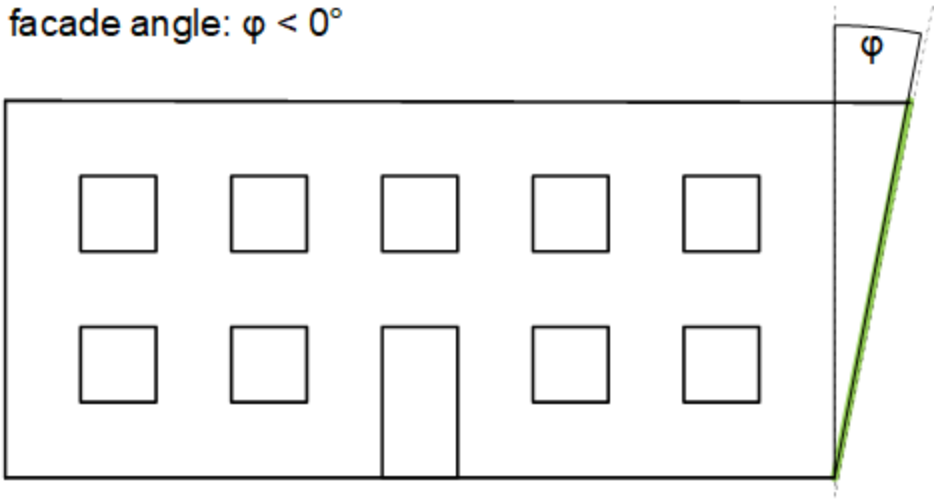

**Influence of the facade inclination**

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:

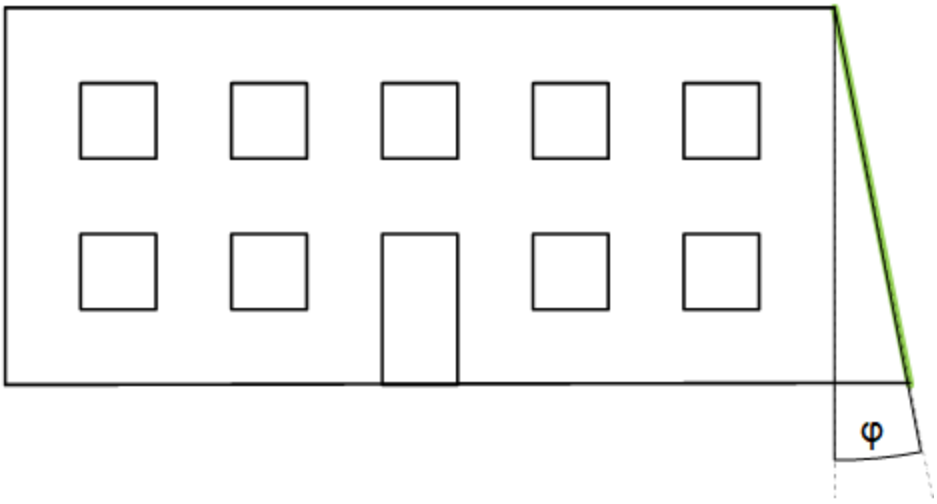
facade angle: φ = 0°



facade angle: φ < 0°

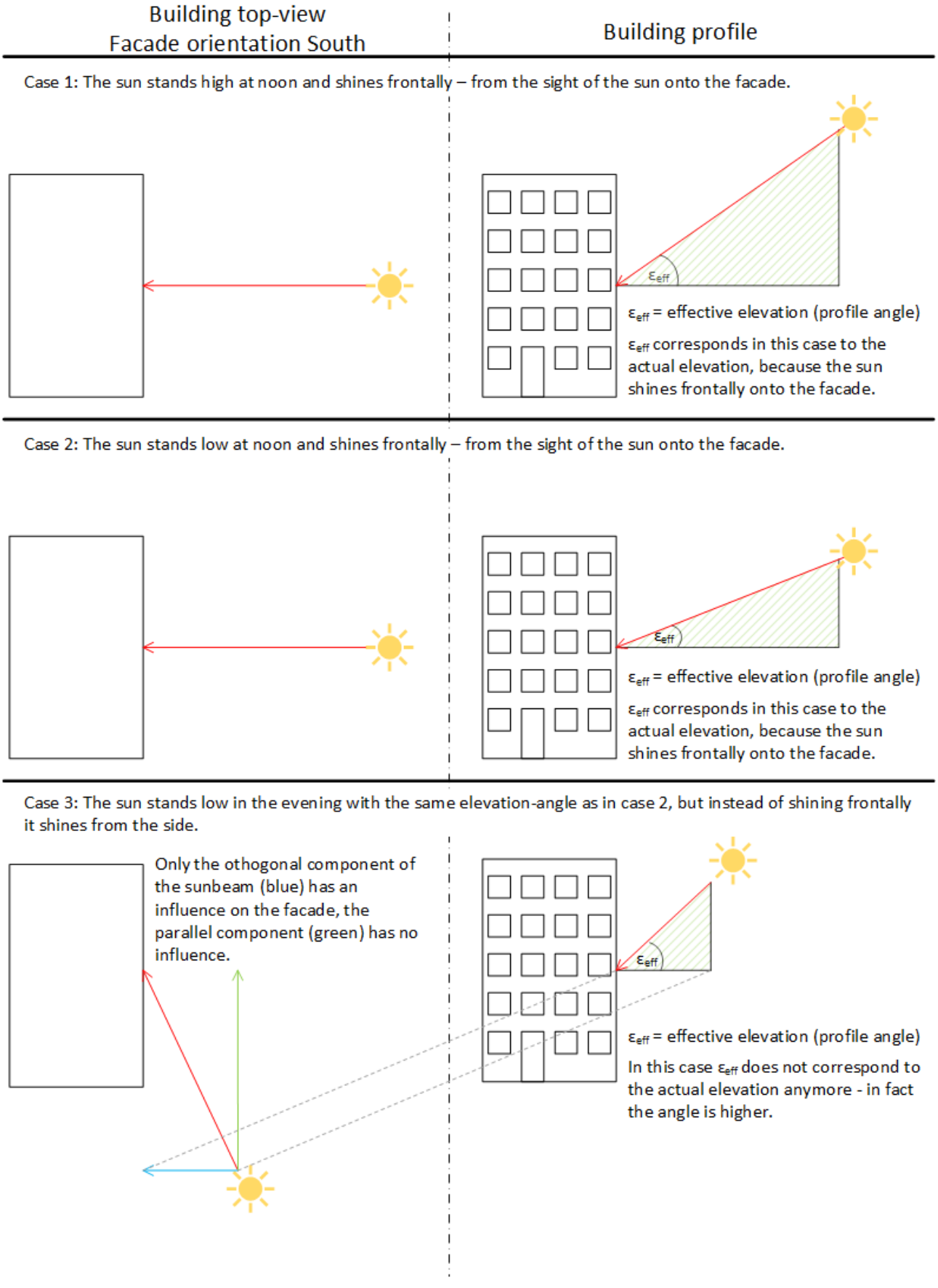



facade angle: φ > 0°



### 7.3.1 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 $\varepsilon_{eff}$ 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 $\varepsilon_{eff}$ 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.

# 8 TcBA PLC library

By using the TcBA library, 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 library.

The object-oriented encapsulation of the building automation functions offers the following benefits:

- Fast creation of system programs.
- Fast parameterization and commissioning of the systems.
- Guarantee of a very large range of system functions at all times.
- Transparency of programs (prerequisite for long-term maintainability and expandability of the systems).
- Once created, good reusability of templates for systems or system subassemblies.
- Easy training of staff.
- Straightforward extension and modification of existing systems.
- Specifications for a clear, object-oriented structure for the creation of visualization objects in MMI and SCADA systems.
- Programs are easier to document.

## 8.1 General Information

**Further libraries required**

For PC systems (x86) and Embedded PCs (CXxxxx):

- Standard.lib
- TcBase.lib
- TcSystem.lib
- TcUtilities.lib
- TcTestAndSet.lib
- TcBACnetRev12.lib

For Bus Terminal Controllers of the BCxx00, BCxx50, BCxx20, BC9191 and BXxx00 series:

- not available

**Memory usage**

Some of the PLC program memory is already used up by integrating the library. Depending on the application program, therefore, the remaining memory may not be sufficient.

## 8.2 Function blocks

**BACNet**

| Name | Description |
|---|---|
| FB_BACnetAI1201 [▶ 46] | Analog input. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetAI1203 [▶ 49] | Analog input. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetAO1201 [▶ 51] | Analog output. BACnet object-generating function block for additional parameterization from the PLC. Small version. |

| Name | Description |
|---|---|
| FB_BACnetAO1203 [▶ 53] | Analog output. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetAV1201 [▶ 59] | Analog value. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetAV1202 [▶ 60] | Analog value. BACnet object-generating function block for additional parameterization from the PLC. Medium version. |
| FB_BACnetAV1203 [▶ 63] | Analog value. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetAV1204 [▶ 67] | Analog value. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetAVDisplay [▶ 69] | Analog value. BACnet object generating function block which can be used to display a value from the PLC in BACnet. |
| FB_BACnetAVSetpoint [▶ 69] | Analog value. BACnet object generating function block which maps the BACnet Property Present Value in the PLC. |
| FB_BACnetBI1201 [▶ 71] | Binary input. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetBI1203 [▶ 72] | Binary input. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetBI1205 [▶ 74] | Binary input. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetBO1201 [▶ 76] | Binary output. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetBO1202 [▶ 78] | Binary output. BACnet object-generating function block for additional parameterization from the PLC. Medium version. |
| FB_BACnetBO1203 [▶ 81] | Binary output. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetBO1205 [▶ 84] | Binary output. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetBV1201 [▶ 86] | Binary value. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetBV1202 [▶ 88] | Binary value. BACnet object-generating function block for additional parameterization from the PLC. Medium version. |
| FB_BACnetBV1203 [▶ 90] | Binary value. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetBV1204 [▶ 94] | Binary value. BACnet object-generating function block for additional parameterization from the PLC. Large version. |

| Name | Description |
|---|---|
| FB_BACnetBVDisplay [▶ 95] | Binary value. BACnet object generating function block which can be used to display a value from the PLC in BACnet. |
| FB_BACnetBVSetpoint [▶ 96] | Binary value. BACnet object generating function block which maps the BACnet Property Present Value in the PLC. |
| FB_BACnetCAL1201 [▶ 97] | Calendar. BACnet object-generating function block for additional parameterization from the PLC. |
| FB_BACnetLoop1201 [▶ 98] | PID controller. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetLoop1202 [▶ 102] | PID controller. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetLoopSeq1201 [▶ 106] | PID sequence controller. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetLoopSeq1202 [▶ 110] | PID sequence controller. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetMI1203 [▶ 114] | Multistate input. BACnet object-generating function block for additional parameterization from the PLC. |
| FB_BACnetMO1202 [▶ 116] | Multistate output. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetMO1203 [▶ 118] | Multistate output. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetMV1201 [▶ 122] | Multistate value. BACnet object-generating function block for additional parameterization from the PLC. Small version. |
| FB_BACnetMV1202 [▶ 123] | Multistate value. BACnet object-generating function block for additional parameterization from the PLC. Medium version. |
| FB_BACnetMV1203 [▶ 126] | Multistate value. BACnet object-generating function block for additional parameterization from the PLC. Large version. |
| FB_BACnetMVDisplay [▶ 129] | Multistate value. BACnet object generating function block which can be used to display a value from the PLC in BACnet. |
| FB_BACnetMVSetpoint [▶ 130] | Multistate value. BACnet object generating function block which maps the BACnet Property Present Value in the PLC. |
| FB_BACnetSchedB1201 [▶ 131] | Time schedule with boolean output values. BACnet object-generating function block for additional parameterization from the PLC. |
| FB_BACnetSchedBinPV [▶ 132] | Schedule with Boolean output values of the type Binary Present Value with the function "Pre-calculating switch-on and switch-off time". BACnet object-generating function block for additional parameterization from the PLC. |
| FB_BACnetSchedR1201 [▶ 133] | Time schedule with REAL output values. BACnet object-generating function block for additional parameterization from the PLC. |

| Name | Description |
|---|---|
| FB_BACnetSchedUdi1201 [▶ 134] | Time schedule with integer output values. BACnet object-generating function block for additional parameterization from the PLC. |
| FB_BACnetTLog1201 [▶ 135] | Trend-Log. BACnet object-generating function block for additional parameterization from the PLC. |

### Control

| Name | Description |
|---|---|
| FB_BA_Anlg3Pnt [▶ 136] | Analog value for three-point converters |
| FB_BA_Cont4Stp_01 [▶ 137] | Step switch with 4 levels |
| FB_BA_RampLmt [▶ 141] | Ramp limitation |
| FB_BA_SldgLmtMonit [▶ 143] | Sliding limit value monitoring |
| FB_BA_StpDly [▶ 145] | Delayed output of switching stages |
| FB_BA_Swi2P [▶ 146] | Two-point switch |
| FB_BA_Swi2P_Dly [▶ 147] | Two-point switch with delay. |
| FB_BA_SwiHys2P [▶ 150] | Two-point switch around a switching point |
| FB_BA_SwiHys2P_Dly [▶ 151] | Two-point switch around a switching point with delay. |
| FB_BA_SwiMonit [▶ 153] | Monitoring of limit switches, e.g. for a two-point damper |

### Control

| Name | Description |
|---|---|
| FB_BA_FltrPT1 [▶ 154] | First order filter |
| FB_BA_PIDCtrl [▶ 155] | Universal PID controller |
| FB_BA_PIDCtrlEx [▶ 159] | Extended PID controller with parallel PID structure or preceding P element, only available under Tc2_BA! |
| FB_BA_PISync1201 [▶ 162] | PI synchronization of two controllers of type FB_BACnetLoop1201 [▶ 98] |
| FB_BA_PISync1202 [▶ 163] | PI synchronization of two controllers of type FB_BACnetLoop1202 [▶ 102] |
| FB_BA_PWM [▶ 163] | Pulse width modulation function block |
| FB_BA_SeqCtrl [▶ 165] | Sequence controller, for introductory explanatory notes regarding sequence controllers see here [▶ 19]. |
| FB_BA_SeqLink [▶ 168] | Sequence controller control function block |

### Mathematical functions

| Name | Description |
|---|---|
| FB_BA_Chrct02 [▶ 171] | Linear interpolation for 2 interpolation points |
| FB_BA_Chrct04 [▶ 172] | Linear interpolation for 4 interpolation points |
| FB_BA_Chrct07 [▶ 174] | Linear interpolation for 7 interpolation points |
| FB_BA_Chrct32 [▶ 176] | Linear interpolation for 32 interpolation points |
| FB_BA_TiAvrg [▶ 177] | Arithmetic average over time |

### Messages

| Name | Description |
|---|---|
| FB_BA_Alarm [▶ 179] | Collective alarm function block with selectable alarm memory and internal acknowledgement |
| FB_BA_AlarmMgnr [▶ 181] | The function block collects the alarms of all plants and pools them in a group alarm |
| FB_BA_AlarmPlt [▶ 183] | The function block collects the alarms of a plant and pools them in a group alarm |
| FB_BA_Alm [▶ 185] | Alarm function block with selectable alarm memory and acknowledgement |
| FB_BA_AlmColt04 [▶ 186] | Collective alarm function block, 4 alarms |
| FB_BA_AlmColt08 [▶ 188] | Collective alarm function block, 8 alarms |
| FB_BA_AlmColt12 [▶ 190] | Collective alarm function block, 12 alarms |
| FB_BA_AlmColt16 [▶ 193] | Collective alarm function block, 16 alarms |
| FB_BA_ComnMsg [▶ 197] | Formation of collective messages |
| FB_BA_ComnMsgTermt [▶ 200] | Global collective function block for collective messages (FB_BA_ComnMsg [▶ 197]) |

### Operators

| Name | Description |
|---|---|
| FB_BA_DMUX_XX [▶ 203] | Demultiplexer function blocks |
| FB_BA_MMux_XX [▶ 205] | The function block activates an input value on the output, depending on a selector and the corresponding input selector condition |
| FB_BA_MultiCalc_XX [▶ 208] | Multi-calculation function blocks |
| FB_BA_MUX_XX [▶ 210] | Multiplexer function blocks |
| FB_BA_PrioSwi_XX [▶ 213] | Priority switch |

### Plant control

| Name | Description |
|---|---|
| FB_BA_AntBlkg [▶ 215] | Blocking protection for pump or actuators |
| FB_BA_DHW2P [▶ 216] | Charge control for a hot water tank via an on-off controller |
| FB_BA_FIFO04 [▶ 218] | Sequential control of up to four aggregates |
| FB_BA_FIFO08 [▶ 219] | Sequential control of up to eight aggregates |
| FB_BA_FnctSel [▶ 221] | Function selection (heating and/or cooling) in two- or four-pipe network |
| FB_BA_FrstPrtc [▶ 224] | Monitoring of frost alarm and emergency heating |
| FB_BA_HX [▶ 227] | Calculation of dew point temperature, specific enthalpy and absolute humidity |
| FB_BA_LglPrev [▶ 228] | Function block for disinfecting service water and destroying legionella |
| FB_BA_LmtCtrl [▶ 230] | Function block for determining the limits and its enable status |
| FB_BA_NgtCol [▶ 233] | Summer night cooling |
| FB_BA_RcvMonit [▶ 234] | Function block for calculating the efficiency of an energy recovery system |
| FB_BA_RmTAdj [▶ 237] | Adjustment of the room temperature setpoint |
| FB_BA_SpRmT [▶ 240] | Adjustment of the room temperature setpoint |

| Name | Description |
|---|---|
| FB_BA_SpSupVis [▶ 243] | Function block for processing and checking the lower and upper setpoint of a supply air humidity or temperature control |
| FB_BA_StepCtrl08 [▶ 246] | Step sequence function block, 8 steps |
| FB_BA_StepCtrl12 [▶ 249] | Step sequence function block, 12 steps |

**Room functions; shading**

Overview of shading correction [▶ 44]

Shading correction: basics principles and definitions [▶ 21]

Overview of automatic sun protection [▶ 45]

Sun protection: basics principles and definitions

| Name | Description |
|---|---|
| FB_BA_BldPosEntry [▶ 254] | Sun protection function: input of blind positions. |
| FB_BA_BrtnsHysDly [▶ 256] | Threshold switch for brightness |
| FB_BA_CalcSunPos [▶ 257] | Calculation of sun position |
| FB_BA_FcdElemEntry [▶ 258] | Shading correction: input of facade elements per function block. |
| FB_BA_InRngAzm [▶ 262] | Verification of valid sun position and sun direction range (azimuth angle) |
| FB_BA_InRngElv [▶ 265] | Verification of valid sun position and sun elevation range (elevation angle) |
| FB_BA_RdFcdElemLst [▶ 267] | Shading correction: input of facade elements via data list (csv). |
| FB_BA_RdShdObjLst [▶ 271] | Shading correction: input of shading objects via data list (csv). |
| FB_BA_RolBldActr [▶ 276] | Roller shutter actuator |
| FB_BA_ShdCorr [▶ 278] | Shading correction function block |
| FB_BA_ShdObjEntry [▶ 281] | Shading correction: input of shading objects per function block. |
| FB_BA_SunBldActr [▶ 284] | Blind actuator |
| FB_BA_SunBldPosDly [▶ 289] | Start-up delay for blinds/groups of blinds |
| FB_BA_SunBldEvt [▶ 291] | Output of a specified blind position and angle in percent |
| FB_BA_SunBldPrioSwi4 [▶ 291] | Priority control, 4 inputs |
| FB_BA_SunBldPrioSwi8 [▶ 293] | Priority control, 8 inputs |
| FB_BA_SunBldScn [▶ 294] | Manual operation with scene selection and programming |
| FB_BA_SunBldSwi [▶ 297] | Manual operation |
| FB_BA_SunBldTwiLgtAuto [▶ 299] | Twilight automatic |
| FB_BA_SunBldWthrPrtc [▶ 300] | Weather protection function |
| FB_BA_SunPrtc [▶ 302] | Sun protection function, see Overview of automatic sun protection (shading correction) |

**Room functions; lighting**

| Name | Description |
|---|---|
| FB_BA_LgtSwi [▶ 306] | Switching and dimming function block - function for two light switches. |

| Name | Description |
|---|---|
| FB_BA_CnstLgtCtrl [▶ 309] | Constant light control for a room. |
| FB_BA_AnlgLgtActr [▶ 311] | Function block for controlling an analog light actuator, e.g. via a KL2751. |
| FB_BA_DALILgtActr [▶ 313] | Function block for controlling a DALI light actuator. |

### Special

| Name | Description |
|---|---|
| FB_BA_Blink [▶ 317] | Simple oscillator function block |

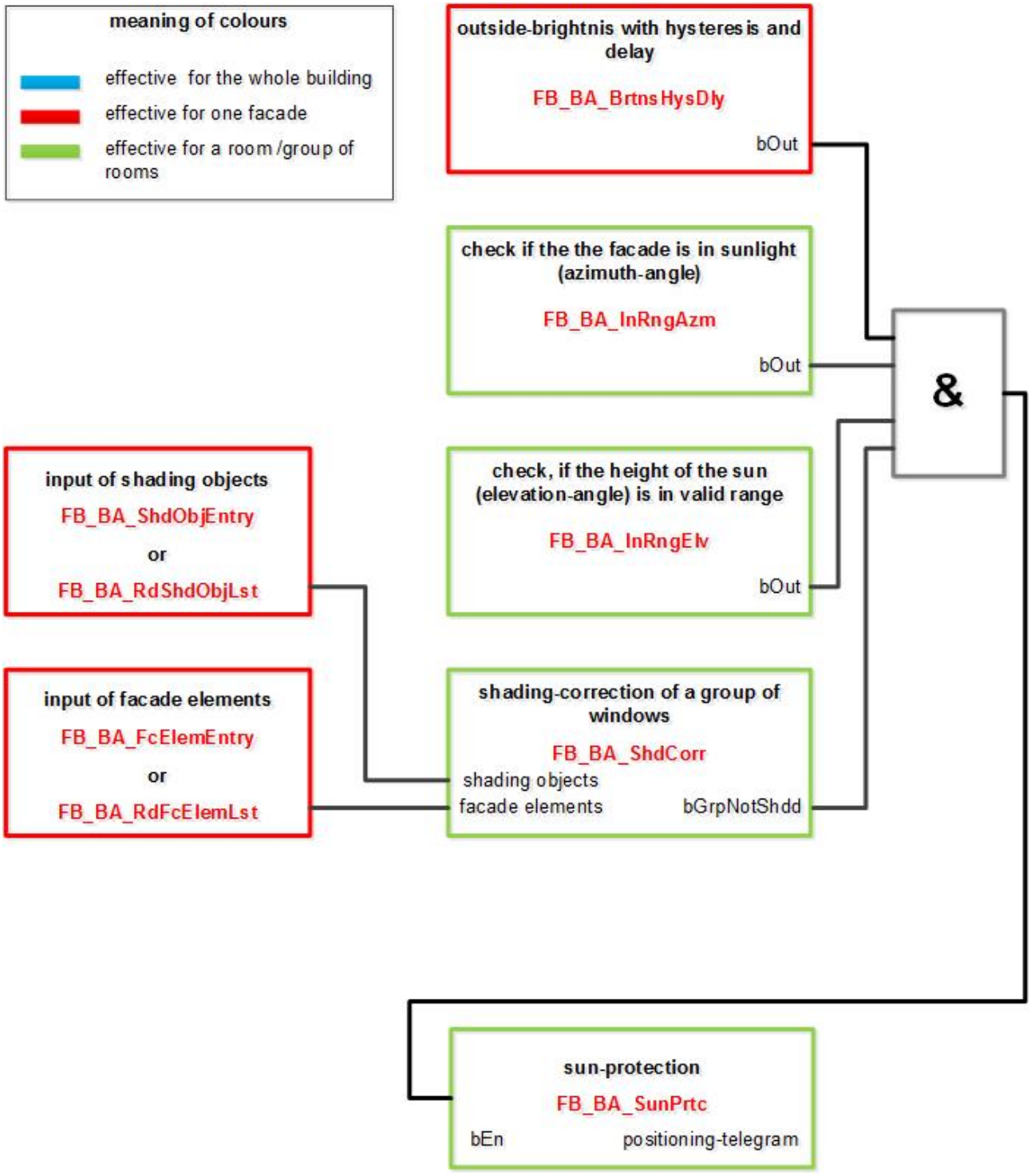
### System

| Name | Description |
|---|---|
| FB_BA_CnvtTiSt [▶ 318] | Conversion year, month, day, hour, minute and second in time structure |
| FB_BA_ExtTiSt [▶ 319] | Conversion time structure in year, month, day, hour, minute and second |
| FB_BA_GetTime [▶ 320] | Internal clock with time information - can be synchronized with system time |
| FB_BA_SetTime [▶ 322] | Setting the system time |

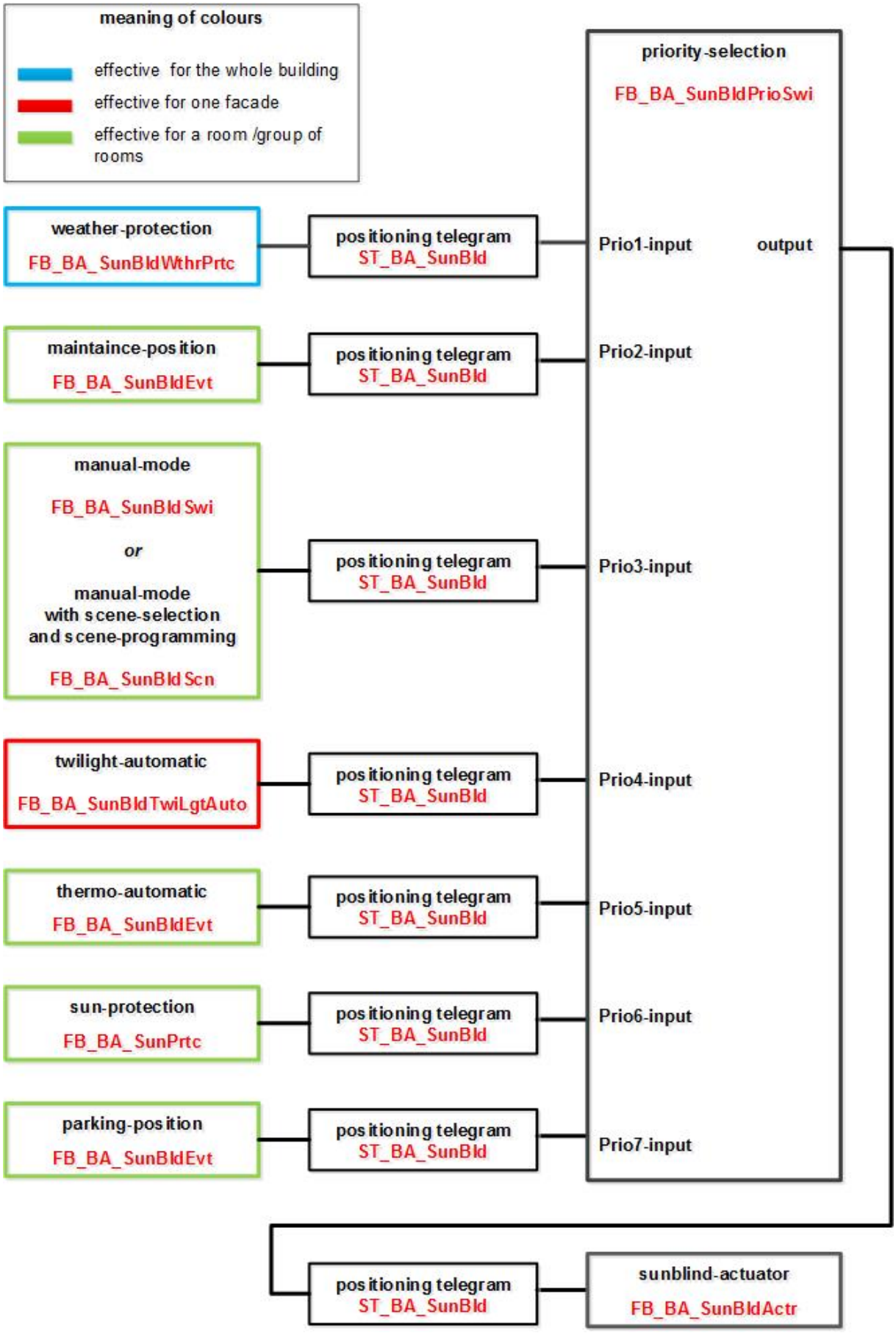
### Overview of enumerations and structures

| Name | Description |
|---|---|
| E_BA_AlmSta [▶ 324] | Enumerator status of the alarm messages |
| E_BA_PosMod [▶ 324] | Enumerator for the definition of the positioning mode |
| E_BA_ShdObjType [▶ 325] | Enumerator for selecting the shading object type |
| ST_BA_BldPosTab [▶ 325] | Structure of the interpolation point entries for the height adjustment of the blind |
| ST_BA_Cnr [▶ 325] | Information about window corners |
| ST_BA_ComnMsg [▶ 326] | Structure of the communication telegram for the FB_BA_ComMsg [▶ 197] function blocks |
| ST_BA_ComnMsgTermt [▶ 326] | Structure of the communication telegram to the function blocks FB_BA_ComMsgTermt [▶ 200] |
| ST_BA_FcdElem [▶ 327] | List entry for a facade element (window). |
| ST_BA_SeqLink [▶ 327] | Structure of the data and command exchange between the control function block FB_BA_SeqLink [▶ 168] and the sequence controllers FB_BA_SeqCtrl [▶ 165]. |
| ST_BA_ShdObj [▶ 329] | List entry of a shading object |
| ST_BA_SpRmT [▶ 330] | Room temperature setpoints |
| ST_BA_SunBld [▶ 331] | Structure of the blind positioning telegram |
| ST_BA_SunBldScn [▶ 331] | Table entry for a blind scene |

## 8.2.1 Shading correction



## 8.2.2 Automatic sun protection



### 8.2.3 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 [▶ 329] within the program.

The shading correction FB_BA_ShdCorr [▶ 278] only reads the information from this list, while the management function block FB_BA_ShdObjEntry [▶ 281] has read and write access 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
```

### 8.2.4 List of facade elements

The data of all windows (facade elements) per facade are saved within the program in a field of structural elements of the type ST_BA_FcdElem [▶ 327].

The management function block FB_BA_FcdElemEntry [▶ 258] and the shading correction FB_BA_ShdCorr [▶ 278] 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:

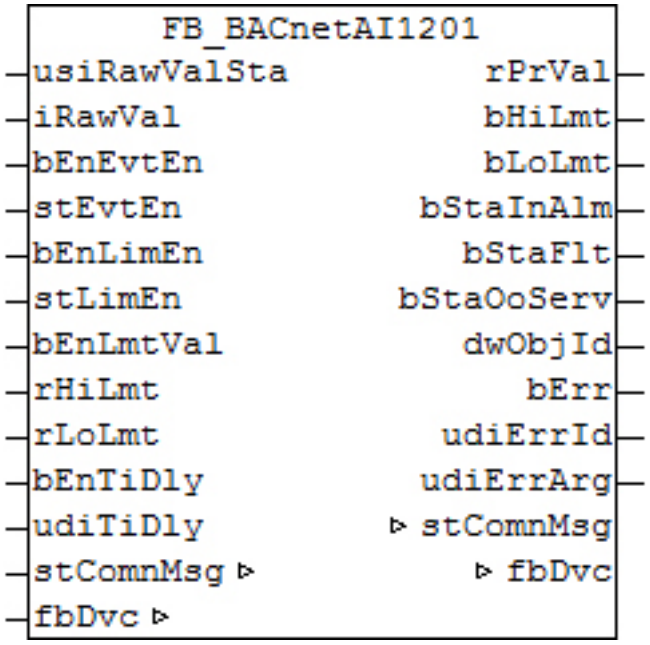
```
VAR_GLOBAL
     arrFcdElem : ARRAY[1..gBA_cMaxColumnFcd, 1..gBA_cMaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

The variables *gBA_cMaxColumnFcd* and *gBA_cMaxRowFcd* define the upper limit of the available elements and are declared as global constants within the program library:

```
VAR_GLOBAL CONSTANT
     gBA_cMaxRowFcd    : INT :=10;
     gBA_cMaxColumnFcd : INT :=20;
END_VAR
```

### 8.2.5 FB_BACnetAI1201

BACnet analog input




### Functional description

This function block generates a BACnet analog input object and provides write and read variables for the object within the PLC.
This function block is the "small" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAI1203 [▶ 49]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnLimEn | Enable for writing the bit pattern of the limit value enables | |
| stLimEn | Property value bit pattern limit value enables | Limit-Enable (52) |
| bEnLmtVal | Enable for writing the limit values | |
| rHiLmt | Property value HighLimit | HighLimit (45) |
| rLoLmt | Property value LowLimit | LowLimit (59) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |

### Inputs/outputs

#### VAR_INPUT

```
usiRawValSta : USINT;
iRawVal      : INT;
bEnEvtEn     : BOOL;
stEvtEn      : ST_BACnet_EventTransitionBits;
bEnLimEn     : BOOL;
stLimEn      : ST_BACnet_LimitEnable;
bEnLmtVal    : BOOL;
rHiLmt       : REAL;
```

```
rLoLmt       : REAL;
bEnTiDly     : BOOL;
udiTiDly     : UDINT;
```

**usiRawValSta:** input for linking with the status byte (State) of the terminal

**iRawVal:** input for linking with the data word (Data In) of the terminal

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnLimEn / stLimEn:** enable/property value bit pattern LimitEnable

**bEnLimVal:** enable writing High-Limit and Low-Limit

**rHiLmt:** property value High-Limit

**rLoLmt:** property value Low-Limit

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

### VAR_OUTPUT

```
rPrVal       : REAL;
bHiLmt       : BOOL;
bLoLmt       : BOOL;
bStaInAlm    : BOOL;
bStaFlt      : BOOL;
bStaOoServ   : BOOL;
dwObjId      : DWORD;
bErrs        : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**rPrVal:** current value of the analog input object - read directly from the BACnet

**bHiLim:** message upper limit reached

**bLoLim:** message lower limit reached

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog input object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog input object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog input object.

**dwObjId:** BACnet object ID of the analog input object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

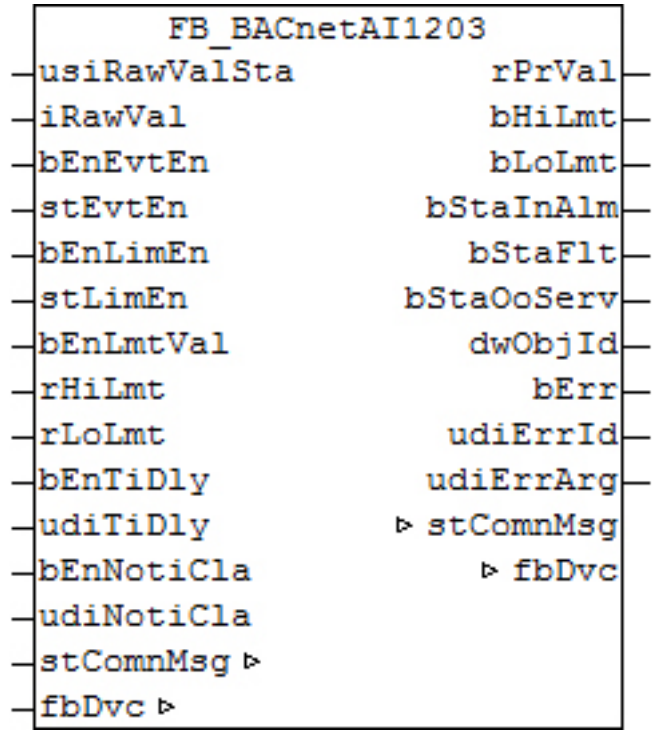
**fbDvc:** reference to the function block of the BACnet device object.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.6 FB_BACnetAI1203

BACnet analog input




### Functional description

This function block generates a BACnet analog input object and provides write and read variables for the object within the PLC.
This function block is the "large" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAI1201 [▶ 46]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnLimEn | Enable for writing the bit pattern of the limit value enables | |
| stLimEn | Property value bit pattern limit value enables | Limit-Enable (52) |
| bEnLmtVal | Enable for writing the limit values | |
| rHiLmt | Property value HighLimit | HighLimit (45) |
| rLoLmt | Property value LowLimit | LowLimit (59) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

#### VAR_INPUT

```
usiRawValSta : USINT;
iRawVal      : INT;
bEnEvtEn     : BOOL;
stEvtEn      : ST_BACnet_EventTransitionBits;
bEnLimEn     : BOOL;
stLimEn      : ST_BACnet_LimitEnable;
bEnLmtVal    : BOOL;
rHiLmt       : REAL;
rLoLmt       : REAL;
bEnTiDly     : BOOL;
udiTiDly     : UDINT;
bEnNotiCla   : BOOL;
udiNotiCla   : UDINT;
```

**usiRawValSta:** input for linking with the status byte (State) of the terminal

**iRawVal:** input for linking with the data word (Data In) of the terminal

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnLimEn / stLimEn:** enable/property value bit pattern LimitEnable

**bEnLimVal:** enable writing High-Limit and Low-Limit

**rHiLmt:** property value High-Limit

**rLoLmt:** property value Low-Limit

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

#### VAR_OUTPUT

```
rPrVal      : REAL;
bHiLmt      : BOOL;
bLoLmt      : BOOL;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOoServ  : BOOL;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**rPrVal:** current value of the analog input object - read directly from the BACnet

**bHiLim:** message upper limit reached

**bLoLim:** message lower limit reached

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog input object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog input object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog input object.

**dwObjId:** BACnet object ID of the analog input object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

#### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

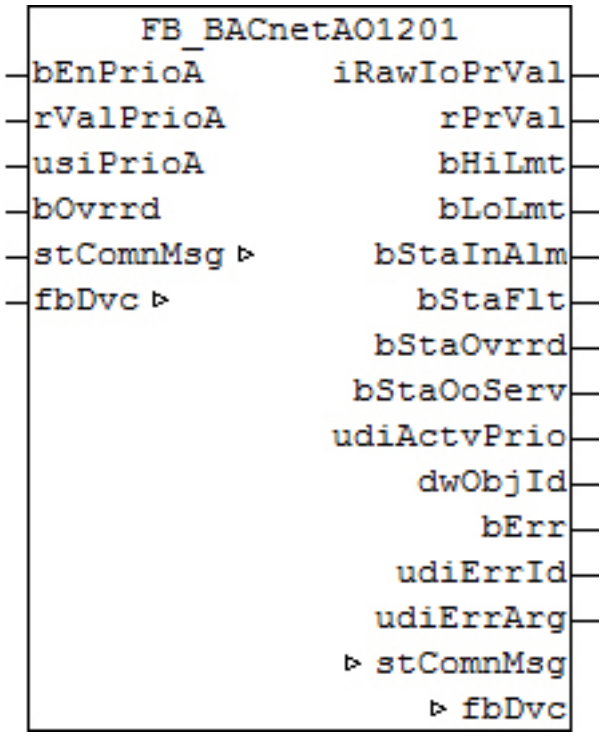
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.7 FB_BACnetAO1201

BACnet analog output




### Functional description

This function block generates a BACnet analog output object and provides write and read variables for the object within the PLC.
This function block is the "small" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAO1203 [▶ 53]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| rValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |

### Inputs/outputs

#### VAR_INPUT

```
bEnPrioA    : BOOL;
rValPrioA   : REAL;
usiPrioA    : USINT;
bOvrrd      : BOOL;
```

**bEnPrioA:** enable for writing

**rValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bOvrrd:** at this input the input signal of a local priority operation, e.g. of an KM4602, can be created. If this input is then "TRUE", this is indicated in the BACnet.

#### VAR_OUTPUT

```
iRawIoPrVal : INT;
rPrVal      : REAL;
bHiLmt      : BOOL;
bLoLmt      : BOOL;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOvrrd   : BOOL;
bStaOoServ  : BOOL;
udiActvPrio : UDINT;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**iRawIoPrVal:** output for linking with the data word (Data Out) of the analog output terminal

**rPrVal:** current value of the analog output object - read directly from the BACnet

**bHiLim:** message upper limit reached

**bLoLim:** message lower limit reached

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the analog output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog output object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the analog output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

#### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197]

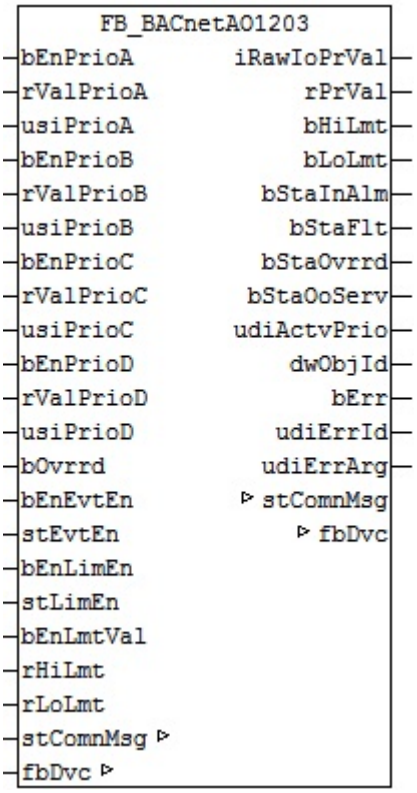
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.8 FB_BACnetAO1203

BACnet analog output




### Functional description

This function block generates a BACnet analog output object and provides write and read variables for the object within the PLC.
This function block is the "large" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAO1201 [▶ 51]
- FB_BACnetAO1205 [▶ 56]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| rValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| rValPrioB | Value to be written | Priority-Array (87) |
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| rValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| rValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnLimEn | Enable for writing the bit pattern of the limit value enables | |
| stLimEn | Property value bit pattern limit value enables | Limit-Enable (52) |
| bEnLmtVal | Enable for writing the limit values | |
| rHiLmt | Property value HighLimit | HighLimit (45) |
| rLoLmt | Property value LowLimit | LowLimit (59) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
rValPrioA   : REAL;
usiPrioA    : USINT;
bEnPrioB    : BOOL;
rValPrioB   : REAL;
usiPrioB    : USINT;
bEnPrioC    : BOOL;
rValPrioC   : REAL;
usiPrioC    : USINT;
bEnPrioD    : BOOL;
rValPrioD   : REAL;
usiPrioD    : USINT;
bOvrrd      : BOOL;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnLimEn    : BOOL;
stLimEn     : ST_BACnet_LimitEnable;
bEnLmtVal   : BOOL;
rHiLmt      : REAL;
rLoLmt      : REAL;
```

**bEnPrioA:** enable for writing

**rValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**rValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**rValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**rValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bOvrrd:** at this input the input signal of a local priority operation, e.g. of an KM4602, can be created. If this input is then "TRUE", this is indicated in the BACnet.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnLimEn / stLimEn:** enable/property value bit pattern LimitEnable

**bEnLimVal:** enable writing High-Limit and Low-Limit

**rHiLmt:** property value High-Limit

**rLoLmt:** property value Low-Limit

### VAR_OUTPUT

```
iRawIoPrVal : INT;
rPrVal      : REAL;
bHiLmt      : BOOL;
bLoLmt      : BOOL;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOvrrd   : BOOL;
bStaOoServ  : BOOL;
udiActvPrio : UDINT;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**iRawIoPrVal:** output for linking with the data word (Data Out) of the analog output terminal.

**rPrVal:** current value of the analog output object - read directly from the BACnet

**bHiLim:** message upper limit reached

**bLoLim:** message lower limit reached

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the analog output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog output object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the analog output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

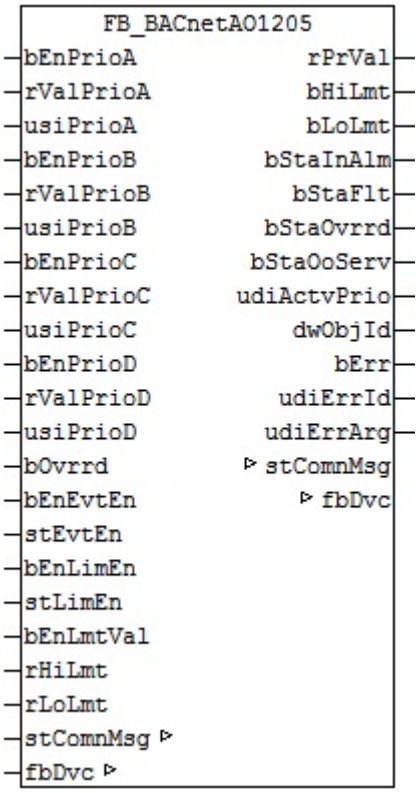
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.9 FB_BACnetAO1205

BACnet analog output




### Functional description

This function block generates a BACnet analog output object and provides write and read variables for the object within the PLC.
This function block is the "large" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAO1201 [▶ 51]
- FB_BACnetAO1203 [▶ 53]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| rValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| rValPrioB | Value to be written | Priority-Array (87) |
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| rValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| rValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnLimEn | Enable for writing the bit pattern of the limit value enables | |
| stLimEn | Property value bit pattern limit value enables | Limit-Enable (52) |
| bEnLmtVal | Enable for writing the limit values | |
| rHiLmt | Property value HighLimit | HighLimit (45) |
| rLoLmt | Property value LowLimit | LowLimit (59) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
rValPrioA   : REAL;
usiPrioA    : USINT;
bEnPrioB    : BOOL;
rValPrioB   : REAL;
usiPrioB    : USINT;
bEnPrioC    : BOOL;
rValPrioC   : REAL;
usiPrioC    : USINT;
bEnPrioD    : BOOL;
rValPrioD   : REAL;
usiPrioD    : USINT;
bOvrrd      : BOOL;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnLimEn    : BOOL;
stLimEn     : ST_BACnet_LimitEnable;
bEnLmtVal   : BOOL;
rHiLmt      : REAL;
rLoLmt      : REAL;
```

**bEnPrioA:** enable for writing

**rValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**rValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**rValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**rValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bOvrrd:** at this input the input signal of a local priority operation, e.g. of an KM4602, can be created. If this input is then "TRUE", this is indicated in the BACnet.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnLimEn / stLimEn:** enable/property value bit pattern LimitEnable

**bEnLimVal:** enable writing High-Limit and Low-Limit

**rHiLmt:** property value High-Limit

**rLoLmt:** property value Low-Limit

### VAR_OUTPUT

```
rPrVal      : REAL;
bHiLmt      : BOOL;
bLoLmt      : BOOL;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOvrrd   : BOOL;
bStaOoServ  : BOOL;
udiActvPrio : UDINT;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**rPrVal:** current value of the analog output object - read directly from the BACnet

**bHiLim:** message upper limit reached

**bLoLim:** message lower limit reached

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the analog output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog output object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the analog output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

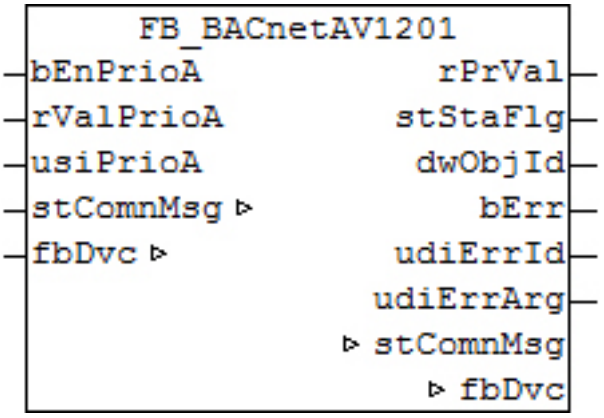
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.10 FB_BACnetAV1201

BACnet analog value




### Functional description

This function block generates a BACnet analog value object and provides write and read variables for the object within the PLC.
This function block is the "smallest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAV1202 [▶ 60]
- FB_BACnetAV1203 [▶ 63]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| rValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |

### Inputs/outputs

#### VAR_INPUT

```
bEnPrioA  : BOOL;
rValPrioA : REAL;
usiPrioA  : USINT;
```

**bEnPrioA:** enable for writing

**rValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

### VAR_OUTPUT

```
rPrVal    : REAL;
stStaFlg  : ST_BACnet_StatusFlags;
dwObjId   : DWORD;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**rPrVal:** current value of the analog value object - read directly from the BACnet

**stStaFlg:** output structure of the BACnet status

**dwObjId:** BACnet object ID of the analog value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

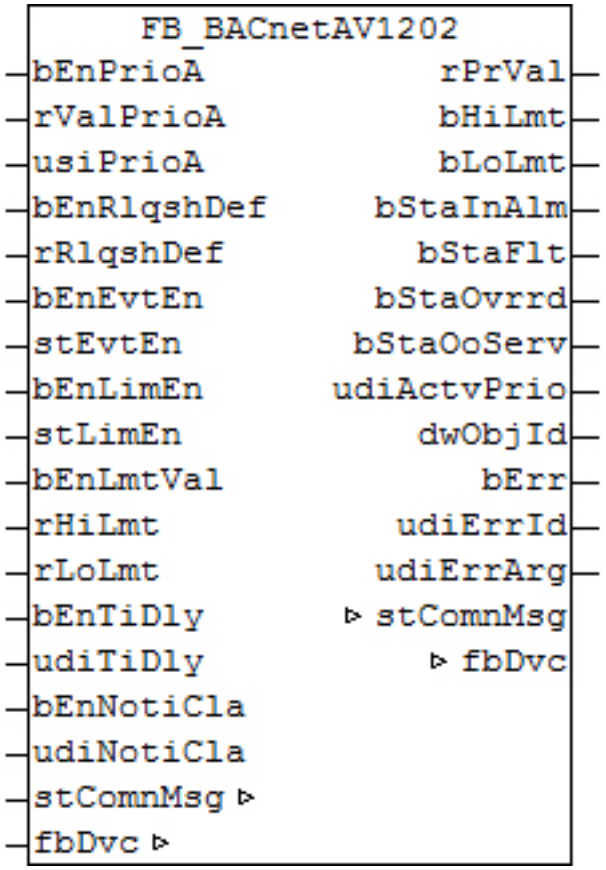
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.11 FB_BACnetAV1202

BACnet analog value




### Functional description

This function block generates a BACnet analog value object and provides write and read variables for the object within the PLC.
This function block is the "medium" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAV1201 [▶ 59]
- FB_BACnetAV1203 [▶ 63]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| rValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnRlqshDef | Enable for writing the relinquish default | |
| rRlqshDef | Value to be written to the property relinquish default | Relinquish-Default (104) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnLimEn | Enable for writing the bit pattern of the limit value enables | |
| stLimEn | Property value bit pattern limit value enables | Limit-Enable (52) |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnLmtVal | Enable for writing the limit values | |
| rHiLmt | Property value HighLimit | HighLimit (45) |
| rLoLmt | Property value LowLimit | LowLimit (59) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
rValPrioA   : REAL;
usiPrioA    : USINT;
bEnRlqshDef : BOOL;
rRlqshDef   : REAL;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnLimEn    : BOOL;
stLimEn     : ST_BACnet_LimitEnable;
bEnLmtVal   : BOOL;
rHiLmt      : REAL;
rLoLmt      : REAL;
bEnTiDly    : BOOL;
udiTiDly    : UDINT;
bEnNotiCla  : BOOL;
udiNotiCla  : UDINT;
```

**bEnPrioA:** enable for writing

**rValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnRlqshDef / rRlqshDef:** enable/property value relinquish default

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnLimEn / stLimEn:** enable/property value bit pattern LimitEnable

**bEnLimVal:** enable writing High-Limit and Low-Limit

**rHiLmt:** property value High-Limit

**rLoLmt:** property value Low-Limit

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

### VAR_OUTPUT

```
rPrVal      : REAL;
bHiLmt      : BOOL;
bLoLmt      : BOOL;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOvrrd   : BOOL;
bStaOoServ  : BOOL;
udiActvPrio : UDINT;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**rPrVal:** current value of the analog value object - read directly from the BACnet

**bHiLim:** message upper limit reached

**bLoLim:** message lower limit reached

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog value object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the analog value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog value object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the analog value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

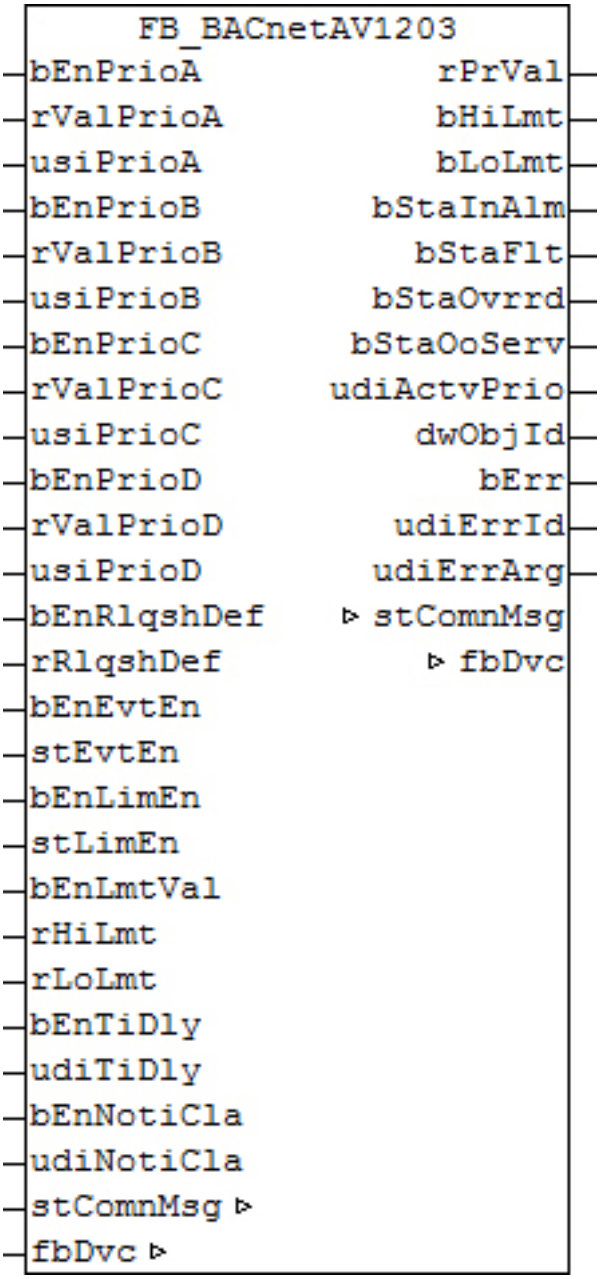
**fbDvc:** reference to the function block of the BACnet device object.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.12 FB_BACnetAV1203

BACnet analog value




**Functional description**

This function block generates a BACnet analog value object and provides write and read variables for the object within the PLC.
This function block is the "largest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAV1201 [▶ 59]
- FB_BACnetAV1202 [▶ 60]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| rValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| rValPrioB | Value to be written | Priority-Array (87) |
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| rValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| rValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnRlqshDef | Enable for writing the relinquish default | |
| rRlqshDef | Value to be written to the property relinquish default | Relinquish-Default (104) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnLimEn | Enable for writing the bit pattern of the limit value enables | |
| stLimEn | Property value bit pattern limit value enables | Limit-Enable (52) |
| bEnLmtVal | Enable for writing the limit values | |
| rHiLmt | Property value HighLimit | HighLimit (45) |
| rLoLmt | Property value LowLimit | LowLimit (59) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
rValPrioA   : REAL;
usiPrioA    : USINT;
bEnPrioB    : BOOL;
rValPrioB   : REAL;
usiPrioB    : USINT;
bEnPrioC    : BOOL;
rValPrioC   : REAL;
usiPrioC    : USINT;
bEnPrioD    : BOOL;
rValPrioD   : REAL;
usiPrioD    : USINT;
bEnRlqshDef : BOOL;
rRlqshDef   : REAL;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnLimEn    : BOOL;
stLimEn     : ST_BACnet_LimitEnable;
bEnLmtVal   : BOOL;
rHiLmt      : REAL;
```

```
rLoLmt      : REAL;
bEnTiDly    : BOOL;
udiTiDly    : UDINT;
bEnNotiCla  : BOOL;
udiNotiCla  : UDINT;
```

**bEnPrioA:** enable for writing

**rValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**rValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**rValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**rValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bEnRlqshDef / rRlqshDef:** enable/property value relinquish default

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnLimEn / stLimEn:** enable/property value bit pattern LimitEnable

**bEnLimVal:** enable writing High-Limit and Low-Limit

**rHiLmt:** property value High-Limit

**rLoLmt:** property value Low-Limit

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

### VAR_OUTPUT

```
rPrVal      : REAL;
bHiLmt      : BOOL;
bLoLmt      : BOOL;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOvrrd   : BOOL;
bStaOoServ  : BOOL;
udiActvPrio : UDINT;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**rPrVal:** current value of the analog value object - read directly from the BACnet

**bHiLim:** message upper limit reached

**bLoLim:** message lower limit reached

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog value object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the analog value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog value object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the analog value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

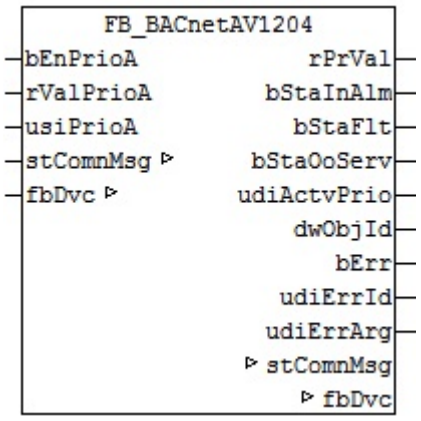
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.13 FB_BACnetAV1204

BACnet analog value




### Functional description

This function block generates a BACnet analog value object and provides write and read variables for the object within the PLC.
This function block is the "smallest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetAV1201 [▶ 59]
- FB_BACnetAV1202 [▶ 60]
- FB_BACnetAV1203 [▶ 63]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| rValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA  : BOOL;
rValPrioA : REAL;
usiPrioA  : USINT;
```

**bEnPrioA:** enable for writing

**rValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

### VAR_OUTPUT

```
rPrVal      : REAL;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOoServ  : BOOL;
udiActvPrio : UDINT;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**rPrVal:** current value of the analog value object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the analog value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the analog value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the analog value object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the analog value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

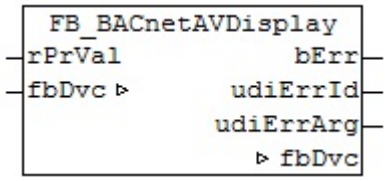
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.14 FB_BACnetAVDisplay

BACnet analog value object that can be used to display a value from the PLC in the BACnet.




### Functional description

This function block generates a BACnet analog value object and provides write and read variables for the object within the PLC.
Alternatively, the following versions are available:

- FB_BACnetAV1201 [▶ 59]
- FB_BACnetAV1202 [▶ 60]
- FB_BACnetAV1203 [▶ 63]
- FB_BACnetAV1204 [▶ 67]
- FB_BACnetAVSetpoint [▶ 69]

### Inputs/outputs

#### VAR_INPUT

```
rPrVal    : REAL;
```

**rPrVal:** Value from the PLC that is written to the BACnet property Present Value in the event of a change of value.

#### VAR_OUTPUT

```
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**bErr:** Indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterisation.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

#### VAR_IN_OUT

```
fbDvc     : FB_BACnet_Device;
```

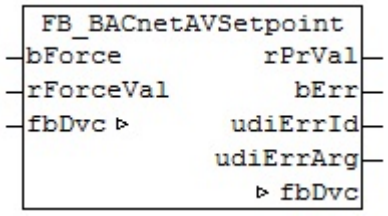
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.15 FB_BACnetAVSetpoint

BACnet analog value object that maps the BACnet property Present Value in the PLC.




### Functional description

This function block generates a BACnet analog value object and provides write and read variables for the object within the PLC.
Alternatively, the following versions are available:

- FB_BACnetAV1201 [▶ 59]
- FB_BACnetAV1202 [▶ 60]
- FB_BACnetAV1203 [▶ 63]
- FB_BACnetAV1204 [▶ 67]
- FB_BACnetAVDisplay [▶ 69]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface.
The value of rForceVal is written once to the Present Value of the AV object on each rising edge at bForce.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bForce | The value of rForceVal is written once to the Present Value of the AV object on each rising edge at bForce. | |
| rForceVal | Value to be written to the BACnet property Present Value | Present Value (85) |

### Inputs/outputs

#### VAR_INPUT

```
bForce    : BOOL;
rForceVal : REAL;
```

**bForce:** The value of rForceVal is written once to the Present Value of the AV object on each rising edge at bForce.

**rForceVal:** Value that is written to the BACnet property Present Value.

#### VAR_OUTPUT

```
rPrVal      : REAL;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**rPrVal:** current value of the BACnet property Present Value – read directly from the BACnet

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

#### VAR_IN_OUT

```
fbDvc     : FB_BACnet_Device;
```

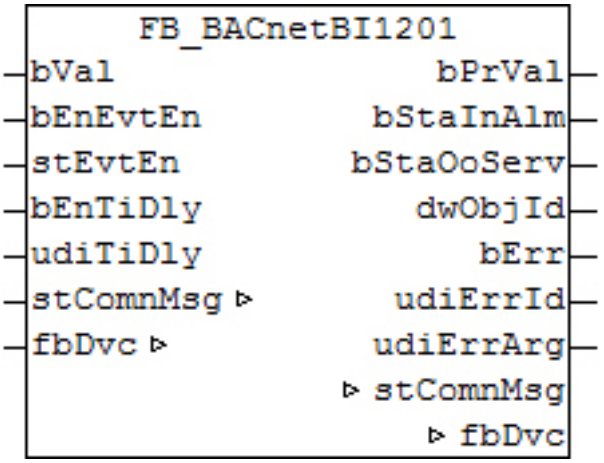
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.16 FB_BACnetBI1201

BACnet binary input




### Functional description

This function block generates a BACnet binary input object and provides write and read variables for the object within the PLC.
This function block is the "large" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBI1203 [▶ 72]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |

### Inputs/outputs

#### VAR_INPUT

```
bVal          : BOOL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
```

**bVal:** input for linking with the input bit (Input) of the terminal

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

### VAR_OUTPUT

```
bPrVal       : BOOL;
bStaInAlm    : BOOL;
bStaOoServ   : BOOL;
dwObjId      : DWORD;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**bPrVal:** current value of the binary input object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary input object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary input object.

**dwObjId:** BACnet object ID of the binary input object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

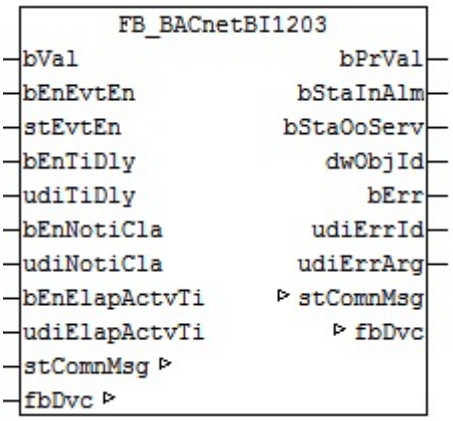
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.17 FB_BACnetBI1203

BACnet binary input



### Functional description

This function block generates a BACnet binary input object and provides write and read variables for the object within the PLC.
This function block is the "large" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBI1201 [▶ 71]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |
| bEnElapActvTi | Enable for writing the elapsed active time | |
| udiElapActvTi | Property value elapsed active time | ElapsedActiveTime (33) |

### Inputs/outputs

#### VAR_INPUT

```
bVal          : BOOL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
bEnElapActvTi : BOOL;
udiElapActvTi : UDINT;
```

**bVal:** input for linking with the input bit (Input) of the terminal

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

**bEnElapActvTi / udiElapActvTi:** enable/property value elapsed active time

#### VAR_OUTPUT

```
bPrVal       : BOOL;
bStaInAlm    : BOOL;
bStaOoServ   : BOOL;
dwObjId      : DWORD;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**bPrVal:** current value of the binary input object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary input object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary input object.

**dwObjId:** BACnet object ID of the binary input object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

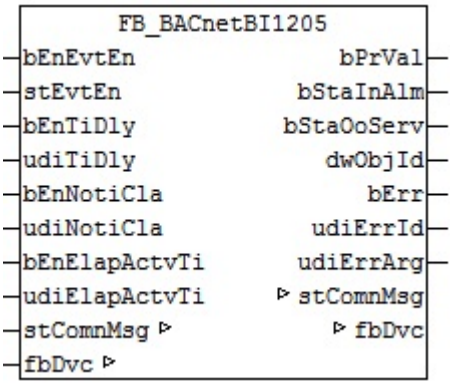
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.18 FB_BACnetBI1205

BACnet binary input




### Functional description

This function block generates a BACnet binary input object and provides write and read variables for the object within the PLC.
This function block is the "large" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBI1201 [▶ 71]
- FB_BACnetBI1203 [▶ 72]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |
| bEnElapActvTi | Enable for writing the elapsed active time | |
| udiElapActvTi | Property value elapsed active time | ElapsedActiveTime (33) |

### Inputs/outputs

### VAR_INPUT

```
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
bEnElapActvTi : BOOL;
udiElapActvTi : UDINT;
```

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

**bEnElapActvTi / udiElapActvTi:** enable/property value elapsed active time

### VAR_OUTPUT

```
bPrVal       : BOOL;
bStaInAlm    : BOOL;
bStaOoServ   : BOOL;
dwObjId      : DWORD;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**bPrVal:** current value of the binary input object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary input object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary input object.

**dwObjId:** BACnet object ID of the binary input object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.19 FB_BACnetBO1201

BACnet binary output


### Functional description

This function block generates a BACnet binary output object and provides write and read variables for the object within the PLC.
This function block is the "smallest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBO1202 [▶ 78]
- FB_BACnetBO1203 [▶ 81]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |

### Inputs/outputs

#### VAR_INPUT

```
bEnPrioA      : BOOL;
bValPrioA     : BOOL;
usiPrioA      : USINT;
```

```
bOvrrd        : BOOL;
bEnFdbVal     : BOOL;
bFdbVal       : BOOL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
bEnElapActvTi : BOOL;
udiElapActvTi : UDINT;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bOvrrd:** at this input the input signal of a local priority operation, e.g. of an KM2652, can be created. If this input is then "TRUE", this is indicated in the BACnet.

**bEnFdbVal / bFdbVal:** enable/property value feedback value. This value is written from the PLC to the Property FeedbackValue (Property Id 40), however **not** via ADS, but via cyclic interface.
It is used for feedback monitoring: if *bEnFdbVal* is set to TRUE, *bFdbVal* must have the same value as the output Present Value within the time specified in the BACnet under TIME Delay (Property Id 113). Otherwise a feedback discrepancy is displayed in the EventState (property ID 36) via the entry "offnormal". If *bEnFdbVal* is set to FALSE, internally *bFdbVal* is set to same value as the Present Value, so that feedback discrepancies are avoided.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

**bEnElapActvTi / udiElapActvTi:** enable/property value elapsed active time

### VAR_OUTPUT

```
bRawIoPrVal      : BOOL;
bPrVal           : BOOL;
bStaInAlm        : BOOL;
bStaFlt          : BOOL;
bStaOvrrd        : BOOL;
bStaOoServ       : BOOL;
udiStaElapActvTi : UDINT;
udiActvPrio      : UDINT;
dwObjId          : DWORD;
bErr             : BOOL;
udiErrId         : UDINT;
udiErrArg        : UDINT;
```

**bRawIoPrVal:** output value to the digital output terminal

**bPrVal:** current value of the binary output object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary output object.

**udiStaElapActvTi:** indicates the elapsed active time of the binary output object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the binary output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

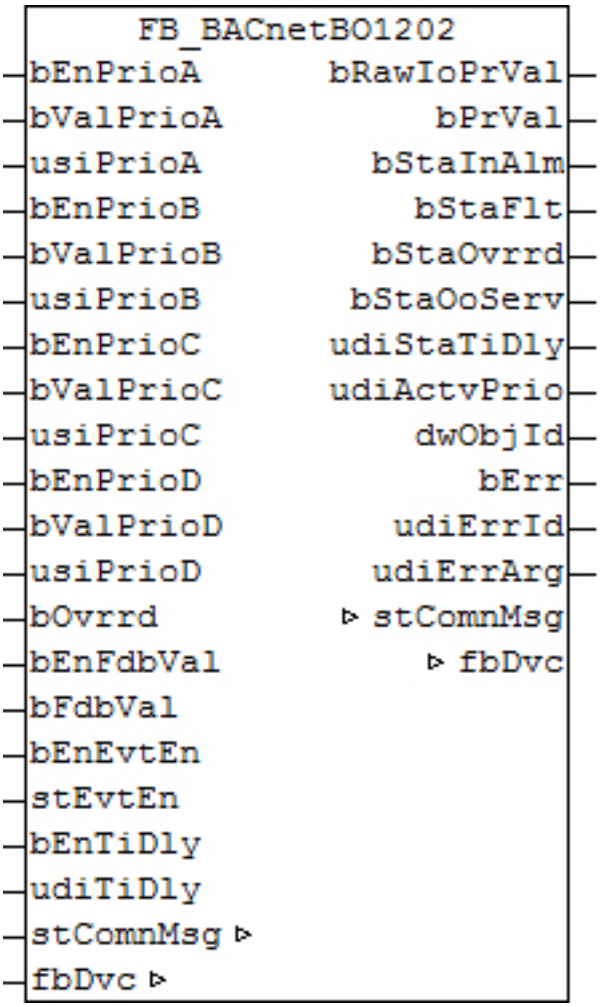
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.20 FB_BACnetBO1202

BACnet binary output



### Functional description

This function block generates a BACnet binary output object and provides write and read variables for the object within the PLC.
This function block is the "medium" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBO1201 [▶ 76]
- FB_BACnetBO1203 [▶ 81]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| bValPrioB | Value to be written | Priority-Array (87) |
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| bValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| bValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |

### Inputs/outputs

#### VAR_INPUT

```
bEnPrioA      : BOOL;
bValPrioA     : BOOL;
usiPrioA      : USINT;
bEnPrioB      : BOOL;
bValPrioB     : BOOL;
usiPrioB      : USINT;
bEnPrioC      : BOOL;
bValPrioC     : BOOL;
usiPrioC      : USINT;
bEnPrioD      : BOOL;
bValPrioD     : BOOL;
usiPrioD      : USINT;
bOvrrd        : BOOL;
bEnFdbVal     : BOOL;
bFdbVal       : BOOL;
```

```
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**bValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**bValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**bValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bOvrrd:** at this input the input signal of a local priority operation, e.g. of an KM2652, can be created. If this input is then "TRUE", this is indicated in the BACnet.

**bEnFdbVal / bFdbVal:** enable/property value feedback value. This value is written from the PLC to the Property FeedbackValue (Property Id 40), however **not** via ADS, but via cyclic interface.
It is used for feedback monitoring: if *bEnFdbVal* is set to TRUE, *bFdbVal* must have the same value as the output Present Value within the time specified in the BACnet under TIME Delay (Property Id 113). Otherwise a feedback discrepancy is displayed in the EventState (property ID 36) via the entry "offnormal". If *bEnFdbVal* is set to FALSE, internally *bFdbVal* is set to same value as the Present Value, so that feedback discrepancies are avoided.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

### VAR_OUTPUT

```
bRawIoPrVal   : BOOL;
bPrVal        : BOOL;
bStaInAlm     : BOOL;
bStaFlt       : BOOL;
bStaOvrrd     : BOOL;
bStaOoServ    : BOOL;
udiStaTiDly   : UDINT;
udiActvPrio   : UDINT;
dwObjId       : DWORD;
bErr          : BOOL;
udiErrId      : UDINT;
udiErrArg     : UDINT;
```

**bRawIoPrVal:** output value to the digital output terminal

**bPrVal:** current value of the binary output object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary output object.

**udiStaTiDly:** indicates the currently parameterized TimeDelay (property ID 113).

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the binary output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

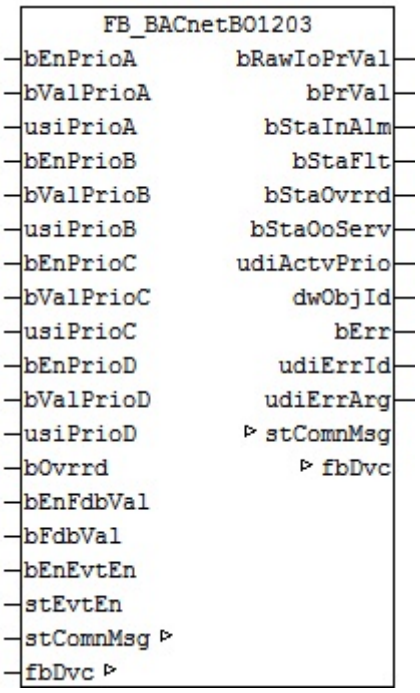
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.21 FB_BACnetBO1203

BACnet binary output




### Functional description

This function block generates a BACnet binary output object and provides write and read variables for the object within the PLC.
This function block is the "largest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBO1201 [▶ 76]

- FB_BACnetBO1202 [▶ 78]
- FB_BACnetBO1205 [▶ 84]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| bValPrioB | Value to be written | Priority-Array (87) |
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| bValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| bValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA      : BOOL;
bValPrioA     : BOOL;
usiPrioA      : USINT;
bEnPrioB      : BOOL;
bValPrioB     : BOOL;
usiPrioB      : USINT;
bEnPrioC      : BOOL;
bValPrioC     : BOOL;
usiPrioC      : USINT;
bEnPrioD      : BOOL;
bValPrioD     : BOOL;
usiPrioD      : USINT;
bOvrrd        : BOOL;
bEnFdbVal     : BOOL;
bFdbVal       : BOOL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**bValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**bValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**bValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bOvrrd:** at this input the input signal of a local priority operation, e.g. of an KM2652, can be created. If this input is then "TRUE", this is indicated in the BACnet.

**bEnFdbVal / bFdbVal:** enable/property value feedback value. This value is written from the PLC to the Property FeedbackValue (Property Id 40), however **not** via ADS, but via cyclic interface.
It is used for feedback monitoring: if *bEnFdbVal* is set to TRUE, *bFdbVal* must have the same value as the output Present Value within the time specified in the BACnet under TIME Delay (Property Id 113). Otherwise a feedback discrepancy is displayed in the EventState (property ID 36) via the entry "offnormal". If *bEnFdbVal* is set to FALSE, internally *bFdbVal* is set to same value as the Present Value, so that feedback discrepancies are avoided.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

### VAR_OUTPUT

```
bRawIoPrVal  : BOOL;
bPrVal       : BOOL;
bStaInAlm    : BOOL;
bStaFlt      : BOOL;
bStaOvrrd    : BOOL;
bStaOoServ   : BOOL;
udiActvPrio  : UDINT;
dwObjId      : DWORD;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**bRawIoPrVal:** output value to the digital output terminal

**bPrVal:** current value of the binary output object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary output object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the binary output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

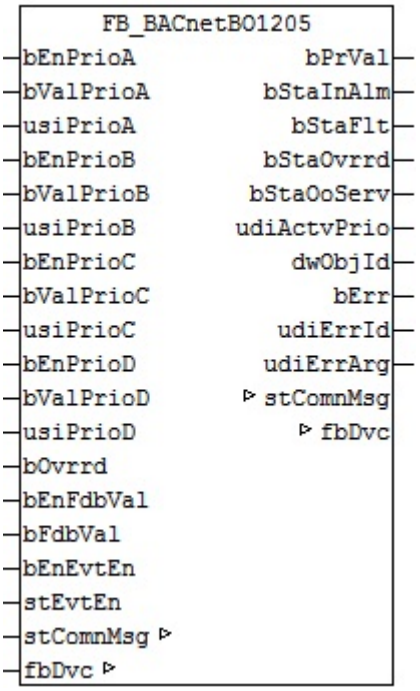
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.22 FB_BACnetBO1205

BACnet binary output




### Functional description

This function block generates a BACnet binary output object and provides write and read variables for the object within the PLC.
This function block is the "largest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBO1201 [▶ 76]
- FB_BACnetBO1202 [▶ 78]
- FB_BACnetBO1203 [▶ 81]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| bValPrioB | Value to be written | Priority-Array (87) |
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| bValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| bValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA      : BOOL;
bValPrioA     : BOOL;
usiPrioA      : USINT;
bEnPrioB      : BOOL;
bValPrioB     : BOOL;
usiPrioB      : USINT;
bEnPrioC      : BOOL;
bValPrioC     : BOOL;
usiPrioC      : USINT;
bEnPrioD      : BOOL;
bValPrioD     : BOOL;
usiPrioD      : USINT;
bOvrrd        : BOOL;
bEnFdbVal     : BOOL;
bFdbVal       : BOOL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**bValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**bValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**bValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bOvrrd:** at this input the input signal of a local priority operation, e.g. of an KM2652, can be created. If this input is then "TRUE", this is indicated in the BACnet.

**bEnFdbVal / bFdbVal:** enable/property value feedback value. This value is written from the PLC to the Property FeedbackValue (Property Id 40), however **not** via ADS, but via cyclic interface.
It is used for feedback monitoring: if *bEnFdbVal* is set to TRUE, *bFdbVal* must have the same value as the output Present Value within the time specified in the BACnet under TIME Delay (Property Id 113). Otherwise a feedback discrepancy is displayed in the EventState (property ID 36) via the entry "offnormal". If *bEnFdbVal* is set to FALSE, internally *bFdbVal* is set to same value as the Present Value, so that feedback discrepancies are avoided.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

### VAR_OUTPUT

```
bPrVal       : BOOL;
bStaInAlm    : BOOL;
bStaFlt      : BOOL;
bStaOvrrd    : BOOL;
bStaOoServ   : BOOL;
udiActvPrio  : UDINT;
dwObjId      : DWORD;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**bPrVal:** current value of the binary output object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary output object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the binary output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

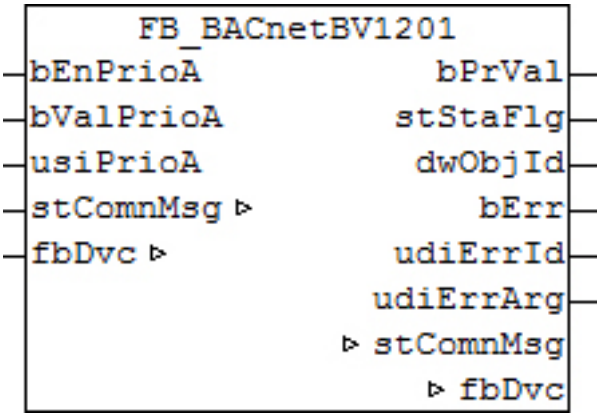
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.23 FB_BACnetBV1201

BACnet binary value




### Functional description

This function block generates a BACnet binary value object and provides write and read variables for the object within the PLC.
This function block is the "smallest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBV1202 [▶ 88]
- FB_BACnetBV1203 [▶ 90]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |

### Inputs/outputs

#### VAR_INPUT

```
bEnPrioA  : BOOL;
bValPrioA : BOOL;
usiPrioA  : USINT;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

#### VAR_OUTPUT

```
bPrVal    : BOOL;
stStaFlg  : ST_BACnet_StatusFlags;
dwObjId   : DWORD;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**bPrVal:** current value of the binary value object - read directly from the BACnet

**stStaFlg:** output structure of the BACnet status

**dwObjId:** BACnet object ID of the binary value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

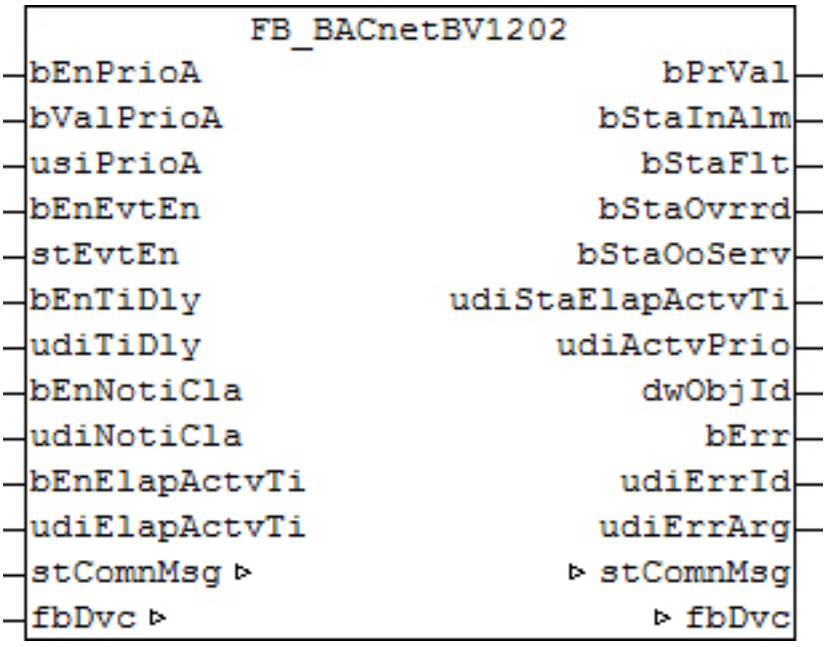
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.24 FB_BACnetBV1202

BACnet binary value




### Functional description

This function block generates a BACnet value output object and provides write and read variables for the object within the PLC.
This function block is the "medium" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBV1201 [▶ 86]
- FB_BACnetBV1203 [▶ 90]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |
| bEnElapActvTi | Enable for writing the elapsed active time | |
| udiElapActvTi | Property value elapsed active time | ElapsedActiveTime (33) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA      : BOOL;
bValPrioA     : BOOL;
usiPrioA      : USINT;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
bEnElapActvTi : BOOL;
udiElapActvTi : UDINT;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

**bEnElapActvTi / udiElapActvTi:** enable/property value elapsed active time

### VAR_OUTPUT

```
bPrVal           : BOOL;
bStaInAlm        : BOOL;
bStaFlt          : BOOL;
bStaOvrrd        : BOOL;
bStaOoServ       : BOOL;
udiStaElapActvTi : UDINT;
udiActvPrio      : UDINT;
dwObjId          : DWORD;
bErr             : BOOL;
udiErrId         : UDINT;
udiErrArg        : UDINT;
```

**bPrVal:** current value of the binary value object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary value object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary value object.

**udiStaElapActvTi:** indicates the elapsed active time of the binary value object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the binary value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

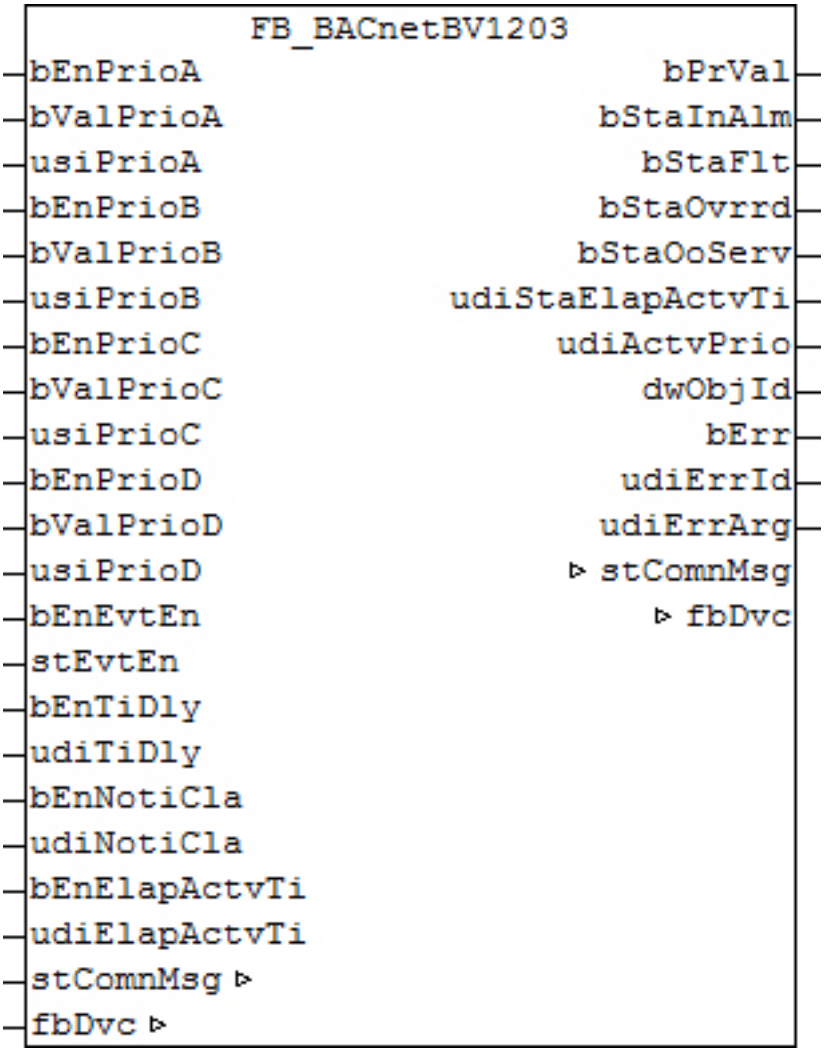
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.25 FB_BACnetBV1203

BACnet binary value




**Functional description**

This function block generates a BACnet binary value object and provides write and read variables for the object within the PLC.
This function block is the "largest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBV1201 [▶ 86]
- FB_BACnetBV1202 [▶ 88]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| bValPrioB | Value to be written | Priority-Array (87) |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| bValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| bValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |
| bEnElapActvTi | Enable for writing the elapsed active time | |
| udiElapActvTi | Property value elapsed active time | ElapsedActiveTime (33) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA      : BOOL;
bValPrioA     : BOOL;
usiPrioA      : USINT;
bEnPrioB      : BOOL;
bValPrioB     : BOOL;
usiPrioB      : USINT;
bEnPrioC      : BOOL;
bValPrioC     : BOOL;
usiPrioC      : USINT;
bEnPrioD      : BOOL;
bValPrioD     : BOOL;
usiPrioD      : USINT;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
bEnElapActvTi : BOOL;
udiElapActvTi : UDINT;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**bValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**bValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**bValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

**bEnElapActvTi / udiElapActvTi:** enable/property value elapsed active time

### VAR_OUTPUT

```
bPrVal           : BOOL;
bStaInAlm        : BOOL;
bStaFlt          : BOOL;
bStaOvrrd        : BOOL;
bStaOoServ       : BOOL;
udiStaElapActvTi : UDINT;
udiActvPrio      : UDINT;
dwObjId          : DWORD;
bErr             : BOOL;
udiErrId         : UDINT;
udiErrArg        : UDINT;
```

**bPrVal:** current value of the binary value object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary value object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary value object.

**udiStaElapActvTi:** indicates the elapsed active time of the binary value object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the binary value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

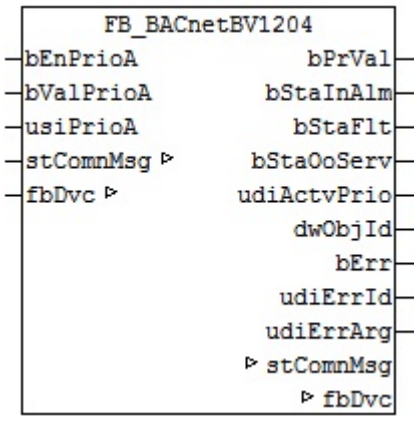
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.26 FB_BACnetBV1204

BACnet binary value




### Functional description

This function block generates a BACnet binary value object and provides write and read variables for the object within the PLC.
This function block is the "smallest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetBV1201 [▶ 86]
- FB_BACnetBV1202 [▶ 88]
- FB_BACnetBV1203 [▶ 90]
- FB_BACnetBVDisplay [▶ 95]
- FB_BACnetBVSetpoint [▶ 96]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| bValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA  : BOOL;
bValPrioA : BOOL;
usiPrioA  : USINT;
```

**bEnPrioA:** enable for writing

**bValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

### VAR_OUTPUT

```
bPrVal       : BOOL;
bStaInAlm    : BOOL;
bStaFlt      : BOOL;
bStaOoServ   : BOOL;
udiActvPrio  : UDINT;
dwObjId      : DWORD;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**bPrVal:** current value of the binary value object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the binary value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the binary value object.

**udiActvPrio:** indicates which priority is active.

**dwObjId:** BACnet object ID of the binary value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

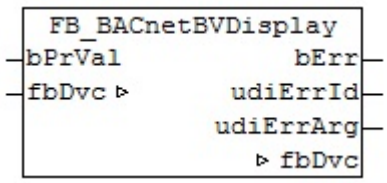
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.27 FB_BACnetBVDisplay

BACnet binary value object that can be used to display a value from the PLC in the BACnet.




### Functional description

This function block generates a BACnet binary value object and provides write and read variables for the object within the PLC.
Alternatively, the following versions are available:

- FB_BACnetBV1201 [▶ 86]
- FB_BACnetBV1202 [▶ 88]

- FB_BACnetBV1203 [▶ 90]
- FB_BACnetBV1204 [▶ 94]
- FB_BACnetSetpoint [▶ 96]

### Inputs/outputs

#### VAR_INPUT

```
bPrVal    : BOOL;
```

b**PrVal:** Boolean value from the PLC that is written to the BACnet property Present Value in the event of a change of value.

#### VAR_OUTPUT

```
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**bErr:** Indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterisation.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

#### VAR_IN_OUT

```
fbDvc     : FB_BACnet_Device;
```

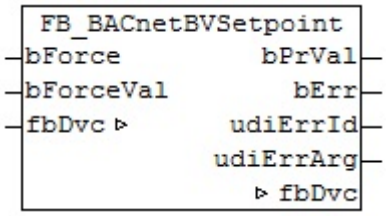
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.28 FB_BACnetBVSetpoint

BACnet binary value object that maps the BACnet property Present Value in the PLC.




### Functional description

This function block generates a BACnet binary value object and provides write and read variables for the object within the PLC.
Alternatively, the following versions are available:

- FB_BACnetBV1201 [▶ 86]
- FB_BACnetBV1202 [▶ 88]
- FB_BACnetBV1203 [▶ 90]
- FB_BACnetBV1204 [▶ 94]
- FB_BACnetBVDisplay [▶ 95]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred out of the PLC via ADS and into the PLC via cyclic interface.
With each rising edge at bForce the value of rForceVal is written once to the Present Value of the BV object.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bForce | The value of bForceVal is written once to the Present Value of the BV object on each rising edge at bForce. | |
| bForceVal | Value to be written to the BACnet property Present Value | Present Value (85) |

### Inputs/outputs

### VAR_INPUT

```
bForce    : BOOL;
bForceVal : BOOL;
```

**bForce:** the value of bForceVal is written once to the Present Value of the BV object on each rising edge at bForce.

b**ForceVal:** value that is written to the BACnet property Present Value.

### VAR_OUTPUT

```
bPrVal      : BOOL;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

b**PrVal:** current value of the BACnet property Present Value – read directly from the BACnet

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
fbDvc     : FB_BACnet_Device;
```

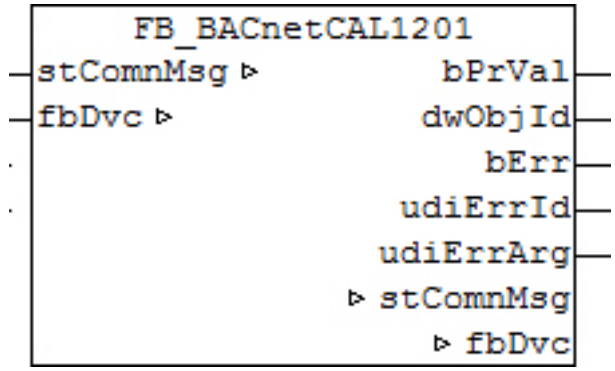
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.29 FB_BACnetCAL1201

BACnet calendar




### Functional description

This function block generates a BACnet calendar object and provides write and read variables for the object within the PLC.

The object is parameterized either from BACnet or via comment lines in the PLC declaration part.

### Inputs/outputs

#### VAR_OUTPUT

```
bPrVal     : BOOL;
dwObjId    : DWORD;
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**bPrVal:** TRUE at this output indicates that the current date is within the parameterized periods.

**dwObjId:** BACnet object ID of the calendar object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

#### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

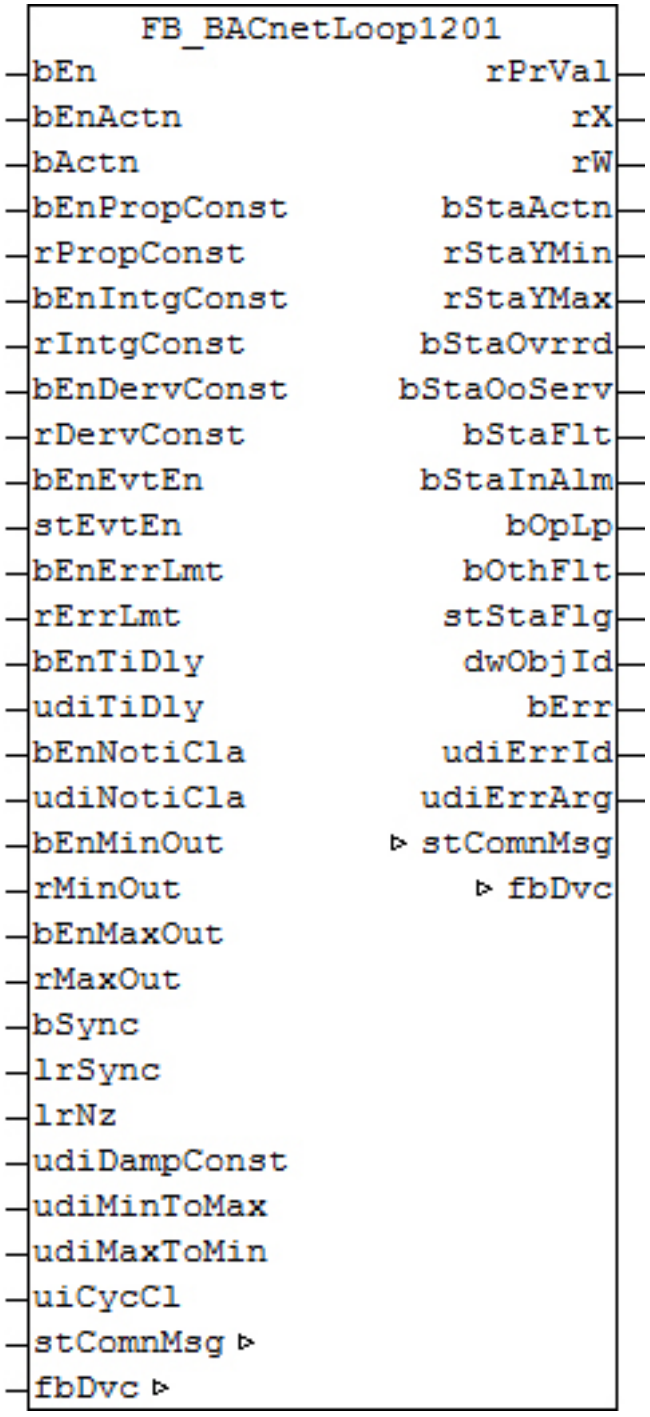
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.30 FB_BACnetLoop1201

Universal BACnet PID controller (large version)




**Functional description**

This function block maps the function block FB_BA_PIDCtrl [▶ 155] as loop object in BACnet. The loop object on the BACnet side essentially has the task of displaying parameters and values from the PLC and transferring them to the PLC. The controller logic is solely in the PLC, in the subordinate FB_BA_PIDCtrl [▶ 155]. The main parameters can be changed from both sides - PLC and BACnet; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnActn | Enable for writing the controller direction of action | |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bActn | Property value control direction<br>bActn=TRUE, BACnet direct → control direction: E=X-W (cooling)<br>bActn=FALSE, BACnet reverse → control direction: E=W-X (heating) | Action (2) |
| bEnPropConst | Enable for writing the proportionality constant (K-factor) | |
| rPropConst | Property value proportionality constant (K-factor) | ProportionalConstant (93) |
| bEnIntgConst | Enable for writing the integration time (I-component) | |
| rIntgConst | Property value integration time [s] (I-component) | IntegralConstant (49) |
| bEnDervConst | Enable for writing the rate time (D-component) | |
| rDervConst | Property value rate time [s] (D-component) | DerivativeConstant (26) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnErrLmt | Enable for writing the message value "maximum value control deviation" | |
| rErrLmt | Property value message value "maximum value control deviation" | ErrorLimit (34) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |
| bEnMinOut | Enable for writing the minimum output at the controller output | |
| rMinOut | Property value minimum output at controller output | MinimumOutput (68) |
| bEnMaxOut | Enable for writing the maximum output at the controller output | |
| rMaxOut | Property value maximum output at controller output | MaximumOutput (61) |

### Inputs/outputs

### VAR_INPUT

```
bEn           : BOOL;
bEnActn       : BOOL;
bActn         : BOOL;
bEnPropConst  : BOOL;
rPropConst    : REAL;
bEnIntgConst  : BOOL;
rIntgConst    : REAL;
bEnDervConst  : BOOL;
rDervConst    : REAL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnErrLmt     : BOOL;
rErrLmt       : REAL;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
bEnMinOut     : BOOL;
rMinOut       : REAL;
bEnMaxOut     : BOOL;
rMaxOut       : REAL;
bSync         : BOOL;
lrSync        : LREAL;
lrNz          : LREAL;
udiDampConst  : UDINT;
udiMinToMax   : UDINT;
udiMaxToMin   : UDINT;
uiCycCl       : UINT;
```

**bEn:** controller activation

**bEnActn / bActn:** enable/property value control direction: "direct" or "reverse"

**bEnPropConst / rPropConst:** enable/property value proportionality constant (K-factor)

**bEnIntgConst / rIntgConst:** enable/property value integration time [s] (I-component)

**bEnDervConst / rDervConst:** enable/property value rate time [s] (D-component)

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnErrLmt / rErrLmt:** enable/property value message value maximum output

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

**bEnMinOut / rMinOut:** enable/property value maximum output at controller output

**bEnMaxOut / rMaxOut:** enable/property value minimum output at controller output

**bSync:** set output to *lrSync*

l**rSync:** synchronization value, to which the control value is set on rising edge at input *bSync*.

**lrNz:** neutral zone

**udiDampConst:** damping time of the D-component [s]

**udiMinToMax:** slope limit of controller output for increase: *udiMinToMax* [s] related to a change from *lrMinOut* to *rMaxOut*.

**udiMaxToMin:** slope limit of controller output for decrease: *udiMaxToMin* [s] related to a change from *lrMaxOut* to *rMinOut*.

**uiCycCl:** call cycle of the function block as a multiple of the cycle time. A zero entry is automatically interpreted as *uiCycleCall* =1.
Example: *tTaskCycleTime* = 20ms, *uiCtrlCycleCall* =10 -> The control algorithm is called every 200 ms. Thus the outputs are also updated only every 200 ms.

### VAR_OUTPUT

```
rPrVal     : REAL;
rX         : REAL;
rW         : REAL;
bStaActn   : BOOL;
rStaYMin   : REAL;
rStaYMax   : REAL;
bStaOvrrd  : BOOL;
bStaOoServ : BOOL;
bStaFlt    : BOOL;
bStaInAlm  : BOOL;
bOpLp      : BOOL;
bOthFlt    : BOOL;
stStaFlg   : ST_BACnet_StatusFlags;
dwObjId    : DWORD;
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**rPrVal:** control value - controller output

**rX:** current valid actual value read from the BACnet.

**rW:** current valid setpoint read from the BACnet.

**bStaActn:** indicates the state of the control direction.

**bStaYMin:** indicates the lower value of the controller output limitation.

**bStaYMax:** indicates the upper value of the controller output limitation.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the loop object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the loop object.

**bStaFlt:** indicates the state of the status flag "Fault" of the loop object.

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the loop object.

**bOpLp:** indicates the state “OpenLoop” of the loop object property “Reliability”.

**bOthFlt:** indicates the state "OtherFault" of the loop object property “Reliability”.

**stStaFlg:** output structure of the BACnet status

**dwObjId:** BACnet object ID of the loop object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.31 FB_BACnetLoop1202

Universal BACnet PID controller. Unlike FB_BACnetLoop1201 [▶ 98], this function block does not permit writing of the following BACnet properties from the PLC:

- Notification Class
- Minimum Output
- Maximum Output


### Functional description

This function block maps the function block FB_BA_PIDCtrl [▶ 155] as loop object in BACnet. The loop object on the BACnet side essentially has the task of displaying parameters and values from the PLC and transferring them to the PLC. The controller logic is solely in the PLC, in the subordinate FB_BA_PIDCtrl [▶ 155]. The main parameters can be changed from both sides - PLC and BACnet; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnActn | Enable for writing the controller direction of action | |
| bActn | Property value control direction<br>bActn=TRUE, BACnet direct → control direction: E=X-W (cooling)<br>bActn=FALSE, BACnet reverse → control direction: E=W-X (heating) | Action (2) |
| bEnPropConst | Enable for writing the proportionality constant (K-factor) | |
| rPropConst | Property value proportionality constant (K-factor) | ProportionalConstant (93) |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnIntgConst | Enable for writing the integration time (I-component) | |
| rIntgConst | Property value integration time [s] (I-component) | IntegralConstant (49) |
| bEnDervConst | Enable for writing the rate time (D-component) | |
| rDervConst | Property value rate time [s] (D-component) | DerivativeConstant (26) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnErrLmt | Enable for writing the message value "maximum value control deviation" | |
| rErrLmt | Property value message value "maximum value control deviation" | ErrorLimit (34) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |

### Inputs/outputs

### VAR_INPUT

```
bEn           : BOOL;
bEnActn       : BOOL;
bActn         : BOOL;
bEnPropConst  : BOOL;
rPropConst    : REAL;
bEnIntgConst  : BOOL;
rIntgConst    : REAL;
bEnDervConst  : BOOL;
rDervConst    : REAL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnErrLmt     : BOOL;
rErrLmt       : REAL;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bSync         : BOOL;
lrSync        : LREAL;
lrNz          : LREAL;
udiDampConst  : UDINT;
udiMinToMax   : UDINT;
udiMaxToMin   : UDINT;
uiCycCl       : UINT;
```

**bEn:** controller activation

**bEnActn / bActn:** enable/property value control direction: "direct" or "reverse"

**bEnPropConst / rPropConst:** enable/property value proportionality constant (K-factor)

**bEnIntgConst / rIntgConst:** enable/property value integration time [s] (I-component)

**bEnDervConst / rDervConst:** enable/property value rate time [s] (D-component)

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnErrLmt / rErrLmt:** enable/property value message value maximum output

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bSync:** set output to *lrSync*.

**lrSync:** synchronization value, to which the control value is set on rising edge at input *bSync*.

**lrNz:** neutral zone

**udiDampConst:** damping time of the D-component [s]

**udiMinToMax:** slope limit of controller output for increase: *udiMinToMax* [s] related to a change from *lrMinOut* to *rMaxOut*.

**udiMaxToMin:** slope limit of controller output for decrease: *udiMaxToMin* [s] related to a change from *lrMaxOut* to *rMinOut*.

**uiCycCl:** call cycle of the function block as a multiple of the cycle time. A zero entry is automatically interpreted as *uiCycleCall* =1.
Example: *tTaskCycleTime* = 20ms, *uiCtrlCycleCall* =10 -> The control algorithm is called every 200 ms. Thus the outputs are also updated only every 200 ms.

### VAR_OUTPUT

```
rPrVal     : REAL;
rX         : REAL;
rW         : REAL;
bStaActn   : BOOL;
rStaYMin   : REAL;
rStaYMax   : REAL;
bStaOvrrd  : BOOL;
bStaOoServ : BOOL;
bStaFlt    : BOOL;
bStaInAlm  : BOOL;
bOpLp      : BOOL;
bOthFlt    : BOOL;
stStaFlg   : ST_BACnet_StatusFlags;
dwObjId    : DWORD;
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**rPrVal:** control value - controller output

**rX:** current valid actual value read from the BACnet.

**rW:** current valid setpoint read from the BACnet.

**bStaActn:** indicates the state of the control direction.

**bStaYMin:** indicates the lower value of the controller output limitation.

**bStaYMax:** indicates the upper value of the controller output limitation.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the loop object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the loop object.

**bStaFlt:** indicates the state of the status flag "Fault" of the loop object.

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the loop object.

**bOpLp:** indicates the state “OpenLoop” of the loop object property “Reliability”.

**bOthFlt:** indicates the state "OtherFault" of the loop object property “Reliability”.

**stStaFlg:** output structure of the BACnet status

**dwObjId:** BACnet object ID of the loop object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

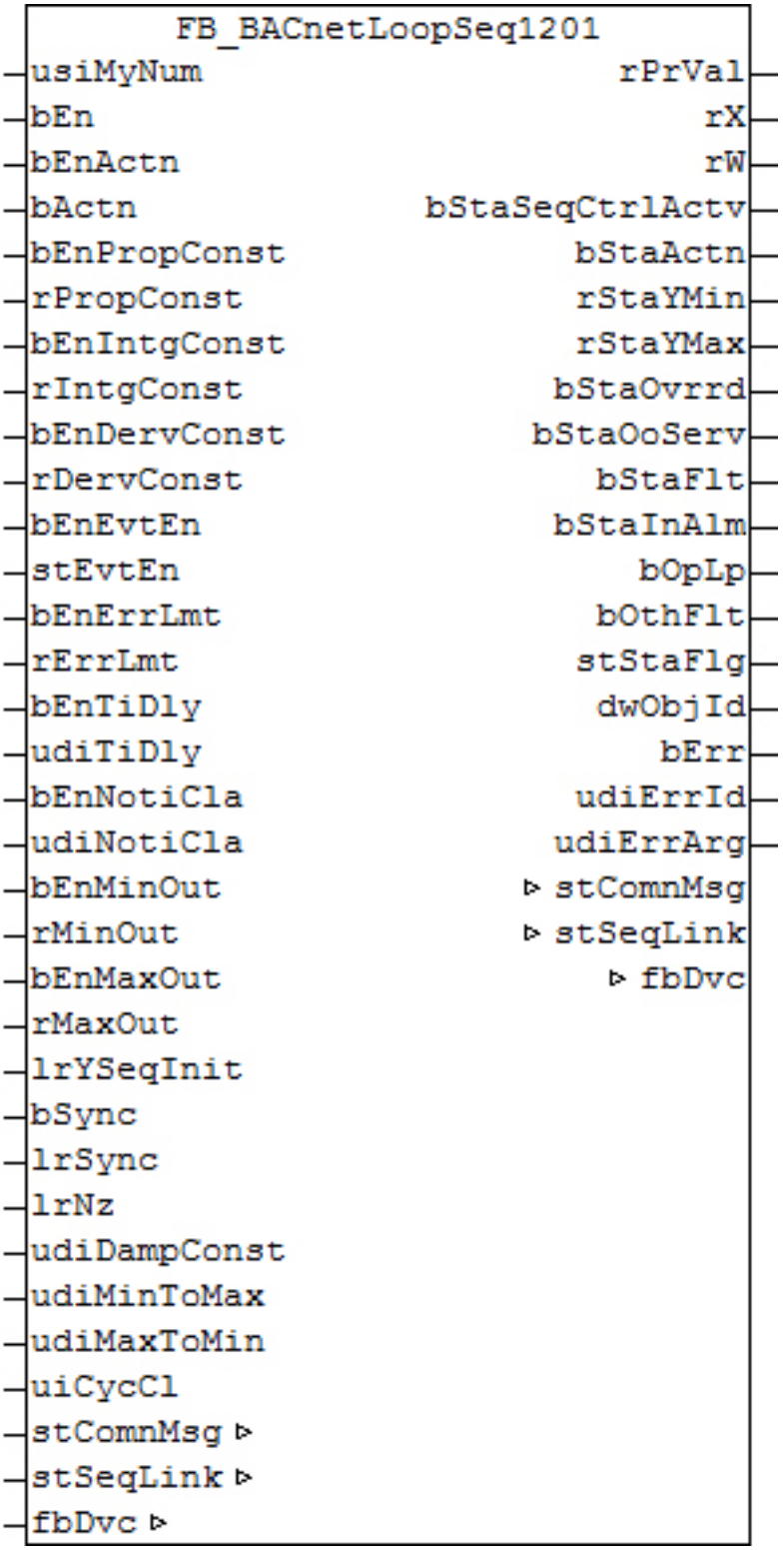
**fbDvc:** reference to the function block of the BACnet device object

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.32 FB_BACnetLoopSeq1201

BACnet sequence controller




**Functional description**

This function block maps the function block FB_BA_SeqCtrl [▶ 165] as loop object in BACnet. The loop object on the BACnet side essentially has the task of displaying parameters and values from the PLC and transferring them to the PLC. The controller logic is solely in the PLC, in the internal FB_BA_SeqCtrl [▶ 165]. The main parameters can be changed from both sides - PLC and BACnet; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for

the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

Like in the "normal" sequence controller of type FB_BA_SeqCtrl [▶ 165], which represents a pure PLC solution, the BACnet version also works only with a sequence link function block FB_BA_SeqLink [▶ 168].

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnActn | Enable for writing the controller direction of action | |
| bActn | Property value control direction<br>bActn=TRUE, BACnet direct → control direction: E=X-W (cooling)<br>bActn=FALSE, BACnet reverse → control direction: E=W-X (heating) | Action (2) |
| bEnPropConst | Enable for writing the proportionality constant (K-factor) | |
| rPropConst | Property value proportionality constant (K-factor) | ProportionalConstant (93) |
| bEnIntgConst | Enable for writing the integration time (I-component) | |
| rIntgConst | Property value integration time [s] (I-component) | IntegralConstant (49) |
| bEnDervConst | Enable for writing the rate time (D-component) | |
| rDervConst | Property value rate time [s] (D-component) | DerivativeConstant (26) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnErrLmt | Enable for writing the message value "maximum value control deviation" | |
| rErrLmt | Property value message value "maximum value control deviation" | ErrorLimit (34) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |
| bEnMinOut | Enable for writing the minimum output at the controller output | |
| rMinOut | Property value minimum output at controller output | MinimumOutput (68) |
| bEnMaxOut | Enable for writing the maximum output at the controller output | |
| rMaxOut | Property value maximum output at controller output | MaximumOutput (61) |

### Inputs/outputs

### VAR_INPUT

```
bEn           : BOOL;
usiMyNum      : USINT;
bEnActn       : BOOL;
bActn         : BOOL;
bEnPropConst  : BOOL;
rPropConst    : REAL;
bEnIntgConst  : BOOL;
rIntgConst    : REAL;
bEnDervConst  : BOOL;
rDervConst    : REAL;
bEnEvtEn      : BOOL;
stEvtEn       : ST_BACnet_EventTransitionBits;
bEnErrLmt     : BOOL;
rErrLmt       : REAL;
bEnTiDly      : BOOL;
udiTiDly      : UDINT;
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
bEnMinOut     : BOOL;
rMinOut       : REAL;
```

```
bEnMaxOut     : BOOL;
rMaxOut       : REAL;
bSync         : BOOL;
lrYSeqInit    : LREAL;
lrSync        : LREAL;
lrNz          : LREAL;
udiDampConst  : UDINT;
udiMinToMax   : UDINT;
udiMaxToMin   : UDINT;
uiCycCl       : UINT;
```

**usiMyNum:** ordinal number of the sequence controller

**bEn:** controller activation

**bEnActn / bActn:** enable/property value control direction: "direct" or "reverse

**bEnPropConst / rPropConst:** enable/property value proportionality constant (K-factor)

**bEnIntgConst / rIntgConst:** enable/property value integration time [s] (I-component)

**bEnDervConst / rDervConst:** enable/property value rate time [s] (D-component)

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnErrLmt / rErrLmt:** enable/property value message value maximum output

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

**bEnMinOut / rMinOut:** enable/property value maximum output at controller output

**bEnMaxOut / rMaxOut:** enable/property value minimum output at controller output

**lrYSeqInit:** start synchronization value. If this controller is the one that is enabled first when the control sequence is activated, this controller is started with this value at the output.

**bSync:** set output to *lrSync*.

**lrSync:** synchronization value, to which the control value is set on rising edge at input *bSync*.

**lrNz:** neutral zone

**udiDampConst:** damping time of the D-component [s]

**udiMinToMax:** slope limit of controller output for increase: *udiMinToMax* [s] related to a change from *lrMinOut* to *rMaxOut*.

**udiMaxToMin:** slope limit of controller output for decrease: *udiMaxToMin* [s] related to a change from *lrMaxOut* to *rMinOut*.

**uiCycCl:** call cycle of the function block as a multiple of the cycle time. A zero entry is automatically interpreted as *uiCycleCall* =1.
Example: *tTaskCycleTime* = 20ms, *uiCtrlCycleCall* =10 -> The control algorithm is called every 200 ms. Thus the outputs are also updated only every 200 ms.

### VAR_OUTPUT

```
rPrVal          : REAL;
rX              : REAL;
rW              : REAL;
bStaSeqCtrlActv : BOOL;
bStaActn        : BOOL;
rStaYMin        : REAL;
rStaYMax        : REAL;
bStaOvrrd       : BOOL;
bStaOoServ      : BOOL;
bStaFlt         : BOOL;
bStaInAlm       : BOOL;
bOpLp           : BOOL;
bOthFlt         : BOOL;
stStaFlg        : ST_BACnet_StatusFlags;
dwObjId         : DWORD;
```

```
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**rPrVal:** control value - controller output

**rX:** current valid actual value read from the BACnet.

**rW:** current valid setpoint read from the BACnet.

**bStaSeqCtrlActv:** sequence controller is enabled and ready for operation (not in error state).

**bStaActn:** indicates the state of the control direction.

**bStaYMin:** indicates the lower value of the controller output limitation.

**bStaYMax:** indicates the upper value of the controller output limitation.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the loop object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the loop object.

**bStaFlt:** indicates the state of the status flag "Fault" of the loop object.

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the loop object.

**bOpLp:** indicates the state “OpenLoop” of the loop object property “Reliability”.

**bOthFlt:** indicates the state "OtherFault" of the loop object property “Reliability”.

**stStaFlg:** output structure of the BACnet status

**dwObjId:** BACnet object ID of the loop object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
stSeqLink : ST_BA_SeqLink;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

**stSeqLink:** data and command structure [▶ 327] between the individual sequence controllers and the function block FB_BA_SeqLink [▶ 168].

**fbDvc:** reference to the function block of the BACnet device object

### Requirements

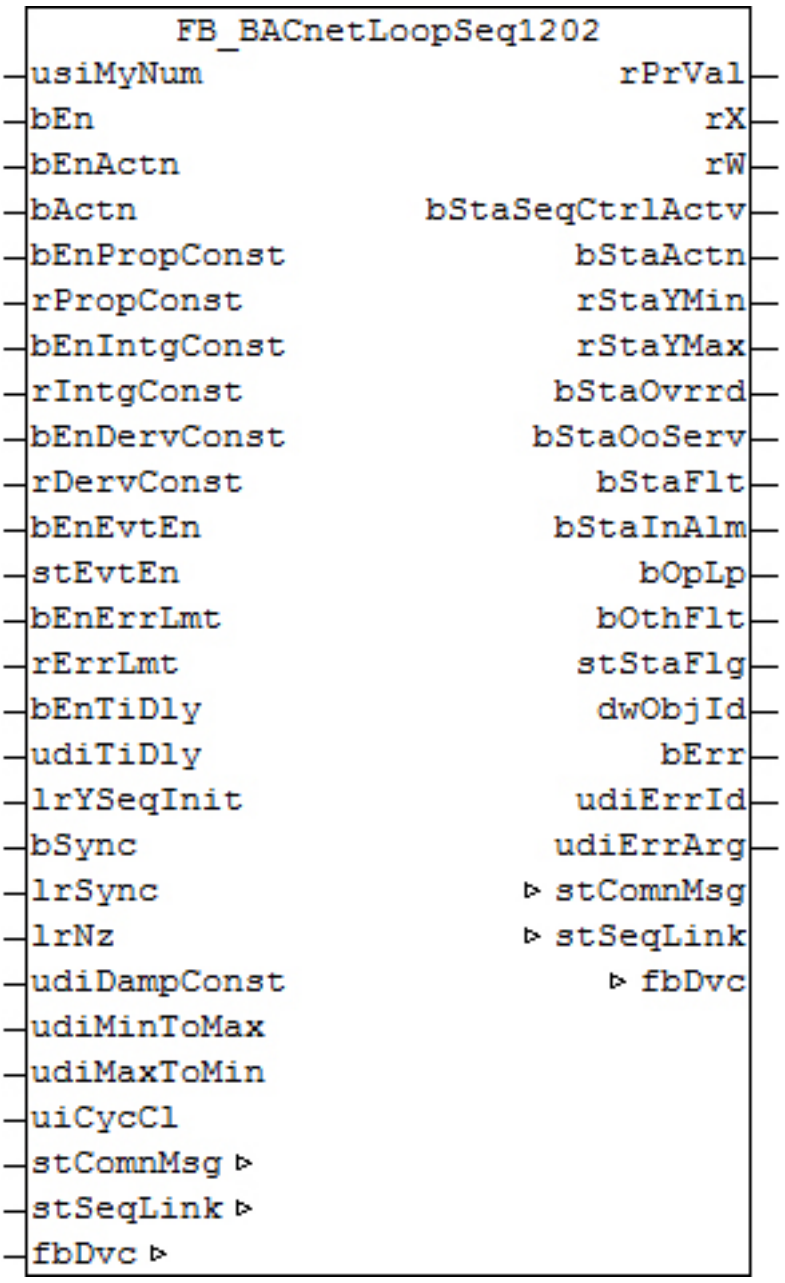
| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.33 FB_BACnetLoopSeq1202

BACnet sequence controller. Unlike FB_BACnetLoopSeq1201 [▶ 106], this function block does not permit writing of the following BACnet properties from the PLC:

- Notification Class
- Minimum Output

- Maximum Output




**Functional description**

This function block maps the function block FB_BA_SeqCtrl [▶ 165] as loop object in BACnet. The loop object on the BACnet side essentially has the task of displaying parameters and values from the PLC and transferring them to the PLC. The controller logic is solely in the PLC, in the internal FB_BA_SeqCtrl [▶ 165]. The main parameters can be changed from both sides - PLC and BACnet; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

Like in the "normal" sequence controller of type FB_BA_SeqCtrl [▶ 165], which represents a pure PLC solution, the BACnet version also works only with a sequence link function block FB_BA_SeqLink [▶ 168].

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnActn | Enable for writing the controller direction of action | |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bActn | Property value control direction<br>bActn=TRUE, BACnet direct → control direction: E=X-W (cooling)<br>bActn=FALSE, BACnet reverse → control direction: E=W-X (heating) | Action (2) |
| bEnPropConst | Enable for writing the proportionality constant (K-factor) | |
| rPropConst | Property value proportionality constant (K-factor) | ProportionalConstant (93) |
| bEnIntgConst | Enable for writing the integration time (I-component) | |
| rIntgConst | Property value integration time [s] (I-component) | IntegralConstant (49) |
| bEnDervConst | Enable for writing the rate time (D-component) | |
| rDervConst | Property value rate time [s] (D-component) | DerivativeConstant (26) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnErrLmt | Enable for writing the message value "maximum value control deviation" | |
| rErrLmt | Property value message value "maximum value control deviation" | ErrorLimit (34) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |

### Inputs/outputs

### VAR_INPUT

```
usiMyNum     : USINT;
bEn          : BOOL;
bEnActn      : BOOL;
bActn        : BOOL;
bEnPropConst : BOOL;
rPropConst   : REAL;
bEnIntgConst : BOOL;
rIntgConst   : REAL;
bEnDervConst : BOOL;
rDervConst   : REAL;
bEnEvtEn     : BOOL;
stEvtEn      : ST_BACnet_EventTransitionBits;
bEnErrLmt    : BOOL;
rErrLmt      : REAL;
bEnTiDly     : BOOL;
udiTiDly     : UDINT;
lrYSeqInit   : LREAL;
bSync        : BOOL;
lrSync       : LREAL;
lrNz         : LREAL;
udiDampConst : UDINT;
udiMinToMax  : UDINT;
udiMaxToMin  : UDINT;
uiCycCl      : UINT;
```

**usiMyNum:** ordinal number of the sequence controller

**bEn:** controller activation

**bEnActn / bActn:** enable/property value control direction: "direct" or "reverse

**bEnPropConst / rPropConst:** enable/property value proportionality constant (K-factor)

**bEnIntgConst / rIntgConst:** enable/property value integration time [s] (I-component)

**bEnDervConst / rDervConst:** enable/property value rate time [s] (D-component)

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnErrLmt / rErrLmt:** enable/property value message value maximum output

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**lrYSeqInit:** start synchronization value. If this controller is the one that is enabled first when the control sequence is activated, this controller is started with this value at the output.

**bSync:** set output to *lrSync*.

**lrSync:** synchronization value, to which the control value is set on rising edge at input *bSync*.

**lrNz:** neutral zone

**udiDampConst:** damping time of the D-component [s]

**udiMinToMax:** slope limit of controller output for increase: *udiMinToMax* [s] related to a change from *lrMinOut* to *rMaxOut*.

**udiMaxToMin:** slope limit of controller output for decrease: *udiMaxToMin* [s] related to a change from *lrMaxOut* to *rMinOut*.

**uiCycCl:** call cycle of the function block as a multiple of the cycle time. A zero entry is automatically interpreted as *uiCycleCall* =1.
Example: *tTaskCycleTime* = 20ms, *uiCtrlCycleCall* =10 -> The control algorithm is called every 200 ms. Thus the outputs are also updated only every 200 ms.

### VAR_OUTPUT

```
rPrVal          : REAL;
rX              : REAL;
rW              : REAL;
bStaSeqCtrlActv : BOOL;
bStaActn        : BOOL;
rStaYMin        : REAL;
rStaYMax        : REAL;
bStaOvrrd       : BOOL;
bStaOoServ      : BOOL;
bStaFlt         : BOOL;
bStaInAlm       : BOOL;
bOpLp           : BOOL;
bOthFlt         : BOOL;
stStaFlg        : ST_BACnet_StatusFlags;
dwObjId         : DWORD;
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**rPrVal:** control value - controller output

**rX:** current valid actual value read from the BACnet.

**rW:** current valid setpoint read from the BACnet.

**bStaSeqCtrlActv:** sequence controller is enabled and ready for operation (not in error state).

**bStaActn:** indicates the state of the control direction.

**bStaYMin:** indicates the lower value of the controller output limitation.

**bStaYMax:** indicates the upper value of the controller output limitation.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the loop object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the loop object.

**bStaFlt:** indicates the state of the status flag "Fault" of the loop object.

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the loop object.

**bOpLp:** indicates the state “OpenLoop” of the loop object property “Reliability”.

**bOthFlt:** indicates the state "OtherFault" of the loop object property “Reliability”.

**stStaFlg:** output structure of the BACnet status

**dwObjId:** BACnet object ID of the loop object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
stSeqLink : ST_BA_SeqLink;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

**stSeqLink:** data and command structure [▶ 327] between the individual sequence controllers and the function block FB_BA_SeqLink [▶ 168].

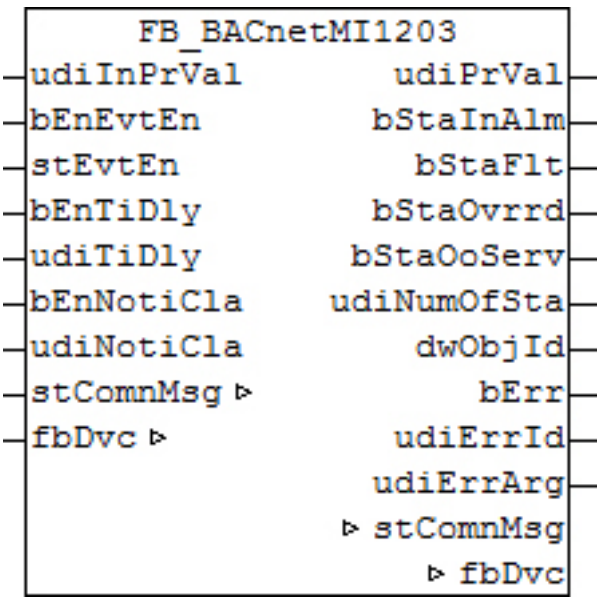
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.34 FB_BACnetMI1203

BACnet multistate input




### Functional description

This function block generates a BACnet multistate input object and provides write and read variables for the object within the PLC.

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective

Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

### VAR_INPUT

```
udiInPrVal  : UDINT
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnTiDly    : BOOL;
udiTiDly    : UDINT;
bEnNotiCla  : BOOL;
udiNotiCla  : UDINT;
```

**udiInPrVal:** input value of the hardware. The status range is 1 to 1000. The input signals of the hardware (e.g. n digital inputs for a rotary switch) may have to be pre-coded via a logic such that values are represented as part of the valid range.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

### VAR_OUTPUT

```
udiPrVal       : UDINT;
bStaInAlm      : BOOL;
bStaFlt        : BOOL;
bStaOvrrd      : BOOL;
bStaOoServ     : BOOL;
udiNumOfSta    : UDINT;
dwObjId        : DWORD;
bErr           : BOOL;
udiErrId       : UDINT;
udiErrArg      : UDINT;
```

**udiPrVal:** current value of the multistate input object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the multistate input object.

**bStaFlt:** indicates the state of the status flag "Fault" of the multistate input object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the analog value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the multistate input object.

**udiNumOfSta:** indicates the set number of states (NumberOfStates, property ID 74).

**dwObjId:** BACnet object ID of the multistate input object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

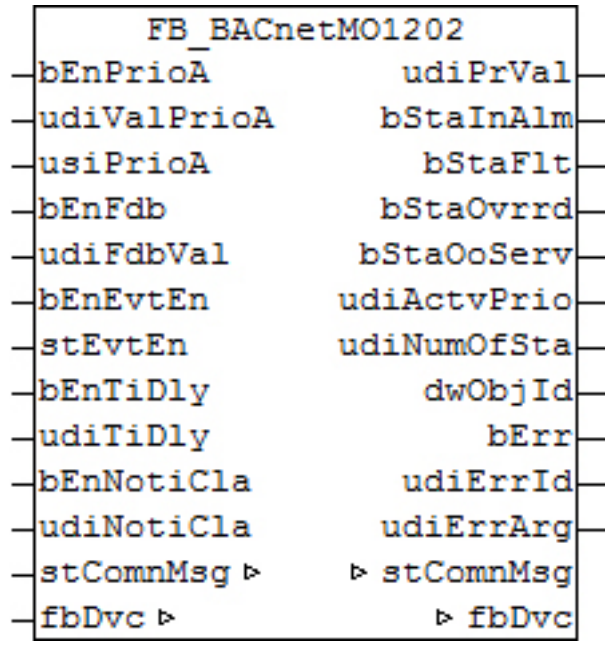
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.35 FB_BACnetMO1202

BACnet multistate output




### Functional description

This function block generates a BACnet multistate output object and provides write and read variables for the object within the PLC.
This function block is the "small" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetMO1203 [▶ 118]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| udiValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
udiValPrioA : UDINT;
usiPrioA    : USINT;
bEnFdb      : BOOL;
udiFdbVal   : UDINT;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnTiDly    : BOOL;
udiTiDly    : UDINT;
bEnNotiCla  : BOOL;
udiNotiCla  : UDINT;
```

**bEnPrioA:** enable for writing

**udiValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnFdbVal / udiFdbVal:** enable/property value feedback value. This value is written from the PLC to the Property FeedbackValue (Property Id 40), however **not** via ADS, but via cyclic interface.
It is used for feedback monitoring: if *bEnFdbVal* is set to TRUE, *udiFdbVal* must have the same value as the output Present Value within the time specified in the BACnet under TIME Delay (Property Id 113). Otherwise a feedback discrepancy is displayed in the EventState (property ID 36) via the entry "offnormal". If *bEnFdbVal* is set to FALSE, internally *udiFdbVal* is set to same value as the Present Value, so that feedback discrepancies are avoided.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

### VAR_OUTPUT

```
udiPrVal       : UDINT;
bStaInAlm      : BOOL;
bStaFlt        : BOOL;
bStaOvrrd      : BOOL;
bStaOoServ     : BOOL;
udiActvPrio    : UDINT;
udiNumOfSta    : UDINT;
dwObjId        : DWORD;
bErr           : BOOL;
udiErrId       : UDINT;
udiErrArg      : UDINT;
```

**udiPrVal:** current value of the multistate output object - read directly from the BACnet. This value may require a subsequent logic, if it is to be linked with a hardware (indicator panel, etc.).

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the multistate output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the multistate output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the multistate output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the multistate output object.

**udiActvPrio:** indicates which priority is active.

**udiNumOfSta:** indicates the set number of states (NumberOfStates, property ID 74).

**dwObjId:** BACnet object ID of the multistate output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

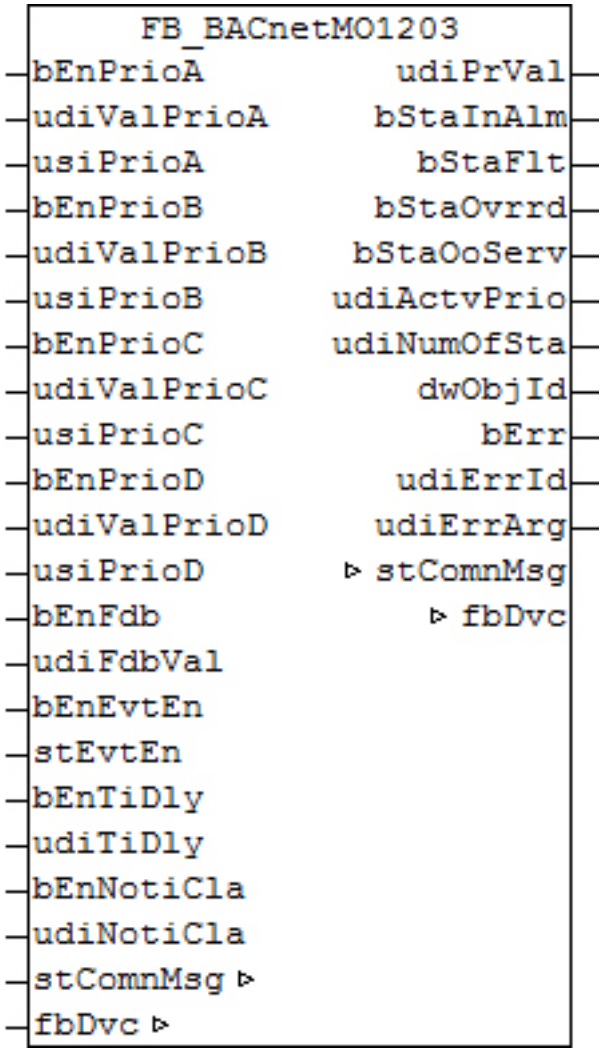
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.36 FB_BACnetMO1203

BACnet multistate output




**Functional description**

This function block generates a BACnet multistate output object and provides write and read variables for the object within the PLC.
This function block is the "large" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetMO1202 [▶ 116]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| udiValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| udiValPrioB | Value to be written | Priority-Array (87) |

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| udiValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| udiValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
udiValPrioA : UDINT;
usiPrioA    : USINT;
bEnPrioB    : BOOL;
udiValPrioB : UDINT;
usiPrioB    : USINT;
bEnPrioC    : BOOL;
udiValPrioC : UDINT;
usiPrioC    : USINT;
bEnPrioD    : BOOL;
udiValPrioD : UDINT;
usiPrioD    : USINT;
bEnFdb      : BOOL;
udiFdbVal   : UDINT;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnTiDly    : BOOL;
udiTiDly    : UDINT;
bEnNotiCla  : BOOL;
udiNotiCla  : UDINT;
```

**bEnPrioA:** enable for writing

**udiValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**udiValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**udiValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written.

**bEnPrioD:** enable for writing

**udiValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bEnFdbVal / udiFdbVal:** enable/property value feedback value. This value is written from the PLC to the Property FeedbackValue (Property Id 40), however **not** via ADS, but via cyclic interface.
It is used for feedback monitoring: if *bEnFdbVal* is set to TRUE, *udiFdbVal* must have the same value as the output Present Value within the time specified in the BACnet under TIME Delay (Property Id 113). Otherwise a feedback discrepancy is displayed in the EventState (property ID 36) via the entry "offnormal". If *bEnFdbVal* is set to FALSE, internally *udiFdbVal* is set to same value as the Present Value, so that feedback discrepancies are avoided.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

### VAR_OUTPUT

```
udiPrVal        : UDINT;
bStaInAlm       : BOOL;
bStaFlt         : BOOL;
bStaOvrrd       : BOOL;
bStaOoServ      : BOOL;
udiActvPrio     : UDINT;
udiNumOfSta     : UDINT;
dwObjId         : DWORD;
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**udiPrVal:** current value of the multistate output object - read directly from the BACnet. This value may require a subsequent logic, if it is to be linked with a hardware (indicator panel, etc.).

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the multistate output object.

**bStaFlt:** indicates the state of the status flag "Fault" of the multistate output object.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the multistate output object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the multistate output object.

**udiActvPrio:** indicates which priority is active.

**udiNumOfSta:** indicates the set number of states (NumberOfStates, property ID 74).

**dwObjId:** BACnet object ID of the multistate output object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

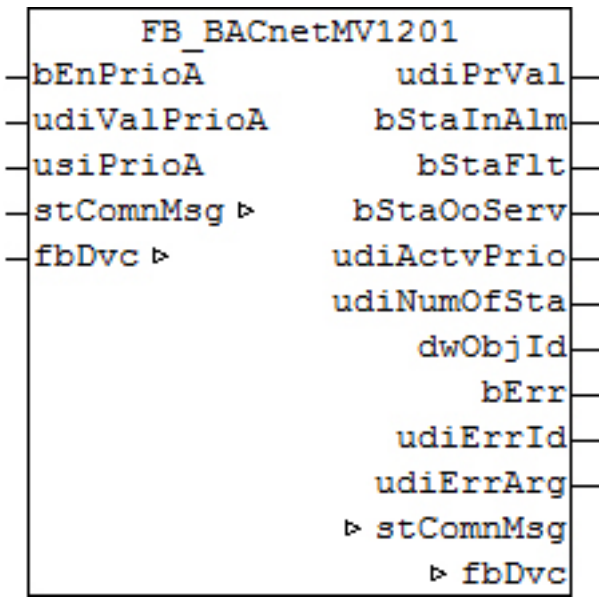
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.37 FB_BACnetMV1201

BACnet multistate value




### Functional description

This function block generates a BACnet multistate value object and provides write and read variables for the object within the PLC.
This function block is the "smallest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetMV1202 [▶ 123]
- FB_BACnetMV1203 [▶ 126]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| udiValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
udiValPrioA : UDINT;
usiPrioA    : USINT;
```

**bEnPrioA:** enable for writing

**udiValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

### VAR_OUTPUT

```
udiPrVal    : UDINT;
bStaInAlm   : BOOL;
bStaFlt     : BOOL;
bStaOoServ  : BOOL;
udiActvPrio : UDINT;
udiNumOfSta : UDINT;
dwObjId     : DWORD;
bErrs       : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**udiPrVal:** current value of the multistate value object - read directly from the BACnet

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the multistate value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the multistate value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the multistate value object.

**udiActvPrio:** indicates which priority is active.

**udiNumOfSta:** indicates the set number of states (NumberOfStates, property ID 74).

**dwObjId:** BACnet object ID of the multistate value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

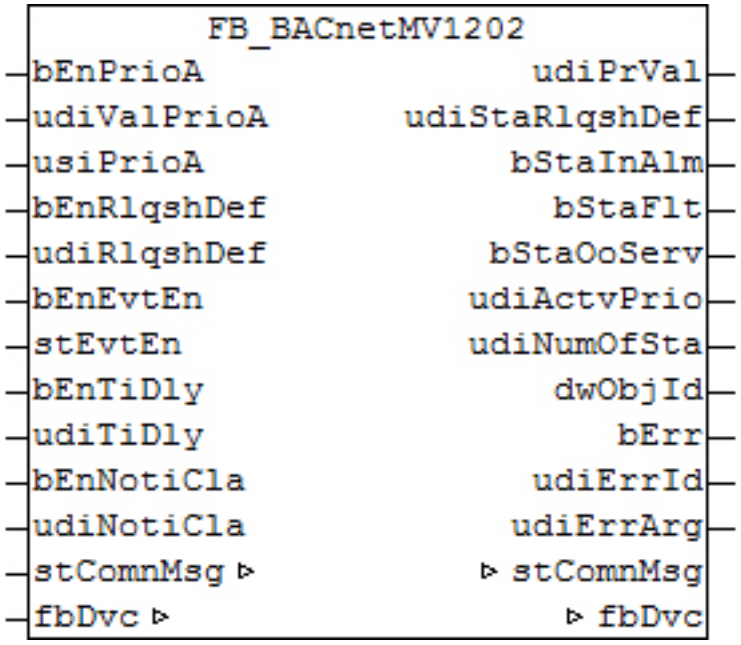
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.38 FB_BACnetMV1202

BACnet multistate value




### Functional description

This function block generates a BACnet multistate value object and provides write and read variables for the object within the PLC.
This function block is the "medium" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetMV1201 [▶ 122]
- FB_BACnetMV1203 [▶ 126]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| udiValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnRlqshDef | Enable for writing the relinquish default | |
| udiRlqshDef | Value to be written to the property relinquish default | Relinquish-Default (104) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

#### VAR_INPUT

```
bEnPrioA    : BOOL;
udiValPrioA : UDINT;
usiPrioA    : USINT;
bEnRlqshDef : BOOL;
udiRlqshDef : UDINT;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnTiDly    : BOOL;
udiTiDly    : UDINT;
bEnNotiCla  : BOOL;
udiNotiCla  : UDINT;
```

**bEnPrioA:** enable for writing

**udiValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnRlqshDef / udiRlqshDef:** enable/property value relinquish default

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

#### VAR_OUTPUT

```
udiPrVal       : UDINT;
udiStaRlqshDef : UDINT;
bStaInAlm      : BOOL;
bStaFlt        : BOOL;
bStaOoServ     : BOOL;
udiActvPrio    : UDINT;
udiNumOfSta    : UDINT;
dwObjId        : DWORD;
bErr           : BOOL;
udiErrId       : UDINT;
udiErrArg      : UDINT;
```

**udiPrVal:** current value of the multistate value object - read directly from the BACnet

**udiStaRlqshDef:** indicates the state of the relinquish default.

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the multistate value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the multistate value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the multistate value object.

**udiActvPrio:** indicates which priority is active.

**udiNumOfSta:** indicates the set number of states (NumberOfStates, property ID 74).

**dwObjId:** BACnet object ID of the multistate value object

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

#### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

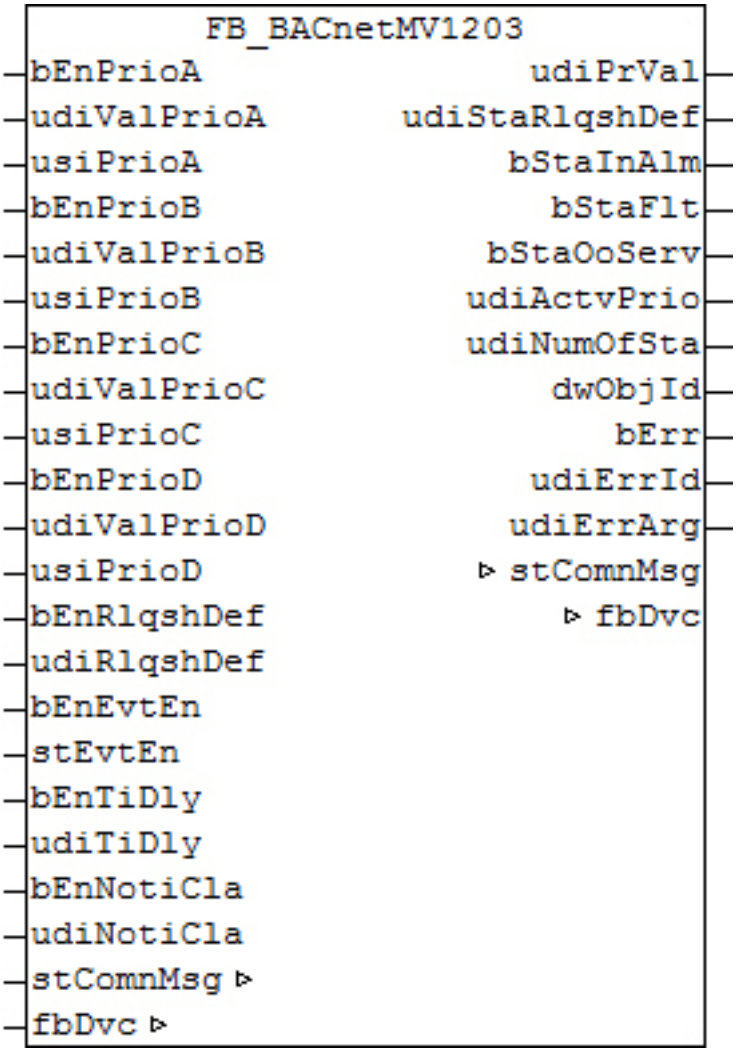
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.39 FB_BACnetMV1203

BACnet multistate value




### Functional description

This function block generates a BACnet multistate value object and provides write and read variables for the object within the PLC.
This function block is the "largest" version in terms of functionality.
Alternatively the following versions are available:

- FB_BACnetMV1201 [▶ 122]
- FB_BACnetMV1202 [▶ 123]
- FB_BACnetMVDisplay [▶ 129]
- FB_BACnetMVSetpoint [▶ 130]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnPrioA | Enable for writing the priority entered under *usiPrioA* into the priority array | |
| udiValPrioA | Value to be written | Priority-Array (87) |
| usiPrioA | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioB | Enable for writing the priority entered under *usiPrioB* into the priority array | |
| udiValPrioB | Value to be written | Priority-Array (87) |
| usiPrioB | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioC | Enable for writing the priority entered under *usiPrioC* into the priority array | |
| udiValPrioC | Value to be written | Priority-Array (87) |
| usiPrioC | Selection of the priority (1..16) to be written in the priority array | |
| bEnPrioD | Enable for writing the priority entered under *usiPrioD* into the priority array | |
| udiValPrioD | Value to be written | Priority-Array (87) |
| usiPrioD | Selection of the priority (1..16) to be written in the priority array | |
| bEnRlqshDef | Enable for writing the relinquish default | |
| udiRlqshDef | Value to be written to the property relinquish default | Relinquish-Default (104) |
| bEnEvtEn | Enable for writing the EventEnable bit pattern | |
| stEvtEn | Property value bit pattern EventEnable | EventEnable (35) |
| bEnTiDly | Enable for writing the message delay [s] | |
| udiTiDly | Property value message delay [s] | TimeDelay (113) |
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

### VAR_INPUT

```
bEnPrioA    : BOOL;
udiValPrioA : UDINT;
usiPrioA    : USINT;
bEnPrioB    : BOOL;
udiValPrioB : UDINT;
usiPrioB    : USINT;
bEnPrioC    : BOOL;
udiValPrioC : UDINT;
usiPrioC    : USINT;
bEnPrioD    : BOOL;
udiValPrioD : UDINT;
usiPrioD    : USINT;
bEnRlqshDef : BOOL;
udiRlqshDef : UDINT;
bEnEvtEn    : BOOL;
stEvtEn     : ST_BACnet_EventTransitionBits;
bEnTiDly    : BOOL;
udiTiDly    : UDINT;
bEnNotiCla  : BOOL;
udiNotiCla  : UDINT;
```

**bEnPrioA:** enable for writing

**udiValPrioA:** value that is written to priority *usiPrioA* in the priority array.

**usiPrioA:** selection of the priority (1..16) to be written

**bEnPrioB:** enable for writing

**udiValPrioB:** value that is written to priority *usiPrioB* in the priority array.

**usiPrioB:** selection of the priority (1..16) to be written

**bEnPrioC:** enable for writing

**udiValPrioC:** value that is written to priority *usiPrioC* in the priority array.

**usiPrioC:** selection of the priority (1..16) to be written

**bEnPrioD:** enable for writing

**udiValPrioD:** value that is written to priority *usiPrioD* in the priority array.

**usiPrioD:** selection of the priority (1..16) to be written

**bEnRlqshDef / udiRlqshDef:** enable/property value relinquish default.

**bEnEvtEn / stEvtEn:** enable/property value bit pattern EventEnable

**bEnTiDly / udiTiDly:** enable/property value message delay [s]

**bEnNotiCla / udiNotiCla:** enable/property value notification class

### VAR_OUTPUT

```
udiPrVal       : UDINT;
udiStaRlqshDef : UDINT;
bStaInAlm      : BOOL;
bStaFlt        : BOOL;
bStaOoServ     : BOOL;
udiActvPrio    : UDINT;
udiNumOfSta    : UDINT;
dwObjId        : DWORD;
bErr           : BOOL;
udiErrId       : UDINT;
udiErrArg      : UDINT;
```

**udiPrVal:** current value of the multistate value object - read directly from the BACnet

**udiStaRlqshDef:** indicates the state of the relinquish default.

**bStaInAlm:** indicates the state of the status flag "InAlarm" of the multistate value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the multistate value object.

**bStaOoServ:** indicates the state of the status flag "OutOfService" of the multistate value object.

**udiActvPrio:** indicates which priority is active.

**udiNumOfSta:** indicates the set number of states (NumberOfStates, property ID 74).

**dwObjId:** BACnet object ID of the multistate value object.

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

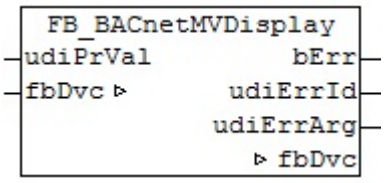
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.40 FB_BACnetMVDisplay

BACnet MultiState value object that can be used to display a value from the PLC in the BACnet.




### Functional description

This function block generates a BACnet multistate value object and provides write and read variables for the object within the PLC.
Alternatively, the following versions are available:

- FB_BACnetMV1201 [▶ 122]
- FB_BACnetMV1202 [▶ 123]
- FB_BACnetMV1203 [▶ 126]
- FB_BACnetMVSetpoint [▶ 130]

### Inputs/outputs

#### VAR_INPUT

```
udiPrVal    : UDINT;
```

udi**PrVal:** Value from the PLC that is written to the BACnet property Present Value in the event of a change of value.

#### VAR_OUTPUT

```
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**bErr:** Indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterisation.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

#### VAR_IN_OUT

```
fbDvc     : FB_BACnet_Device;
```

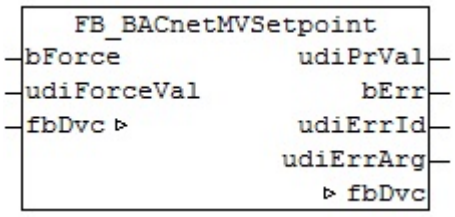
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.41 FB_BACnetMVSetpoint

BACnet MultiState value object that maps the BACnet property Present Value in the PLC.




### Functional description

This function block generates a BACnet multistate value object and provides write and read variables for the object within the PLC.
Alternatively, the following versions are available:

- FB_BACnetMV1201 [▶ 122]
- FB_BACnetMV1202 [▶ 123]
- FB_BACnetMV1203 [▶ 126]
- FB_BACnetMVDisplay [▶ 129]

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface.
The value of udiForceVal is written once to the Present Value of the MV object on each rising edge at bForce.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bForce | The value of udiForceVal is written once to the Present Value of the MV object on each rising edge at bForce. | |
| udiForceVal | Value to be written to the BACnet property Present Value | Present Value (85) |

### Inputs/outputs

#### VAR_INPUT

```
bForce      : BOOL;
udiForceVal : UDINT;
```

**bForce:** The value of rForceVal is written once to the Present Value of the AV object on each rising edge at bForce.

**udiForceVal:** value that is written to the BACnet property Present Value.

#### VAR_OUTPUT

```
udiPrVal      : UDINT;
bErr          : BOOL;
udiErrId      : UDINT;
udiErrArg     : UDINT;
```

**udiPrVal:** current value of the BACnet property Present Value – read directly from the BACnet

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
fbDvc     : FB_BACnet_Device;
```

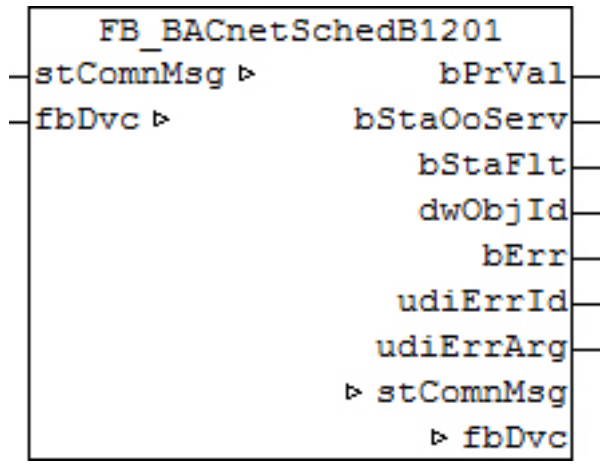
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.42 FB_BACnetSchedB1201

BACnet scheduler of output type BOOL




### Functional description

This function block generates a BACnet scheduler object and provides write and read variables for the object within the PLC. The output type of the object is BOOL.

The object is parameterized either from BACnet or via comment lines in the PLC declaration part.

### Inputs/outputs

### VAR_OUTPUT

```
bPrVal     : BOOL;
bStaOoServ : BOOL;
bStaFlt    : BOOL;
dwObjId    : DWORD;
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**bPrVal:** output value of the schedule depending on the currently selected date/time and the schedule entries.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary value object.

**dwObjId:** BACnet object ID of the binary value object.

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

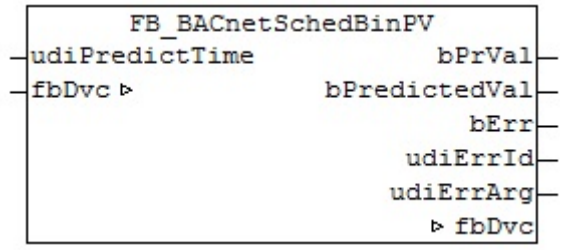
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.43 FB_BACnetSchedBinPV

BACnet scheduler of the type Binary Present Value with the function "predictive switch-on and switch-off time".




### Functional description

This function block generates a BACnet scheduler object and provides write and read variables for the object within the PLC. The output type of the object is BOOL.

The object is parameterized either from BACnet or via comment lines in the PLC declaration part.

### Inputs/outputs

### VAR_INPUT

```
udiPredictTime : UDINT;
```

**udiPredictTime:** predicted switch-on and switch-off time in seconds. The time value specified here switches the output bPredictedVal on or off in advance, depending on the timer channels of the BACnet property WeeklySchedule.

### VAR_OUTPUT

```
bPrVal         : BOOL;
bPredictedVal  : BOOL;
bErr           : BOOL;
udiErrId       : UDINT;
udiErrArg      : UDINT;
```

**bPrVal:** Output value of the schedule depending on the currently selected date/time and the schedule entries.

**bPredictedVal:** output that is switched depending on the switch-on and switch-off time udiPredictTime and the BACnet property WeeklySchedule.

**bErr:** Indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### VAR_IN_OUT

```
fbDvc     : FB_BACnet_Device;
```

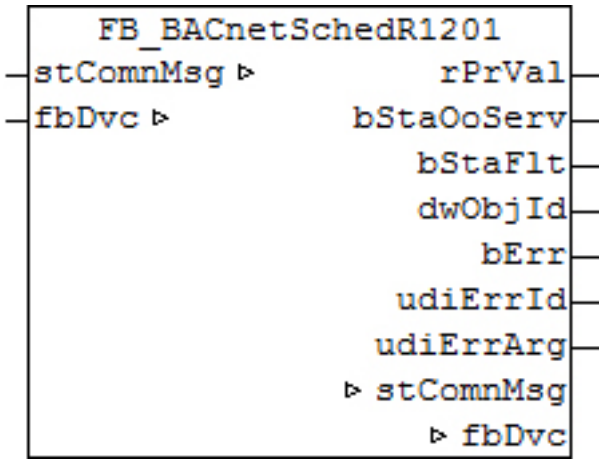
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.44 FB_BACnetSchedR1201

BACnet scheduler of output type REAL




### Functional description

This function block generates a BACnet scheduler object and provides write and read variables for the object within the PLC. The output type of the object is REAL.

The object is parameterized either from BACnet or via comment lines in the PLC declaration part.

### Inputs/outputs

### VAR_OUTPUT

```
rPrVal     : REAL;
bStaOoServ : BOOL;
bStaFlt    : BOOL;
dwObjId    : DWORD;
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**rPrVal:** output value of the schedule depending on the currently selected date/time and the schedule entries.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary value object.

**dwObjId:** BACnet object ID of the binary value object.

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

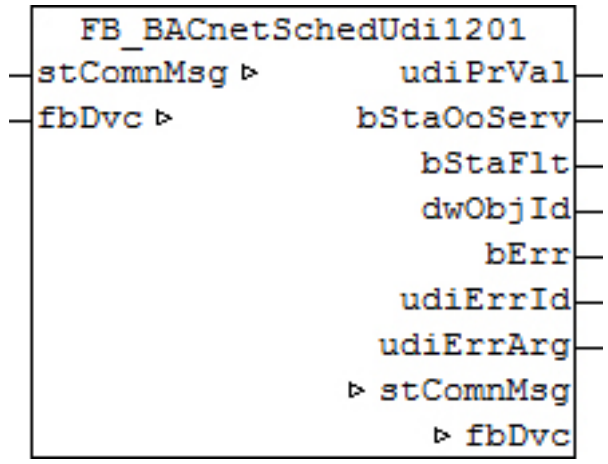
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.45 FB_BACnetSchedUdi1201

BACnet scheduler of output type UDINT




### Functional description

This function block generates a BACnet scheduler object and provides write and read variables for the object within the PLC. The output type of the object is UDINT.

The object is parameterized either from BACnet or via comment lines in the PLC declaration part.

### Inputs/outputs

### VAR_OUTPUT

```
udiPrVal   : UDINT;
bStaOoServ : BOOL;
bStaFlt    : BOOL;
dwObjId    : DWORD;
```

```
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**udiPrVal:** output value of the schedule depending on the currently selected date/time and the schedule entries.

**bStaOvrrd:** indicates the state of the status flag "Overridden" of the binary value object.

**bStaFlt:** indicates the state of the status flag "Fault" of the binary value object.

**dwObjId:** BACnet object ID of the binary value object.

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

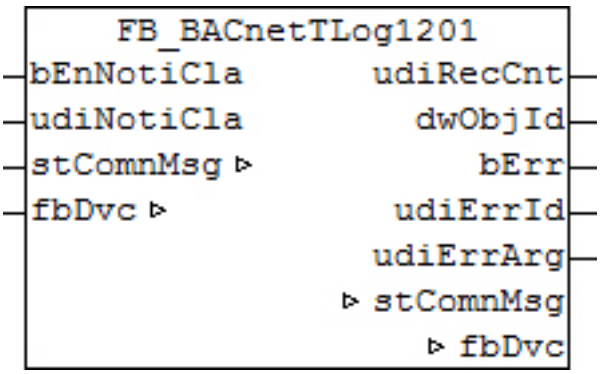
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.46 FB_BACnetTLog1201

BACnet trend log




### Functional description

This function block generates a BACnet trend log object and provides write and read variables for the object within the PLC.

The parameters, which can be written from the PLC, can also be written from the BACnet side; the last change is always the valid one. These parameters are transferred from the PLC via ADS and into the PLC via a cyclic interface. On the PLC side, all parameters have an additional Enable input: when the respective Enable input is set for the first time (rising edge), the value at the parameter input is transferred in any case; subsequently, if Enable is set, it is only transferred if it changes, in order to reduce the ADS traffic. If the Enable input is not set, no transfer takes place.

| PLC variable | Comment | BACnet-Property (Property ID) |
|---|---|---|
| bEnNotiCla | Enable for writing the notification class | |
| udiNotiCla | Property value of the notification class | NotificationClass (17) |

### Inputs/outputs

#### VAR_INPUT

```
bEnNotiCla    : BOOL;
udiNotiCla    : UDINT;
```

**bEnNotiCla / udiNotiCla:** enable/property value notification class

#### VAR_OUTPUT

```
udiRecCnt : UDINT;
dwObjId   : DWORD;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**udiRecCnt:** number of logged values (property ID 141).

**dwObjId:** BACnet object ID of the trend log object.

**bErr:** indicates a general error in the function block. The cause can be in BACnet, in the ADS data exchange, or due to incorrect parameterization.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

#### VAR_IN_OUT

```
stComnMsg : ST_BA_ComnMsg;
fbDvc     : FB_BACnet_Device;
```

**stComnMsg:** reference to the connection structure [▶ 326] for the collective message function block FB_BA_ComMsg [▶ 197].

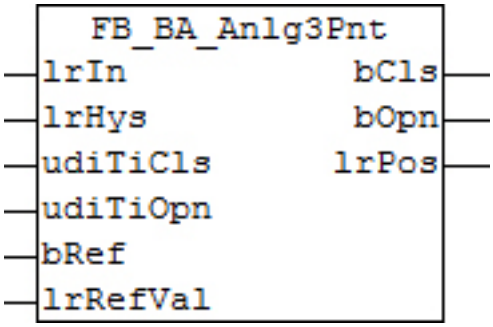
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.47 FB_BA_Anlg3Pnt

The function block is intended for control of three-point actuators for valves or dampers.



### Functional description

The function block converts a constant control signal for positioning of the actuator into binary commands for opening and closing.

If the deviation between the set position value *lrIn* and the calculated actual position value *lrPos* of the actuator exceeds the threshold value set with the variable *lrHys/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 |
|---|---|---|
| lrIn - lrPos > lrHys/2 | TRUE | FALSE |
| lrIn - lrPos < - lrHys/2 | FALSE | TRUE |

The input *lrIn* is automatically limited to the range 0..100% internally.

A rising edge at *bRef* triggers a referencing command (the calculated actual position is set to *lrRefVal*).

If the drive has limit switches, they can be sampled directly via the digital input and used for referencing at *bRef*.

### Inputs/outputs

#### VAR_INPUT

```
lrIn     : LREAL;
lrHys    : LREAL;
udiTiCls : UDINT;
udiTiOpn : UDINT;
bRef     : BOOL;
lrRefVal : LREAL;
```

**lrIn:** setpoint for the actuator position [0…100 %]

**lrHys:** hysteresis for the actuator position [0…100 %]

**udiTiCls:** run time of the actuator from closed to open [ms]

**udiTiOpn:** run time of the actuator from closed to open [ms]

**bRef:** edge references the internal position memory of the drive to value of *lrRefVal* [0…100 %]

**lrRefVal:** value for referencing the actuator with *bRef* [0…100 %]

#### VAR_OUTPUT

```
bCls     : BOOL;
bOpn     : BOOL;
lrPos    : LREAL;
```

**bCls:** output for closing the actuator

**bOpn:** output for opening the actuator

**lrPos:** current calculated actuator position [0…100 %]

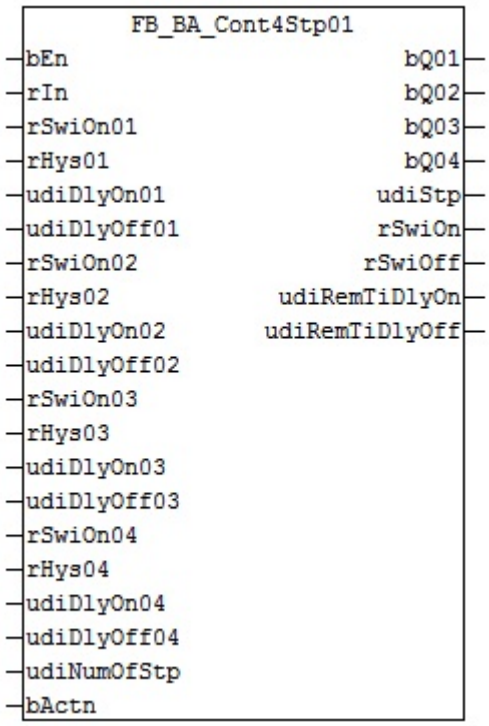
### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.48 FB_BA_Cont4Stp01

Step switch with 4 levels

### Interface

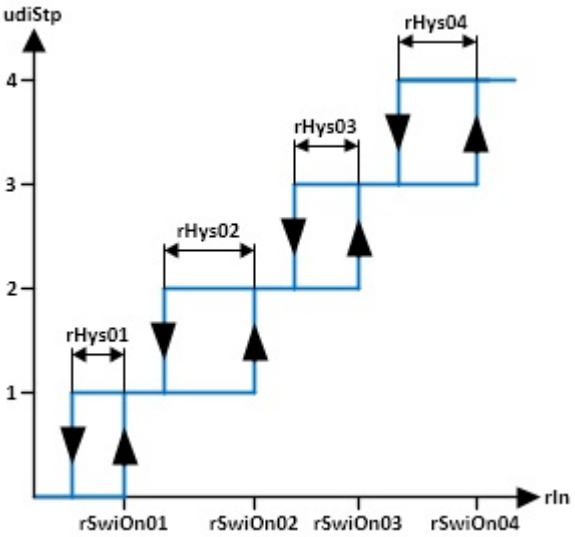



### Functional description

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 parameterised.

### Diagram 01

Direction of action *bActn* = FALSE = reverse = heating

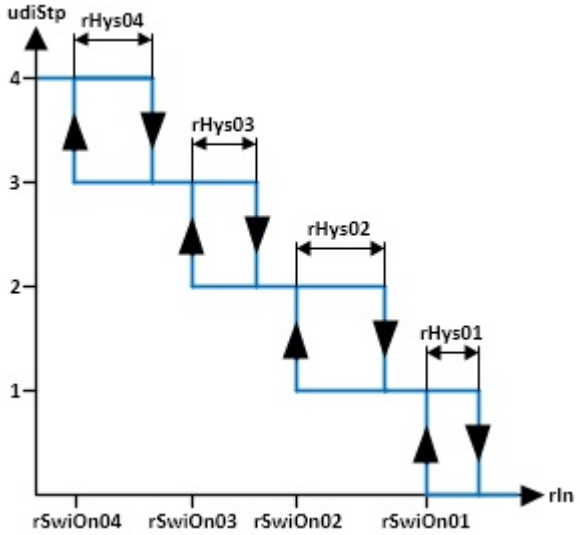



| udiStp | udiNumOfStp | rSwiOn | rSwiOff | udiRemTiDlyOn | udiRemTiDlyOff | bQ01 | bQ02 | bQ03 | bQ04 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | rSwiOn01 | rSwiOn01 - rHys01 | udiDlyOn01 | 0 | FALSE | FALSE | FALSE | FALSE |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | >= 1 | rSwiOn02 | rSwiOn01 - rHys01 | udiDlyOn02 | udiDlyOff01 | TRUE | FALSE | FALSE | FALSE |
| 2 | >= 2 | rSwiOn03 | rSwiOn02 - rHys02 | udiDlyOn03 | udiDlyOff02 | TRUE | TRUE | FALSE | FALSE |
| 3 | >= 3 | rSwiOn04 | rSwiOn03 - rHys03 | udiDlyOn04 | udiDlyOff03 | TRUE | TRUE | TRUE | FALSE |
| 4 | >= 4 | rSwiOn04 | rSwiOn04 - rHys04 | 0 | udiDlyOff04 | TRUE | TRUE | TRUE | TRUE |

**Diagram 02**

Direction of action *bActn* =TRUE = direct = cooling



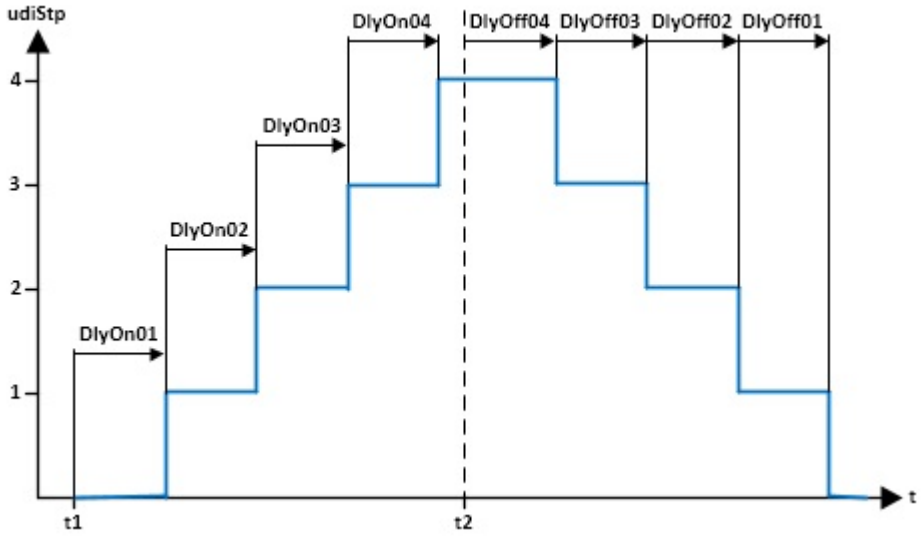

| udiStp | udiNumOfStp | rSwiOn | rSwiOff | udiRemTiDlyOn | udiRemTiDlyOff | bQ01 | bQ02 | bQ03 | bQ04 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | rSwiOn01 | rSwiOn01 + rHys01 | udiDlyOn01 | 0 | FALSE | FALSE | FALSE | FALSE |
| 1 | >= 1 | rSwiOn02 | rSwiOn01 + rHys01 | udiDlyOn02 | udiDlyOff01 | TRUE | FALSE | FALSE | FALSE |
| 2 | >= 2 | rSwiOn03 | rSwiOn02 + rHys02 | udiDlyOn03 | udiDlyOff02 | TRUE | TRUE | FALSE | FALSE |
| 3 | >= 3 | rSwiOn04 | rSwiOn03 + rHys03 | udiDlyOn04 | udiDlyOff03 | TRUE | TRUE | TRUE | FALSE |
| 4 | 4 | rSwiOn04 | rSwiOn04 + rHys04 | 0 | udiDlyOff04 | TRUE | TRUE | TRUE | TRUE |

**Diagram 03**

Timing of the switch-on and switch-off delays

At time t1 *rIn* changes from £*rSwiOn01* to *rSwiOn04*

At time t2 *rIn* changes from *rSwiOn04* to £ rSwiOn01 – *rHys01*




### Inputs/outputs

### VAR_INPUT

```
bEn           : BOOL;
rIn           : REAL;
rSwiOn01      : REAL;
rHys01        : REAL;
udiDlyOn01    : UDINT;
udiDlyOff01   : UDINT;
rSwiOn02      : REAL;
rHys02        : REAL;
udiDlyOn02    : UDINT;
udiDlyOff02   : UDINT;
rSwiOn03      : REAL;
rHys03        : REAL;
udiDlyOn03    : UDINT;
udiDlyOff03   : UDINT;
rSwiOn04      : REAL;
rHys04        : REAL;
udiDlyOn04    : UDINT;
udiDlyOff04   : UDINT;
udiNumOfStp   : UDINT;
bActn         : BOOL;
```

**bEn: G**eneral 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 level 01

**rHys01:** Absolute value hysteresis level 01

**udiDlyOn01:** Switch-on delay level 01

**udiDlyOff01:** Switch-off delay level 01

**rSwiOn02:** Switch-on point level 02

**rHys02:** Absolute value hysteresis level 02

**udiDlyOn02:** Switch-on delay level 02

**udiDlyOff02:** Switch-off delay level 02

**rSwiOn03:** Switch-on point level 03

**rHys03:** Absolute value hysteresis level 03

**udiDlyOn03:** Switch-on delay level 03

**udiDlyOff03:** Switch-off delay level 03

**rSwiOn04:** Switch-on point level 04

**rHys04:** Absolute value hysteresis level 04

**udiDlyOn04:** Switch-on delay level 04

**udiDlyOff04:** Switch-off delay level 04

**udiNumOfStp:** Number levels that are required.
The input is limited to a range from 0 to 4

**bActn:** Input variable used to determine the direction of action 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  : UDINT;
udiRemTiDlyOff : 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:** display of the next switch-on point

**rSwiOff:** display of the next switch-off point

**udiRemTiDlyOn:** if the switch-on point for switching to the next level is met, the progress of the switch-on delay time is displayed here.

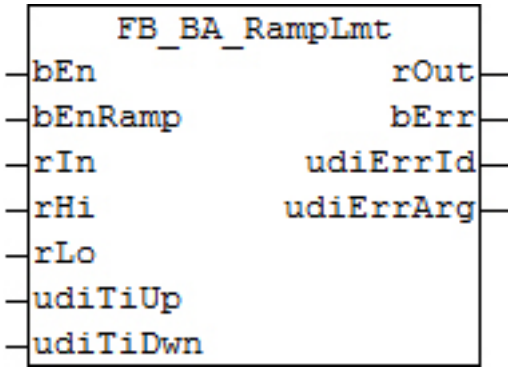
**udiRemTiDlyOff:** if the switch-off point for switching down to the next level is met, the progress of the switch-off delay time is displayed here.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

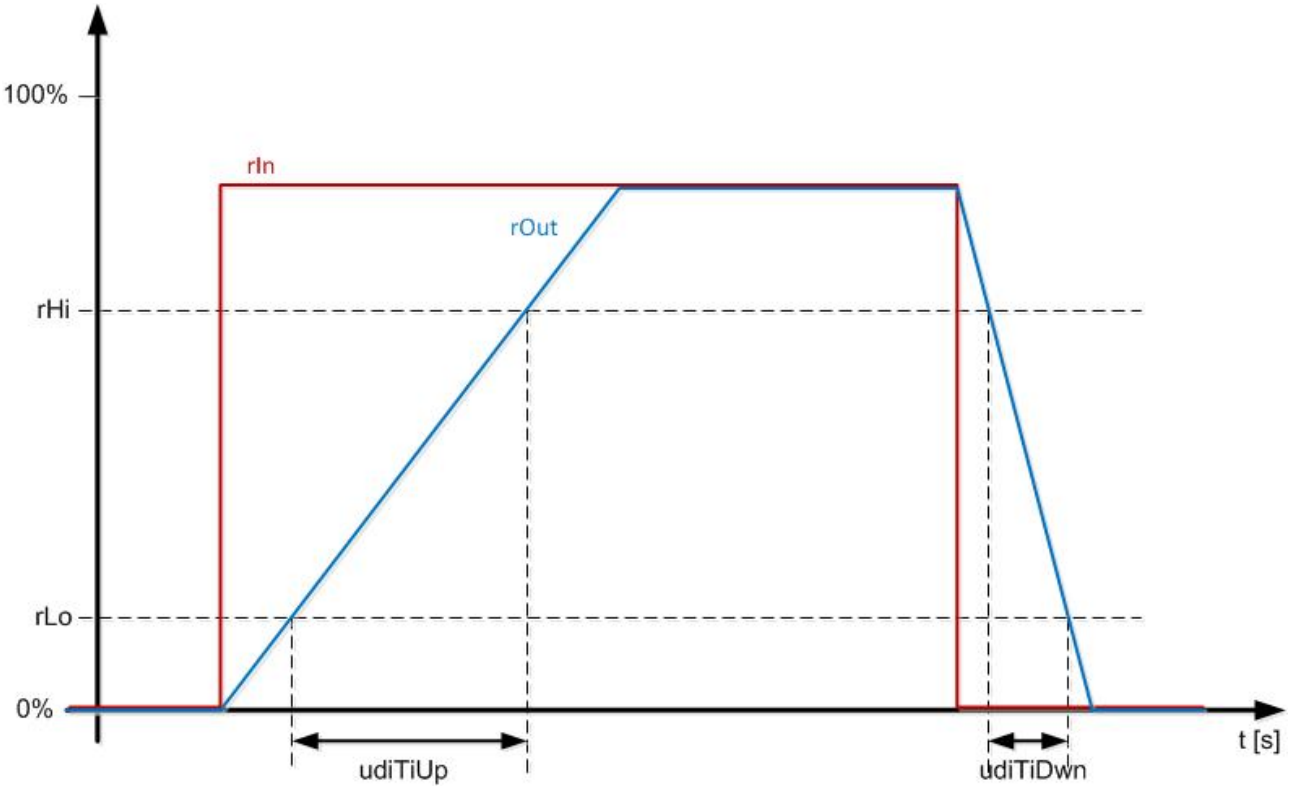
## 8.2.49 FB_BA_RampLmt

Ramp limitation




### Functional description

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*)/*udiTiDwn*.




### Inputs/outputs

#### VAR_INPUT

```
bEn          : BOOL;
bEnRamp      : BOOL;
rIn          : REAL;
rHi          : REAL;
rLo          : REAL;
udiTiUp      : UDINT;
udiTiDwn     : UDINT;
```

**bEn:** enable function block if FALSE, then rOut = 0.0

**bEnRamp:** enable ramp limit, if FALSE, then *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:** rise time [s]

**udiTiDwn:** fall time [s]

### VAR_OUTPUT

```
rOut         : REAL;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**rOut:** Output signal, slope-limited through the ramps

**bErr:** This output is switched to TRUE if the parameters entered are erroneous.

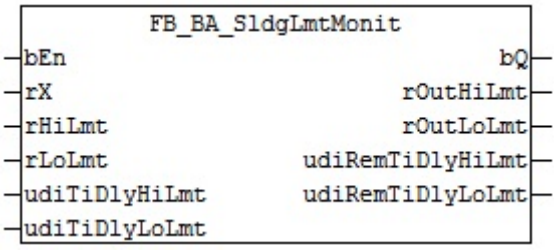
**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.50 FB_BA_SldgLmtMonit

Function block for sliding limit value monitoring



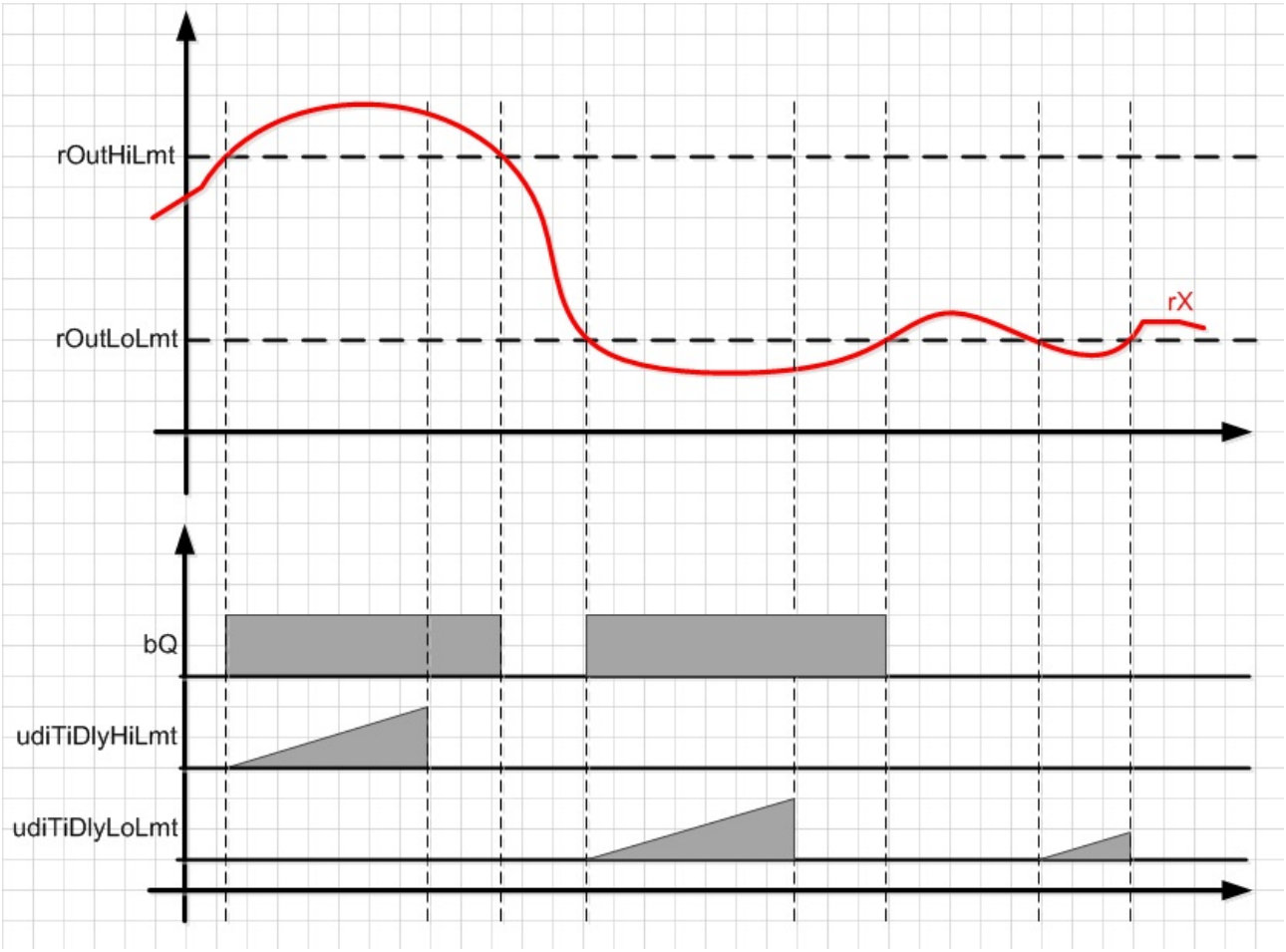

### Functional description

The function block can be used for sliding limit value monitoring. This may be function monitoring of a control valve through comparison of the command output and the position feedback.

**Note that must not be greater than or equal to . If this is the case, = Lmt and = - 0.05.** *rHiLmt rLoLmt rOutHiLmt rHi rOutLoLmt rHiLmt*

### Diagram 1

Diagram 1




**Inputs/outputs**

**VAR_INPUT**

```
bEn             : BOOL;
rX              : REAL;
rHiLmt          : REAL;
rLoLmt          : REAL;
udiTiDlyHiLmt   : UDINT;[s]
udiTiDlyLoLmt   : UDINT;[s]
```

**bEn:** Block enable

**rX:** Actual value

**rHiLmt:** Upper limit value. *rOutHiLmt* = *rHiLmt*

**rLoLmt:** Lower limit value. *rOutLoLmt* = *rLoLmt*

**udiTiDlyHiLmt :** If *rX* is greater than *rOutHiLmt* and the delay *udiTiDlyHiLmt* has elapsed, *bQ* is set, see diagram 1 [▶ 143].

**udiTiDlyLoLmt :** If *rX* is less than *rOutLoLmt* and the delay *udiTiDlyLoLmt* has elapsed, *bQ* is set, see diagram 1 [▶ 143].

**VAR_OUTPUT**

```
bQ              : BOOL;
rOutHiLmt       : REAL;
rOutLoLmt       : REAL;
udiRemTiDlyHiLmt : UDINT;[s]
udiRemTiDlyLoLmt : UDINT;[s]
```

**bQ:** limit value monitoring is enabled. *bQ* can only be reset via *bEn* = FALSE.

**rOutHiLmt:** output upper limit

**rOutLoLmt :** output lower limit

**udiRemTiDlyHiLmt :** delay countdown *udiTiDlyHiLmt*

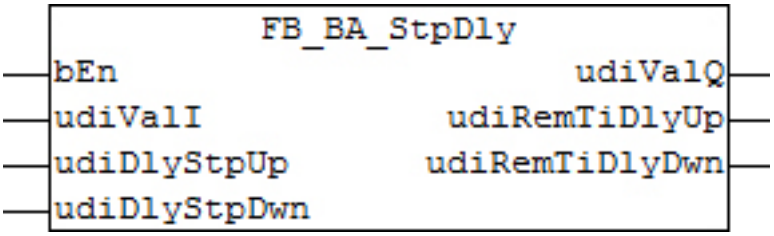
**udiRemTiDlyLoLmt :** delay countdown *udiTiDlyLoLmt*

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.51 FB_BA_StpDly

This function block is used for delayed output of switching stages.




### Functional description

The function is enabled via the input *bEn*. If this is the case, the number present at input *udiValI* is output at *udiValQ* with a delay. If the number at the input is greater than one at the output, the output is delayed by *udiDlyStpUp* [s], if it is smaller, the delay is *udiDlyStpDwn*.

> If the number at the output continues to increase while the timer counts up, the timer is **NOT** restarted. The same applies for a countdown. The respective switching cycle is regarded as complete, if the corresponding countdown has elapsed, or if during the countdown the same number occurs at the input (again) as at the output.

If *bEn* is FALSE, the input value *udiValI* is output without delay to *udiValQ*.

### Inputs/outputs

#### VAR_INPUT

```
bEn          : BOOL;
udiValI      : UDINT;
udiDlyStpUp  : UDINT;
udiDlyStpDwn : UDINT;
```

**bEn:** a TRUE signal at this input activates the function block. The value at input *udiValI* is only output with a delay at *udiValQ*. If *bEn* is FALSE, the input value *udiValI* is output without delay.

**udiValI:** input value

**udiDlyStpUp:** delay time for up-switching [s]

**udiDlyStpDwn:** delay time for down-switching [s]

#### VAR_OUTPUT

```
udiValQ        : UDINT;
udiRemTiDlyUp  : UDINT;
udiRemTiDlyDwn : UDINT;
```

**udiValQ:** output value.

**udiRemTiDlyUp:** countdown to up-switching [s]

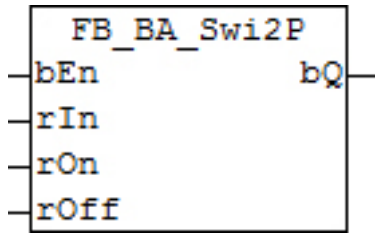
**udiRemTiDlyDwn:** countdown to down-switching [s]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.52 FB_BA_Swi2P

Two-point switch

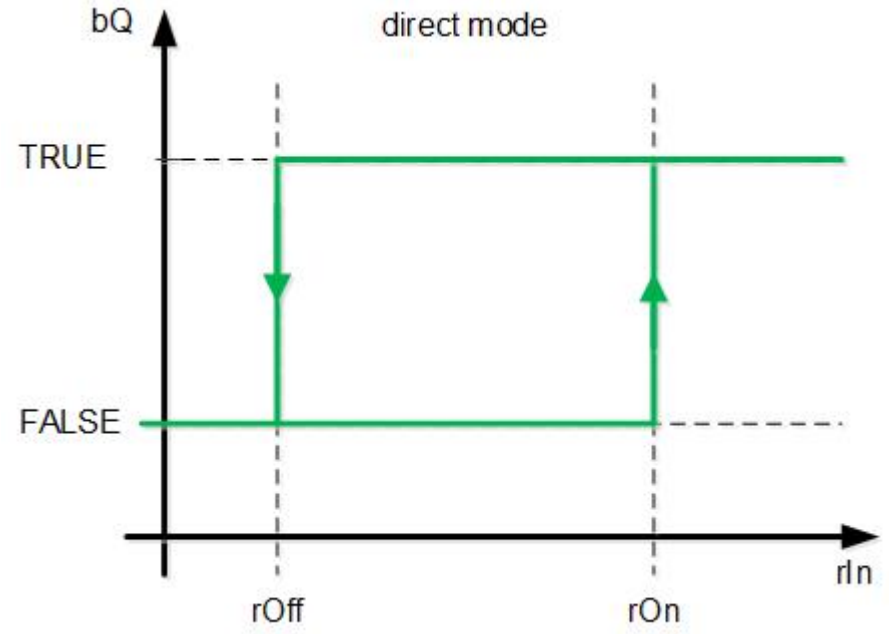



### Functional description

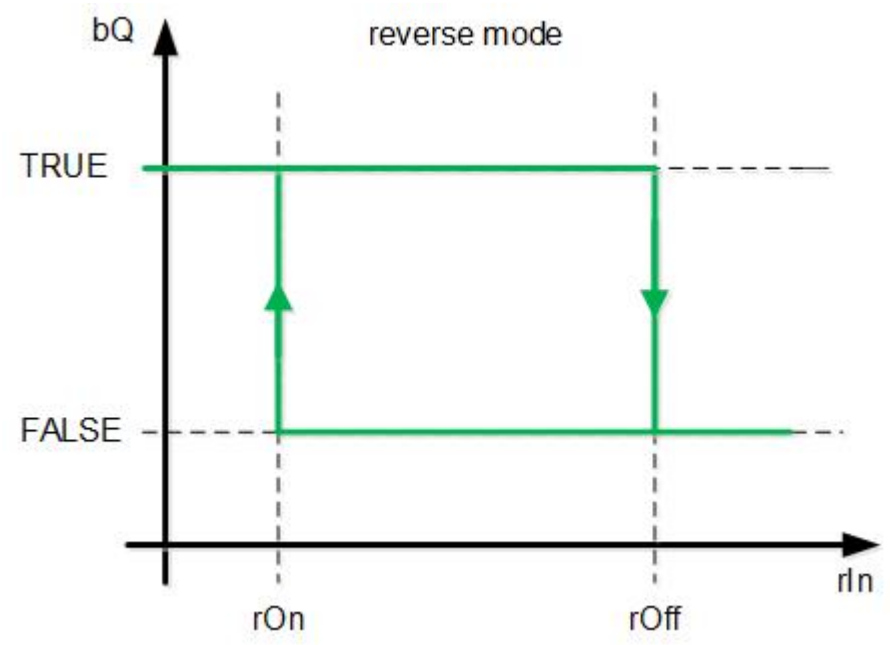
The function block FB_BA_Swi2P is a two-point switch with one switch-on point and one switch-off point.

A general function block enable can be implemented at input *bEn*. The output *bQ* is FALSE as long as *bEn* is FALSE. The direction of action of the 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 direction of action is direct/synchronous (cooling mode).




Is the switch-off point is greater than the switch-on point, the direction of action is direct/reversed (heating mode).




### Inputs/outputs

#### VAR_INPUT

```
bEn     : BOOL;
rIn     : REAL;
rOn     : REAL;
rOff    : REAL;
```

**bEn:** general enable of the function block

**rIn:** input value

**rOn:** switch-on point

**rOff:** switch-off point

#### VAR_OUTPUT

```
bQ      : BOOL;
```

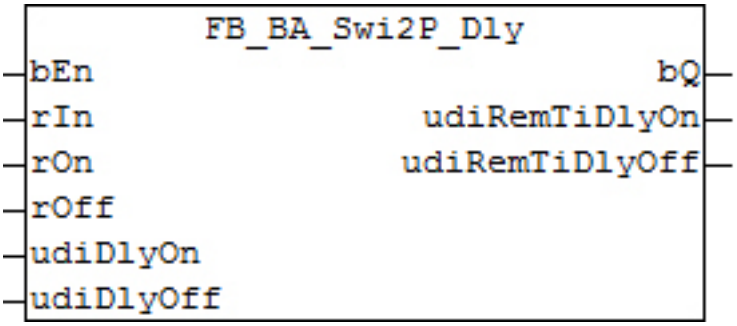
**bQ:** control output

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.53 FB_BA_Swi2P_Dly

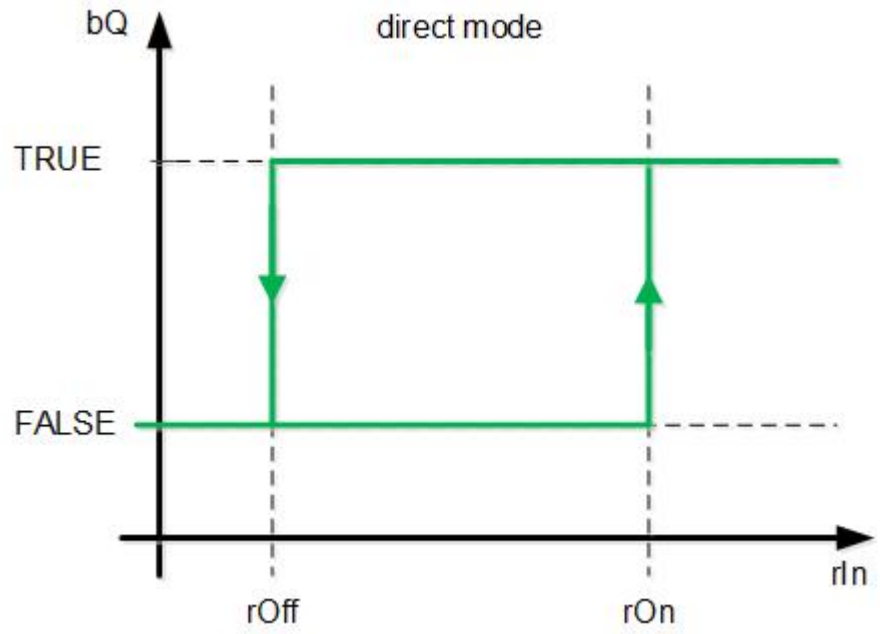
Two-point switch with delay



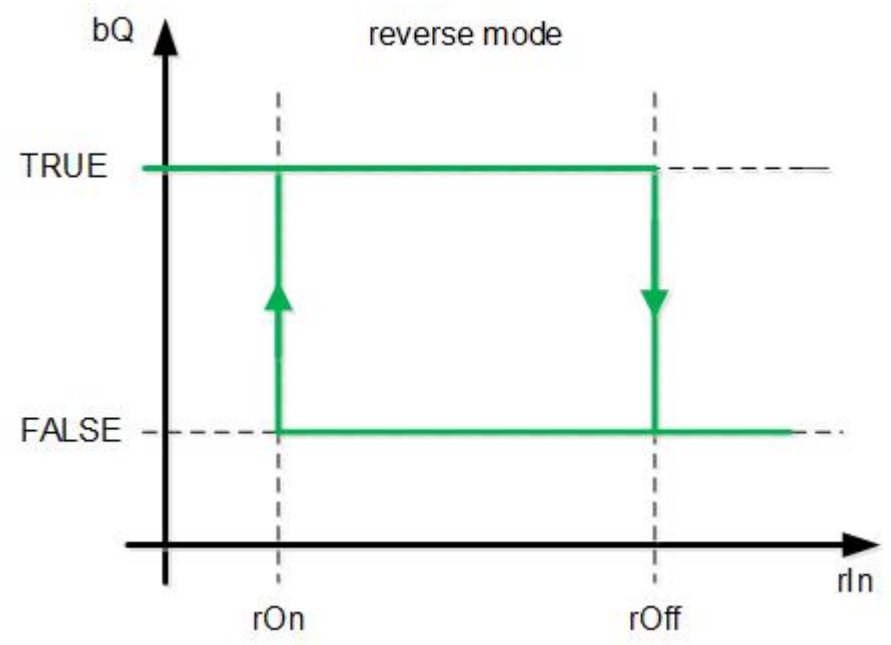

**Functional description**

The function block FB_BA_Swi2P_Dly is a two-point switch with one switch-on point and one switch-off point. In contrast to FB_BA_Swi2P [▶ 146], a change in the output signal is delayed.
A general function block enable can be implemented at input *bEn*. The output *bQ* is FALSE as long as *bEn* is FALSE. The direction of action of the 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 direction of action is direct/synchronous (cooling mode).




Is the switch-off point is greater than the switch-on point, the direction of action is direct/reversed (heating mode).




### Inputs/outputs

### VAR_INPUT

```
bEn        : BOOL;
rIn        : REAL;
rOn        : REAL;
rOff       : REAL;
udiDlyOn   : UDINT;
udiDlyOff  : UDINT;
```

**bEn:** general enable of the function block

**rIn:** input value

**rOn:** switch-on point

**rOff:** switch-off point

**udiDlyOn:** start-up delay [s]

**nDlyOff:** switch-off delay [s]

### VAR_OUTPUT

```
bQ             : BOOL;
udiRemTiDlyOn  : UDINT;
udiRemTiDlyOff : UDINT;
```

**bQ:** control output

**udiRemTiDlyOn:** countdown start-up delay [s]

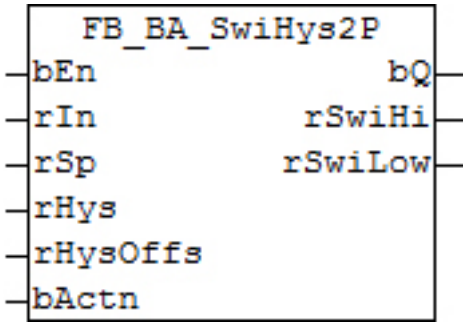
**udiRemTiDlyOff:** countdown switch-off delay [s]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.54 FB_BA_SwiHys2P

Two-point switch around a switching point




### Functional description

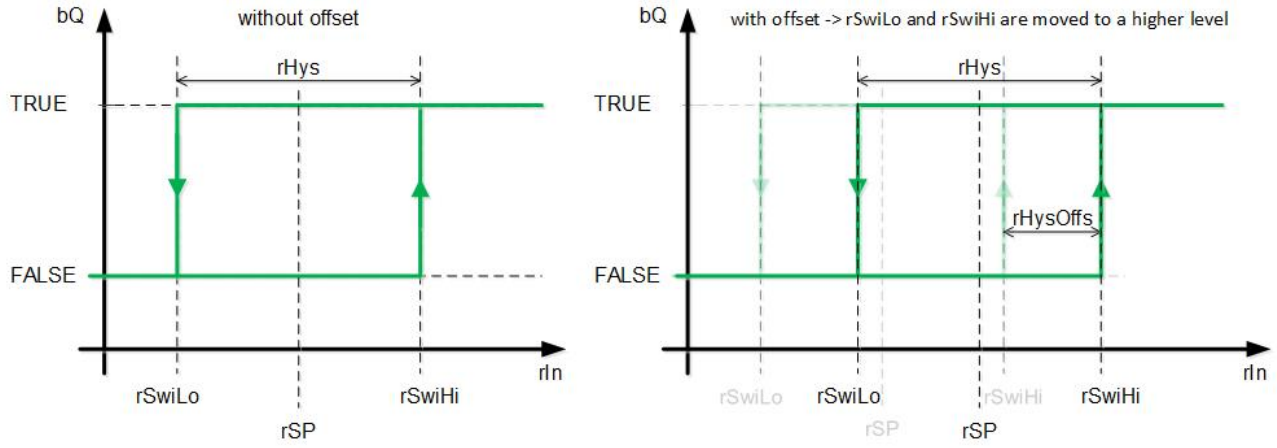
The function block FB_BA_SwiHys2P is a two-point switch with adjustable hysteresis and hysteresis offset.

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 direction of action of the function block depends on the input variable *bActn*.

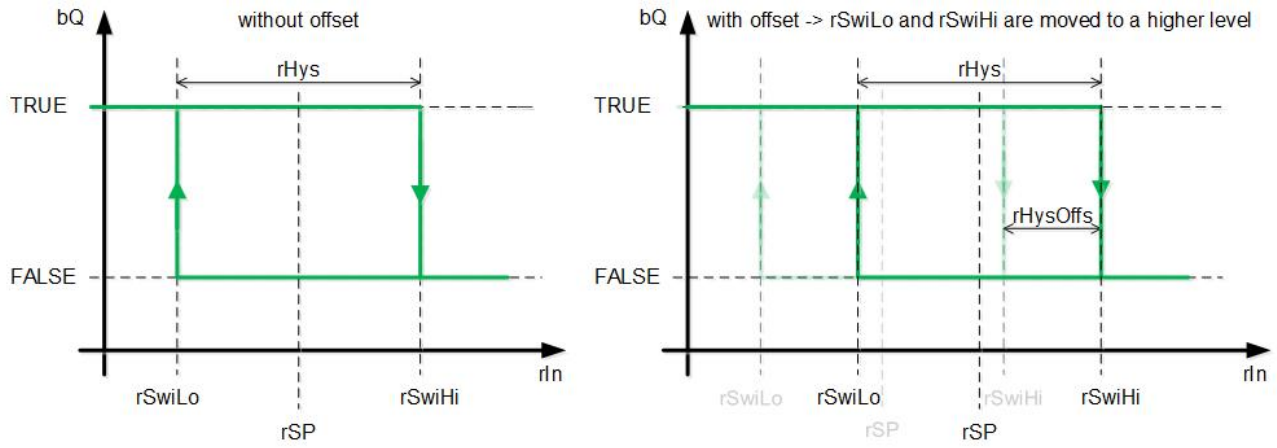
The active switching points result from the setpoint, the hysteresis and the hysteresis offset. They are output at *rSwiHi* and *rSwiLo*.

- 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 direction of action (cooling mode).




If *bActn* is FALSE, the result is indirect/reversed direction of action (heating mode).



### Inputs/outputs

#### VAR_INPUT

```
bEn       : BOOL;
rIn       : REAL;
rSp       : REAL;
rHys      : REAL;
rHysOffs  : REAL;
bActn     : BOOL;
```

**bEn:** general enable of the function block

**rIn:** input value

**rSp:** setpoint input

**rHys:** hysteresis

**rHysOffs:** hysteresis offset

**bActn:** control direction

#### VAR_OUTPUT

```
bQ       : BOOL;
rSwiHi   : REAL;
rSwiLo   : REAL;
```

**bQ:** output

**rSwiHi:** upper switching point

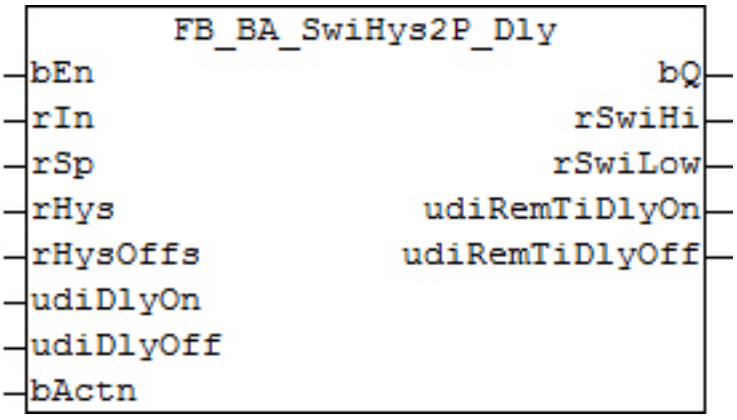
**rSwiLo:** lower switching point

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.55 FB_BA_SwiHys2P_Dly

Two-point switch around a switching point with delay




### Functional description

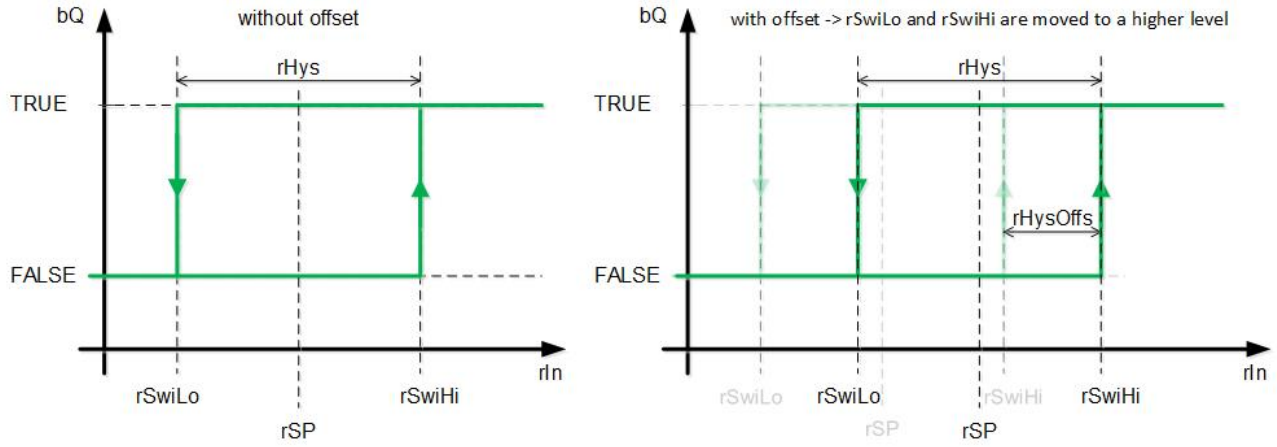
The function block FB_BA_SwiHys2P_Dly is a two-point switch with hysteresis. In contrast to FB_BA_SwiHys2P [▶ 150], a change in the output signal is delayed.

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 direction of action of the function block depends on the input variable *bActn*.

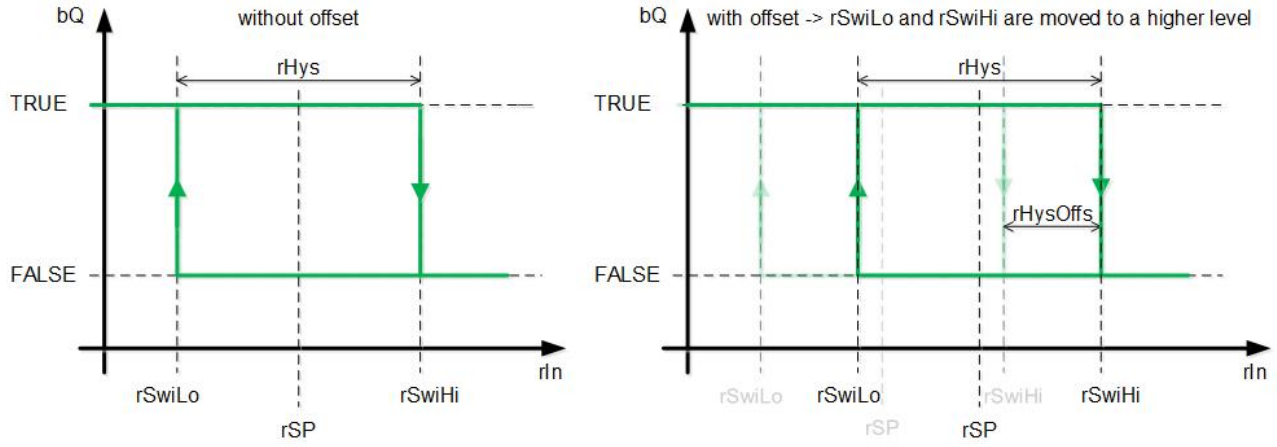
The active switching points result from the setpoint, the hysteresis and the hysteresis offset. They are output at *rSwiHi* and *rSwiLo*.

- 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 direction of action (cooling mode).




If *bActn* is FALSE, the result is indirect/reversed direction of action (heating mode).




### Inputs/outputs

### VAR_INPUT

```
bEn        : BOOL;
rIn        : REAL;
rSp        : REAL;
rHys       : REAL;
rHysOffs   : REAL;
udiDlyOn   : UDINT;
udiDlyOff  : UDINT;
bActn      : BOOL;
```

**bEn:** general enable of the function block

**rIn:** input value

**rSp:** setpoint input

**rHys:** hysteresis

**rHysOffs:** hysteresis offset

**udiDlyOn:** start-up delay [s]

**nDlyOff:** switch-off delay [s]

**bActn:** control direction

### VAR_OUTPUT

```
bQ       : BOOL;
rSwiHi   : REAL;
rSwiLo   : REAL;
```

**bQ:** output

**rSwiHi:** upper switching point

**rSwiLo:** lower switching point

**udiRemTiDlyOn:** countdown start-up delay [s]

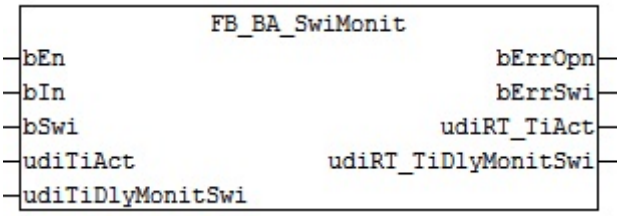
**udiRemTiDlyOff:** countdown switch-off delay [s]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.56 FB_BA_SwiMonit

This function block is used for monitoring limit switches, e.g. for a two-point damper




### Functional description

The function block monitors a limit switch in two steps.
Step 1 includes monitoring while the actuator is moving.
Step 2 includes monitoring of the opened state of the actuator.

The function is activated via the input *bEn*. If *bEn* is FALSE, the outputs *bErrOpn*/*bErrSwi* are also FALSE.

Step 1: If the input *bIn* is TRUE, the move time for the actuator *udiTiAct/udiRT_TiAct* is started. If the input *bSwi* does not become TRUE within this time, a fault is displayed via output *bErrOpn*. If *bSwi* becomes TRUE within the time *udiTiAct*, step 1 is complete, and step 2 becomes active.

Step 2: *bIn* and *bSwi* are TRUE. If *bIn* becomes FALSE, step 1 becomes active again. If *bSwi* becomes FALSE and does not become TRUE again within the time *udiTiDlyMonitSwi*/*udiRT_TiDlyMonitSwi*, a fault is displayed via output *bErrSwi*.

The faults *bErrOpn*/*bErrSwi* are reset, if either *bEn* or *bIn* are FALSE.

### Inputs/outputs

#### VAR_INPUT

```
bEn                : BOOL;
bIn                : BOOL;
bSwi               : BOOL;
udiTiAct           : UDINT;
udiTiDlyMonitSwi   : UDINT;
```

**bEn:** a TRUE signal at this input activates the function block. The value at input *udiVall* is only output with a delay at *udiValQ*. If *bEn* is FALSE, the input value *udiVall* is output without delay.

**bIn:** the actuator control must be applied at this input.

**bSwi:** limit switch actuator

**udiTiAct:** actuator stroke time [s]

**udiTiDlyMonitSwi:** delay time of a limit switch in open state [s]. The time becomes active, if *bIn* is TRUE and the limit switch *bSwi* changes its state from TRUE to FALSE.

#### VAR_OUTPUT

```
bErrOpn              : BOOL;
bErrSwi              : BOOL;
udiRT_TiAct          : UDINT;
udiRT_TiDlyMonitSwi  : UDINT;
```

**bErrOpn:** fault step 1; limit switch has not reached its end position during the procedure

**bErrSwi**: fault step 2; limit switch or actuator has left its end position in opened state

**udiRT_TiAct:** countdown for setting the output bErrOpn [s]. The preset originates from *udiTiAct.*

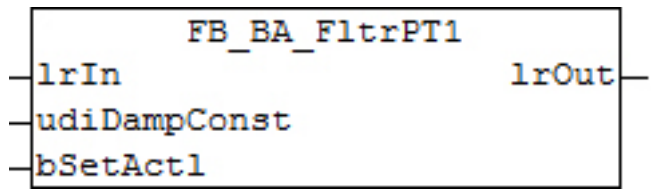
**udiRT_TiDlyMonitSwi:** countdown for setting the output bErrSwi [s]. The preset originates from *udiTiDlyMonitSwi.*

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.57 FB_BA_FltrPT1

First order filter




### Inputs/outputs

#### VAR_INPUT

```
lrIn         : LREAL;
udiDampConst : UDINT;
bSetActl     : BOOL;
```

**lrIn:** input signal

**udiDampConst:** filter time constant [s]

**bSetActI:** a rising edge at this input switches the output value *lrOut* to the input value *lrIn*.

### VAR_OUTPUT

```
lrOut        : LREAL;
```

**lrOut:** filtered output signal

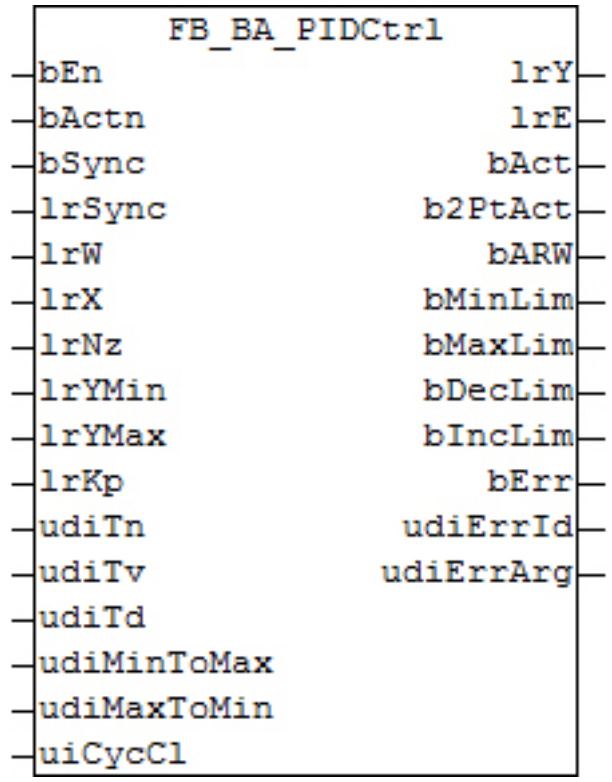
When the function block is first called (system startup), the output *lrOut* is automatically set (once) to the input *lrIn*.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.58 FB_BA_PIDCtrl

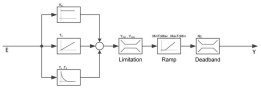
Universal PID controller




### Functional description

This FB is a standard controller with a parallel PID-struchture. If a controller with a selectable structure is needed (either PID in parallel or with a prepositioned P-Part) please use the FB_BA_PIDCtrlEx [▶ 159].

### Diagram



### Passive behaviour (bEn = FALSE or bErr = TRUE)

The outputs are set as follows:

| | |
|---|---|
| lrY | 0.0 |
| lrE | 0.0 |
| bAct | FALSE |
| bARW | FALSE |
| bMinLim | FALSE |
| bMaxLim | FALSE |
| bIncLim | FALSE |
| bDecLim | FALSE |

In the event of an error, *bErr* is TRUE; *udiErrId* and *udiErrArg* describe the error. The internal values for the P-, I-, and D-components are set to 0, also the values for the I- and D-components of the preceding cycle. This ensures that the control value is "clean" in the first cycle after a restart, i.e. it is calculated without historical values.

### Active behaviour (bEn = TRUE and bErr = FALSE)

In the first cycle, the I and D components are calculated "clean", i.e. without historical values, as already mentioned. A positive signal on *bSync* sets the I component so that the control value assumes the value *lrSync*. If *bEn* and *bSync* 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 component is not active, the D component is set accordingly. Note that only the rising edge of *bSync* is evaluated internally as this is a setting action. A TRUE signal must be applied again to the input *bSync* for renewed synchronization, for instance with a transfer value. If the I component is active, the controller ensures that it is retained, if the controller output *lrY* is at the limits *lrYMin* or *lrYMax* and about to fall or increase further. This procedure is referred to as Anti-Reset-Wind-Up. It ensures that the I component is always only just sufficiently large to enable the control value to assume values within the limit immediately after a control deviation, without having to deal with an integral component that has become too large.

### Direction of action

Direction of action

*bActn* = FALSE can be used to reverse the direction of action such that a control deviation of less than 0 results in a change in control value to positive. This is achieved by a negative calculation of the control deviation:

| **bActn** | **rXW (control deviation)** | **Direction of action** |
|---|---|---|
| TRUE | lrX-lrW (actual value-set value) | direct (*cooling)* |
| FALSE | lrW-lrX (set value-actual value) | indirect (heating) |

### Anti-Reset-Windup when the maximum or minimum value is reached

If the controller reaches its upper limit at the output and the control deviation is still positive, the integral component will continue to increase, until the control deviation is less than or equal to zero again. This may lead to an unnecessarily large integral component, which would have to be reduced again, if the sign of the control deviation changes and would make the control behavior sluggish. The same applies if the minimum values is reached at the output, while the control deviation is still negative. To prevent this, the I component is not recalculated when one of these two cases occurs, but it is held at the value of the preceding PLC cycle. If a sign inversion occurs, i.e. if the maximum value is reached and the control deviation is less than 0.0 or the minimum value is reached and the control deviation is greater than 0.0, the I-component is calculated again.

### Slope limitation

If the controller is set faster than the actuator, it is unable to follow the controller, which can lead to jitter. It is therefore possible to limit the slope of the control value.

It is based on the following parameters:

**udiMinToMax:** Slope limitation of controller output for increase: *udiMinToMax* [s] in relation to a change from *lrYMin* to *lrYMax*.

**udiMaxToMin:** Slope limitation of controller output for decrease: *udiMaxToMin* [s] in relation to a change from *lrYMax* to *lrYMin*.

This can be used to calculate the maximum change per PLC cycle (maximum increment or decrement). If the calculated change in the control signal over a PLC cycle is now higher than the change set by *udiMinToMax* or *udiMaxToMin*, the control signal is only increased or decreased by the maximum increment or decrement. Internally, the I-component is automatically adjusted in the same way (I-component of last PLC cycle + maximum increment or I-component of the last PLC cycle - maximum decrement).

### Neutral zone (deadband)

A value of *lrNZ* > 0.0 enables the neutral zone function. If lrNZ is 0 the deadband-part is deactivated and the values at its input are put through.
If the deadband is activated, the value at the input will only be put through as a new ouput-value if the difference between input-value and current output-value is greater or equal to *lrNZ/2*.

**Example**: *lrNZ* = 1, *lrYin* = 55.0, *lrYout*= 55.0 (the input-value was just put through)

| | | |
|---|---|---|
| PLC-Cycle+1 | lrYin=55.2 | lrYout=55.0 |
| PLC-Cycle+2 | lrYin=55.3 | lrYout=55.0 |
| PLC-Cycle+3 | lrYin=55.1 | lrYout=55.0 |
| PLC-Cycle+4 | lrYin=55.6 | lrYout=55.6 |
| PLC-Cycle+5 | lrYin=55.4 | lrYout=55.6 |
| PLC-Cycle+6 | lrYin=55.3 | lrYout=55.6 |
| PLC-Cycle+7 | lrYin=55.1 | lrYout=55.1 |

This function is intended to avoid an unnecessarily large number of actuating pulses.

### On-off control behaviour

**- removed -**

### Autocorrection of parameters

For parameters it makes sense to correct or limit them automatically, without issuing an error message, since the user expectation is clear:

- lrSync > *lrYMax* -> *lrSync* := *lrYMax*
- lrSync < *lrMin* -> *lrSync* := *lrMin*
- uiCycCl = 0 -> *uiCycCl* := 1

### Inputs/outputs

### VAR_INPUT

```
bEn          : BOOL;
bActn        : BOOL;
bSync        : BOOL;
lrSync       : LREAL;
lrW          : LREAL;
lrX          : LREAL;
lrNZ         : LREAL;
lrYMin       : LREAL;
lrYMax       : LREAL;
lrKp         : LREAL;
udiTn        : UDINT;
udiTv        : UDINT;
udiTd        : UDINT;
udiMinToMax  : UDINT;
udiMaxToMin  : UDINT;
uiCycCl      : UINT;
```

**bEn:** controller activation

**bActn:** control direction [▶ 156] of the controller

**bSync / lrSync:** synchronization command: set output value *lrY* to *lrSync*

**lrW:** setpoint

**lrX:** actual value

**lrNZ:** neutral zone

**lrYMin:** lower controller output limit

**lrYMax:** upper limit of the working range of the controller

**lrKp:** controller amplification. Only affects the P component

**udiTn:** integral action time of the I component [ms]. A zero value at this parameter disables the I component.

**udiTv:** integral action time of the D component [ms]. A zero value at this parameter disables the D component.

**udiTd:** damping time of the D component [ms]

**udiMinToMax:** slope limit of controller output for increase: *udiMinToMax* [s] related to a change from *lrYMin* to *lrYMax*.

**udiMaxToMin:** slope limit of controller output for decrease: *udiMaxToMin* [s] related to a change from *lrYMax* to *lrYMin*.

**uiCycCl:** call cycle of the function block as a multiple of the cycle time. A zero entry is automatically interpreted as *uiCycleCall* =1.
Example: *tTaskCycleTime* = 20 ms, *uiCtrlCycleCall* =10 -> The control algorithm is called every 200 ms. Thus the outputs are also updated only every 200 ms.

### VAR_OUTPUT

```
lrY         : LREAL;
lrE         : LREAL;
bAct        : BOOL;
b2PtAct     : BOOL;
bARW        : BOOL;
bMinLim     : BOOL;
bMaxLim     : BOOL;
bDecLim     : BOOL;
bIncLim     : BOOL;
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**lrY:** control value. Range limited by *lrYMin* and *lrYMax*.

**lrE:** control deviation (calculation dependent on control direction [▶ 156])

**bAct:** the controller is active, i.e. enabled (*bEn* = TRUE) and not in the error state (*bErr* = FALSE).

**bARW**: Anti-Reset-Windup function is active

**b2PtAct: is no longer required.**

**bMaxLim:** the control value has reached its upper limit value.

**bMinLim:** the control value has reached its lower limit value.

**bDecLim:** the control value slope has reached its limit value for the maximum decrease, see *udiMaxToMin* (VAR_INPUT).

**bIncLim :** the control value slope has reached its limit value for the maximum increase, see *udiMinToMax* (VAR_INPUT).

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

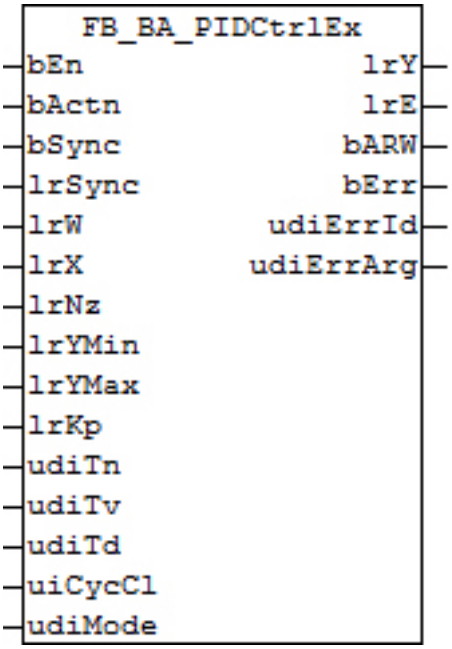
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.59 FB_BA_PIDCtrlEx

Universal PID controller, alternatively in parallel structure or with upstream proportional component

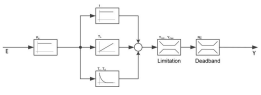



### Functional description

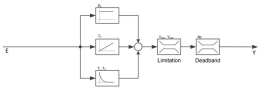
This controller can be used either in parallel PID-structure or with a pre-positioned P-Part. The mode can be selected wit a proper entra at the input *udiMode*.

### Functional diagram

***udiMode*=0 (P-part pre-positioned):**



***udiMode*=1 (parallel-structure):**



### Passive behaviour (bEn = FALSE or bErr = TRUE)

The outputs are set as follows:

| | |
|---|---|
| lrY | 0.0 |
| lrE | 0.0 |
| bAct | FALSE |
| bARW | FALSE |
| bMinLim | FALSE |

| | |
|---|---|
| bMaxLim | FALSE |
| bIncLim | FALSE |
| bDecLim | FALSE |

In the event of an error, *bErr* is TRUE; *udiErrId* and *udiErrArg* describe the error. The internal values for the P-, I-, and D-components are set to 0, also the values for the I- and D-components of the preceding cycle. This ensures that the control value is "clean" in the first cycle after a restart, i.e. it is calculated without historical values.

### Active behaviour (bEn = TRUE and bErr = FALSE)

In the first cycle, the I and D components are calculated "clean", i.e. without historical values, as already mentioned. A positive signal on *bSync* sets the I component so that the control value assumes the value *lrSync*. If *bEn* and *bSync* 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 component is not active, the D component is set accordingly. Note that only the rising edge of *bSync* is evaluated internally as this is a setting action. A TRUE signal must be applied again to the input *bSync* for renewed synchronization, for instance with a transfer value. If the I component is active, the controller ensures that it is retained, if the controller output *lrY* is at the limits *lrYMin* or *lrYMax* and about to fall or increase further. This procedure is referred to as Anti-Reset-Wind-Up. It ensures that the I component is always only just sufficiently large to enable the control value to assume values within the limit immediately after a control deviation, without having to deal with an integral component that has become too large.

### Control direction

If *bActn* = FALSE, the control direction of the controller is reversed so that a control deviation of less than 0 causes a change in the control value in the positive direction. This is achieved by a negative calculation of the control deviation:

| bActn | rXW (control deviation) | Control direction |
|---|---|---|
| TRUE | lrX-lrW (actual value-set value) | direct (cooling) |
| FALSE | lrW-lrX (set value-actual value) | indirect (heating) |

### Anti-Reset-Windup when the maximum or minimum value is reached

If the controller reaches its upper limit at the output and the control deviation is still positive, the integral component will continue to increase, until the control deviation is less than or equal to zero again. This may lead to an unnecessarily large integral component, which would have to be reduced again, if the sign of the control deviation changes and would make the control behavior sluggish. The same applies if the minimum values is reached at the output, while the control deviation is still negative. To prevent this, the I component is not recalculated when one of these two cases occurs, but it is held at the value of the preceding PLC cycle. If a sign inversion occurs, i.e. if the maximum value is reached and the control deviation is less than 0.0 or the minimum value is reached and the control deviation is greater than 0.0, the I-component is calculated again.

### Neutral zone (deadband)

A value of *lrNZ* > 0.0 enables the neutral zone function. If lrNZ is 0 the deadband-part is deactivated and the values at its input are put through.
If the deadband is activated, the value at the input will only be put through as a new ouput-value if the difference between input-value and current output-value is greater or equal to *lrNZ/2*.

**Example**: *lrNZ* = 1, *lrYin* = 55.0, *lrYout*= 55.0 (the input-value was just put through)

| | | |
|---|---|---|
| PLC-Cycle+1 | lrYin=55.2 | lrYout=55.0 |
| PLC-Cycle+2 | lrYin=55.3 | lrYout=55.0 |
| PLC-Cycle+3 | lrYin=55.1 | lrYout=55.0 |
| PLC-Cycle+4 | lrYin=55.6 | lrYout=55.6 |
| PLC-Cycle+5 | lrYin=55.4 | lrYout=55.6 |
| PLC-Cycle+6 | lrYin=55.3 | lrYout=55.6 |
| PLC-Cycle+7 | lrYin=55.1 | lrYout=55.1 |

This function is intended to avoid an unnecessarily large number of actuating pulses.

### Autocorrection of parameters

For parameters it makes sense to correct or limit them automatically, without issuing an error message, since the user expectation is clear:

- lrSync > *lrYMax* -> *lrSync* := *lrYMax*
- lrSync < *lrMin* -> *lrSync* := *lrMin*
- uiCycCl = 0 -> *uiCycCl* := 1

### Inputs/outputs

### VAR_INPUT

```
bEn          : BOOL;
bActn        : BOOL;
bSync        : BOOL;
lrSync       : LREAL;
lrW          : LREAL;
lrX          : LREAL;
lrNZ         : LREAL;
lrKp         : LREAL;
udiTn        : UDINT;
udiTv        : UDINT;
udiTd        : UDINT;
uiCycCl      : UINT;
udiMode      : UDINT;
```

**bEn:** controller activation

**bActn:** control direction [▶ 160] of the controller

**bSync / lrSync:** synchronization command: set output value *lrY* to *lrSync*.

**lrW:** setpoint

**lrX:** actual value

**lrNZ:** neutral zone

**lrYMin:** lower controller output limit

**lrYMax:** upper limit of the working range of the controller

**lrKp:** controller amplification. Only affects the P component

**udiTn:** integral action time of the I component [ms]. A zero value at this parameter disables the I component.

**udiTv:** integral action time of the D component [ms]. A zero value at this parameter disables the D component.

**udiTd:** damping time of the D component [ms] .**uiCycCl:** call cycle of the function block as multiple of the cycle time. A zero entry is automatically interpreted as *uiCycleCall* =1.
Example: *tTaskCycleTime* = 20 ms, *uiCtrlCycleCall* =10 -> The control algorithm is called every 200 ms. Thus the outputs are also updated only every 200 ms.

**udiMode:** *udiMode*=0: controller with upstream P component, *udiMode*=1: controller in parallel structure.

### VAR_OUTPUT

```
lrY          : LREAL;
lrE          : LREAL;
bARW         : BOOL;
bErr         : BOOL;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**lrY:** control value. Range limited by *lrYMin* and *lrYMax*.

**lrE:** control deviation (calculation dependent on control direction [▶ 160])

**bARW:** Anti-Reset-Windup function is active.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

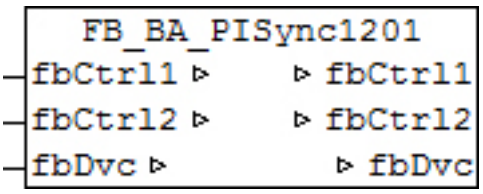
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.60 FB_BA_PISync1201

Parameter synchronization for 2 controllers of type FB_BACnetLoop1201 [▶ 98]. This function block can be used to synchronize the master controllers, for example, which are used to generate the inlet air set values for an air conditioning system with an air cooler and an air heater.




### Functional description

This function block automatically synchronises the P- and I-parameters and the minimum and maximum output values *lrYMin* and *lrYMax* of two controllers of type FB_BACnetLoop1201 [▶ 98]. It is irrelevant where the change takes place: at one of the two controllers in the PLC or in BACnet.

The function block automatically detects a change and transfers it to the respective other controllers via ADS. To this end, both controllers should be referenced via the IN-OUT-variables *fbCtrl1* and *fbCtrl2* at the function block - *fbDvc* is the reference to the BACnet device, under which both function blocks run.

### Inputs/outputs

#### VAR_IN_OUT

```
fbCtrl1     : FB_BACnetLoop1201;
fbCtrl2     : FB_BACnetLoop1201;
fbDvc       : FB_BACnet_Device;
```

**fbCtrl1:** reference to controller no. 1 of the parameter adjustment

**fbCtrl2:** reference to controller no. 2 of the parameter adjustment

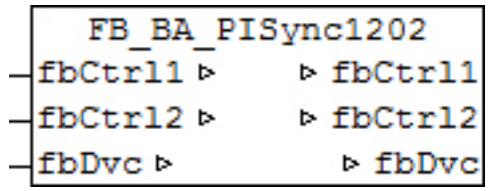
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.61 FB_BA_PISync1202

Parameter synchronisation for 2 controllers of type FB_BACnetLoop1202 [▶ 102]. This function block can be used to synchronise the master controllers, for example, which are used to generate the inlet air set values for an air conditioning system with an air cooler and an air heater.




### Functional description

This function block automatically synchronises the P- and I-parameters and the minimum and maximum output values *IrYMin* and *IrYMax* of two controllers of type FB_BACnetLoop1202 [▶ 102]. It is irrelevant where the change takes place: at one of the two controllers in the PLC or in BACnet.

The function block automatically detects a change and transfers it to the respective other controllers via ADS. To this end, both controllers should be referenced via the IN-OUT-variables *fbCtrl1* and *fbCtrl2* at the function block - *fbDvc* is the reference to the BACnet device, under which both function blocks run.

### Inputs/outputs

#### VAR_IN_OUT

```
fbCtrl1     : FB_BACnetLoop1202;
fbCtrl2     : FB_BACnetLoop1202;
fbDvc       : FB_BACnet_Device;
```

**fbCtrl1:** reference to controller no. 1 of the parameter adjustment

**fbCtrl2:** reference to controller no. 2 of the parameter adjustment

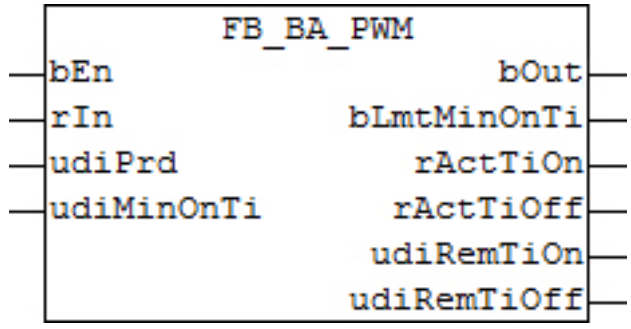
**fbDvc:** reference to the function block of the BACnet device object

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.62 FB_BA_PWM

Pulse width modulation function block



### Functional description

The function block calculates switch-on and switch-off times from an analog input signal *rIn* (0..100%, **internally limited**) and a period *udiPrd* [s], which are displayed at the outputs as *rActTiOn* and *rActTiOff* in seconds.

The following relationships apply:

- 100% at the input of a switch-on time of the total period *udiPrd* and a switch-off time of 0 s.
- 0% at the input corresponds to a switch-on time of 0 s and a switch-off time of the whole period *udiPrd*.

In addition, *udiMinOnTi* [s] can be used to set a lower limit for the duty cycle, in order to prevent damage to drives caused by too short actuating pulses. However, this behavior is only valid for 0>*rIn*>100!
For the input signals:

- *rIn*=0: switch-on time=0, switch-off time=*udiPrd* -> *bOut* remains permanently FALSE
- rIn=100: switch-on time=*udiPrd*, switch-off time=0 -> *bOut* remains permanently TRUE

### 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*/*rActTiOff* [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 switching output *bQ*, it now jumps to the switching step, **unless** input *rIn* is 0.0. It then jumps to the off step (*bQ*=FALSE).
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*/*udiRemTiOff* in full seconds.
4. Completion of the on- or off-step is followed by a jump back to the decision step (point 2).

### Inputs/outputs

#### VAR_INPUT

```
bEn         : BOOL;
rIn         : REAL;
udiPrd      : UDINT;
udiMinOnTi  : UDINT;
```

**bEn :** activation of pulse width modulation

**rIn:** input signal, internally limited to 0...100 %

**udiPrd:** period time [s]

**udiMinOnTi:** minimum switch-on time [s]

#### VAR_OUTPUT

```
bOut         : BOOL;
bLmtSwiTi    : BOOL;
rActTiOn     : REAL;
rActTiOff    : REAL;
udiRemOnTi   : UDINT;
udiRemOffTi  : UDINT;
```

**bOut:** 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:** information output: calculated switch-on time

**rActTiOff:** information output: calculated switch-off time

**udiRemOnTi:** countdown switch-on timer

**udiRemOffTi:** countdown switch-off timer

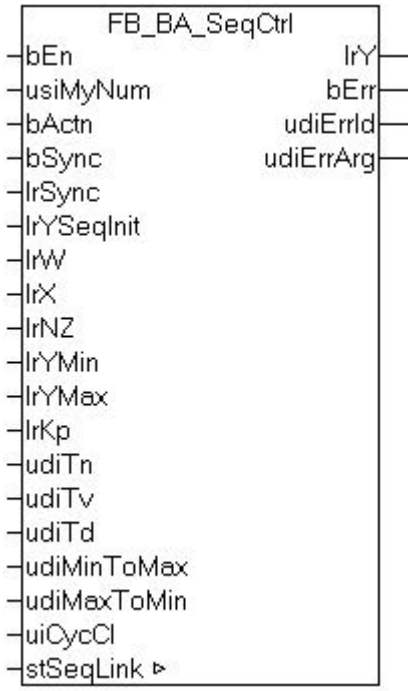
### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.63 FB_BA_SeqCtrl

PID controller as part of a sequence. The functionalities of this controller are identical to FB_BA_PIDCtrl [▶ 155]. However, if it is enabled via *bEn*=TRUE, it is controlled via a higher-level control function block FB_BA_SeqLink [▶ 168].

The data exchange between the control function block FB_BA_SeqLink [▶ 168] and the sequence controllers FB_BA_SeqCtrl takes place via the structure variable stSeqLink [▶ 327].




### Functional description

#### Heating/cooling sequence

The controller sequence should be configured such that the sequence controller with lower ordinal number are used for heating and the ones with the higher number for cooling. Only one change is permitted:

- Sequence controller n (*usiMyNum*=n, *bActn*=TRUE)
- Sequence controller n+1 (*usiMyNum*=n+1, *bActn*=FALSE)

Exclusive programming of cooling and heating controllers is also possible.

Any parameterisation that contradicts this convention is detected and indicated as an error at control function block FB_BA_SeqLink [▶ 168].

### Controller output

While the control function block FB_BA_SeqLink [▶ 168] specifies which sequence controller is active, each sequence controller independently decides its output at control output *lrY*. Since each sequence controller knows the state of the other controllers via the in/out variable stSeqLink [▶ 327], it can distinguish between four cases:

1. The sequence controller detects that it and all other sequence controllers are not enabled, be it due to missing activations (*bEn*) at the input or because an error was detected at the control function block FB_BA_SeqLink [▶ 168]
   -> The sequence controller disables its internal PID controller and issues 0.0 at the control output *lrY*.
2. The sequence controller detects that it is enabled and set as active by the control function block FB_BA_SeqLink [▶ 168]
   -> The sequence controller enables its internal PID controller. Its output signal is output at the control output *lrY*.
3. The sequence controller detects that it is enabled, but a sequence controller with a **higher** ordinal number was set as active by the control function block FB_BA_SeqLink [▶ 168]
   -> The sequence controller disables its internal PID controller. If the sequence controller is in heating mode (*bActn*=FALSE), it will output its minimum value *lrYMin* at the control output *lrY*. If it is in cooling mode (*bActn*=TRUE), it will output its maximum value *lrYMax* at the control output *lrY*.
4. The sequence controller detects that it is enabled, but a sequence controller with a **lower** ordinal number was set as active by the control function block FB_BA_SeqLink [▶ 168]
   -> The sequence controller disables its internal PID controller. If the sequence controller is in heating mode (*bActn*=FALSE), it will output its maximum value *lrYMax* at the control output *lrY*. If it is in cooling mode (*bActn*=TRUE), it will output its minimum value *lrYMin* at the control output *lrY*.

### Synchronisation

If a sequence controller is activated by the higher-level controller, this always results in synchronisation, i.e. the controller starts with a fixed value at the output *lrY*. 3 cases are distinguished:

1. The entire sequence control was switched on via the input *bEn* of the higher-level controller FB_BA_SeqLink [▶ 168]. The controller with the ordinal number *usiStartSeq* at the input of FB_BA_SeqLink [▶ 168] is the start controller.
   -> The sequence controller is synchronised with the value, which is entered at its input *lrYSeqInit*.
2. The sequence controller, which has just has been activated, had a higher ordinal number than the "previous" one
   -> If the sequence controller is in heating mode (*bActn*=FALSE), it is synchronised with its minimum value *lrYMin*. If it is in cooling mode (*bActn*=TRUE), the synchronisation value is its maximum value *lrYMax*.
3. The sequence controller, which has just has been activated, had a lower ordinal number than the "previous" one
   -> If the sequence controller is in heating mode (*bActn*=FALSE), it is synchronised with its maximum value *lrYMax*. If it is in cooling mode (*bActn*=TRUE), the synchronisation value is its minimum value *lrYMin*.

Each sequence controller can also be synchronised by specifying a value *lrSync* and activating *bSync*, if it has just been activated by the higher-level controller. A constant TRUE signal at the input *bSync* (e.g. accidental) is internally intercepted through edge formation, so that obstruction of the synchronisation described above on activation is avoided.

### Start-up behaviour

In order to enable "sensible" adjustment of the entire control sequence, the start controller is maintained in active state as a minimum for the time *udiIniswiOvrDly* [s], which is entered at the function block FB_BA_SeqLink [▶ 168]. During this time, no switching takes place to another controller of this sequence. The output *lrY* of the start controller is synchronised **once** to its value *lrYSeqInit*.

### Inputs/outputs

### VAR_INPUT

```
bEn           : BOOL;
usiMyNum      : USINT;
bActn         : BOOL;
bSync         : BOOL;
lrSync        : LREAL;
lrYSeqInit    : LREAL;
lrW           : LREAL;
lrX           : LREAL;
lrNZ          : LREAL;
lrYMin        : LREAL;
lrYMax        : LREAL;
lrKp          : LREAL;
udiTn         : UDINT;
udiTv         : UDINT;
udiTd         : UDINT;
udiMinToMax   : UDINT;
udiMaxToMin   : UDINT;
uiCycCl       : UINT;
```

**bEn:** activation of the sequence controller

**usiMyNum:** ordinal number of the sequence controller

**bActn:** reversal of control direction of the controller. For heating/cooling operation: *bActn*=FALSE corresponds to heating mode, *bActn*=TRUE corresponds to cooling mode.

**bSync / lrSync:** synchronization command: set output value *lrY* to *lrSync*.

**lrYSeqInit:** start value of the controller after restart of the whole control sequence.

**lrW:** setpoint

**lrX:** actual value

**lrNZ:** neutral zone

**lrYMin:** lower controller output limit [%]. Selectable range: 0..100%.

**lrYMax:** upper limit of the working range of the controller [%]. Selectable range: 0..100%.

**lrKp:** controller amplification. Only affects the P component.

**udiTn:** integral action time of the I component [ms]. A zero value at this parameter disables the I component.

**udiTv:** integral action time of the D component [ms]. A zero value at this parameter disables the D component.

**udiTd:** damping time of the D component [ms]

**udiMinToMax:** slope limit of controller output for increase: *udiMinToMax* [s] related to a change from *lrYMin* to *lrYMax*.

**udiMaxToMin:** slope limit of controller output for decrease: *udiMaxToMin* [s] related to a change from *lrYMax* to *lrYMin*.

**uiCycCl:** call cycle of the function block as a multiple of the cycle time. A zero entry is automatically interpreted as *uiCycleCall* =1.
Example: *tTaskCycleTime* = 20 ms, *uiCtrlCycleCall* =10 -> The control algorithm is called every 200 ms. Thus the outputs are also updated only every 200 ms.

### VAR_IN_OUT

```
stSeqLink : ST_BA_SeqLink;
```

**stSeqLink:** data and command structure between the individual sequence controllers and the function block FB_BA_SeqLink [▶ 168].

If several sequence controllers have the same number (*usiMyNum*), this is detected and output as an error at the sequence controller and at the control function block FB_BA_SeqLink [▶ 168].

### VAR_OUTPUT

```
lrY         : LREAL;
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**lrY:** control value. Range: 0..100%, unless limited further by *lrYMin* and *lrYMax*.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

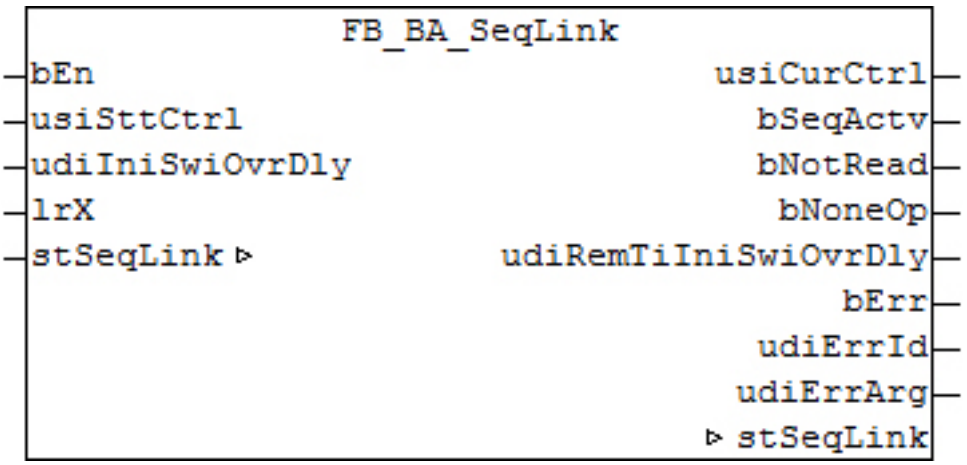
### See also:

ST_BA_SeqLink / ST_BA_SeqLinkData [▶ 327]

## 8.2.64 FB_BA_SeqLink

This function block represents the higher-level control unit, which specifies which sequence controller is currently active.

The data exchange between the control function block FB_BA_SeqLink and the sequence controllers FB_BA_SeqCtrl [▶ 165] takes place via the structure variable stSeqLink [▶ 327].




### Functional description

#### Start-up behavior

A TRUE signal at input *bEn* activates the entire sequence control. The function block will initially activate the sequence controller that is known at *usiStartSeq*. All other sequence controller base their output value on the ranking of the active controller, see FB_BA_SeqCtrl [▶ 165]. The start controller will be set once to its value *lrSyncVal* at the start of the sequence.

In order to enable "sensible" adjustment of the entire control sequence, the start controller is maintained in active state as a minimum for the time *udiIniSwiOvrDly*. During this time, no switching takes place to another controller of this sequence.

### Switching behaviour

Switching behaviour

When the sequence controller reaches its maximum or minimum value, the next controller in the sequence is activated, depending on the controller direction of action, if the actual value is below or above the setpoint of the next controller.
4 cases are distinguished:

- The still active controller has direct direction of action (cooling) and is at its maximum value: The next higher controller in the sequence is then selected, if the actual value exceeds the set value for this controller.
- The still active controller has direct direction of action (cooling) and is at its minimum value: The next lower controller in the sequence is then selected, if the actual value falls below the set value for this controller.
- The still active controller has indirect direction of action (heating) and is at its maximum value: The next lower controller in the sequence is then selected, if the actual value falls below the set value for this controller.
- The still active controller has indirect direction of action (heating) and is at its minimum value: The next higher controller in the sequence is then selected, if the actual value exceeds the set value for this controller.

### Switch-off behavior

If the enable status is removed from a controller within the sequence or if it develops a fault, it is no longer available for the whole 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.
If the enable status is removed from the active controller, a "meaningful" controller must be selected. The sequence link function block uses the following rules:

- The deactivated controller had direct control direction (cooling)

  An operational controller with higher ordinal number is available → switch to the next higher operational controller. Only an operational controller with lower ordinal number is available → switch to the next lower operational controller. No operational controller is available → fault message.
- The deactivated controller had indirect control direction (heating)
- An operational controller with lower ordinal number is available → switch to the next lower operational controller. Only an operational controller with higher ordinal number is available → switch to the next higher operational controller. No operational controller is available → fault message.

### Sequence behavior

If a controller is added to the sequence, it is in any case initially inactive and will output its minimum or maximum value, depending on the direction of action and positioning within the sequence order. The resulting temperature change is compensated by the controller sequence, if possible.

### Error detection

The following errors are detected by the sequence link function block:

1. At least two controller with the same ordinal number exist, which are assigned to the sequence.
2. Within the sequence, more than just a change from indirect to direct control direction (heating to cooling) was detected.
3. The setpoints of the controllers in the controller sequence are not monotonically increasing.
4. The controller, which was defined as start controller via the input *usiStartSeq*, does not exist or is not assigned to this sequence (in this case, the next valid controller is selected).

5. A start controller with the ordinal number 0 or greater than the maximum permitted controller number was specified (in this case, the next valid controller is selected).

**Only the first error causes the sequence link function block to go into error or blocks its processing (=FALSE).** *bSeqActv* None of the associated controllers is then active and all controllers output the control value "0". The function block is not active:
although all other errors are indicated as *bErr*=TRUE, they have an error number (*udiErrId*), they only identify it as a warning.

### Inputs/outputs

### VAR_INPUT

```
bEn             : BOOL;
usiStartCtrl    : USINT;
udiIniSwiOvrDly : UDINT;
lrX             : LREAL
```

**bEn:** activation of the sequence controller

**usiStartSeq:** ordinal number of the sequence controller to be used as start controller for the overall activation.

**udiIniSwiOvrDly:** as mentioned at the beginning, a sequence control needs some time at the beginning to settle with the start controller. The start controller remains active for at least the time *udiIniSwiOvrDly* [s] in the sequence until other criteria cause it to switch to another controller. See also "Start behavior [▶ 168]" and "Switching behavior [▶ 169]".

**lrX:** actual value of the control

### VAR_IN_OUT

```
stSeqLink : ST_BA_SeqLink;
```

**stSeqLink:** data and command structure between the individual sequence controllers and the function block FB_BA_SeqLink. This structure is used by the sequence link function block to receive all relevant sequence controller data and at the same time to notify the controllers which is the active one.

If several sequence controllers have the same number (*usiMyNum*), this is detected and output as an error at the sequence controller and at the control function block.

### VAR_OUTPUT

```
usiCurCtrl            : USINT;
bSeqActv              : BOOL;
bNotRead              : BOOL;
bNoneOp               : BOOL;
udiRemTiIniSwiOvrDly  : UDINT;
bErr                  : BOOL;
udiErrId              : UDINT;
udiErrArg             : UDINT;
```

**usiCurCtrl:** ordinal number of the currently active sequence controller. If no controller is active, 0 is output here.

**bSeqActv:** the sequence function block is enabled (*bEn*) and has no error resulting in switch-off, see error detection [▶ 169].

**bNotRead:** each sequence controller transfers data to the control function block via the structure *stSeqLink*. This output is TRUE, as long as no data were transmitted - this is the case when the PLC is switched on.

**bNoneOp:** this output is switched to TRUE, if none of the sequence controller is enabled (*bEn*=TRUE).

**bErr:** this output is switched to TRUE if the parameters entered are erroneous. This function block may not suspend its execution in the event of an error, see error detection [▶ 169].

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

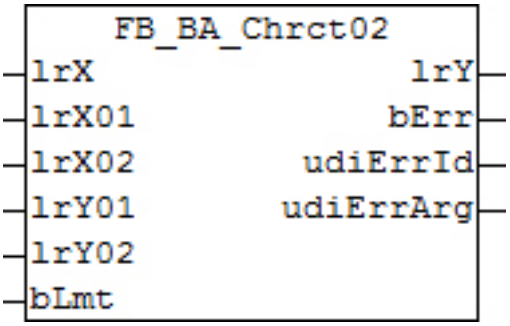
| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

**See also:**

ST_BA_SeqLink / ST_BA_SeqLinkData [▶ 327]

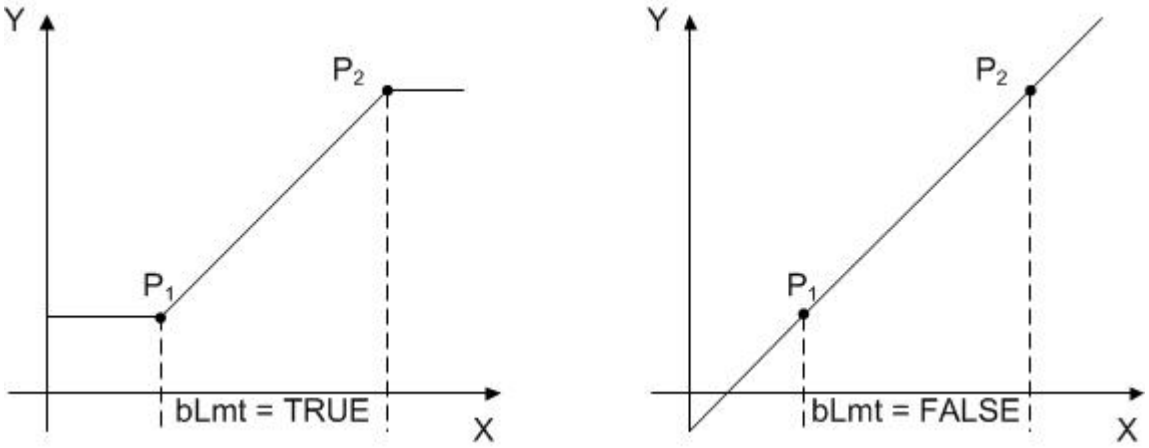
## 8.2.65 FB_BA_Chrct02

Linear interpolation with 2 interpolation points




### Functional description

The function block FB_BA_Chrct02 represents a linear interpolation with two interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [*lrX1/lrY1*] and [*lrX2/lrY2*]. If the input variable *bLmt* is TRUE, *lrY* is limited by *lrY01* and *lrY02*. If *bLmt* is FALSE, *lrY* is not limited.




### Error handling

The input values for *lrX[n+1]* must always be at least 0.0000001 greater than the values for *lrX[n]*.
In the event of an error the variable *udiErrId* indicates at which point of the characteristic curve the values are not monotonically increasing.

### Inputs/outputs

#### VAR_INPUT

```
lrX   : LREAL;
lrX01 : LREAL;
lrX02 : LREAL;
```

```
lrY01 : LREAL;
lrY02 : LREAL;
bLmt  : BOOL;
```

**lrX:** input value of the characteristic curve.

**lrX01:** X-value for interpolation point P1.
**lrX02:** X-value for interpolation point P2.

**lrY01:** Y-value for interpolation point P1.
**lrY02:** Y-value for interpolation point P2.

**bLmt:** limitation of the output value *lrY.*

### VAR_OUTPUT

```
lrY       : LREAL;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**lrY:** calculated output value of the characteristic curve

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

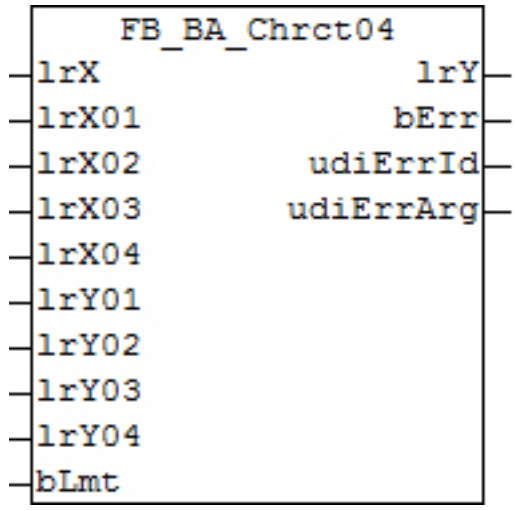
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

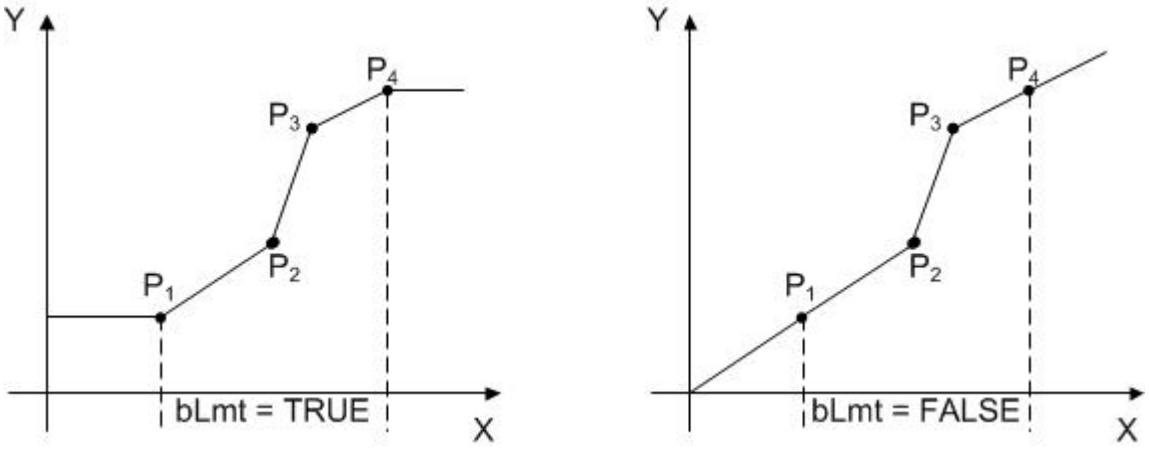
## 8.2.66 FB_BA_Chrct04

Linear interpolation with 4 interpolation points




### Functional description

The function block FB_BA_Chrct04 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 [*lrX1/lrY1*] to [*lrX4/lrY4*]. If the input variable *bLmt* is TRUE, *lrY* is limited by *lrY01* and *lrY04*. If *bLmt* is FALSE, *lrY* is not limited.




### Error handling

The input values for *lrX[n+1]* must always be at least 0.0000001 greater than the values for *lrX[n]*.
In the event of an error the variable *udiErrId* indicates at which point of the characteristic curve the values are not monotonically increasing.

### VAR_INPUT

```
lrX   : LREAL;
lrX01 : LREAL;
lrX02 : LREAL;
lrX03 : LREAL;
lrX04 : LREAL;
lrY01 : LREAL;
lrY02 : LREAL;
lrY03 : LREAL;
lrY04 : LREAL;
bLmt  : BOOL;
```

**lrX:** input value of the characteristic curve

**lrX01:** X-value for interpolation point P1
**lrX02:** X-value for interpolation point P2
**lrX03:** X-value for interpolation point P3
**lrX04:** X-value for interpolation point P4

**lrY01:** Y-value for interpolation point P1
**lrY02:** Y-value for interpolation point P2
**lrY03:** Y-value for interpolation point P3
**lrY04:** Y-value for interpolation point P4

**bLmt:** limitation of the output value *lrY*

### Inputs/outputs

### VAR_OUTPUT

```
lrY       : LREAL;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**lrY:** calculated output value of the characteristic curve

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

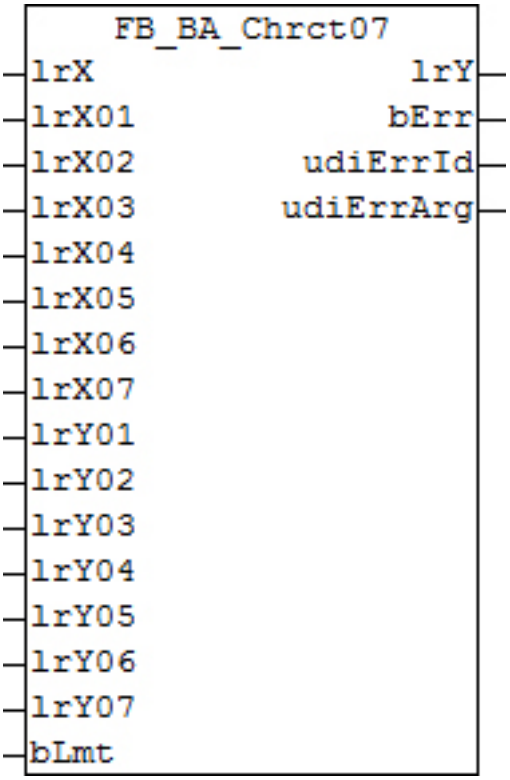
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

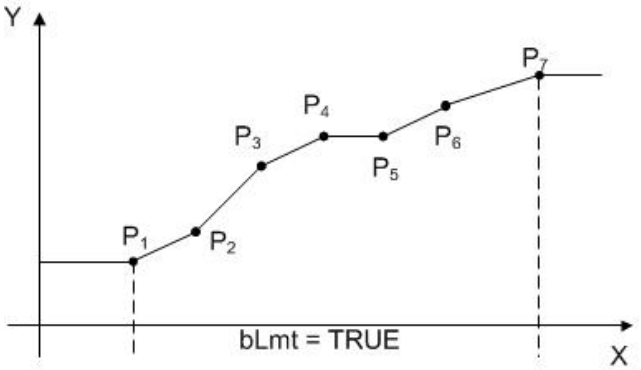
## 8.2.67 FB_BA_Chrct07

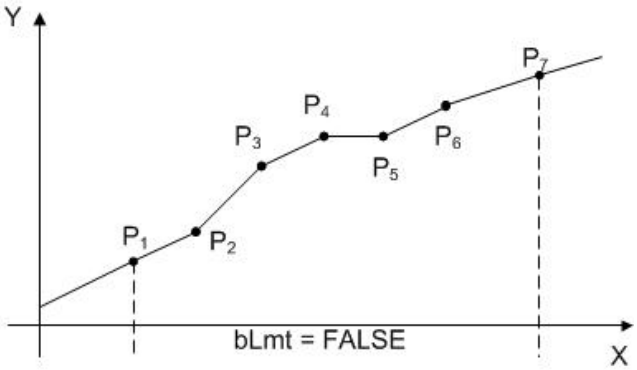
Linear interpolation with 7 interpolation points




### Functional description

The function block FB_BA_Chrct07 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 [*lrX1/lrY1*] to [*lrX7/lrY7*]. If the input variable *bLmt* is TRUE, *lrY* is limited by *lrY01* and *lrY07*. If *bLmt* is FALSE, *lrY* is not limited.



### Error handling

The input values for *lrX[n+1]* must always be at least 0.0000001 greater than the values for *lrX[n]*.
In the event of an error the variable *udiErrId* indicates at which point of the characteristic curve the values are not monotonically increasing.

### Inputs/outputs

### VAR_INPUT

```
lrX   : LREAL;
lrX01 : LREAL;
lrX02 : LREAL;
lrX03 : LREAL;
lrX04 : LREAL;
lrX05 : LREAL;
lrX06 : LREAL;
lrX07 : LREAL;
lrY01 : LREAL;
lrY02 : LREAL;
lrY03 : LREAL;
lrY04 : LREAL;
lrY05 : LREAL;
lrY06 : LREAL;
lrY07 : LREAL;
bLmt  : BOOL;
```

**lrX:** input value of the characteristic curve

**lrX01:** X-value for interpolation point P1
**lrX02:** X-value for interpolation point P2
**lrX03:** X-value for interpolation point P3
**lrX04:** X-value for interpolation point P4
**lrX05:** X-value for interpolation point P5
**lrX06:** X-value for interpolation point P6
**lrX07:** X-value for interpolation point P7

**lrY01:** Y-value for interpolation point P1
**lrY02:** Y-value for interpolation point P2
**lrY03:** Y-value for interpolation point P3
**lrY04:** Y-value for interpolation point P4
**lrY05:** Y-value for interpolation point P5
**lrY06:** Y-value for interpolation point P6
**lrY07:** Y-value for interpolation point P7

**bLmt:** limitation of the output value *lrY*

### VAR_OUTPUT

```
lrY       : LREAL;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**lrY:** calculated output value of the characteristic curve

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

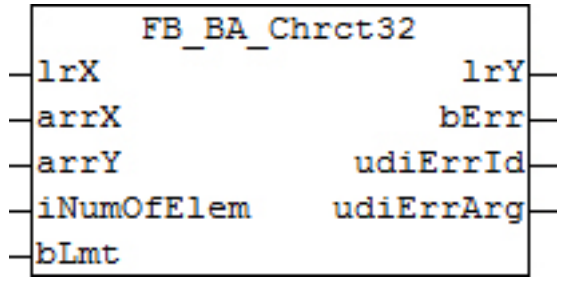
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

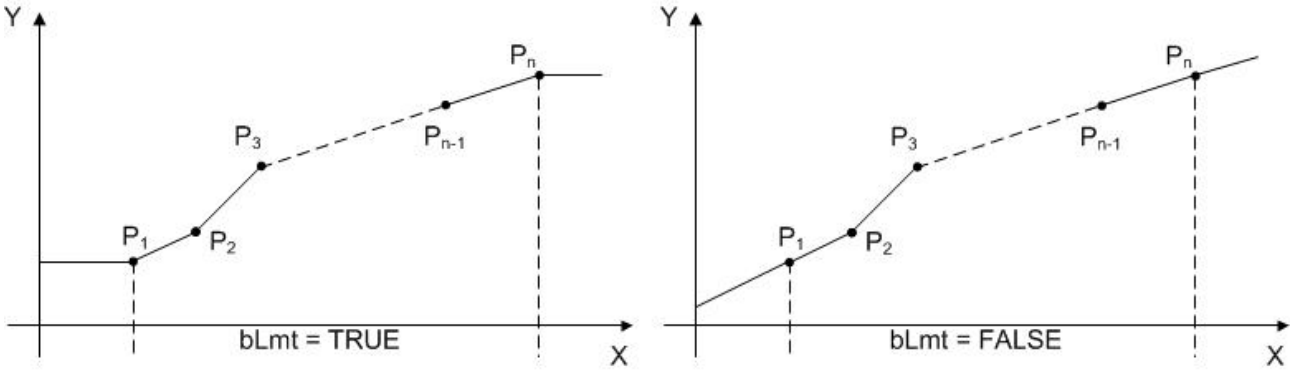
## 8.2.68 FB_BA_Chrct32

Linear interpolation with up to 32 interpolation points (parameterizable)




### Functional description

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 [▶ 171], FB_BA_Chrct04 [▶ 172] and FB_BA_Chrct07 [▶ 174], 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, *lrY* is limited by *arrY[1]* and *arrY[n]*. If *bLmt* is FALSE, *lrY* is not limited.




### Error handling

The input values for *lrX[n+1]* must always be at least 0.0000001 greater than the values for *lrX[n]*.
In the event of an error the variable *udiErrId* indicates at which point of the characteristic curve the values are not monotonically increasing.

The parameter for the number of interpolation points, *iNumOfElem*, must be in the range 2..32.

### Inputs/outputs

### VAR_INPUT

```
lrX         : LREAL;
arrX        : ARRAY [1..gBA_cChrct32_NumOfElem] OF LREAL;
arrY        : ARRAY [1..gBA_cChrct32_NumOfElem] OF LREAL;
iNumOfElem  : INT;
bLmt        : BOOL;
```

**lrX:** input value of the characteristic curve

**arrX :** field with the X-values for the interpolation points

**iNumOfElem:** number of interpolation points; this value must be in the range 2-32, otherwise an error is issued.

**arrY :** field with the Y-values for the interpolation points

**bLmt:** limitation of the output value *lrY*

### VAR_OUTPUT

```
lrY       : LREAL;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**lrY:** calculated output value of the characteristic curve

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

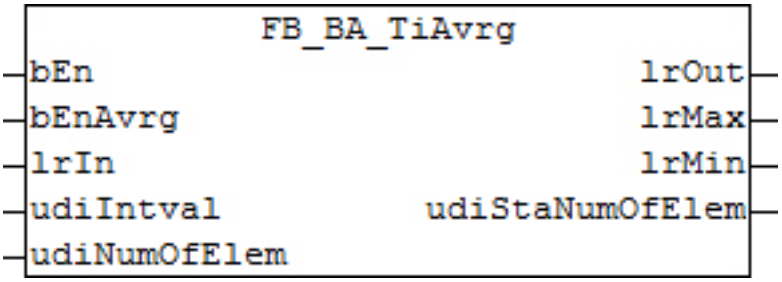
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.69 FB_BA_TiAvrg

Averaging over a time interval




### Functional description

The function block FB_BA_TiAvrg calculates the arithmetic average of an analog value that was logged over a certain period. Discrete values are written into a FIFO buffer. *udiIntval* specifies the time interval [s] over which the values are logged and written into the FIFO. Values are only written if the input *bEnAvrg* is TRUE. The variable *udiNumOfElem* is used to determine the size of the FIFO buffer. It is limited to 1…256. If a value outside this limits is entered, the number is automatically set to a default value of 48. The actual number is output at the output *udiStaNumOfElem* for verification purposes.
The function block can be used for calculating an hourly mean outside temperature over a day, for example. In that case *udiNumOfElem* = 24 and *udiIntval* = 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.

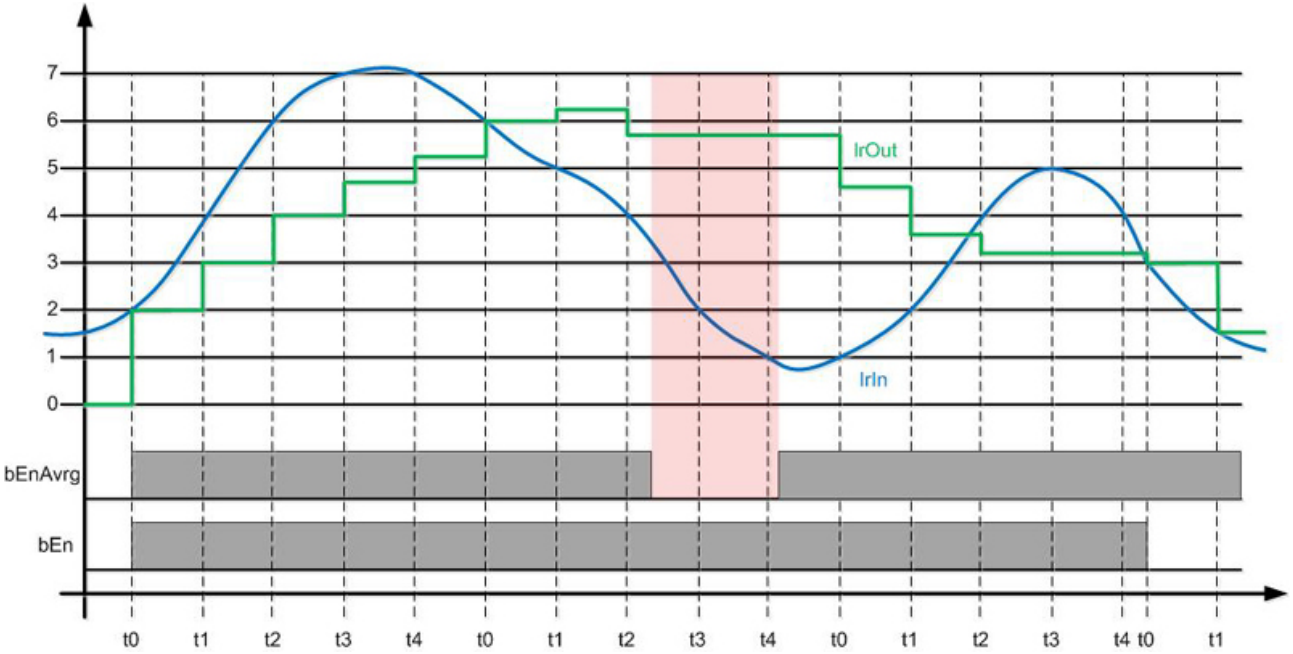
### Example:

**udiNumOfElem = 5**

The area highlighted in pink indicates the phase, in which *bEnAvrg* is set to FALSE. No values are written into the FIFO, and the output value *lrOut* is maintained.

| | First cycle | | Second cycle | | Third cycle | | Fourth cycle | |
|---|---|---|---|---|---|---|---|---|
| | **lrIn** | **lrOut** | **lrIn** | **lrOut** | **lrIn** | **lrOut** | **lrIn** | **lrOut** |
| **t0** | 2 | 2/1 = 2 | 6 | (4+6+7+7+6)/5 = 6 | 1 | (7+6+5+4+1)/5 = 4.6 | 3 | lrIn = 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 | lrIn = 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 | | |

| | First cycle | | Second cycle | | Third cycle | | Fourth cycle | |
|---|---|---|---|---|---|---|---|---|
| **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 | | |




### Inputs/outputs

### VAR_INPUT

```
bEn          : BOOL;
bEnAvrg      : BOOL;
lrIn         : LREAL;
udiIntval    : UDINT;
udiNumOfElem : UDINT;
```

**bEn:** enable of the function block

**bEnAvrg:** enable averaging, *bEnAvrg* = TRUE activates logging, if *bEnAvrg* is FALSE, the calculated value remains at the output.

**lrIn:** input value for averaging

**udiIntVal:** time interval [s] for writing new values into the FIFO

**udiNumOfElem:** size of the FIFO buffer. A change resets the previous averaging. The number of values is automatically limited to 1..256.

### VAR_OUTPUT

```
lrOut           : LREAL;
lrMax           : LREAL;
lrMin           : LREAL;
udiStaNumOfElem : UDINT;
```

**lrOut:** average value

**lrMax:** largest value in the FIFO buffer

**lrMin:** smallest value in the FIFO buffer

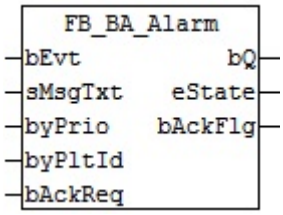
**udiStaNumOfElem:** number of entries in memory

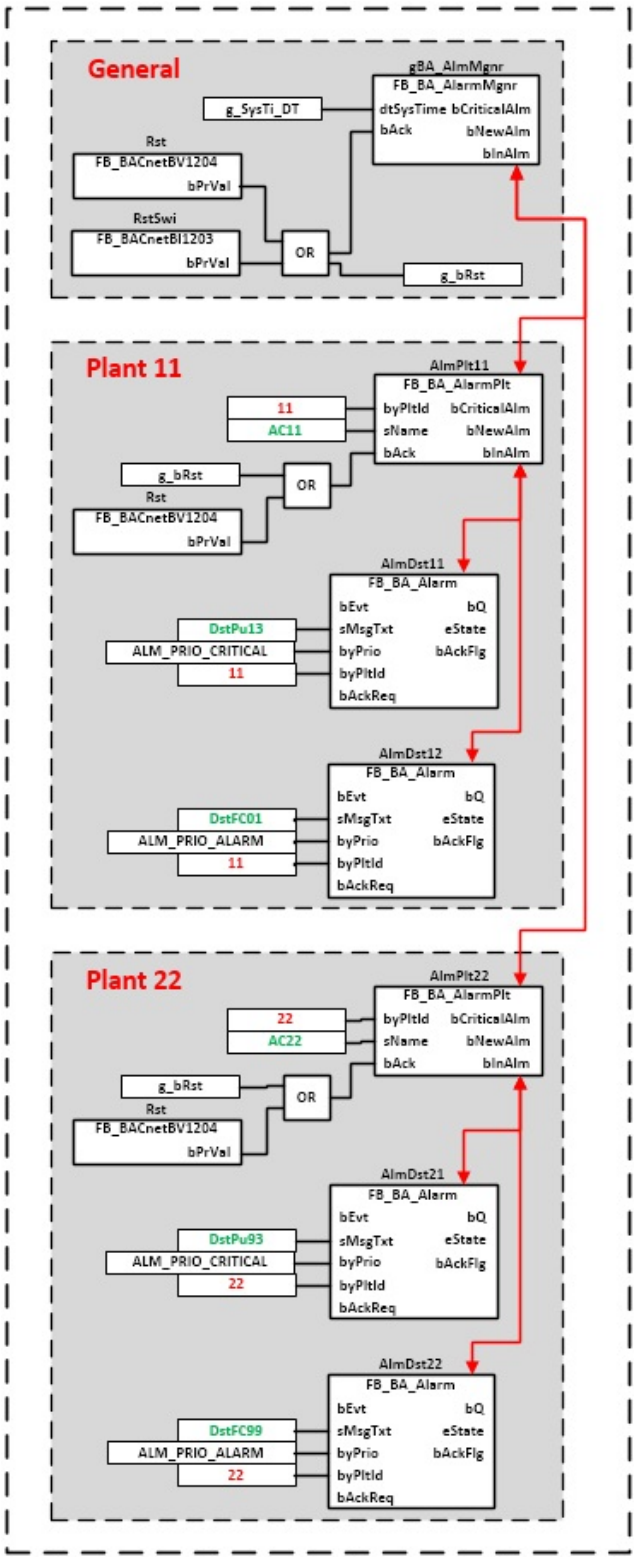
**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.70 FB_BA_Alarm

Alarm output function block with selectable alarm memory and internal acknowledgement.
The acknowledgement pulse is generated in FB_BA_AlarmPlt [▶ 183]. It is internally passed to the collective alarm function blocks of a plant.






### Inputs/outputs

### VAR_INPUT

```
bEvt          : BOOL;
sMsgTxt       : STRING;
byPrio        : BYTE;
byPltId       : BYTE;
bAckReq       : BOOL;
```

**bEvt:** input for detecting the alarm

**sMsgTxt:** input field for message text (configurable via Project Builder or Excel import)

**byPrio:** defines the action that is to be triggered when the alarm occurs in relation to the control of the respective plant.
*byPrio* = 0 = no message = ALM_PRIO_EMPTY
*byPrio* = 1 = various messages = ALM_PRIO_NOTE
*byPrio* = 2 = warning = ALM_PRIO_WARNING
*byPrio* = 3 = alarm without shutdown = ALM_PRIO_ALARM
*byPrio* = 4 = alarm with shutdown = ALM_PRIO_CRITICAL , sets the output bCriticalAlm of the function block FB_BA_AlarmPlt [▶ 183].

**byPltId:** number of the corresponding plant. **Important: All templates of plant must be assigned to the same plant number in the PLC!**

**bAckReq:**TRUE = requires acknowledgement. Is the alarm requires acknowledgement, it is not removed from the alarm list until the plant operator acknowledges the alarm.

### VAR_OUTPUT

```
bQ        : BOOL;
eState    : E_BA_AlmSta;
bAckFlg   : BOOL;
```

**bQ:** Alarm output

**eState:** Enumerator [▶ 324] status of the alarm messages

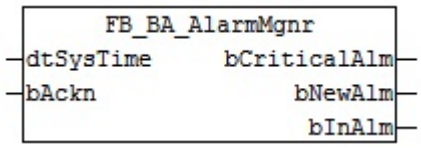
**bAckFlg:** When an acknowledgement is triggered, this output is active for one cycle. The pulse can be used to acknowledge faults. The acknowledgement pulse is generated in FB_BA_AlarmPlt [▶ 183]. It is internally passed to the collective alarm function blocks of a plant.

### Requirements

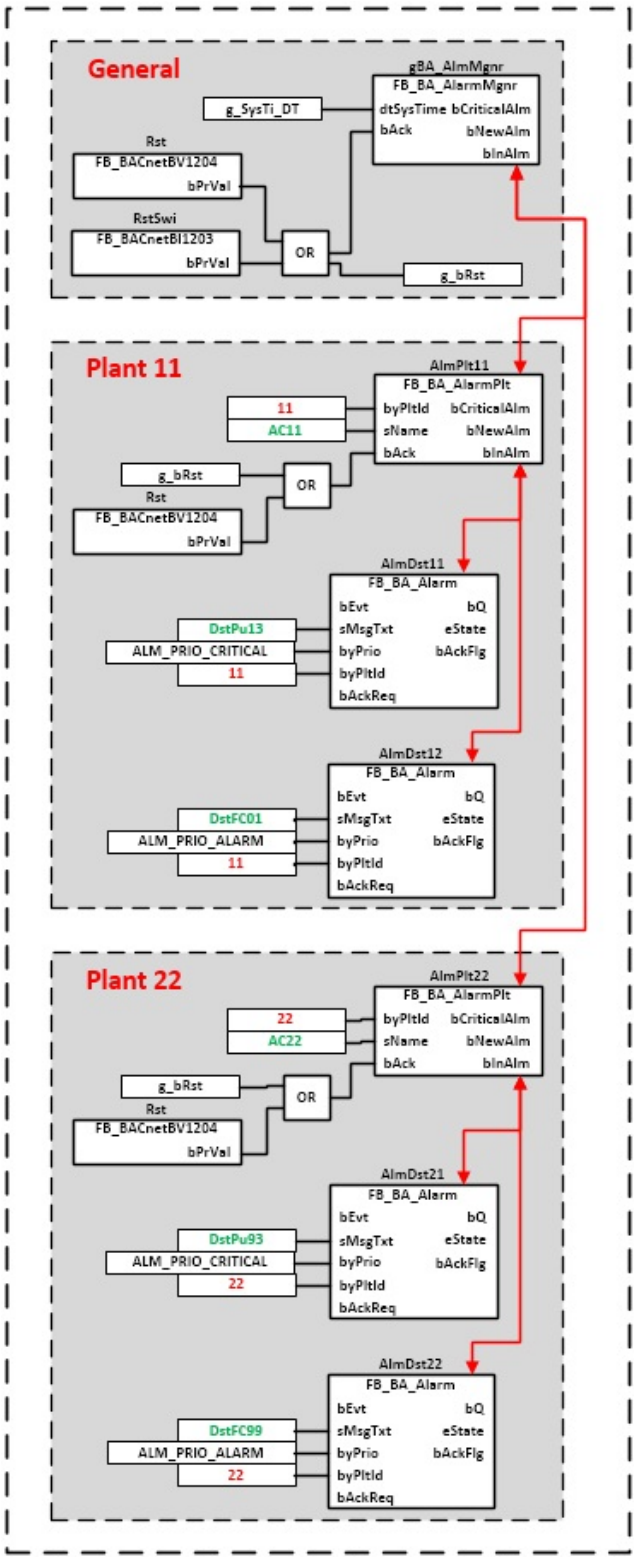
| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.71 FB_BA_AlarmMgnr

The alarm manager function block collects the alarms of all plants and pools them in a group alarm. Internally, it passes on the system time to the plants and the associated alarm output function blocks, so that a timestamp can be output in the event of an alarm.




The alarm manager is the link between all alarms of a controller and their representation in the target visualisation. The display of the alarms in the target visualisation is automatically created with the template BAC_GenAlm_01.




### Inputs/outputs

### VAR_INPUT

```
dtSysTime     : DT;
sName         : STRING;
bAckn         : BOOL;
```

**dtSysTime:** System time used as timestamp in the event of an alarm. It is internally passed to the plants **FB_BA_AlarmPlt** and the associated alarm output function blocks **FB_BA_Alarm**.

**bAckn:** Input for central alarm acknowledgement of all plants. This acknowledgement is internally passed to the alarm output function blocks **FB_BA_Alarm**.

### VAR_OUTPUT

```
bCriticalAlm : BOOL;
bNewAlm      : BOOL;
bInAlm       : BOOL;
```

**bCriticalAlm:** critical alarm within a plant. This is used in the templates for switching off a plant. To this end, the input *byPrio* in the alarm output function block **FB_BA_Alarm** must have the value 4.

**bNewAlm:** shows the new value message of an alarm within a plant.

**bInAlm:** indicates that an alarm is active within a plant.

### Requirements

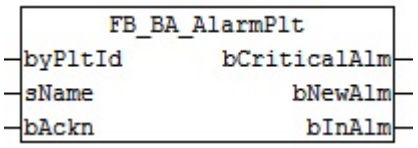
| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

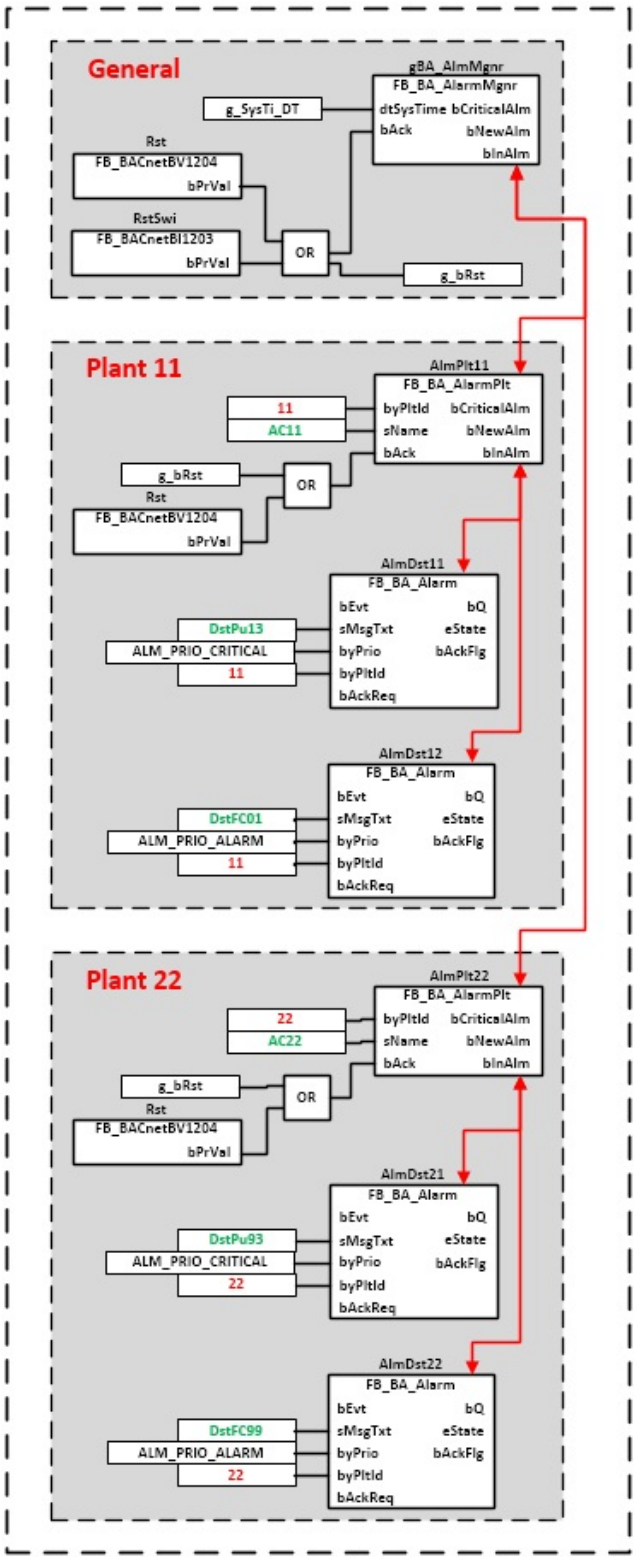
## 8.2.72 FB_BA_AlarmPlt

The function block collects the alarms of a plant and pools them in a group alarm.
The alarms are generated within the plant-related templates by the function block FB_BA_Alarm [▶ 179].
The alarms are allocated to a plant via the plant number.
The alarms are prioritised/weighted in the alarm output function block **FB_BA_Alarm** through the input *byPrio*.

**It is important that the group alarm function block FB_BA_AlarmPlt and the alarm output function blocks FB_BA_Alarm [▶ 179] within the templates of a plant are set to the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.






### Inputs/outputs

### VAR_INPUT

```
byPltId  : BYTE;
sName    : STRING;
bAck     : BOOL;
```

**byPltId:** Number of the corresponding plant. **Important: All templates of plant must be assigned to the same plant number in the PLC!**

**sName:** Input field for the plant ID (configurable via Project Builder or Excel import)

**bAck:** Input for alarm acknowledgement of the plant. This acknowledgement is internally passed to the alarm output function blocks **FB_BA_Alarm**.

### VAR_OUTPUT

```
bCriticalAlm : BOOL;
bNewAlm      : BOOL;
bInAlm       : BOOL;
```

**bCriticalAlm:** critical alarm. This is used in the templates for switching off a plant. To this end, the input *byPrio* in the alarm output function block **FB_BA_Alarm** must have the value 4.

**bNewAlm:** shows the new value message of an alarm

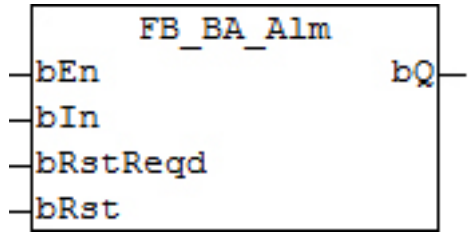
**bInAlm:** indicates that an alarm signal is present

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

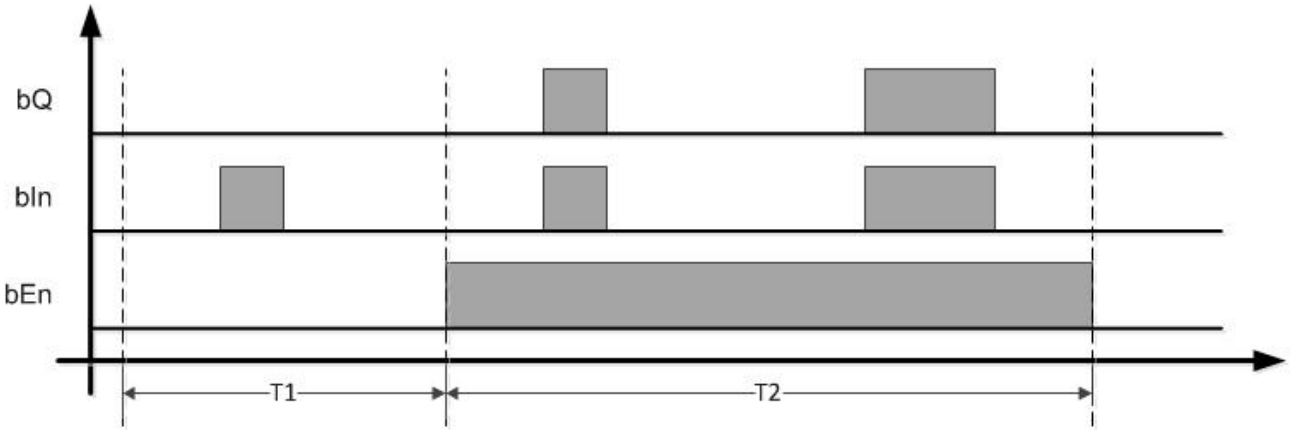
## 8.2.73 FB_BA_Alm

Alarm function block with selectable alarm memory and acknowledgement



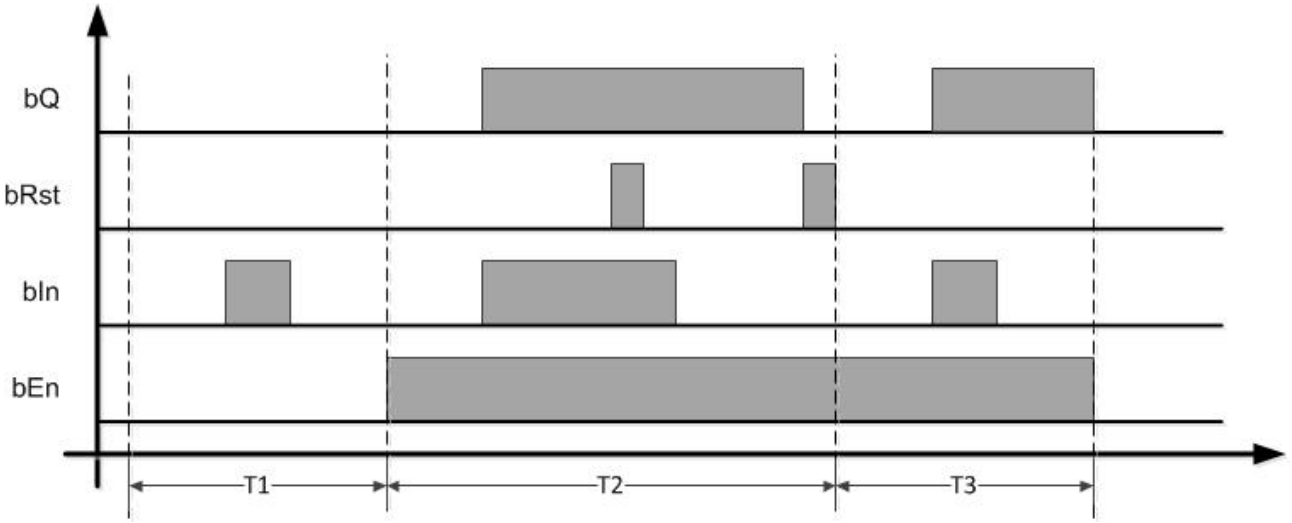

### Functional description

**without alarm memory (bRstReqd = FALSE)**




- T1: The function block is not enabled (*bEn*=FALSE). A message at input *bIn* has not effect on the alarm output *bQ*.
- T2: The function block is enabled (*bEn*=TRUE). The alarm output *bQ* remains set as long as the message is active at input *bIn*.

**with alarm memory (bRstReqd = TRUE)**




- T1: The function block is not enabled (*bEn*=FALSE). A message at input *bIn* has not effect on the alarm output *bQ*.
- T2: The function block is enabled (*bEn*=TRUE). A rising edge at alarm input *bIn* sets the alarm output *bQ*. It can be reset through a rising edge at reset input *bRst*, but only if the message is no longer active at *bIn*.
- T3: As a rule, alarm messages at *bQ* are only present if the function block *bEn* is enabled.

### Inputs/outputs

#### VAR_INPUT

```
bEn       : BOOL;
bIn       : BOOL;
bRstReqd  : BOOL;
bRst      : BOOL;
```

**bEn:** enable of the function block

**bIn:** message input

**bRstReqd:** alarm memory activation (requires acknowledgement)

**bRst:** reset alarm

#### VAR_OUTPUT

```
bQ    : BOOL;
```

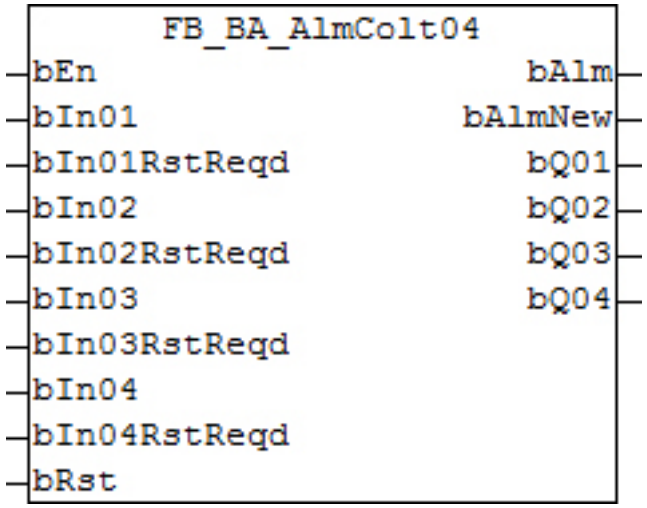
**bQ:** alarm output

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.74 FB_BA_AlmColt04

Collective alarm function block, 4 alarms




### Functional description

This function block represents a collective alarm for up to four individual alarms. Each of these individual alarms *bIn01*..*bIn04* is processed with a FB_BA_Alm [▶ 185] within the function block, with a corresponding choice of storing the alarm at inputs *bIn01RstReqd*..*bIn04RstReqd*. The states of the four individual alarms are output at *bQ01* to *bQ04*.

The output of the group alarm *bAlm* results from a logical OR link of the individual alarms. Acknowledgement is not required, if no individual alarm was parameterised to require acknowledgement. If alarm storage is enabled via the inputs *bIn01RstReqd*..*bIn04RstReqd*, the individual alarm and the group alarm can be reset globally via the input *bRst*. As already mentioned under FB_BA_Alm [▶ 185], an alarm can only be reset, if its trigger message is no longer active at *bIn*.

The output *bAlmNew* is set whenever a new message event occurs at the inputs *bIn01*..*bIn04*. It is in any case retentive - irrespective of the parameterisations at *bIn01RstReqd*..*bIn04RstReqd* - and is only reset through a rising edge at *bRst* until the next message event.

### Inputs/outputs

#### VAR_INPUT

```
bEn          : BOOL;
bIn01        : BOOL;
bIn01RstReqd : BOOL;
bIn02        : BOOL;
bIn02RstReqd : BOOL;
bIn03        : BOOL;
bIn03RstReqd : BOOL;
bIn04        : BOOL;
bIn04RstReqd : BOOL;
```

**bEn:** enable of the function block

**bIn01:** alarm input 1
**bIn01RstReqd:** alarm at input 1 requires acknowledgement

**bIn02:** alarm input 2
**bIn02RstReqd:** alarm at input 2 requires acknowledgement

**bIn03:** alarm input 3
**bIn03RstReqd:** alarm at input 3 requires acknowledgement

**bIn04:** alarm input 4
**bIn04RstReqd:** alarm at input 4 requires acknowledgement

**bRst:** all alarms that are parameterized to require acknowledgement and the group alarms *bAlm* and *bAlmNew* are reset.

### VAR_OUTPUT

```
bAlm    : BOOL;
bAlmNew : BOOL;
bQ01    : BOOL;
bQ02    : BOOL;
bQ03    : BOOL;
bQ04    : BOOL;
```

**bAlm:** group alarm - logical OR link of the individual alarms at the inputs

**bAlmNew:** flag for new alarm event

**bQ01:** alarm output 1

**bQ02:** alarm output 2

**bQ03:** alarm output 3

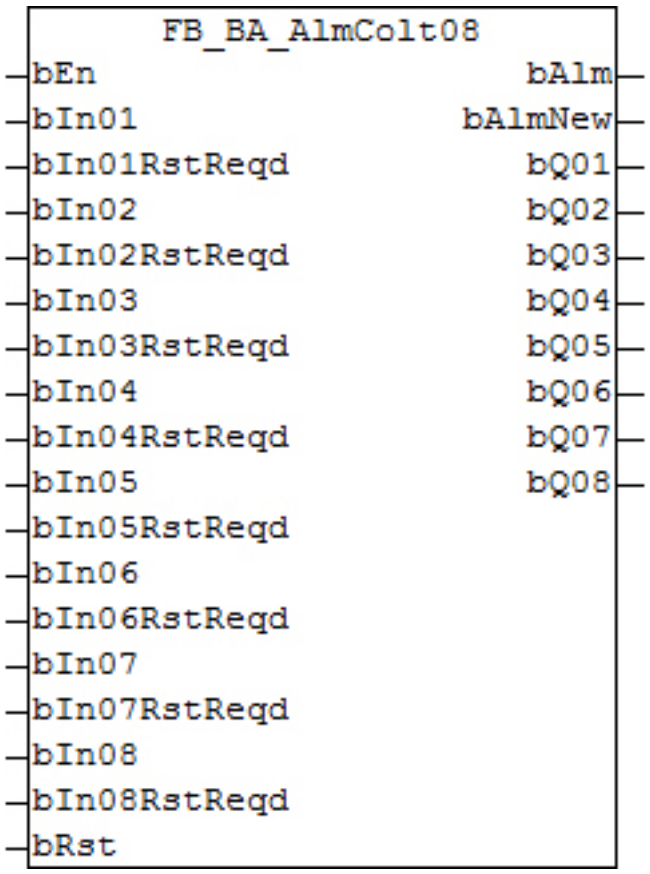
**bQ04:** alarm output 4

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.75 FB_BA_AlmColt08

Collective alarm function block, 8 alarms



### Functional description

This function block represents a collective alarm for up to 8 individual alarms. Each of these individual alarms *bIn01*..*bIn08* is processed with a FB_BA_Alm [▶ 185] within the function block, with a corresponding choice of storing the alarm at inputs *bIn01RstReqd*..*bIn08RstReqd*. The states of the eight individual alarms are output at *bQ01* to *bQ08*.

The output of the group alarm *bAlm* results from a logical OR link of the individual alarms. Acknowledgement is not required, if no individual alarm was parameterised to require acknowledgement. If alarm storage is enabled via the inputs *bIn01RstReqd*..*bIn08RstReqd*, the individual alarm and the group alarm can be reset globally via the input *bRst*. As already mentioned under FB_BA_Alm [▶ 185], an alarm can only be reset, if its trigger message is no longer active at *bIn*.

The output *bAlmNew* is set whenever a new message event occurs at the inputs *bIn01*..*bIn08*. It is in any case retentive - irrespective of the parameterisations at *bIn01RstReqd*..*bIn08RstReqd* - and is only reset through a rising edge at *bRst* until the next message event.

### Inputs/outputs

#### VAR_INPUT

```
bEn          : BOOL;
bIn01        : BOOL;
bIn01RstReqd : BOOL;
bIn02        : BOOL;
bIn02RstReqd : BOOL;
bIn03        : BOOL;
bIn03RstReqd : BOOL;
bIn04        : BOOL;
bIn04RstReqd : BOOL;
bIn05        : BOOL;
bIn05RstReqd : BOOL;
bIn06        : BOOL;
bIn06RstReqd : BOOL;
bIn07        : BOOL;
bIn07RstReqd : BOOL;
bIn08        : BOOL;
bIn08RstReqd : BOOL;
```

**bEn:** enable of the function block

**bIn01:** alarm input 1
**bIn01RstReqd:** alarm at input 1 requires acknowledgement

**bIn02:** alarm input 2
**bIn02RstReqd:** alarm at input 2 requires acknowledgement

**bIn03:** alarm input 3
**bIn03RstReqd:** alarm at input 3 requires acknowledgement

**bIn04:** alarm input 4
**bIn04RstReqd:** alarm at input 4 requires acknowledgement

**bIn05:** alarm input 5
**bIn05RstReqd:** alarm at input 5 requires acknowledgement

**bIn06:** alarm input 6
**bIn06RstReqd:** alarm at input 6 requires acknowledgement

**bIn07:** alarm input 7
**bIn07RstReqd:** alarm at input 7 requires acknowledgement

**bIn08:** alarm input 8
**bIn08RstReqd:** alarm at input 8 requires acknowledgement

**bRst:** all alarms that are parameterized to require acknowledgement and the group alarms *bAlm* and *bAlmNew* are reset.

### VAR_OUTPUT

```
bAlm    : BOOL;
bAlmNew : BOOL;
bQ01    : BOOL;
bQ02    : BOOL;
bQ03    : BOOL;
bQ04    : BOOL;
bQ05    : BOOL;
bQ06    : BOOL;
bQ07    : BOOL;
bQ08    : BOOL;
```

**bAlm:** group alarm - logical OR link of the individual alarms at the inputs

**bAlmNew:** flag for new alarm event

**bQ01:** alarm output 1

**bQ02:** alarm output 2

**bQ03:** alarm output 3

**bQ04:** alarm output 4

**bQ05:** alarm output 5

**bQ06:** alarm output 6

**bQ07:** alarm output 7

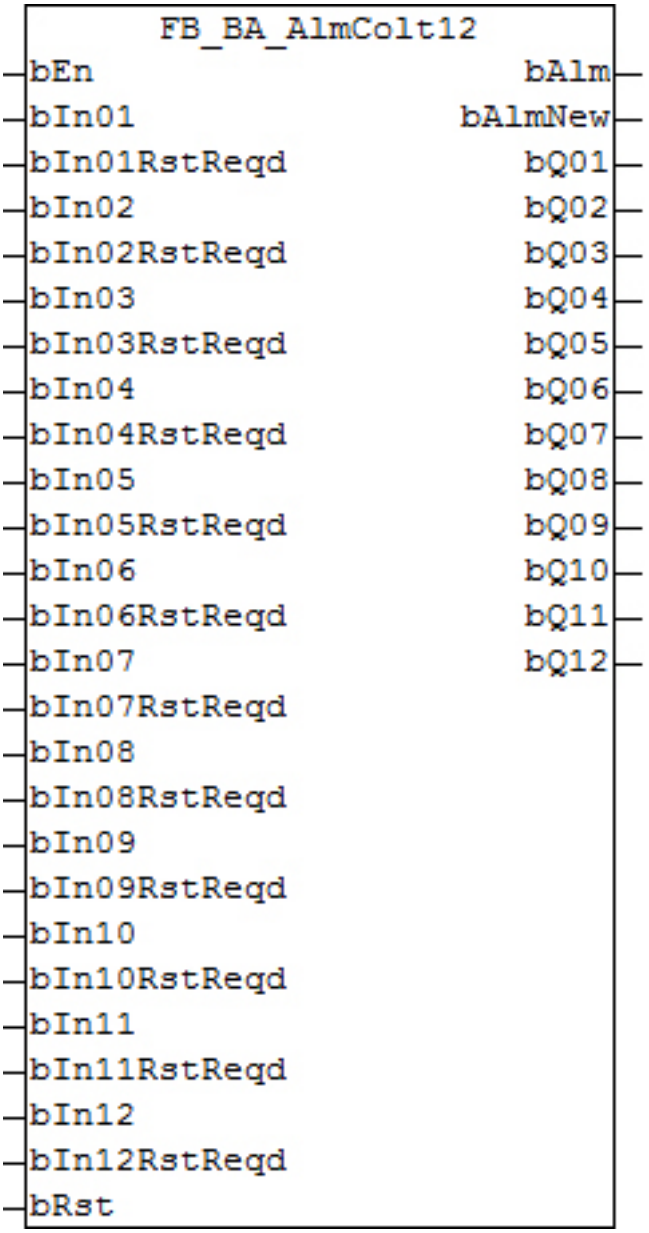
**bQ08:** alarm output 8

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.76 FB_BA_AlmColt12

Collective alarm function block, 12 alarms




**Functional description**

This function block represents a collective alarm for up to 12 individual alarms. Each of these individual alarms *bIn01*..*bIn12* is processed with a FB_BA_Alm [▶ 185] within the function block, with a corresponding choice of storing the alarm at inputs *bIn01RstReqd*..*bIn12RstReqd*. The states of the 12 individual alarms are output at *bQ01* to *bQ12*.

The output of the group alarm *bAlm* results from a logical OR link of the individual alarms. Acknowledgement is not required, if no individual alarm was parameterised to require acknowledgement. If alarm storage is enabled via the inputs *bIn01RstReqd*..*bIn12RstReqd*, the individual alarm and the group alarm can be reset globally via the input *bRst*. As already mentioned under FB_BA_Alm [▶ 185], an alarm can only be reset, if its trigger message is no longer active at *bIn*.

The output *bAlmNew* is set whenever a new message event occurs at the inputs *bIn01*..*bIn12*. It is in any case retentive - irrespective of the parameterisations at *bIn01RstReqd*..*bIn12RstReqd* - and is only reset through a rising edge at *bRst* until the next message event.

### Inputs/outputs

### VAR_INPUT

```
bEn          : BOOL;
bIn01        : BOOL;
bIn01RstReqd : BOOL;
bIn02        : BOOL;
bIn02RstReqd : BOOL;
bIn03        : BOOL;
bIn03RstReqd : BOOL;
bIn04        : BOOL;
bIn04RstReqd : BOOL;
bIn05        : BOOL;
bIn05RstReqd : BOOL;
bIn06        : BOOL;
bIn06RstReqd : BOOL;
bIn07        : BOOL;
bIn07RstReqd : BOOL;
bIn08        : BOOL;
bIn08RstReqd : BOOL;
bIn09        : BOOL;
bIn09RstReqd : BOOL;
bIn10        : BOOL;
bIn10RstReqd : BOOL;
bIn11        : BOOL;
bIn11RstReqd : BOOL;
bIn12        : BOOL;
bIn12RstReqd : BOOL;
```

**bEn:** enable of the function block

**bIn01:** alarm input 1
**bIn01RstReqd:** alarm at input 1 requires acknowledgement

**bIn02:** alarm input 2
**bIn02RstReqd:** alarm at input 2 requires acknowledgement

**bIn03:** alarm input 3
**bIn03RstReqd:** alarm at input 3 requires acknowledgement

**bIn04:** alarm input 4
**bIn04RstReqd:** alarm at input 4 requires acknowledgement

**bIn05:** alarm input 5
**bIn05RstReqd:** alarm at input 5 requires acknowledgement

**bIn06:** alarm input 6
**bIn06RstReqd:** alarm at input 6 requires acknowledgement

**bIn07:** alarm input 7
**bIn07RstReqd:** alarm at input 7 requires acknowledgement

**bIn08:** alarm input 8
**bIn08RstReqd:** alarm at input 8 requires acknowledgement

**bIn09:** alarm input 9
**bIn09RstReqd:** alarm at input 9 requires acknowledgement

**bIn10:** alarm input 10
**bIn10RstReqd:** alarm at input 10 requires acknowledgement

**bIn11:** alarm input 11
**bIn11RstReqd:** alarm at input 11 requires acknowledgement

**bIn12:** alarm input 12
**bIn12RstReqd:** alarm at input 12 requires acknowledgement

**bRst:** all alarms that are parameterized to require acknowledgement and the group alarms *bAlm* and *bAlmNew* are reset.

### VAR_OUTPUT

```
bAlm    : BOOL;
bAlmNew : BOOL;
bQ01    : BOOL;
bQ02    : BOOL;
bQ03    : BOOL;
bQ04    : BOOL;
bQ05    : BOOL;
bQ06    : BOOL;
bQ07    : BOOL;
bQ08    : BOOL;
bQ09    : BOOL;
bQ10    : BOOL;
bQ11    : BOOL;
bQ12    : BOOL;
```

**bAlm:** group alarm - logical OR link of the individual alarms at the inputs

**bAlmNew:** flag for new alarm event

**bQ01:** alarm output 1

**bQ02:** alarm output 2

**bQ03:** alarm output 3

**bQ04:** alarm output 4

**bQ05:** alarm output 5

**bQ06:** alarm output 6

**bQ07:** alarm output 7

**bQ08:** alarm output 8

**bQ09:** alarm output 9

**bQ10:** alarm output 10

**bQ11:** alarm output 11

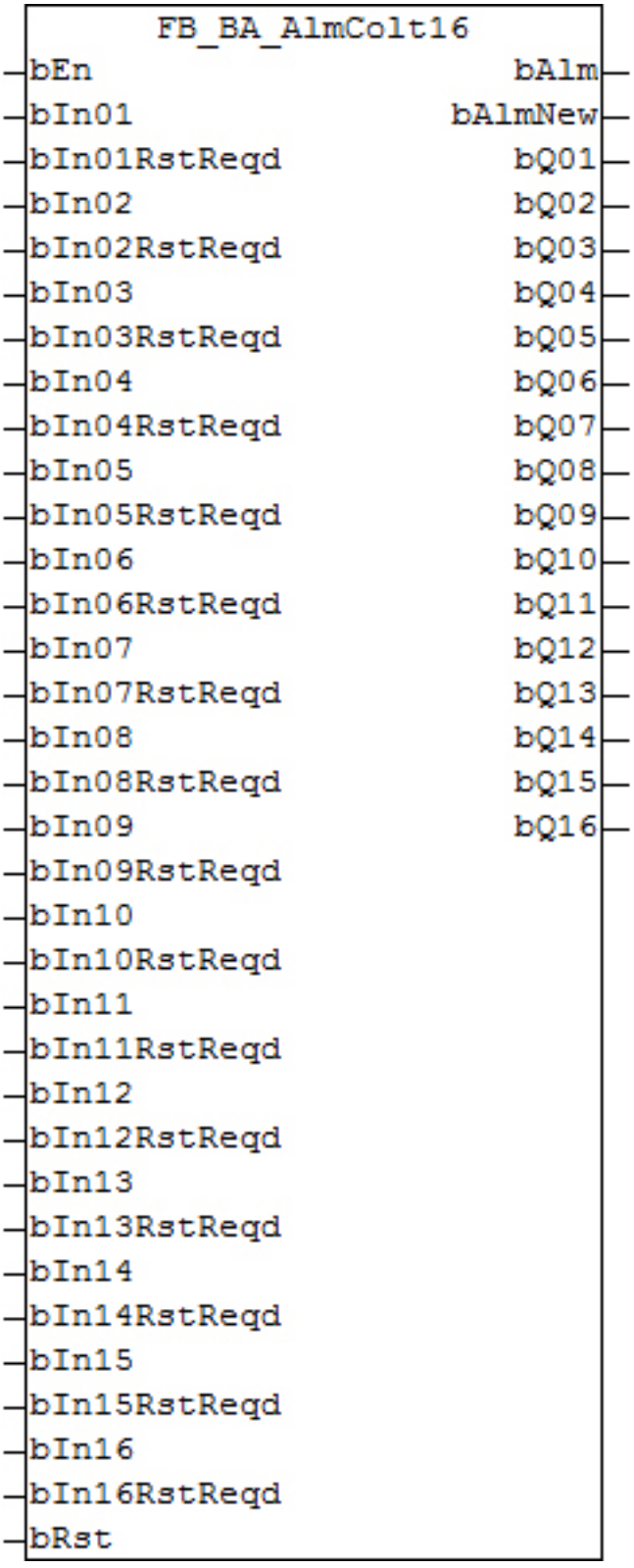
**bQ12:** alarm output 12

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.77 FB_BA_AlmColt16

Collective alarm function block, 16 alarms




### Functional description

This function block represents a collective alarm for up to 16 individual alarms. Each of these individual alarms *bIn01*..*bIn16* is processed with a FB_BA_Alm [▶ 185] within the function block, with a corresponding choice of storing the alarm at inputs *bIn01RstReqd*..*bIn16RstReqd*. The states of the 16 individual alarms are output at *bQ01* to *bQ16*.

The output of the group alarm *bAlm* results from a logical OR link of the individual alarms. Acknowledgement is not required, if no individual alarm was parameterised to require acknowledgement. If alarm storage is enabled via the inputs *bIn01RstReqd..bIn16RstReqd*, the individual alarm and the group alarm can be reset globally via the input *bRst*. As already mentioned under FB_BA_Alm [▶ 185], an alarm can only be reset, if its trigger message is no longer active at *bIn*.

The output *bAlmNew* is set whenever a new message event occurs at the inputs *bIn01..bIn16*. It is in any case retentive - irrespective of the parameterisations at *bIn01RstReqd..bIn16RstReqd* - and is only reset through a rising edge at *bRst* until the next message event.

### Inputs/outputs

### VAR_INPUT

```
bEn          : BOOL;
bIn01        : BOOL;
bIn01RstReqd : BOOL;
bIn02        : BOOL;
bIn02RstReqd : BOOL;
bIn03        : BOOL;
bIn03RstReqd : BOOL;
bIn04        : BOOL;
bIn04RstReqd : BOOL;
bIn05        : BOOL;
bIn05RstReqd : BOOL;
bIn06        : BOOL;
bIn06RstReqd : BOOL;
bIn07        : BOOL;
bIn07RstReqd : BOOL;
bIn08        : BOOL;
bIn08RstReqd : BOOL;
bIn09        : BOOL;
bIn09RstReqd : BOOL;
bIn10        : BOOL;
bIn10RstReqd : BOOL;
bIn11        : BOOL;
bIn11RstReqd : BOOL;
bIn12        : BOOL;
bIn12RstReqd : BOOL;
bIn13        : BOOL;
bIn13RstReqd : BOOL;
bIn14        : BOOL;
bIn14RstReqd : BOOL;
bIn15        : BOOL;
bIn15RstReqd : BOOL;
bIn16        : BOOL;
bIn16RstReqd : BOOL;
bRst         : BOOL;
```

**bEn:** enable of the function block

**bIn01:** alarm input 1
**bIn01RstReqd:** alarm at input 1 requires acknowledgement

**bIn02:** alarm input 2
**bIn02RstReqd:** alarm at input 2 requires acknowledgement

**bIn03:** alarm input 3
**bIn03RstReqd:** alarm at input 3 requires acknowledgement

**bIn04:** alarm input 4
**bIn04RstReqd:** alarm at input 4 requires acknowledgement

**bIn05:** alarm input 5
**bIn05RstReqd:** alarm at input 5 requires acknowledgement

**bIn06:** alarm input 6
**bIn06RstReqd:** alarm at input 6 requires acknowledgement

**bIn07:** alarm input 7
**bIn07RstReqd:** alarm at input 7 requires acknowledgement

**bIn08:** alarm input 8
**bIn08RstReqd:** alarm at input 8 requires acknowledgement

**bIn09:** alarm input 9
**bIn09RstReqd:** alarm at input 9 requires acknowledgement

**bIn10:** alarm input 10
**bIn10RstReqd:** alarm at input 10 requires acknowledgement

**bIn11:** alarm input 11
**bIn11RstReqd:** alarm at input 11 requires acknowledgement

**bIn12:** alarm input 12
**bIn12RstReqd:** alarm at input 12 requires acknowledgement

**bIn13:** alarm input 13
**bIn13RstReqd:** alarm at input 13 requires acknowledgement

**bIn14:** alarm input 14
**bIn14RstReqd:** alarm at input 14 requires acknowledgement

**bIn15:** alarm input 15
**bIn15RstReqd:** alarm at input 15 requires acknowledgement

**bIn16:** alarm input 16
**bIn16RstReqd:** alarm at input 16 requires acknowledgement

**bRst:** all alarms that are parameterized to require acknowledgement and the group alarms *bAlm* and *bAlmNew* are reset.

### VAR_OUTPUT

```
bAlm    : BOOL;
bAlmNew : BOOL;
bQ01    : BOOL;
bQ02    : BOOL;
bQ03    : BOOL;
bQ04    : BOOL;
bQ05    : BOOL;
bQ06    : BOOL;
bQ07    : BOOL;
bQ08    : BOOL;
bQ09    : BOOL;
bQ10    : BOOL;
bQ11    : BOOL;
bQ12    : BOOL;
bQ13    : BOOL;
bQ14    : BOOL;
bQ15    : BOOL;
bQ16    : BOOL;
```

**bAlm:** group alarm - logical OR link of the individual alarms at the inputs

**bAlmNew:** flag for new alarm event

**bQ01:** alarm output 1

**bQ02:** alarm output 2

**bQ03:** alarm output 3

**bQ04:** alarm output 4

**bQ05:** alarm output 5

**bQ06:** alarm output 6

**bQ07:** alarm output 7

**bQ08:** alarm output 8

**bQ09:** alarm output 9

**bQ10:** alarm output 10

**bQ11:** alarm output 11

**bQ12:** alarm output 12

**bQ13:** alarm output 13

**bQ14:** alarm output 14

**bQ15:** alarm output 15

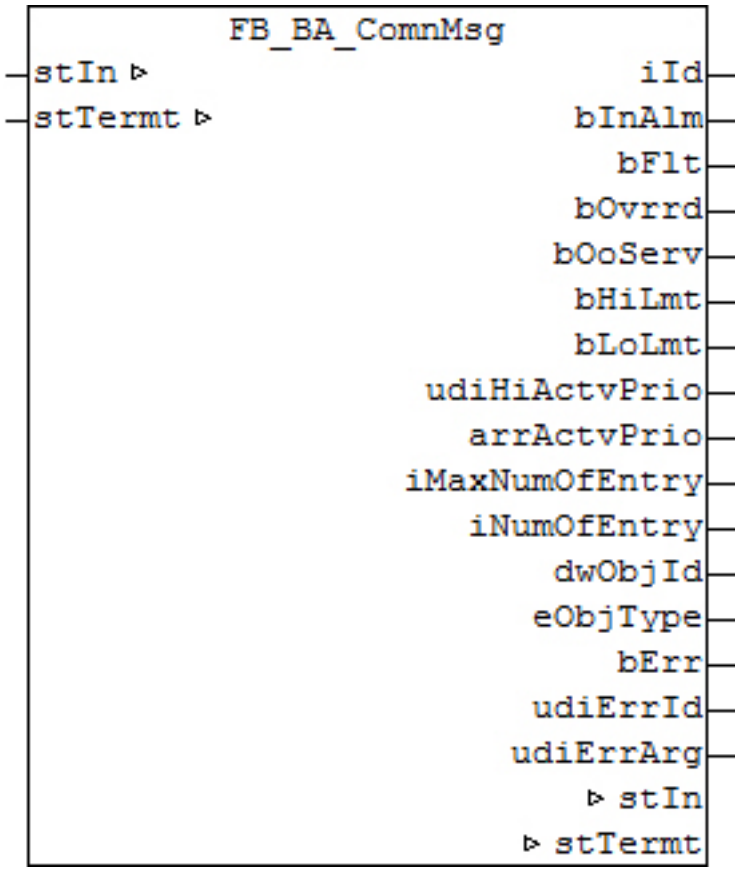
**bQ16:** alarm output 16

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

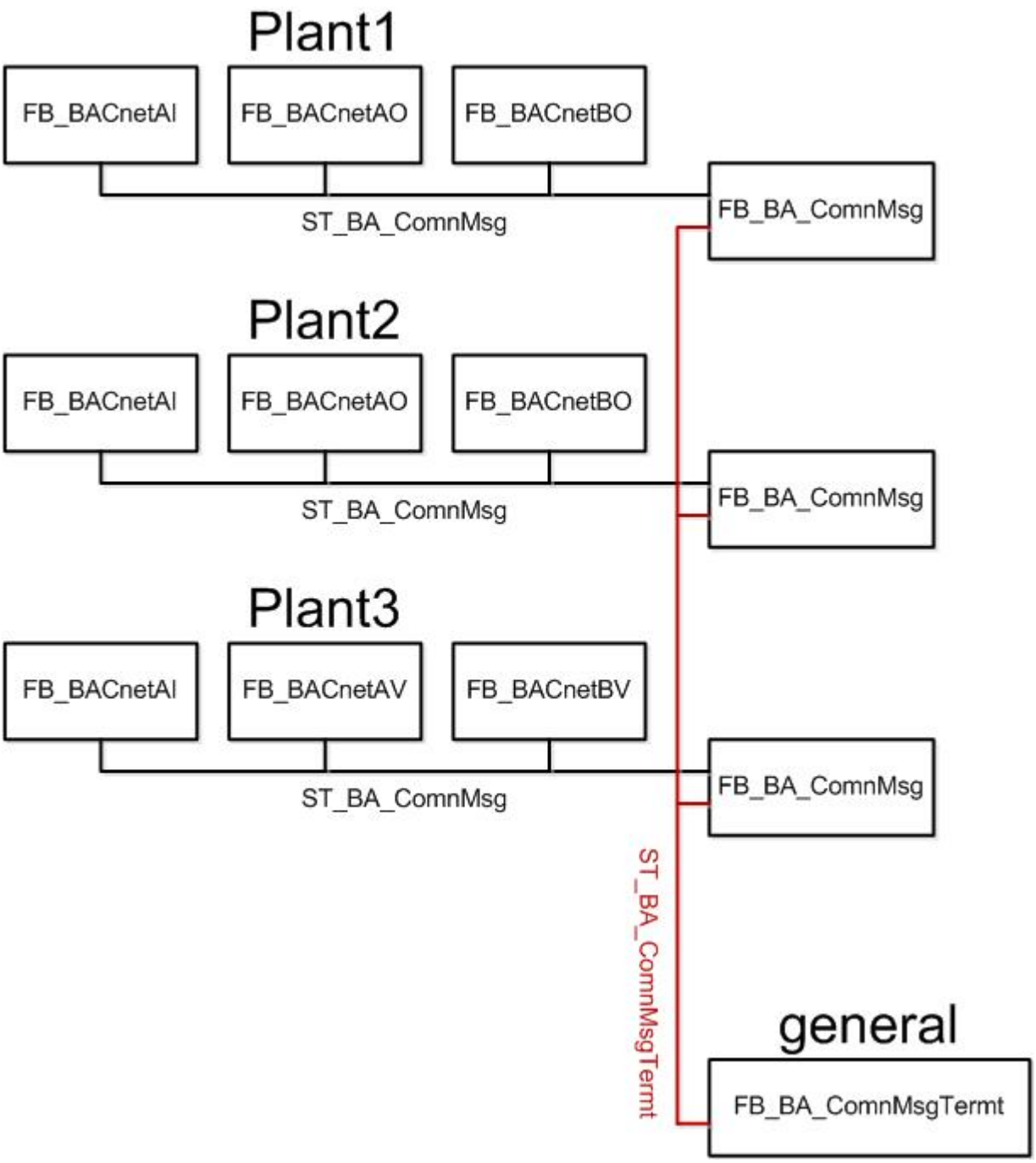
## 8.2.78 FB_BA_ComnMsg

Formation of collective messages



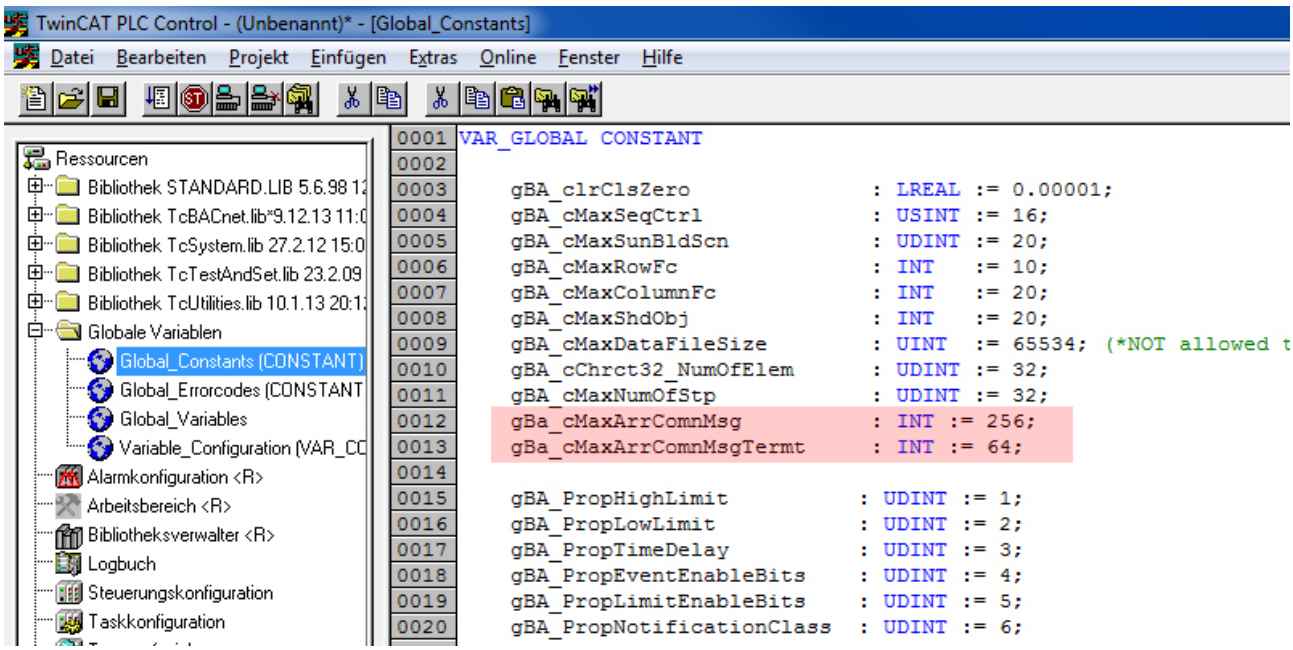

### Functional description

The function blocks FB_BA_ComnMsg and FB_BA_ComnMsgTermt [▶ 200] are used to form collective messages at the plant level and the controller level. The information of the BACnet objects is transferred to the function blocks ST_BA_ComnMsg [▶ 326] in the structure ST_BA_ComnMsg. The variables *stOut* are used to transfer the collective plant messages to the function block FB_BA_ComnMsgTermt [▶ 200]. All messages of the BACnet objects of a BACnet controller are consolidated here.




**Error handling:**

In the first cycle after a controller restart, each BACnet object reserves a field in a one-dimensional array within the message structure ST_BA_ComnMsg [▶ 326]. The size of the array is defined with the global constant *gBA_cMaxArrComnMsg*. The default for this constant is 256. If more than 256 BACnet objects are connected to an instance of FB_BA_ComnMsg, it has to be increased. The transfer of messages to FB_BA_ComnMsgTermt [▶ 200] is set to 64 within the structure ST_BA_ComnMsgTermt [▶ 326]. For a higher number the constant *gBA_cMaxArrComnMsgTermt* has to be increased.




**Inputs/outputs**

### VAR_IN_OUT

```
stIn     : ST_BA_ComnMsg;
stTermt  : ST_BA_ComnMsgTermt;
```

**stIn:** data structure for connecting the BACnet objects at the plant level

**stTermt:** data structure [▶ 326] for transferring collective messages from the plant level to function block FB_BA_ComnMsgTermt [▶ 200].

### VAR_OUTPUT

```
iId             : INT;
bInAlm          : BOOL;
bFlt            : BOOL;
bOvrrd          : BOOL;
bOoServ         : BOOL;
bHiLmt          : BOOL;
bLoLmt          : BOOL;
udiHiActvPrio   : UDINT;
arrActvPrio     : ARRAY [1..16] OF BOOL;
iMaxNumOfEntry  : INT;
iNumOfEntry     : INT;
dwObjId         : DWORD;
eObjType        : E_BACnetObjectType;
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**iId:** ID of the last information telegram (*stTermt.arrData[iId]*) sent to FB_BA_ComnMsgTermt

**bInAlm:** the last sending BACnet object is in alarm status.

**bFlt:** the last sending BACnet object is in fault status.

**bOvrrd:** in the last sending BACnet object, the local mechanical priority operation is enabled.

**bOoServ:** the last sending BACnet object is "out of service".

**bHiLmt:** in the last sending BACnet object the Hi limit is exceeded.

**bLoLmt:** in the last sending BACnet object the Lo limit is undershot.

**udiHiActvPrio:** indicates the highest written priority of all BACnet objects.

**arrActvPrio:** the array provides an overview of the active priorities of all connected BACnet objects.

**iMaxNumOfEntry:** indicates how many BACnet objects can be connected to the function block.

**iNumOfEntry:** indicates how many BACnet objects are connected to the function block.

**dwObjId:** ID of the object that last entered the error state.

**eObjType:** object type of the object that last entered the error state.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

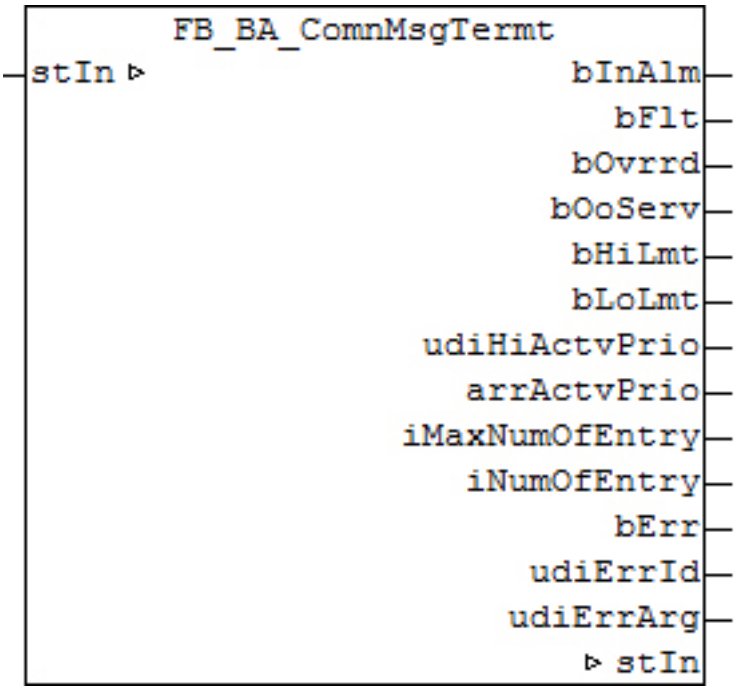
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.79 FB_BA_ComnMsgTermt

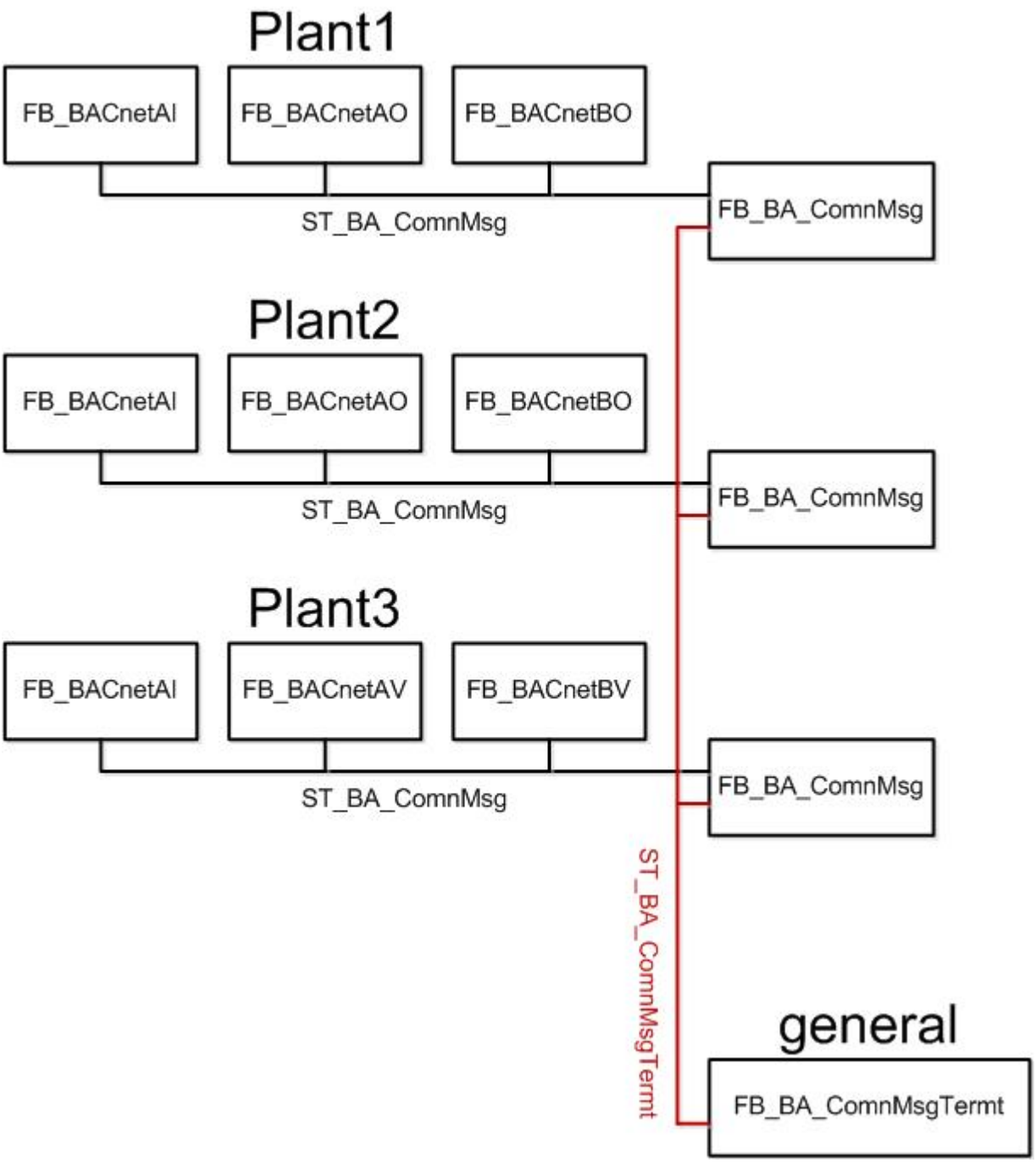
Formation of collective messages




### Functional description

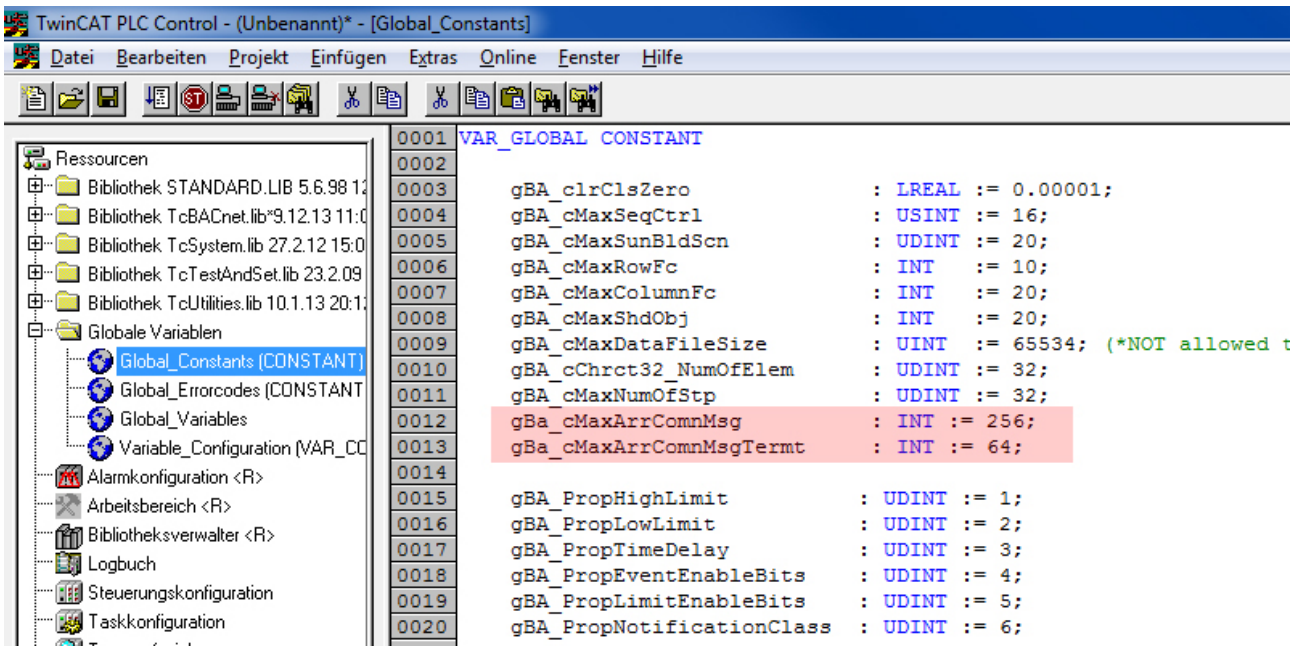
The function blocks FB_BA_ComnMsg [▶ 197] and FB_BA_ComnMsgTermt are used to form collective messages at the plant level and the controller level.

The information of the BACnet objects is transferred to the function blocks FB_BA_ComnMsg [▶ 197] in the structure ST_BA_ComnMsg [▶ 326]. The variables *stOut* are used to transfer the collective plant messages to the function block FB_BA_ComnMsgTermt. All messages of the BACnet objects of a BACnet controller are consolidated here.




**Error handling:**

In the first cycle after a controller restart, each BACnet object reserves a field in a one-dimensional array within the message structure ST_BA_ComnMsg [▶ 326]. The size of the array is defined with the global constant *gBA_cMaxArrComnMsg*. The default for this constant is 256. If more than 256 BACnet objects are connected to an instance of FB_BA_ComnMsg [▶ 197], it has to be increased. The transfer of messages to FB_BA_ComnMsgTermt is set to 64 within the structure ST_BA_ComnMsgTermt. For a higher number the constant *gBA_cMaxArrComnMsgTermt* has to be increased.




### Inputs/outputs

### VAR_IN_OUT

```
stIn     : ST_BA_ComnMsgTermt;
```

**stIn:** Data structure [▶ 326] for receiving collective messages from the plant level to function block FB_BA_ComnMsgTermt.

### VAR_OUTPUT

```
bInAlm          : BOOL;
bFlt            : BOOL;
bOvrrd          : BOOL;
bOoServ         : BOOL;
bHiLmt          : BOOL;
bLoLmt          : BOOL;
udiHiActvPrio   : UDINT;
arrActvPrio     : ARRAY [1..16] OF BOOL;
iMaxNumOfEntry  : INT;
iNumOfEntry     : INT;
dwObjId         : DWORD;
eObjType        : E_BACnetObjectType;
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**bInAlm:** the last sending BACnet object is in alarm status.

**bFlt:** the last sending BACnet object is in fault status.

**bOvrrd:** in the last sending BACnet object, the local mechanical priority operation is enabled.

**bOoServ:** the last sending BACnet object is "out of service".

**bHiLmt:** in the last sending BACnet object the Hi limit is exceeded.

**bLoLmt:** in the last sending BACnet object the Lo limit is undershot.

**udiHiActvPrio:** indicates the highest written priority of all BACnet objects.

**arrActvPrio:** the array provides an overview of the active priorities of all connected BACnet objects.

**iMaxNumOfEntry:** indicates how many BACnet objects can be connected to the function block.

**iNumOfEntry:** indicates how many BACnet objects are connected to the function block.

**eObjType:** object type of the object that last entered the error state.

**dwObjId:** ID of the object that last entered the error state.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

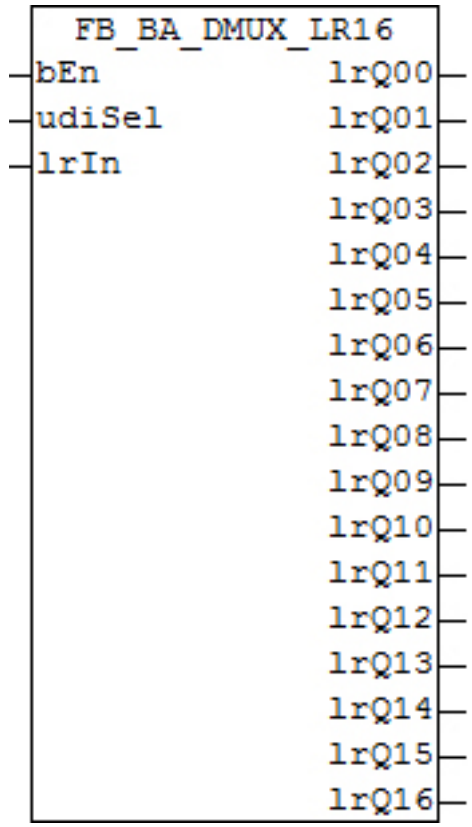
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.80 FB_BA_DMUX_XX

Demultiplexer blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT and UDINT) and in different output parameters (5, 9, 13 and 17), but they all have the same functionality.
The block FB_BA_DMUX_LR16 is described as an example.




### Functional description

In the activated state (*bEn*=TRUE), the function block outputs the value at input *lrIn* to that 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 | lrQ00 = 0.0 |
| udiSel = 5 | lrQ01 = 0.0 |
| lrIn = 32.5 | lrQ02 = 0.0 |
| | lrQ03 = 0.0 |

| Inputs | Outputs |
|---|---|
| | 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 | lrQ00 = 0.0 |
| udiSel = 25 | lrQ01 = 0.0 |
| rIn = 32.5 | lrQ02 = 0.0 |
| | 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.

### Inputs/outputs

### VAR_INPUT

```
bEn    : BOOL;
udiSel : UDINT;
lrIn   : 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.

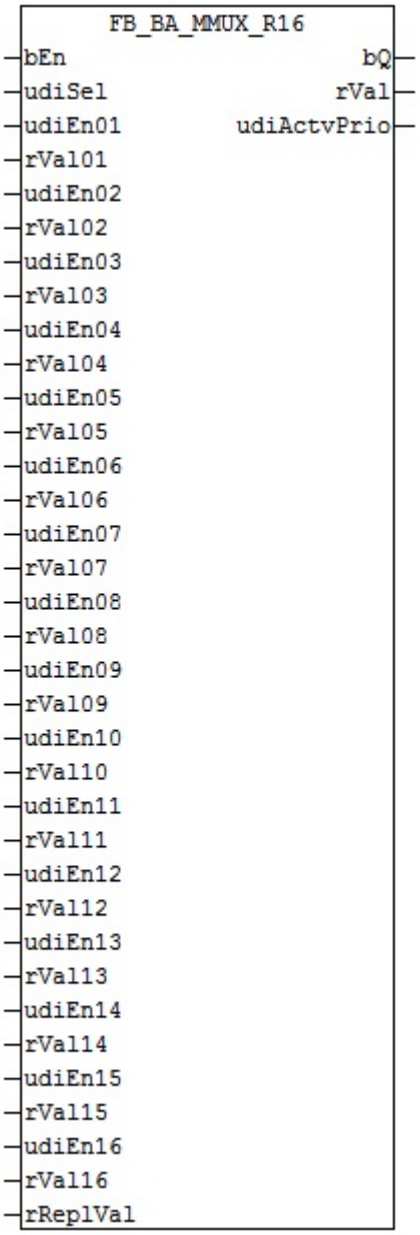
### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.81 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 and UDINT) 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.




**Functional description**

The function block switches one of the input values *rValxx* to the output *rVal* in the activated state (*bEn*=TRUE) depending on a selector *udiSel* and the corresponding input selector condition *udiEnxx*.
If several input selector conditions *udiEn01*...*udiEn16* are equal and the selector *udiSel* matches a condition, then the input value *rVal01*...*rVal16* of the lowest active selector condition is switched to the output *rVal*. *udiEn01* is the lowest, *udiEn16* the highest selector condition.

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 to *rVal* . *bQ* is then FALSE and *udiActvPrio* indicates a 255.**

**Example:**

| **Inputs** | | **Output** | |
|---|---|---|---|
| **Variable** | **Value** | **Variable** | **Value** |
| bEn | TRUE | bQ | TRUE |

| Inputs | | Output | |
|---|---|---|---|
| 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 | | | |

### Inputs/outputs

### VAR_INPUT

```
bEn        : BOOL;
udiSel     : UDINT;
udiEn01    : UDINT := 255;
rVal01     : REAL;
udiEn02    : UDINT := 255;
rVal02     : REAL;
udiEn03    : UDINT := 255;
rVal03     : REAL;
udiEn04    : UDINT := 255;
rVal04     : REAL;
udiEn05    : UDINT := 255;
rVal05     : REAL;
udiEn06    : UDINT := 255;
rVal06     : REAL;
udiEn07    : UDINT := 255;
rVal07     : REAL;
udiEn08    : UDINT := 255;
```

```
rVal08    : REAL;
udiEn09   : UDINT := 255;
rVal09    : REAL;
udiEn10   : UDINT := 255;
rVal10    : REAL;
udiEn11   : UDINT := 255;
rVal11    : REAL;
udiEn12   : UDINT := 255;
rVal12    : REAL;
udiEn13   : UDINT := 255;
rVal13    : REAL;
udiEn14   : UDINT := 255;
rVal14    : REAL;
udiEn15   : UDINT := 255;
rVal15    : REAL;
udiEn16   : UDINT := 255;
rVal16    : REAL;
rReplVal  : REAL;
```

**bEn:** activation of the block function

**udiSel:** selector

**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

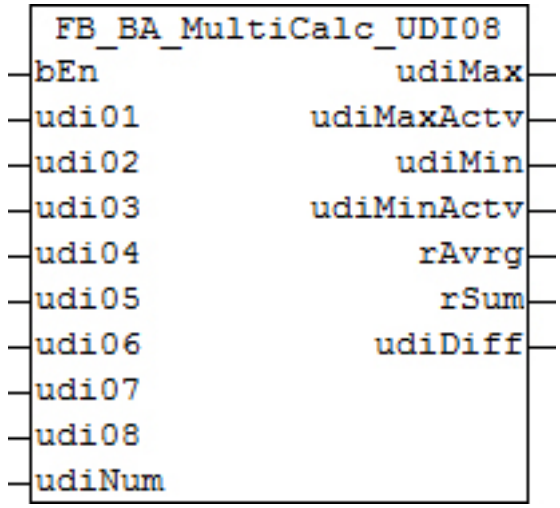
**udiActvPrio:** indicates which input selector condition is active.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.82 FB_BA_MultiCalc_XX

Multi-calculation function blocks exist for different variable types (LREAL, REAL, INT, UINT and UDINT) and in different input parameters (4 and 8), but they all have the same functionality.
The function block FB_BA_UDI08 is described as an example.




### Functional description

In activate state (*bEn*=TRUE), the function block determines the following from the 8 input values *udi01*...*udi08*:

- the maximum value of all inputs *udiMax*
- the input at which this maximum value is applied *udiMinActv*
- the minimum value of all inputs *udiMin*
- the input at which this minimum value is applied *udiMinActv*
- the average of all inputs *rAvrg*
- the sum of all inputs *rSum*
- the difference between the maximum and minimum value *udiDiff*

If not all inputs are to be calculated, the number can be limited by an entry at *udiNum*: with *udiNum*=6, for example, the calculations are only performed for the inputs *udi01*...*udi06*.
An entry greater than 8 is automatically limited to 8, an entry less than 1 is automatically limited to 1.

### Example:

| Inputs | Output |
|---|---|
| bEn = TRUE | udiMax = 32 |
| udi01 = 32 | udiMaxActv = 1 |
| udi02 = 17 | udiMin = 5 |
| udi03 = 5 | udiMinActv = 3 |
| udi04 = 9 | rAvrg = 18.5 |
| udi05 = 16 | rSum = 111 |
| udi06 = 32 | udiDiff = 27 |
| udi07 = 25 | |
| udi08 = 44 | |
| udiNum = 6 | |

If *bEn* = FALSE, 0 is output at all outputs.

### Inputs/outputs

#### VAR_INPUT

```
bEn    : BOOL;
udi01  : UDINT;
udi02  : UDINT;
udi03  : UDINT;
udi04  : UDINT;
```

```
udi05  : UDINT;
udi06  : UDINT;
udi07  : UDINT;
udi08  : UDINT;
udiNum : UDINT;
```

**bEn:** activation of the block function

**udi01...udi08:** input values from which to calculate.

**udiNum:** number of input values to be used for the calculation.

### VAR_OUTPUT

```
udiMax     : UDINT;
udiMaxActv : UDINT;
udiMin     : UDINT;
udiMinActv : UDINT;
rAvrg      : REAL;
rSum       : REAL;
udiDiff    : UDINT;
```

**udiMax:** maximum value of all inputs

**udiMaxActv:** input to which the maximum value is applied.

**udiMin:** minimum value of all inputs

**udiMinActv:** input to which the minimum value is applied

**rAvrg:** average of all inputs

**rSum:** sum of all inputs

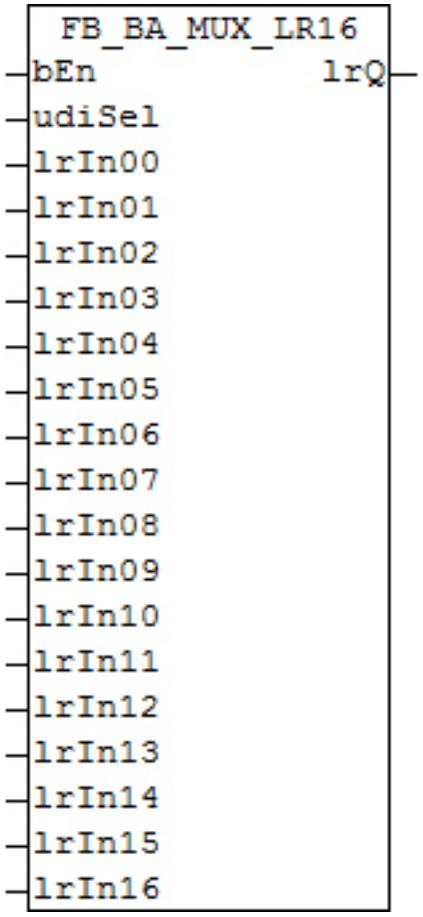
**udiDiff:** difference between the maximum and minimum value

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.83 FB_BA_MUX_XX

Multiplexer blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT and UDINT) and in different input parameters (5, 9, 13 and 17), but they all have the same functionality.
The function block FB_BA_MUX_LR16 is described as an example.




### Functional description

The function block outputs in the activated state (*bEn*=TRUE) that input value *lrIn00..lrIn16* at output *lrQ* whose number is entered at input *udiSel*.

### Example:

| Inputs | Output |
|---|---|
| bEn = TRUE | lrQ = 16.5 |
| udiSel = 5 | |
| lrIn00 = 15.9 | |
| lrIn01 = 32.5 | |
| lrIn02 = 17.4 | |
| lrIn03 = 5.84 | |
| lrIn04 = 9.56 | |
| lrIn05 = 16.5 | |
| lrIn06 = 32,781 | |
| lrIn07 = 25.4 | |
| lrIn08 = 44.5 | |
| lrIn09 = 66.1 | |
| lrIn10 = 45.5 | |
| lrIn11 = 83.3 | |
| lrIn12 = 54.56 | |
| lrIn13 = 33.8 | |
| lrIn14 = 98.5 | |
| lrIn15 = 71.3 | |

| Inputs | Output |
|---|---|
| lrIn16 = 2.3 | |

If the entered value at *udiSel* is greater than the number of inputs, the "highest" input is output at *lrQ*:

| Inputs | Output |
|---|---|
| bEn = TRUE | lrQ = 2.3 |
| udiSel = 25 | |
| lrIn00 = 15.9 | |
| lrIn01 = 32.5 | |
| lrIn02 = 17.4 | |
| lrIn03 = 5.84 | |
| lrIn04 = 9.56 | |
| lrIn05 = 16.5 | |
| lrIn06 = 32,781 | |
| lrIn07 = 25.4 | |
| lrIn08 = 44.5 | |
| lrIn09 = 66.1 | |
| lrIn10 = 45.5 | |
| lrIn11 = 83.3 | |
| lrIn12 = 54.56 | |
| lrIn13 = 33.8 | |
| lrIn14 = 98.5 | |
| lrIn15 = 71.3 | |
| lrIn16 = 2.3 | |

If *bEn*=FALSE, 0.0 is output at output *lrQ* or FALSE for boolean multiplexers.

### Inputs/outputs

### 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   : UDINT;
```

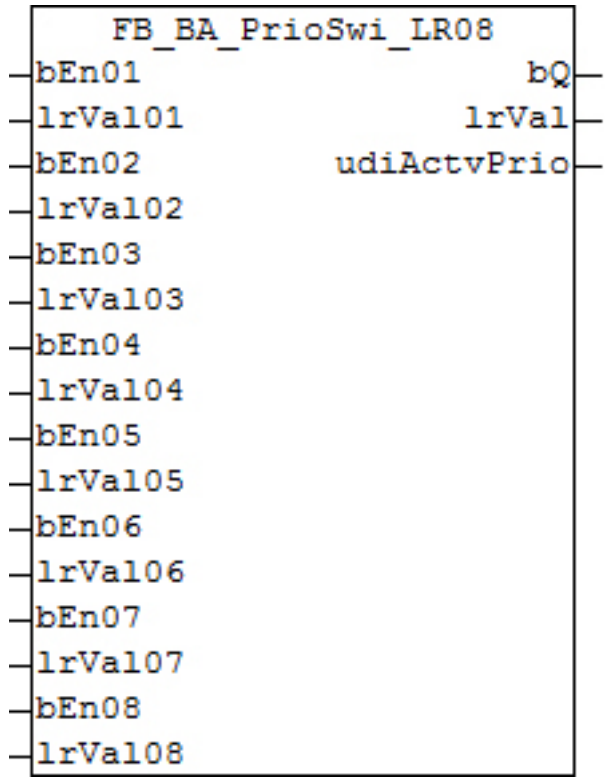
**lrQ:** value of the selected input

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.84 FB_BA_PrioSwi_XX

The priority switches exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT and UDINT) and in different output sizes (4, 8, 12, 16 and 24), but they all have the same functionality.
The block FB_BA_PrioSwi_LR08 is described as an example.




### Functional description

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 | | | |

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| 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 | | | |

### Inputs/outputs

### 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:** enable of 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 | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.85 FB_BA_AntBlkg

Blocking protection for pumps and actuators

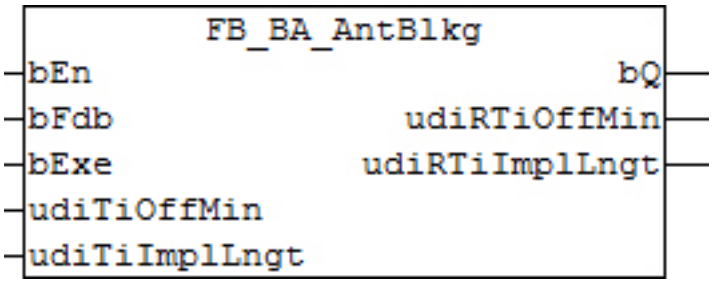



Fig. 1: FB_BA_AntBlkg

### Functional description

This function block prevents blocking of pumps or actuators after prolonged idle periods by issuing a switch-on pulse.
The maximum idle period before such a pulse is issued is determined by the value of the variable *udiTiOffMin*. For logging the idle time, the input *bFdb* must be linked to the operating feedback from the unit. The length of the pulse is parameterised with the variable *udiTiImplLngt*. The input *bExe* should be used if the blocking protection pulses are to be issued cyclically based on a switching schedule, rather than depending on the idle times. A rising edge at *bExe* immediately triggers output of a pulse to *bQ*. Generally, a pulse output only occurs if the function block at *bEn* is enabled.

### Inputs/outputs

#### VAR_INPUT

```
bEn           : BOOL;
bFdb          : BOOL;
bExe          : BOOL;
udiTiOffMin   : UDINT;
udiTiImplLngt : 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:** minimum switch-off time [s]: a pulse is issued once the time *udiTiOffMin* has elapsed without movement of the aggregate.

**udiTiImplLngt:** length of the blocking protection pulse [s] at *bQ*

#### VAR_OUTPUT

```
bQ             : BOOL;
udiRTiOffMin   : UDINT;
udiRTiImplLngt : UDINT;
```

**bQ:** output for the output of the pulse

**udiRTiOffMin:** remaining time [s] before the next pulse is issued in the absence of movement.

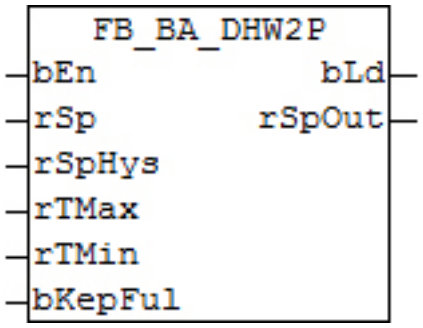
**udiTiImplLngt:** remaining residual time [s] of the pulse at *bQ*

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.86 FB_BA_DHW2P

Hot water tank control




### Functional description

This function block 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 variables *rSp* is used to transfer the set value 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.

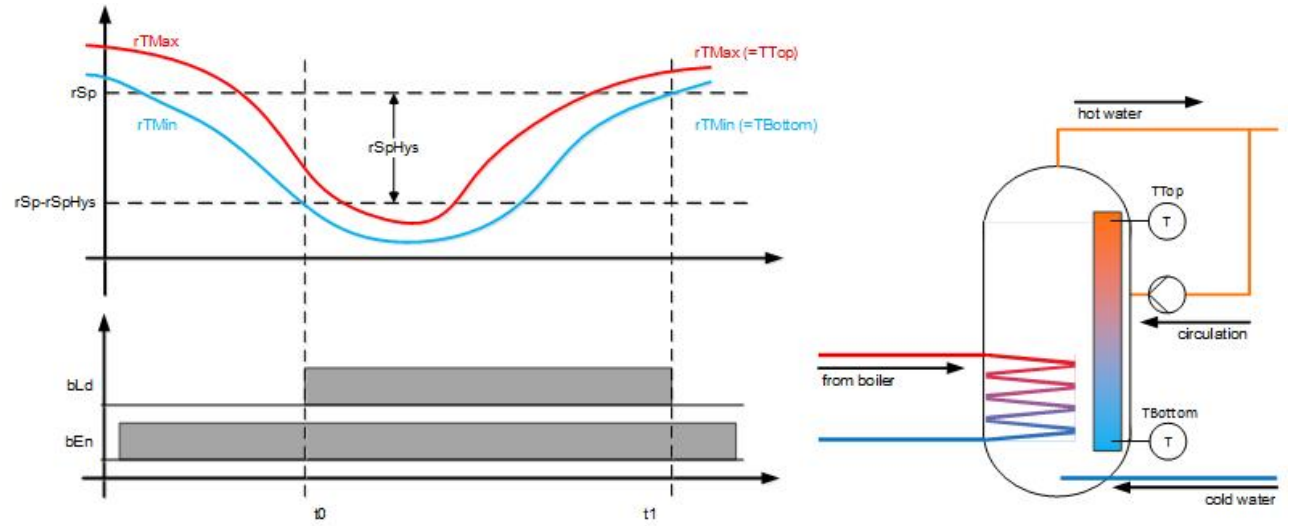
The tank can be charged in two ways via the variables *bKepFul*:

#### bKepFul = FALSE

Charging is requested if *rTMax* falls below the value of *rSp-rSpHys*. Charging is disabled if *rTMin* is above the set value of *rSp*.
Since 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*. Charging is disabled if *rTMin* is above the set value 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/outputs

### 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 temperature sensors [°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 of the charging mode

**rSpOut:** setpoint transfer to charging circuit

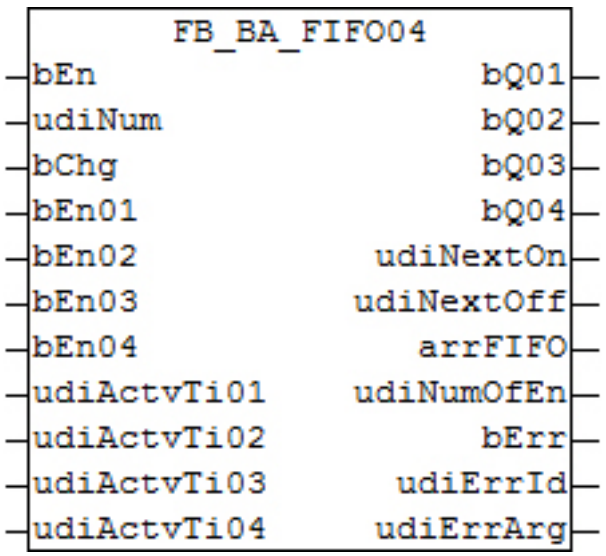
- *rSpOut* = *rSp* (input) if the function block is enabled
- *rSpOut* = 0 if the function block is not enabled

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.87 FB_BA_FIFO04

Sequential control of up to four units




### Functional description

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 [▶ 219] 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*. *udiNum* indicates the number of requested units.
The operating hours of the units are entered at inputs *udiActvTi01* to *udiActvTi04*. 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*.

### Inputs/outputs

### VAR_INPUT

```
bEn          : BOOL;
udiNum       : UDINT;
bChg         : BOOL;
bEn01        : BOOL;
bEn02        : BOOL;
bEn03        : BOOL;
bEn04        : BOOL;
udiActvTi01  : UDINT;
udiActvTi02  : UDINT;
udiActvTi03  : UDINT;
udiActvTi04  : UDINT;
```

**bEn:** enable of the function block

**udiMyNum:** number of aggregates

**bChg:** force sequence change

**bEn01...bEn04:** enable aggregate 1...enable aggregate 4

**udiActvTi01...udiActvTi04:** operating hours aggregate 1...operating hours aggregate 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;
udiErrId     : UDINT;
udiErrArg    : UDINT;
```

**bQ01...bQ04:** switches aggregate 1…4

**udiNextOn:** number of the aggregate that is switched on next.

**udiNextOff:** number of the aggregate that is switched off 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.

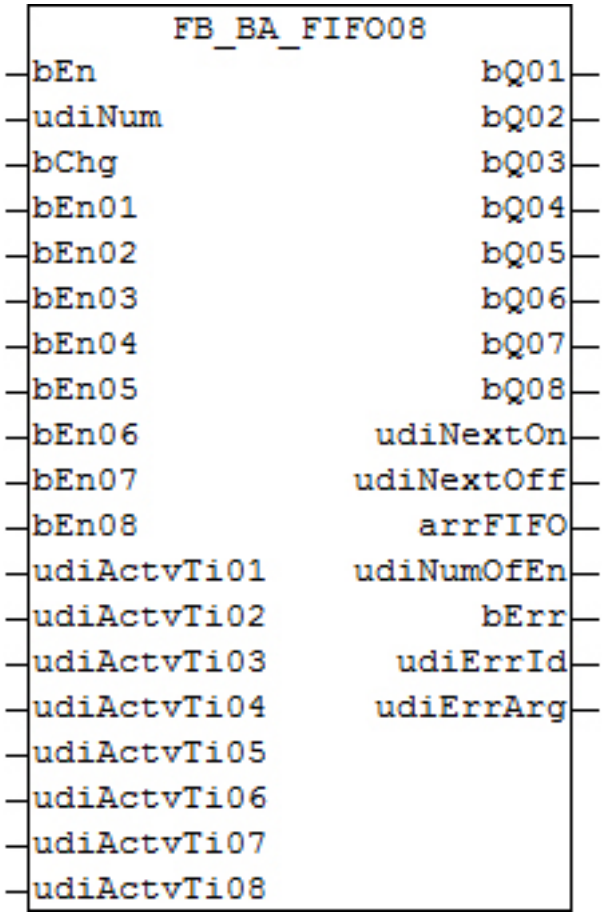
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.88 FB_BA_FIFO08

Sequential control of up to eight aggregates




### Functional description

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 [▶ 218] 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, which are enabled at inputs *bEn01*..*bEn08*. *udiNum* indicates the number of requested units.
The operating hours of the units are entered at inputs *udiActvTi01* to *udiActvTi08*. 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*.

### Inputs/outputs

#### 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 : UDINT;
udiActvTi02 : UDINT;
udiActvTi03 : UDINT;
udiActvTi04 : UDINT;
udiActvTi05 : UDINT;
udiActvTi06 : UDINT;
udiActvTi07 : UDINT;
udiActvTi08 : UDINT;
```

**bEn:** enable of the function block

**udiMyNum:** number of aggregates

**bChg:** force sequence change

**bEn01...bEn08:** enable aggregate 1...enable aggregate 8

**udiActvTi01...udiActvTi08:** operating hours aggregate 1...operating hours aggregate 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;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**bQ01...bQ08:** switches aggregate 1…8

**udiNextOn:** number of the aggregate that is switched on next.

**udiNextOff:** number of the aggregate that is switched off 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.

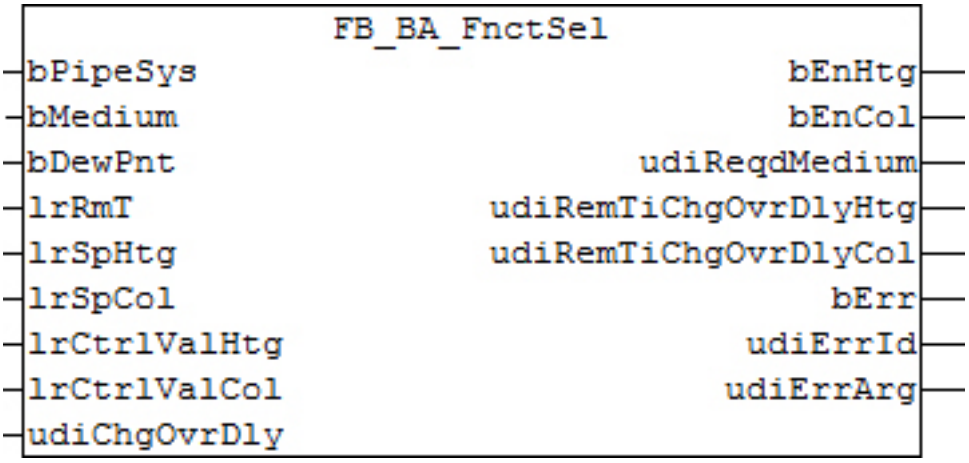
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.89 FB_BA_FnctSel

Automatic selection of heating or cooling controller




### Functional description

The function block is used for room heating/cooling controller activation as part of an air-conditioning plant. 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 any one 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 *uiChgOvrDly* [s]. In other words: The heating cannot be enabled until the cooling enable state *bEnCol* for *uiChgOvrDly* is FALSE. In addition to the elapsing of this switching time, the system checks that the output from controller to be switched off is 0.0. This is based on feedback at the inputs *lrCtrlValHtg* and *lrCtrlValCol*. 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 *lrRmT* and the set values *lrSpHtg* for heating and *lrSpCol* 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 *uiChgOvrDly* [s] (see two-pipe network). In addition to the elapsing of this switching time, the system checks that the output from controller to be switched off is 0.0. This is based on feedback at the inputs *lrCtrlValHtg* and *lrCtrlValCol*. In this way, a drastic change from heating to cooling and vice versa is avoided, if the switching 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. Since this is a safety function, the respective input *bDewPnt* is configured based on the quiescent current principle, i.e. inverted: *bDewPnt* = FALSE: Cooling controller is locked.

### 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 (lrCtrlValCol) | Heating controller output = 0 (lrCtrlValHtg) |
| Room temperature (lrRmT) < heating setpoint (lrSpHtg) | Room temperature (lrRmT) > cooling setpoint (lrSpCol) |
| Cooling controller enable (bEnCol) is FALSE over at least the switching time udiChgOvrDel [s] | Heating controller enable (bEnHtg) is FALSE over at least the switching time udiChgOvrDel [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 sensor (quiescent current principle) 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 (*lrCtrlValHtg*/ *lrCtrlValCol*). 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 sensor was triggered (*bDewPnt*=FALSE) in cooling mode (two- or four-pipe system)

In these cases the heating or cooling enable states are cancelled, 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 | rTRm > SpHtg AND rRTm < SpCol |
| 2 | Heating medium is requested | rTRm < SpHtg |
| 3 | Cooling medium is requested | rTRm > SpCol |

### 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, *udiErrId*=gBA_WarnIdFnctSel); the function block does not interrupt its cycle.

### Inputs/outputs

#### VAR_INPUT

```
bPipeSys        : BOOL;
bMedium         : BOOL;
bDewPnt         : BOOL;
lrRmT           : LREAL;
lrSpHtg         : LREAL;
lrSpCol         : LREAL;
```

```
lrCtrlValHtg   : LREAL;
lrCtrlValCol   : LREAL;
udiChgOvrDel   : UINT;
```

**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 sensor based on **quiescent current principle** (negative logic): if *bDewPnt* = FALSE, the cooling controller is locked.

**lrTRm:** room temperature

**lrSpHtg:** heating setpoint value

**lrSpCol:** cooling setpoint

**lrCtrlValHtg:** current output value of the heating controller. Used internally as switching criterion from heating to cooling: *lrCtrlValHtg* must be 0.

**fCtrlValCol:** current output value of the cooling controller. Used internally as switching criterion from cooling to heating: *lrCtrlValCol* must be 0.

**udiChgOvrDel:** switchover delay [s] from heating to cooling or vice versa

### VAR_OUTPUT

```
bEnHtg               : BOOL;
bEnCol               : BOOL;
udiReqdMedium        : UDINT;
udiRemTiChgOvrDlyHtg : UDINT;
udiRemTiChgOvrDlyCol : UDINT;
bErr                 : BOOL;
udiErrId             : UDINT;
udiErrArg            : UDINT;
```

**bEnHtg:** enable of the heating controller

**bEnCol:** enable of the cooling controller

**udiReqdMedium:**

| udiReqdMedium | Medium | Room temperature |
|---|---|---|
| 1 | No medium is requested | rTRm > SpHtg AND rRTm < SpCol |
| 2 | Heating medium is requested | rTRm < SpHtg |
| 3 | Cooling medium is requested | rTRm > SpCol |

**udiRemTiChgOvrDlyHtg:** countdown [s] switchover delay from cooling to heating

**udiRemTiChgOvrDlyCol:** countdown [s] switchover delay from heating to cooling

**bErr:** in case of a fault, e.g. if warning states are active, this output is set to TRUE.

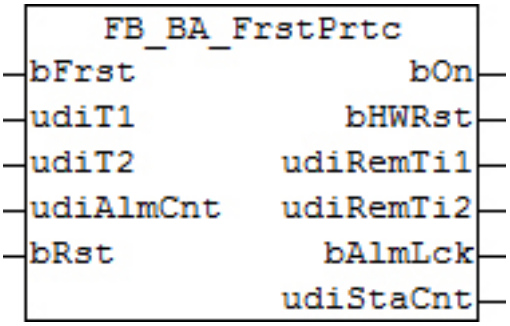
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2244 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.90 FB_BA_FrstPrtc

Monitoring of frost alarm and emergency heating with two time monitors: T1 alarm ceases and T2 alarm does not recur.




**Functional description**

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 a frost risk is detected, the output *bOn* is set and the timer T1 is started. If the frost risk persists (*bFrst*=TRUE) once the time *udiT1* (seconds) has elapsed, *bOn* remains set. It can only be reset at input *bRst*.

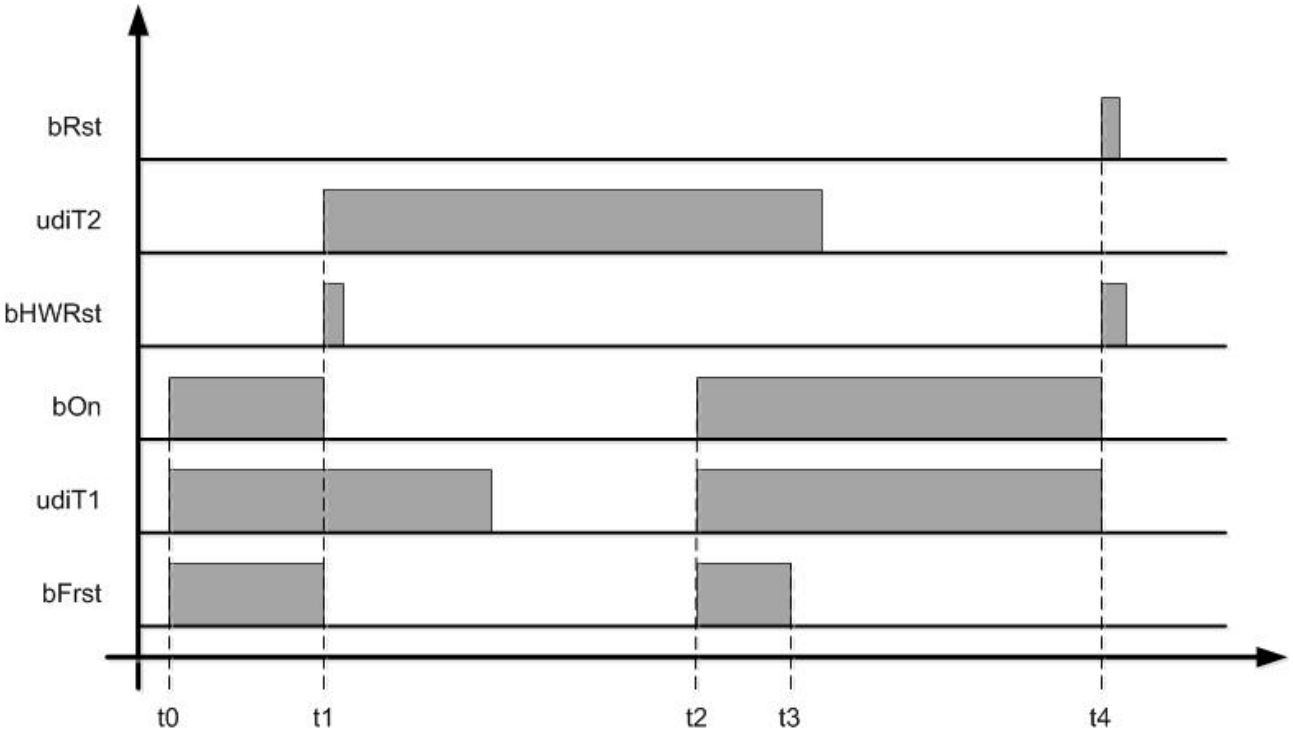
If the frost alarm ceases due to activation of the heating coil within the time *udiT1* (*bFrst*=FALSE), the plant automatically restarts. For the plant restart *bOn* becomes FALSE, and at output *bHWRst* a pulse for acknowledgement of a latching circuit in the control cabinet is issued. Timer T2 is started with the plant restart. If another frost alarm occurs within a period of *udiT2* (seconds), 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*, 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* [s])
t1 = frost alarm off, resetting of *bOn*, output of hardware pulse, start of timer T2 (*udiT2* [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/outputs**

**VAR_INPUT**

```
bFrst     : BOOL;
udiT1     : UDINT;
udiT2     : UDINT;
udiAlmCnt : UDINT;
bRst      : BOOL;
```

**bFrst:** connection for frost events on the air and water side

**udiT1:** timer for restart delays [s]

**udiT2:** timer monitoring time [s]

**udiAlmCnt:** maximum number of automatic plant restarts without reset. The maximum possible entry is 4; it is internal limited.

**bRst:** resetting and acknowledgement of the frost alarm

**VAR_OUTPUT**

```
bOn       : BOOL;
bHWRst    : BOOL;
udiRemTi1 : UDINT;
udiRemTi2 : UDINT;
bAlmLck   : BOOL;
udiStaCnt : UDINT;
```

**bOn:** frost alarm active

**bHWRst:** output of a pulse for acknowledgement of the frost protection hardware

**udiRemTi1:** time remaining to plant restart after frost alarm

**udiRemTi2:** remaining monitoring time

**bAlmLck:** alarm lock - stored alarm

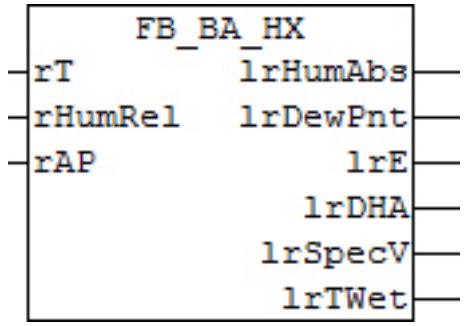
**udiStaCnt:** status counter – current number of unacknowledged false starts

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.91 FB_BA_HX

Enthalpy calculation




### Functional description

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.

### Inputs/outputs

#### 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]

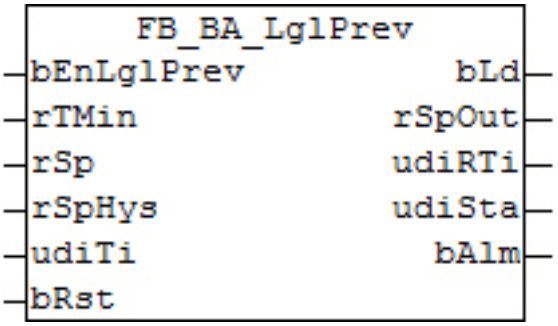
**lrTWet:** wet bulb temperature [°C]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

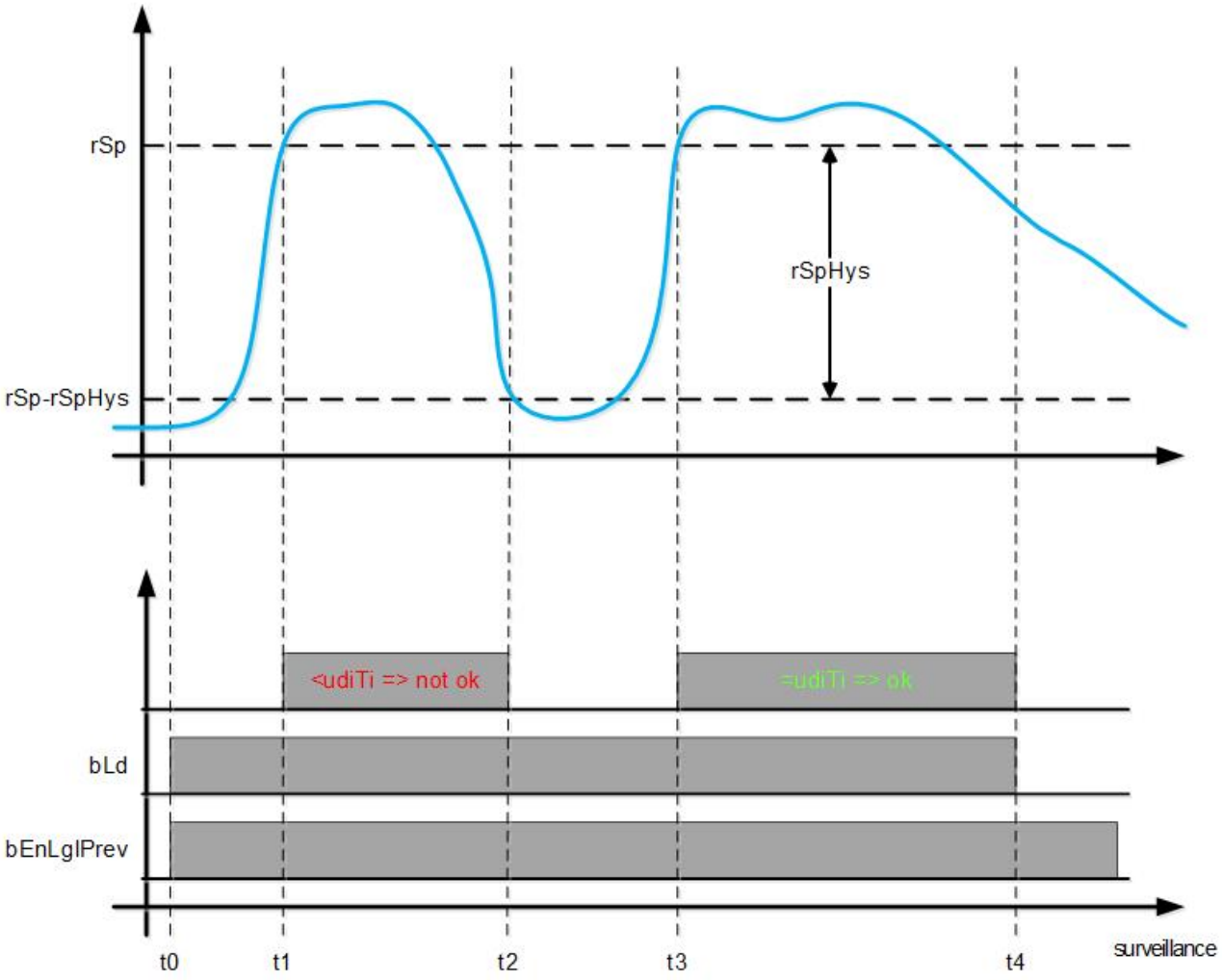
## 8.2.92 FB_BA_LglPrev

Legionella protection




### Functional description

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 two temperature sensors, a minimum selection feature for both sensors must be connected at *rTMin*.
If *rTMin* exceeds the value of *rSp,* a monitoring timer is started with a time of *udiTi* [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 hot 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* 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 the input *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*; the disinfection was successful. *bLd* is reset, and the hot water tank switches back to normal operation.

### Inputs/outputs

### VAR_INPUT

```
bEnLglPrev : BOOL;
rTMin      : REAL;
rSp        : REAL;
rSpHys     : REAL;
udiTi      : 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:** 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.

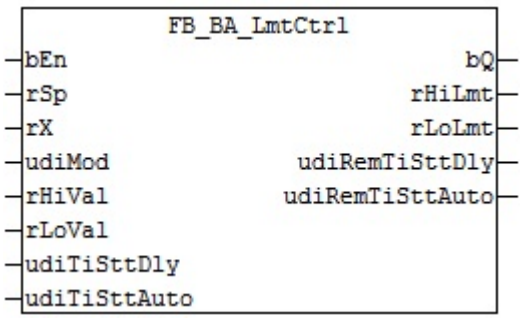
**bAlm:** the temperature setpoint was not reached consistently over the interval *udiTime*, so that adequate disinfection is not guaranteed.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.93 FB_BA_LmtCtrl

Function block for determining the limits and its enable status



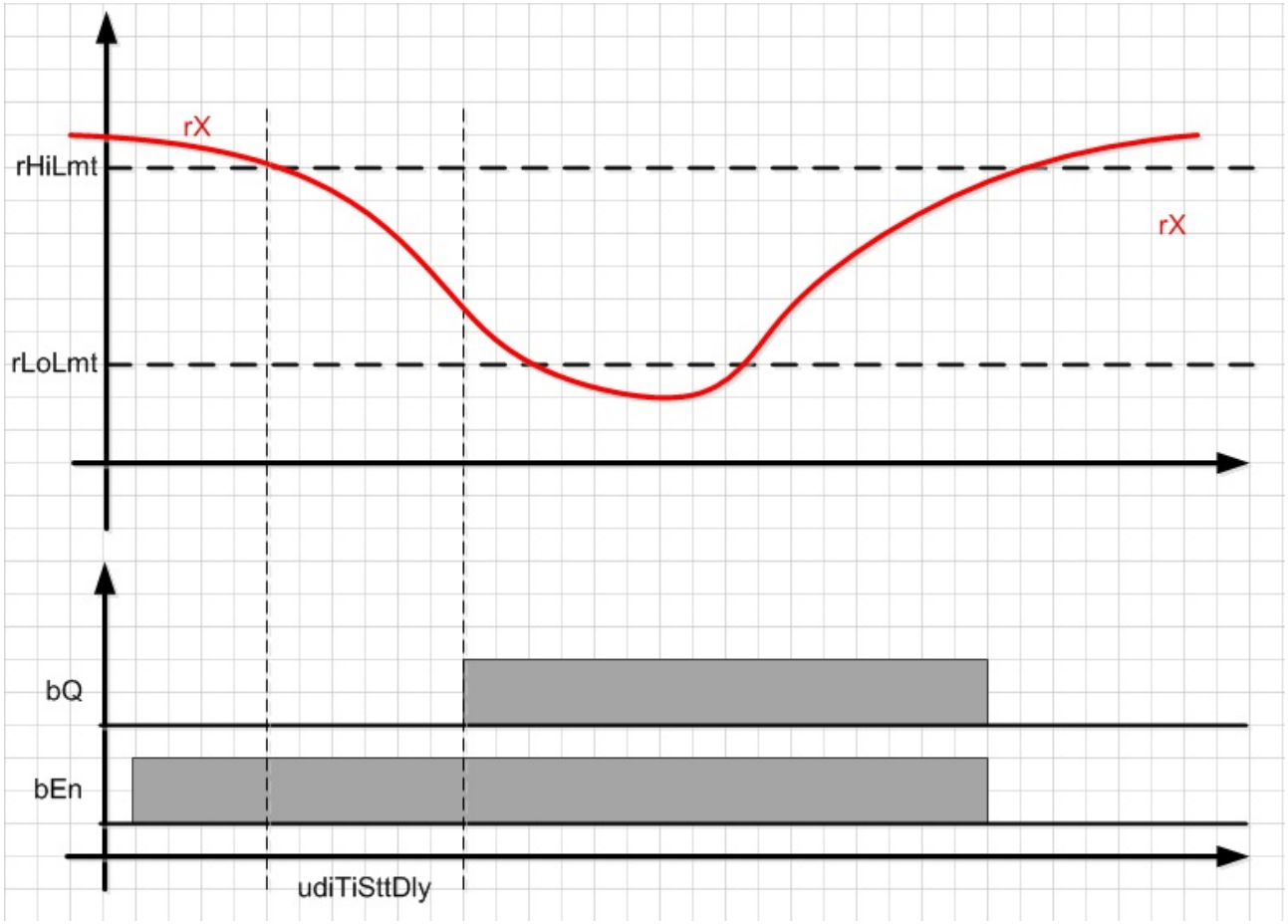
### Functional description

The function block is used to determine limit values (*rHiLmt* / *rLoLmt*) depending on the set operating mode and its enable (*bQ*).
The function block can be used for the following conditions, e.g. during a plant shutdown, at the moment of startup and until a plant is in a controlled state. During these phases, reporting of a loop object, for example, should be suppressed, in order to avoid sending of incorrect messages to the MCL (management and control level). Reporting of the loop object should not be activated until the air-conditioning system is fully running and the control is stable.

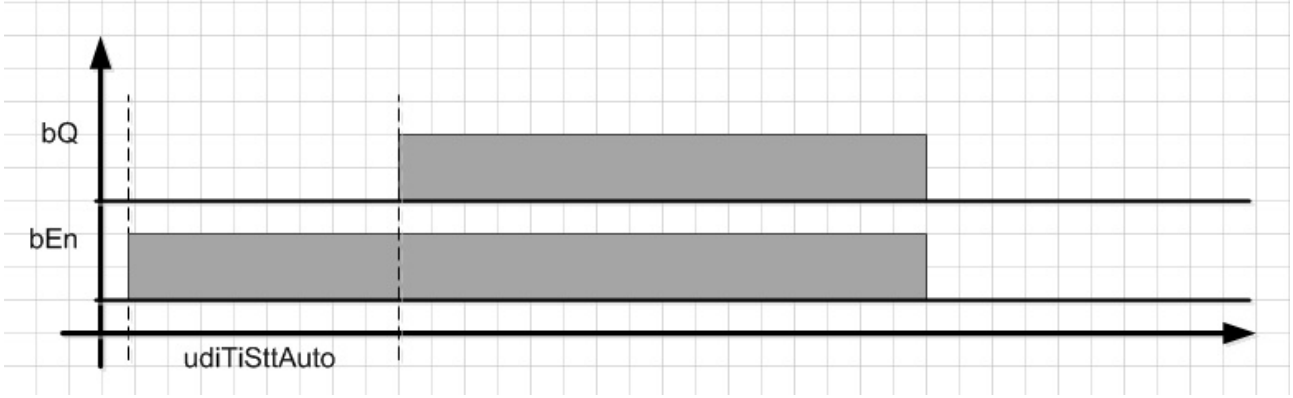
**It is important that only via = FALSE can be reset.** *bQ bEn*

### Diagram 1

Diagram 1




### Diagram 2



### Inputs/outputs

### VAR_INPUT

```
bEn             : BOOL;
rSp             : REAL;
rX              : REAL;
udiOpMod        : UDINT;
rHiVal          : REAL;
rLoVal          : REAL;
udiTiSttDly     : UDINT;[s]
udiTiSttAuto    : UDINT;[s]
```

**bEn:** enable function block. If *bQ* = TRUE, the output can only be reset, if *bEn* is set to FALSE.

**rSp:** setpoint

**rX:** actual value

**udiOpMod:** operating mode specification. You can choose between 3 modes. This results in

udiOpMod = 1 = fixed: *rHiLmt* = *rHiVal* and *rLoLmt* := *rLoVal.***It is important that***rLoVal***must not be greater than or equal to***rHiVal***. If this is the case, then***rHiLmt***=***rHiVal***and***rLoLmt***:=***rHiVal***- 0.05**

udiOpMod = 2 = sliding absolute: the absolute value for the two input variables *rHiVal* / *rLoVal* is determined via the ABS function. *rHiLmt* = *rSp* + *rHiVal* and *rLoLmt* = *rSp* - *rLoVal*

udiOpMod = 3 = sliding percent: the function ABS is used to determine the absolute value for the two input variables *rHiVal* / *rLoVal*. The formulas for determining the limit values are: *rHiLmt* = *rSp* + ABS((*rSp* x *rHiVal*) / 100)); *rLoHiLmt* = *rSp* - ABS((*rSp* x *rLoVal*) / 100))

**rHiVal:** upper value for calculating the upper limit *rHiLmt*. The calculation of the upper limit value depends on the operating mode *udiOpMod*.

**rLoVal:** lower value for calculating the lower limit *rLoLmt*. The calculation of the lower limit value depends on the operating mode *udiOpMod***.**

**udiTiSttDly:** Start delay for *bQ*, if function is met, see diagram 1 [▶ 231].

**udiTiSttAuto:** automatic start delay. If *bEn* = TRUE, *bQ* becomes TRUE once the automatic delay has elapsed, see diagram 2 [▶ 231].

### VAR_OUTPUT

```
bQ                : BOOL;
rHiLmt            : REAL;
rLoLmt            : REAL;
udiRemTiSttDly    : UDINT;
udiRemTiSttAuto   : UDINT;
```

**bQ:** limit value monitoring is enabled. *bQ* can only be reset via *bEn* = FALSE.

**rHiLmt:** upper limit value for enabling limit value monitoring. The value depends on the operating mode *udiOpMod*

**rLoLmt :** lower limit value for enabling limit value monitoring. The value depends on the operating mode *udiOpMod*

**udiRemTiSttDly:** start delay countdown *udiTiSttDly*

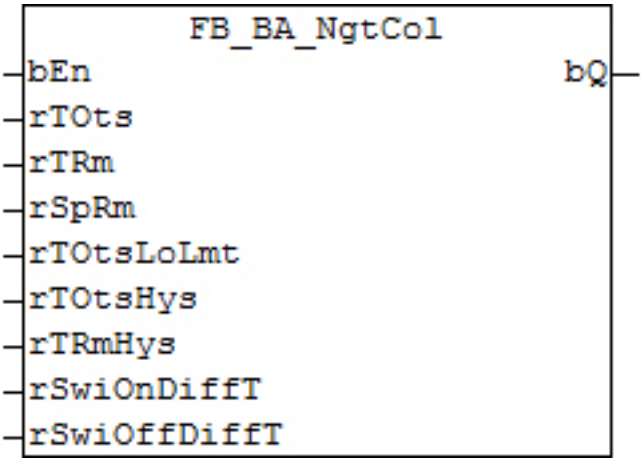
**udiRemTiSttAuto:** automatic start delay countdown *udiTiStt*Auto

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.94 FB_BA_NgtCol

Summer night cooling




### Functional description

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 parameterising the FB_BA_NgtCol function block. The 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 such that it is not worth switching on 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.*

### Inputs/outputs

#### VAR_INPUT

```
bEn           : BOOL;
rTOts         : REAL;
rTRm          : REAL;
rSpRm         : REAL;
rTOtsLoLmt    : REAL;
rTOtsHys      : REAL;
rTRmHys       : REAL;
rSwiOnDiffT   : REAL;
rSwiOffDiffT  : REAL;
```

**bEn:** enable of the 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 oscillates 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;
```

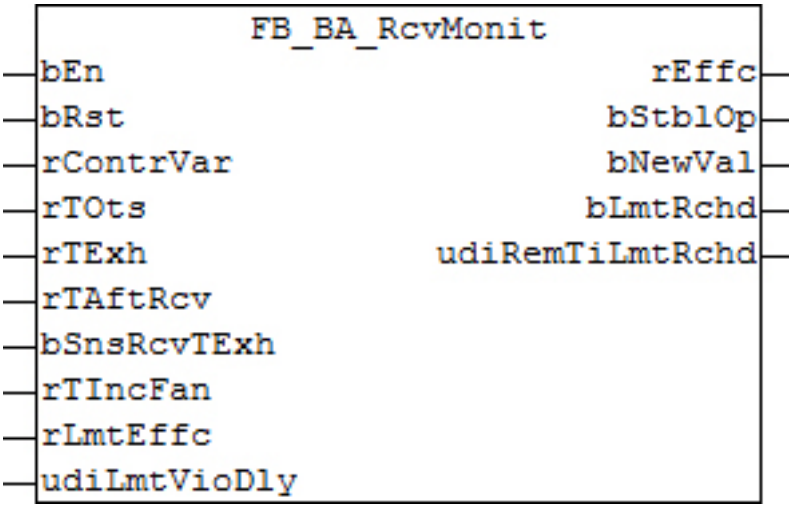
**bQ:** summer night cooling on

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.95 FB_BA_RcvMonit

Function block for calculating the efficiency of an energy recovery system

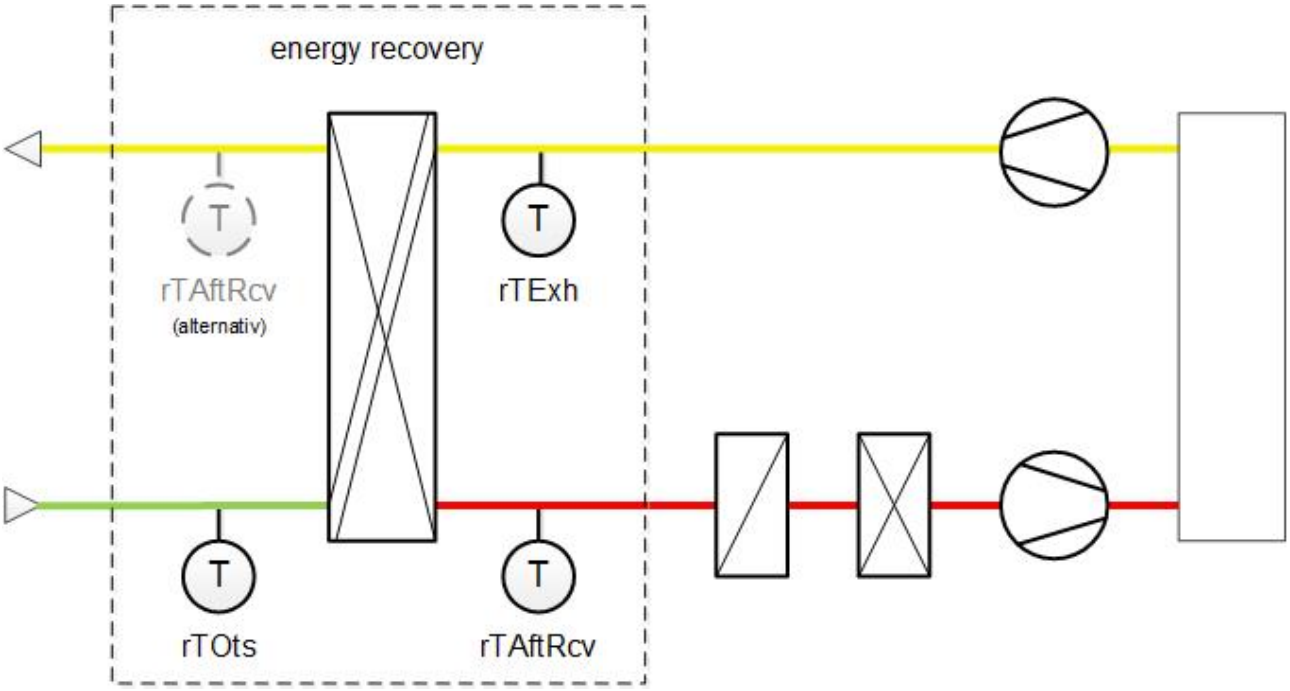



### Functional description

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 six consecutive values. The results are used to assessed 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 *rEffc*. If the state is not "stable", a FALSE signal appears at output rEffc, 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 only operates 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 heat recovery in the **supply air duct** is used to calculate the efficiency.
If, on the other hand, the temperature measurement after heat recovery in the **exhaust air duct** is to be used, a TRUE must be applied to *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 the input *rContrVar*.
The limit violation message can be delayed by an entry at *udiLmtVioDly*: if the two criteria, violation and override, are met for longer than *udiLmtVioDly* [s], this is indicated with a TRUE signal at *bLmtRchd*. The

occurrence can be tracked at the countdown output *udiRemTiLmtRchd*.
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.

Such a warning message can only occur, if the plant is in stable operation (*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 quantified via the parameter rTIncFan. Internally, the measured outlet air temperature is then reduced by this value.

### Inputs/outputs

### VAR_INPUT

```
bEn           : BOOL;
bRst          : BOOL;
rContrVar     : REAL;
rTOts         : REAL;
rTExh         : REAL;
rTAftRcv      : REAL;
bSnsRcvTExh   : BOOL;
rTIncFan      : REAL;
rLmtEffc      : REAL;
udiLmtVioDly  : UDINT;
```

**bEn:** enable of the function block

**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 the supply air duct (SupplyAir) - TRUE -> in the exhaust air duct (ExhaustAir).

**rTIncFan:** temperature increase due to fan

**rLmtEffc:** limit value efficiency

**udiLmtVioDly:** limit violation delay [s]

### VAR_OUTPUT

```
rEffc            : REAL;
bStblOp          : BOOL;
bNewVal          : BOOL;
bLmtRchd         : BOOL;
udiRemTiLmtRchd  : UDINT;
```

**rEffc:** efficiency

**bStblOp:** stable operation

**bNewVal:** output trigger new value *rEffc*

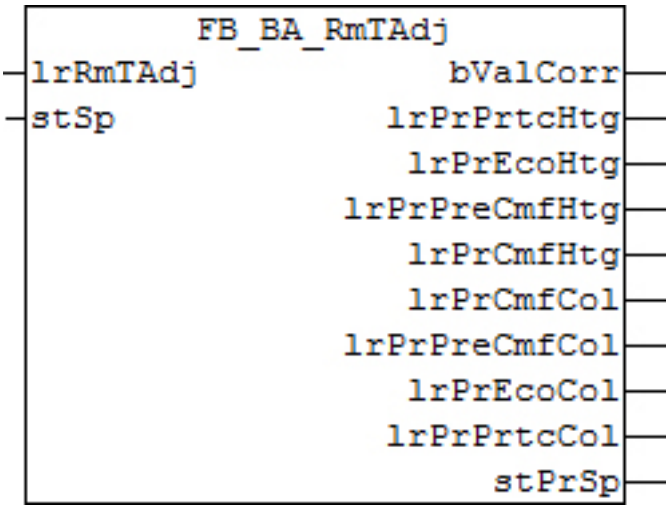
**bLmtRchd:** limit value reached

**udiRemTiLmtRchd:** countdown timer limit value reached

### Requirements

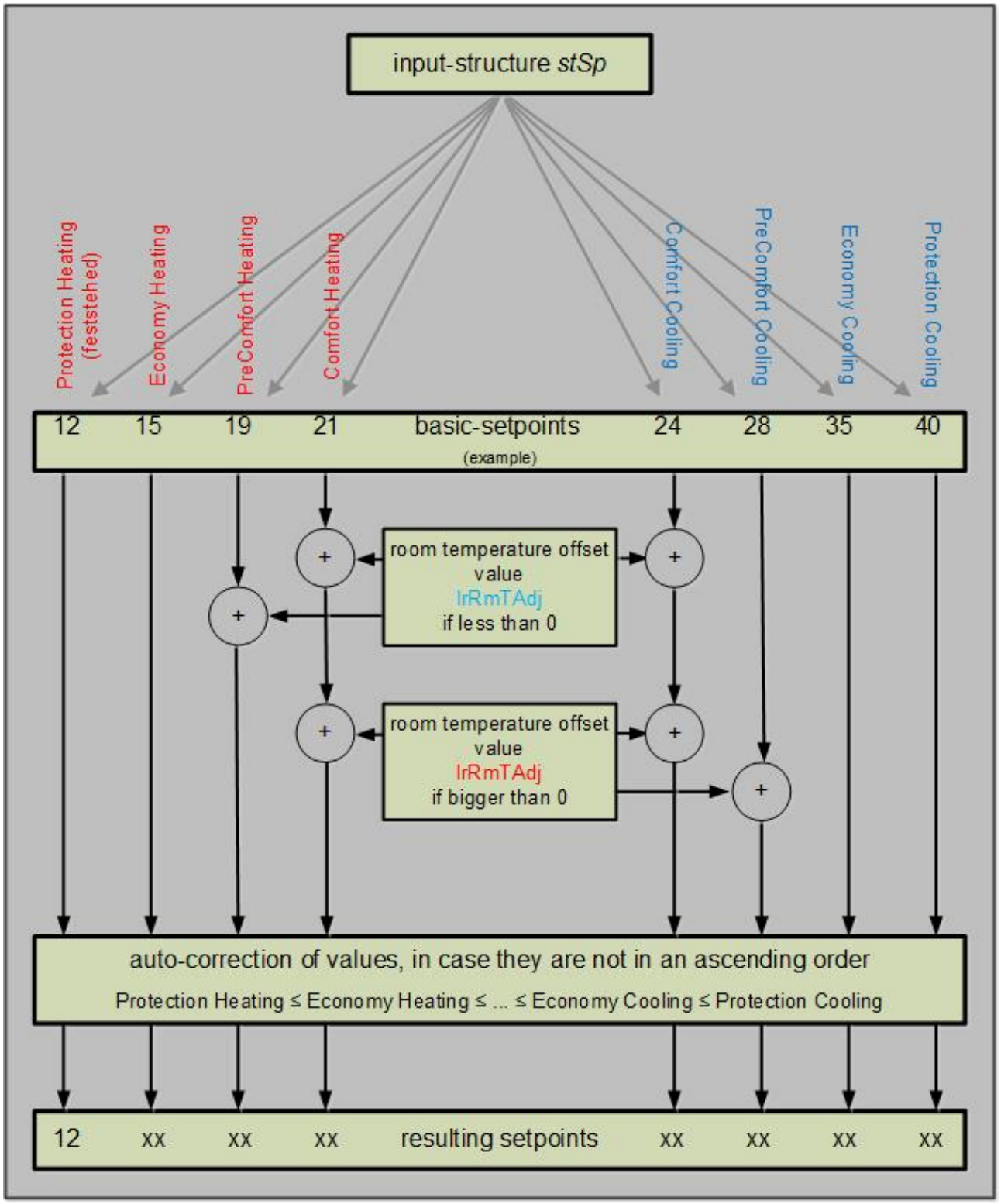
| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.96 FB_BA_RmTAdj




### Functional description

The function block *FB_BA_RmTAdj* is used to adjust the room temperature setpoint from the user in the room. 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 temperature heating is desired: the Comfort Heating value is increased by the value *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. 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

The auto-correction is based on the principle that, starting from the value for Economy Heating, the system checks whether this value is smaller than the next lower parameter, in this case 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/outputs

#### VAR_INPUT

```
lrRmTAdj : LREAL;
stSp     : ST_BA_SpRmT;
```

**lrRmTAdj:** Room temperature offset value

**stSp :** Input structure [▶ 330] for the setpoints

#### 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

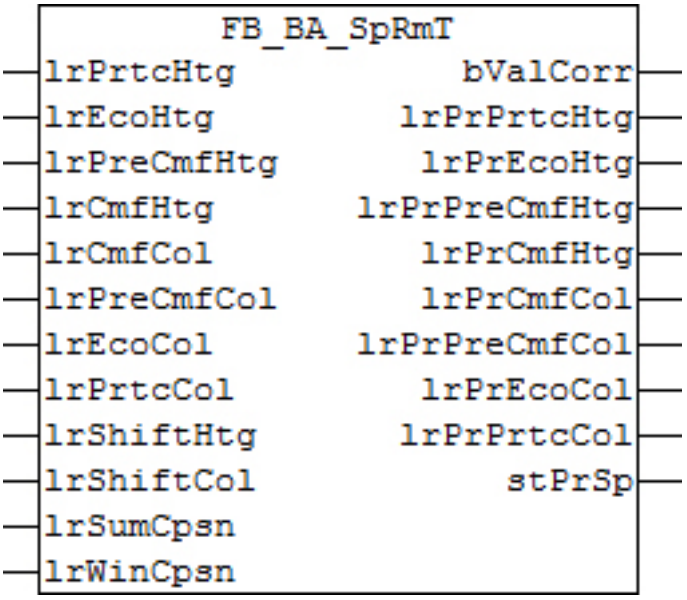
**stPrSp:** consolidated output of the resulting values in a structure

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

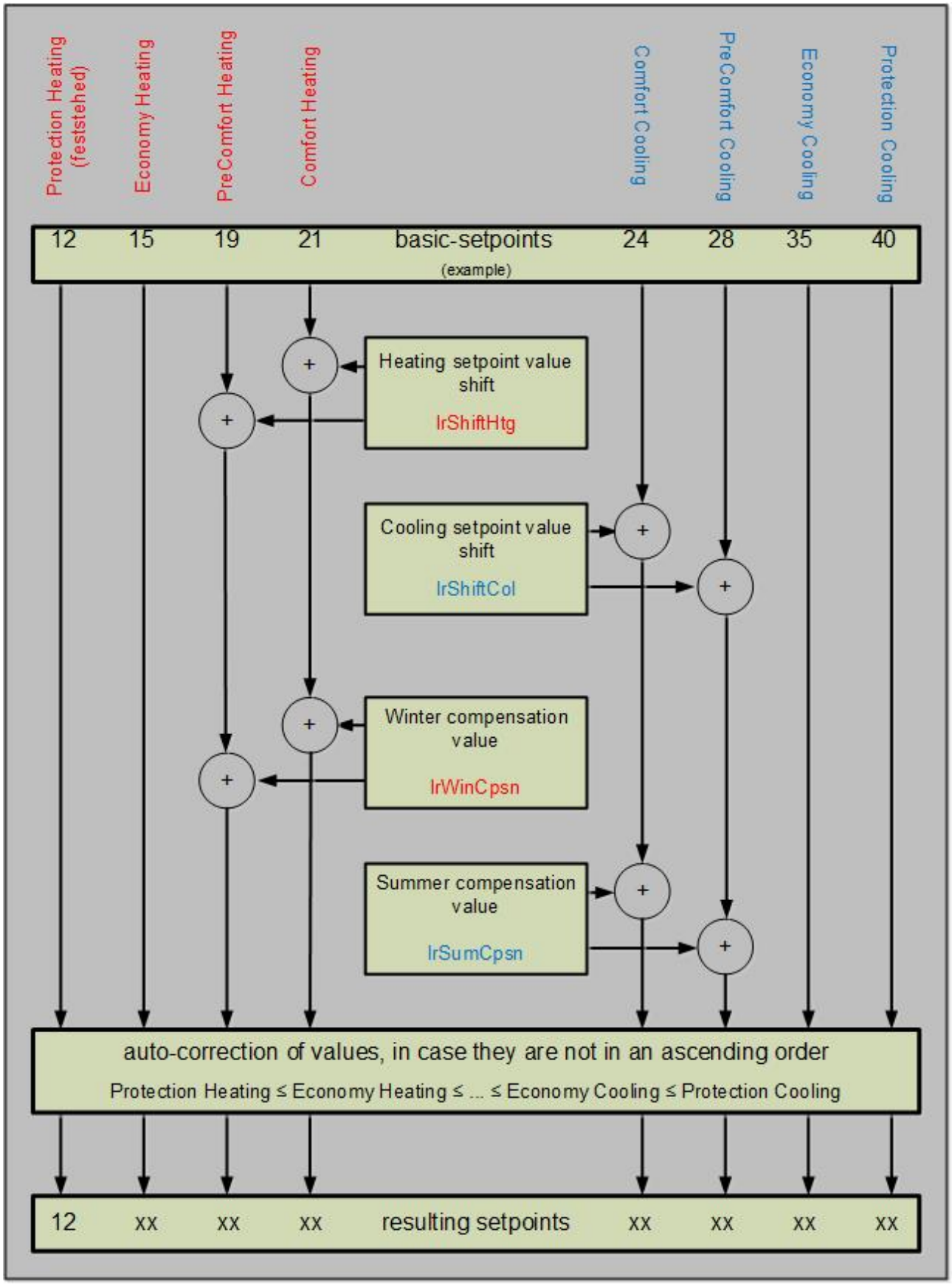
## 8.2.97 FB_BA_SpRmT

Formation of room temperature setpoints




### Functional description

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 behaviour of the function block; the entered values should be regarded as examples:




The value *lrShiftHeating* is applied to the Comfort and Precomfort values for the heating mode as central setpoint shift. In addition, winter compensation *lrWinCpsn* is applied.
Similarly, the following applies for the cooling mode: The value *lrShiftCooling* is applied to the Comfort and Precomfort values. In addition, the summer compensation value *lrSumCpsn* 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

The auto-correction is based on the principle that, starting from the value for Economy Heating, the system checks whether this value is smaller than the next lower parameter, in this case 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/outputs

### VAR_INPUT

```
lrPrtcHtg   : LREAL;
lrEcoHtg    : LREAL;
lrPreCmfHtg : LREAL;
lrCmfHtg    : LREAL;
lrCmfCol    : LREAL;
lrPreCmfCol : LREAL;
lrEcoCol    : LREAL;
lrPrtcCol   : LREAL;
lrShiftHtg  : LREAL;
lrShiftCol  : LREAL;
lrSumCpsn   : LREAL;
lrWrWinCpsn : LREAL;
```

**bValCorr:** autocorrection: at least one of the resulting setpoints was adjusted such that the values continue to monotonically increase.

**lrPrtcHtg :** basic Protection Heating setpoint

**lrEcoHtg:** basic Economy Heating setpoint

**lrPreCmfHtg:** basic PreComfort Heating setpoint

**lrCmfHtg:** basic Comfort Heating setpoint

**lrCmfCol:** basic Comfort Cooling setpoint

**lrPreCmfCol:** basic PreComfort Cooling setpoint

**lrEcoCol:** basic Economy Cooling setpoint

**lrPrtcCol :** basic Protection Cooling setpoint

**lrShiftHtg :** setpoint value shift heating

**lrShiftCol:** setpoint value shift cooling

**lrSumCpsn:** value summer compensation

**lrWinCpsn:** value winter compensation

### VAR_OUTPUT

```
lrPrPrtcHtg    : LREAL;
lrPrEcoHtg     : LREAL;
lrPrPreCmfHtg  : LREAL;
lrPrCmfHtg     : LREAL;
lrPrPrtcCol    : LREAL;
lrPrEcoCol     : LREAL;
lrPrPreCmfCol  : LREAL;
lrPrCmfCol     : LREAL;
stPrSp         : ST_BA_SpRmT;
```

**lrPrPrtcHtg :** resulting Protection Heating setpoint

**lrPrEcoHtg:** resulting Economy Heating setpoint

**lrPrPreCmfHtg:** resulting PreComfort Heating setpoint

**lrPrCmfHtg:** resulting Comfort Heating setpoint

**lrPrCmfCol:** resulting Comfort Cooling setpoint

**lrPrPreCmfCol:** resulting PreComfort Cooling setpoint

**lrPrEcoCol:** resulting Economy Cooling setpoint

**lrPrPrtcCol :** resulting Protection Cooling setpoint

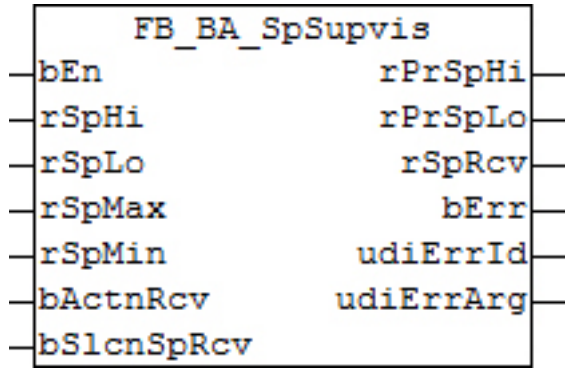
**stPrSp:** consolidated output of the resulting values in a structure [▶ 330].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.98 FB_BA_SpSupvis

Function block for processing and checking the lower and upper setpoint of a supply air humidity or temperature control.




### Functional description

#### Checks and limits for the setpoints

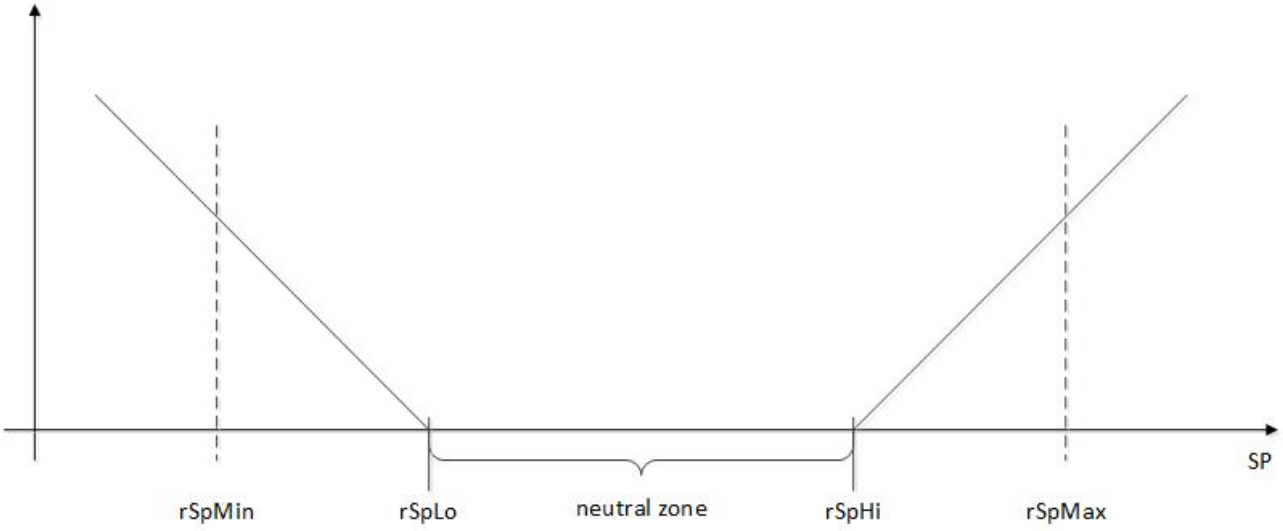
The function block checks and limits the setpoints. The following two tables show which parameters are checked and what the response is in the event of an error.

| Checking | Action |
|---|---|
| rSpLo > rSpHi | last valid values of rSpLo and rSpHi are used |

| Checking | Action |
|---|---|
| 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)<br><br>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 possibly limited setpoints are output at the function block output as *rPrSpHi* and *rPrSpLo* (Present Setpoint).

### Setpoint for heat recovery

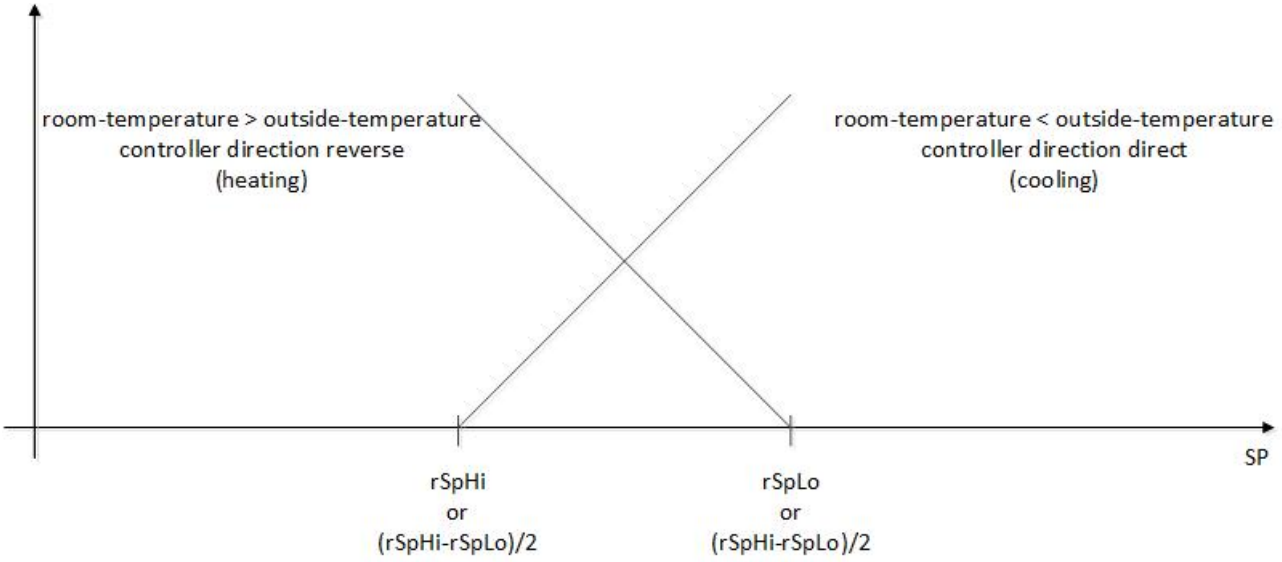
For heat recovery, the setpoint *rSpRcv* is calculated either from the average of the upper and lower setpoints, *rSoHi* and *rSpLo*, or as a function of the control direction of the heat recovery system. The method is defined by the input variable *bSlcnSpRcv*:

| b SlcnSpRcv | rSpRcv |
|---|---|
| TRUE | Average of rSpLo and rSpHi |
| 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 | rSpRcv |
|---|---|---|
| TRUE | direct (cooling) | lrSpHi |
| FALSE | indirect (heating) | lrSpLo |

### Heat recovery




### Inputs/outputs

#### 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:** input value of the upper setpoint to be checked

**rSpLo:** input value of the lower setpoint to be checked

**rSpMax:** maximum setpoint

**rSpMin:** minimum setpoint

**bActnRcv:** control direction of the downstream heat recovery

**bSlcnSpRcv:** setpoint selection of the downstream heat recovery

#### VAR_OUTPUT

```
rPrSpHi    : REAL;
rPrSpLo    : REAL;
rSpRcv     : REAL;
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**rPrSpHi:** output value of the upper setpoint

**rPrSpLo:** output value of 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.

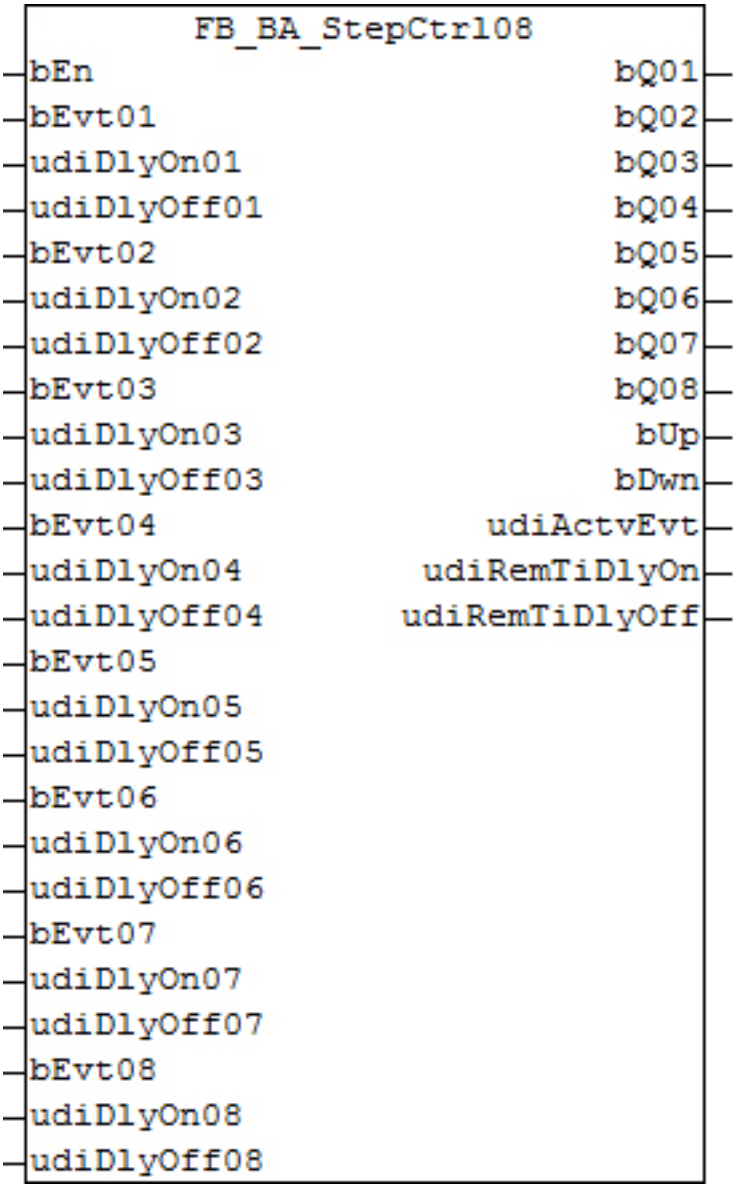
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.99 FB_BA_StepCtrl08

Switching step function block, 8x



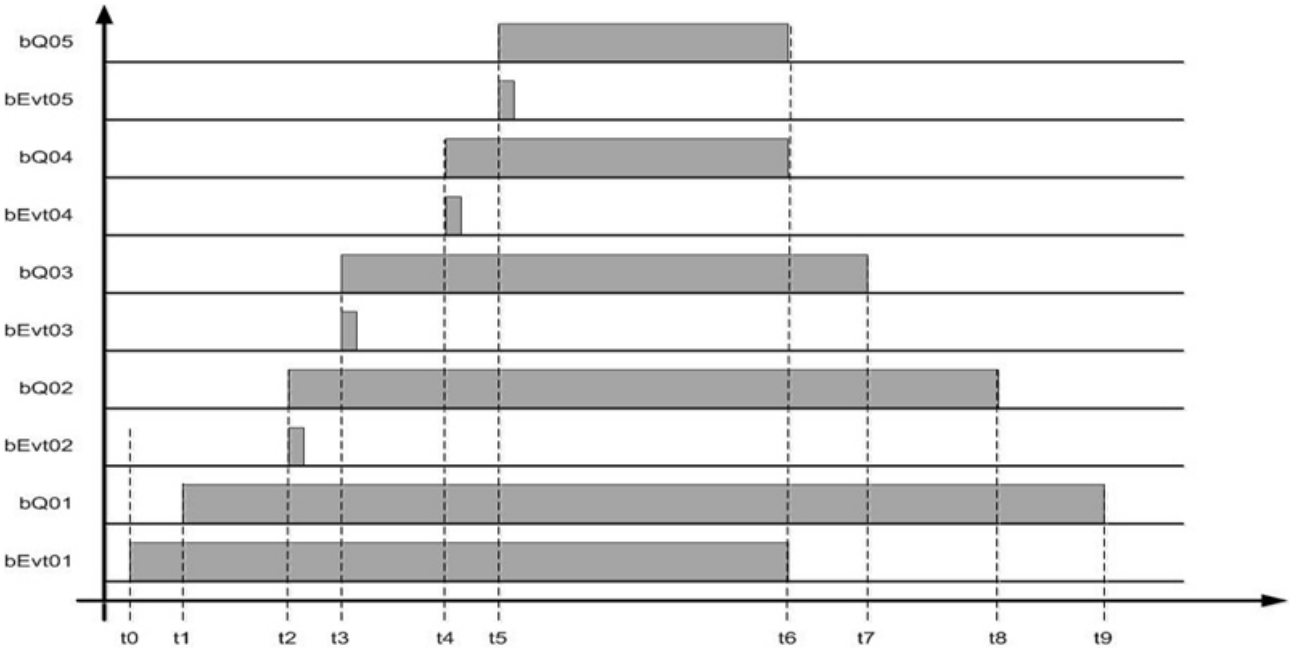

### Functional description

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* 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* to *udiDlyOn08*. If *bEvt01* becomes FALSE once the control chain is up and running, the control sequence switches back in reverse order. The switch-off of the outputs is delayed through the timers *udiDlyOff01* to *udiDlyOff08*.

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 *udiStep* which shows the output *udiActvEvt*+1, is available for application with a BACnet multistate output object, which cannot show "0".

*udiRemTiDlyOn* indicates the time remaining to the next step during up-switching of the control chain. *udiRemTiDlyOff* indicates the time remaining to the next lower step during down-switching of the control chain.

**Example**




t0 step sequence switch-on
t1 step 1 switch-on *udiDlyOn01* = t1 - t0
t2 event enable step 2, switch-on step 2, *udiDlyOn02* = 0
t3 event enable step 3, switch-on step 3, *udiDlyOn03* = 0
t4 event enable step 4, switch-on step 4, *udiDlyOn04* = 0
t5 event enable step 5, switch-on step 5, *udiDlyOn05* = 0
t6 step sequence switch-off, switch-off step 5, switch-off step 4; *udiDlyOff05* = 0, *udiDlyOff04* = 0
t7 switch-off step 3, *udiDlyOff03* = t7 -t6
t8 switch-off step 2, *udiDlyOff02* = t8 -t7
t9 switch-off step 1, *udiDlyOff01* = t9 -t8

**Inputs/outputs**

**VAR_INPUT**

```
bEn          : BOOL;
bEvt01       : BOOL;
udiDlyOn01   : UDINT;
udiDlyOff01  : UDINT;
bEvt02       : BOOL;
udiDlyOn02   : UDINT;
udiDlyOff02  : UDINT;
bEvt03       : BOOL;
udiDlyOn03   : UDINT;
udiDlyOff03  : UDINT;
bEvt04       : BOOL;
udiDlyOn04   : UDINT;
udiDlyOff04  : UDINT;
bEvt05       : BOOL;
udiDlyOn05   : UDINT;
udiDlyOff05  : UDINT;
bEvt06       : BOOL;
udiDlyOn06   : UDINT;
udiDlyOff06  : UDINT;
bEvt07       : BOOL;
```

```
udiDlyOn07  : UDINT;
udiDlyOff07 : UDINT;
bEvt08      : BOOL;
udiDlyOn08  : UDINT;
udiDlyOff08 : UDINT;
```

**bEn:** enable of the function block

**bEvt01:** control chain switch-on

**udiDlyOn01:** start-up delay for output *bQ01* [s]

**udiDlyOff01:** switch-off delay for output *bQ01* [s]

**bEvt02:** advance command step 2

**udiDlyOn02:** start-up delay for output *bQ02* [s]

**udiDlyOff02:** switch-off delay for output *bQ02* [s]

**bEvt03:** advance command step 3

**udiDlyOn03:** start-up delay for output *bQ03* [s]

**udiDlyOff03:** switch-off delay for output *bQ03* [s]

**bEvt04:** advance command step 4

**udiDlyOn04:** start-up delay for output *bQ04* [s]

**udiDlyOff04:** switch-off delay for output *bQ04* [s]

**bEvt05:** advance command step 5

**udiDlyOn05:** start-up delay for output *bQ05* [s]

**udiDlyOff05:** switch-off delay for output *bQ05* [s]

**bEvt06:** advance command step 6

**udiDlyOn06:** start-up delay for output *bQ06* [s]

**udiDlyOff06:** switch-off delay for output *bQ06* [s]

**bEvt07:** advance command step 7

**udiDlyOn07:** start-up delay for output *bQ07* [s]

**udiDlyOff07:** switch-off delay for output *bQ07* [s]

**bEvt08:** advance command step 8

**udiDlyOn08:** start-up delay for output *bQ08* [s]

**udiDlyOff08:** switch-off delay for output *bQ08* [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  : UDINT;
udiRemTiDlyOff : UDINT;
```

**bQ01:** step 1 on

**bQ02:** step 2 on

**bQ03:** step 3 on

**bQ04:** step 4 on

**bQ05:** step 5 on

**bQ06:** step 6 on

**bQ07:** step 7 on

**bQ08:** step 8 on

**bUp:** control chain is in ascending state

**bDwn:** control chain is in descending state

**udiActvEvt:** active step, display 0...8; "0" represents a non-active step sequence.

**udiRTiDlyOn:** remaining time until switching up to the next step [s]

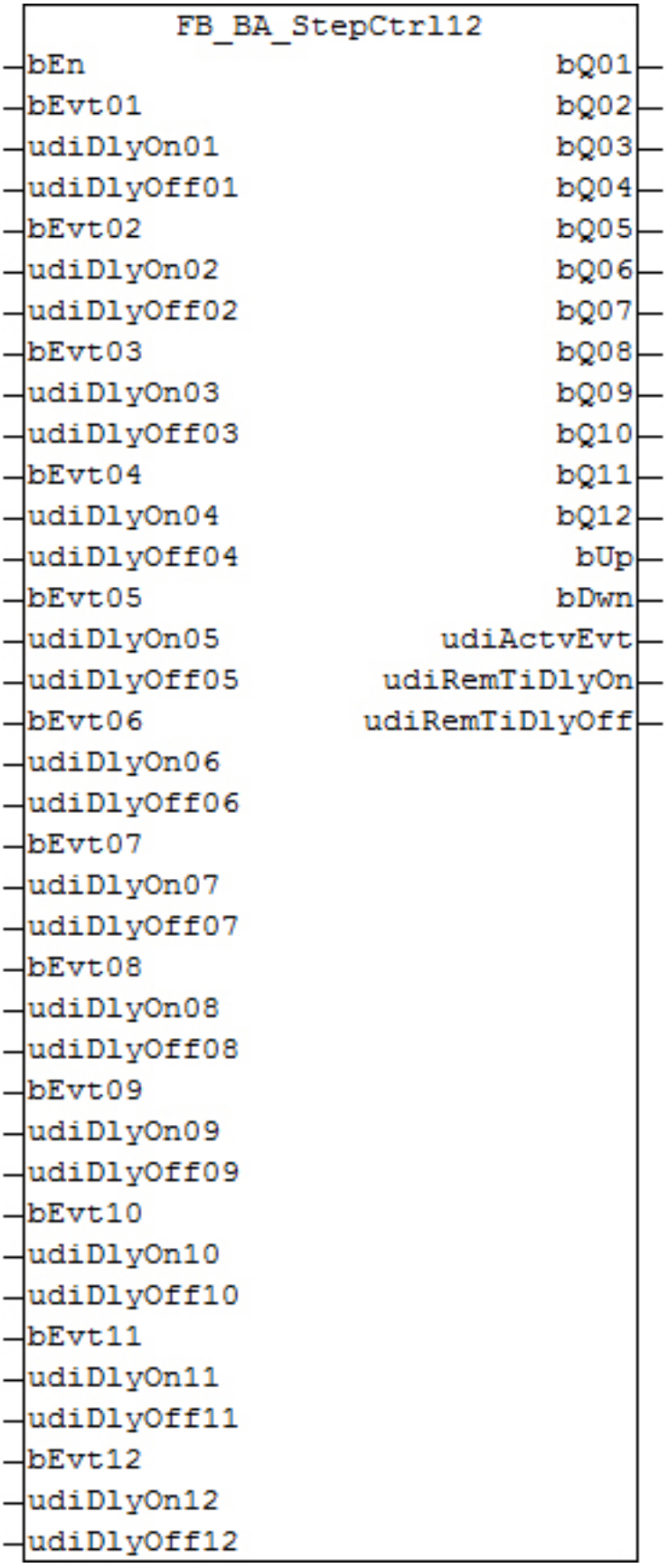
**udiRTiDlyOff:** remaining time until switching down to the previous step [s]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.100 FB_BA_StepCtrl12

Switching step function block, 12x




**Functional description**

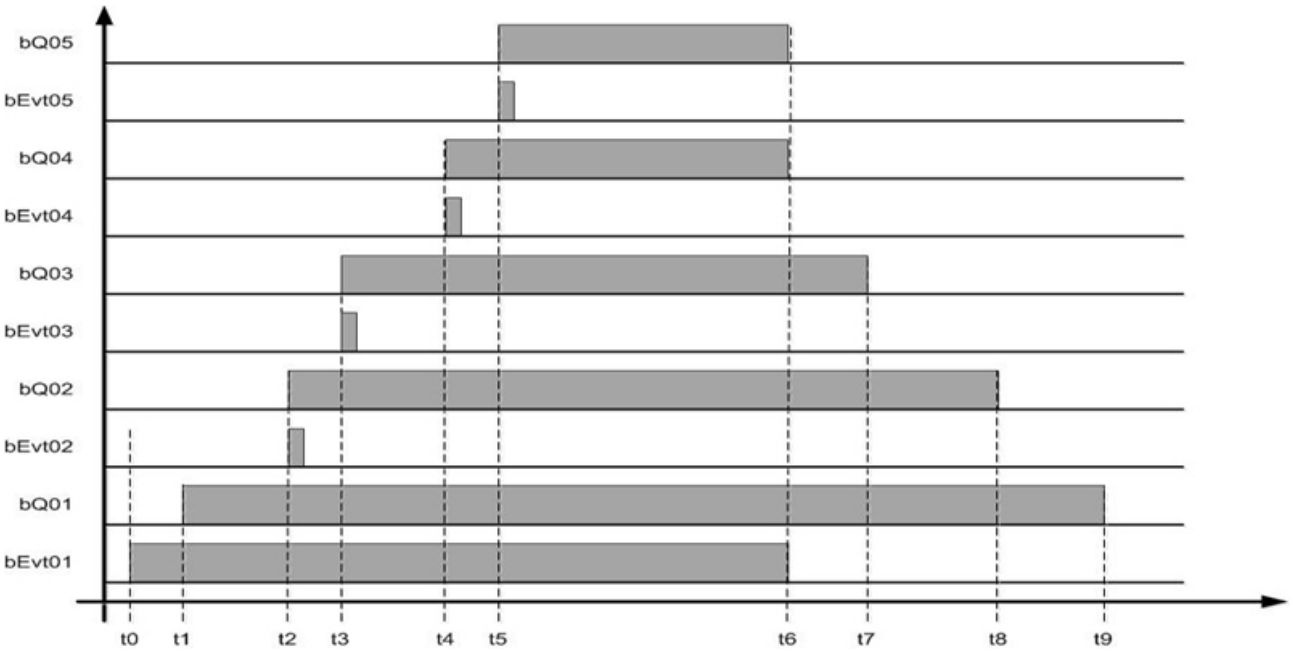
The function block is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. *bEn* is used for general enable of the function block. If *bEn* = FALSE, all outputs of *bQ01* to *bQ12* are set to FALSE. The control sequence starts at input *bEvt01*. Once

the timer *udiDlyOn01* 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* to *udiDlyOn12*. If *bEvt01* becomes FALSE once the control chain is up and running, the control sequence switches back in reverse order. The switch-off of the outputs is delayed through the timers *udiDlyOff01* to *udiDlyOff12*.

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 *udiStep* which shows the output *udiActvEvt*+1, is available for application with a BACnet multistate output object, which cannot show "0".

*udiRemTiDlyOn* indicates the time remaining to the next step during up-switching of the control chain. *udiRemTiDlyOff* indicates the time remaining to the next lower step during down-switching of the control chain.

### Example




t0 step sequence switch-on
t1 step 1 switch-on, *udiDlyOn01* = t1 - t0
t2 event enable step 2, switch-on step 2, *udiDlyOn02* = 0
t3 event enable step 3, switch-on step 3, *udiDlyOn03* = 0
t4 event enable step 4, switch-on step 4, *udiDlyOn04* = 0
t5 event enable step 5, switch-on step 5, *udiDlyOn05* = 0
t6 step sequence switch-off, switch-off step 5, switch-off step 4; *udiDlyOff05* = 0, *udiDlyOff04* = 0
t7 switch-off step 3, *udiDlyOff03* = t7 -t6
t8 switch-off step 2, *udiDlyOff02* = t8 -t7
t9 switch-off step 1, *udiDlyOff01* = t9 -t8

### Inputs/outputs

### VAR_INPUT

```
bEn          : BOOL;
bEvt01       : BOOL;
udiDlyOn01   : UDINT;
udiDlyOff01  : UDINT;
bEvt02       : BOOL;
udiDlyOn02   : UDINT;
udiDlyOff02  : UDINT;
bEvt03       : BOOL;
udiDlyOn03   : UDINT;
udiDlyOff03  : UDINT;
bEvt04       : BOOL;
udiDlyOn04   : UDINT;
udiDlyOff04  : UDINT;
bEvt05       : BOOL;
```

```
udiDlyOn05  : UDINT;
udiDlyOff05 : UDINT;
bEvt06      : BOOL;
udiDlyOn06  : UDINT;
udiDlyOff06 : UDINT;
bEvt07      : BOOL;
udiDlyOn07  : UDINT;
udiDlyOff07 : UDINT;
bEvt08      : BOOL;
udiDlyOn08  : UDINT;
udiDlyOff08 : UDINT;
bEvt09      : BOOL;
udiDlyOn09  : UDINT;
udiDlyOff09 : UDINT;
bEvt10      : BOOL;
udiDlyOn10  : UDINT;
udiDlyOff10 : UDINT;
bEvt11      : BOOL;
udiDlyOn11  : UDINT;
udiDlyOff11 : UDINT;
bEvt12      : BOOL;
udiDlyOn12  : UDINT;
udiDlyOff12 : UDINT;
```

**bEn:** Enable function block

**bEvt01:** Control chain switch-on

**udiDlyOn01:** Switch-on delay for output *bQ01* [s]

**udiDlyOff01:** Switch-off delay for output *bQ01* [s]

**bEvt02:** Advance command step 2

**udiDlyOn02:** Switch-on delay for output *bQ02* [s]

**udiDlyOff02:** Switch-off delay for output *bQ02* [s]

**bEvt03:** Advance command step 3

**udiDlyOn03:** Switch-on delay for output *bQ03* [s]

**udiDlyOff03:** Switch-off delay for output *bQ03* [s]

**bEvt04:** Advance command step 4

**udiDlyOn04:** Switch-on delay for output *bQ04* [s]

**udiDlyOff04:** Switch-off delay for output *bQ04* [s]

**bEvt05:** Advance command step 5

**udiDlyOn05:** Switch-on delay for output *bQ05* [s]

**udiDlyOff05:** Switch-off delay for output *bQ05* [s]

**bEvt06:** Advance command step 6

**udiDlyOn06:** Switch-on delay for output *bQ06* [s]

**udiDlyOff06:** Switch-off delay for output *bQ06* [s]

**bEvt07:** Advance command step 7

**udiDlyOn07:** Switch-on delay for output *bQ07* [s]

**udiDlyOff07:** Switch-off delay for output *bQ07* [s]

**bEvt08:** Advance command step 8

**udiDlyOn08:** Switch-on delay for output *bQ08* [s]

**udiDlyOff08:** Switch-off delay for output *bQ08* [s]

**bEvt09:** Advance command step 9

**udiDlyOn09:** Switch-on delay for output *bQ09* [s]

**udiDlyOff09:** Switch-off delay for output *bQ09* [s]

**bEvt10:** Advance command step 10

**udiDlyOn10:** Switch-on delay for output *bQ10* [s]

**udiDlyOff10:** Switch-off delay for output *bQ10* [s]

**bEvt11:** Advance command step 11

**udiDlyOn11:** Switch-on delay for output *bQ11* [s]

**udiDlyOff11:** Switch-off delay for output *bQ11* [s]

**bEvt12:** Advance command step 12

**udiDlyOn12:** Switch-on delay for output *bQ12* [s]

**udiDlyOff12:** Switch-off delay for output *bQ12* [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  : UDINT;
udiRemTiDlyOff : UDINT;
```

**bQ01:** step 1 on

**bQ02:** step 2 on

**bQ03:** step 3 on

**bQ04:** step 4 on

**bQ05:** step 5 on

**bQ06:** step 6 on

**bQ07:** step 7 on

**bQ08:** step 8 on

**bQ09:** step 9 on

**bQ10:** step 10 on

**bQ11:** step 11 on

**bQ12:** step 12 on

**bUp:** control chain is in ascending state

**bDwn:** control chain is in descending state

**udiActvEvt:** active step, display 0...12; "0" represents a non-active step sequence.

**udiRTiDlyOn:** remaining time until switching up to the next step [s]

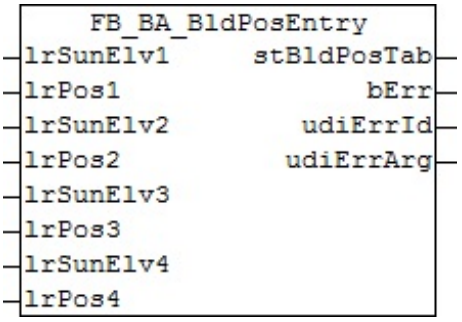
**udiRTiDlyOff:** remaining time until switching down to the previous step [s]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.101 FB_BA_BldPosEntry

This block is used for entering interpolation points for the function block FB_BA_SunPrtc [▶ 302], if this function block is operated in height positioning mode with the aid of a table, see E_BA_PosMod [▶ 324].

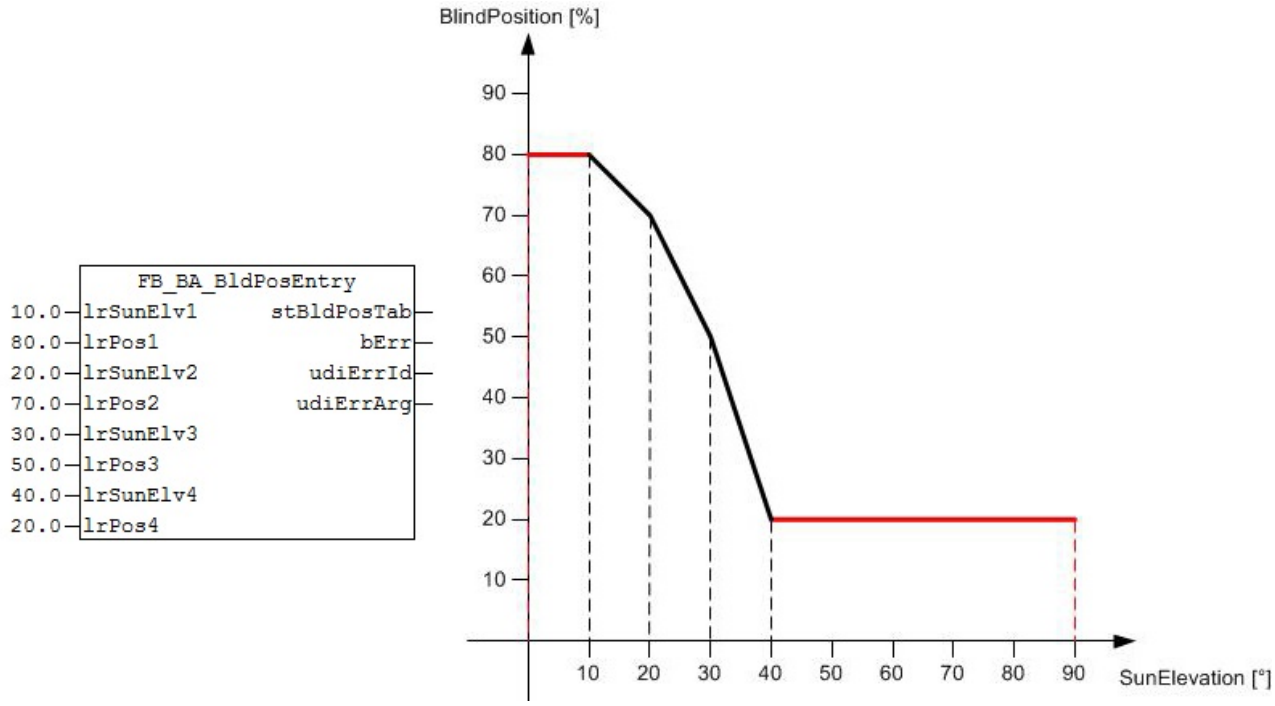



### Functional description

In addition to the operating modes "Fixed shutter height" and "Maximum incidence of light", the function block FB_BA_SunPrtc [▶ 302] 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 [▶ 302], this block is to be preceded by the function block FB_BA_BldPosEntry. Four interpolation points can be parameterised on this 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 *lrSunElv1* .. *lrSunElv4* must be unique; for example, the following situation must be avoided:
[*lrSunElv1* = 10 ; *rBldPos1* = 50] and at the same time [*lrSunElv2* = 10 *; rBldPos2* = 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:

- lrSunElv1 < *lrSunElv2* < *lrSunElv3* < *lrSunElv4* - (values ascending and not equal)
- 0 ≤ *rSunElv* ≤ 90 ( [°] - validity range source values)
- 0 ≤ *rBldPos* ≤ 100 (in percent - validity range target values)

The function block checks the values entered for these conditions and outputs an error code [▶ 334] if they are not satisfied. In addition, the output *bValid* 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 *lrSunElv4* at *rSunElv* = 90 with *rBldPos4*. This ensures that 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 [▶ 302] to 6; see ST_BA_BldPosTab [▶ 325].

The interpolation of the values takes place in the glare protection block.

### Inputs/outputs

### VAR_INPUT

```
lrSunElv1 : LREAL;
lrPos1    : LREAL;
lrSunElv2 : LREAL;
lrPos2    : LREAL;
lrSunElv3 : LREAL;
lrPos3    : LREAL;
lrSunElv4 : LREAL;
lrPos4    : LREAL;
```

**lrSunElv1:** position of the sun of the 1st interpolation point (0°…90°)

**rBldPos1:** blind position (degree of closure) at the 1st interpolation point (0%…100%)

**lrSunElv2:** position of the sun of the 2nd interpolation point (0°…90°)

**rBldPos2:** blind position (degree of closure) at the 2nd interpolation point (0%…100%)

**lrSunElv3:** position of the sun of the 3rd interpolation point (0°…90°)

**rBldPos3:** blind position (degree of closure) at the 3rd interpolation point (0%…100%)

**lrSunElv4:** position of the sun of the 4th interpolation point (0°…90°)

**rBldPos4:** blind position (degree of closure) at the 4th interpolation point (0%…100%)

### VAR_OUTPUT

```
bValid       : BOOL;
uiErrorId    : UDINT;
stBldPosTab  : ST_BA_BldPosTab;
```

**stBldPosTab:** transfer structure of the interpolation points, see ST_BA_BldPosTab [▶ 325]

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

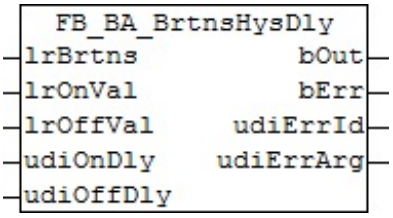
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.102 FB_BA_BrtnsHysDly

Brightness threshold switch




### Functional description

This function block represents a threshold switch for brightness. The switch-on and switch-off behaviour can additionally be delayed.

### Inputs/outputs

#### VAR_INPUT

```
lrBrtns     : LREAL;
lrOnVal     : LREAL;
lrOffVal    : LREAL;
udiOnDly    : UDINT;
udiOffDly   : UDINT;
```

**lrBrtns:** outdoor brightness [lx]

**lrOnVal:** switch-on threshold value [lx]. This must be larger than the switch-off threshold value *usiOffValue.*

**lrOffVal:** switch-off threshold value [lx]. This must be smaller than the switch-on threshold value *usiOnValue.*

**udiOnDly:** start-up delay [s]

**udiOffDly:** switch-off delay [s]

#### VAR_OUTPUT

```
bOut      : BOOL;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**bOut:** Binary delayed output of the threshold switch.

**bErr:** This output is switched to TRUE if the parameters entered are erroneous.

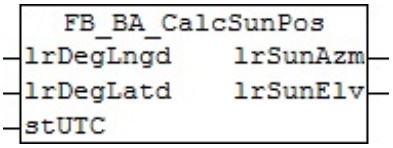
**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.103 FB_BA_CalcSunPos

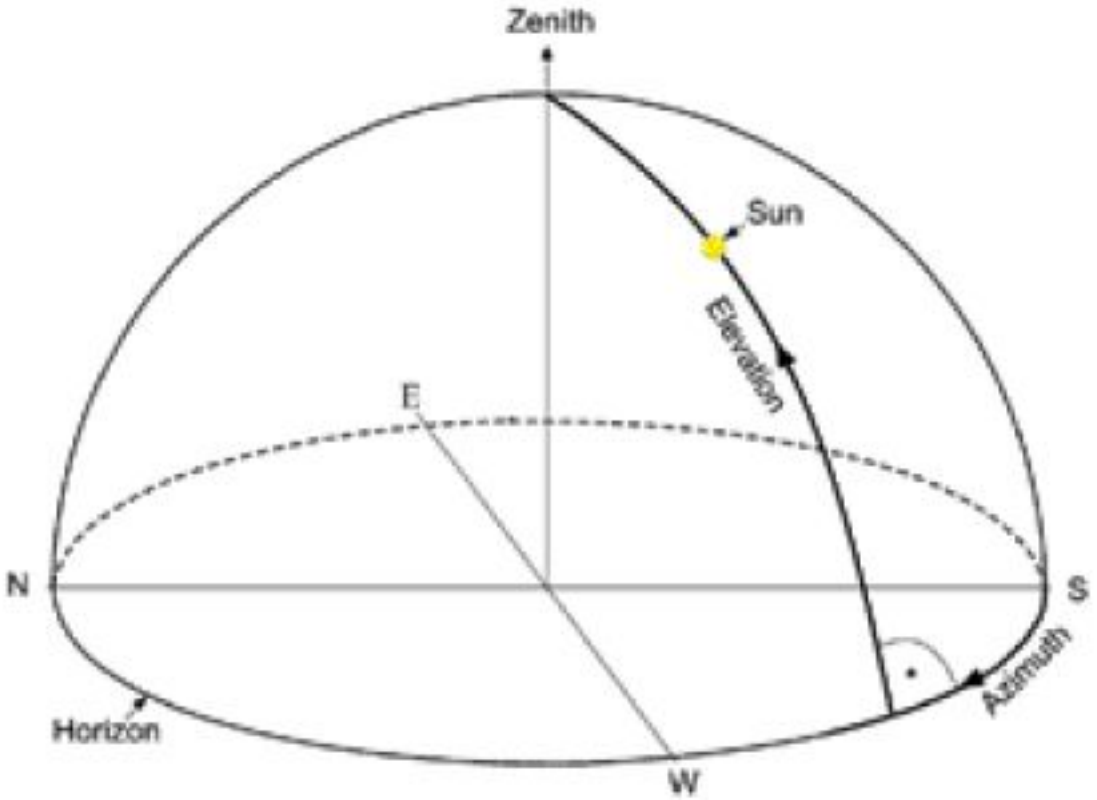
Calculation of sun position based on the date, time, longitude and latitude.




### Functional description

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, where 0° means that the sun is in the horizontal plane of the observation site and 90° means that the sun is directly over the observer's head. The other angle indicates the direction in which the sun is standing. 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 universal 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 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 *lrSunElv*, the sun is invisible. This can be used to determine sunrise and sunset.

### Inputs/outputs

### VAR_INPUT

```
lrDegLngd    : LREAL;
lrDegLatd    : LREAL;
stUTC        : TIMESTRUCT;
```

**lrDegLngd:** longitude [°]

**lrDegLatd:** latitude [°]

**stUTC:** input of the current time as Coordinated Universal Time. The function block FB_BA_GetTime [▶ 320] can be used to read this time from a target system.

### VAR_OUTPUT

```
lrSunAzm     : LREAL;
lrSunElv     : LREAL;
```

**lrSunAzm:** sun direction (northern hemisphere: 0° north ... 90° east ... 180° south ... 270° west ... / southern hemisphere: 0° south ... 90° east ... 180° north ... 270° west ...)

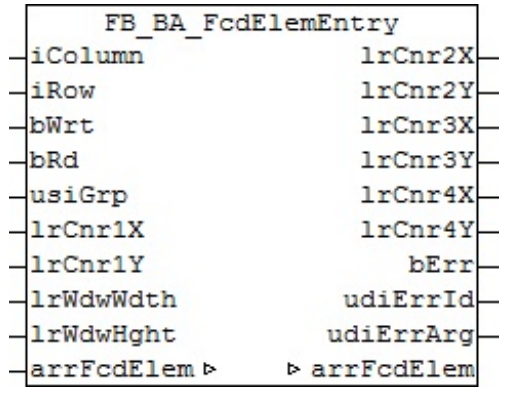
**lrSunElv:** sun elevation (0° horizontal ... 90° vertical)

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.104 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 [▶ 46]. It is intended to facilitate the input of the element information - also about the use of the target visualisation. A schematic representation of the objects with description of the coordinates is shown in Shading correction: principles and definitions [▶ 21].



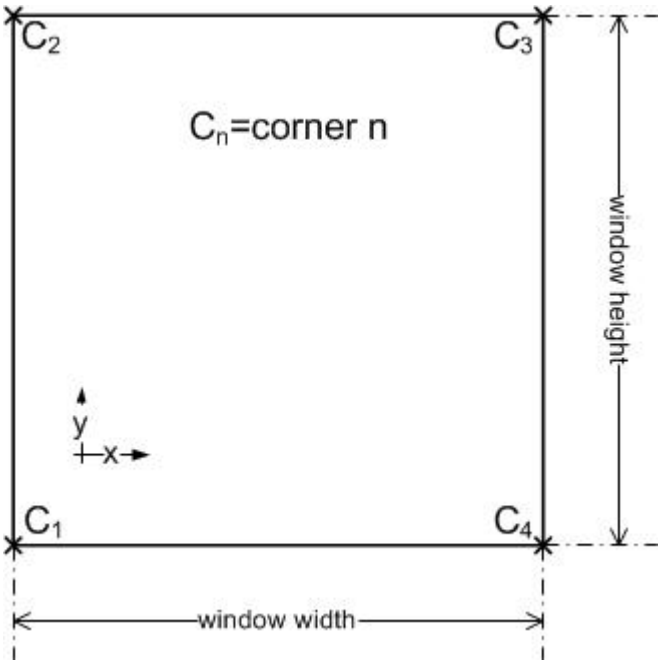

### Function description

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..gBA_cMaxColumnFcd, 1..gBA_cMaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Each individual element *arrFcdElem[x,y]* carries the information for one facade element (ST_BA_FcdElem [▶ 327]). The information includes the group membership, 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 data in [m]!**

*lrCnr2X* = *lrCnr1X*
*lrCnr2Y* = *lrCnr1Y* + *lrWdwHght* (window height)
*lrCnr3X* = *lrCnr1X* + *lrWdwWdth* (window width)
*lrCnr3Y* = *lrCnr2Y*
*lrCnr4X* = *lrCnr1X* + *lrWdwWdth* (window width)
*lrCnr4Y* = *lrCnr1Y*

The function block is used in three steps:

- Read

- Change
- Write

**Read**

With the entries in *iColumn* and *iRow* the corresponding element is selected from the list *arrFcdElem[iColumn,iRow]*. A rising edge on *bRd* reads the following data from the list element:

- usiGrp group membership,
- lrCnr1X X-coordinate of corner point 1 [m]
- lrCnr1Y Y-coordinate of corner point 1 [m]
- lrWdwWdth window width [m]
- lrWdwHght 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 this is not the case, a corresponding error code [▶ 334] is output at *udiErrId/udiErrArg*. See also "Error (*bErr*=TRUE)" below.

**Write**

The parameterised 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 [▶ 327] therefore also contains a plausibility bit *bVld*, which forwards precisely this information to the function block FB_BA_ShdCorr [▶ 278] to prevent miscalculations.

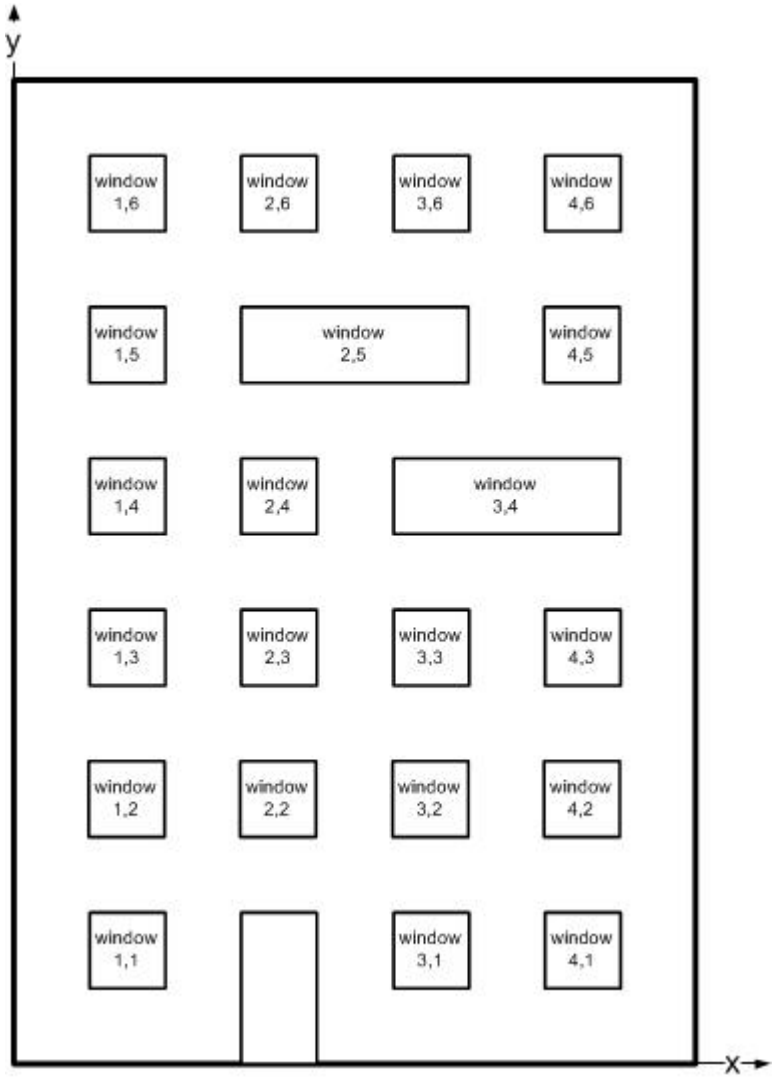
This approach is to be regarded only as a proposal. It is naturally also possible to parameterise the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on *bWrt*.

**Error (bErr=TRUE)**

The function block FB_BA_ShdCorr [▶ 278], 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
- lrCnr1X must be greater than or equal to 0.0
- lrCnr1Y must be greater than or equal to 0.0
- lrWdwWdth must be greater than 0
- lrWdwHght 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 [▶ 327], 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.

### Inputs/outputs

### VAR_INPUT

```
iColumn     : INT;
iRow        : INT;
bWrt        : BOOL;
bRd         : BOOL;
usiGrp      : USINT;
lrCnr1X     : LREAL;
lrCnr1Y     : LREAL;
lrWdwWdth   : LREAL;
lrWdwHght   : LREAL;
```

**iColumn:** 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*.

**iRow:** ditto Row index. **and may not be zero!***iRowiColumn* 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 selected element, *arrFcdElem[iColumn,iRow],* to be read into the function block and assigned to the input variables *usiGrp* to *lrWdwHght*. The resulting output variables are *lrCnr2X* to *lrCnr4Y*. If data are already present on the inputs *usiGrp* to *lrWdwHght* at time of reading, then the data previously read are immediately overwritten with these data.

**bWrt:** A rising edge writes both the entered values and the calculated values into the selected field element *arrFcdElem[iColumn,iRow]*.

**usiGrp:** Association with a group.

**lrCnr1X:** X-coordinate of corner point 1 [m].

**lrCnr1Y:** Y-coordinate of corner point 1 [m].

**lrWdwWdth:** Window width [m].

**lrWdwHght:** Window height [m].

### Inputs/outputs

### VAR_OUTPUT

```
lrCnr2X   : LREAL;
lrCnr2Y   : LREAL;
lrCnr3X   : LREAL;
lrCnr3Y   : LREAL;
lrCnr4X   : LREAL;
lrCnr4Y   : LREAL;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**lrCnr2X:** Calculated X-coordinate of corner point 2 of the window [m]. See "Note [▶ 259]" above.

**lrCnr2Y:** Calculated Y-coordinate of corner point 2 of the window [m]. See "Note [▶ 259]" above.

**lrCnr3X:** Calculated X-coordinate of corner point 3 of the window [m]. See "Note [▶ 259]" above.

**lrCnr3Y:** Calculated Y-coordinate of corner point 3 of the window [m]. See "Note [▶ 259]" above.

**lrCnr4X:** Calculated X-coordinate of corner point 4 of the window [m]. See "Note [▶ 259]" above.

**lrCnr4Y:** Calculated Y-coordinate of corner point 4 of the window [m]. See "Note [▶ 259]" above.

**bErr :** Result check for the entered values.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### VAR_IN_OUT

```
arrFcdElem : ARRAY[1..gBA_cMaxColumnFcd, 1..gBA_cMaxRowFcd] OF ST_BA_FcdElem;
```

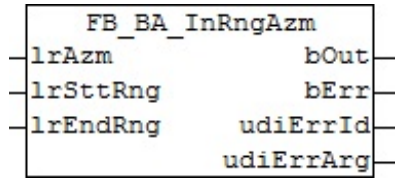
**arrFcdElem:** List of facade elements [▶ 46].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

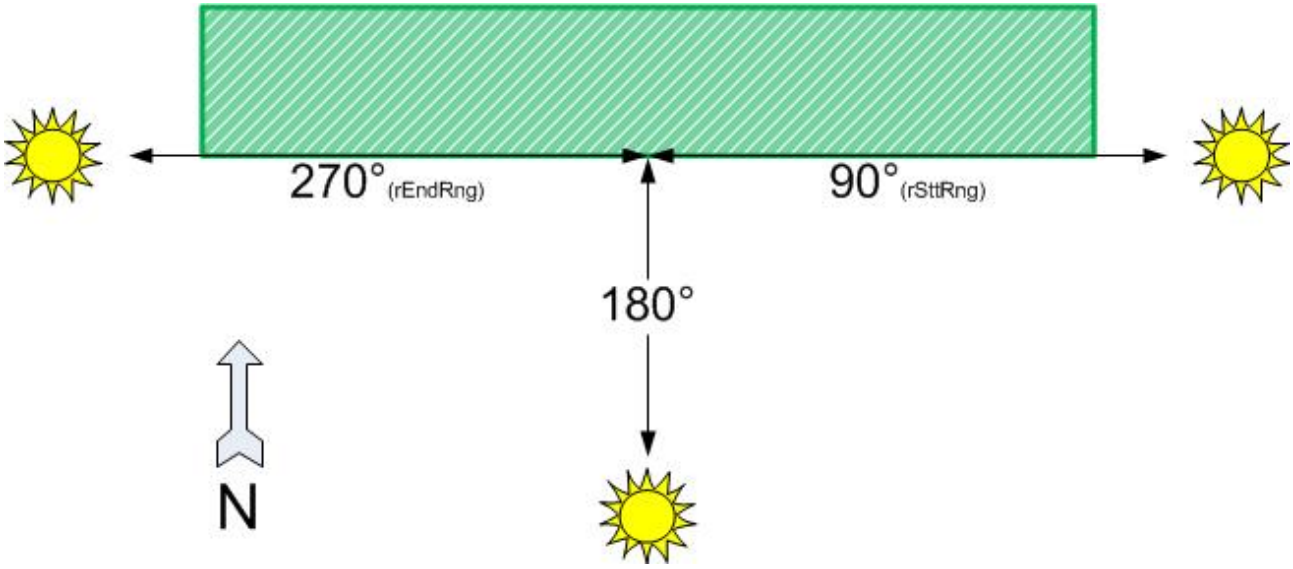
## 8.2.105 FB_BA_InRngAzm

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 [▶ 45], 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.

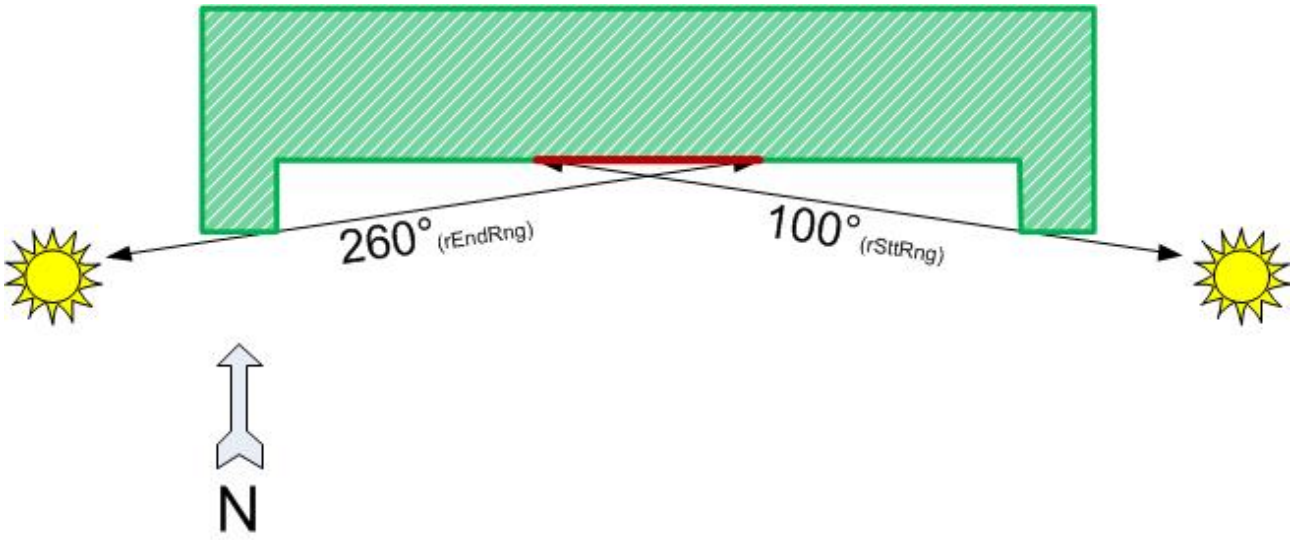



### Functional description

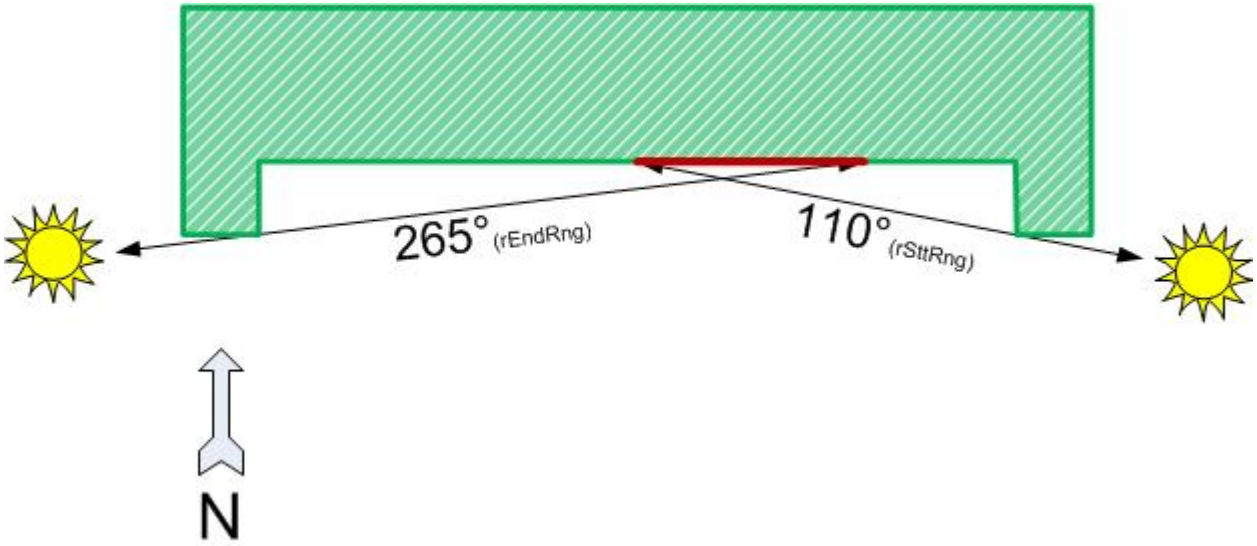
The sun incidence azimuth angle on a smooth facade will always be *facade orientation-90°... facade orientation+90°*.




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 *lrSttRng* may thereby be larger than the end *lrEndRng*; it is then regarded beyond 0°:

*Table 1: Example:*

| | |
|---|---|
| lrAzm | 10.0° |
| lrSttRng | 280.0° |
| lrEndRng | 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/outputs

### VAR_INPUT

```
lrAzm    : LREAL;
lrSttRng : LREAL;
lrEndRng : LREAL;
```

**lrAzm:** current azimuth angle

**lrSttRng:** start of range [°]

**lrEndRng:** end of range [°]

### VAR_OUTPUT

```
bOut       : BOOL;
bErr       : BOOL;
udiErrId   : UDINT;
udiErrArg  : UDINT;
```

**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.

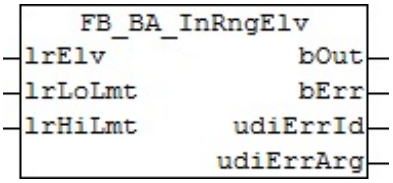
**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

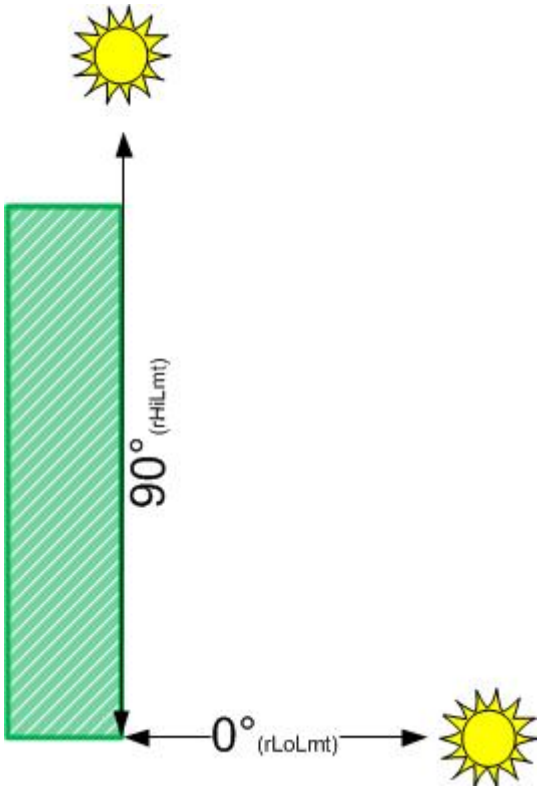
## 8.2.106 FB_BA_InRngElv

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 [▶ 45], 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.

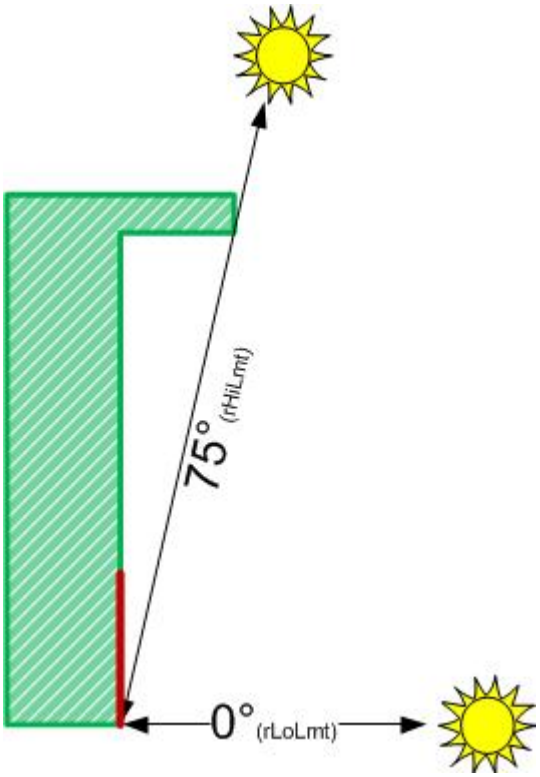



### Functional description

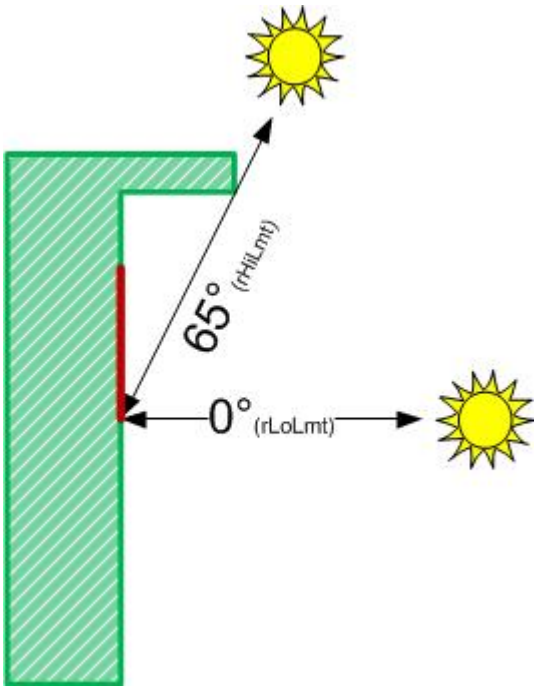
A normal vertical facade is irradiated by the sun at an angle of elevation of 0° to maximally 90°.




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, *lrLoLmt*, may thereby not be greater than or equal to the upper limit, *lrHiLmt*. Such entries result in an error on the output *bErr* – the test output *bOut* is then additionally set to FALSE.

### Inputs/outputs

### VAR_INPUT

```
lrElv   : LREAL;
lrLoLmt : LREAL;
lrHiLmt : LREAL;
```

**lrElv:** current elevation angle [°]

**lrLoLmt:** lower limit.[°]

**lrHiLmt:** upper limit [°]

### VAR_OUTPUT

```
bOut      : BOOL;
bErr      : BOOL;
udiErrId  : UDINT;
udiErrArg : UDINT;
```

**bOut:** facade element is in the sun.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

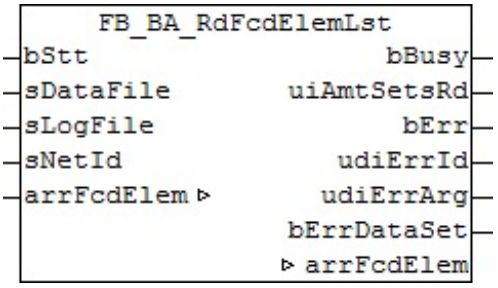
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.107 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 [▶ 46]. In addition the imported data are checked for plausibility and errors are written to a log file.




### Functional description

#### Excel table

The following example shows the Excel table with the entries of the window elements.
All text fields are freely writeable; important are the green marked fields, wherein each line of a data set is marked there.
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 [▶ 46]. These indices directly describe the facade element in the list *arrFcdElems* 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 from 1 to 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 [▶ 278], 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

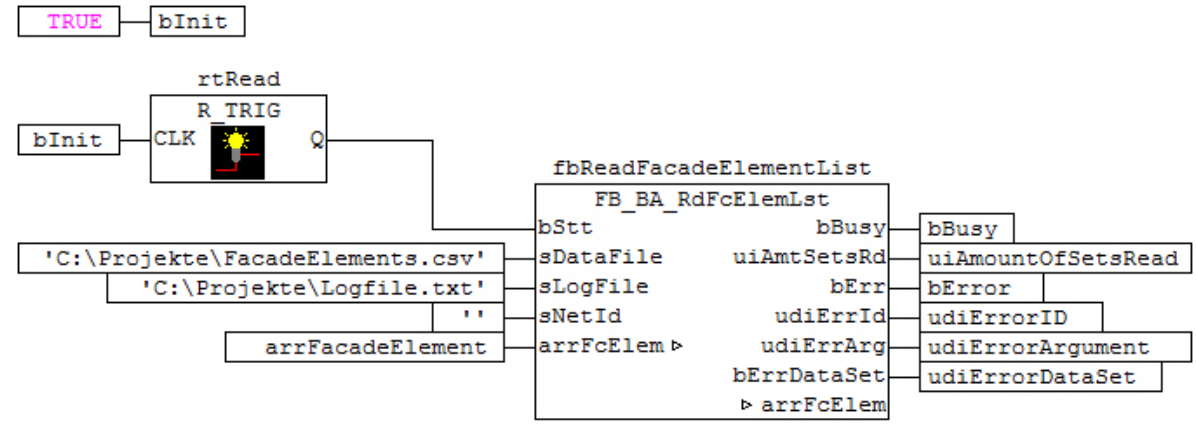
Each time the reading function block is restarted, the log file is rewritten and the old contents are deleted. If there is no log file, it will be automatically created first. The log file then contains either an OK message or a list of all errors that have occurred. Errors connected with the opening, writing or closing of the log file itself

cannot be written at the same time. Therefore, the outputs *udiErrId / udiErrArg* of the reading function block, which show the last error code, should always be considered. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.

### Program Example

```
PROGRAM ReadFacadeElements
VAR
    bInit                   : BOOL;
    rtRead                  : R_TRIG;
    fbReadFacadeElementList : FB_BA_RdFcElemLst;
    arrFacadeElement        : ARRAY[1..gBA_cMaxColumnFc, 1..gBA_cMaxRowFc] OF ST_BA_FcElem;

    bBusy                   : BOOL;
    uiAmountOfSetsRead      : UINT;
    bError                  : BOOL;
    udiErrorID              : UDINT;
    udiErrorArgument        : UDINT;
    udiErrorDataSet         : UDINT;
END_VAR
```




In this example 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 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. The output *bBusy* is TRUEs as long as reading and writing are taking place. The last file handling error that occurred is displayed at *udiErrId / udiErrArg*, in which case *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 *uiAmtSetsRd* for verification purposes.

The errors marked were "built into" the following Excel list. This gives rise to the log file shown:

EN_FacadeElements.csv

| | 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 | 1,4 | 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 | 1,3 | 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
Validation-Error in Data-Set #6, ErrId=33056, ErrArg=5
Validation-Error in Data-Set #22, ErrId=33056, ErrArg=3
Validation-Error in Data-Set #24, ErrId=33056, ErrArg=2
```

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 [▶ 281] and was therefore provided in greater detail with an error number, which is explained in greater detail under error codes [▶ 334]. 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/outputs

#### 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 of the 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, then the values must be displayed separated by semicolon. Example of an entry: *sDataFile:= 'C:\Projects\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. If it is to be run on the local computer, an empty string can be entered.

The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

#### VAR_OUTPUT

```
bBusy       : BOOL;
uiAmtSetsRd : UINT;
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
bErrDataSet : BOOL;
```

**bBusy:** this output is TRUE as long as elements are being read from the file.

**uiAmtSetsRd:** number of data sets read

**bErr:** this output is switched to TRUE, if a file write or read error has occurred.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

**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..gBA_cMaxColumnFcd, 1..gBA_cMaxRowFcd] OF ST_BA_FcdElem;
```

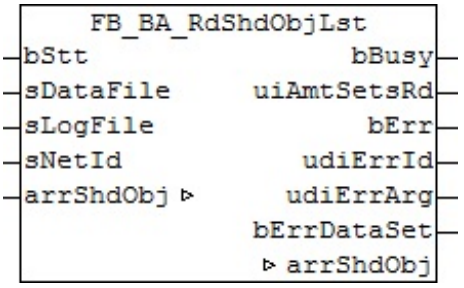
**arrFcdElem:** list of facade elements [▶ 46].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.108 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 [▶ 46]. In addition the imported data are checked for plausibility and errors are written to a log file.




### Functional description

### Excel table

The following example shows the Excel table with the entries of the window elements.
All text fields are freely writeable; important are the green marked fields, wherein each line of a data set is marked there. 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 [▶ 281].
The following rules are to be observed:

- A data set must always start with a '@'.
- The monthly entries 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).
- 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 | | | |

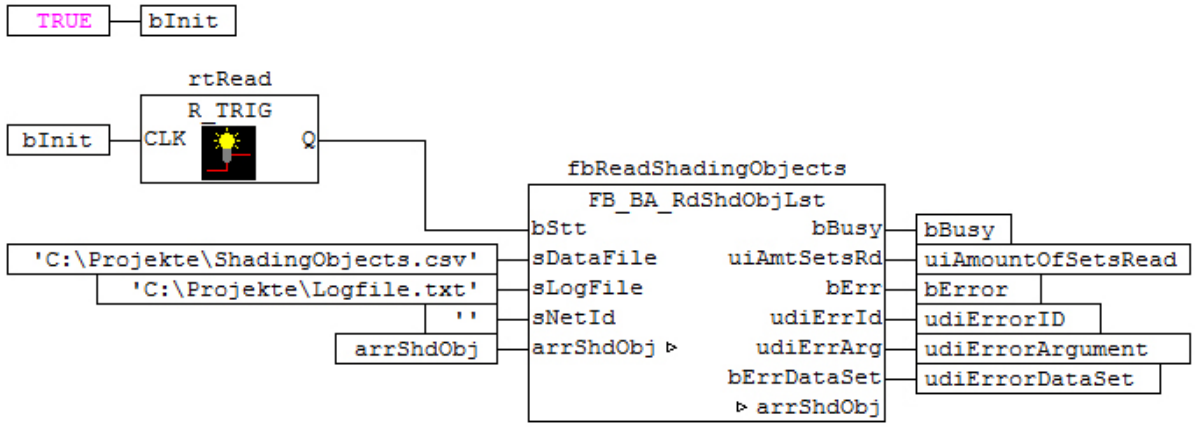
### Log file

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, the outputs *udiErrId / udiErrArg* of the reading function block, which show the last error code, should always be considered. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.

### Program Example

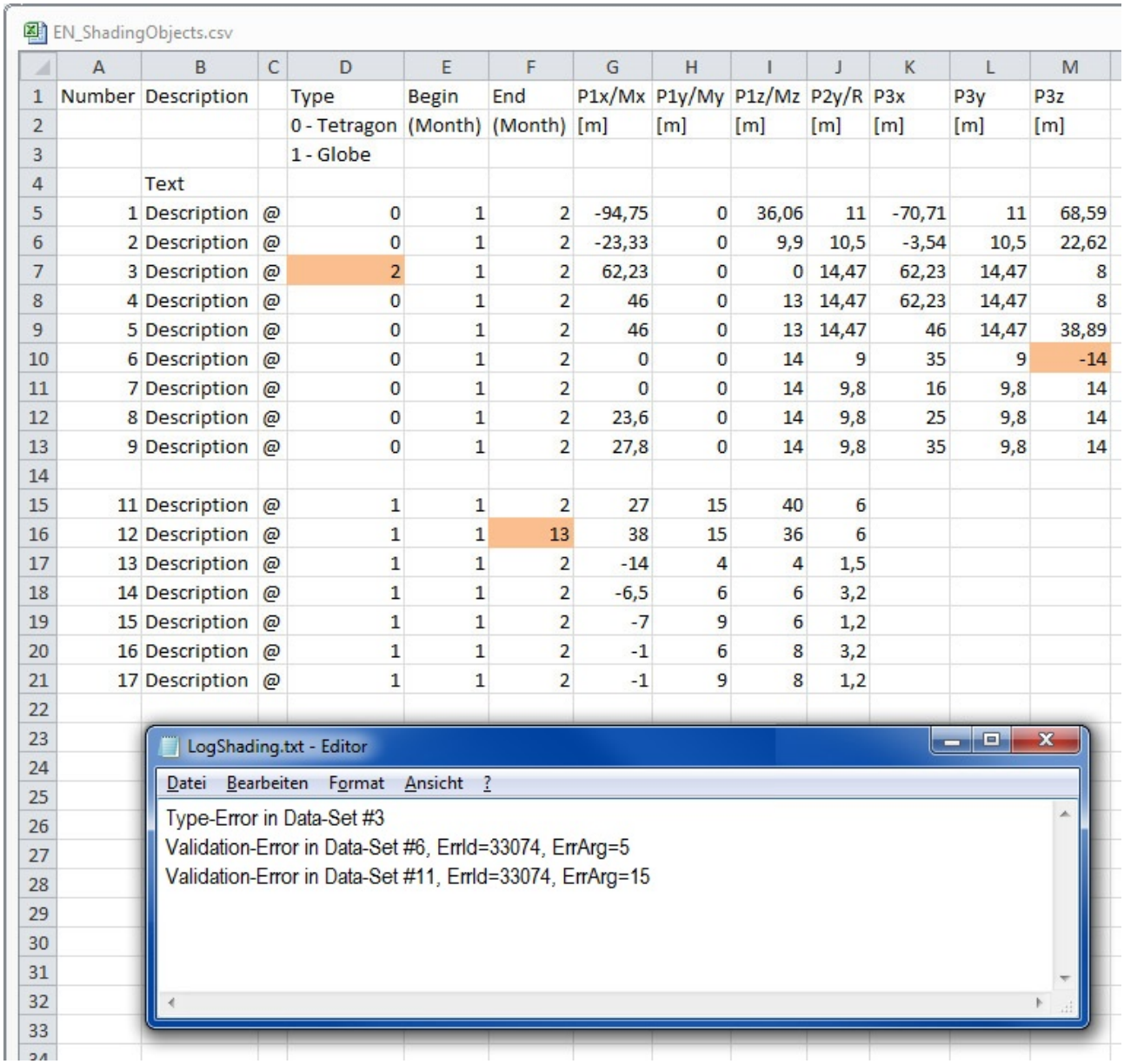
```
PROGRAM ReadShadingObjects
VAR
    bInit                   : BOOL;
    rtRead                  : R_TRIG;
    fbReadShadingObjects    : FB_BA_RdShdObjLst;
    arrShdObj               : ARRAY[1..gBA_cMaxShdObj] OF ST_BA_ShdObj;

    bBusy                   : BOOL;
    uiAmountOfSetsRead      : UINT;
    bError                  : BOOL;
    udiErrorID              : UDINT;
    udiErrorArgument        : UDINT;
    udiErrorDataSet         : UDINT;
END_VAR
```




In this example 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 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. The output *bBusy* is TRUEs if reading and writing are taking place. The last file handling error that occurred is displayed at *udiErrId / udiErrArg*, in which case *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 *uiAmtSetsRd* for verification purposes.

The errors marked were built into the following Excel list. This gives rise to the log file shown:



EN_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 | @ | 2 | 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 | 11 | Description | @ | 1 | 1 | 2 | 27 | 15 | 40 | 6 | | | |
| 16 | 12 | Description | @ | 1 | 1 | 13 | 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 | | | |
| 22 | | | | | | | | | | | | | |



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 [▶ 281] and was therefore provided in greater detail with an error number, which is explained in greater detail under error codes [▶ 334]. 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.**

### Inputs/outputs

### 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 of the 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, then the values must be displayed separated by semicolon. 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. If it is to be run on the local computer, an empty string can be entered.

The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

### VAR_OUTPUT

```
bBusy       : BOOL;
uiAmtSetsRd : UINT;
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
bErrDataSet : BOOL;
```

**bBusy:** this output is TRUE as long as elements are being read from the file.

**uiAmtSetsRd:** number of data sets read

**bErr:** this output is switched to TRUE, if a file write or read error has occurred.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

**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..gBA_cMaxShdObj] OF ST_BA_ShdObj;
```

**arrShdObj:** list of shading objects [▶ 46].

### Requirements

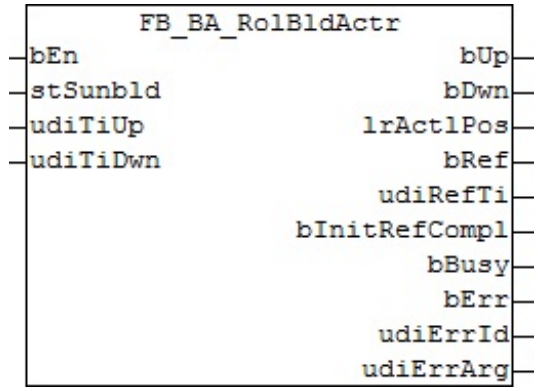
| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.109 FB_BA_RolBldActr

This function block is used for positioning of a blind via two outputs: up and down. The blind can be driven to any desired position with the positioning telegram stSunBld [▶ 331]. In addition, the positioning telegram stSunBld [▶ 331] 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 [▶ 297].

The function block has an internal fixed toggle latch (output *bUp* to output *bDwn*) of **500 ms**.

This function block must be called in every PLC cycle, since the PLC cycle time is included in the calculation of the positions.




### Functional description

Structure of the blind positioning telegram stSunBld [▶ 331].

```
TYPE ST_BA_SunBld:
STRUCT
     lrPos         : LREAL;
     lrAngl        : LREAL;
     bManUp        : BOOL;
     bManDwn       : BOOL;
     bManMod       : BOOL;
     bActv         : BOOL;
END_STRUCT
END_TYPE
```

The current height position and the slat angle are not read in by an additional encoder, but are determined internally by the runtime of the blind.
The two different runtime parameters *udiTiUp* (runtime blind up in [ms]) and *udiTiDwn* (runtime blind down in [ms]) take into account the different travel characteristics.

The function block fundamentally controls the blind via the information from the positioning telegram stSunBld [▶ 331]. 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 blind.

### Referencing

Secure referencing is ensured if the blind is driven upward for longer than its complete drive-up time. The position is then always "0". Since blind positioning without an encoder is naturally always susceptible to error, it is important to automatically reference as often as possible: Whenever "0" is specified as the target position, the blind 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 blind 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 blind can still be moved with the manual "blind 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 "blind down" command.

### Inputs/outputs

#### VAR_INPUT

```
bEn         : BOOL;
stSunBld    : ST_BA_Sunblind;
udiTiUp     : UDINT;
udiTiDwn    : 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 [▶ 331]

**udiTiUp:** complete time for driving up [ms]

u**diTiDwn:** complete time for driving down in ms

### VAR_OUTPUT

```
bUp             : BOOL;
bDwn            : BOOL;
lrActlPos       : LREAL;
udiRefTi        : UDINT;
bInitRefCompl   : BOOL;
bRef            : BOOL;
bBusy           : BOOL;
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**bUp:** control output blind up

**bDwn:** control output blind down

**lrActlPos:** current position in percent

**udiRefTi:** referencing countdown display [s]

**bInitRefCompl:** initial referencing process complete.

**bRef:** the 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 blind in the opposite direction and terminate this mode.

**bBusy:** a positioning or a referencing procedure is in progress.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

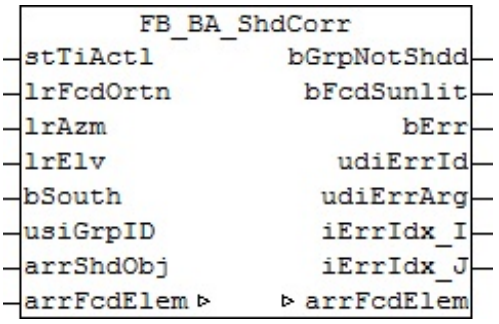
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.110 FB_BA_ShdCorr

Function block for the shading evaluation of a window group on a facade



### Functional description

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 [▶ 46] 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 [▶ 46] 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.

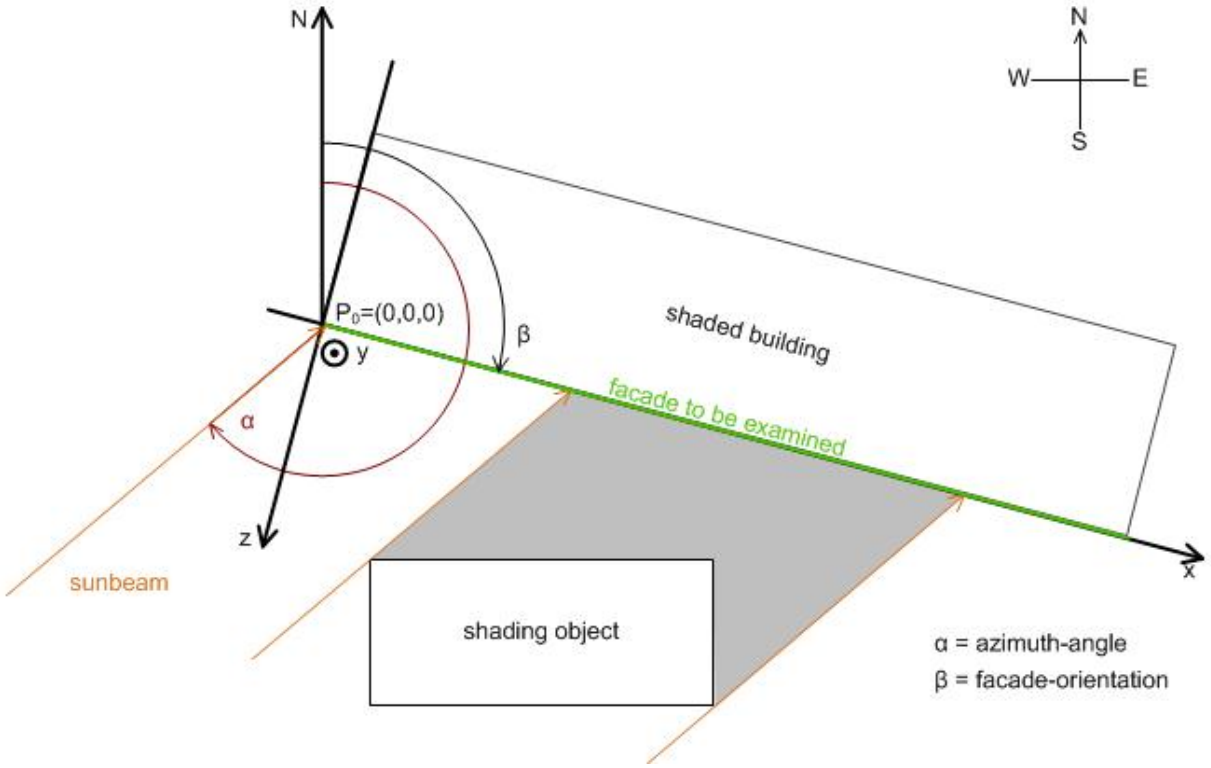
On the basis of the facade orientation (*lrFcdOrtn*), the direction of the sun (*lrAzm*) and the height of the sun (*lrElv*), a calculation can be performed for each corner of a window to check whether this lies in a shaded area. A window group is considered to be completely shaded if all corners are 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. If one regards the drawing presented in the introduction, then this is the case if the following is true:

Facade orientation < azimuth angle < facade orientation + 180°

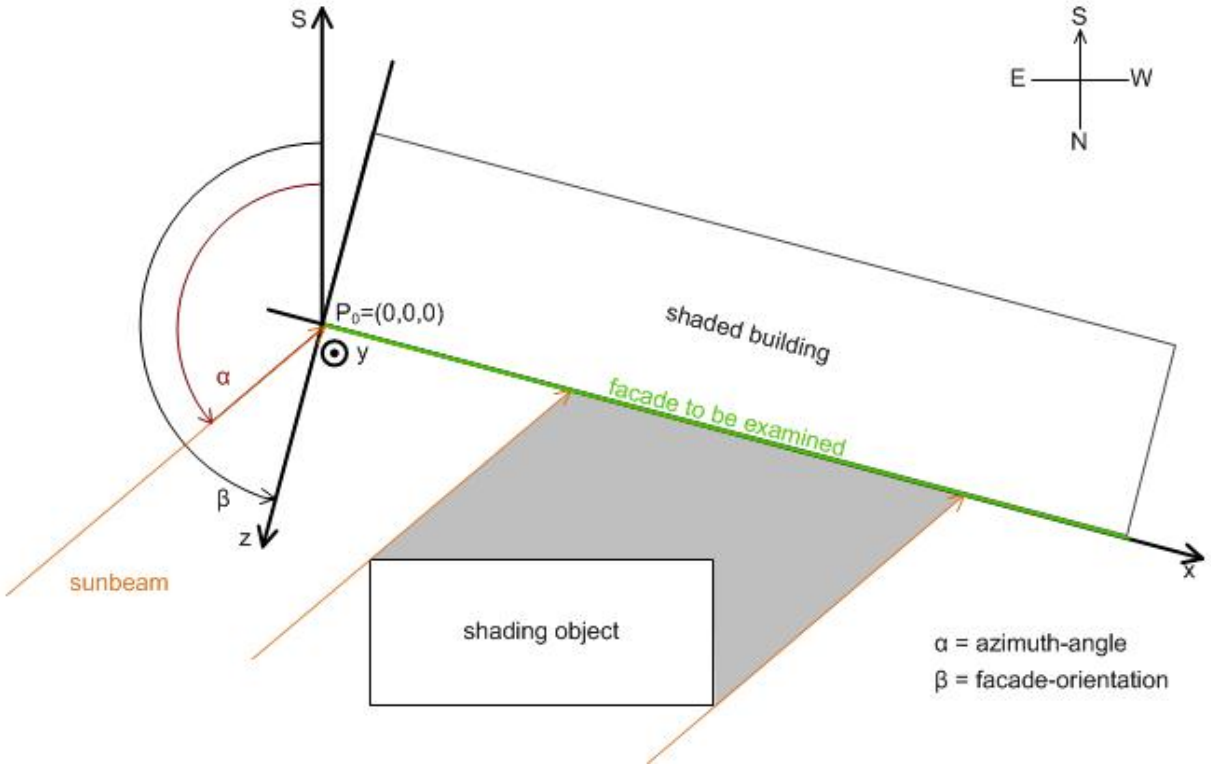



In addition, a calculation is also not required, if the sun has not yet risen, i.e. the solar altitude (elevation) is below 0°. In both cases the output *bFcdSunlit* is set to FALSE.

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)

### Inputs/outputs

### VAR_INPUT

```
stTiActl    : TIMESTRUCT;
lrFcdOrtn   : LREAL;
lrAzm       : LREAL;
lrElv       : LREAL;
bSouth      : BOOL;
usiGrpID    : USINT;
arrShdObj   : ARRAY[1..gBA_cMaxShdObj] 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.

**lrFcdOrtn:** facade orientation, see illustration above

**lrAzm:** direction of the sun at the time of observation [°]

**lrElv:** sun elevation at the time of observation [°]

**bSouth:** FALSE: calculations refer to conditions in the northern hemisphere - TRUE: in the southern hemisphere

**usiGrpId:** window group regarded. The group 0 is reserved here for unused window elements, see FB_BA_FcdElemEntry [▶ 258]. 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.

**arrShdObj:** list of shading objects [▶ 46].

### VAR_OUTPUT

```
bGrpNotShdd : BOOL;
bFcdSunlit  : BOOL;
bErr        : BOOL;
udiErrId    : UDINT;
udiErrArg   : UDINT;
```

**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.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### VAR_IN_OUT

```
arrFcdElem     : ARRAY[1..gBA_cMaxColumnFcd, 1..gBA_cMaxRowFcd] OF ST_BA_FcdElem;
```

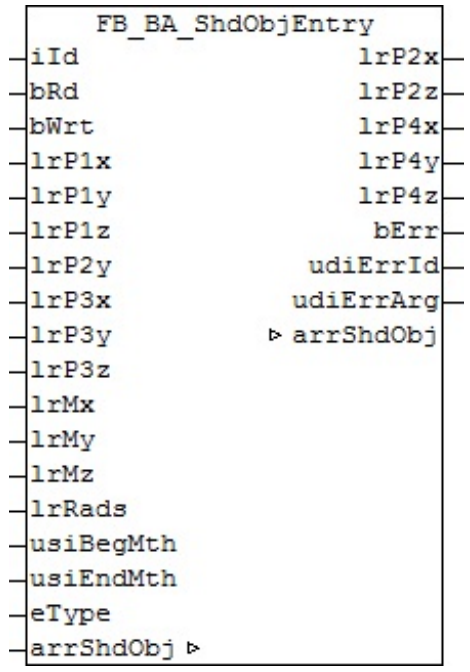
**arrFcdElem:** list of facade elements [▶ 46].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.111 FB_BA_ShdObjEntry

This function block serves for the administration of all shading elements in a facade, which is globally saved in a list of shading elements [▶ 46]. It is intended to facilitate the input of the element information - also regarding the use of the target visualisation. A schematic representation of the objects with description of the coordinates is shown in Shading correction: principles and definitions [▶ 21].



### Functional description

The shading elements are declared in the global variable:

```
VAR_GLOBAL
     arrShadingObject : ARRAY[1..iShadingObjects] OF ST_BA_ShdObj;
END_VAR
```

Each individual element *arrShadingObject[1]* to *arrShadingObject[gBA_cMaxShdObj]* carries the information for one shading element (ST_BA_ShdObj [▶ 329]). 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 *(lrP1x*, *lrP1y*, *lrP1z), (lrP2x*, *lrP2y*, *lrP2z),(lrP3x*, *lrP3y*, *lrP3z)* and *(lrP4x*, *lrP4y*, *lrP4z)*, for a sphere this are the center point *(lrMx*, *lrMy*, *lrMz)* and the radius *lrRads*. In addition, the phase of the shading can be defined via the inputs *usiBegMth* and *usiEndMth*, 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 *arrShadingObject*.

Note**:** The fact that the rectangle coordinates *lrP2x*, *lrP2z*, *lrP4x*, *lrP4y* and *lrP4z* are output values results from the fact that they are formed from the input parameters:

*lrP2x* = *lrP1x*; *lrP2z* = *lrP1z*; *lrP4x* = *lrP3x*; *lrP4y* = *lrP1y*; *lrP4z* = *lrP3z*;

That limits the input of a rectangle to the extent that the lateral edges stand vertically on the floor (*lrP2x* = *lrP1x* and *lrP4x* = *lrP3x*), that the rectangle has no inclination (*lrP2z* = *lrP1z* and *lrP4z* = *lrP3z*) and can only have a different height "upwards", i.e. in the positive y-direction (*lrP4y* = *lrP1y*).

The function block is used in three steps:

- Read
- Change
- Write

#### Read

With the entry to *iId* the appropriate element is selected from the list *arrShadingObject[iId]*. 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 *lrP1x*, *lrP1y*, *lrP1z*, *lrP2y*, *lrP3x*, *lrP3y*, *lrP3z*, *lrMx*, *lrMy*, *lrMz*, *rRadius*, the object enumerator *eType* and the output values *lrP2x*, *lrP2z*, *lrP4x*, *lrP4y* and *lrP4z.* 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 the use of a rectangle is preselected via the value "*eObjectTypeTetragon"* at the input eType [▶ 325], then the output values *lrP2x, lrP2z, lrP4x, lrP4y* and *lrP4z* result from the rectangle coordinates entered, see above.

The values entered are constantly checked for plausibility. The output *bErr* indicates whether the values are valid (*bErr*=FALSE). If this is not the case, a corresponding error code [▶ 334] is output at *udiErrId/udiErrArg*. If a rectangle is defined, then only the inputs *lrP1x*, *lrP1y*, *lrP1z*, *lrP2y*, *lrP3x*, *lrP3y* and *lrP3z* need to be described; the inputs *lrMx*, *lrMy*, *lrMz* and *lrRads* do not need to be linked. In case of a ball definition, only *lrMx*, *lrMy*, *lrMz* and *lrRads* need to be described and the rectangle coordinates can remain unlinked.

#### Write

The parameterised data are written to the list element with the index *iId* upon a rising edge on *bWrt*, regardless of whether they represent valid values or not. The element structure ST_BA_ShdObj [▶ 329] therefore contains a plausibility bit *bVld*, which forwards precisely this information to the function block FB_BA_ShdCorr [▶ 278] to prevent miscalculations.

This approach is to be regarded only as a proposal. It is naturally also possible to parameterise 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/outputs

### VAR_INPUT

```
iId             : INT;
bRd             : BOOL;
bWrt            : BOOL;
lrP1x           : LREAL;
lrP1y           : LREAL;
lrP1z           : LREAL;
lrP2y           : LREAL;
lrP3x           : LREAL;
lrP3y           : LREAL;
lrP3z           : LREAL;
lrMx            : LREAL;
lrMy            : LREAL;
lrMz            : LREAL;
lrRads          : LREAL;
usiBegMth       : USINT;
usiEndMth       : USINT;
eType           : E_BA_ShdObjType;
```

**iId:** index of the selected element. This refers to the selection of a field element of the array stored in the IN-OUT variable *arrShdObj*. **iId may not be zero!** This is due to the field definition, which starts with 1; see above.

**bRd:** the information of the selected element, *arrShdObj[iId]*, is read into the function block with a positive edge at this input and assigned to the input variables *lrP1x* to *eType* and the output variables *lrP2x* to *lrP4z*. If at this time data have already been applied to the inputs *lrP1x* to *eType*, the previously read data are immediately overwritten with these.

**bWrt:** a positive edge writes the values applied to inputs *lrP1x* to *eType* and the values determined and assigned to outputs *lrP2x* to *lrP4z* to the selected field element *arrShdObj[iId].*

**lrP1x:** X-coordinate of point 1 of the shading element (rectangle) [m]

**lrP1y:** Y-coordinate of point 1 of the shading element (rectangle) [m]

**lrP1z:** Z-coordinate of point 1 of the shading element (rectangle) [m]

**lrP2y:** Y-coordinate of point 2 of the shading element (rectangle) [m]

**lrP3x:** X-coordinate of point 3 of the shading element (rectangle) [m]

**lrP3y:** Y-coordinate of point 3 of the shading element (rectangle) [m]

**lrP3z:** Z-coordinate of point 3 of the shading element (rectangle) [m]

**lrMx:** X-coordinate of the center of the shading element (sphere) [m]

**lrMy:** Y-coordinate of the center of the shading element (sphere) [m]

**lrMz:** Z-coordinate of the center of the shading element (sphere) [m]

**lrRads:** radius of the shading element (sphere) [m]

**usiBegMth:** beginning of the shading period (month)

**usiEndMth:** end of the shading period (month)

**eType:** selected element type: rectangle or sphere. See E_BA_ShdObjType [▶ 325].

**Comment on 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).

### VAR_OUTPUT

```
lrP2x     : LREAL;
lrP2z     : LREAL;
lrP4x     : LREAL;
lrP4y     : LREAL;
lrP4z     : LREAL;
bErr      : BOOL;
udiErrId  : UDINT;
```

**lrP2x:** Calculated X-coordinate of point 2 of the shading element (rectangle) [m]. See "Note [▶ 282]" above.

**lrP2z:** Calculated Z-coordinate of point 2 of the shading element (rectangle) [m]. See "Note [▶ 282]" above.

**lrP4x:** Calculated X-coordinate of point 4 of the shading element (rectangle) [m]. See "Note [▶ 282]" above.

**lrP4y:** Calculated Y-coordinate of point 4 of the shading element (rectangle) [m]. See "Note [▶ 282]" above.

**lrP4z:** Calculated Z-coordinate of point 4 of the shading element (rectangle) [m]. See "Note [▶ 282]" above.

**bErr:** Result of the plausibility check for the values entered. With respect to a square it is required that the internal angle is 360° and that the points lie in one plane and *in front of* the facade regarded. In the case of a ball the centre must likewise lie in front of the facade and the radius must be greater than zero.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### VAR_IN_OUT

```
arrShdObj : ARRAY[1..gBA_cMaxShdObj] OF ST_BA_ShdObj;
```

**arrShdObj:** list of shading objects [▶ 46].

### Requirements

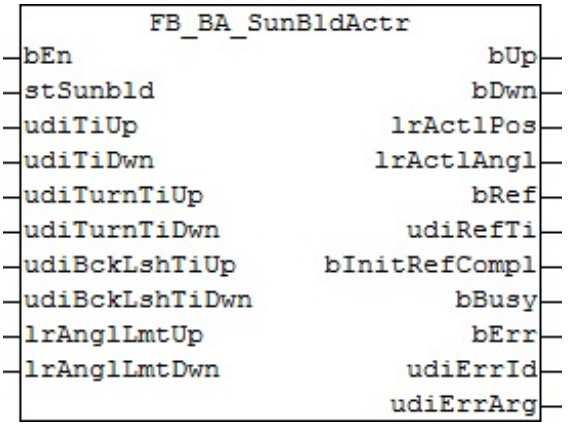
| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.112 FB_BA_SunBldActr

This function block is used for positioning of a slatted 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 [▶ 331]. In addition, the positioning telegram stSunBld [▶ 331] 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 [▶ 297].

The function block has an internal fixed toggle latch (output *bUp* to output *bDwn*) of **500 ms**.

This function block must be called in every PLC cycle, since the PLC cycle time is included in the calculation of the positions.



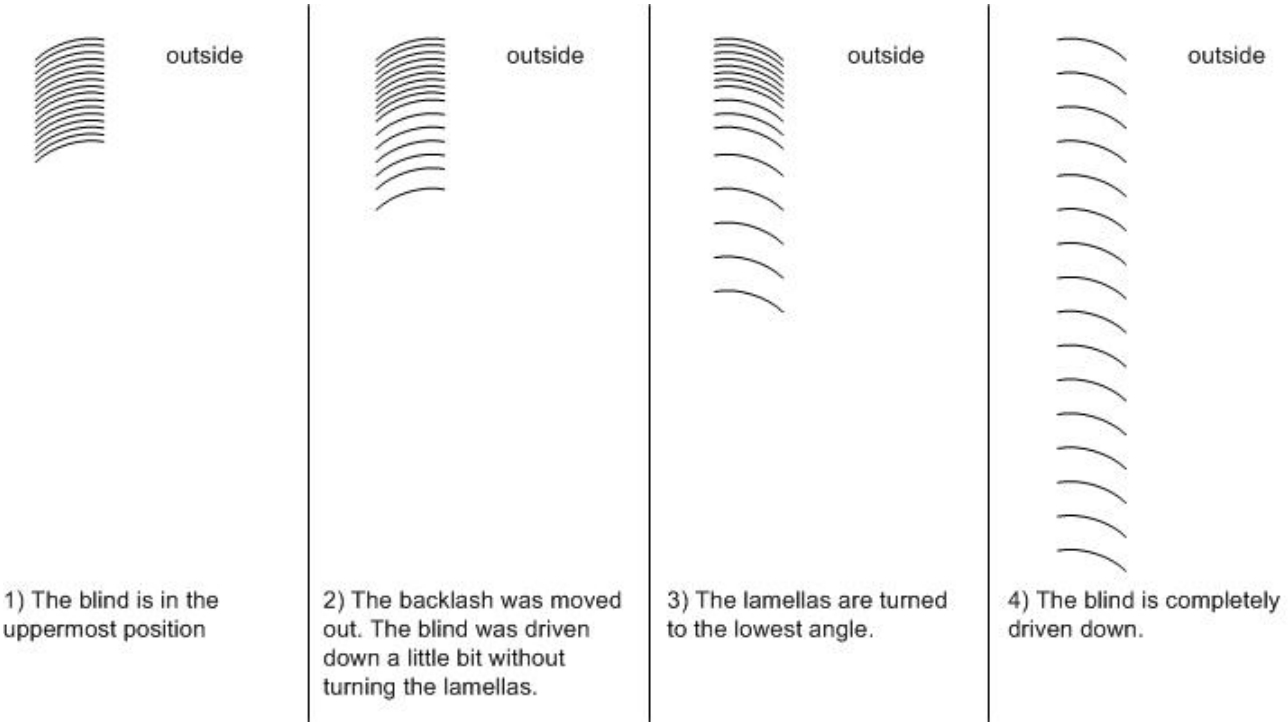

### Functional description

Structure of the blind positioning telegram stSunBld [▶ 331]

```
TYPE ST_BA_SunBld:
STRUCT
     lrPos         : LREAL;
     lrAngl        : LREAL;
     bManUp        : BOOL;
     bManDwn       : BOOL;
     bManMod       : BOOL;
     bActv         : BOOL;
END_STRUCT
END_TYPE
```

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 drive profile:**

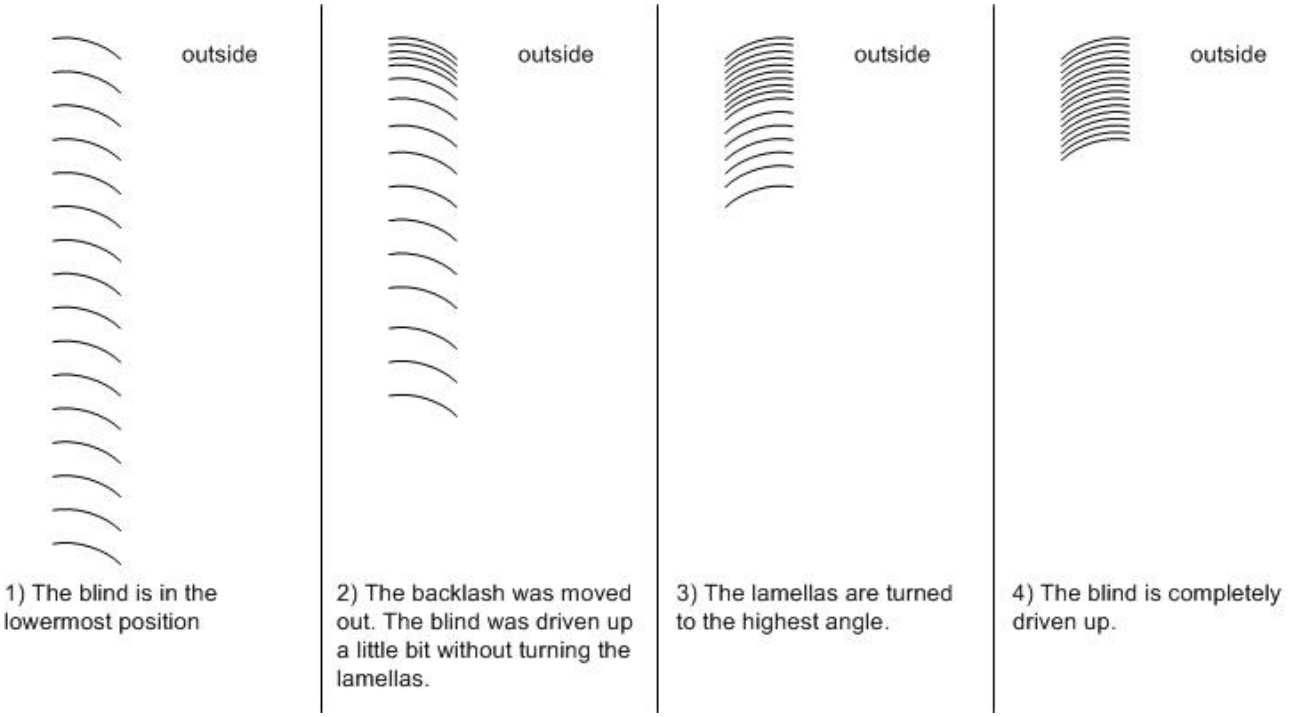



**More detailed explanations of the terms "backlash" and "turning" are given here in the downward movement:**

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 *udiBckLshTiDwn* [ms]. Since it is not known at an arbitrary point after a blind movement of an arbitrary length whether part of the backlash has already been traveled, the most secure way to measure the backlash of the downward movement or its travel time is when the blind has initially been driven fully upward. A further important parameter is the timespan of the subsequent turning of the slats from the "Inward low point" to the "Outward low point". This time is to be entered as *udiTurnTiDwn* [ms] in 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). This time is an important parameter for the movement calculation and is entered in the function block under *udiBckLshTiUp* [ms]. Since it is not known at an arbitrary point after a blind movement of an arbitrary length whether part of the backlash has already been traveled, the most secure way to measure the backlash of the upward movement or its travel time is when the blind has initially been driven fully downward. A further important parameter is the timespan of the subsequent turning of the slats from the "Outward low point" to the "Inward low point". This time is to be entered as *udiTurnTiUp* [ms] in the function block.

**Parameterisation**

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:

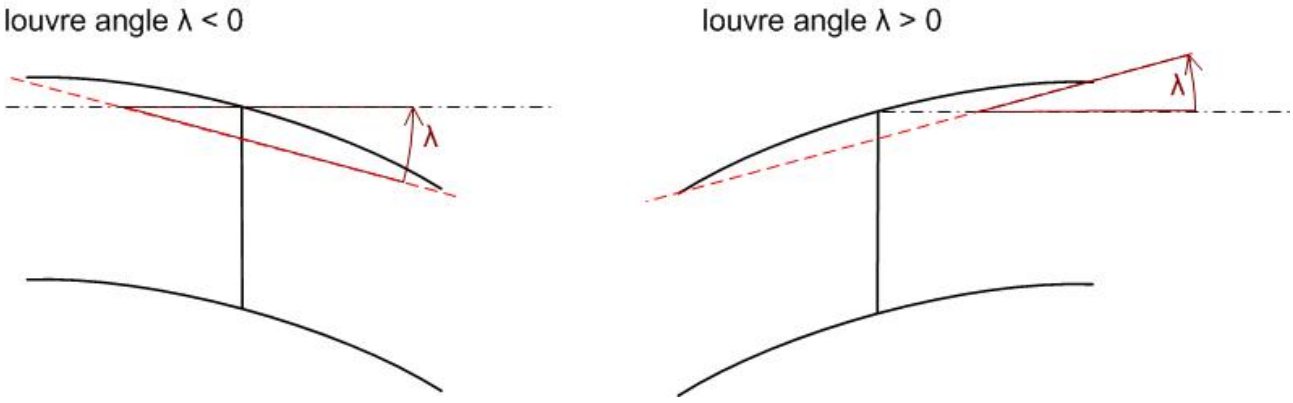
- the travel time of the backlash (*udiBckLshTiUp* / *udiBckLshTiDwn* [ms])
- the turning duration (*udiTurnTiUp* / *udiTurnTiDwn* [ms])
- the total travel time (*udiTiUp* / *udiTiDwn* [ms])

Furthermore the following are required for the calculation:

- the highest louvre angle after turning upwards (*lrAnglLmtUp* [°])

- the lowest louvre angle after turning downwards (*lrAnglLmtDwn* [°])

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 [▶ 331]. 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 blind 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 blind can still be moved with the manual "blind 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 **cannot** be terminated prematurely through a manual "blind down" command.

### Target accuracy

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°.

### Inputs/outputs

#### VAR_INPUT

```
bEn             : BOOL;
stSunbld        : ST_BA_SunBld;
udiTiUp         : UDINT;
udiTiDwn        : UDINT;
udiTurnTiUp     : UDINT;
udiTurnTiDwn    : UDINT;
udiBckLshTiUp   : UDINT;
```

```
udiBckLshTiDwn  : UDINT;
lrAnglLmtUp     : LREAL;
lrAnglLmtDwn    : LREAL;
```

**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 [▶ 331]

**udiTiUp:** complete time for driving up [ms]

**udiTiDwn:** complete time for driving down [ms]

**udiTurnTiUp:** time for turning the slats in the upward direction [ms]

**udiTurnTiDwn:** time for turning the slats in the downward direction [ms]

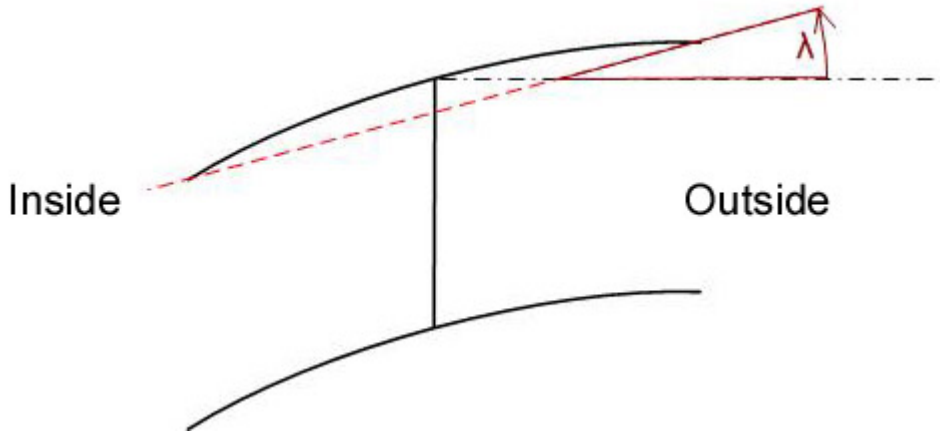
**udiBckLshTiUp:** time to traverse the backlash in the upward direction [ms]

**udiBckLshTiDwn:** time to traverse the backlash in the downward direction [ms]

**lrAnglLmtUp:** highest position of the slats [°]

This position is reached once the blind has moved to the top position.

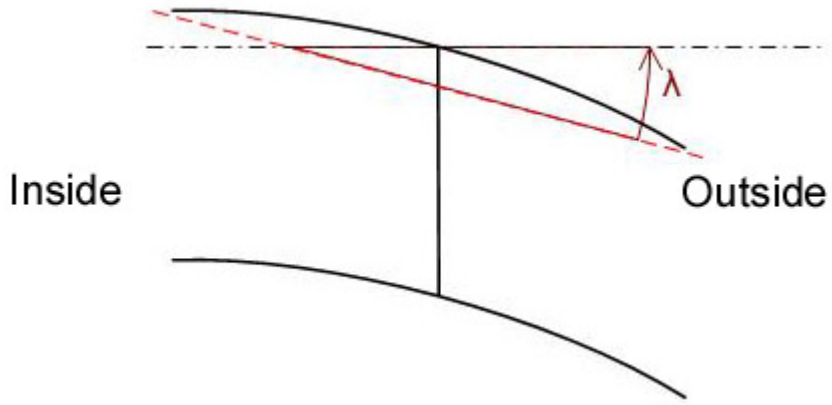
The slat angle λ, as defined above, is then typically greater than zero.




**lrAnglLmtDwn:** 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.



### VAR_OUTPUT

```
bUp           : BOOL;
bDwn          : BOOL;
lrActlPos     : LREAL;
lrActlAngl    : LREAL;
bRef          : BOOL;
bInitRefCompl : BOOL;
bRef          : BOOL;
bBusy         : BOOL;
bErr          : BOOL;
udiErrId      : UDINT;
udiErrArg     : UDINT;
```

**bUp:** control output blind up

**bDwn:** control output blind down

**lrActlPos:** current position in percent

**lrActlAngl:** current slat angle [°]

**bRef:** the 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 blind in the opposite direction and terminate this mode.

**udiRefTi:** 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.

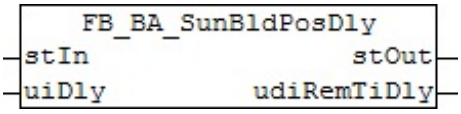
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.113 FB_BA_SunBldPosDly

This function block delays changes in position based on automatic commands.




### Functional description

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 desirable to start the blind drives slightly staggered, in order to the avoid excessive total current values.

This function block relays automatic commands from the input telegram stIn [▶ 331] to the output telegram stOut [▶ 331] with a delay. A distinction is made between three cases:

1. the blind position *lrPos* has changed in automatic mode (*bManMode* = FALSE in telegram *stIn*).
2. the slat angle *lrAngl* 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*. However, in these three cases the following values are set in the output telegram *stOut* for the time of *uiDly* [ms]:

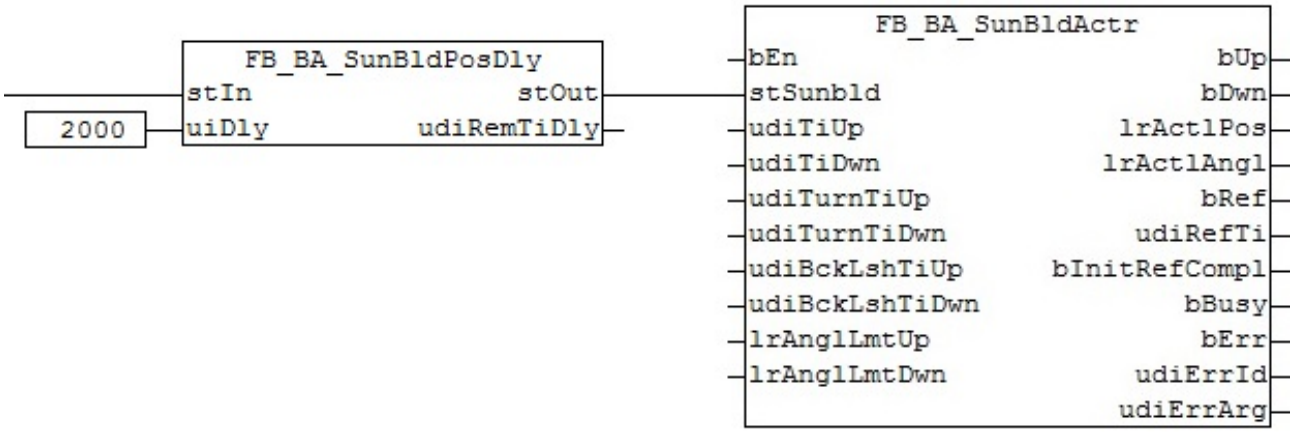
- bManMode = TRUE
- bManUp = FALSE
- bManDwn = FALSE

This ensures that the blind controlled via the function block FB_BA_SunBldActr [▶ 284] 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/outputs

### VAR_INPUT

```
stIn      : ST_BA_Sunblind;
uiDly     : UINT;
```

**stIn:** input positioning telegram, see ST_BA_SunBld [▶ 331]

**uiDly:** delay time of the active bit in the positioning telegram [ms]

### VAR_OUTPUT

```
stOut     : ST_BA_Sunblind;
```

**stOut:** output positioning telegram, see ST_BA_SunBld [▶ 331]

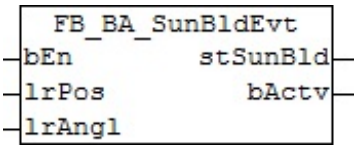
**udiRemTiDly:** display output for elapsed delay time [s]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.114 FB_BA_SunBldEvt

This function block serves to preset the position and angle for any desired event. It can be used, for example, to drive to a parking position or to drive the blind upward for maintenance.




### Functional description

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 [▶ 331] is set, and the values entered for the In/Out variables *lrPos* for the blind height [%] and *lrAngl* 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 [▶ 331] is reset and the positions for height and angle are set to "0". If the priority function block is used, then a function with a lower priority can take over the control.

### Inputs/outputs

#### VAR_INPUT

```
bEn       : BOOL;
lrPos     : LREAL;
lrAngl    : LREAL;
```

**bEn:** 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 [▶ 331]. A FALSE signal resets the active flag again and sets position and angle to zero.

**lrPos:** height position of the blind [%] in case of activation

**lrAngl:** slat 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 [▶ 331] 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 [▶ 331]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.115 FB_BA_SunBldPrioSwi4

Priority control for up to 4 positioning telegrams (*stSunBld_Prio1* ... *stSunBld_Prio4*) of type ST_BA_SunBld [▶ 331].

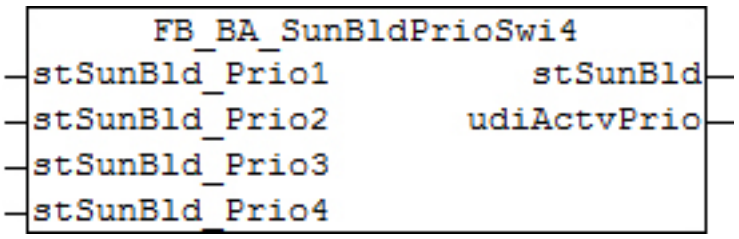



Fig. 2: FB_BA_SunBldPrioSwi4

### Functional description

Structure of the blind positioning telegram stSunBld [▶ 331].

```
TYPE ST_BA_SunBld:
STRUCT
     lrPos        : LREAL;
     lrAngl       : LREAL;
     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 *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. *lrPos*=0, *lrAngl*=0, *bManUp*=FALSE, *bManDwn*=FALSE, *bManMod*=FALSE, *bActv*=FALSE. Since the blind function block FB_BA_SunBldActr [▶ 284] or the roller blind function block FB_BA_RolBldActr [▶ 276] 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/outputs

#### 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;
```

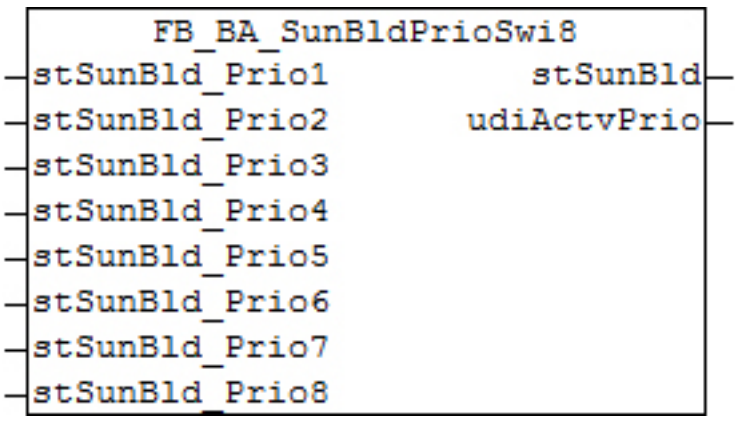
**stSunBld:** resulting positioning telegram
**udiActvPrio:** active positioning telegram. If none is active, "0" is output

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.116 FB_BA_SunBldPrioSwi8

Priority controller for up to 8 positioning telegrams (*stSunBld_Prio1* ... *stSunBld_Prio8*) of the type ST_BA_SunBld [▶ 331].




### Functional description

Structure of the blind positioning telegram stSunBld [▶ 331].

```
TYPE ST_BA_SunBld:
STRUCT
     lrPos         : LREAL;
     lrAngl        : LREAL;
     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 *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. *lrPos*=0, *lrAngl*=0, *bManUp*=FALSE, *bManDwn*=FALSE, *bManMod*=FALSE, *bActv*=FALSE. Since the blind function block FB_BA_SunBldActr [▶ 284] or the roller blind function block FB_BA_RolBldActr [▶ 276] 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/outputs

#### 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;
```

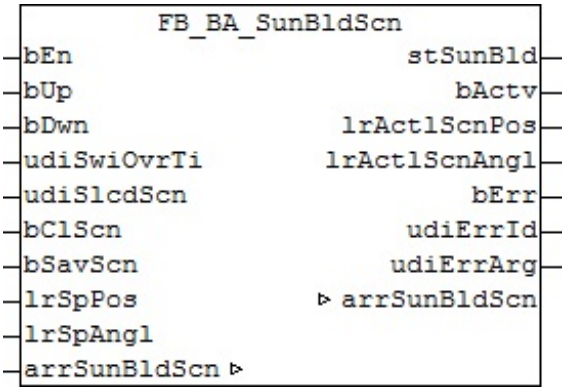
**stSunBld:** resulting positioning telegram
**udiActvPrio:** active positioning telegram. If none is active, "0" is output.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.117 FB_BA_SunBldScn

This function block represents an extension of the manual controller FB_BA_SunBldSwi [▶ 297] by a scene memory and a call function. The blind control FB_BA_SunBldActr [▶ 284] or the roller blind control FB_BA_RolBldActr [▶ 276] can be active in manual mode and also directly target previously stored positions (scenes). Up to 21 scenes can be saved.




### Functional description

Structure of the blind positioning telegram stSunBld [▶ 331]

```
TYPE ST_BA_SunBld:
STRUCT
     lrPos         : LREAL;
     lrAngl        : LREAL;
     bManUp        : BOOL;
     bManDwn       : BOOL;
     bManMod       : BOOL;
     bActv         : BOOL;
END_STRUCT
END_TYPE
```

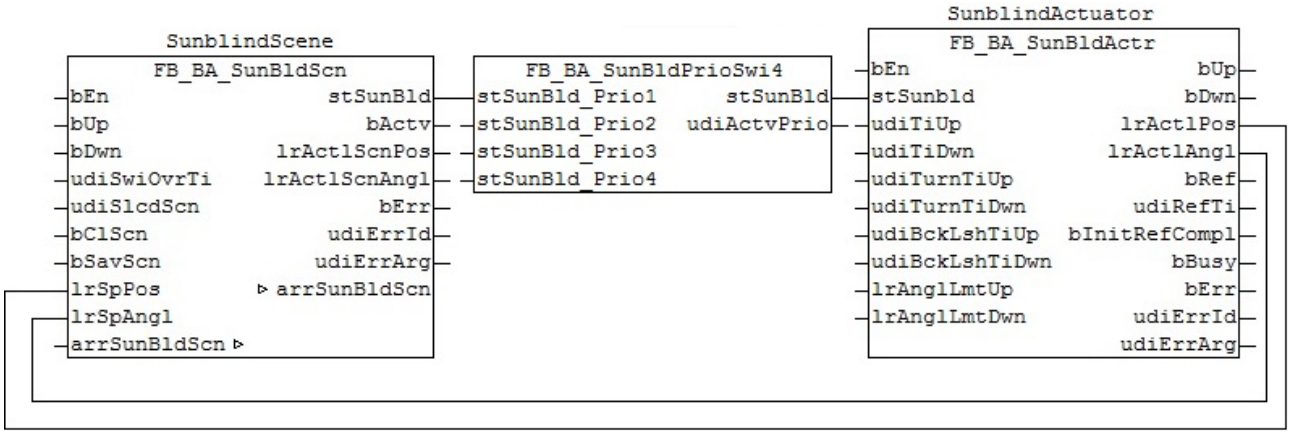
### Operation

In manual mode, the function block controls the blind function block FB_BA_SunBldActr [▶ 284] or the roller blind function block FB_BA_RolBldActr [▶ 276] 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 for longer than the entered time *udiSwiOvrTi* [ms], the corresponding control command latches. Activating a command input again clears this latching.
A rising edge at *bSavScn* saves the current position and the slat angle in the scene selected at *udiSlcdScn*. This procedure is possible at any time, even during active positioning. With *bClScn* the selected scene is called up, i.e. the stored values of position and angle are driven to.
If the function block is activated by *bEn* = TRUE, bit *bActv* and *bManMod* is set immediately in the positioning telegram. The function block uses this to notify a priority switch (FB_BA_SunBldPrioSwi4 [▶ 291] or FB_BA_SunBldPrioSwi8 [▶ 293]) of its priority over lower priorities. The setting of *bManMod* indicates to the actuator function block that no automatic positioning command has to be executed.

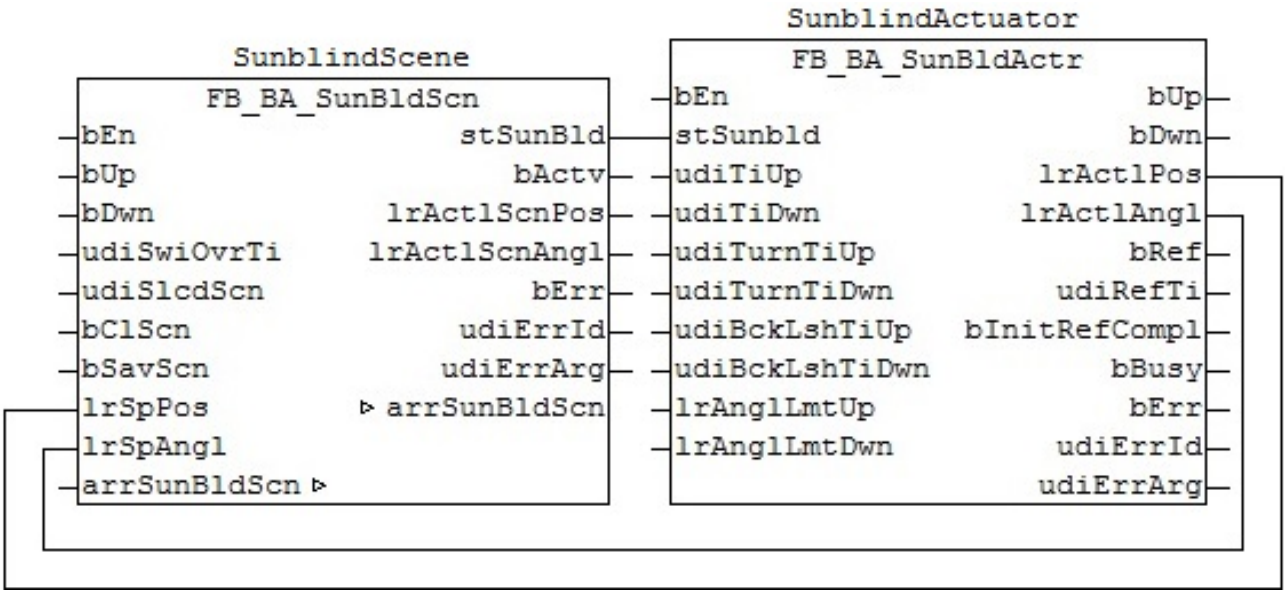
### Linking to the blind function block

Like the "normal" manual mode function block FB_BA_SunBldSwi [▶ 297], the scene selection function block can be connected either via an upstream priority control FB_BA_SunBldPrioSwi4 [▶ 291] or FB_BA_SunBldPrioSwi8 [▶ 293], or directly via the blind function block. The link is based on the positioning telegram stSunBld [▶ 331]. 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/outputs

### VAR_INPUT

```
bEn             : BOOL;
bUp             : BOOL;
bDwn            : BOOL;
udiSwiOvrTi     : UDINT;
udiSlcdScn      : UDINT;
bClScn          : BOOL;
bSavScn         : BOOL;
lrSpPos         : LREAL;
lrSpAngl        : LREAL;
```

**bEn:** the function block has no function if this input is FALSE. 0 is output for the position and the angle in the positioning telegram stSunBld [▶ 331] - *bManualMode* 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:** command input blind up

**bDwn:** command input blind down

**udiSwiOvrTi:** time [ms] until the corresponding manual command in the positioning telegram stSunBld [▶ 331] switches to latching mode, if the command input is activated permanently.

**udiSlcdScn:** selected scene which should either be saved (*bSaveScene*) or called (*bInvokeScene*).

**bClScn:** call selected scene

**bSavScn:** save selected scene

**lrSpPos:** 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 [▶ 284] or FB_BA_RolBldActr [▶ 276] of the reference blind/ roller blind, in order to be able to save a position that was previously approached manually.

**lrSpAngl:** ditto slat angle [°]

### VAR_IN_OUT

```
arrSunBldScn : ARRAY[0..cMaxSunBldScn] OF ST_BA_SunBldScn;
```

**arrSunBldScn:** Table with the scene entries of the type ST_BA_SunBldScn [▶ 331]. Up to 21 scenes can be stored (0..20).

### VAR_OUTPUT

```
stSunBld       : ST_BA_SunBld;
bActv          : BOOL;
lrActlScnPos   : LREAL;
lrActlScnAngl  : LREAL;
bErr           : BOOL;
udiErrId       : UDINT;
udiErrArg      : UDINT;
```

**stSunBld:** positioning telegram, see ST_BA_SunBld [▶ 331]

**bActv:** corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 331] and is solely used to indicate whether the function block sends an active telegram.

**lrActlScnPos:** indicates the saved relative blind height position [%] for the currently selected scene.

**lrActlScnAngl:** ditto slat angle [°]

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

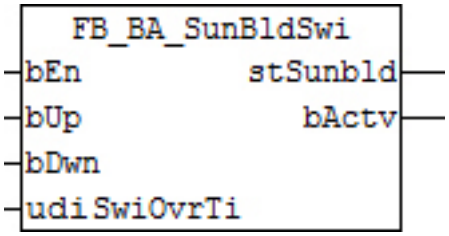
If an error should occur, then this automatic function is deactivated and position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview [▶ 45]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.118 FB_BA_SunBldSwi

With the help of this function block the blind controller FB_BA_SunBldActr [▶ 284] or the roller blind controller FB_BA_RolBldActr [▶ 276] can be controlled in manual operation mode. The connection takes place via the positioning telegram stSunBld [▶ 331] either directly or with an additional priority controller.




### Functional description

Structure of the blind positioning telegram stSunBld [▶ 331].

```
TYPE ST_BA_SunBld:
STRUCT
     lrPos         : LREAL;
     lrAngl        : LREAL;
     bManUp        : BOOL;
     bManDwn       : BOOL;
     bManMod       : BOOL;
     bActv         : BOOL;
END_STRUCT
END_TYPE
```

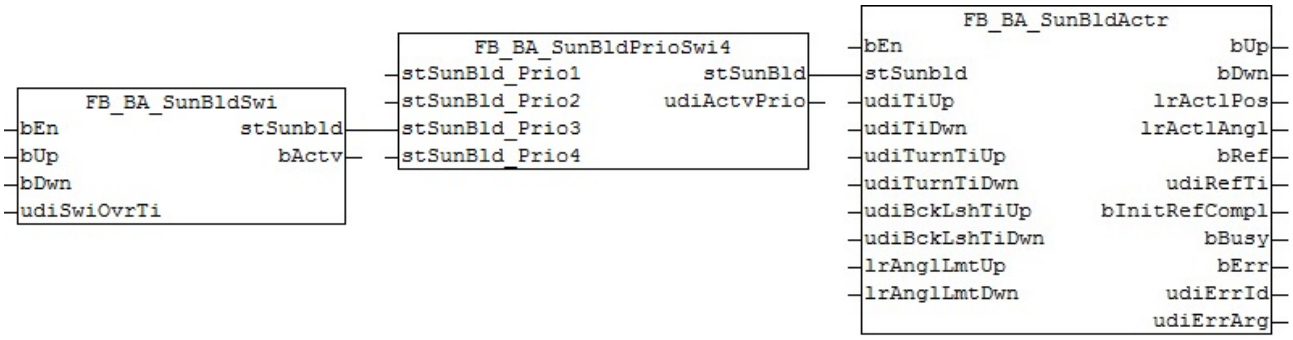
### Operation

In manual mode, the function block controls the blind function block FB_BA_SunBldActr [▶ 284] or the roller blind function block FB_BA_RolBldActr [▶ 276] 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 for longer than the entered time *udiSwiOvrTi* [ms], the corresponding control command latches. Activating a command input again clears this latching.
When the function block is activated via *bEn*, the bit *bActv* is set immediately in the positioning telegram. The function block uses this to notify a priority switch (FB_BA_SunBldPrioSwi4 [▶ 291] or FB_BA_SunBldPrioSwi8 [▶ 293]) of its priority over lower priorities. The function block only knows two modes: manual movement via *bUp* and *bDwn*, with the result that the bit *bManMod* is set in the positioning telegram, thereby preventing acceptance of automatic positioning commands by the actuator function block. The second mode is the scene call via *bClScn*. Via the positioning telegram, this specifies a position and an angle for the scene selected via *udiSlcdScn* (selection of the field *arrSunBldScn*). The manual mode bit in the positioning telegram *bManMod* is deleted, and the actuator function block knows that it has to move to a particular position. The input *bClScn* is regarded as subordinate to the manual commands. If, in the first cycle of the function block activation (*bEn*) neither *bUp*, nor *bDwn*, nor *bClScn* are set, the function block "jumps" to manual mode (positioning telegram: bit *bManMod*) and in this case behaves just like FB_BASunBldSwi [▶ 297].

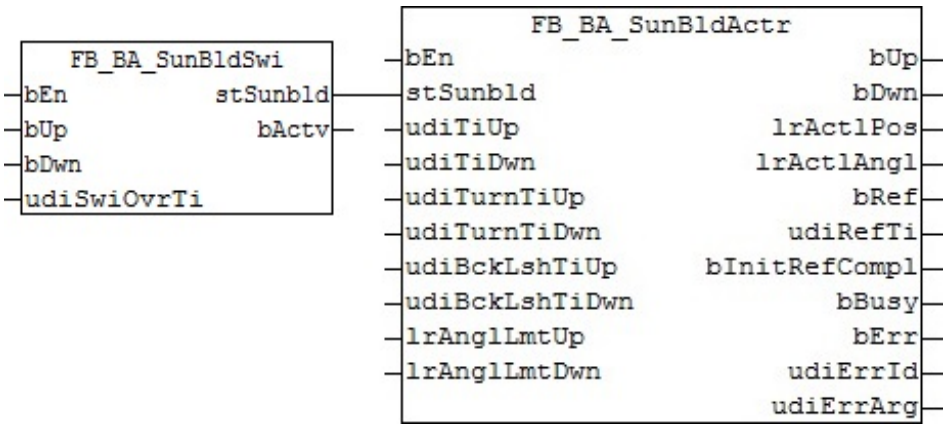
### Linking to the blind function block

The manual mode function block can be connected either via an upstream priority control FB_BA_SunBldPrioSwi4 [▶ 291] or FB_BA_SunBldPrioSwi8 [▶ 293], or directly at the blind function block. The link is based on the positioning telegram stSunBld [▶ 331].

**Use of a priority controller:**




**Direct connection:**




**Inputs/outputs**

**VAR_INPUT**

```
bEn            : BOOL;
bUp            : BOOL;
bDwn           : BOOL;
udiSwiOvrTi    : UDINT;
```

**bEn:** the function block has no function if this input is FALSE. 0 is output for the position and the angle in the positioning telegram stSunBld [▶ 331] - *bManMod* and *bAct* 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:** command input blind up

**bDwn:** command input blind down

**udiSwiOvrTi:** time [ms] until the corresponding manual command in the positioning telegram stSunBld [▶ 331] switches to latching mode, if the command input is activated permanently.

**VAR_OUTPUT**

```
stSunBld    : ST_BA_SunBld;
bActv       : BOOL;
```

**stSunBld:** positioning telegram, see ST_BA_SunBld [▶ 331]

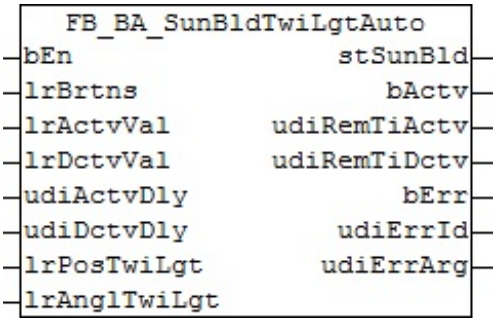
**bActv:** corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 331] and is solely used to indicate whether the function block sends an active telegram.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.119 FB_BA_SunBldTwiLgtAuto

This function block controls the blind if the outdoor brightness has fallen below a limit value.




### Functional description

The automatic twilight function operates with both a value hysteresis and a temporal hysteresis: If the outdoor brightness value *lrBrtns* [lux] falls below the value *lrActvVal* [lux] for the time *udiActvDly* [s], the function block is active and will provide the blind positions *lrPosTwiLgt* (height [%]) and *lrAnglTwiLgt* (louvre angle [°]) at the output in the positioning telegram stSunBld [▶ 331], as specified in the IN variables. Conversely, if the outdoor brightness exceeds the value *lrDctvVal* [lux] for the time *udiDctvDly* [s], then the automatic function is no longer active. The active flag in the positioning telegram stSunBld [▶ 331] 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/outputs

### VAR_INPUT

```
bEn           : BOOL;
lrBrtns       : LREAL;
lrActvVal     : LREAL;
lrDctvVal     : LREAL;
udiActvDly    : UDINT;
udiDctvDly    : UDINT;
lrPosTwiLgt   : LREAL;
lrAnglTwiLgt  : LREAL;
```

**bEn:** the function block has no function if this input is FALSE. In the positioning telegram stSunBld [▶ 331], 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.

**lrBrtns:** outdoor brightness [lx]

**lrActvVal:** activation limit value [lx]

**lrDctvVal:** deactivation limit value [lx]

**udiActvDly:** activation delay [s]

**udiDctvDly:** deactivation delay [s]

**lrPosTwiLgt:** vertical position of the blind [%] if the twilight automatic is active

**lrAnglTwiLgt:** slat angle of the blind [°] if the twilight automatic is active

### VAR_OUTPUT

```
stSunBld        : ST_BA_SunBld;
bActv           : BOOL;
udiRemTiActv    : UINT;
udiRemTiDctv    : UINT;
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**stSunBld:** output structure of the blind positions, see ST_BA_SunBld [▶ 331]

**bActv :** corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 331] and is solely used to indicate whether the function block sends an active telegram.

**udiRemTiActv:** shows the time remaining [s] after falling below the switch value *lrActvVal* until automatic mode is activated. This output is 0 as long as no countdown of the time is taking place.

**udiRemTiDctv:** shows the time remaining [s] after exceeding the switch value *lrDctvVal* until automatic mode is disabled. This output is 0 as long as no countdown of the time is taking place.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

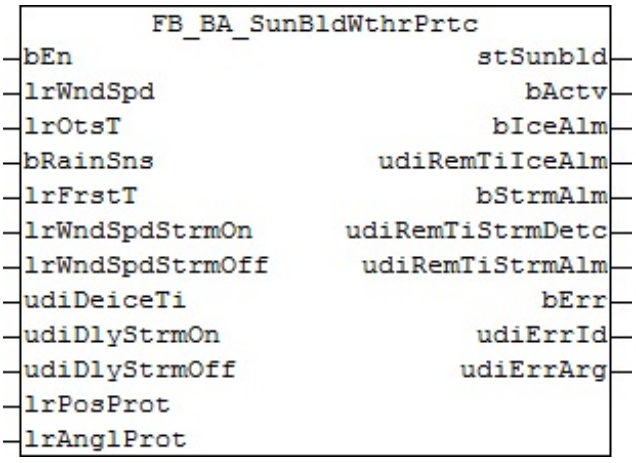
If an error should occur, then this automatic function is deactivated and position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview [▶ 45]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.120 FB_BA_SunBldWthrPrtc

The weather protection has the highest priority in the blind controller (see overview [▶ 45]) and is intended to ensure that the blind is not damaged by ice or wind.




### Functional description

The task of the automatic weather protection function is to protect the blind against two impending dangers and, in order to do so, to drive it to a safe position:

- **Icing up:** impending icing up is detected when the measured outside temperature *lrOtsT* falls below the frost limit value *lrFrstT* 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 *udiDeiceTi* [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.
- **Storm:** if the measured wind speed lies above the value *lrWndSpdStrmOn* for the time *udiDlyStrmOn* [s], then it is assumed that a storm is directly impending. Only if the wind speed falls below the value *lrWndSpdStrmOff* for the time *udiDlyStrmOff* [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 both cases of danger the blind is driven to the protection position specified by *lrPosProt* (height position in percent) and *lrAnglProt* (slat angle [°]).

### Inputs/outputs

### VAR_INPUT

```
bEn              : BOOL;
lrWndSpd         : LREAL;
lrOtsT           : LREAL;
bRainSns         : BOOL;
lrFrstT          : LREAL;
lrWndSpdStrmOn   : LREAL;
lrWndSpdStrmOff  : LREAL;
udiDeiceTi       : UINT;
udiDlyStrmOn     : UDINT;
udiDlyStrmOff    : UDINT;
lrPosProt        : LREAL;
lrAnglProt       : LREAL;
```

**bEn:** the function block has no function if this input is FALSE. In the positioning telegram stSunBld [▶ 331], 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.

**lrWndSpd:** 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.

**lrOtsT:** outside temperature [°C]

**bRainSns:** input for a rain sensor

**lrFrstT:** icing-up temperature limit value [°] Celsius. This value may not be greater than 0. Otherwise an error is output.

**lrWndSpdStrmOn:** 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 *lrWndSpd*. A value greater than this limit value triggers the alarm after the entered time *udiDlyStrmOn*.

**lrWndSpdStrmOff:** 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 *lrWndSpd*. A value smaller than or equal to this limit value resets the alarm after the entered time *udiDlyStrmOff*.

**udiDeiceTi:** time until the deicing of the blind after icing up [s]. After that the icing up alarm is reset.

**udiDlyStrmOn:** time delay until the storm alarm is triggered [s]

**udiDlyStrmOff:** time delay until the storm alarm is reset [s]

**lrPosProt:** height position of the blind [%] in the case of protection

**lrAnglProt:** slat angle of the blind [°] in the case of protection

### VAR_OUTPUT

```
stSunBld          : ST_BA_SunBld;
bActv             : BOOL;
bIceAlm           : BOOL;
udiRemTiIceAlm    : UDINT;
bStrmAlm          : BOOL;
udiRemTiStrmDetc  : UDINT;
udiRemTiStrmAlm   : UDINT;
bErr              : BOOL;
udiErrId          : UDINT;
udiErrArg         : UDINT;
```

**stSunBld:** output structure of the blind positions, see ST_BA_SunBld [▶ 331]

**bActv:** corresponds to the boolean value *bActv* in the blind telegram ST_BA_SunBld [▶ 331] and is solely used to indicate whether the function block sends an active telegram.

**bIceAlm:** displays the icing-up alarm.

**udiRemTiIceAlm:** 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 (*lrFrstT*), 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.

**bStrmAlm:** displays the storm alarm.

**udiRemTiStrmDetc:** in an uncritical case this second counter constantly displays the alarm delay time *udiDlyStrmOn*. If the measured wind speed *lrWndSpd* is above the activation limit value *lrWndSpdStrmOn*, the seconds to the alarm are counted down. This output is 0 as long as no countdown of the time is taking place.

**udiRemTiStrmAlm:** as soon as the storm alarm is triggered, this second counter first constantly displays the deactivation time delay of the storm alarm *udiDlyStrmOff*. If the measured wind speed *lrWndSpd* falls below the deactivation limit value *lrWndSpdStrmOff*, 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.

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

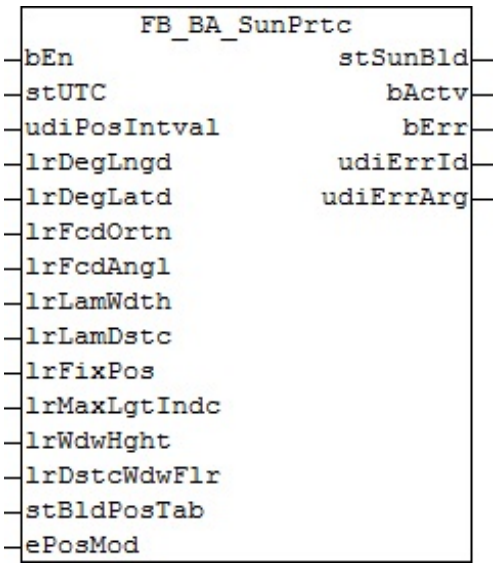
If an error should occur, then this automatic function is deactivated and position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview [▶ 45]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.121 FB_BA_SunPrtc

Function block for the control of glare protection with the aid of a slatted blind




**Functional description**

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 *lrFixPos*.
2. The blind position is varied as a function of the sun position. The position is specified in the table (stBldPosTab [▶ 325]). See also description of FB_BA_BldPosEntry [▶ 254].
3. The vertical blind position 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 *lrMaxLgtIndc*.

In order to avoid excessive repositioning of the slat angle, the variable *udiPosIntval* 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 *bEnable* must be TRUE.
2. The sun must have risen. (elevation > 0)
3. The function block is parameterized correctly (*bErr* = FALSE)

The function block FB_BA_SunPrtc enables glare protection in two ways, which work in parallel:

- Lamella setpoint tracing, so that the direct incident light cannot quite enter through the lowered part of the blind.
- Control of the blind height, with 3 different possibilities ( Adjustable via the enumerator at *ePosMod*) :
  1) fixed blind height, i.e. no change (preset)
  2) blind height depending on the sun position, defined via a table (stBldPosTab [▶ 325]), see also description of FB_BA_BldPosEntry [▶ 254]
  3) maximum desired light incidence

On the basis of the parameters entered, which are described further below, the function block calculates the necessary slat angle and blind position and transfers them to the output structure stSunBld [▶ 331]. Of course, the output does not take place continuously, since a constant blind movement would be perceived as distracting. At the input *udiPosIntval* it is possible to set in minutes the interval at which new position values are to be output.
However, the shading criteria must always be fulfilled between two positioning times: no direct light may pass through the slats and the desired light incidence through the blind height must remain limited, assuming

initially that the blind height is controlled via the "maximum desired light incidence" mode. Therefore, two blind and slat positions are calculated internally: the one for the current switching point and the one for the next switching point. The position in which the blind is more closed is then the valid position.

The positioning in intervals starts precisely when the following three conditions are satisfied:

- The input *bEn* must be TRUE.
- The function block must not be in an error state due to incorrect parameterization (*bErr*=TRUE).
- The sun must have risen, i.e. the sun elevation must be greater than 0°. This is an internal safety query, since the limitation to at least 0° should actually be done by the user by programming the input *bEn*; see Overview of automatic sun protection (shading correction) [▶ 45].

If these three conditions are not satisfied, then the active bit (*bActv*) is set to FALSE, the blind height to 0% and the slat angle to 0% in the positioning structure.

### Error handling

The following invalid inputs were detected:

Always:

- The duration of the positioning interval is zero or it exceeds 720 min.
- The longitude entered is not within the valid range from -180° to 180°.
- The latitude entered is not within the valid range from -90° to 90°.
- The value entered for the facade inclination *lrFcdAngl* is outside the valid range of -90°..90°.
- The value for the louvre spacing (*lrLamDstc*) is greater than or equal to the value for the louvre width (*lrLamWdth*). This does not represent a "valid" blind since the louvres cannot close fully. Mathematically, this would lead to errors.
- The value entered for the louvre spacing *lrLamDstc* is zero.
- The value entered for the louvre width *lrLamWdth* is zero.

Only if "fixed blind height" positioning is selected - *ePosMod=ePosModFix:*

- The value entered for the fixed blind height (*lrFixPos*) is greater than 100 or less than 0.

Only if "maximum light incidence" is selected - *ePosMod=ePosModMaxIndc:*

- The bit "values valid" (*bVld*) in the positioning table *stBldPosTab* is not set - invalid values: see *FB_BA_BldPosEntry*.

Only if "maximum light incidence" is selected - *ePosMod=ePosModMaxIndc*.

- The value entered for the window height *lrWdwHght* is less than or equal to zero.
- The distance between lower window edge and floor *lrDstcWdwFlr* that was entered is less than zero.
- The value entered for the maximum required light incidence *lrMaxLgtIndc* is less than or equal to zero.

### Inputs/outputs

### VAR_INPUT

```
bEn            : BOOL;
stUTC          : TIMESTRUCT;
udiPosIntval   : UDINT;
lrDegLngd      : LREAL;
lrDegLatd      : LREAL;
lrFcdOrtn      : LREAL;
lrFcdAngl      : LREAL;
lrLamWdth      : LREAL;
lrLamDstc      : LREAL;
lrFixPos       : LREAL;
lrMaxLgtIndc   : LREAL;
lrWdwHght      : LREAL;
lrDstcWdwFlr   : LREAL;
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 [▶ 331] 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 universal time (UTC - Universal Time Coordinated, previously referred to as GMT, Greenwich Mean Time). The function block FB_BA_GetTime [▶ 320] 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:** positioning interval in minutes - time between two blind position outputs. Valid range: 1 min...720 min.

**lrDegLngd:** longitude [°]. Valid range: - 180°...180°

**lrDegLatd:** latitude [°]. Valid range: - 90°...90°

**lrFcdOrtn:** 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° |

**lrFcdAngl:** facade inclination [°]. See facade inclination

**lrLamWdth:** width of the slats in mm, see sketch

**lrLamDstc:** slat spacing in mm, see sketch

**lrFixPos:** fixed (constant) blind height [0…100 %]. Applies if *ePosMod* = *ePosModFix* (see enumerator E_BA_PosMod [▶ 324]).

**lrMaxLgtIndc:** maximum desired incidence of light in mm measured from the outside of the wall (see Height adjustment). With the aid of the parameters *lrWdwHght* and *lrDstcWdwFlr*, a calculation is performed in relation to the position of the sun to determine how high the blind must be so that the incidence of light does not exceed the value *lrMaxLgtIndc*. Valid if *ePosMod* = *ePosModeMaxIncidence* (see enumerator E_BA_PosMod [▶ 324]).

**lrWdwHght:** window height in mm for the calculation of the blind height if the mode "maximum desired incidence of light" is selected.

**lrDstcWdwFlr:** 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:** 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 [▶ 324]). For a more detailed description please refer to FB_BA_BldPosEntry [▶ 254].

**ePosMod:** selection of the positioning mode, see enumerator E_BA_PosMod [▶ 324].

### VAR_OUTPUT

```
stSunBld    : ST_BA_SunBld;
bActv       : BOOL;
bErr        : BOOL;
udiErrorId  : UDINT;
udiErrArg   : UDINT;
```

**stSunBld:** Output structure of the blind positions, see ST_BA_SunBld [▶ 331]

**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.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

If an error should occur, then this automatic function is deactivated and 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 [▶ 45]) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

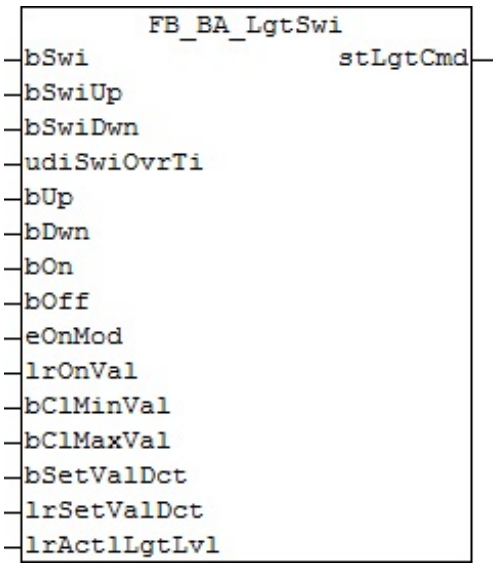
### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.122 FB_BA_LgtSwi

Switching and dimming function block - function for two light switches

This function block requires light value feedback at input lrActlLgtLvl, in order to be able to change the light level (toggle command). Without this feedback, which either comes from a single device or as an average from a group, the switch function block does not know the logical state of the light and is unable to toggle to the other state when the keyboard shortcut bSwi is pressed. An example can be found below.




### Functional description

This function block is a combined switch/dimmer.

Various modes (*eOnMod*) are defined for central switching on at *bOn*, which can be selected via this enumerator input:

- eBA_ClMaxVal : The light level is set directly to the maximum value applicable at the light actuator function block.
- eBA_ClMinVal : The light level is set directly to the minimum value applicable at the light actuator function block.
- eBA_ClOnVal : The light level is set directly to the value that is active at input *lrOnVal*.
- eBA_ClRstVal : The light level is set directly to the reset value applicable at the light actuator function block. This is the value that was applicable before the last switch-off. Application sample: The light is switched back on again in a conference room after a presentation.

All other commands see below.

The function block sends its commands via the command telegram stLgtCmd [▶ 333].

### Inputs/outputs

### VAR_INPUT

```
bSwi          : BOOL;
bSwiUp        : BOOL;
bSwiDwn       : BOOL;
udiSwiOvrTi   : UDINT;
bUp           : BOOL;
bDwn          : BOOL;
bOn           : BOOL;
bOff          : BOOL;
eOnMod        : E_BA_LgtOnMode;
lrOnValue     : LREAL;
bClMinVal     : BOOL;
bClMaxVal     : BOOL;
bSetValDct    : BOOL;
lrSetValDct   : LREAL;
lrActlLgtLvl  : LREAL;
```

**bSwi:** switch input on/off, dimming up/down: a short signal (< *udiSwiOvrTi*) turns the light off or on depending on the state. If the switch is kept pressed, the light is alternately dimmed up or down. Dimming up ends when the maximum value is reached, and dimming down ends when the minimum value is reached. The dimming direction is not reversed automatically. If the light is switched off, it is first set to the minimum value, before it is dimmed up.

**bSwiUp:** switch input On/Off, dimming up: a short signal (< *udiSwiOvrTi*) turns the light off or on depending on the state. If the switch is kept pressed, the light is dimmed up. Dimming up ends when the maximum value is reached. If the light is switched off, it is first set to the minimum value, before it is dimmed up.

**bSwiDwn:** switch input On/Off, dimming down: a short signal (< *udiSwiOvrTi*) turns the light off or on depending on the state. If the switch is kept pressed, the light is dimmed down. Dimming down stops when the minimum value is reached. If the light is switched off, it is first set to the maximum value, before it is dimmed down.

**udiSwiOvrTi:** switching time [ms] between short/long switch pressing detection, and therefore between switching and dimming.

**bUp:** dimming up. This input has **no** switching function and is **not** affected by the switching time *udiSwiOvrTi*. The function has no effect when the light is switched off.

**bDwn:** dimming down. This input has **no** switching function and is **not** affected by the switching time *udiSwiOvrTi*. The function has no effect when the light is switched off.

**bOn:** central switch-on. The switch-on value depends on the choice of *eOnMod*, see below.

**bOff:** central switch-off

**eOnMod:** selection of the switch-on value (E_BA_LgtOnMode [▶ 332])

**lrOnVal:** [0...100%] switch-on value, if a central switch-on command comes via *bOn* and mode *eBA_ClOnVal* is selected at *eOnMod*.

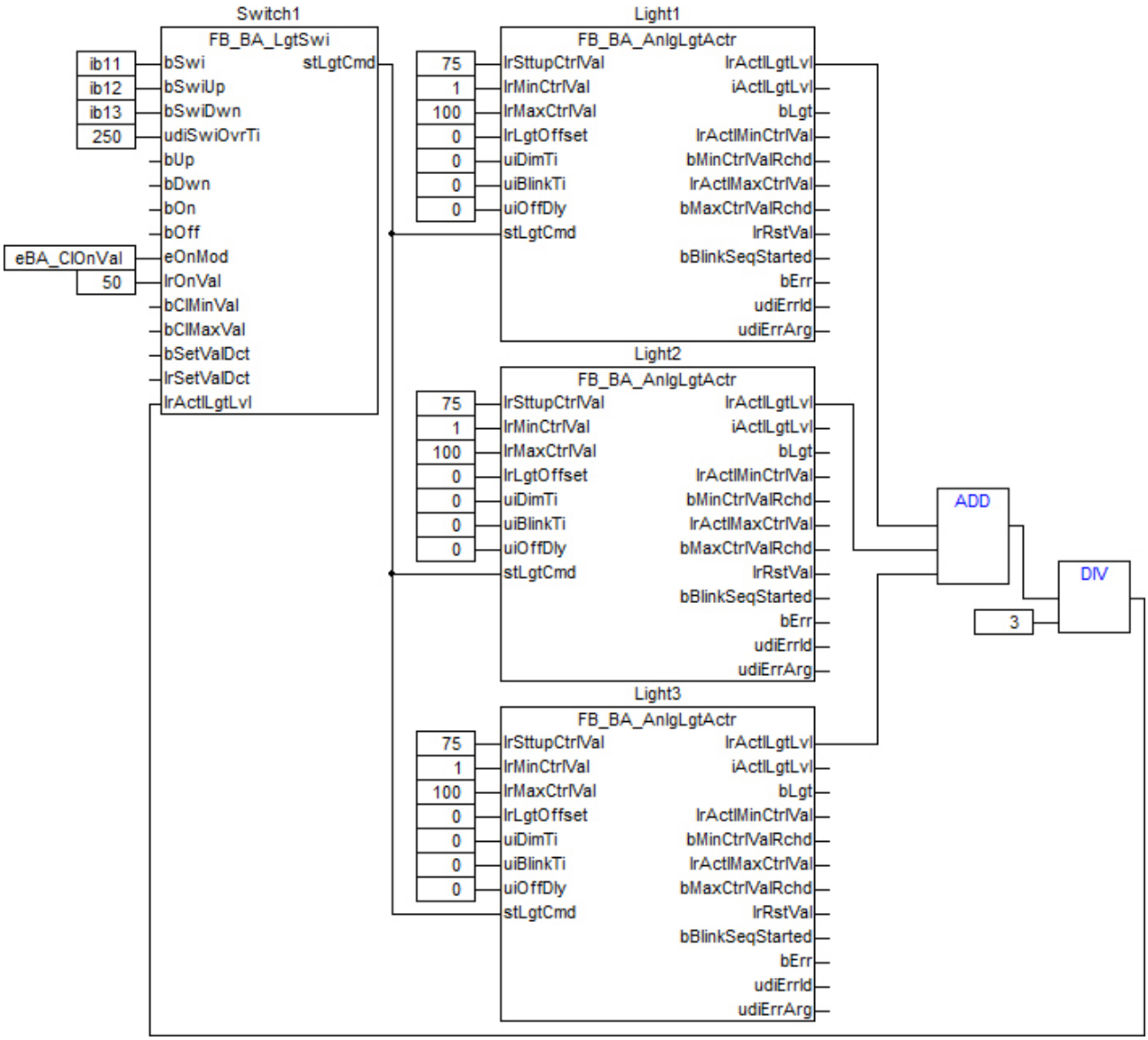
**bClMinVal:** the light actuator function block is instructed to switch to the minimum value.

**bClMaxVal:** the light actuator function block is instructed to switch to the maximum value.

**bSetValDct/lrSetValDct:** switches the light actuator function block to the value *lrSetValDct* [0..100%]. **The value *lrLgtOffset* is applied to this value at the light actuator.**

**lrActlLgtLvl:** [0...100%] this input is used for toggle evaluation. For a group of lamps, it is recommended to create either the average of all actuators or the light value of the group master. The assignment of this input is obligatory, since without this feedback the switch function block does not know the logical light state of the single device or the group.

**Sample:**




## VAR_OUTPUT

```
stLgtCmd : ST_BA_LgtCmd;
```

**stLgtCmd:** command telegram of type ST_BA_LgtCmd [▶ 333]

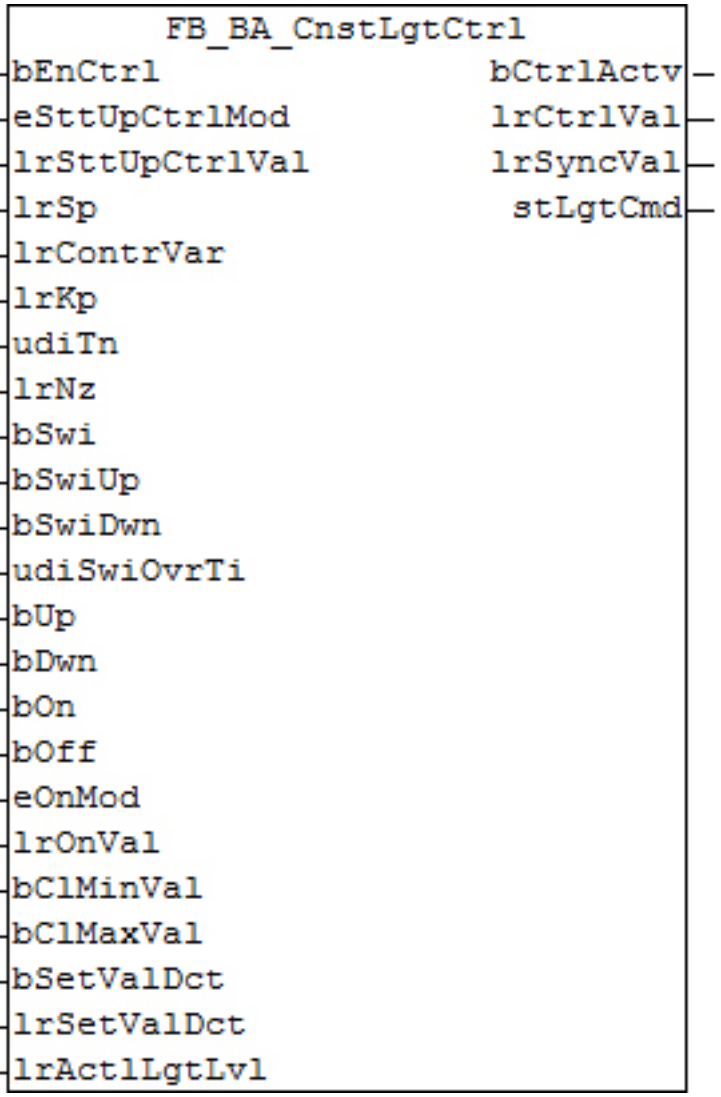
### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.123 FB_BA_CnstLgtCtrl

Constant light control for a room

> This function block requires light value feedback at input lrActlLgtLvl, in order to be able to change the light level (toggle command). Without this feedback, which either comes from a single device or as an average from a group, the switch function block does not know the logical state of the light and is unable to toggle to the other state when the keyboard shortcut bSwi is pressed. See also FB_BA_LgtSwi [▶ 306].




### Functional description

This function block is a can be combined switch/dimmer with additional constant light control mode. Internally, the constant light control uses a PI controller.

Various modes (*eOnMod*) are defined for central switching on at *bOn*, which can be selected via this enumerator input:

- eBA_ClMaxVal : The light level is set directly to the maximum value applicable at the light actuator function block.
- eBA_ClMinVal: The light level is set directly to the minimum value applicable at the light actuator function block.
- eBA_ClOnVal : The light level is set directly to the value that is active at input *lrOnVal*.
- eBA_ClRstVal := The light level is set directly to the reset value applicable at the light actuator function block. This is the value that was applicable before the last switch-off. Application example: The light is switched back on again in a conference room after a presentation.

All other commands see below.

The function block sends its commands via the command telegram stLgtCmd [▶ 333].

### Inputs/outputs

### VAR_INPUT

```
bEnCtrl         : BOOL;
eSttUpCtrlMod   : E_BA_CtrlSttupMod;
lrSttUpCtrlVal  : LREAL;
lrSp            : LREAL;
lrContrVar      : LREAL;
lrKp            : LREAL;
udiTn           : UDINT;
lrNz            : LREAL;
bSwi            : BOOL;
bSwiUp          : BOOL;
bSwiDwn         : BOOL;
udiSwiOvrTi     : UDINT;
bUp             : BOOL;
bDwn            : BOOL;
bOn             : BOOL;
bOff            : BOOL;
eOnMod          : E_BA_LgtOnMode;
lrOnValue       : LREAL;
bClMinVal       : BOOL;
bClMaxVal       : BOOL;
bSetValDct      : BOOL;
lrSetValDct     : LREAL;
lrActlLgtLvl    : LREAL;
```

**bEnCtrl:** a positive edge on this input switches the control on. **If the switching inputs** *bSwi...bSetValDct* **are actuated, the control is disabled again, irrespective of whether a TRUE signal is present at input** *bEnCtrl***.**

**eSttUpCtrlMod/lrSttUpCtrlVal:** specifies with which value the controller should start up: see E_BA_CtrlSttupMod [▶ 332].

**lrSp:** setpoint input [lx]

**lrContrVar:** actual value input [lx]

**lrKp:** gain factor of the internal PI controller

**udiTn:** integral action time (integration time) of the internal PI controller [ms]

**lrNz:** neutral zone: if the absolute value of the control deviation is less than *lrNz*/2, the controller stops its internal control value calculation, but remains active.

**bSwi:** switch input on/off, dimming up/down: a short signal (< *udiSwiOvrTi*) turns the light off or on depending on the state. If the switch is kept pressed, the light is alternately dimmed up or down. Dimming up ends when the maximum value is reached, and dimming down ends when the minimum value is reached. The dimming direction is not reversed automatically. If the light is switched off, it is first set to the minimum value, before it is dimmed up.

**bSwiUp:** switch input On/Off, dimming up: a short signal (< *udiSwiOvrTi*) turns the light off or on depending on the state. If the switch is kept pressed, the light is dimmed up. Dimming up ends when the maximum value is reached. If the light is switched off, it is first set to the minimum value, before it is dimmed up.

**bSwiDwn:** switch input On/Off, dimming down: a short signal (< *udiSwiOvrTi*) turns the light off or on depending on the state. If the switch is kept pressed, the light is dimmed down. Dimming down stops when the minimum value is reached. If the light is switched off, it is first set to the maximum value, before it is dimmed down.

**udiSwiOvrTi:** switching time [ms] between short/long switch pressing detection, and therefore between switching and dimming.

**bUp:** dimming up. This input has **no** switching function and is **not** affected by the switching time *udiSwiOvrTi*. The function has no effect when the light is switched off.

**bDwn:** dimming down. This input has **no** switching function and is **not** affected by the switching time *udiSwiOvrTi*. The function has no effect when the light is switched off.

**bOn:** central switch-on. The switch-on value depends on the choice of *eOnMod*, see below.

**bOff:** central switch-off.

**eOnMod:** selection of the switch-on value (E_BA_LgtOnMode [▶ 332]).

**lrOnVal:** [0...100%] switch-on value, if a central switch-on command comes via *bOn* and mode *eBA_ClOnVal* is selected at *eOnMod*.

**bClMinVal:** the light actuator function block is instructed to switch to the minimum value.

**bClMaxVal:** the light actuator function block is instructed to switch to the maximum value.

**bSetValDct/lrSetValDct:** switches the light actuator function block to the value *lrSetValDct* [0..100%]. **Note**: The value *lrLgtOffset* is applied to this value at the light actuator.

**lrActlLgtLvl:** [0...100%] this input is used for toggle evaluation. For a group of lamps, it is recommended to create either the average of all actuators or the light value of the group master.

### VAR_OUTPUT

```
bCtrlActv : BOOL;
lrCtrlVal : LREAL;
stLgtCmd  : ST_BA_LgtCmd;
```

**bCtrlActv:** constant light regulation is active

**lrCtrlVal:** current value that is output in command telegram stLgtCmd (of type ST_BA_LgtCmd) [▶ 333].

**lrSyncVal:** [0..100%]  for control: current start value selected through mode *eSttUpCtrlMod* (see inputs).

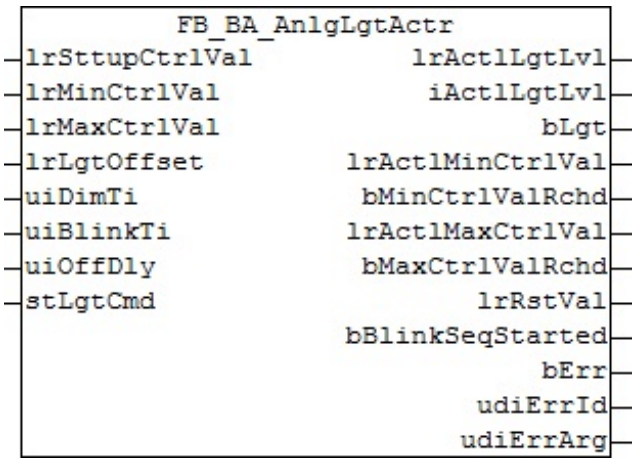
**stLgtCmd:** command telegram of type ST_BA_LgtCmd. [▶ 333]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.124 FB_BA_AnlgLgtActr

Function block for controlling an analog light actuator, e.g. via a KL2751




### Functional description

The function block receives its commands via the command telegram stLgtCmd. [▶ 333]
As a conventional analog control function block, it has light value outputs as LREAL value [0..100%], as INT value [0..32767] and as Boolean value (ON/OFF), which enables switching of an OFF relay for certain types of lamps.

This function block must be called in every PLC cycle, since the PLC cycle time is included in the calculation of the dimming ramp.

### Inputs/outputs

#### VAR_INPUT

```
lrSttupCtrlVal : LREAL;
lrMinCtrlVal   : LREAL;
lrMaxCtrlVal   : LREAL;
lrLgtOffset    : LREAL;
udiDimTi       : UDINT;
uiBlinkTi      : UINT;
uiOffDly       : UINT;
stLgtCmd       : ST_BA_LgtCmd;
```

**lrSttupCtrlVal :** light value after restart. [*lrMinCtrlVal*..*lrMaxCtrlVal*]. Input in percentage (0%..100%). This value is only used in the first call cycle. Incorrect entries less than *lrMinCtrlVal* or greater than *lrMaxCtrlVal* are automatically limited.

**lrMinCtrlVal:** minimum light output value. [1..*lrMaxCtrlVal*]. Input in percentage (1%..100%). A change in this value takes effect immediately. Incorrect entries less than 1.0 or greater than 100.0 are automatically limited. For verification, the resulting value is output at *lrActlMinCtrlVal*; note that value changes via the command telegram *stLgtCmd* are also accepted. In each case the last change is valid.

**lrMaxCtrlVal:** maximum light output value. [*lrMinCtrlVal*..100%]. Input in percentage (1%..100%). A change in this value takes effect immediately. Incorrect entries less than *lrMinCtrlVal* or greater than 100.0 are automatically limited.
For verification, the resulting value is output at *lrActlMaxCtrlVal*; note that value changes via the command telegram *stLgtCmd* are also accepted. In each case the last change is valid.

**lrLgtOffset:** option to increase or reduce the brightness value. This only takes effect in conjunction with the command *bSetCtrlValDct* in the command structure *stLgtCmd*. In this case, the light actuator is set to *lrSetCtrlValDct+lrLgtOffset*. The value cannot exceed the set maximum value or fall below the minimum value. Switch-off is possible, if *lrSetCtrlValDct+lrLgtOffset* is less than or equal to zero.

**uiDimTi:** dimming ramp [ms]: time in which dimming from 0 to 100% takes place.

**uiBlinkTi / udiOffDly:** these two times [s] can be used to implement a switch-off warning, e.g. for staircase lighting. These two intervals **additionally** take effect at the time when the lamp usually switches off: the light is switched off for the time *uiBlinkTi* [s] and switched on again for the time *uiOffDly* [s].

**stLgtCmd:** command telegram of type ST_BA_LgtCmd [▶ 333]

### VAR_OUTPUT

```
lrActlLgtLvl    : LREAL;
iActlLgtLvl     : INT;
bLgt            : BOOL;
lrActlMinCtrlVal: LREAL;
bMinCtrlValRchd : BOOL;
lrActlMaxCtrlVal: LREAL;
bMaxCtrlValRchd : BOOL;
lrRstVal        : LREAL;
bErr            : BOOL;
udiErrId        : UDINT;
udiErrArg       : UDINT;
```

**lrActlLgtLvl:** current control value [0..100%]. Calculation see below.

**iActlLgtLvl:** current control value [0..32767]

**bLgt:** current control value (ON / OFF)

**lrActlMinCtrlVal:** current min value (described via input or command structure)

**bMinCtrlValRchd:** the actuator has reached its minimum value.

**lrActlMaxCtrlVal:** current max value (described via input or command structure)

**bMaxCtrlValRchd:** the actuator has reached its maximum value.

**lrRstVal:** last value before off as resetting value, i.e. the current control value is stored here before switch-off.

**bErr:** light function block in error state

**udiErrId / udiErrArg:** error number and argument for determining the cause; see error codes [▶ 334].

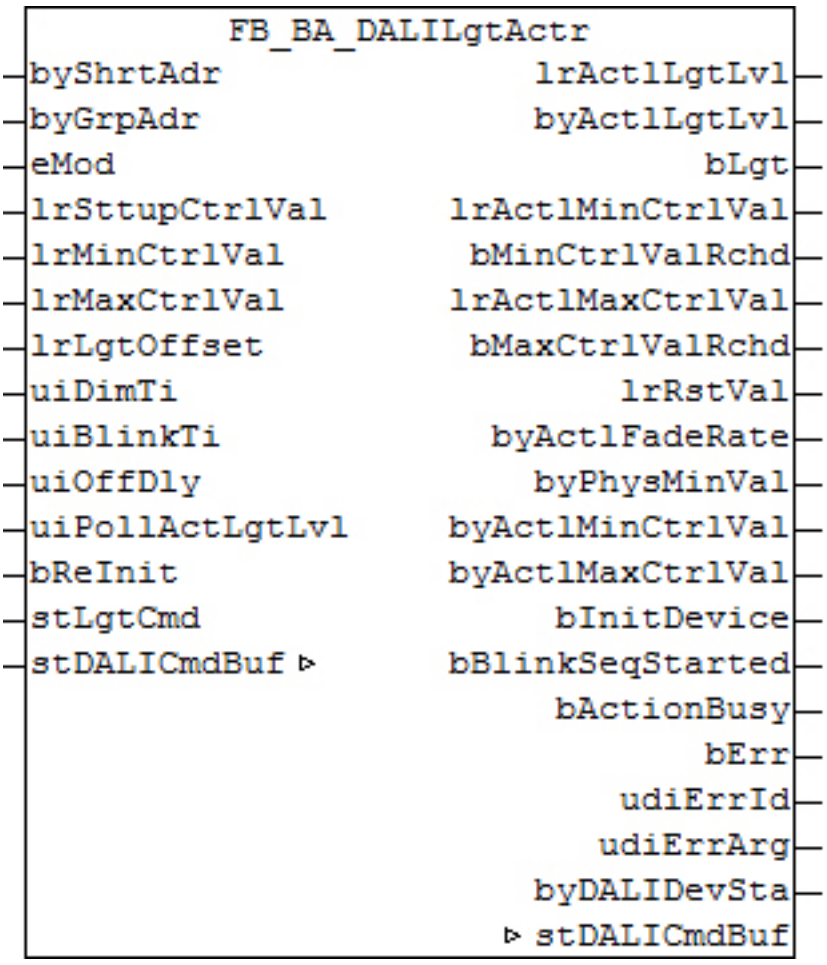
### Requirements

The warning messages for auto-correcting the minimum and maximum values are only active for one PLC cycle, since the correction function rectifies the error itself.

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.125 FB_BA_DALILgtActr

Function block for controlling a DALI light actuator




### Functional description

The function block receives its commands via the command telegram stLgtCmd [▶ 333].
As an IN/OUT variable and light control function block for the DALI subsystem, it contains a reference to the command memory structure *stDALICmdBuf*.

### Inputs/outputs

### VAR_INPUT

```
byShrtAdr        : BYTE;
byGrpAdr         : BYTE;
eMod             : BOOL;
lrSttupCtrlVal   : LREAL;
lrMinCtrlVal     : LREAL;
lrMaxCtrlVal     : LREAL;
lrLgtOffset      : LREAL;
uiDimTi          : UINT;
uiBlinkTi        : UINT;
uiOffDly         : UINT;
uiPollActLgtLvl  : UINT;
bReinit          : BOOL;
stLgtCmd         : ST_BA_LgtCmd;
```

**byShrtAdr :** short address of the DALI control gear to be addressed. This address must be in the valid range of 0..63, since the function block needs an existing control gear to query the state. If an invalid address is entered, the function block is not executed, and a corresponding error code is issued. The short address relates to the DALI commands resulting from the inputs at this function block:

- lrSttupCtrlVal (light value after restart)
- lrMinCtrlVal (sets minimum light output value)
- lrMaxCtrlVal (sets maximum light output value)
- uiDimTi (defines the dimming ramp -> change in FadeRate)

**byGrpAdr:** group address. Only taken into account if the control gear is operated as a master by this function block: *eMod=eBA_DALIModGrpMst* or *eMod=eBA_DALIModGrpSgl.*

Even if this function block should control a group as a master, the short address of the master must always be entered at *byShrtAdr*, since data is read from the corresponding control gear as a representative of the entire group.

**eMod:** selection of the individual/group or master/slave behavior of the function block; see E_BA_DALIMod [▶ 334].

**lrSttupCtrlVal:** light value after restart

**lrMinCtrlVal:** minimum light output value. [0..100%]. A change at the function block input causes a re-parameterization (short address) in the control gear. There, the lower limit is set to the physical minimum value, the upper limit to the maximum value.

**lrMaxCtrlVal:** maximum light output value. [0..100%]. A change at the function block input causes a re-parameterization (short address) in the control gear. There, the lower limit is set to the physical minimum value.

**lrLgtOffset:** option to increase or reduce the brightness value. This can be useful in a lamp group circuit for lamps that are situated near daylight and have to be less bright than the lamps in darker areas of the room. However, the light value cannot be controlled by this parameter via *lrMaxCtrlVal* or at *lrMinCtrlVal*.
This offset only affects commands that reach the function block via the command structure as *bSetCtrlValDct*.

**udiDimTi:** dimming ramp [s]: time in which dimming from 0 to 100% takes place. This is converted to a DALI FadeRate within the function block and stored in the control gear.

**uiBlinkTi / uiOffDly:** these two times [ms] can be used to implement a switch-off warning, e.g. for staircase lighting. These two intervals **additionally** take effect at the time when the lamp usually switches off: the light is switched off for the time *uiBlinkTi* [s] and switched on again for the time *uiOffDly* [s].

**uiPollActLgtLvl:** cycle time [s] required to read the current actual value (ACTUAL DIM LEVEL) in the background. So that the dimming of the lamps is not disturbed, reading always has the lowest priority. If the value is set to 0, reading is prohibited. In addition to the current light value, the status byte of the control gear is read and written to output *byDALIDevSta*.

**bReInit:** edge-triggered input: restarts the initialization [▶ 315] of the function block.

**stLgtCmd:** command telegram of type ST_BA_LgtCmd. [▶ 333]

### VAR_OUTPUT

```
lrActlLgtLvl     : LREAL;
byActlLgtLvl     : BYTE;
bLgt             : BOOL;
lrActlMinCtrlVal : LREAL;
bMinCtrlValRchd  : BOOL;
lrActlMaxCtrlVal : LREAL;
bMaxCtrlValRchd  : BOOL;
lrRstVal         : LREAL;
byActlFadeRate   : BYTE;
byPhysMinVal     : BYTE;
byActlMinCtrlVal : BYTE;
byActlMaxCtrlVal : BYTE;
bInitDevice      : BOOL;
bBlinkSeqStarted : BOOL;
bActionBusy      : BOOL;
bErr             : BOOL;
udiErrId         : UDINT;
udiErrArg        : UDINT;
byDALIDevSta     : BYTE;
```

**lrActlLgtLvl:** current light output value [0..100%]

**byActlLgtLvl:** current light output value [0..254]

**bLgt:** light is switched on or off

**lrActlMinCtrlVal:** current valid minimum value. The output of this value is only verified in the initialization phase of the function block (and therefore the respective control gear) through a QueryMinLevel query. Subsequently, the value is output at the output that is programmed in the control gear via the function block input or the command structure. It can be limited to the physical minimum or maximum value.

**bMinCtrlValRchd:** the actuator has reached its minimum value.

**lrActlMaxCtrlVal:** current valid maximum value. The output of this value is only verified in the initialization phase of the function block (and therefore the respective control gear) through a QueryMinLevel query. Subsequently, the value is output at the output that is programmed in the control gear via the function block input or the command structure. It can be limited to the physical minimum or to 100%.

**bMaxCtrlValRchd:** the actuator has reached its maximum value.

**lrRstVal:**this variable is used to store the light value before switching off. The command *bClRstVal* in the command structure can be used to restore this value. Application example: the light is switched back on again in a conference room after a presentation.

**byActlFadeRate:** currently applicable DALI FadeRate, which results from the dimming time (input *uiDimTi*).

**byPhysMinVal:** physical minimum value [1..254] read during the initialization phase of the function block.

**byActlMinCtrlVal:** see *lrActlMinCtrlVal*. Converted from [0..100%] to the range [0..254].

**byActlMaxCtrlVal:** see *lrActlMaxCtrlVal*. Converted from [0..100%] to the range [0..254].

**bInitDevice:** light function block in the initialization phase: the following actions are executed:

- The light-ramp (*uiDimTi*) present at the input is programmed into the control gear (short address) as DALI FadeRate.
- The minimum value (*lrMinCtrlVal*) present at the input is programmed into the control gear (short address) as DALI MinValue.
- The maximum value (*lrMaxCtrlVal*) present at the input is programmed into the control gear (short address) as DALI MaxValue.
- The physical minimum value of the DALI control gear is read and output at the function block output.
- The actually valid minimum value of the DALI control gear is read and output at the function block output.
- The actually valid maximum value of the DALI control gear is read and output at the function block output.
- The control gear (short address) is set to the starting value (*lrSttupCtrlVal*) present at the input.
- The actually valid light output value of the DALI control gear is read and output at the function block output.

**bBlinkSeqStarted:** the flash sequence has been started.

**bActionBusy:** a DALI command is executed.

**bErr:** light function block in error state

**udiErrId / udiErrArg:** error number and argument for determining the cause; see error codes [▶ 334].

**byDALIDevSta:** status byte of the DALI control gear. It is read in the background, together with the automatic query of the light value; see input *uiPollActLgtLvl*.

### VAR_IN_OUT

```
stDALICmdBuf : ST_DALIV2CommandBuffer;
```

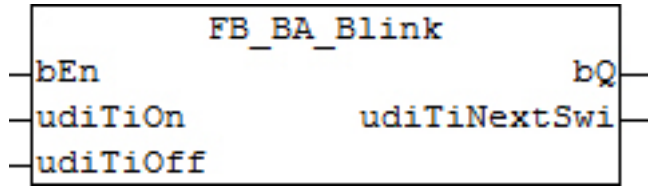
**stDALICmdBuf:** reference to the command memory structure stDALICmdBuf

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBASubsystems library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

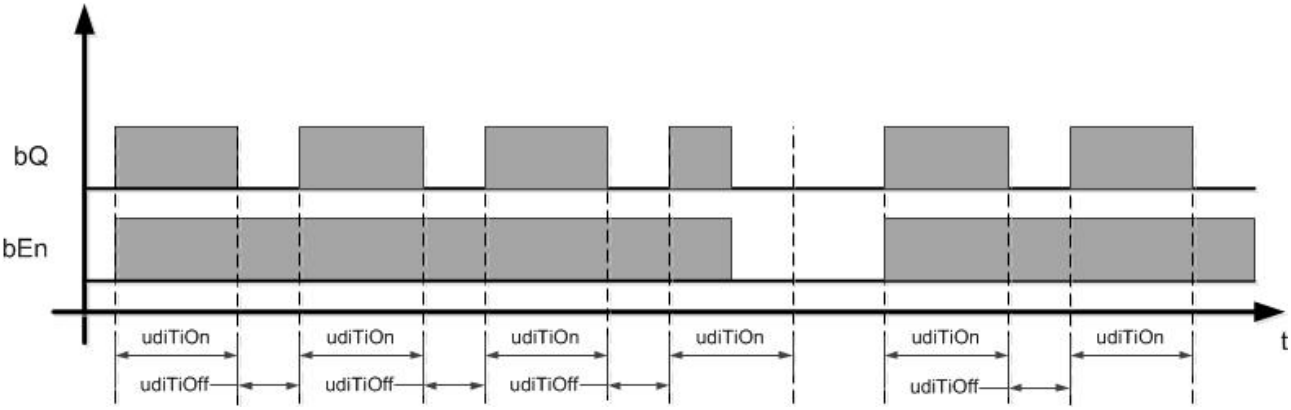
## 8.2.126 FB_BA_Blink

Simple oscillator function block




### Functional description

This function block is an oscillator with configurable pulse and pause time, *udiTiOn* and *udiTiOff* [ms]. It is enabled with a TRUE signal at *bEn* and starts with the pulse phase.




*udiTiNextSwi* is a countdown value [s] to the next change of *bQ*.

### Inputs/outputs

#### VAR_INPUT

```
bEn          : BOOL;
udiTiOn      : UDINT;
udiTiOff     : UDINT;
```

**bEn:** function block enable

**udiTiOn:** pulse time [ms]

**udiTiOff:** pause time [ms]

#### VAR_OUTPUT

```
bQ           : BOOL;
udiTiNextSwi  : UDINT;
```

**bQ:** oscillator output.

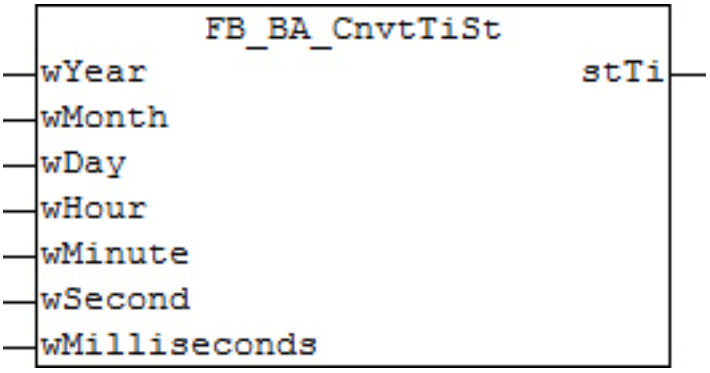
**udiTiNextSwi:** countdown to next change of *bQ* [s]

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.127 FB_BA_CnvtTiSt

Assembling a time structure




### Functional description

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 inputs, e.g. a value of 99 for the hour, since it make sense for this verification to take place in the connected function blocks, which have to verify 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;
```

**nYear:** year (1970...2106)

**wMonth:** month (1..12)

**wDay:** day of the month (1…31)

**wHour**: hour (0..23)

**wMinute:** minutes (0..59)

**wSecond s:** seconds (0..59)

**wMillisecond:** milliseconds (0..999)

### VAR_OUTPUT

```
stTi      : TIMESTRUCT;
```

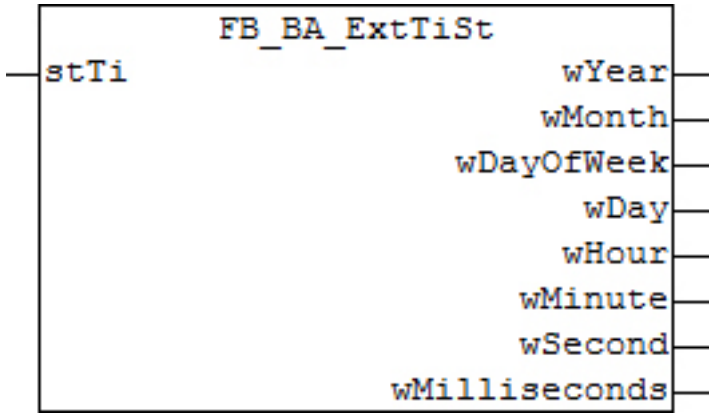
**stTi:** output: time structure

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.128 FB_BA_ExtTiSt

Time structure resolution




### Functional description

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/outputs

#### VAR_INPUT

```
stTi      : TIMESTRUCT;
```

**stTi:** Input: Time structure

#### VAR_OUTPUT

```
wYear         : WORD;
wMonth        : WORD;
wDay          : WORD;
wHour         : WORD;
wMinute       : WORD;
wSecond       : WORD;
wMilliseconds : WORD;
```

**nYear:** year (1970...2106)

**wMonth:** month (1..12)

**wDay:** day of the month (1…31)

**wHour**: hour (0..23)

**wMinute:** minutes (0..59)

**wSecond ys:** seconds (0..59)

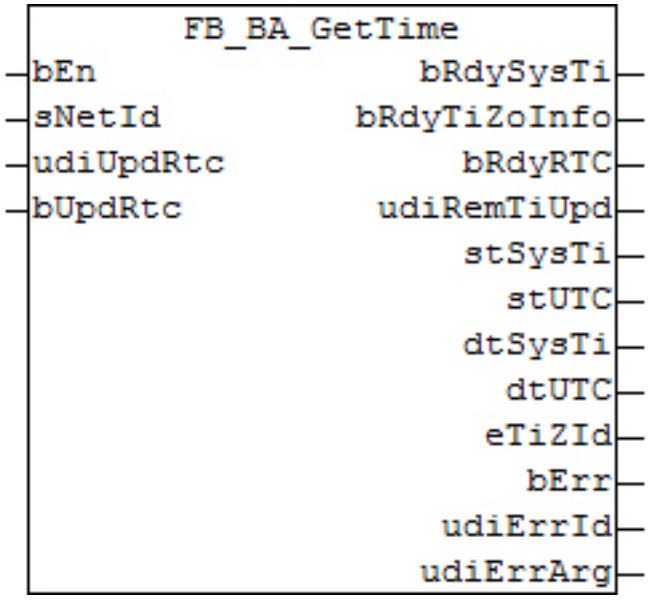
**wMillisecond:** milliseconds (0..999)

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.129 FB_BA_GetTime

Reading the system time




### Functional description

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. 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 *stSysTi* for the read system time and *stUtcTi* 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]); 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* indicates the seconds remaining to the next read cycle. The time structures that are output, *stSysTi* and *stUtcTi*, can be resolved with the aid of the function block FB_BA_ExtTiSt [▶ 319] into the components day, month, hour, minute etc.

**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 udiErrId / udiErrArg. The countdown output is not restarted until the wait cycle starts.

### Inputs/outputs

#### VAR_INPUT

```
bEn       : BOOL;
sNetId    : T_AmsNetId;
udiUpdRtc : UDINT;
bUpdRtc   : BOOL;
```

**bEn:** Enables the function block. If *bEn* = TRUE, then the RTC clock is initialised with the NT system time.

**sNetId :** This parameter can be used to specify the AmsNetID of the TwinCAT computer, whose NT system time is to be read as timebase. If it is to be run on the local computer, an empty string can be entered.

**udiUpdRtc:** Time specification [s] with which the RTC clock is regularly synchronised 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 *tGetSystemTime,* the RTC clock can be synchronised via a positive edge at this input.

#### VAR_OUTPUT

```
bRdySysTi     : BOOL;
bRdyTiZoInfo  : BOOL;
bRdyRTC       : BOOL;
udiRemTiUpd   : UDINT;
stSysTi       : TIMESTRUCT;
stUTC         : TIMESTRUCT;
dtSysTi       : DT;
dtUTC         : DT;
eTiZId        : E_TimeZoneID;
bErr          : BOOL;
udiErrId      : UDINT;
udiErrArg     : UDINT;
```

**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:** countdown to next synchronization/update of the time information.

**stSysTi:** system time of the read target system. The time structure can be resolved with the aid of the function block FB_BA_ExtTiSt [▶ 319] 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 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 [▶ 319] 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:** same as *stSysTi* / *stUTC*, but in DATE-AND-TIME format: year-month-day-hour-minute-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*.

**eTiZId:** enumerator for summer/winter time information

**bErr:** this output is switched to TRUE if the parameters entered are erroneous.

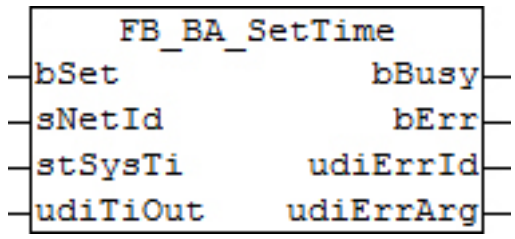
**udiErrId / udiErrArg:** contains the error number and the error argument. See error codes [▶ 334].

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.2.130 FB_BA_SetTime

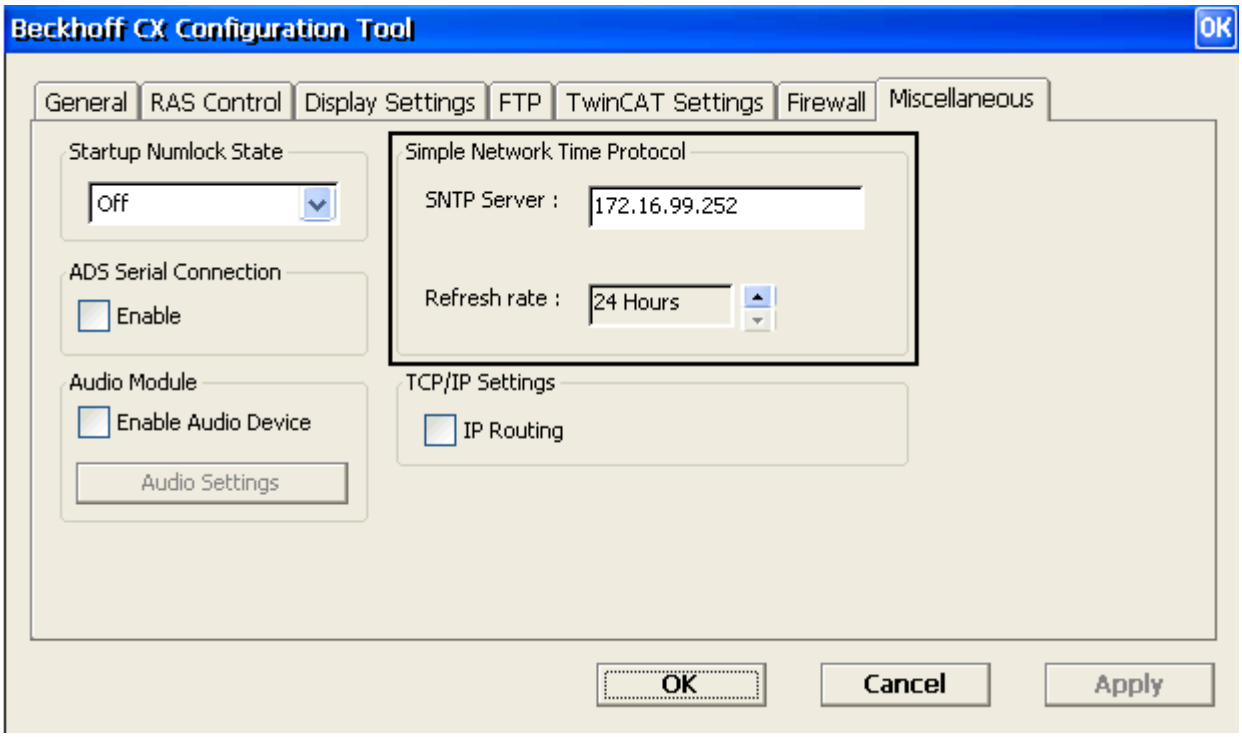
Setting the system time




### Functional description

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 *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. 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/outputs

### VAR_INPUT

```
bSet          : BOOL;
sNetId        : T_AmsNetId;
stSysTi       : TIMESTRUCT;
udiTiOut      : 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. If applicable, an empty string *sNetId* := ''; can be specified for the local computer.

**stS ysTi** 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 [▶ 318], which brings the subvariables of date and time in a structure together.

**udiTiOut:** indicates the timeout time [s], which must not be exceeded during execution.

### VAR_OUTPUT

```
bBusy             : BOOL;
bError            : BOOL;
udiErrorId        : UDINT;
bInvalidParameter : BOOL;
```

**bBusy:** If the function block is activated via a rising edge at *bSetLocalTime,* this output is set and remains set until feedback occurs.

**bError:** This output is set if an error occurs during the transfer of the NT system time. *bError* is reset when the function block is activated via the input variable *bSetLocalTime*.

**bErr:** This output is set to TRUE, if either the system time to be transferred is faulty (see below) or an ADS error occurs during the transfer. Both error types are reflected in a clear description of the output variables *udiErrId* / *udiErrArg*.

**udiErrId / udiErrArg:** Contains the error number and the error argument. See Error codes [▶ 334].

### Time specification limits

The created time structure *stSysTi* is checked for limits within the function block:

**Year:** 1970..2106 - this range refers to the maximum time domain (years) that can be displayed in the TwinCAT system

**Month:** 1..12

**Day of the month:** 1..28/29/30/31, depending on month and leap year - both are determined and considered

**Hour:** 0..23

**Minute:** 0..59

**Second:** 0..59

**Millisecond:** 0..999

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

# 8.3 Enumerations and structs

## 8.3.1 E_BA_AlmSta

Enumerator status of the alarm messages

```
TYPE E_BA_AlmSta :
(
     E_BA_AlmSta_UNKOWN        := 0,
     E_BA_AlmSta_OK            := 1,
     E_BA_AlmSta_TRIGGERED     := 2,
     E_BA_AlmSta_GONE          := 3,
     E_BA_AlmSta_ACKNOWLEDGED  := 4,
     E_BA_AlmSta_OFFLINE       := 5
);
END_TYPE
```

**E_BA_AlmSta_UNKOWN:** the alarm was not assigned to a system.

**E_BA_AlmSta_OK:** no alarm pending.

**E_BA_AlmSta_TRIGGERED:** an alarm was triggered.

**E_BA_AlmSta_GONE:** the alarm was cleared.

**E_BA_AlmSta_ACKNOWLEDGED:** the alarm has been acknowledged but is still pending.

**E_BA_AlmSta_OFFLINE:** the alarm is offline.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.2 E_BA_PosMod

Enumerator for the definition of the positioning mode

```
TYPE E_BA_PosMod :
(
     eBA_PosModFix:= 0,
     eBA_PosModTab,
     eBA_PosModMaxIndc
);
END_TYPE
```

**eBA_PosModFix:** the blind height adopts a fixed value, which is set (in %) via the *IrFixPos* value on the function block FB_BA_SunPrtc [▶ 302].

**eBA_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 [▶ 254].

**eBA_PosModMaxIndc:** the positioning takes place with specification of the maximum desired incidence of light.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.3 E_BA_ShdObjType

Enumerator for selecting the shading object type

```
TYPE E_BA_ShdObjType :
(
     eBA_ObjTypeTetragon := 0,
     eBA_ObjTypeGlobe    := 1
);
END_TYPE
```

**eBA_ObjTypeTetragon:** object type is a rectangle

**eBA_ObjTypeGlobe:** object type is a globe

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.4 ST_BA_BldPosTab

Structure of the interpolation point entries for the height adjustment of the blind

```
TYPE ST_BA_BldPosTab:
STRUCT
     lrSunElv    : ARRAY[0..5] OF LREAL;
     lrBlindPos  : ARRAY[0..5] OF LREAL;
     bValid      : BOOL;
     byDummy     : ARRAY[1..3] OF BYTE;
END_STRUCT
END_TYPE
```

**lrSunElv / lrBlindPos:** the 6 interpolation points that are transferred, wherein the array elements 0 and 5 represent the automatically generated edge elements mentioned above.

**bValid :** validity flag for the function block FB_BA_SunPrtc [▶ 302]. It is set to TRUE by the function block FB_BA_BldPosEntry [▶ 254] if the data entered correspond to the validity criteria described.

**byDummy:** unused filling variables for creating a 4-byte configuration. This configuration is important, if this structure part of a data exchange between controllers has a different architecture (ARM/I486).

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.5 ST_BA_Cnr

Information about window corners

```
TYPE ST_BA_Cnr :
STRUCT
     lrX      : LREAL;
     lrY      : LREAL;
     bShdd    : BOOL;
     byDummy  : ARRAY[1..3] OF BYTE;
END_STRUCT
END_TYPE
```

**lrX:** X-coordinate of the window (on the facade)

**lrY:** Y-coordinate of the window (on the facade)

**bShdd:** information whether this corner point is shaded: *bShdd*=TRUE: corner point is shaded.

**byDummy:** unused filling variables for creating a 4-byte configuration. This configuration is important, if this structure part of a data exchange between controllers has a different architecture (ARM/I486).

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.6 ST_BA_ComnMsg

Structure of the communication telegram to the function blocks of type FB_BA_ComMsg [▶ 197].

```
TYPE ST_BA_ComnMsg :
STRUCT
   arrData      : ARRAY [1..gBa_cMaxArrComnMsg] OF ST_BA_ComnMsgData;
   iCntEntryArr : INT;
   bDummy01     : BOOL;
   bDummy02     : BOOL;
END_STRUCT
END_TYPE
```

```
TYPE ST_BA_ComnMsgData :
STRUCT
   arrActvPrio : ARRAY [1..16] OF BOOL;
   udiActvPrio : UDINT;
   dwObjId     : DWORD;(*neu*)
   eObjType    : E_BACnetObjectType;
   iId         : INT;
   bAck        : BOOL;
   bInAlm      : BOOL;
   bOoServ     : BOOL;
   bOvrrd      : BOOL;
   bFlt        : BOOL;
   bHiLmt      : BOOL;
   bLoLmt      : BOOL;
   bErr        : BOOL;
END_STRUCT
END_TYPE
```

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.7 ST_BA_ComnMsgTermt

Structure of the communication telegram to the function blocks of type FB_BA_ComMsgTermt [▶ 200].

```
TYPE ST_BA_ComnMsgTermt :
STRUCT
   arrData      : ARRAY [1..gBa_cMaxArrComnMsgTermt] OF ST_BA_ComnMsgTermtData;
   iCntEntryArr : INT;
   bDummy01     : BOOL;
   bDummy02     : BOOL;
END_STRUCT
END_TYPE
```

```
TYPE ST_BA_ComnMsgTermtData :
STRUCT
   arrActvPrio : ARRAY [1..16] OF BOOL;
   udiActvPrio : UDINT;
```

```
   iId         : INT;
   bAck        : BOOL;
   bInAlm      : BOOL;
   bOoServ     : BOOL;
   bOvrrd      : BOOL;
   bFlt        : BOOL;
   bHiLmt      : BOOL;
   bLoLmt      : BOOL;
   bErr        : BOOL;
END_STRUCT
END_TYPE
```

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.8 ST_BA_FcdElem

List entry for a facade element (window)

```
TYPE ST_BA_FcdElem:
STRUCT
    lrWdwWdth : LREAL;
    lrWdwHght : LREAL;
    stCnr     : ARRAY [1..4] OF ST_BA_Cnr;
    usiGrp    : USINT;
    bVld      : BOOL;
    byDummy   : ARRAY[1..2] OF BYTE;
END_STRUCT
END_TYPE
```

**lrWdwWdth:** width of the window

**lrWdwHght:** 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 [▶ 325].

**bVld:** plausibility of the entered data: *bVld*=TRUE: data are plausible.

**usiGrp:** group membership of en element

**byDummy:** unused filling variables for creating a 4-byte configuration. This configuration is important, if this structure part of a data exchange between controllers has a different architecture (ARM/I486).

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.9 ST_BA_SeqLink / ST_BA_SeqLinkData

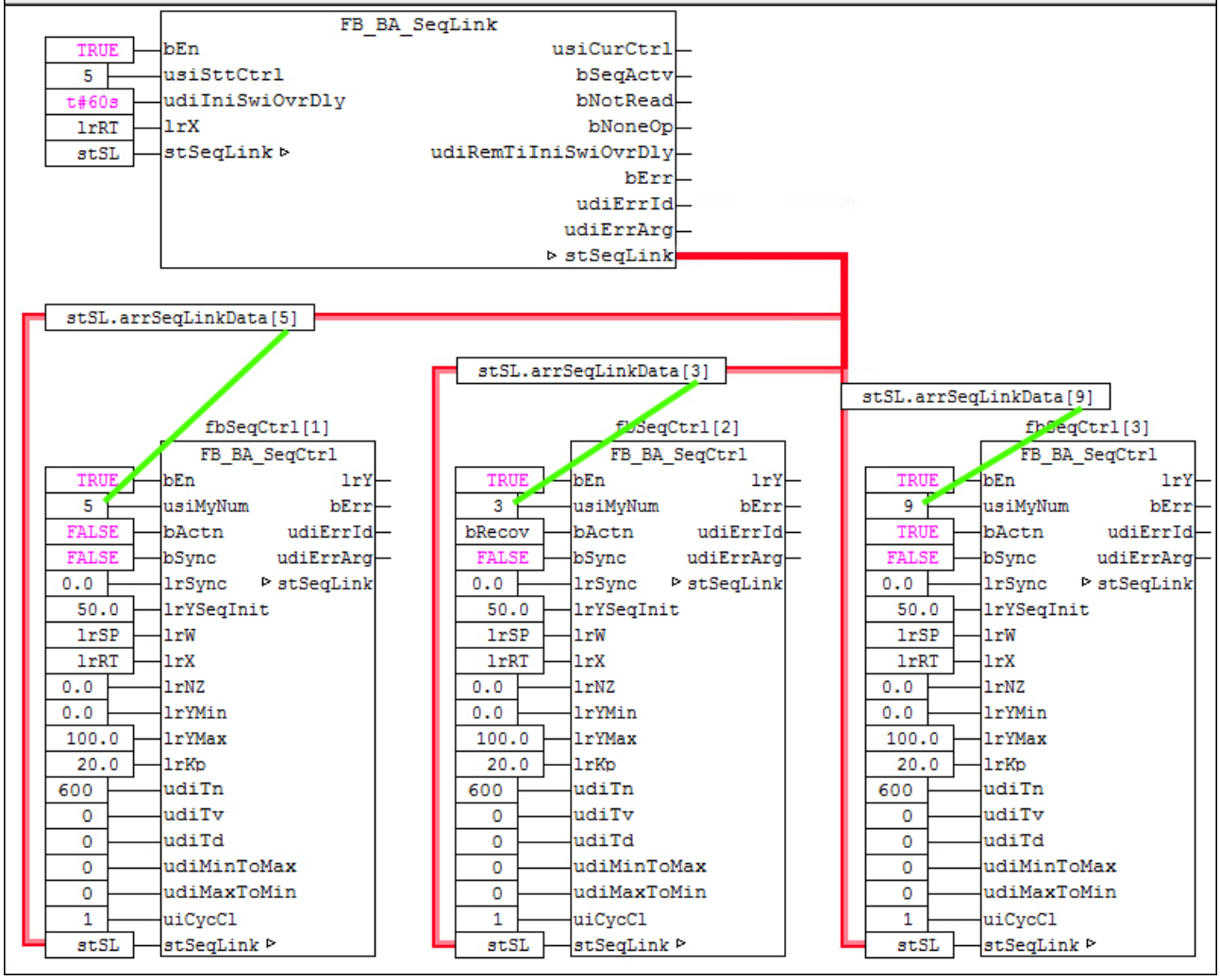
Structure of the data and command exchange between the control function block FB_BA_SeqLink [▶ 168] and the sequence controllers FB_BA_SeqCtrl [▶ 165].

This structure has to be created once per sequence control:

```
stSeqLink : ST_BA_SeqLink;
```

Within this structure, a further field structure is declared automatically, through which the sequence link function block and the individual sequence controllers exchange all relevant data. Each sequence controller writes its data into the field element corresponding to its ordinal number (entry at input *usiMyNum* at the sequence controller function block). It is always the complete structure with all field elements that is linked to the function blocks.

```
0001 PROGRAM SEQUENZ
0002 VAR
0003     fbSeqLink       : FB_BA_SeqLink;
0004     fbSeqCtrl       : ARRAY[1..3] OF FB_BA_SeqCtrl;
0005     stSL            : ST_BA_SeqLink;
0006 END_VAR
0007
```




The structures have the following setup:

```
TYPE ST_BA_SeqLink :
STRUCT
  arrSeqLinkData : ARRAY[1..cMaxSeqCtrl] OF ST_BA_SeqLinkData;
  bSeqActv       : BOOL;
  usiCurSeq      : USINT;
  byDummy        : ARRAY[1..2] OF BYTE;
END_STRUCT
END_TYPE
```

**bSeqActv:** the sequence control is enabled and active.

**usiCurSeq:** from FB_BA_SeqLink: specification of current sequence controllers

**byDummy:** unused filling variables for creating a 4-byte configuration. This configuration is important, if this structure part of a data exchange between controllers has a different architecture (ARM/I486).

```
TYPE ST_BA_SeqLinkData:
STRUCT
  lrY         : LREAL;
  lrYMin      : LREAL;
  lrYMax      : LREAL;
  lrW         : LREAL;
  bActn       : BOOL;
  bOp         : BOOL;
  bPresence   : BOOL;
  bErrDouble  : BOOL;
  usiCurSeq   : USINT;
  byDummy     : ARRAY[1..3] OF BYTE;
END_STRUCT
END_TYPE
```

**lrY:** from FB_BA_SeqCtrl: transfer of current control value

**lrYMin:** from FB_BA_SeqCtrl: transfer of minimum control value

**lrYMax:** from FB_BA_SeqCtrl: transfer of maximum control value

**lrW:** from FB_BA_SeqCtrl: transfer of current setpoint

**bActn:** from FB_BA_SeqCtrl: transfer of inverse control direction (*bActn* = FALSE: heating mode - *bActn* = TRUE: cooling mode).

**bOp:** from FB_BA_SeqCtrl: sequence controller is enabled, i.e. its input *bEn* is set to TRUE.

**bPresence:** from FB_BA_SeqCtrl: check bit, see below.

**bErrDouble:** from FB_BA_SeqCtrl: error during number verification: there are at least 2 sequence controllers of the same ordinal number *usiMyNum.*

**usiCurSeq:** from FB_BA_SeqLink: specification of current sequence controllers

**byDummy:** unused filling variables for creating a 4-byte configuration. This configuration is important, if this structure part of a data exchange between controllers has a different architecture (ARM/I486).

**Note regarding check bit:**

Each sequence controller sets the *bPresence* flag in the structure that is applicable to it. If the flag is already set, *usiMyNum* must inevitably have been assigned twice, which means that two sequence controllers access the same structure. After the evaluation, the sequence link function block resets all check bits, so that this test takes place cyclically. This means that an error can automatically be rectified via an online change, and new sequence controllers can be added, if required

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.10 ST_BA_ShdObj

List entry of a shading object

```
TYPE ST_BA_ShdObj :
STRUCT
     lrP1x      : LREAL;
     lrP1y      : LREAL;
     lrP1z      : LREAL;
     lrP2x      : LREAL;
     lrP2y      : LREAL;
     lrP2z      : LREAL;
     lrP3x      : LREAL;
     lrP3y      : LREAL;
     lrP3z      : LREAL;
     lrP4x      : LREAL;
```

```
    lrP4y      : LREAL;
    lrP4z      : LREAL;
    lrMx       : LREAL;
    lrMy       : LREAL;
    lrMz       : LREAL;
    lrRads     : LREAL;
    usiBegMth  : USINT;
    usiEndMth  : USINT;
    eType      : E_BA_ShdObjType;
    bVld       : BOOL;
END_STRUCT
END_TYPE
```

**lrP1x .. lrP4z:** corner coordinates. Of importance only if the element is a rectangle.

**lrMx .. lrMz:** center coordinates. Of importance only if the element is a sphere.

**lrRads:** radius of the sphere. Of importance only if the element is a sphere.

**usiBegMth:** beginning of the shading period (month)

**usiEndMth:** end of the shading period (month)

**eType:** object type, see E_BA_ShdObjType [▶ 325]

**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 (of the following year).

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TS8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.11 ST_BA_SpRmT

Room temperature setpoints

```
TYPE ST_BA_SpRmT :
STRUCT
   lrPrtcHtg    : LREAL := 12.0;
   lrEcoHtg     : LREAL := 15.0;
   lrPreCmfHtg  : LREAL := 19.0;
   lrCmfHtg     : LREAL := 21.0;
   lrPrtcCol    : LREAL := 40.0;
   lrEcoCol     : LREAL := 35.0;
   lrPreCmfCol  : LREAL := 28.0;
   lrCmfCol     : LREAL := 24.0;
END_STRUCT
END_TYPE
```

The values in the structure are defined with the preset values.

The variables have the following meaning:

**lrPrtcHtg:** Protection Heating

**lrEcoHtg:** Economy Heating

**lrPreCmfHtg:** Pre-Comfort Heating

**lrCmfHtg:** Comfort Heating

**lrPrtcCol:** Protection Cooling

**lrEcoCol:** Economy Cooling

**lrPreCmfCol:** Pre-Comfort Cooling

**lrCmfCol:** Comfort Cooling

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.12 ST_BA_Sunbld

Structure of the blind positioning telegram

```
TYPE ST_BA_SunBld:
STRUCT
     lrPos       : LREAL;
     lrAngl      : LREAL;
     bManUp      : BOOL;
     bManDwn     : BOOL;
     bManMod     : BOOL;
     bActv       : BOOL;
END_STRUCT
END_TYPE
```

**lrPos:** transferred blind height [%]

**lrAngl:** transferred slat 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 [▶ 291] or FB_BA_SunBldPrioSwi8 [▶ 293]. The sun protection actuators FB_BA_SunBldActr [▶ 284] and FB_BA_RolBldActr [▶ 276] ignore it.

### Requirements

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.13 ST_BA_SunBldScn

Table entry for a blind scene

```
TYPE ST_BA_SunBldScn:
STRUCT
     lrPos     : LREAL;
     lrAngl    : LREAL;
END_STRUCT
END_TYPE
```

**lrPos:** blind height [%]

**lrAngl:** slat position [°]

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.14 E_BA_CtrlSttupMod

Indicates the start light value for constant light control.

```
TYPE E_BA_CtrlSttupMod :
(
eBA_SttMaxVal := 0,
eBA_SttMinVal := 1,
eBA_SttDctVal := 2
);
END_TYPE
```

**eBA_SttMaxVal:** starts with the maximum value. The control starts with 100% - this makes all connected light actuators jump to their maximum value.

**eBA_SttMinVal:** starts with the minimum value. The control starts with 1% - this makes all connected light actuators jump to their minimum value ( not OFF).

**eBA_SttDctVal:** starts with the value specified under *lrSttDctVal*.

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.15 E_BA_OnMod

Describes the switch-on behavior via the command *bOn* or via a short click on the inputs *bSwi*, *bSwiUp* and *bSwiDwn*.

```
TYPE E_BA_OnMod :
(
eBA_ClMaxVal := 0,
eBA_ClMinVal := 1,
eBA_ClOnVal  := 2,
eBA_ClRstVal := 3
);
END_TYPE
```

**eBA_ClMaxVal:** switches the light actuator to the maximum value.

**eBA_ClMinVal:** switches the light actuator to the minimum value.

**eBA_ClOnVal:** switches the light actuator to the value *lrOnVal*.

**eBA_ClRstVal:** switches the light actuator to the value, which the switch actuator had before the last switch-off.

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.3.16 ST_BA_LgtCmd

Command telegram from the light switching functions to the light actuators

```
TYPE ST_BA_LgtCmd :
STRUCT
     bOff            : BOOL;
     bClMinCtrlVal   : BOOL;
     bClMaxCtrlVal   : BOOL;
     bClOnVal        : BOOL;
     lrOnVal         : LREAL;
     bClRstVal       : BOOL;
     bSetCtrlValDct  : BOOL;
     lrSetCtrlValDct : LREAL;
     bChgMaxCtrlVal  : BOOL;
     lrChgMaxCtrlVal : LREAL;
     bChgMinCtrlVal  : BOOL;
     lrChgMinCtrlVal : LREAL;
     bDimUp          : BOOL;
     bDimDwn         : BOOL;
     bStepUp         : BOOL;
     bStepDwn        : BOOL;
END_STRUCT
END_TYPE
```

**bOff:** switches the light actuator off immediately.

**bClMinCtrlVal:** switches the light actuator to the minimum value.

**bClMaxCtrlVal:** switches the light actuator to the maximum value.

**bCalOnVal / lrOnVal:** switches the light actuator to the selected on value.

**bClRstCtrlVal:** switches the light actuator to the value, which the switch actuator had before the last switch-off.

**bSetCtrlValDct / lrSetCtrlValDct :** switches the light actuator to the value *lrSetCtrlValDct*.

**bChgMaxCtrlVal / lrChgMaxCtrlVal:** assigns a new maximum value *lrChgMaxCtrlVal* to the light actuator.

**bChgMinCtrlVal / lrChgMinCtrlVal:** assigns a new maximum value *lrChgMinCtrlVal* to the light actuator.

**bDimUp:** static signal for up dimming; for DALI light actuators, the command "UP" is issued every 200 ms.

**bDimUp:** static signal for down dimming; for DALI light actuators, the command "DOWN" is issued every 200 ms.

**bStepUp:** switches the light value up by one step. For DALI light actuators this involves ONE step up, which corresponds to the DALI command "STEP UP". For analog light actuators it corresponds to an increase by (100/254) %.

**bStepDwn:** switches the light value down by one step. For DALI light actuators this involves ONE step down, which corresponds to the DALI command "STEP DOWN". For analog light actuators it corresponds to a decrease by (100/254) %

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

### 8.3.17 E_BA_DALIMod

Functional specification of the DALI light actuator function block

```
TYPE E_BA_DALIMod :
(
eBA_DALIModGrpMst := 0,
eBA_DALIModGrpSlv := 1,
eBA_DALIModGrpSgl := 2,
eBA_DALIModSglDvc := 3
);
END_TYPE
```

**eBA_DALIModGrpMst:** master of a group with one or several light actuators representing slaves (*eBA_DALIModGrpSlv*). The master switches the group commands through. The parameterization via the light actuator inputs is done individually for all devices.

**eBA_DALIModGrpSlv:** slave of a group with one or several light actuators representing slaves (*eBA_DALIModGrpSlv*). The master switches the group commands through. The parameterization via the light actuator inputs is done individually for all devices.

**eBA_DALIModGrpSgl:** master of a group, in which the slaves are **not** listed as function blocks. The master switches the group commands through. The parameterization via the light actuator inputs is done for all devices as a group command from the master.

**eBA_DALIModSglDvc:** a light actuator function block switches a DALI device via a short address.

**Requirements**

| Development environment | Target system | required library | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 | PC/CX | TcBA library from V1.0.0 | TF8040 \| TwinCAT Building Automation from V1.0.0 |

## 8.4 Error Codes

| ErrId (hex) Warning | ErrId (hex) stopping error | ErrArg | Description |
|---|---|---|---|
| 0x0 | 0x0 | | No error |
| 0x40008001 | 0x80008001 | 1 | The lower limit value of the control value *lrMinOut* is greater than the upper limit value *lrMaxOut* |
| | | 2 | The value of the neutral zone (*lrNz*) must not be less than 0.0 |
| | | 3 | The value of the proportionality constant (*lrKp*) must not be less than 0.0 |
| 0x40008002 | 0x80008002 | 1 | A controller ordinal number *usiMyNum* has been assigned twice |
| | | 2 | A controller ordinal number *usiMyNum* is greater than the maximum permitted number *gBA_cMaxSeqCtrl*. |
| | | 3 | The lower limit value of the control value *lrMinOut* is greater than the upper limit value *lrMaxOut*. |
| | | 4 | The value of the neutral zone (*lrNz*) must not be less than 0.0. |

| ErrId (hex) Warning | ErrId (hex) stopping error | ErrArg | Description |
|---|---|---|---|
| | | 5 | The value of the proportionality constant (*lrKp*) must not be less than 0.0. |
| | | 6 | The controller ordinal number *usiMyNum* of the enabled controller is 0. That is only allowed for controllers that are not in use and thus not enabled. |
| 0x40008003 | 0x80008003 | 1 | The sequence link was notified that a controller ordinal number *usiMyNum* has been assigned twice. |
| | | 2 | Direction of action changed twice in the controller sequence. |
| | | 3 | In the controller sequence, a controller with a higher ordinal number has a lower setpoint than its "predecessor". No correction takes place; the controller sequence runs with the parameters that were entered. |
| | | 4 | The sequence controller, which is defined as start controller (*usiSttCtrl*) is not parameterized at all, i.e. it is not present. The controller with the lowest ordinal number is used as start controller. |
| | | 5 | The ordinal number of the start controller is higher than the maximum permitted number of controllers or zero. The controller with the lowest ordinal number is used as start controller. |
| | | 6 | The sequence controller, which is defined as start controller (*usiSttCtrl*) is not enabled (present). The controller with the lowest ordinal number is used as start controller. |
| 0x40008004 | 0x80008004 | 1 | The lower limit value of the control value *lrMinOut* is greater than the upper limit value *lrMaxOut* |
| 0x40008005 | 0x80008005 | 1 | The minimum input signal *lrMinIn* is greater than the maximum input signal *lrMaxIn* |
| 0x40008010 | 0x80008010 | 1 | Wrong mapping: the mapping for the BACnet device is incorrect and must be checked |
| | | 2 | Operational; the BACnet device is not operational |
| | | 3 | ADS connection disturbed |
| | | 4 | no valid Ams port, must be greater than 1000 |
| 0x40008011 | 0x80008011 | 1 | Wrong entry priority A, PrioA must not be <1 or =6 or >16 |
| | | 2 | Wrong entry priority A, PrioA must not be <1 or >16 |
| | | 3 | Wrong entry priority B, PrioB must not be <1 or =6 or >16 |
| | | 4 | Wrong entry priority B, PrioB must not be <1 or >16 |
| | | 5 | Wrong entry priority C, PrioC must not be <1 or =6 or >16 |
| | | 6 | Wrong entry priority C, PrioC must not be <1 or >16 |
| | | 7 | Wrong entry priority D, PrioD must not be <1 or =6 or >16 |
| | | 8 | Wrong entry priority D, PrioD must not be <1 or >16 |
| | | 9 | Wrong input priority array, arrPrio[1] or arrPrio[2] or arrPrio[3] must not be > 255 |
| 0x40008013 | 0x80008013 | ADS no | ADS error FB_BACnet_WriteProp. Processing of the function block continues. |
| 0x40008014 | 0x80008014 | ADS no | ADS error FB_BACnet_ReadProp. Processing of the function block continues. |
| 0x40008015 | 0x80008015 | 1 | Overflow of the structure ST_BA_ComnMsg. Too many inputs from the individual BACnet objects in the Common Message structure. Remedy through new FB_BA_ComnMsg. |

| ErrId (hex) Warning | ErrId (hex) stopping error | ErrArg | Description |
|---|---|---|---|
| | | 2 | Overflow of the structure ST_BA_ComnMsgTermt. Too many inputs from the FB_BA_ComnMsg analysis function blocks in the Common Message Terminated structure. Remedy through new FB_BA_ComnMsgTermt and global structure |
| 0x40008016 | 0x80008016 | 1 | Wrong entry PresentValue priority A, PrioA must not be < 1 OR > NumberOfState |
| | | 2 | Wrong entry PresentValue priority B, PrioB must not be < 1 OR > NumberOfState |
| | | 3 | Wrong entry PresentValue priority C, PrioC must not be < 1 OR > NumberOfState |
| | | 4 | Wrong entry PresentValue priority D, PrioD must not be < 1 OR > NumberOfState |
| 0x40008017 | 0x80008017 | Locality | Shows the ID of the FB_BA_ComnMsg function block to indicate the fault location |
| 0x40008018 | 0x80008018 | 1 | Wrong entry Relinquish Default, must not be < 1 or > NumberOfStates |
| | | 2 | Wrong entry HighLimit/LowLimit, HighLimit is less than LowLimit |
| | | 3 | FB_BACSChrct: wrong declaration *usiNumOfPnt* |
| 0x40008019 | 0x80008019 | Locality | Shows the field in the array containing the error. The entries in the array arrX must have the following structure: arr[1] > arrX[2] > arrX[n] OR arr[1] < arrX[2] < arrX[n] |
| 0x40008020 | 0x80008020 | 1 | The object with an error is indicated by the two outputs dwObjId / eObjType |
| 0x40008021 | 0x80008021 | 1 | lr01 exceeds the range of LREAL |
| | | 2 | Addition *lr01/lr02* exceeds the range of LREAL |
| | | 3 | Addition *lr01/lr02/lr03* exceeds the range of LREAL |
| | | 4 | Addition *lr01/lr02/lr03/lr04* exceeds the range of LREAL |
| | | 5 | Addition *lr01/lr02/lr03/lr04/lr05* exceeds the range of LREAL |
| | | 6 | Addition *lr01/lr02/lr03/lr04/lr05/lr06* exceeds the range of LREAL |
| | | 7 | Addition *lr01/lr02/lr03/lr04/lr05/lr06/lr07* exceeds the range of LREAL |
| | | 8 | Addition *lr01/lr02/lr03/lr04/lr05/lr06/lr07/lr08* exceeds the range of LREAL |
| | | 9 | Difference between MAX and MIN exceeds the range of LREAL |
| 0x40008027 | 0x80008027 | Locality | udiErrArg; the variable *lrX0x* indicates where the error is located. The following sequence must be followed: *lrX01* > *lrX02* > *lrXn* OR *lrX01* < *lrX02* < *lrXn* |
| 0x40008028 | 0x80008028 | 1 | Wrong entry PresentValue; must not be <1 or >*udiNumOfSta* (NumberOfStates) |
| 0x4000802F | 0x8000802F | 1 | Heating setpoint is greater than or equal to cooling setpoint |
| 0x40008030 | 0x80008030 | 1 | Wrong parameter *udiNum* > *udiNumOfEn* |
| 0x40008030 | 0x80008030 | 1 | Internal error MEMSET |
| | | 2 | Internal error *tTaskCycleTime* |
| | | 3 | Internal error *tCtrlCycleTime* |
| | | 4 | Internal error *InvalidParam_tTi* |
| | | 5 | Internal error *InvalidParam_fBaseTime* |
| | | 6 | Internal error Error_MEMCPY |
| 0x40008033 | 0x80008033 | 1 | Wrong parameter *udiNumOfStep* |
| 0x40008034 | 0x80008034 | ADS no | ADS error during reading of the time |

| ErrId (hex) Warning | ErrId (hex) stopping error | ErrArg | Description |
|---|---|---|---|
| 0x40008035 | 0x80008035 | ADS no | ADS error during setting of the time |
| 0x40008036 | 0x80008036 | 1 | Error: range exceeded year |
| | | 2 | Error: range exceeded month |
| | | 3 | Error: range exceeded day of the month |
| | | 4 | Error: range exceeded hour |
| | | 5 | Error: range exceeded minute |
| | | 6 | Error: range exceeded second |
| | | 7 | Error: range exceeded millisecond |
| 0x40008100 | 0x80008100 | 1 | The x-values (elevation values) in the table are either not listed in ascending order, or they are duplicated. |
| | | 2 | An elevation value that was entered is outside the valid range of 0°...90°. |
| | | 3 | A position value that was entered is outside the valid range of 0%...100%. |
| 0x40008101 | 0x80008101 | 1 | The switch-on hysteresis value *lrOnVal* is less than or equal to the switch-off hysteresis value *lrOffVal*. |
| | | 2 | The switch-on hysteresis value *lrOnVal* or the switch-off hysteresis value *lrOffVal* is less than zero. |
| 0x40008102 | 0x80008102 | 1 | Parameter error: *lrSttRng=lrEndRng*. |
| | | 2 | Parameter error: *lrSttRng* or *rEndRng* less than 0° or greater than 360°. |
| | | 3 | Parameter error: the difference between *lrSttRng* and *lrEndRng* is greater than 180°. This range is too large for analyzing the insolation on a facade. |
| 0x40008103 | 0x80008103 | 1 | Parameter error: *lrHiLmt* less than or equal to rLoLmt. |
| | | 2 | Parameter error: *lrLoLmt* is less than 0° or *lrHiLmt* is greater than 90°. |
| 0x40008104 | 0x80008104 | 1 | The total travel-up or travel-down time (*udiTiUp/udiTiDwn*) is zero. |
| 0x40008105 | 0x80008105 | 1 | Parameter error: roller shutter height is less than or equal to 0. |
| | | 2 | Parameter error: Up/Down timer = 0. |
| | | 3 | Parameter error: turning/turning timer = 0. |
| | | 4 | Parameter error: slat angle limits: upper limit is less than or equal to lower limit (*lrAnglLmtUp<=lrAnglLmtDwn)*. |
| | | 5 | One of the hysteresis offset values for repositioning, *lrHysIncPos* or *lrHysIncAngl*, is less than zero |
| 0x40008106 | 0x80008106 | 1 | The entered scene number *udiSlcdScn* is greater than the maximum permitted number of scenes. |
| | | 2 | The position setpoint *lrSpPos* is greater than 100 or less than 0. |
| 0x40008107 | 0x80008107 | 1 | The setpoints transferred via the structure *stRmTSp* are not in ascending order. Correct: *stRmTSp.rPreCmfHtg <= stRmTSp.rCmfHtg <= stRmTSp.rCmfCol <= stRmTSp.rPreCmfCol*. |
| | | 2 | The switch-on hysteresis value *lrBrtnsActvVal* is less than or equal to the switch-off hysteresis value *lrBrtnsDctvVal*. |
| | | 3 | The blind position in heating mode *lrPosHtg* is greater than the position in cooling mode *lrPosCol*. This makes no sense. |
| | | 4 | One of the switching values, *lrBrtnsActvVal* or *lrBrtnsDctvVal*, is less than 0. |
| | | 5 | One of the position setpoints, *lrPosHtg* or *lrPosCol*, is greater than 100 or less than 0. |

| ErrId (hex) Warning | ErrId (hex) stopping error | ErrArg | Description |
|---|---|---|---|
| 0x40008108 | 0x80008108 | 1 | The switch-on hysteresis value *lrActvVal* is less than or equal to the switch-off hysteresis value *lrDctvVal*. |
| | | 2 | One of the switching values, *lrActvVal* or *lrDctvVal*, is less than 0. |
| | | 3 | The position setpoint *lrPosTwiLgt* is greater than 100 or less than 0. |
| 0x40008109 | 0x80008109 | 1 | The frost temperature that was entered is higher than 10°C. |
| | | 2 | The wind speed value for switching off the alarm that was entered is greater than or equal to the switch-on value. |
| | | 3 | The wind speed value that was entered for switching the alarm on or off is less than 0. |
| | | 4 | The position setpoint *lrPosProt* is greater than 100 or less than 0. |
| 0x4000810A | 0x8000810A | 1 | The duration of the positioning interval is zero or it exceeds 720 min. |
| | | 2 | The value entered for the longitude is not in the valid range of -180°...180°. |
| | | 3 | The value entered for the latitude is not in the valid range of -90°...90°. |
| | | 4 | The value entered for the facade inclination *lrFcdAngl* is outside the valid range of -90°..90°. |
| | | 5 | The value for the slat spacing (*lrLamDstc*) is greater than or equal to the value for the slat width (*lrLamWdth*). This does not represent a "valid" blind, since the slats cannot close fully. Mathematically, this would lead to errors. |
| | | 6 | The value entered for the slat spacing *lrLamDstc* is zero. |
| | | 7 | The value entered for the slat width *lrLamWdth* is zero. |
| | | 8 | The value entered for the fixed blind height (*lrFixPos*) is greater than 100 or less than 0. **At the same time, positioning "fixed blind height" is selected -** *ePosMod=ePosModFix.* |
| | | 9 | Tthe "Values valid" bit (*bVld*) in the stBldPosTab [▶ 325] positioning table is not set - invalid values, see FB_BA_BldPosEntry [▶ 254]. **At the same time, "Table" positioning is selected -** *ePosMod=ePosModTab.* |
| | | 10 | The value entered for the window height *lrWdwHght* is less than or equal to zero. **At the same time, "maximum light incidence" is selected -** *ePosMod=ePosModMaxIndc.* |
| | | 11 | The distance between lower window edge and floor *lrDstcWdwFlr* that was entered is less than zero. **At the same time, "maximum light incidence" is selected -** *ePosMod=ePosModMaxIndc.* |
| | | 12 | The value entered for the maximum required light incidence *lrMaxLgtIndc* is less than or equal to zero. **At the same time, "maximum light incidence" is selected -** *ePosMod=ePosModMaxIndc.* |
| 0x40008120 | 0x80008120 | 1 | Index error! *iColumn* and/or *iRow* are outside the permissible limits *1..gBA_cMaxColumnFcd* or *1..gBA_cMaxRowFcd*. See list of facade elements [▶ 46]. |
| | | 2 | 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. |

| ErrId (hex) Warning | ErrId (hex) stopping error | ErrArg | Description |
|---|---|---|---|
| | | 3 | The X-component of the first corner (Corner1) is less than zero. |
| | | 4 | The Y-component of the first corner (Corner1) is less than zero. |
| | | 5 | The window width is less than or equal to zero. |
| | | 6 | The window height is less than or equal to zero. |
| 0x40008121 | 0x80008121 | 1 | Error reading the data file: file is too large (number of bytes greater than *cMaxDataFileSize*) |
| 0x40008122 | 0x80008122 | ADS no | File handling error: Open Logfile |
| 0x40008123 | 0x80008123 | ADS no | File handling error: Open Datafile |
| 0x40008124 | 0x80008124 | ADS no | File handling error: Read Datafile |
| 0x40008125 | 0x80008125 | ADS no | File handling error: Write Logfile |
| 0x40008126 | 0x80008126 | ADS no | File handling error: Close Datafile |
| 0x40008127 | 0x80008127 | ADS no | File handling error: Write Logfile |
| 0x40008128 | 0x80008128 | ADS no | File handling error: Close Logfile |
| 0x40008129 | 0x80008129 | 1 | Error reading the data file: file is too large (number of bytes greater than *cMaxDataFileSize*) |
| 0x4000812A | 0x8000812A | ADS no | File handling error: Open Logfile |
| 0x4000812B | 0x8000812B | ADS no | File handling error: Open Datafile |
| 0x4000812C | 0x8000812C | ADS no | File handling error: Read Datafile |
| 0x4000812D | 0x8000812D | ADS no | File handling error: Write Logfile |
| 0x4000812E | 0x8000812E | ADS no | File handling error: Close Datafile |
| 0x4000812F | 0x8000812F | ADS no | File handling error: Write Logfile |
| 0x40008130 | 0x80008130 | ADS no | File handling error: Close Logfile |
| 0x40008131 | 0x80008131 | 1 | The index of the window group *usiGrpId* under consideration is 0. |
| | | 2 | An element of the window list *arrFcdElem[i,j]* is invalid. See outputs iErrIdx_I and iErrIdx_J. |
| 0x40008132 | 0x80008132 | 1 | Index error! iId lies outside the permissible limits 1..gBA_cMaxShdObj. |
| | | 2 | 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. |
| | | 3 | The corners of the rectangle are not on the same level. |
| | | 4 | The z-component of P1 is less than zero. This corner would thus lie behind the facade. |
| | | 5 | The z-component of P3 is less than zero. This corner would thus lie behind the facade. |
| | | 6 | P1 is equal to P2. The object entered is thus not a rectangle. |
| | | 7 | P1 is equal to P3. The object entered is thus not a rectangle. |
| | | 8 | P1 is equal to P4. The object entered is thus not a rectangle. |
| | | 9 | P2 is equal to P3. The object entered is thus not a rectangle. |
| | | 10 | P2 is equal to P4. The object entered is thus not a rectangle. |
| | | 11 | P3 is equal to P4. The object entered is thus not a rectangle. |
| | | 12 | The radius entered is zero. |
| | | 13 | The z-component of the sphere center is less than zero. This point would thus lie behind the facade. |
| | | 14 | Error object type eType - neither rectangle nor sphere. |
| | | 15 | Month input error. |

| ErrId (hex) Warning | ErrId (hex) stopping error | ErrArg | Description |
|---|---|---|---|
| 0x40008200 | 0x80008200 | 1 | Parameter "*IrMinCtrlVal*" (minimum light output value) less than 1.0. The value was automatically corrected to 1.0. |
| | | 2 | Parameter "*IrMinCtrlVal*" (minimum light output value) above 100.0. The value was automatically corrected to 100.0. |
| | | 3 | Parameter "*IrMaxCtrlVal*" (maximum light output value) less than 1.0. The value was automatically corrected to 1.0. |
| | | 4 | Parameter "*IrMinCtrlVal*" (maximum light output value) above 100.0. The value was automatically corrected to 100.0. |
| | | 5 | Parameter "*IrMaxCtrlVal*" (maximum light output value) below "*IrMinCtrlVal*" (maximum light output value). The value was automatically corrected to "*IrMinCtrlVal*". |
| 0x40008201 | 0x80008201 | DALI-ErrId | Carries the error argument directly from the error code list of the DALIV2 library, see TcPlcLibDALIV2_Errorcodes |
| 0x40008202 | 0x80008202 | 1 | Invalid short address, above 63. |
| | | 2 | Invalid group address, above 15. |
| | | | |
| | | | |
| | | | |

# 9 TwinCAT BA PLC Templates

### Introduction

The TwinCAT BA PLC templates consist of ready-to-use TwinCAT program blocks for sensors and actuators, for complete modules and for system parts or complete systems for heating, ventilation and air conditioning and room automation. As opposed to a regular PLC library, TwinCAT BA PLC templates are imported as program blocks into the PLC programs. The system integrator can carry out any necessary adaptations himself. In addition, it is possible to create user-defined TwinCAT BA PLC templates. No special tools are required.

### General

| Name | Description |
|---|---|
| BAC_Gen_01 [▶ 348] | Call template providing the basic function of a BACnet controller. |
| BAC_GenAlm_01 [▶ 349] | Collection of all alarms of a controller |
| BAC_GenComnMsg_01 [▶ 353] | Collection of all event messages of a controller |
| BAC_GenDvc_01 [▶ 356] | Template that instantiates the BACnet device. **For a BACnet plant this template is imperative; it only has to be positioned once.** |
| BAC_GenNC_01 [▶ 361] | The template provides eight BACnet message classes (notification class) for transferring object-integrated messages (intrinsic reporting) . |
| BAC_GenSys_01 [▶ 363] | Template system functions, e.g. flash sequence for fault indications or provision of the system time. |
| BAC_GenWthT_01 [▶ 364] | Outside temperature and averaged outside temperature |

### Plant call templates

| Name | Description |
|---|---|
| BAC_AC_Identification_System_Plant_Key [▶ 608] | Identification system for HVAC plant templates |
| BAC_AC_SE_3_4_1_1_1_0 [▶ 610] | Call template ventilation systems with supply air fan and exhaust air fan and thermal air treatment: exhaust/supply cascade single - fan pressure - preheater - cooler - ERG plate |
| BAC_AC_SE_4_4_1_1_0_1 [▶ 615] | Call template ventilation systems with supply air fan and exhaust air fan and thermal air treatment: exhaust/supply air cascade - fan pressure - preheater - cooler - mixed air |
| BAC_AC_SE_4_4_1_1_3_0 [▶ 621] | Call template ventilation systems with supply air fan and exhaust air fan and thermal air treatment: exhaust/supply cascade - fan pressure - preheater - cooler - ERG rotation |
| BAC_AC_Sx_001 [▶ 628] | Basic system program for ventilation systems with thermal air treatment |

### Basic system programs

| Name | Description |
|---|---|
| BAC_PltAlm_01 [▶ 365] | Collection and acknowledgement of all alarms of a plant |
| BAC_PltComnMsg_01 [▶ 369] | Collection of all event messages of a plant |

### Heating

| Unit | Name | Description |
|---|---|---|
| Heating circuit | BAC_H_HtgCir_01 [▶ 543] | Call template of an atmospheric temperature controlled heating circuit |
| | BAC_H_HtgCirSp_01 [▶ 545] | The template is an atmospheric temperature controlled heating circuit control. It is comprised of a heating curve, a heating limit switch and the operating modes Frost / Night / Day / Auto with the associated setpoint. |
| Hot water | BAC_DHW_01 [▶ 547] | Call template control and charging of a hot water tank |
| | BAC_DHW_Ctrl_01 [▶ 549] | The template controls the charging of a hot water tank. |
| | BAC_HW_LglPrev_01 [▶ 551] | The template provides a protective function for preventing the formation of legionella in the hot water. |

### Air conditioning system

| Unit | Name | Description |
|---|---|---|
| Humidification | BAC_AC_Humf_01 [▶ 373] | Call template for controlling a steam humidifier |
| | BAC_AC_Humf_PID_01 [▶ 376] | Sequence controller humidification |
| | BAC_AC_SteamGenerator_01_xx [▶ 381] | Steam generator |
| Fire dampers | BAC_AC_FireDmp_01_xx [▶ 385] | Call template control and monitoring of a motorized fire damper with binary switching output |
| Energy recovery | BAC_AC_ErcPl_01 [▶ 391] | Call template for controlling a plate heat exchanger |
| | BAC_AC_ErcPl_02 [▶ 394] | Call template for controlling a plate heat exchanger with bypass damper |
| | BAC_AC_ErcRecup_01 [▶ 399] | Call template for controlling a run-around-coil system |
| | BAC_AC_ErcRot_01 [▶ 404] | Call template for controlling a rotary heat exchanger with bypass damper |
| | BAC_AC_ErcRot_02 [▶ 409] | Call template for controlling a rotary heat exchanger |
| | BAC_AC_ErcT_PID_01 [▶ 412] | Sequence controller energy recovery temperature |
| | BAC_ErcIcPrt_01 [▶ 418] | The template is used to protect the energy recovery units from icing up via a differential pressure switch |
| | BAC_ErcIcPrt_02 [▶ 420] | The template is used to protect the energy recovery units from icing up via a differential pressure sensor |

| Unit | Name | Description |
|---|---|---|
| Filter | BAC_AC_Filter_01 [▶ 436] | The template realizes filter monitoring within an air conditioning plant via a differential pressure switch. |
| Dampers | BAC_AC_ExhADmp2P_01_xx [▶ 438] | Call template for control and monitoring of an exhaust air damper with spring return actuator and end position monitor |
| | BAC_AC_OuADmp2P_01_xx [▶ 445] | Call template for control and monitoring of an outside air damper with spring return actuator and end position monitor |
| Cooler | BAC_AC_ColT_01 [▶ 457] | Call template of a temperature controlled cold-water air cooler (valve, pump) |
| | BAC_AC_ColT_02 [▶ 460] | Call template of a temperature controlled cold-water air cooler (valve) |
| | BAC_AC_ColT_PID_01 [▶ 463] | Sequence controller for cooler, temperature |
| | BAC_AC_ColTH_01 [▶ 468] | Call template of a temperature and humidity controlled cold-water air cooler with valve and pump |
| | BAC_AC_ColTH_02 [▶ 472] | Call template for a temperature and humidity-controlled cold-water air cooler with valve |
| | BAC_AC_ColH_PID_01 [▶ 452] | Sequence controller for a cooler, dehumidification |
| Mixed air | BAC_AC_MixAT_01 [▶ 475] | Call template mixed air dampers temperature-controlled for controlling three analog dampers |
| | BAC_AC_MixAT_PID_01 [▶ 480] | Sequence controller for mixed-air damper control |
| Reheater | BAC_AC_ReHtr_01 [▶ 485] | Call template for a reheater |
| | BAC_AC_ReHtr_PID_01 [▶ 488] | Sequence controller reheater |
| Fans | BAC_AC_ExtAFan_FC_01 [▶ 422] | Call template with integrated pressure regulator for a speed-controlled exhaust air fan |
| | BAC_AC_ExtAFan1st_01 [▶ 424] | Call template for a single-stage exhaust air fan |
| | BAC_AC_SuAFan_FC_01 [▶ 427] | Call template with integrated pressure regulator for a speed-controlled supply air fan |
| | BAC_AC_SuAFan1st_01 [▶ 430] | Call template for a single-stage supply air fan |
| | BAC_DiffPrssMonit_01 [▶ 433] | Differential pressure monitoring via a differential pressure monitor |
| | BAC_DiffPrssMonit_02 [▶ 435] | Differential pressure monitoring via a differential pressure sensor |
| Preheater | BAC_AC_PreHtr_01 [▶ 493] | Call template for a hot water air heater |
| | BAC_AC_PreHtr_PID_01 [▶ 498] | Sequence controller preheater |
| | BAC_AC_RetWtrCtrl_01 [▶ 503] | Call template for return temperature control of a preheater |

| Unit | Name | Description |
|---|---|---|
| | BAC_FrstPrt_01 [▶ 507] | Frost protection monitoring of preheater |
| Volume flow controller | BAC_AC_VAV_01_xx [▶ 509] | Call template control of a volume flow controller via analog output |

| Setpoint | Description |
|---|---|
| BAC_AC_CasCtrlH_01 [▶ 632] | Cascade controller for supply air humidity, consisting of a master controller for calculating the setpoints for humidification and dehumidification |
| BAC_AC_CasCtrlH_02 [▶ 635] | Cascade controller for supply air humidity, consisting of two master controllers for room or exhaust air/ supply air cascade humidity control. |
| BAC_AC_CasCtrlT_01 [▶ 639] | Cascade controller for supply air temperature, consisting of a master controller for setpoint calculation for heating, cooling and recovery |
| BAC_AC_CasCtrlT_02 [▶ 643] | Cascade controller for supply air temperature, consisting of two master controllers for setpoint calculation for heating, cooling and recovery |
| BAC_AC_SpRmTH_01 [▶ 654] | Setpoint program for an exhaust air/supply air cascade, each with a setpoint for relative and absolute humidity and a room temperature setpoint, including summer and winter compensation |
| BAC_AC_SpRmTH_02 [▶ 657] | Setpoint program for an exhaust air/supply air cascade with two setpoints for relative and absolute humidity and a room temperature setpoint, including summer and winter compensation |
| BAC_AC_SpRmTH_03 [▶ 661] | Setpoint program for an exhaust air/supply air cascade, each with a setpoint for relative and absolute humidity and two room temperature setpoints, including summer and winter compensation |
| BAC_AC_SpRmTH_04 [▶ 665] | Setpoint program for an exhaust air/supply air cascade, each with two setpoints for relative and absolute humidity and two room temperature setpoints, including summer and winter compensation |
| BAC_AC_SpRmT_01 [▶ 648] | Setpoint program for an exhaust air/supply air cascade with only one room temperature setpoint, including summer and winter compensation |
| BAC_AC_SpRmT_02 [▶ 651] | Setpoint program for an exhaust air/supply air cascade with a separate room temperature setpoint for heating and cooling mode, including summer and winter compensation |
| BAC_AC_SpSuAT_01 [▶ 669] | Setpoint program for supply air temperature control with a supply air temperature setpoint, including summer/winter compensation via a characteristic curve. |
| BAC_AC_SpSuAT_02 [▶ 672] | Setpoint program for supply air temperature control with separate supply air setpoints for heating, cooling and energy recovery, including two separate characteristic setpoint curves with summer and winter compensation |
| General | Description |
| BAC_AC_OpMod_01 [▶ 515] | Operation mode of a ventilation system |
| BAC_AC_SeqH_01 [▶ 520] | Starting and controlling the supply air sequence control humidity of an air-conditioning plant |

| Setpoint | Description |
|---|---|
| BAC_AC_SeqT_01 [▶ 524] | Starting and controlling the supply air sequence control temperature of an air-conditioning plant |
| BAC_AC_StartT_01 [▶ 530] | Step-by-step startup of an air-conditioning plant without humidification/dehumidification. |
| BAC_AC_StartTH_01 [▶ 535] | Step-by-step startup of an air-conditioning plant with humidification/dehumidification. |
| BAC_AC_SumNgtCol_01 [▶ 540] | Summer night cooling |

### Universal

| | Name | Description |
|---|---|---|
| Frequency converter | BAC_Uni_FC_01_xx [▶ 555] | Control of a frequency converter |
| Damper | BAC_Uni_Dmp_01_xx [▶ 560] | Control of an analog damper drive |
| | BAC_Uni_Dmp2P_01_xx [▶ 563] | Control of a two-point damper |
| Motor | BAC_Uni_Mot1st_01_xx [▶ 567] | Control of a single-stage motor |
| Pump | BAC_Uni_Pu1st_01_xx [▶ 571] | Control of a single-stage pump |
| Smoke detector | BAC_Uni_SmokeDetc_001 [▶ 576] | Control and monitoring of a duct smoke detector |
| Control | BAC_Cont4Stp_01 [▶ 579] | Step switch with 4 stages for controlling multi-stage units |
| | BAC_HX_01 [▶ 582] | Calculation of dew point temperature, enthalpy, wet bulb temperature and absolute humidity |
| | BAC_Hys_01 [▶ 584] | Hysteresis function with fixed switching points |
| | BAC_Hys_02 [▶ 585] | Hysteresis function with dynamic switching points |
| | BAC_PID_01 [▶ 587] | Universal PID controller |
| | BAC_PID_02 [▶ 589] | Universal PID controller |
| | BAC_PID_03 [▶ 590] | Universal PID controller without referencing objects |
| | BAC_Ramp_01 [▶ 592] | Falling ramp limitation |
| | BAC_Ramp_02 [▶ 593] | Rising ramp limitation |
| | BAC_Scale_02 [▶ 594] | Linear interpolation with 2 interpolation points |
| | BAC_Scale_04 [▶ 595] | Linear interpolation with 4 interpolation points |
| | BAC_Scale_07 [▶ 597] | Linear interpolation with 7 interpolation points |
| Valve | BAC_Uni_Vlv_01_xx [▶ 598] | Control of an analog control valve |
| | BAC_Uni_Vlv3P_01_xx [▶ 602] | Control of a three-point valve |

### BACnet objects

| Name | Description |
|---|---|
| BAC_AI_01 [▶ 675] | Analog input object with alarm logging |
| BAC_AI_02 [▶ 676] | Analog input object |
| BAC_AI_Enthalpy_01 [▶ 678] | The template logs the two analog input values **temperature** and **relative humidity**. The function block **HX** is used to calculate the **dew point temperature**, the specific **enthalpy**, the **wet bulb temperature** and the **absolute humidity**. |

| Name | Description |
|---|---|
| BAC_AO_01 [▶ 682] | Analog output object with manual intervention via a digital input **LocSwi** and a trend object |
| BAC_AO_02 [▶ 684] | Analog output object |
| BAC_AO_03 [▶ 686] | Analog output object with manual intervention via a digital input **LocSwi** |
| BAC_AV_01 [▶ 688] | Analog value object with prioritization of a REAL process value |
| BAC_AV_02 [▶ 690] | Analog value object for input of a setpoint or parameter |
| BAC_AV_03 [▶ 691] | Analog value object with prioritization of a REAL process value with trend object |
| BAC_AV_04 [▶ 693] | Analog value object for input of a setpoint or parameter with trend object |
| BAC_BI_01 [▶ 694] | Binary input object with alarm logging |
| BAC_BI_02 [▶ 695] | Binary input object |
| BAC_BI_CMD_01 [▶ 697] | Binary input object with alarm recording and group order object for writing to specified properties of other objects |
| BAC_BO_01 [▶ 698] | Binary output object with prioritization of the switch value and manual intervention via a digital input **LocSwi** |
| BAC_BO_02 [▶ 701] | Binary output object with prioritization of the switching value |
| BAC_BV_01 [▶ 703] | Binary value object with prioritization of the process value |
| BAC_BV_02 [▶ 704] | Binary value object for input of a setpoint or parameter |
| BAC_CMD_01 [▶ 705] | Group order object |
| BAC_MV_01 [▶ 706] | Multistate value object with simple prioritization |
| BAC_MV_02 [▶ 707] | Multistate value object for operating and monitoring of a process value |
| BAC_SchedBinPV_01 [▶ 708] | The template includes a BACnet scheduler of type binary present value. In addition, the template contains the function "Pre-calculating switch-on/ switch-off time" |
| BAC_SchedB_01 [▶ 709] | The template contains a BACnet scheduler of output type BOOL |
| BAC_SchedR_01 [▶ 710] | The template contains a BACnet scheduler of output type REAL |
| BAC_SchedUdi_01 [▶ 711] | The template contains a BACnet scheduler of output type UDINT |
| BAC_TL_01 [▶ 712] | Trendlog memory object |

### IO templates

| Name | Description |
|---|---|
| P_KL1501 [▶ 713] | I/O template for parameterization of a KL1501: 1-channel up-down counter. |
| P_KL27x1 [▶ 714] | I/O template for parameterization of a KL2751 / KL2761: 1-channel dimmer terminal. |
| P_KL320x [▶ 715] | I/O template for parameterization of a KL3201 or KL3202: input terminal for resistance sensors. |

| Name | Description |
|---|---|
| P_KL3204 [▶ 716] | I/O template for parameterization of a KL3204: 4-channel input terminal for resistance sensors. |
| P_KL3208 [▶ 718] | I/O template for parameterization of a KL3208-0010: 8-channel input terminal for resistance sensors. |
| P_KL3228 [▶ 719] | I/O template for parameterization of a KL3228: 8-channel input terminal for resistance sensors. |
| P_KL8519 [▶ 720] | I/O template for parameterization of a KL8519: 16-channel digital input signal module. |
| P_KL8524 [▶ 723] | I/O template for parameterization of a KL8524: 4 x 2-channel digital output module. |
| P_KL8528 [▶ 725] | I/O template for parameterization of a KL8528: 8-channel digital output module. |
| P_KL8548 [▶ 726] | I/O template for parameterization of a KL8548: 8-channel analog output module 0...10 V. |

## 9.1 Global_Variablen_Alarming

In the PLC the following function blocks, variables and constants can be found under **Resources\GlobalVariables**:

| **VAR_GLOBAL** | | | |
|---|---|---|---|
| gBA_AlmMgnr | FB_BA_AlarmMgnr | | Alarmsammler mit Neuwertmeldung//Alarm collector with new value message |
| **END_VAR** | | | |
| **VAR_GLOBAL CONSTANT** | | | |
| Alarming Priority | | | |
| ALM_PRIO_EMPTY | BYTE | 0 | keine Meldung//no message |
| ALM_PRIO_NOTE | BYTE | 1 | diverse Meldungen// various message |
| ALM_PRIO_WARNING | BYTE | 2 | Warnung//warning |
| ALM_PRIO_ALARM | BYTE | 3 | Alarm ohne Abschaltung// Alarm without shutdown |
| ALM_PRIO_CRITICAL | BYTE | 4 | Alarm mit Abschaltung// Alarm with shutdown<br><br>setzt den Ausgang *bCriticalAlm* vom Funktionsbaustein AlarmPlt in Template BAC_PltAlm_01 [▶ 365]// sets the output*bCriticalAlm* by the function blockAlarmPltinTemplateBAC_PltAlm_01 [▶ 365] |
| **END_VAR** | | | |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

# 9.2 BAC_Gen_01

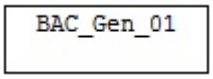
### Application

The template BAC_Gen_01 generates basic functions, system data and global variables. It is a prerequisite for the operation of many other call templates and subtemplates.
**It is therefore mandatory to load the call template BAC_Gen_01 once per automation station!**

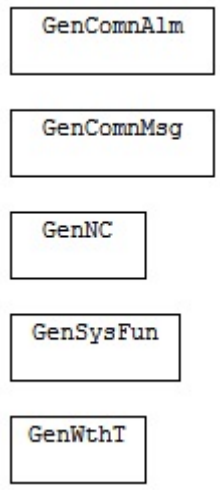
The template BAC_Gen_01 is a call template and includes the following subtemplates:

- **GenComnAlm** The subtemplate BAC_GenAlm_01 [▶ 349] collects all alarms within an automation station.
- **GenComnMsg** The subtemplate BAC_GenComnMsg_01 [▶ 353] collects all event messages of the BACnet objects within an automation station.
- BAC_GenDvc_01 [▶ 356] The subtemplate is responsible for interfacing of the PLC program with a local BACnet device object (server).
- **GenNC** The subtemplate BAC_GenNC_01 [▶ 361] provides eight BACnet notification classes for the transmission of object-integrated messages (intrinsic reporting).
- **GenSysFun** The subtemplate BAC_GenSys_01 [▶ 363] reads the system time from the PC and maps the time information in the PLC to variables. In addition, a flash pulse is generated.
- **GenWthT** The subtemplate BAC_GenWthT_01 [▶ 364] maps the outside temperature and the averaged outside temperature via AV objects.
- BAC_GenGlobal_01 [▶ 357] The subtemplate generates a list of global variables and constants for the operation of other subtemplates and call templates.

### Interface




### Block diagram




### Program description

| Instance | Type | Task |
|---|---|---|
| GenComnAlm | BAC_GenAlm_01 [▶ 349] | The subtemplate BAC_GenAlm_01 [▶ 349] collects all alarms within an automation station |

| Instance | Type | Task |
|---|---|---|
| GenComnMsg | BAC_GenComnMsg_01 [▶ 353] | The subtemplate BAC_GenComnMsg_01 [▶ 353] collects all event messages of the BACnet objects within an automation station. |
| BAC_GenDvc_01 [▶ 356] | | The subtemplate is responsible for interfacing of the PLC program with a local BACnet device object (server). |
| GenNC | BAC_GenNC_01 [▶ 361] | The subtemplate BAC_GenNC_01 [▶ 361] provides eight BACnet notification classes for the transmission of object-integrated messages (intrinsic reporting) |
| GenSysFun | BAC_GenSys_01 [▶ 363] | The subtemplate BAC_GenSys_01 [▶ 363] reads the system time from the PC and maps the time information in the PLC to variables. In addition, a flash pulse is generated. |
| GenWthT | BAC_GenWthT_01 [▶ 364] | The subtemplate BAC_GenWthT_01 [▶ 364] maps the outside temperature and the averaged outside temperature via AV objects. |
| | BAC_GenGlobal_01 [▶ 357] | The subtemplate generates the global variables and constants within an automation station. |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.3 BAC_GenAlm_01

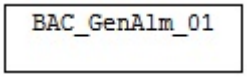
**Functional description**

The template **BAC_GenAlm_01** collects all alarms within an automation station. A group alarm is forwarded as BV object for controlling a collective failure signaling lamp to a BO object and for display in the MCL.
A further BV object enables central acknowledgement of all alarms from the MCL or a local operating device.
A BI object is available for acknowledging the alarms with an acknowledgement button in the control cabinet.
The acknowledgement pulse is linked to a BO object for interfacing in relay circuits (e.g. frost protection relay).
All alarms are displayed in the target visualization of the PLC control.

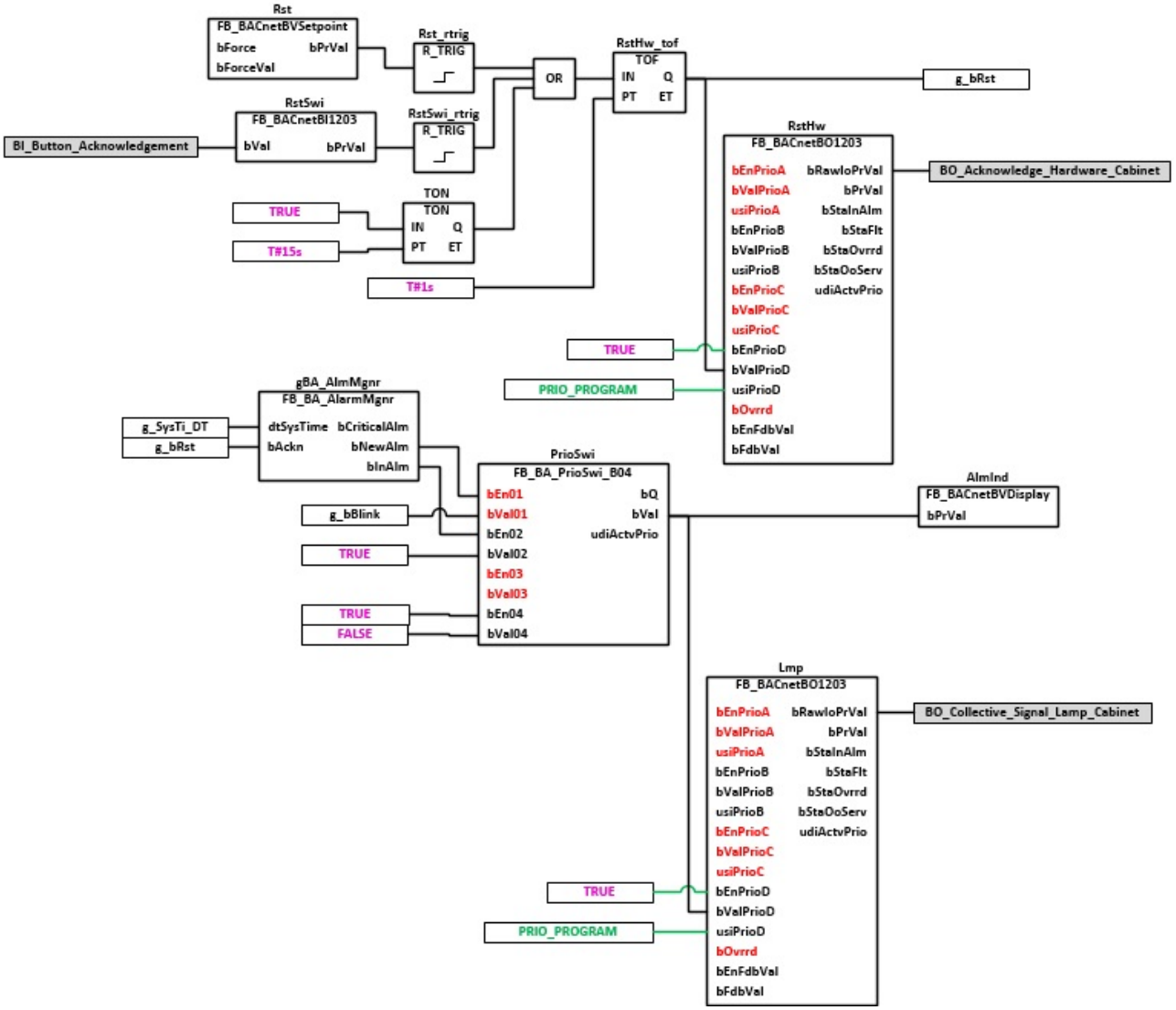
The template **BAC_GenAlm_01** generates the required Globale Variablen_Alarming [▶ 347] for alarm purposes.

The variables, which are linked with the process image of the input and output level in the PLC, can be found under **IO linking**.

**Interface**

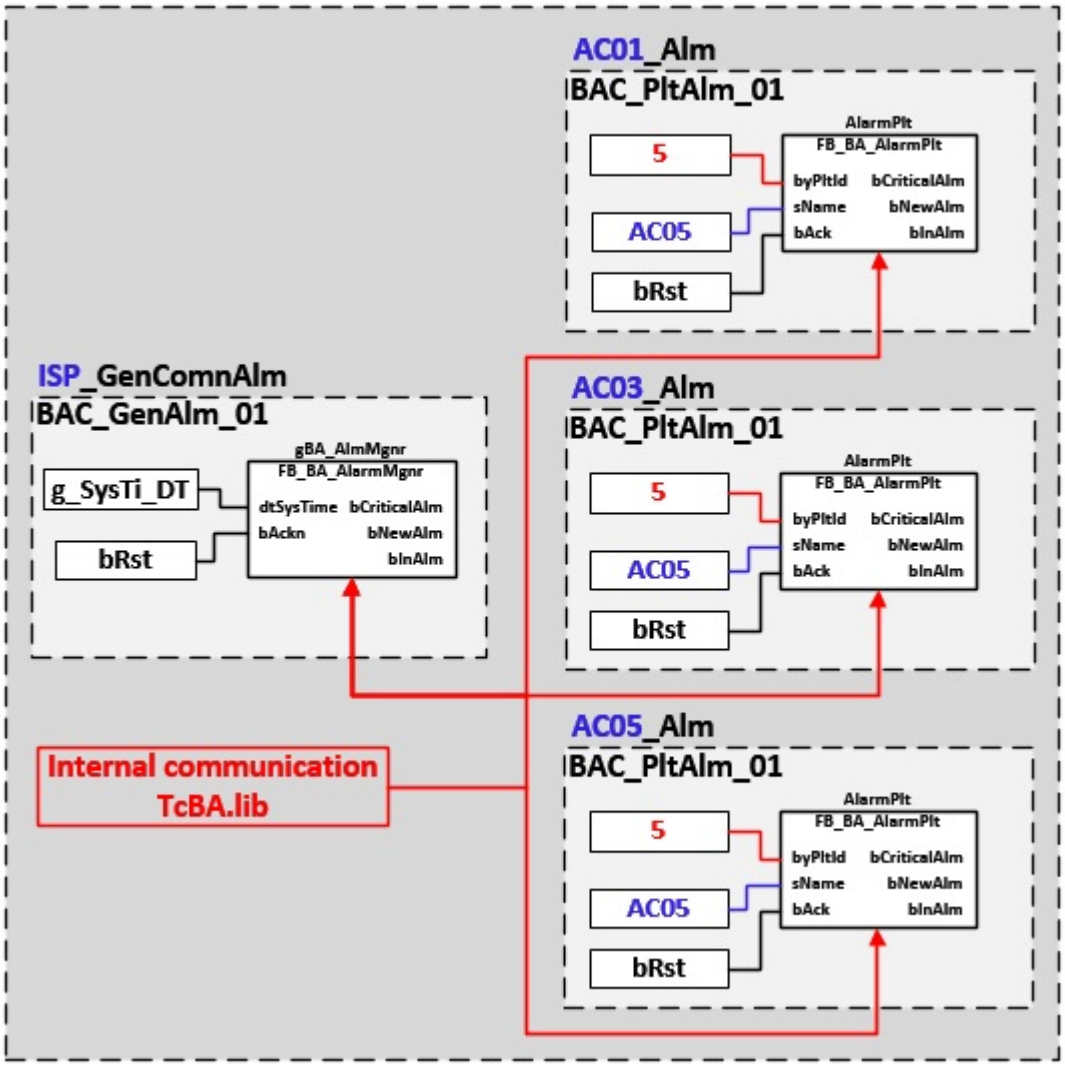


### Block diagram

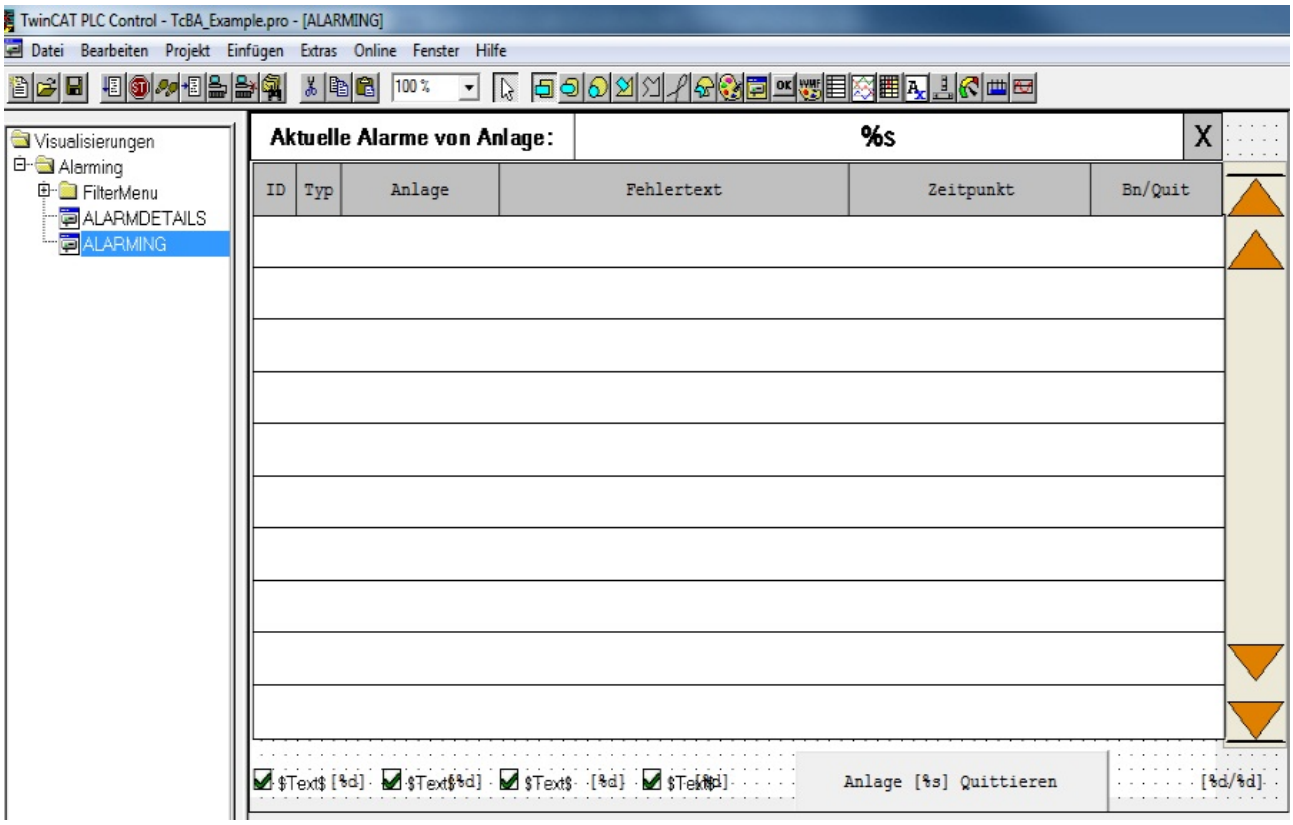



### Block diagram of the internal communication

The function block FB_BA_AlarmPlt [▶ 183] collects the alarms of a plant. The communication structure **gBA_AlmMgnr** is used to transfer the group alarms of the plants to the function block FB_BA_AlarmMgnr [▶ 181]. This function block forms a further group alarm to consolidate the alarms from all plants to a global group alarm. The communication structure **gBA_AlmMgnr** is used to transfer a global acknowledgement pulse to all alarm function blocks within the PLC program. In addition, the system time is transferred to the alarm function blocks so that a time stamp can be formed. The communication structure is an internal component of TcBa.lib


AC01_Alm
BAC_PltAlm_01
AlarmPlt
FB_BA_AlarmPlt
5
byPltId
bCriticalAlm
sName
bNewAlm
AC05
bAck
bInAlm
bRst
ISP_GenComnAlm
BAC_GenAlm_01
gBA_AlmMgnr
FB_BA_AlarmMgnr
g_SysTi_DT
dtSysTime
bCriticalAlm
bAckn
bNewAlm
bRst
bInAlm
AC03_Alm
BAC_PltAlm_01
AlarmPlt
FB_BA_AlarmPlt
5
byPltId
bCriticalAlm
sName
bNewAlm
AC05
bAck
bInAlm
bRst
AC05_Alm
BAC_PltAlm_01
AlarmPlt
FB_BA_AlarmPlt
5
byPltId
bCriticalAlm
sName
bNewAlm
AC05
bAck
bInAlm
bRst
Internal communication
TcBA.lib

### Target visualisation PLC Control




### VAR CONSTANT

```
PLT_NUM    : BYTE := 0;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| Rst | FB_BACnetBVSetpoint [▶ 96] | BV object for acknowledging all alarms in a controller |
| RstSwi | FB_BACnetBI1203 [▶ 72] | BI object for connecting an acknowledgement button. |
| Rst_rtrig | R_TRIG | Flank evaluation of the acknowledgement signal from BV object Rst |
| RstSwi_rtrig | R_TRIG | Flank evaluation of the acknowledgement signal from BI object RstSwi |
| RstAuto | TON | Start-up delay automatically acknowledges all alarms 15 seconds after a restart of the PLC. |

| Instance | Type | Task |
|---|---|---|
| | OR | Consolidates the automatic acknowledgement commands after a PLC restart, BV object (acknowledgement via MCL) or BI object (reset button in the control cabinet) |
| RstHw_tof | TOF | Extension of the acknowledgement pulse to one second |
| RstHw | FB_BACnetBO1203 [▶ 81] | The BO object is used to output the acknowledgement pulse to a digital output, e.g. for interfacing with a frost protection relay in the control cabinet. |
| gBA_AlmMgnr | FB_BA_AlarmMgnr [▶ 181] | The function block FB_BA_AlarmMgnr [▶ 181] is the central function block of this template.<br>It collects the alarms of the entire controller and bundles them into a group alarm.<br>The alarms are generated within the plant-related templates by the function block FB_BA_Alarm. [▶ 179]. |
| PrioSwi | FB_BA_PrioSwi_B04 [▶ 213] | If an alarm is not acknowledged, the output **bNewAlarm** of the function block becomes TRUE. In this case the priority switch **PrioSwi** generates a flashing signal and the output **bVal** is switched through. The downstream fault indicator lamp at the BO object and the BV object starts flashing. Once the alarm has been acknowledged, the flashing switches to continuous light. If the group alarm goes off, the output of **PrioSwi**is set to FALSE. When a new alarm is received, the common alarm lamp starts blinking again until it is acknowledged again. (new value message) |
| AlmInd | FB_BACnetBVDisplay [▶ 95] | BV object for displaying a group alarm |
| Lmp | FB_BACnetBO1203 [▶ 81] | BO object for controlling a collective failure signaling lamp in the control cabinet |

### IO linking

Variables for linking with the Bus Terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_Button_Acknowledgement | BOOL | | Input | Digital input - acknowledgement button - message - triggered |
| BO_Acknowledge_Hardware_Cabinet | BOOL | | Output | Digital output - acknowledgement hardware control cabinet - switching command - on/off |
| BO_Collective_Signal_Lamp_Cabinet | BOOL | | Output | Digital output - collective message lamp control cabinet - switching command - on/off |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.4 BAC_GenComnMsg_01

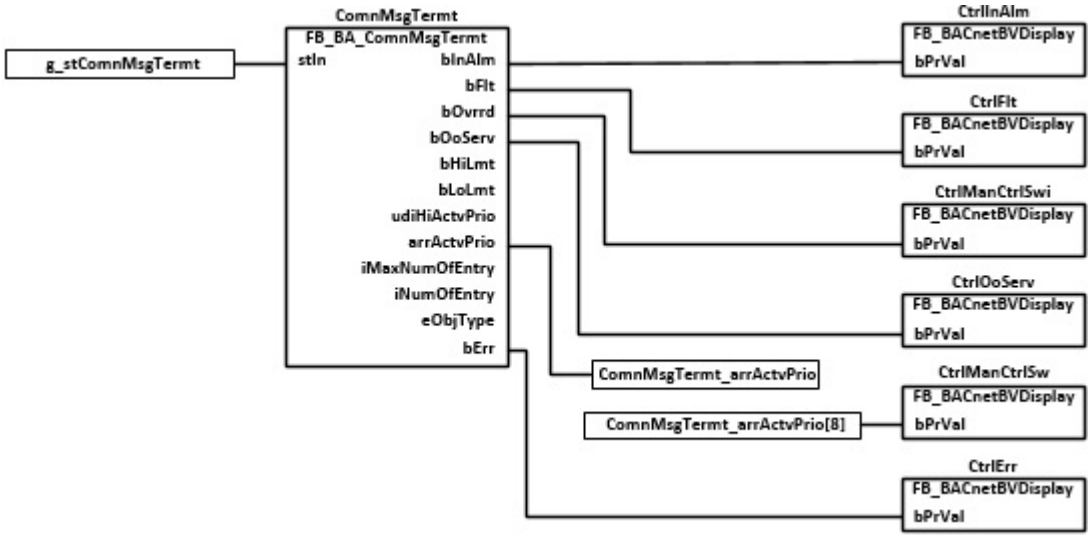
### Functional description

The template **BAC_GenComnMsg_01** collects all event messages of the BACnet objects within an automation station. For display of these messages at the management and control level (OWS) or at a local operator display, the main messages are linked with BV objects.

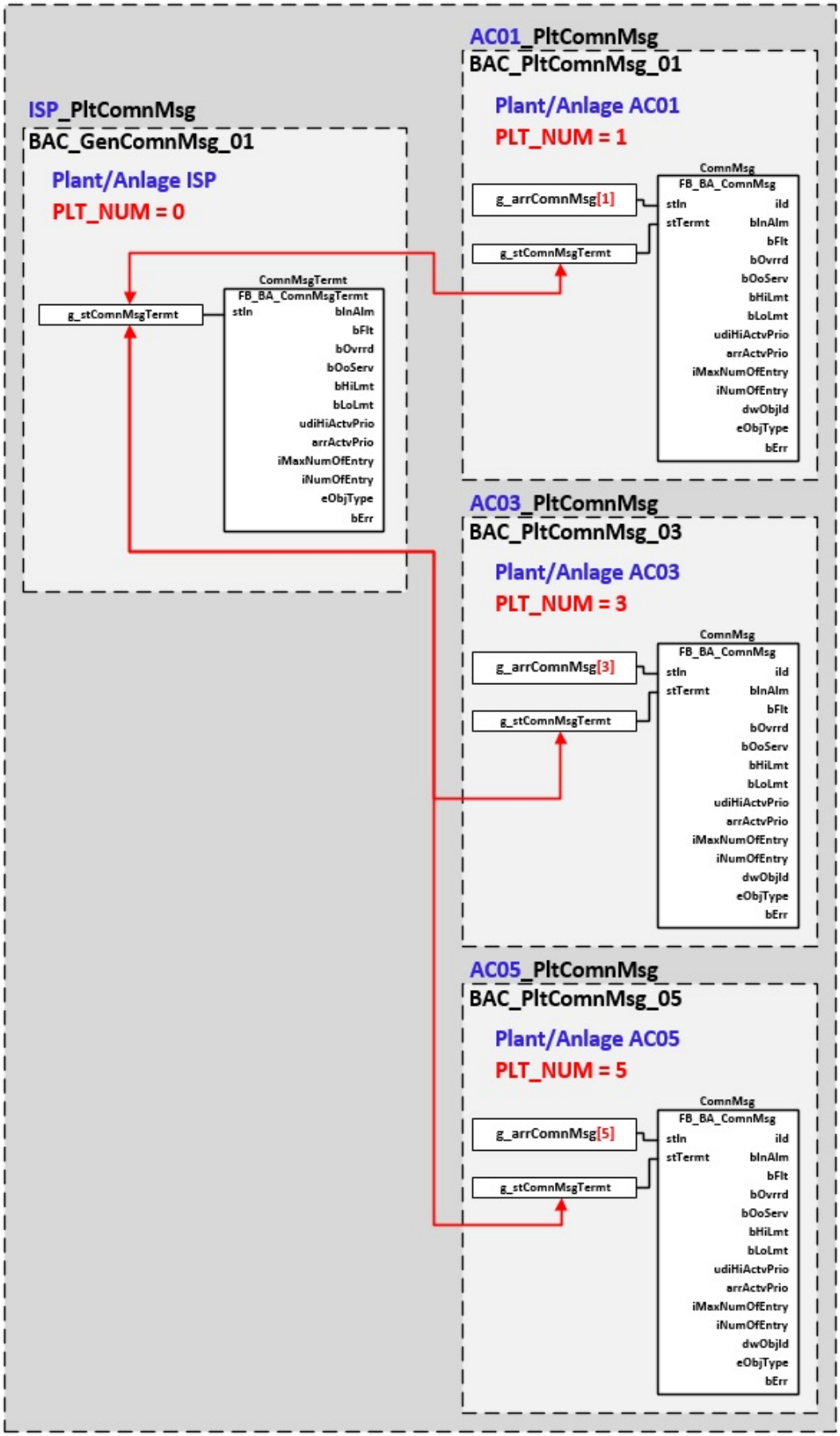
### Interface

```
BAC_GenComnMsg_01
```

### Block diagram



**Block diagram networking**



**Program description**

| Instance | Type | Task |
|---|---|---|
| ComnMsgTermt | FB_BA_ComnMsgTermt [▶ 200] | The function block **ComnMsgTermt** collects event messages of the BACnet objects in the complete device or controller. With the globally declared variable g_stComnMsgTermt [▶ 357] the function block receives the messages of all message collectors on the plant level. To display these collective messages, some information of the function block **ComnMsgTermt** is linked to BV objects. |
| CtrlInAlm | FB_BACnetBVDisplay [▶ 95] | Controller level - collective message of the status flags of all BACnet objects in the device |
| CtrlFlt | FB_BACnetBVDisplay [▶ 95] | Controller level - collective message of the BACnet fault objects in the device |
| CtrlManCtrlSw | FB_BACnetBVDisplay [▶ 95] | Controller level - collective message manual override of one of the BACnet objects in the device is at Prio 8 |
| CtrlManCtrlSwi | FB_BACnetBVDisplay [▶ 95] | Controller level - at one of the BACnet-output objects in the device the local mechanical priority operation is enabled |
| CtrlOoServ | FB_BACnetBVDisplay [▶ 95] | Controller level - one of the BACnet objects in the device is OutOfService |
| CtrlErr | FB_BACnetBVDisplay [▶ 95] | Controller level - one of the BACnet objects in the device is faulty, e.g. ADS error |

**Version history**

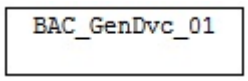
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.5 BAC_GenDvc_01

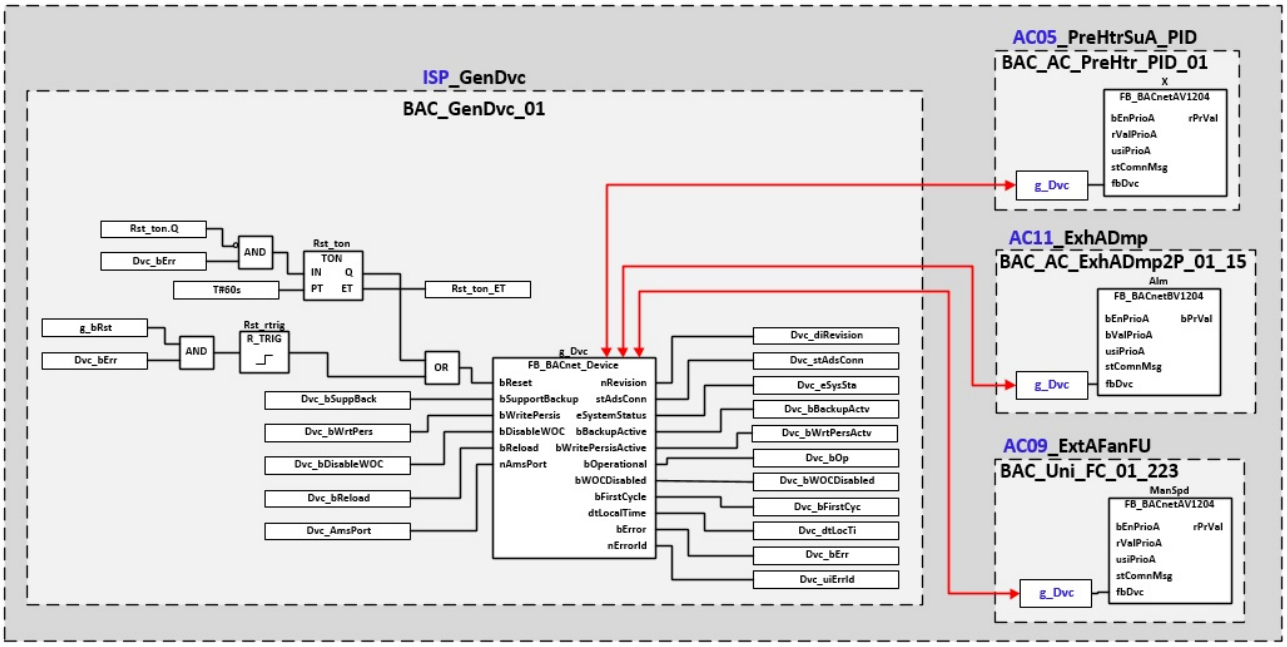
**Functional description**

The template BAC_GenDvc_01 contains the function block FB_BACnet_Device. It is responsible for interfacing of the PLC with the BACnet device object (server).
**It is therefore imperative that this template is placed once in the PLC program!**

**Interface**



### Block diagram




### Program description

| Instance | Type | Task |
|---|---|---|
| g_Dvc | FB_BACnet_Device | The function block "FB_BACnet_Device" is used to read the state of the local BACnet device object (System_Status) and output it in the PLC program. In addition, the process data "Server Control" is used to control the local BACnet server (activation and handling of the PLC backup) |
| **OR**, **Rst_ton**, **Rst_trig** | | In case of an error **Dvc_bErr** the BACnet device object is reset either automatically via the timing element **Rst_ton** or via the global acknowledgement **g_bRst**. |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.6 Global_Variables_General

In the PLC the following function blocks, variables and constants can be found under **Resources\GlobalVariables\Globale_Variablen_General**:

| **VAR_GLOBAL** | | | |
|---|---|---|---|
| gBA_AlmMgnr | FB_BA_AlarmMgnr | | Alarm collector with new value message |
| g_stComnMsg Termt | ST_BA_ComnMsgTermt [▶ 326] | | CommonMessage Abschluss/Terminated<br>**This structure may only be accessed once from a template per controller!**<br>The following templates access the global structure:<br>BAC_GenComnMsg_01 [▶ 353] |
| g_bRst | BOOL | | Reset<br>**The variable may only be written to once per control!**<br>The following templates write to the global variable:<br>- BAC_GenAlm_01 [▶ 349] |

| | | | |
|---|---|---|---|
| g_bBlink | BOOL | | Flash sequence<br>**The variable may only be written to once per control!**<br>The following templates write to the global variable:<br>- BAC_GenSys_01 [▶ 363] |
| g_WthT_rPrVal | REAL | | weather temperature<br>**The variable may only be written to once per control!**<br>The following templates write to the global variable:<br>- BAC_AI_WthT_01 [▶ 681] |
| g_WthTDamp_rPrVal | REAL | | Weather temperature damped<br>**The variable may only be written to once per control!**<br>The following templates write to the global variable:<br>- BAC_GenWthT_01 [▶ 364] |
| g_SysTi_DT | DT | | System time of the target system<br>**The variable may only be written to once per control.**<br>The following templates write to the global variable:<br>- BAC_GenSys_01 [▶ 363] |
| g_arrComnMsg | ARRAY [MIN_PLT_NUM..MAX_PLT_NUM] OF ST_BA_ComnMsg [▶ 326] | | Array Recording the object information |
| g_stSeqLinkT | ARRAY [MIN_PLT_NUM..MAX_PLT_NUM] OF ST_BA_SeqLink [▶ 327] | | Array Sequence link structure temperature |
| g_stSeqLinkH | ARRAY [MIN_PLT_NUM..MAX_PLT_NUM] OF ST_BA_SeqLink [▶ 327] | | Array Sequence link structure humidity |
| **END_VAR** | | | |
| **VAR_GLOBAL CONSTANT** | | | |
| Gemeinsame Meldungen//Common message | | | |
| gBA_cMaxArrComnMsg | INT | 100 | Common messages - Maximum number per collective message |
| gBA_cMaxArrComnMsgTermt | INT | 12 | Maximum number of common messages |
| Alarmsystem//Alarming | | | |
| gBA_MAX_NUM_OF_ALARMS | UDINT | 100 | Alarm system - Maximum number of alarms in the alarm system |
| gBA_MAX_NUM_OF_PLANT_DEFS | BYTE | 10 | Alarm system - Maximum number of plants |
| Limit value plant number | | | |
| MAX_PLT_NUM | BYTE | 1 | Maximum Plant Number |
| Operation Mode Air Conditioning | | | |
| OPMOD_AC_OFF | UDINT | 1 | Off |
| OPMOD_AC_ON | UDINT | 2 | On |
| OPMOD_AC_EMERG | UDINT | 3 | Emergency |

| | | | |
|---|---|---|---|
| OPMOD_AC_MANOFF | UDINT | 4 | Manual off |
| OPMOD_AC_MANON | UDINT | 5 | Manual on |
| OPMOD_AC_FRST | UDINT | 6 | Frost |
| OPMOD_AC_SMEXTTPRG | UDINT | 7 | Smoke extraction programm |
| OPMOD_AC_SMEXTTSUA | UDINT | 8 | Smoke extraction supply |
| OPMOD_AC_SMEXTTEXHA | UDINT | 9 | Smoke extraction exhaust |
| OPMOD_AC_FIRE | UDINT | 10 | Fire |
| OPMOD_AC_NGTCOL | UDINT | 11 | Night cooling |
| OPMOD_AC_COLDWNPRTC | UDINT | 12 | Cool down protection |
| OPMOD_AC_OVRHTGPRTC | UDINT | 13 | Over heating protection |
| OPMOD_AC_ALM | UDINT | 14 | Alarm |
| OPMOD_AC_FORCVENT | UDINT | 15 | Forced ventilation |
| OPMOD_AC_CENTSWIOFF | UDINT | 16 | Central switch-off |
| Plant step start programm Air Conditioning | | | |
| PLTSTP_AC_OFF | UDINT | 1 | Off |
| PLTSTP_AC_PRERI | UDINT | 2 | pre-rinse |
| PLTSTP_AC_DMP | UDINT | 3 | Damper |
| PLTSTP_AC_FANSUA | UDINT | 4 | Fan supply air |
| PLTSTP_AC_FANEXTA | UDINT | 5 | Fan extract air |
| PLTSTP_AC_ENTEMPCTRL | UDINT | 6 | Enable temperature control, Sequence link |
| PLTSTP_AC_ENHUMCTRL | UDINT | 7 | Enable humidity control, Sequence link |
| PLTSTP_AC_ENLMTMONIT | UDINT | 8 | Enable limit monitoring |
| PLTSTP_AC_ON | UDINT | 9 | On |
| Operation Mode switch | | | |
| OPMOD_SWI_AUTO | UDINT | 1 | Auto |
| OPMOD_SWI_OFF | UDINT | 2 | Off |
| OPMOD_SWI_STP01 | UDINT | 3 | Step 1 |
| OPMOD_SWI_STP02 | UDINT | 4 | Step 2 |

| | | | |
|---|---|---|---|
| OPMOD_SWI_STP03 | UDINT | 5 | Step 3 |
| Operation Mode manual | | | |
| OPMOD_MAN_AUTO | UDINT | 1 | Auto |
| OPMOD_MAN_OFF | UDINT | 2 | Off |
| OPMOD_MAN_STP01 | UDINT | 3 | Step 1 |
| OPMOD_MAN_STP02 | UDINT | 4 | Step 2 |
| OPMOD_MAN_STP03 | UDINT | 5 | Step 3 |
| Operation Mode manual | | | |
| OPMOD_AUTO | UDINT | 1 | Auto |
| OPMOD_OFF | UDINT | 2 | Manuell |
| Operation Mode heat distribution | | | |
| OPMOD_HTGCIR_AUTO | UDINT | 1 | Auto |
| OPMOD_HTGCIR_OFF | UDINT | 2 | Off |
| OPMOD_HTGCIR_DAY | UDINT | 3 | Day |
| OPMOD_HTGCIR_NIGHT | UDINT | 4 | Night |
| OPMOD_HTGCIR_FROST | UDINT | 5 | Frost |
| Operation Mode 3P valve | | | |
| OPMOD_3PVLV_AUTO | UDINT | 1 | Auto |
| OPMOD_3PVLV_OFF | UDINT | 2 | Off |
| OPMOD_3PVLV_OPEN | UDINT | 3 | Open |
| OPMOD_3PVLV_CLOSE | UDINT | 4 | Close |
| Operation Mode 2P Damper | | | |
| OPMOD_2P_AUTO | UDINT | 1 | Operation Mode 2P Damper Auto |
| OPMOD_2P_CLOSE | UDINT | 2 | Operation Mode 2P Damper Close |
| OPMOD_2P_OPEN | UDINT | 3 | Operation Mode 2P Damper Open |
| Sequence Number controller<br>The order of the sequence numbers must be observed | | | |
| SEQNUM_T_REHTR | USINT | 1 | Sequence number reheater |
| SEQNUM_T_PREHTR | USINT | 2 | Sequence number preheater |
| SEQNUM_T_MIX | USINT | 3 | Sequence number mixed air |
| SEQNUM_T_ERC | USINT | 4 | Sequence number energy recovery |

| | | | |
|---|---|---|---|
| SEQNUM_T_COL | USINT | 5 | Sequence number cooler |
| SEQNUM_T_OFF | USINT | 6 | no sequence controller active |
| SEQNUM_H_HUMF | USINT | 1 | Sequence number humidifier |
| SEQNUM_H_DEHUMF | USINT | 2 | Sequence number dehumidifier |
| SEQNUM_H_OFF | USINT | 3 | no sequence controller active |
| atmospheric pressure | | | |
| AP | REAL | 1013.25 | |
| BACnet Priorität | | | |
| PRIO_SAFETY | USINT | 1 | Priorität Sicherheit |
| PRIO_DISTURBANCE | USINT | 3 | Priorität Störung |
| PRIO_LOCAL | USINT | 8 | Priorität lokaler Zugriff, Panel |
| PRIO_PROGRAM | USINT | 15 | Priorität Program, Auto |
| Alarming Priority | | | |
| ALM_PRIO_EMPTY | BYTE | 0 | no message |
| ALM_PRIO_NOTE | BYTE | 1 | various message |
| ALM_PRIO_WARNING | BYTE | 2 | warning |
| ALM_PRIO_ALARM | BYTE | 3 | Alarm without shutdown |
| ALM_PRIO_CRITICAL | BYTE | 4 | Alarm with shutdown<br><br>sets the output*bCriticalAlm* by the function blockAlarmPltinTemplateBAC_PltAlm_01 [▶ 365] |
| **END_VAR** | | | |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.7 BAC_GenNC_01

**Functional description**

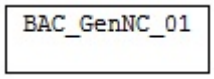
The template BAC_GenNC_01 provides eight BACnet notification classes for the transmission of object-integrated messages (intrinsic reporting). The intrinsic reporting of the BACnet objects within the templates is adapted to the use of these notification classes.

The definition of the notification classes is based on the German AMEV standard. (**A**rbeitskreis **M**aschinen- und **E**lektrotechnik staatlicher und kommunaler **V**erwaltungen)

**Loading of the template BAC_Gen_NC_01 is imperative! The template BAC_Gen_NC_01 may only be loaded once!**

Example notification class and sink

### Interface




### VAR CONSTANT

```
PLT_NUM    : BYTE := 0;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| NC10 | FB_BACnetNC1202 | Notification class 10 danger to life |
| NC20 | FB_BACnetNC1202 | Notification class 20 safety messages |
| NC30 | FB_BACnetNC1202 | Notification class 30 message indicates system failure or requires immediate intervention |
| NC40 | FB_BACnetNC1202 | Notification class 40 fault message |
| NC50 | FB_BACnetNC1202 | Notification class 50 maintenance message |
| NC60 | FB_BACnetNC1202 | Notification class 60 system message |
| NC70 | FB_BACnetNC1202 | Notification class 70 manual intervention |
| NC80 | FB_BACnetNC1202 | Notification class 80 other messages |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

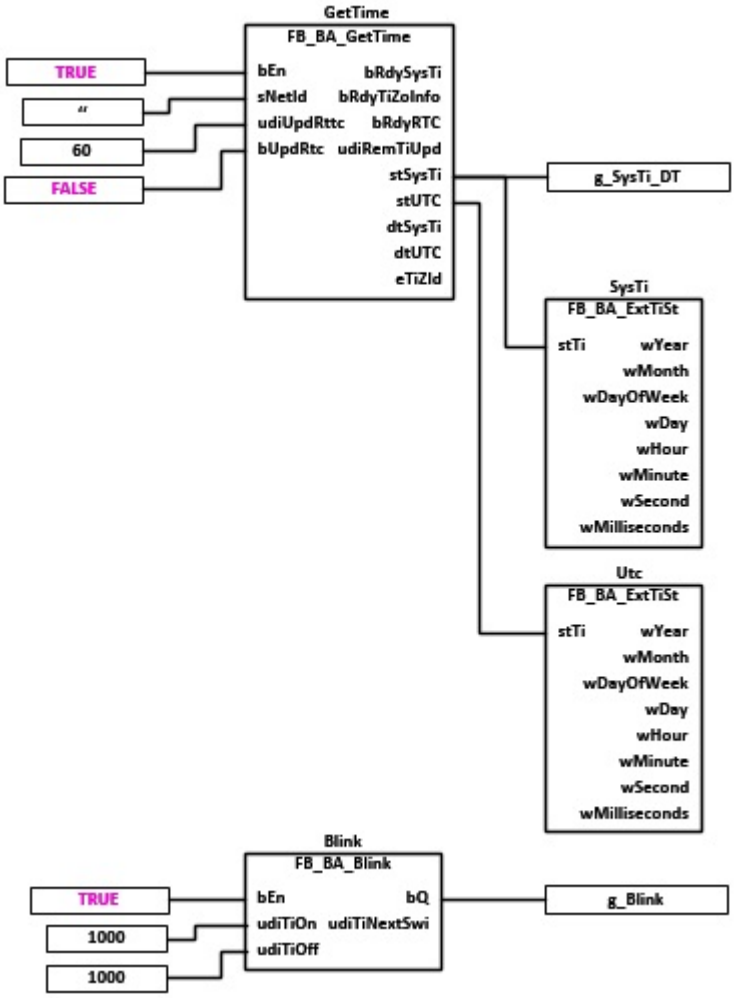
# 9.8 BAC_GenSys_01

### Functional description

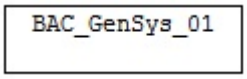
The template **BAC_GenSys_01** is a system function template.
The template reads the system time from the PC and maps the time information in the PLC to variables.
Various time specifications are required within the templates for plant and room automation.
In addition, a flash pulse is generated and made available to other templates via a global variable, e.g. see the block diagram in the template BAC_PltAlm_01 [▶ 365].

**Loading of the template BAC_GenSys_01 is imperative! The template BAC_GenSys_01 may only be loaded once!**

### Block diagram




### Interface




### Program description

| Instance | Type | Task |
|---|---|---|
| GetTime | FB_BA_GetTime [▶ 320] | This function block is used to realize 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 |

| Instance | Type | Task |
|---|---|---|
| | | 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. The time is output at the outputs **stSysTi** for the read system time and **stUtcTi** 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. |
| SysTi | FB_BA_ExtTiSt [▶ 319] | The function block **SysTi** resolves the time structure of the read target system into the different components. |
| Utc | FB_BA_ExtTiSt [▶ 319] | The function block **Utc** resolves the Coordinated Universal Time (Universal Time Coordinated - previously referred to as GMT, Greenwich Mean Time) into the different components. |
| Blink | FB_BA_Blink [▶ 317] | The function block **Blink** generates a flash pulse and makes it available for other templates via a global variable g_Blink [▶ 357], e.g. collective failure signaling lamp |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.9 BAC_GenWthT_01

**Functional description**

The template **BAC_GenWthT_01** reads the measured outside temperature via the global variable g_WthT_rPrVal [▶ 357].

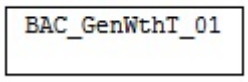
The measured outside temperature is used to calculate an averaged or attenuated outside temperature. This is made available to other templates via the global variable g_WthTDamp_rPrVal [▶ 357].

The BACnet object WthTLmtCrit describes the global variable g_WthTLmtCrit. It describes an outside temperature value below which the frost protection mode should be activated in the systems.

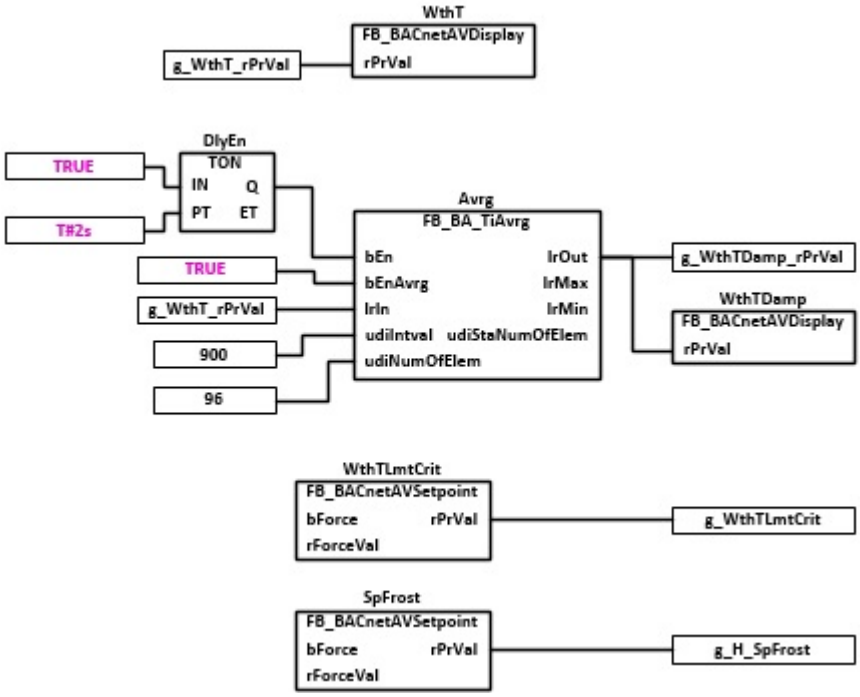
The SpFrost AV object is used to describe the global variable g_H_SpFrost. This global value can be used to assign a setpoint for frost protection operation to one or more plants.

**For the application of many other templates it is necessary to position this template once!**

**Interface**



### Block diagram




### Program description

| Instance | Type | Task |
|---|---|---|
| WthT | FB_BACnetAVDisplay [▶ 69] | AV object for displaying the current outside temperature |
| WthTLmtCrit | FB_BACnetAVSetpoint [▶ 69] | AV object for entering and providing a global value for the critical outside temperature |
| SpFrost | FB_BACnetAVSetpoint [▶ 69] | AV object for input and provision of a global frost protection setpoint |
| DlyEn | TON | Delays reading of the first temperature value by 2 seconds after the start of the PLC |
| Avrg | FB_BA_TiAvrg [▶ 177] | The function block Avrg measures the outside temperature every 15 minutes (900 seconds) and writes the value into a FIFO buffer (first in first out). The buffer has capacity for 96 elements. This results in a timeframe of 24 hours for averaging of the outside temperature. |
| WthTDamp | FB_BACnetAVDisplay [▶ 69] | Output of the value of the averaged outside temperature |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.10 BAC_PltAlm_01

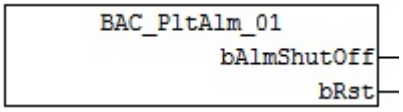
### Application

The template **BAC_PltAlm_01** deals with the following tasks in a plant:

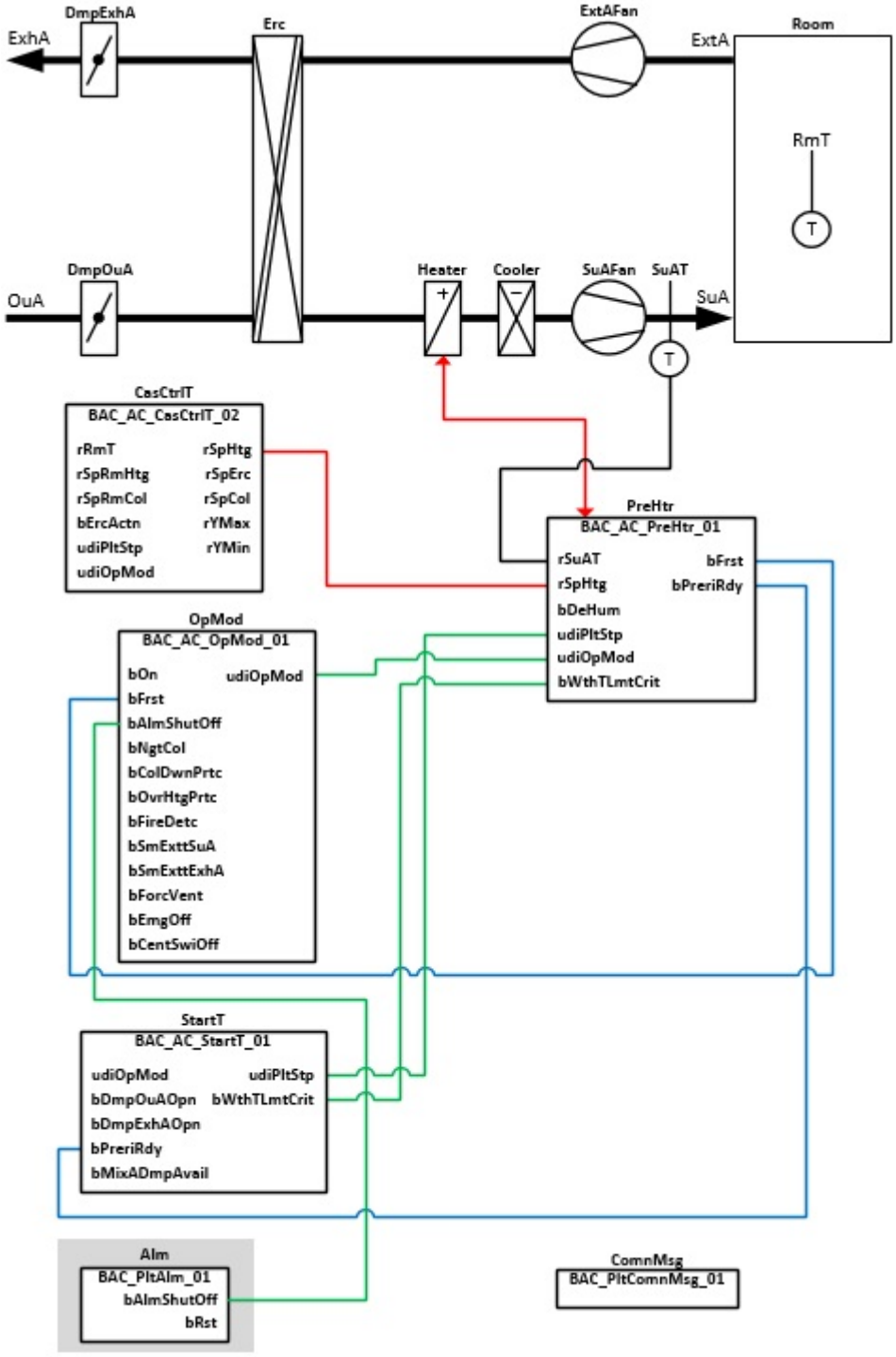
- Collecting the alarms of a plant and bundling them into a group alarm or plant shutdown alarm **bAlmShutOff**
- Display to indicate that an alarm is present in a plant.

- Plant-related acknowledgement and resetting of the alarms

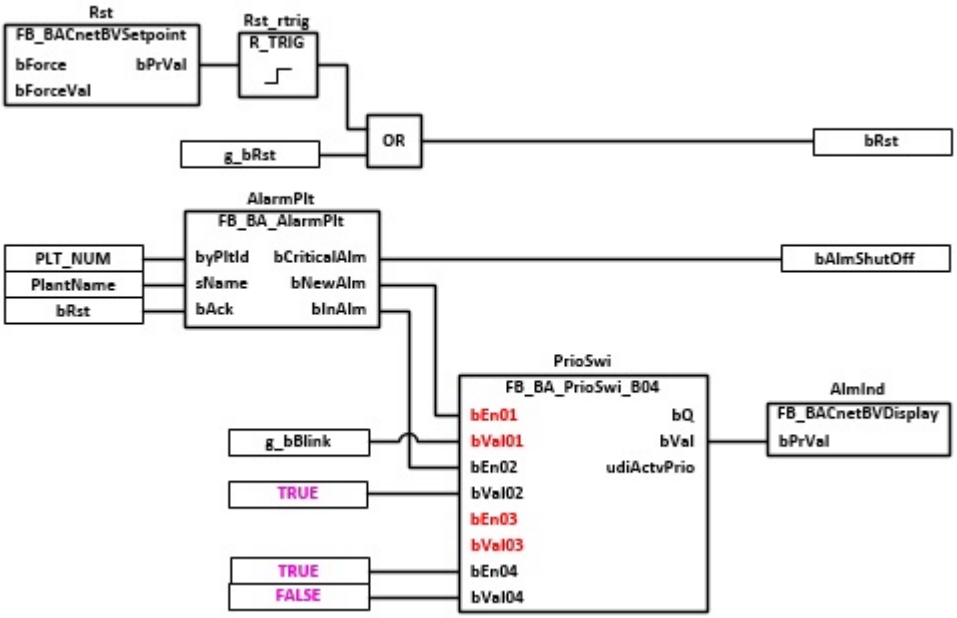
### Interface




### Plant diagram

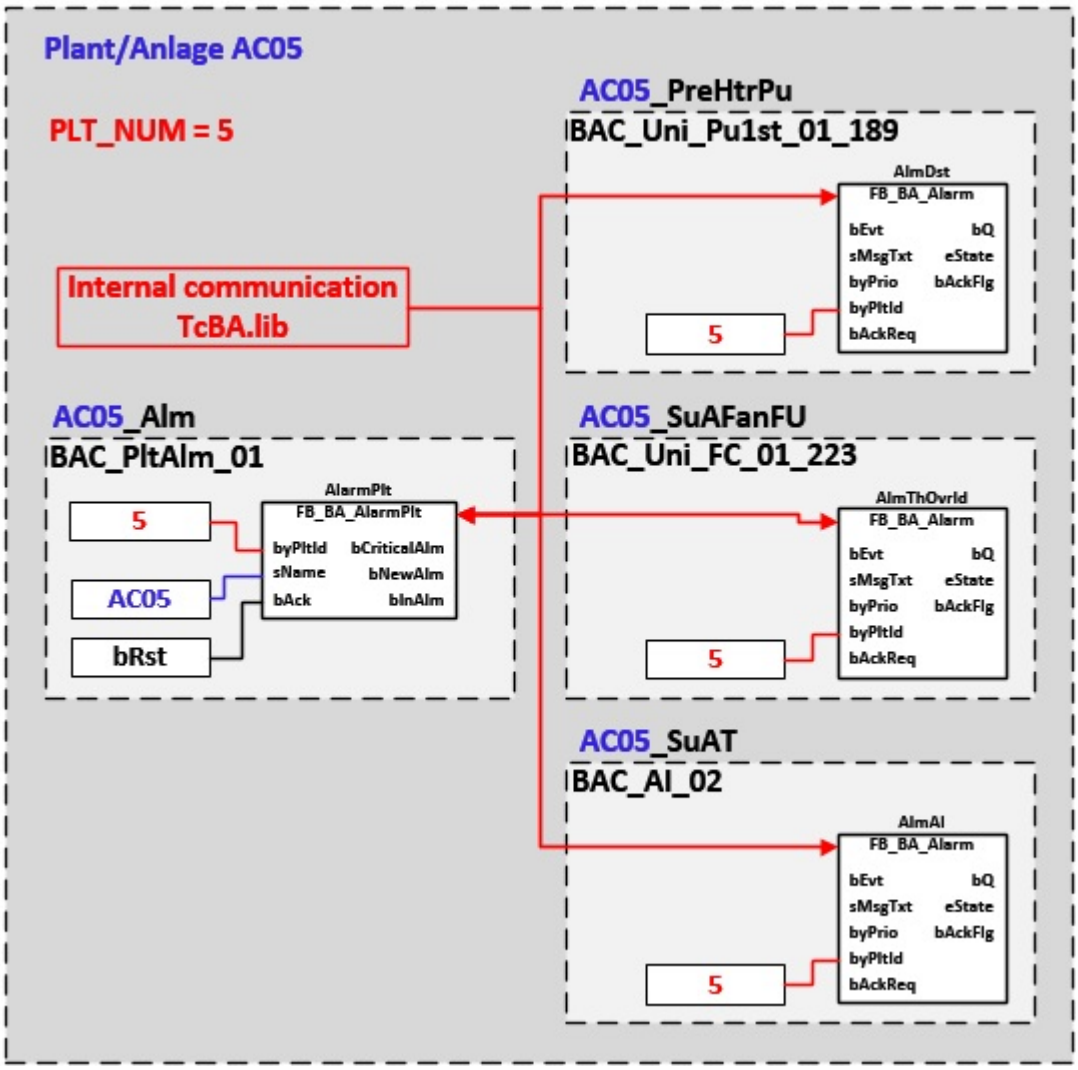


### Block diagram




### Block diagram of the internal communication structure

Within the templates the alarms are logged with the function block FB_BA_Alarm [▶ 179]. The communication between the function block FB_BA_Alarm [▶ 179] and the alarm collector FB_BA_AlarmPlt [▶ 183] takes place within the library via a data structure.
It is important that the value of the variable byPltId at the sender of the alarm FB_BA_Alarm [▶ 179] and the receiver of the alarm FB_BA_AlarmPlt [▶ 183] is identical.




### VAR_OUTPUT

```
bAlmShutOff    : BOOL;
bRst           : BOOL;
```

**bAlmShutOff:** A plant shutdown alarm is active.

**bRst:** When an acknowledgement is triggered, this output is active for one cycle. The pulse can be used to acknowledge faults.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltIniAlm_01 by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of various plant events within the templates of a plant is carried out within the template **BAC_PltIniComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| Rst | FB_BACnetBVSetpoint [▶ 96] | BV object for local acknowledgement of a collective plant alarm. |
| Rst_rtrig | R_TRIG | The function block Rst_rtrig evaluates the rising edge of the local acknowledgement of BV object Rst. |
| | OR | The OR link consolidates the local acknowledgement of the BACnet object and the global fault message acknowledgement.<br>The global acknowledgement of alarms from all systems of a controller via the variable g_bRst [▶ 357] is triggered in the template BAC_GenAlm_01 [▶ 349]. |
| AlarmPlt | FB_BA_AlarmPlt [▶ 183] | The function block AlarmPlt is the central function block of this template.<br>It collects the alarms of a plant and pools them in a group alarm. |
| PrioSwi | FB_BA_PrioSwi_B04 [▶ 213] | If an alarm is not acknowledged, the output *bNewAlarm* of the function block *AlarmPlt* becomes TRUE. In this case the priority switch *PrioSwi* generates a flashing signal and the output *bVal* is switched through. A new, unacknowledged alarm is pending. (new value message). If an acknowledged alarm is pending, the priority switch switches through TRUE. The output *bVal* is permanently on. |
| AlmInd | FB_BACnetBVDisplay [▶ 95] | Display of the group alarms of a plant with new value message. |

### Version history

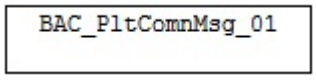
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.11 BAC_PltComnMsg_01

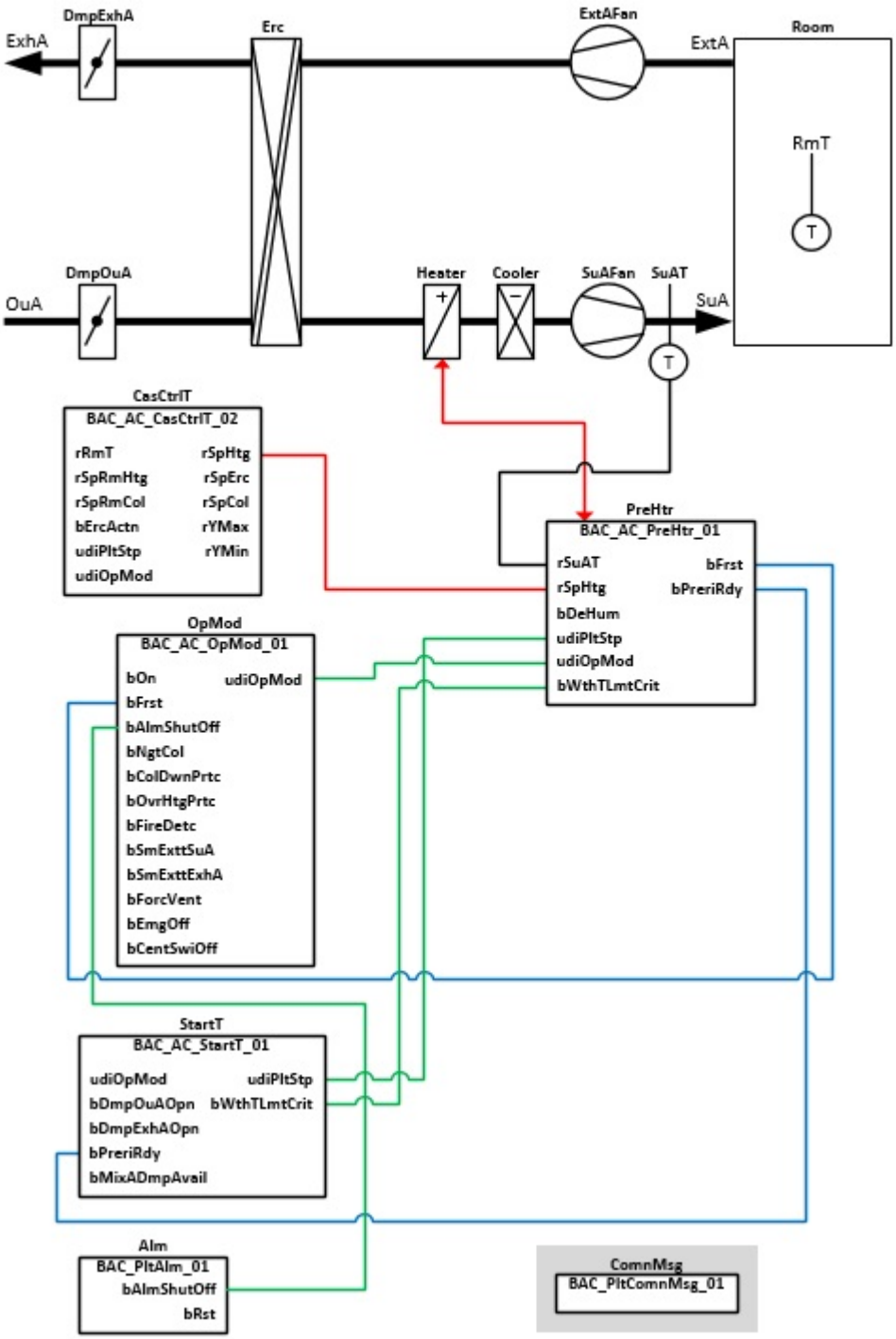
### Functional description

The template **BAC_PltComnMsg_01** collects all event messages of the BACnet objects within a plant. For display of these messages at the management and control level (OWS) or at a local operator display, the main messages are linked with a BV object.

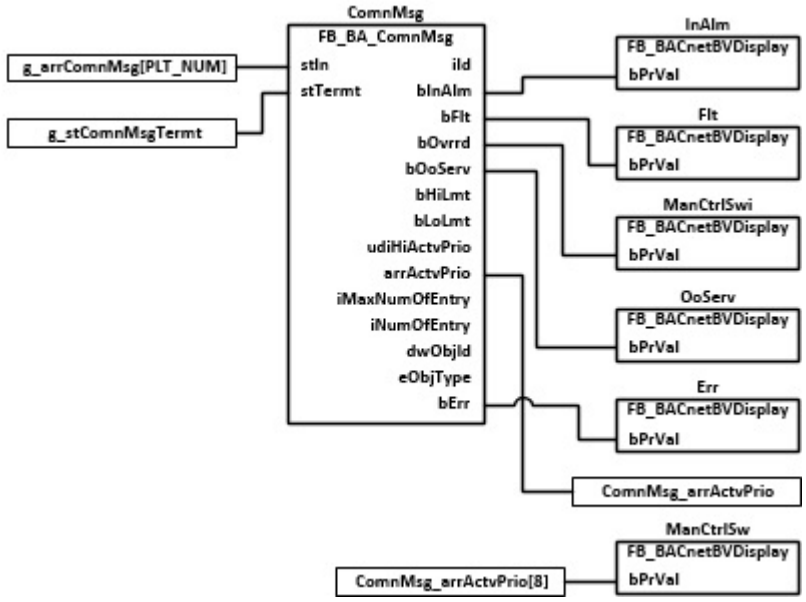
### Interface



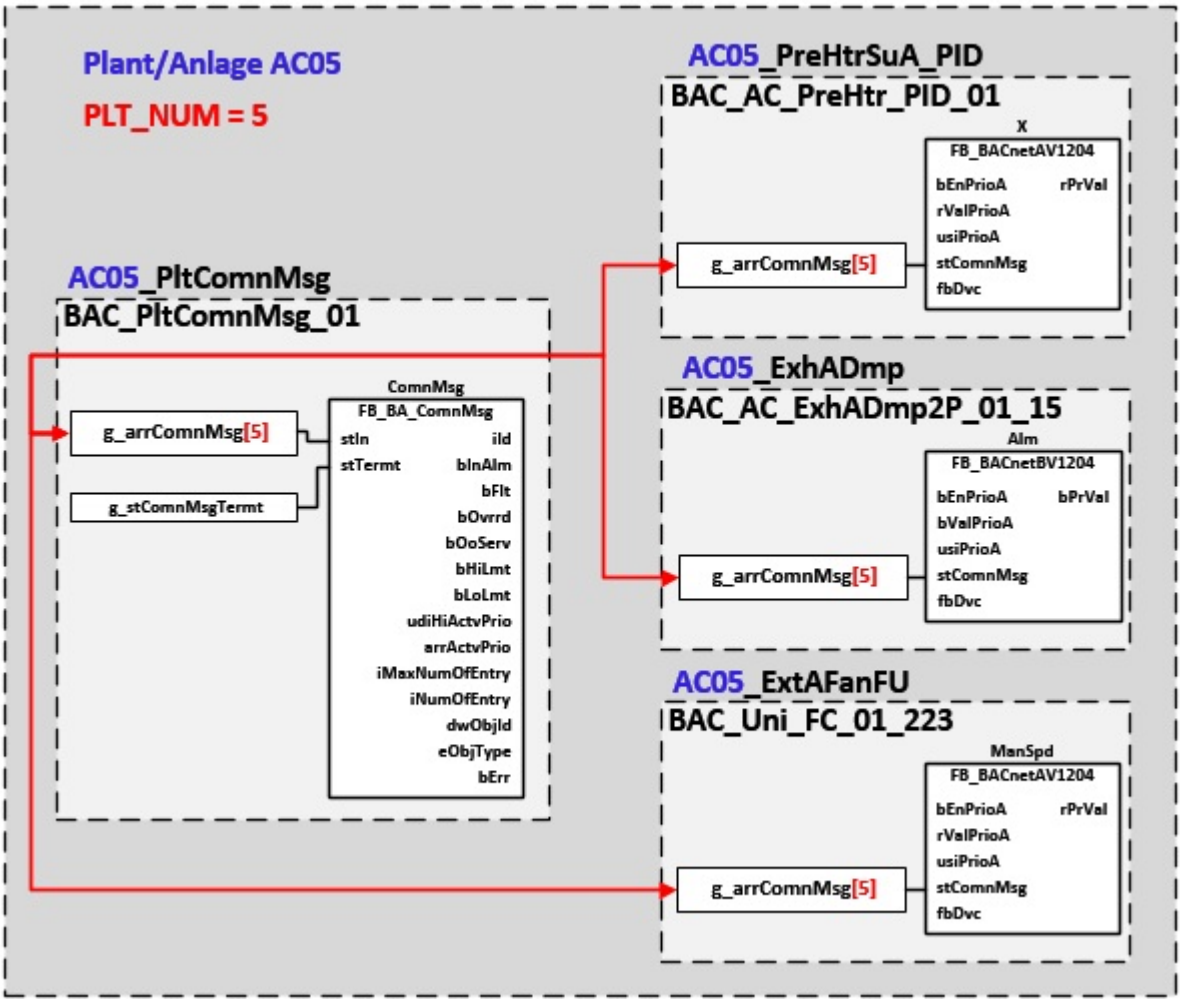
**Plant**


DmpExhA
ExhA
Erc
ExtAFan
ExtA
Room
RmT
T
DmpOuA
OuA
Heater
Cooler
SuAFan
SuAT
SuA
T
CasCtrlT
BAC_AC_CasCtrlT_02
rRmT
rSpRmHtg
rSpRmCol
bErcActn
udiPltStp
udiOpMod
rSpHtg
rSpErc
rSpCol
rYMax
rYMin
PreHtr
BAC_AC_PreHtr_01
rSuAT
rSpHtg
bDeHum
udiPltStp
udiOpMod
bWthTLmtCrit
bFrst
bPreriRdy
OpMod
BAC_AC_OpMod_01
bOn
bFrst
bAlmShutOff
bNgtCol
bColDwnPrtc
bOvrHtgPrtc
bFireDetc
bSmExttSuA
bSmExttExhA
bForcVent
bEmgOff
bCentSwiOff
udiOpMod
StartT
BAC_AC_StartT_01
udiOpMod
bDmpOuAOpn
bDmpExhAOpn
bPreriRdy
bMixADmpAvail
udiPltStp
bWthTLmtCrit
Alm
BAC_PltAlm_01
bAlmShutOff
bRst
ComnMsg
BAC_PltComnMsg_01

### Block diagram 01




### Block diagram 02




### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.

The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| ComnMsg | FB_BA_ComnMsg [▶ 197] | The ComnMsg function block collects event messages from the BACnet objects of a plant.<br>To display these collective messages, some information of the ComnMsg function block is linked to a BV object.<br><br>The transfer of the messages from the BACnet objects to the message function block ComnMsg is done by means of a one-dimensional array of the structure ST_BA_ComMsg.<br>Each plant reserves an element of this array via the plant number. The array for collecting the messages from the BACnet objects within a controller<br>is declared globally in the folder Global_Variables_General [▶ 357].<br><br>**It is important that the message collector FB_BA_ComnMsg and the BACnet objects within the templates of a plant are set to the same plant number!**<br><br>VAR CONSTANT<br>PLT_NUM : BYTE := 1; (*Anlagennummer//Plant Number*)<br>END_VAR<br><br>The plant number can be set in the project builder in the parameter menu of the templates or by a column within the Excel import. |
| InAlm | FB_BACnetBVDisplay [▶ 95] | Collective message of the status flag of the BACnet objects |
| Flt | FB_BACnetBVDisplay [▶ 95] | Collective message the BACnet Fault objects |
| ManCtrlSw | FB_BACnetBVDisplay [▶ 95] | Collective message manual override of one of the BACnet objects is on Prio 8 |
| ManCtrlSwi | FB_BACnetBVDisplay [▶ 95] | Local mechanical priority operation is enabled at one of the BACnet output objects |
| OoServ | FB_BACnetBVDisplay [▶ 95] | One of the BACnet objects in the plant is OutOfService |
| Err | FB_BACnetBVDisplay [▶ 95] | One of the BACnet objects is faulty, e.g. ADS error |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.12 BAC_AC_Humf_01

### Application

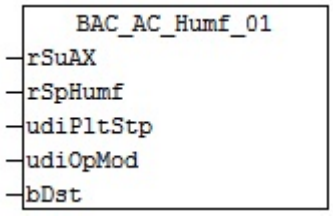
The template **BAC_AC_Humf_01** is used for controlling the humidifier.

The template **BAC_AC_Humf_01** is a call template.

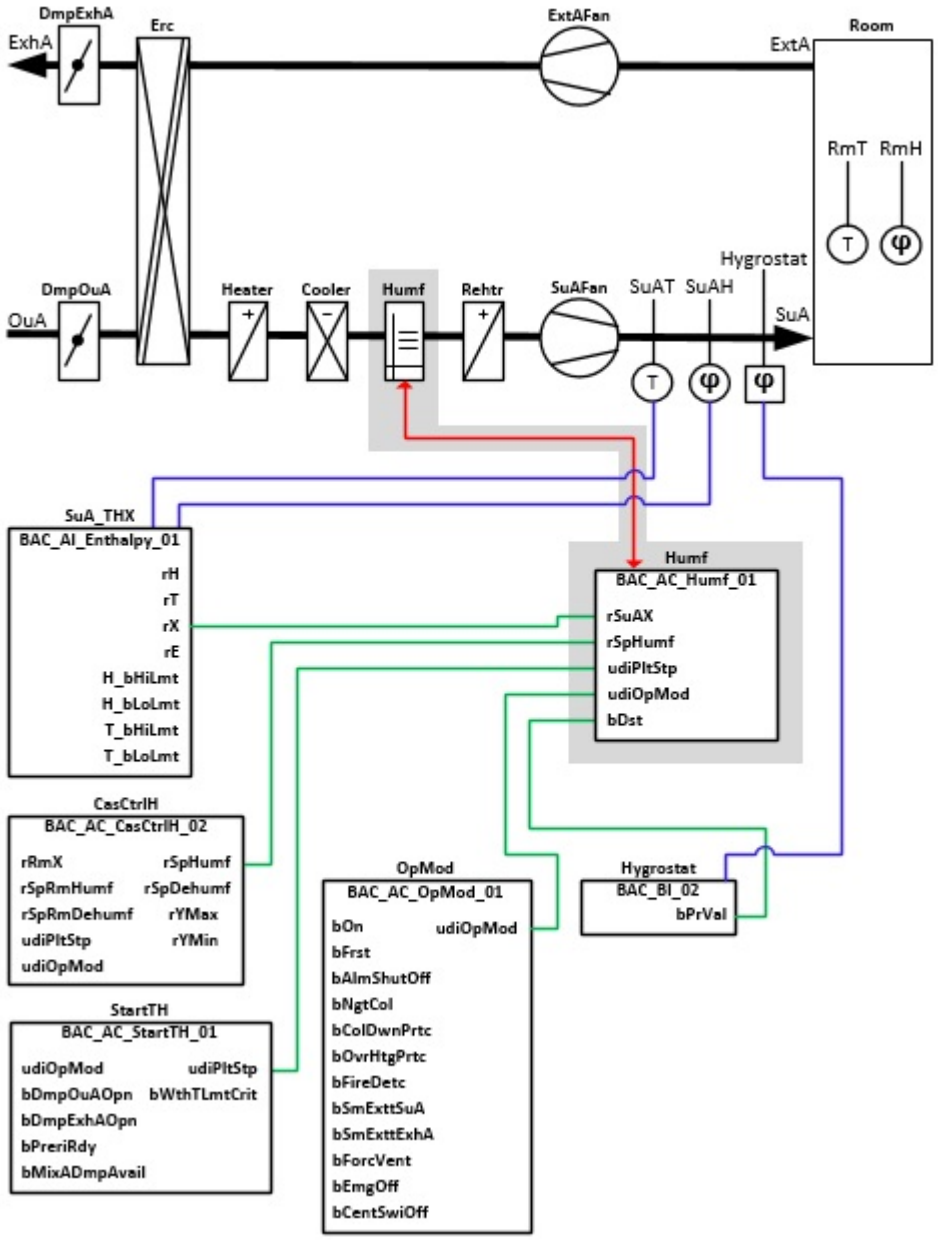
Within the call template the following sub templates are called and linked with each other:

- **HumfPID** Control of the absolute supply air humidity
- **Steam** Control of the steam generator

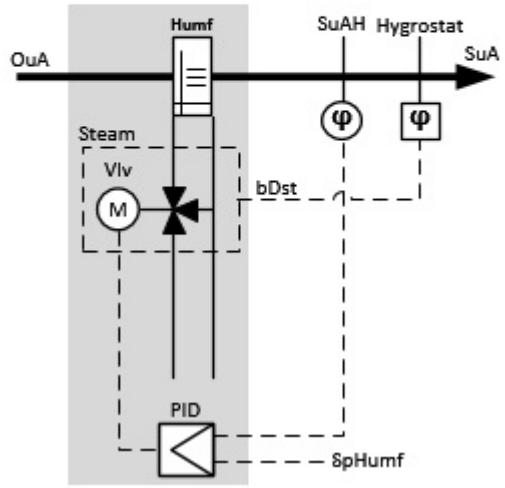
### Interface



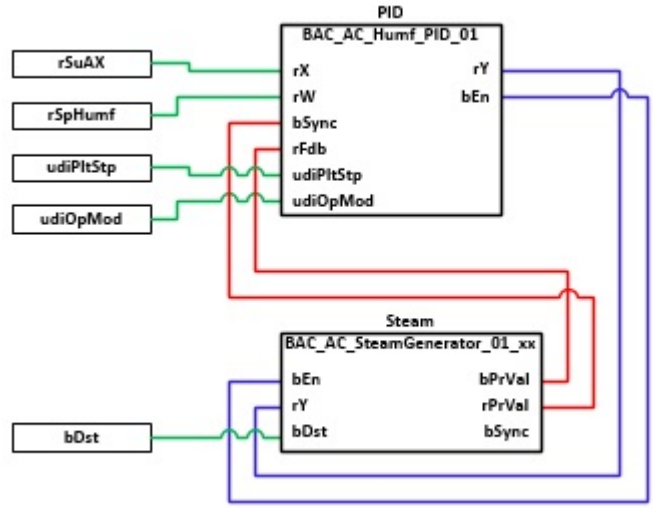
**Plant diagram 01**



### Plant diagram 02


### Block diagram




### VAR_INPUT

```
rSuAX       : REAL;
rSpHumf     : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
bDst        : BOOL;
```

**rSuAX:** calculated value of the absolute supply air humidity. This value is generated in template BAC_AI_Enthalpy_01 [▶ 678].

**rSpDehumf:** setpoint for the absolute supply air humidity

**udiPltStp:** plant start sequence steps. The plant steps are generated in the plant start program BAC_AC_StartTH_01 [▶ 535].

**udiOpMod:** plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

**bDst:** input for an external humidifier fault, e.g. a hygrostat

### Program description

| Instance | Type | Task |
|---|---|---|
| HumfPID | BAC_AC_Humf_PID_01 [▶ 376] | Subtemplate including PID sequence controller for absolute supply air humidity control. |
| HumfSteam | BAC_AC_SteamGenerator_01_03 [▶ 381] | Subtemplate for controlling a steam generator |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.13 BAC_AC_Humf_PID_01

### Functional description

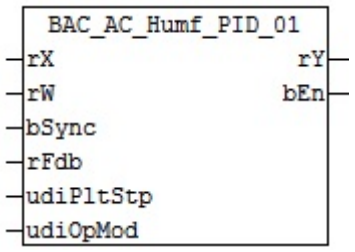
The subtemplate **BAC_AC_Humf_PID_01** is the sequence controller for a humidifier.
The set value, actual value and control output are referenced via the BACnet value objects **X**, **W** and **Y**.

The PID sequence controller is enabled based on the plant operation mode **udiOpMod** and the global humidity communication structure **g_stSeqLinkH[PLT_NUM]**.
This data and command structure is the link between the individual sequence controllers and the corresponding control function block FB_BA_SeqLink [▶ 168] of a plant.

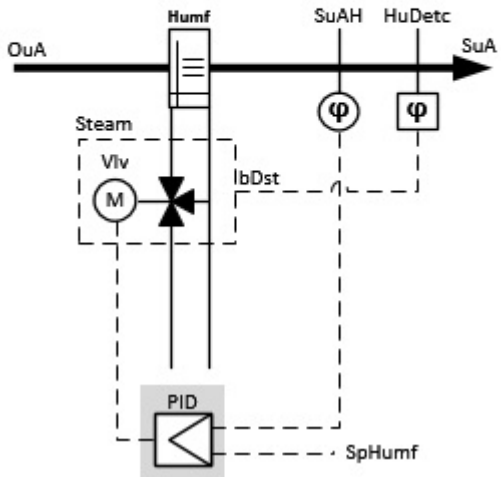
The BACnet BV object **En** is used to display the controller enable.

The limit value monitoring of PID controller is controlled by the function block **EnEvtEn** as a function of the plant startup process.

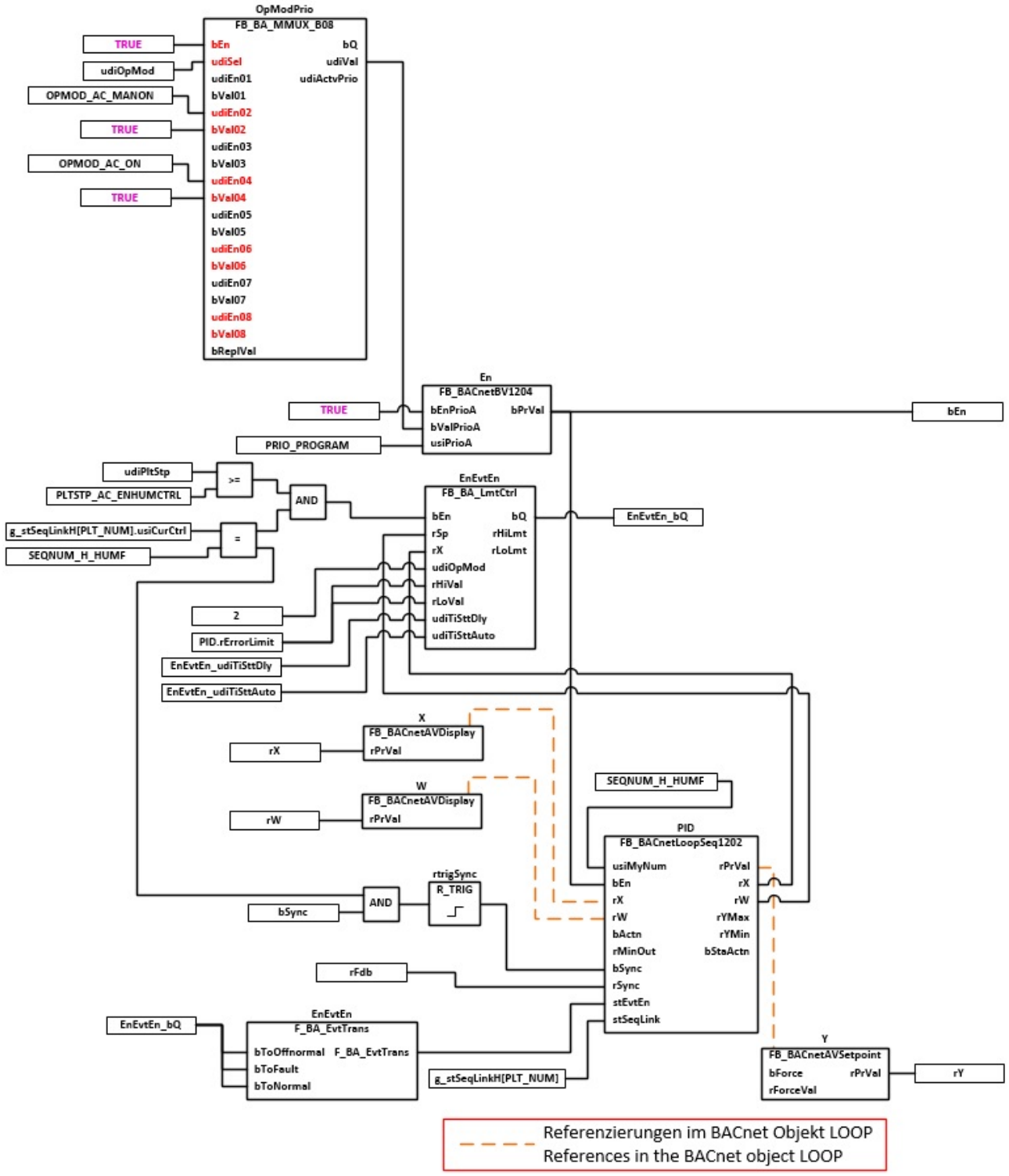
### Interface



### System diagram

### Block diagram




### VAR_INPUT

```
rX          : REAL;
rW          : REAL;
bSync       : BOOL;
rFdbVlv     : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rX:** Calculated value of the absolute supply air humidity

**rW:** Set value of the absolute supply air humidity

**bSync:** Input for synchronization of the controller

**rFdb:** Position feedback actuator

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartTH_01 [▶ 535].

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rY    : REAL;
```

**rY:** Control value output

**bEn:** Enable output of the PID sequence controller

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkH[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

<table>
<tr><th>Instance</th><th>Type</th><th colspan="4">Task</th></tr>
<tr><td>X</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the actual value input of the BACnet loop object</td></tr>
<tr><td>W</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the setpoint input of the BACnet loop object</td></tr>
<tr><td>OpModPrio</td><td>FB_BA_MMUX_B08 [▶ 205]</td><td colspan="4">The multiplexer defines the enable conditions of the sequence controller depending on the plant operation mode.</td></tr>
<tr><td rowspan="3"></td><td rowspan="3"></td><td colspan="2">udiOpMod</td><td>Enable</td><td>Comment</td></tr>
<tr><td>OPMOD_AC_MANON [▶ 357]</td><td>Manual on</td><td>TRUE</td><td>The plant is switched on manually via the plant selector switch</td></tr>
<tr><td>OPMOD_AC_ON [▶ 357]</td><td>On</td><td>TRUE</td><td>The plant runs in automatic mode via the timer program</td></tr>
<tr><td>En</td><td>FB_BACnetBV1204 [▶ 94]</td><td colspan="4">The BV object is used to display and activate the controller enable in the MCL or in a local operating display.<br>The PID sequence controller is enabled using the plant operation mode udiOpMod and the communication structure stSeqLink.</td></tr>
<tr><td>EnEvtEn</td><td>FB_BA_LmtCtrl [▶ 230]</td><td colspan="4">The BACnet loop object PID monitors the control function by comparing the setpoint W with the actual value X. If the deviation W-X is greater than the property ErrorLimit, then the loop object sends a message to the</td></tr>
</table>

| Instance | Type | Task |
|---|---|---|
| | | MCL.<br><br>When the plant is at a standstill, at the moment of startup and until the plant is in a controlled state, the loop object is suppressed so that no incorrect messages are sent to the MCL. Reporting of the loop object should not be activated until the air-conditioning system is fully running and the control is stable. In addition, reporting is enabled if the control has not reached the range around the setpoint defined by the property **ErrorLimit** after a long time.<br><br>Enabling the object internal reporting is done by writing to the BACnetEventTransitionBits of the loop object.<br><br>**The following conditions must be met to enable reporting from the loop object:**<br><br>**1.**<br>The plant start program BAC_AC_StartTH_01 [▶ 535] has enabled control monitoring and sensor limit monitoring<br>**udiPltStp** >= **PLTSTP_AC_ENHUMCTRL**<br>and<br>the dehumidifier controller is the active controller in the control sequence.<br>g_stSeqLinkH[PLT_NUM] [▶ 357]**.usiCurCtrl** = **SEQNUM_H_HUMF**<br>and<br>the absolute supply air humidity has approached the setpoint so far that it has settled in a range between **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit**.<br>and<br>the absolute supply air humidity must have remained within the range of **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit** for at least the duration of **EnEvtEn_udiTiSttDly**.<br><br>**2.**<br>The timer **EnEvtEn_udiTiSttAuto** has expired and the control has not reached its setpoint range. |
| | F_BA_EvtTrans | writes to the **BACnetEventTransitionBits** to_offnormal, to_fault and to_normal of the loop object. The input **bEnEvtEn** must be TRUE |
| PID | FB_BACnetLoopSeq1202 [▶ 102] | Sequence controller humidification. |
| rtrigSync | R_TRIG | A rising edge at input **bSync** triggers synchronization of the loop object to the value of **lrSync**.<br>If the control valve of the cooler was overridden by writing a higher priority to the corresponding AO object of the MCL or by activating the local priority operation, the current position of the control valve deviates from the output of the loop object. The variables **bSync** and **rFdb** can be used to restore synchronicity between the position of the control valve and the controller. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.14 BAC_AC_SteamGenerator_01_xx

### Functional description

The template **BAC_AC_SteamGenerator_01_xx** is used for controlling a steam generator with binary and analog inputs and outputs. It essentially consists of an AO object and a BO object for controlling the steam generator, an MV object for manual control, and the corresponding AV object for entering the control value. The template is complemented through optional BACnet objects, see table with version overview.
The variables, which are linked with the process image of the input and output level in the PLC, can be found under **IO linking**.

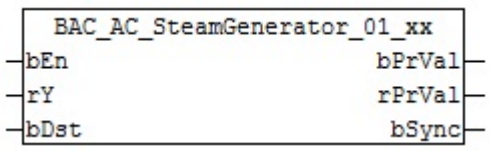
The two output variables rPrVal / bSync are only active, if the mechanical priority operation FdbOutAO exists in the template that is used. If this is not the case, the two variables return the value zero.

### Versions

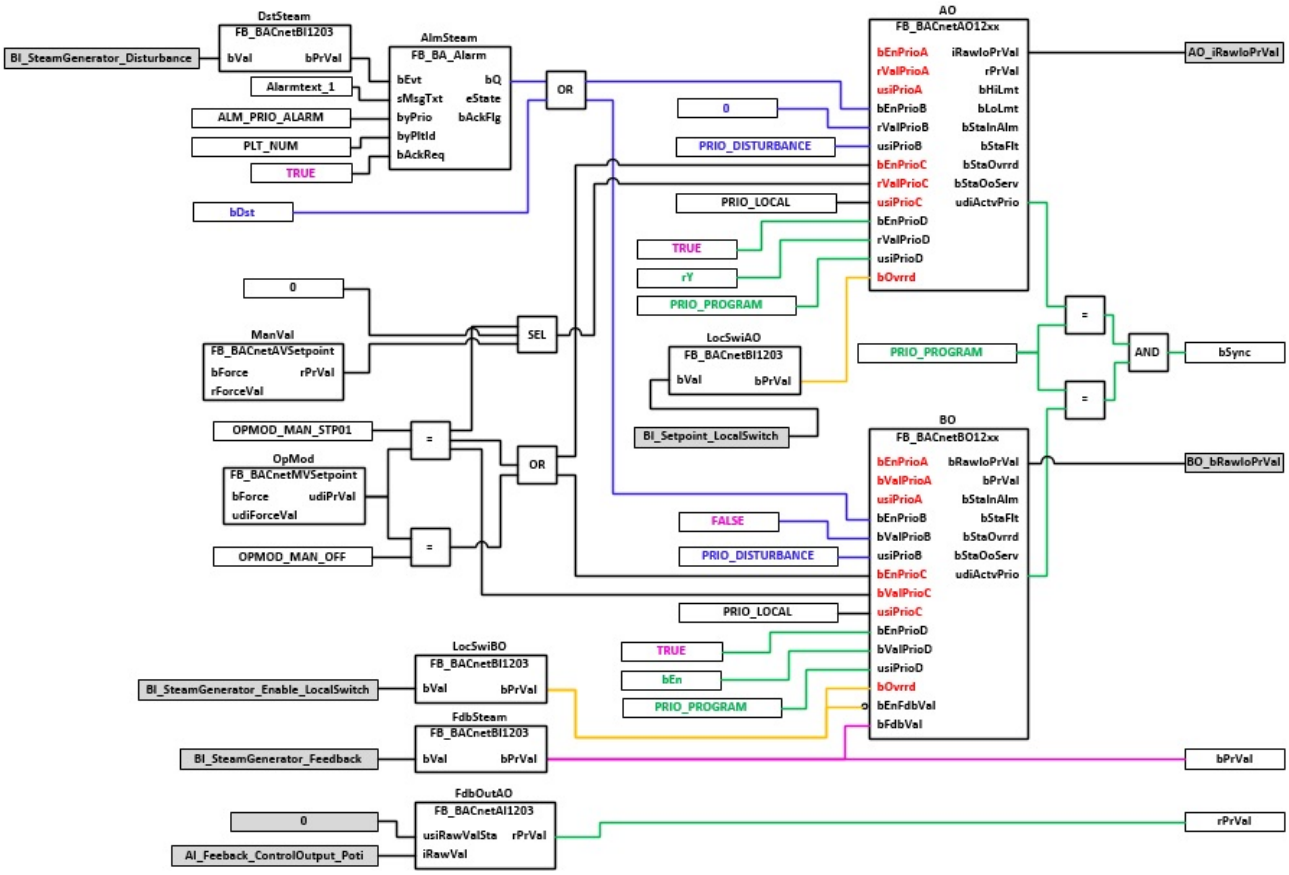
The template **BAC_AC_SteamGenerator_01_xx** is available in different versions.
The template versions are identified by means of a key. The identification key is derived from the table below.

| Options | Mechanical priority operation enable steam humidifier feedback hand switch (Enbl A-0-H) | Mechanical priority operation enable steam humidifier feedback relay (Rm-output) | Mechanical priority operation set value steam humidifier feedback hand switch (Rm-A-H) | Mechanical priority operation control value steam humidifier feedback position potentiometer (Rm-Poti) | Operating feedback Steam humidifier (Op-Steam-Gen) | Fault message steam humidifier (Fault-SteamGen) |
|---|---|---|---|---|---|---|
| **Instance** | **LocSwiBO** | **FdbOutBO** | **LocSwiAO** | **FdbOutAO** | **FdbSteam** | **DstSteam** |
| **Data point type** | **BI** | **BI** | **BI** | **AI** | **BI** | **BI** |
| | **32** | **16** | **8** | **4** | **2** | **1** |
| BAC_AC_SteamGenerator_01_03 | 0 | 0 | 0 | 0 | 1 | 1 |
| BAC_AC_SteamGenerator_01_47 | 1 | 0 | 1 | 1 | 1 | 1 |

### Interface



### Block diagram version BAC_AC_SteamGenerator_01_47




### VAR_INPUT

```
bEn     : BOOL;
rY      : REAL;
bDst    : BOOL;
```

**bEn:** Steam generator enable; program priority

**rY:** Control value of the steam generator

**bDst:** Input external fault steam generator.

### VAR_OUTPUT

```
bPrVal    : BOOL;
rPrVal    : REAL;
bSync     : BOOL;
```

**bPrVal:** operating feedback steam generator

**rPrVal :** current control value of the steam generator

**bSync:** output of a pulse to synchronize the controller associated with the steam generator to the current manipulated control value of the steam generator when resetting from manual to automatic mode.
The synchronization pulse **bSync** should only be used if the mechanical priority operation **FdbOutAO** is available in the template used.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]

The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

**Program description**

<table>
<tr><th>Instance</th><th>Type</th><th>op-tional</th><th>Task</th></tr>
<tr><td>DstSteam</td><td>FB_BACnetBI1203 [▶ 72]</td><td>X</td><td>BI object fault steam generator</td></tr>
<tr><td>LocSwiBO</td><td>FB_BACnetBI1203 [▶ 72]</td><td>X</td><td>BI object mechanical priority operation feedback hand switch binary</td></tr>
<tr><td>LocSwiAO</td><td>FB_BACnetBI1203 [▶ 72]</td><td>X</td><td>BI object mechanical priority operation feedback hand switch analog</td></tr>
<tr><td>FdbSteam</td><td>FB_BACnetBI1203 [▶ 72]</td><td>X</td><td>BI object operating feedback steam generator</td></tr>
<tr><td>FdbOutAO</td><td>FB_BACnetAI1203 [▶ 49]</td><td>X</td><td>Return value mechanical priority operation control value steam generator</td></tr>
<tr><td>OpMod</td><td>FB_BACnetMVSetpoint [▶ 130]</td><td></td><td>MV object for manual control of the steam generator via the MCL or a local operator display</td></tr>
<tr><td>ManVal</td><td>FB_BACnetAVSetpoint [▶ 69]</td><td></td><td>AV object for entering the control values of the steam generator with manual override</td></tr>
<tr><td>AlmSteam</td><td>FB_BA_Alarm [▶ 179]</td><td>x</td><td>Logging and further processing of a steam generator fault</td></tr>
<tr><td>AO</td><td>FB_BACnetAO1203 [▶ 53]</td><td></td><td>AO object for specifying the set speed value for the steam generator
<table>
<tr><th>Priority:</th><th>Enable</th><th>Value</th><th>Comment</th></tr>
<tr><td>PRIO_SAFETY (1)</td><td>FALSE</td><td>0</td><td></td></tr>
<tr><td>PRIO_DISTURBANCE (3)</td><td>Input bDst OR DstSteam</td><td>0</td><td></td></tr>
<tr><td>PRIO_LOCAL (8)</td><td>OpMod_udiPrVal = OPMOD_MAN_OFFOR OPMOD_MAN_OFF</td><td>Selector 0 ORManVal_rPrVal</td><td>In manual mode, value of AV object ManVal</td></tr>
<tr><td>PRIO_PROGRAM (15)</td><td>TRUE</td><td>rY</td><td>Value of input rY (e.g. control value of the controller)</td></tr>
</table></td></tr>
<tr><td>BO</td><td>FB_BACnetBO1203 [▶ 81]</td><td></td><td>BO object for specifying the frequency converter enable
<table>
<tr><th>Priority:</th><th>Enable</th><th>Value</th><th>Comment</th></tr>
<tr><td>PRIO_SAFETY (1)</td><td>FALSE</td><td>FALSE</td><td></td></tr>
<tr><td>PRIO_DISTURBANCE (3)</td><td>Input bDst OR DstSteam</td><td>FALSE</td><td></td></tr>
</table></td></tr>
</table>

| Instance | Type | op-tional | Task | | | |
|---|---|---|---|---|---|---|
| | | | PRIO_LOCAL (8) | **OpMod_udiPrVal** = **OPMOD_MAN_OFF**OR**OPMOD_MAN_STP01** | TRUE, if **OpMod_udiPrVal** = **OPMOD_MAN_STP01** | In manual mode, value of AV object ManVal |
| | | | PRIO_PROGRAM (15) | TRUE | **bEn** | Value of input **bEn** |
| | EQ, EQ, AND | | Value of the network is TRUE, if the active priority is **PRIO_PROGRAM** (15). Can be used for synchronizing the controller on return to automatic mode | | | |
| TLogAO | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the AO object for specifying the control value for the steam generator | | | |

### IO linking

In the XML description associated with the template, variables with the ID **Input** or **Output** are declared in the **Parameter** section. These parameters can be linked with the process image of the input and output level in the PLC in the Project Builder or via the Excel import interface.

| Parameter | Type | Optional | Process image | |
|---|---|---|---|---|
| BI_SteamGenerator_Disturbance | BOOL | | Input | Digital input - steam generator fault - message - triggered |
| BI_SteamGenerator_Enable_LocalSwitch | BOOL | x | Input | Digital input - switch manual enable steam generator - message - manual/auto |
| BI_Setpoint_LocalSwitch | BOOL | x | Input | Digital input - switch control value - message - manual/auto |
| BI_SteamGenerator_Feedback | BOOL | | Input | Digital input - steam generator - message - operation |
| AI_Feeback_ControlOutput_Poti | INT | x | Input | Analog input - manual potentiometer – feedback - control output |
| AO_SteamGenerator_ControlCommand | INT | | Output | Analog output - steam generator control command |
| BO_SteamGenerator_Enable | BOOL | | Output | Digital output - steam generator control command - enable on/off |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.15 BAC_AC_FireDmp_01_xx

### Functional description

The template **BAC_AC_FireDmp_01_xx** is used for controlling and monitoring of a motor-driven fire damper with binary switching output.

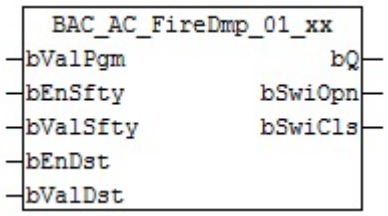
The two output variables bSwiOpn / bSwiCls only output the actual state of the damper, if the template used includes feedback from the limit switches SwiOpn/SwiCls . If no end position monitoring is available, it is emulated internally. If the damper is controlled, output bSwiOpn becomes TRUE. If the damper is not controlled, the output bSwiCls becomes TRUE. This may mean that delay time for opening the damper must be specified in the start program for a ventilation system.

### Versions

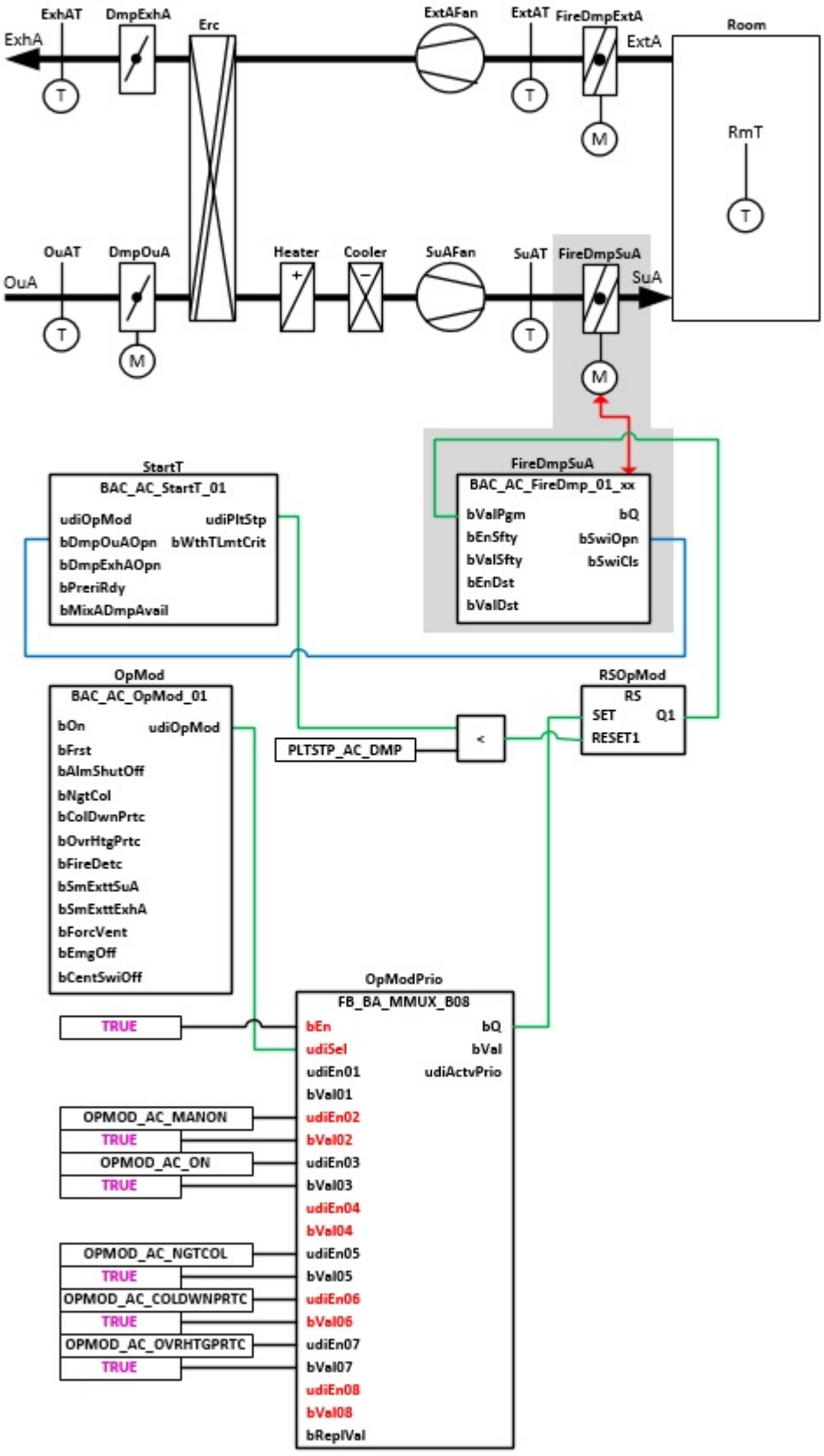
The template **BAC_AC_FireDmp_01_xx** is available in different versions.
The damper versions are identified by means of a key. The identification key is derived from the table below.

| Options | Thermal trig-gering mes-sage | mechanical priority opera-tion Feedback hand switch | mechanical priority opera-tion Feedback Relay output | Final position Open | Final position Close |
|---|---|---|---|---|---|
| **Instance** | **ThOvrld** | **LocSwi** | **FdbOut** | **SwiOpn** | **SwiCls** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** | **BI** |
| | **16** | **8** | **4** | **2** | **1** |
| BAC_AC_FireDmp_01_01 | 0 | 0 | 0 | 0 | 1 |
| BAC_AC_FireDmp_01_02 | 0 | 0 | 0 | 1 | 0 |
| BAC_AC_FireDmp_01_03 | 0 | 0 | 0 | 1 | 1 |
| BAC_AC_FireDmp_01_13 | 0 | 1 | 1 | 0 | 1 |
| BAC_AC_FireDmp_01_14 | 0 | 1 | 1 | 1 | 0 |
| BAC_AC_FireDmp_01_15 | 0 | 1 | 1 | 1 | 1 |
| BAC_AC_FireDmp_01_17 | 1 | 0 | 0 | 0 | 1 |
| BAC_AC_FireDmp_01_18 | 1 | 0 | 0 | 1 | 1 |
| BAC_AC_FireDmp_01_19 | 1 | 0 | 0 | 1 | 1 |
| BAC_AC_FireDmp_01_29 | 1 | 1 | 1 | 0 | 1 |
| BAC_AC_FireDmp_01_30 | 1 | 1 | 1 | 1 | 0 |
| BAC_AC_FireDmp_01_31 | 1 | 1 | 1 | 1 | 1 |

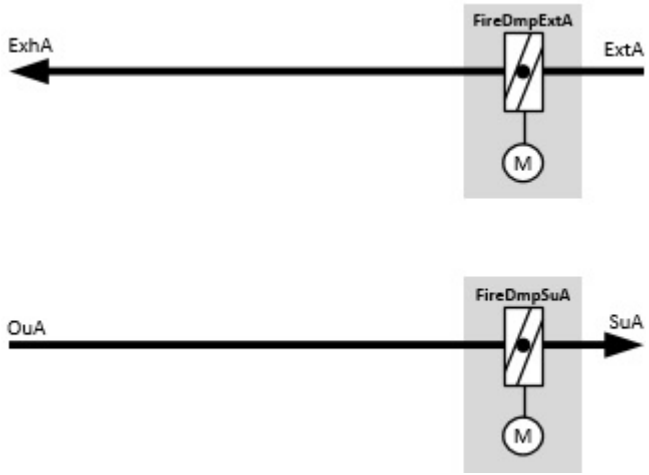
**Interface**


BAC_AC_FireDmp_01_xx
bValPgm
bEnSfty
bValSfty
bEnDst
bValDst
bQ
bSwiOpn
bSwiCls

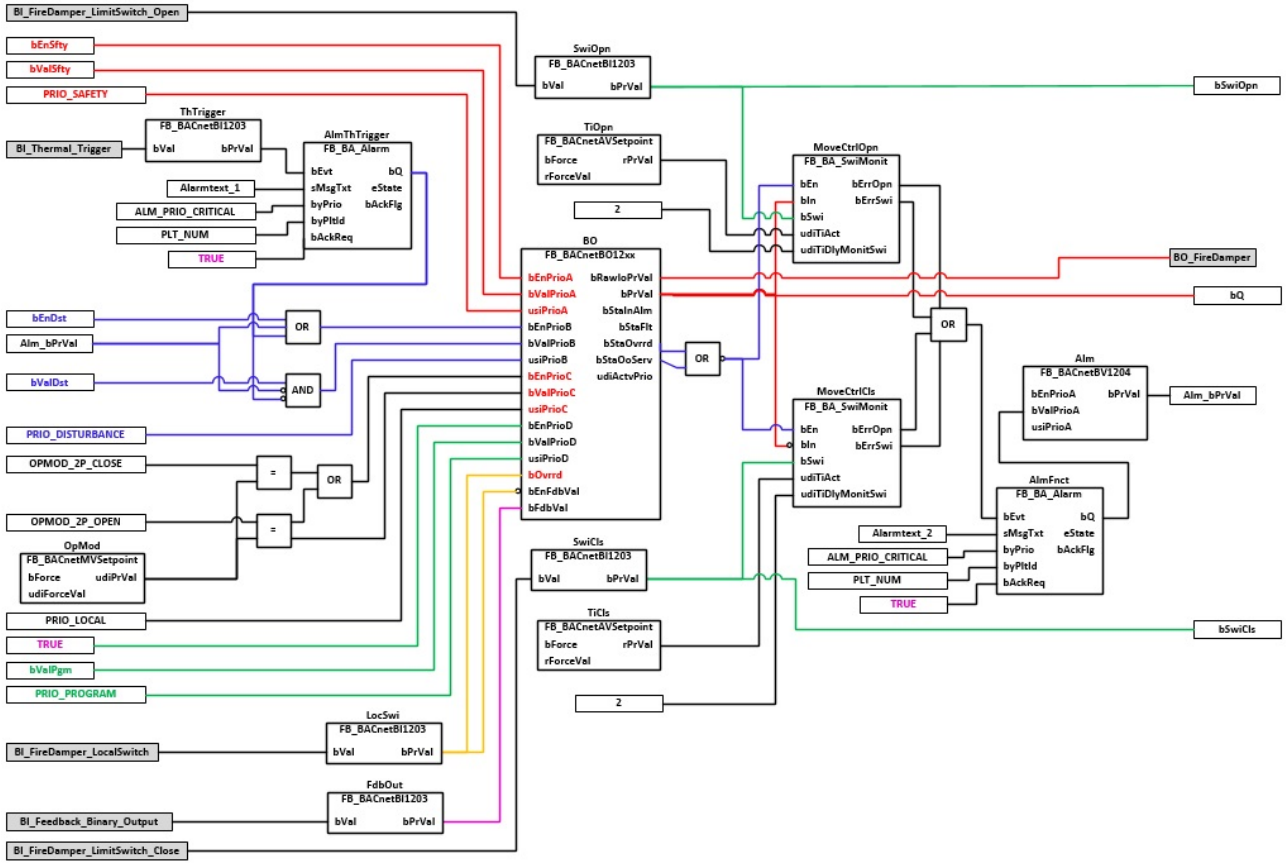
### Plant diagram 01



### Plant diagram 02




### Block diagram version BAC_AC_FireDmp_01_31




### VAR_INPUT

```
bValPgm    : BOOL;
bEnSfty    : BOOL;
bValSfty   : BOOL;
bEnDst     : BOOL;
bValDst    : BOOL;
```

**bValPgm:** Binary value program priority

**bEnfty:** Safety priority enable

**bValSfty:** Binary value safety priority

**bEnDst:** Enable fault priority

**bValDst:** Binary value fault priority

### VAR_OUTPUT

```
bQ        : BOOL;
bSwiOpn   : BOOL;
bSwiCls   : BOOL;
```

**bQ:** Damper control output status

**bSwiOpn:** End position 'open' of the damper has been reached. If no end position monitoring is available, it is emulated internally. If the damper is controlled, output **bSwiOpn** becomes TRUE. If the damper is not controlled, the output **bSwiCls** becomes TRUE.

**bSwiCls:** End position 'closed' of the damper has been reached. If no end position monitoring is available, it is emulated internally. If the damper is controlled, output **bSwiOpn** becomes TRUE. If the damper is not controlled, the output **bSwiCls** becomes TRUE.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 10;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| SwiOpn | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'open' |
| SwiCls | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'closed' |
| ThTrigger | FB_BACnetBI1203 [▶ 72] | X | BI object for the connection of the thermal triggering, e.g. of the fusible link |
| AlmThTrigger | FB_BA_Alarm [▶ 179] | x | The AlmThTrigger function block detects the thermal trigger event, e.g. the fusible link.<br>Actions that are to take place after the thermal trigger is received can be parameterized in the template at the AlmThTrigger function block. |
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the fire damper via the MCL or a local operator display |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object for feedback from mechanical priority operation. (manual/emergency operating level) |
| FdbOut | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation position feedback relay |
| TiOpn | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the opening time value |

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| TiCls | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the closing time value | | |
| MoveCtrlOpn | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'open' end position of the damper | | |
| MoveCtrlCls | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'closed' end position of the damper | | |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for the control of the fire damper | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** |
| | | | PRIO_DISTURBANCE (3) | The OR module pools events which enable writing to the disturbance priority of the downstream BO object.<br>Events:<br>1. Input template **bEnDst**<br>2. Limit switch fault from the function block **AlmFnct**<br>3. Thermal triggering of fusible link from the function block **AlmThTrigger** | The fire damper can be forcibly opened or closed at the input of the template **bEnDst**, e.g. in case of smoke extraction or fire alarm. The forced switch-on is, however, locked on the AND module in case of the thermal triggering of the fusible link or a limit switch fault. |
| | | | PRIO_LOCAL (8) | The OR module pools events which enable writing to the priority manual override (local) of the downstream BO object.<br>Events:<br>1. The MV object has the value **OPMOD_2P_CLOSE** (close fire damper)<br>2. The MV object has the value **OPMOD_2P_OPEN** (open fire damper) | TRUE, if **OpMod_udiPrVal** = **OPMOD_2P_OPEN** |
| | | | PRIO_PROGRAM (15) | TRUE | Input **bValPgm** |
| AlmFnct | FB_BA_Alarm [▶ 179] | x | The AlmFnct function block records the event of the limit position switch monitoring.<br>Actions that are to take place after the limit switch fault is received can be parameterized in the template on the AlmFnct function block. | | |
| Alm | FB_BACnetBV1204 [▶ 94] | x | BV object for displaying the damper fault in the MCL | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_Damper_LimitSwitch_Open | BOOL | X | Input | limit switch - message - open |
| BI_Damper_LimitSwitch_Close | BOOL | X | Input | limit switch - message - closed |
| BI_Damper_LocalSwitch | BOOL | X | Input | switch M-0-A - message - operation mode |
| BI_Feedback_Binary_Output | BOOL | X | Input | feedback Binary output |
| BI_Damper_Thermal_Trigger | BOOL | X | Input | message thermal triggering |
| BO_Damper | BOOL | | Output | switching command - open |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.16 BAC_AC_ErcPl_01

### Application

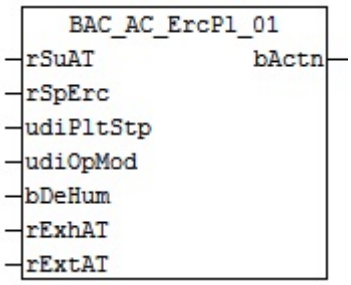
The template BAC_AC_ErcPl_01 is used for control and anti-icing of an energy recovery unit with plate heat exchanger.

Within the call template the following subtemplates are called and linked with each other:

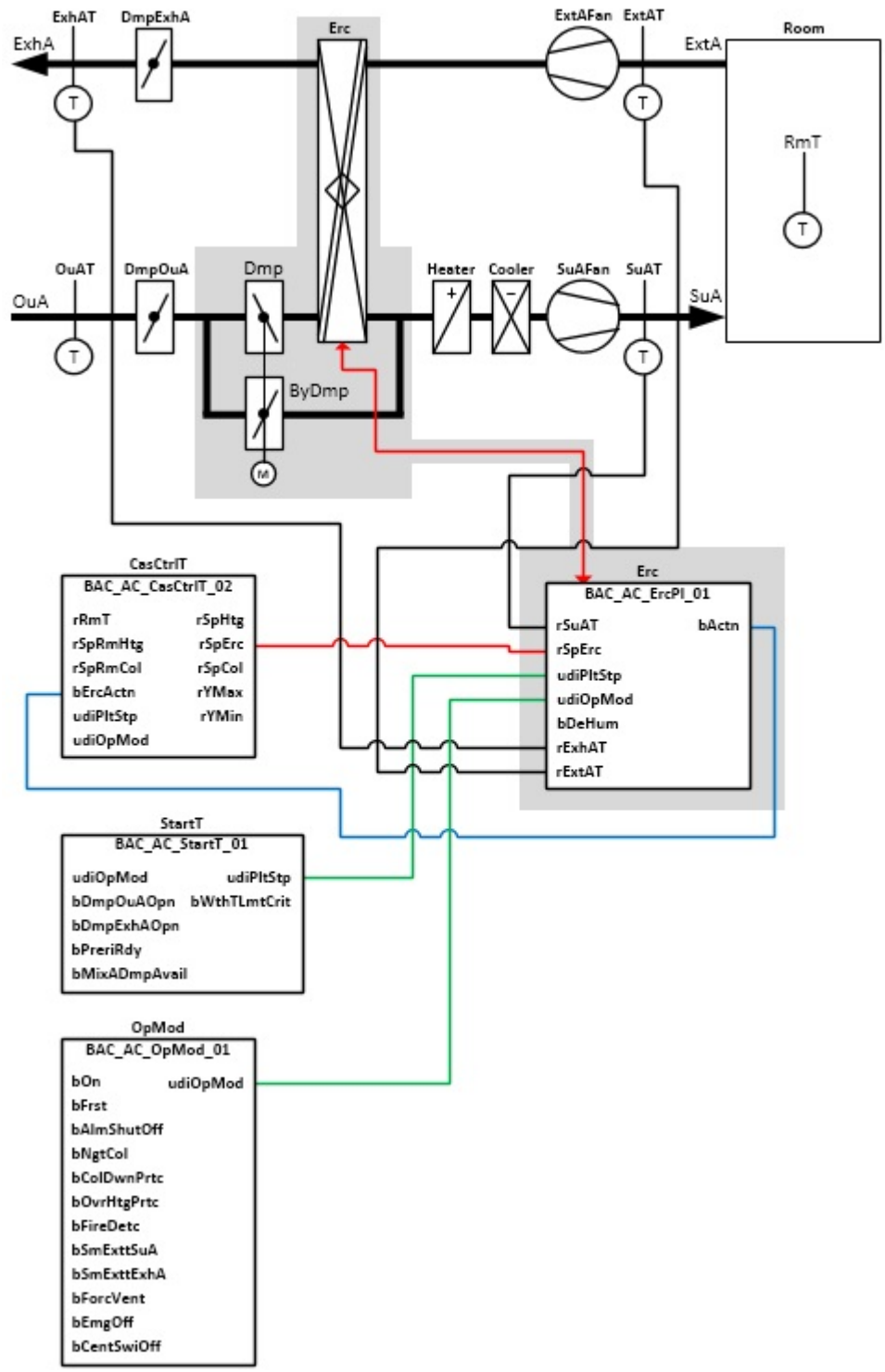
- **ErcSuA_PID:** supply air temperature control
- **ErcExhA_PID:** minimum limitation of exhaust air temperature
- **ErcIcPrt:** anti-icing of the heat exchanger by means of a differential pressure switch
- **ErcDmp:** control of a bypass damper system

The exhaust air minimum limiter **ErcExhA_PID** and the frost protection program **ErcIcPrt** limit the control value of the supply air temperature controller **ErcSuA_PID** of the energy recovery unit via the minimum value option, in order to prevent icing up of the heat exchanger.

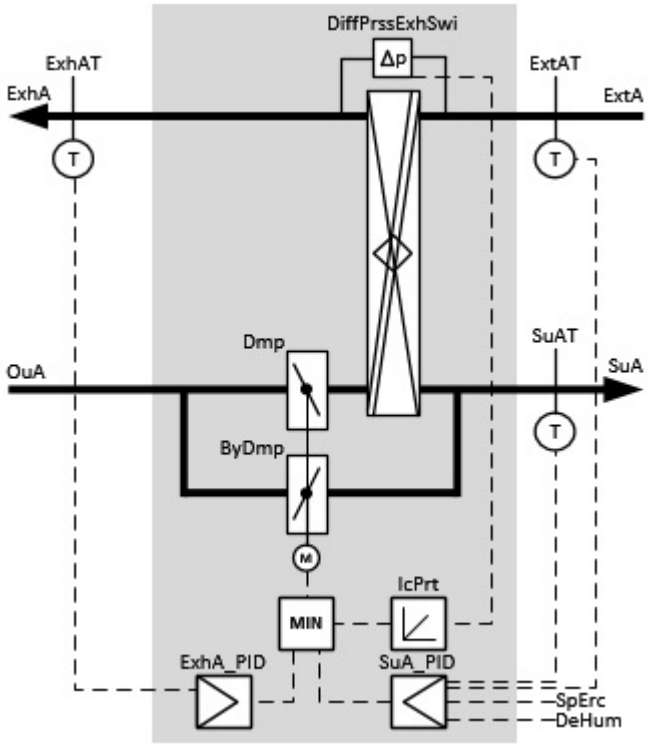
### Interface



**Plant diagram 01**

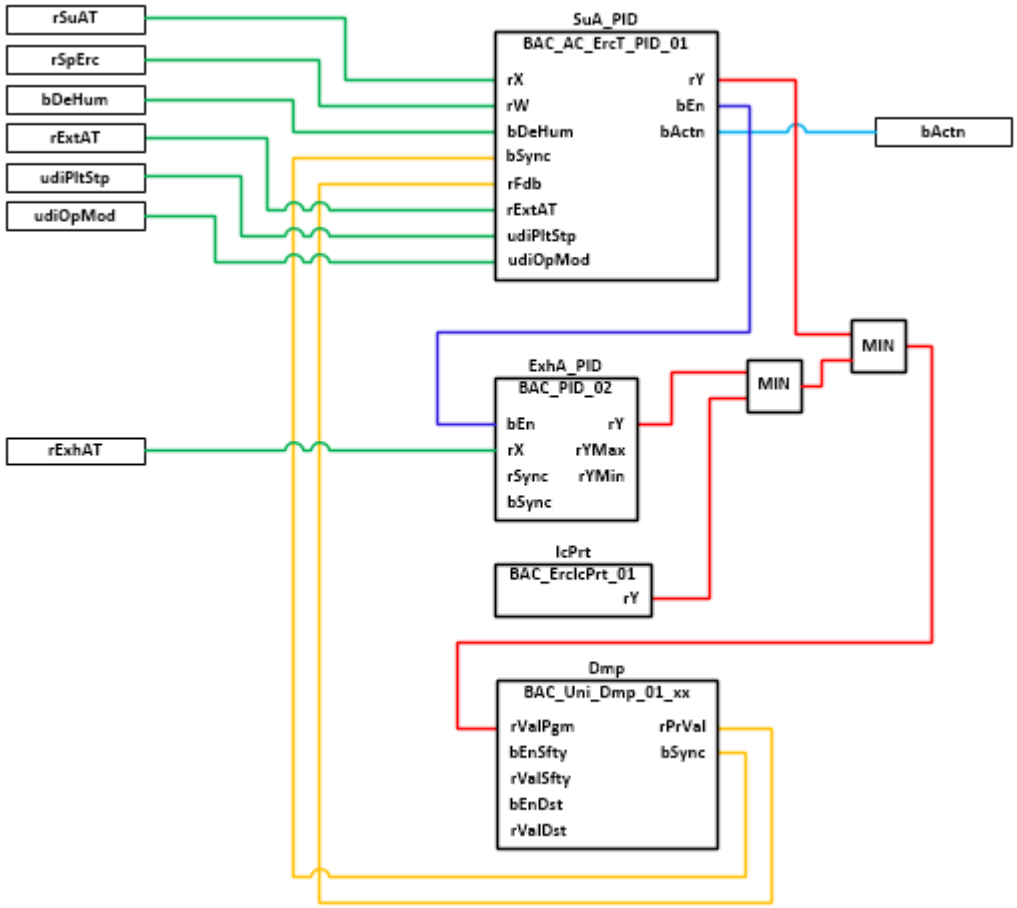


### Plant diagram 02




The output of the control signal for energy recovery output control refers to damper **Dmp**, not damper **ByDmp**. (see diagram). During commissioning of the air-conditioning plant, ensure that damper **Dmp** is operated with direct direction of action and damper **ByDmp** with inverted direction of action.

### Block diagram



### VAR_INPUT

```
rSuAT       : REAL;
rSpErc      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
bDeHum      : BOOL;
rExhAT      : REAL;
rExtAT      : REAL;
```

**rSuAT:** Measured value supply air temperature

**rSpErc:** Set value supply air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection program BAC_AC_OpMod_01 [▶ 515].

**bDeHum:** Status message dehumidification mode. Disabling of energy recovery in heating mode (outside temperature > outlet air temperature) .

**rExhAT:** Measured value exhaust air temperature

**rExtAT:** Measured value outlet air temperature

### VAR_OUTPUT

```
bActn       : BOOL;
```

**bActn:** control direction of the supply air temperature control **SuA_PID**. The control direction of the energy recovery is required by various setpoint programs or cascade controllers for determining setpoint for the supply air temperature of the energy recovery unit, e.g. BAC_AC_SpSuAT_02 [▶ 672], BAC_AC_CasCtrlT_02 [▶ 643].

### Program description

| Instance | Type | Task |
|---|---|---|
| SuA_PID | BAC_AC_ErcT_PID_01 [▶ 412] | Sub template supply air temperature control. The temperature control is analog via PID sequence controller |
| ExhA_PID | BAC_PID_02 [▶ 589] | Sub template exhaust air temperature control. The control is used for anti-icing and serves as prevention, before the ramp function **IcPrt** becomes active. |
| IcPrt | BAC_ErcIcPrt_01 [▶ 418] | Sub template anti-icing via differential pressure switch |
| Dmp | BAC_Uni_Dmp_01_05 [▶ 560] | Sub template for controlling an analog damper |
| | MIN, MIN | MIN selection of the control signals for supply air temperature control *SuA_PID*,exhaust air temperature control *ExhA_PID* and the frost protection ramp function |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.17 BAC_AC_ErcPl_02

### Application

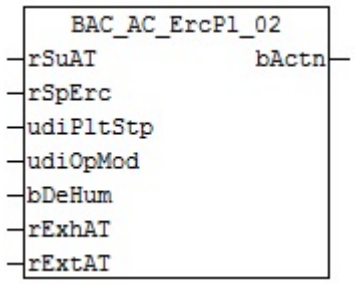
The template BAC_AC_ErcPl_02 is used for control and anti-icing of an energy recovery unit with plate heat exchanger.

Within the call template the following subtemplates are called and linked with each other:

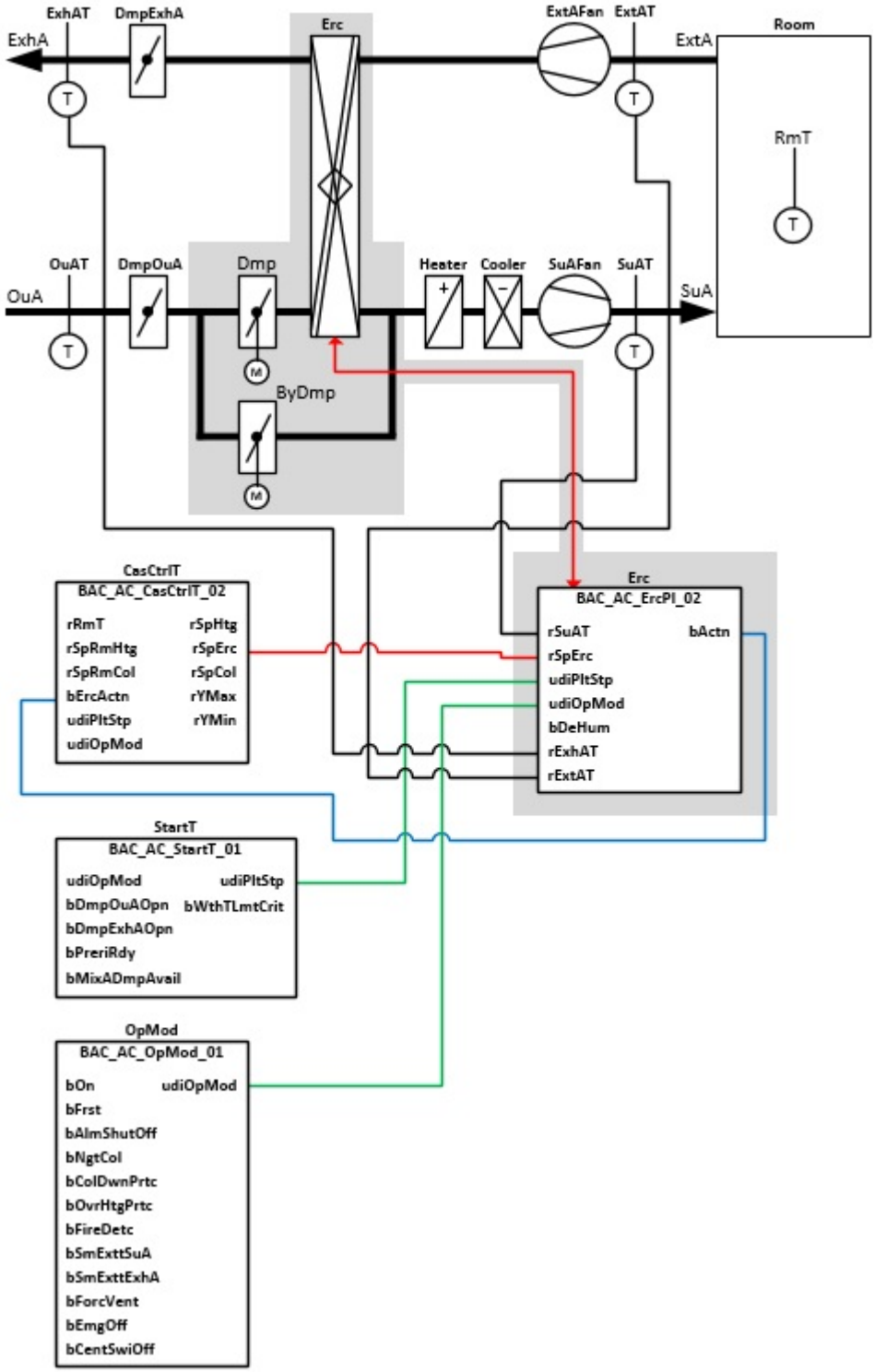
- **ErcSuA_PID:** supply air temperature control
- **ErcExhA_PID:** minimum limitation of exhaust air temperature
- **ErcIcPrt:** anti-icing of the heat exchanger by means of a differential pressure switch
- **ErcDmp:** control of a damper
- **ErcByDmp:** control of a bypass damper system

The exhaust air minimum limiter **ErcExhA_PID** and the frost protection program **ErcIcPrt** limit the control value of the supply air temperature controller **ErcSuA_PID** of the energy recovery unit via the minimum value option, in order to prevent icing up of the heat exchanger.

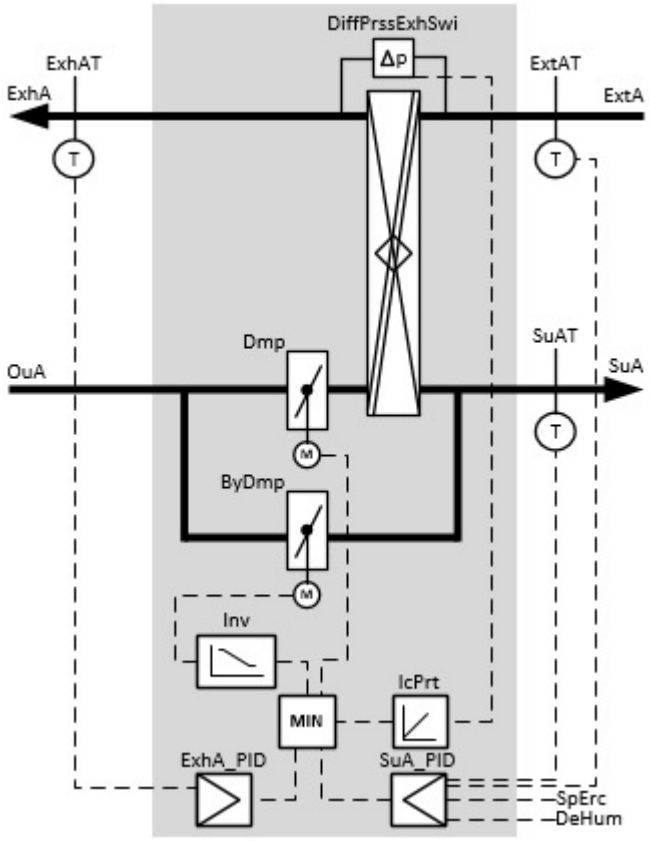
**Interface**



**Plant diagram 01**

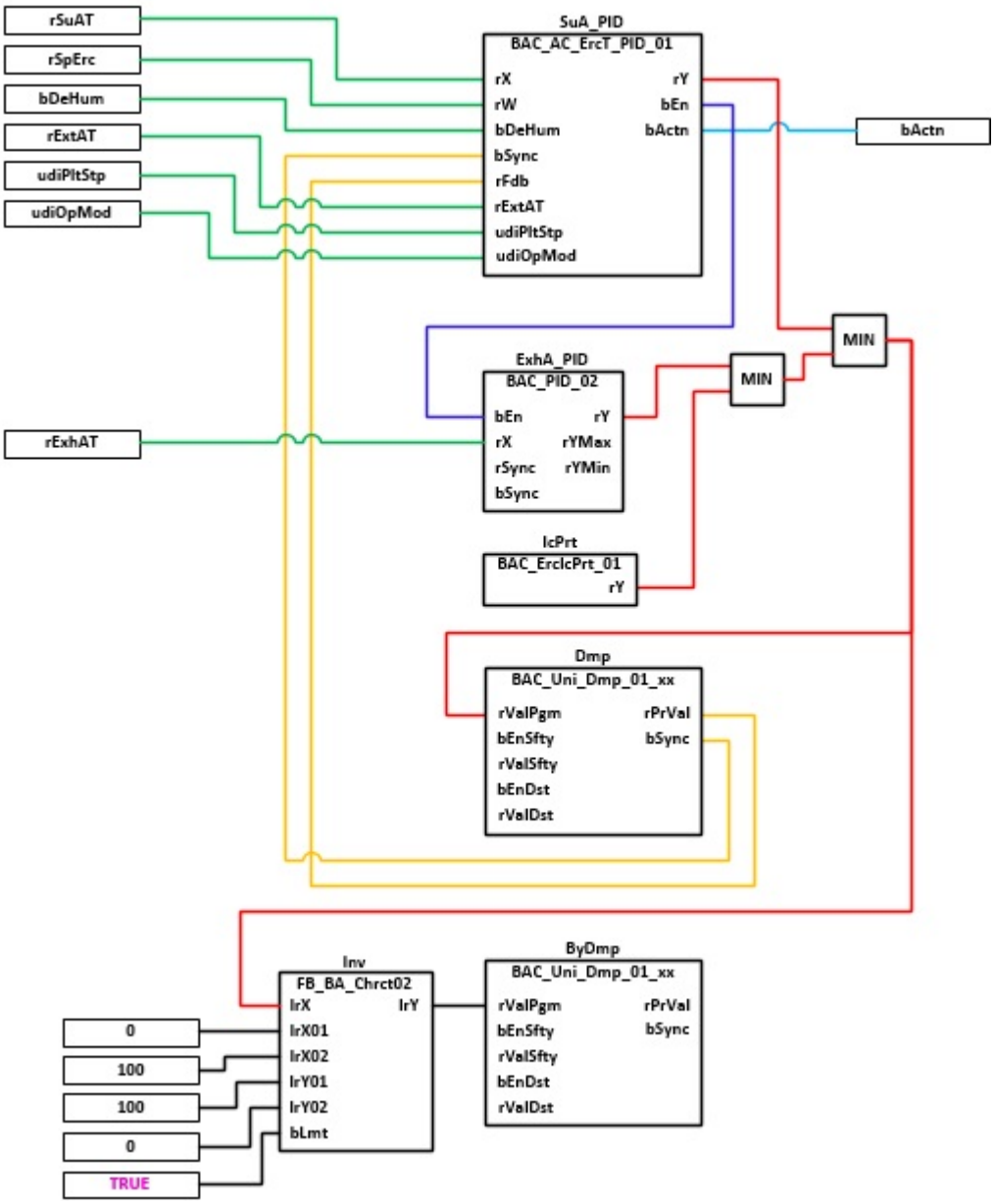


**Plant diagram 02**




The output of the control signal for controlling the performance of the energy recovery refers to the damper **Dmp** (see diagram). The control signal for the bypass damper **ByDmp** is inverted.
When commissioning the VAC system, care must be taken that both dampers are operated with a direct direction of action.

### Block diagram




### VAR_INPUT

```
rSuAT       : REAL;
rSpErc      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
bDeHum      : BOOL;
rExhAT      : REAL;
rExtAT      : REAL;
```

**rSuAT:** Measured value inlet air temperature

**rSpErc:** Set value inlet air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the plant start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection program BAC_AC_OpMod_01 [▶ 515].

**bDeHum:** Status message dehumidification mode. Disabling of energy recovery in heating mode (outside temperature > outlet air temperature) .

**rExhAT:** Measured value exhaust air temperature

**rExtAT:** Measured value outlet air temperature

### VAR_OUTPUT

```
bActn       : BOOL;
```

**bActn:** control direction of the supply air temperature control **SuA_PID**. The control direction of the energy recovery is required by various setpoint programs or cascade controllers for determining setpoint for the supply air temperature of the energy recovery unit, e.g. BAC_AC_SpSuAT_02 [▶ 672], BAC_AC_CasCtrlT_02 [▶ 643].

### Program description

| Instance | Type | Task |
|---|---|---|
| SuA_PID | BAC_AC_ErcT_PID_01 [▶ 412] | Sub template inlet air temperature control. The temperature control is analog via PID sequence controller |
| ExhA_PID | BAC_PID_02 [▶ 589] | Sub template exhaust air temperature control. The control is used for anti-icing and serves as prevention, before the ramp function **IcPrt** becomes active. |
| IcPrt | BAC_ErcIcPrt_01 [▶ 418] | Sub template anti-icing via differential pressure switch |
| Dmp | BAC_Uni_Dmp_01_05 [▶ 560] | Sub template for controlling an analog damper |
| Inv | FB_BA_Chrct02 [▶ 171] | Inversion of the control signal of the analog damper for controlling the bypass damper. |
| ByDmp | BAC_Uni_Dmp_01_05 [▶ 560] | Sub template for controlling an analog bypass damper |
| | MIN, MIN | MIN selection of the control signals for inlet air temperature control *SuA_PID,*exhaust air temperature control *ExhA_PID* and the frost protection ramp function |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.18 BAC_AC_ErcRecup_01

### Application

The template **BAC_AC_ErcRecup_01** is used for control and anti-icing of an energy recovery unit with a run-around-coil system.

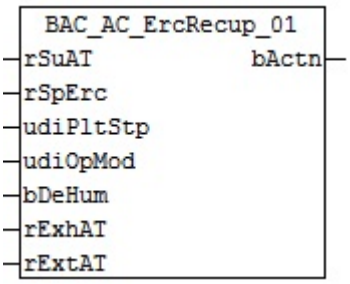
The template **BAC_AC_ErcRecup_01** is a call template.

Within the call template the following subtemplates are called and linked with each other:

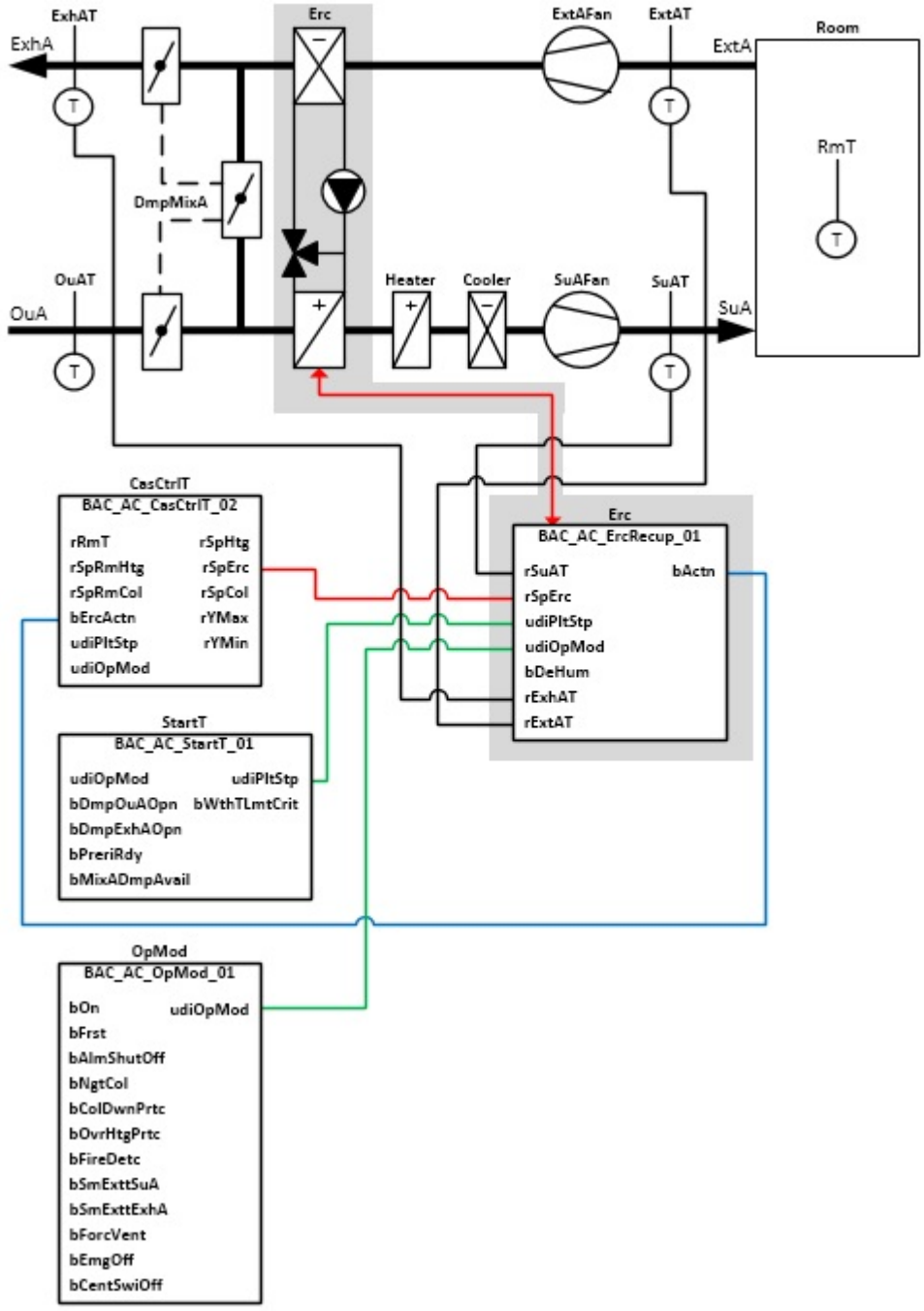
- **ErcRetWtrT:** return temperature of the run-around-coil system
- **ErcSuA_PID:** control of the supply air temperature
- **ErcExhA_PID:** minimum limitation of exhaust air temperature
- **ErcPu** control of the circulation pump
- **ErcRetWtr_PID:** minimum limitation of the water temperature at the heat exchanger return in the exhaust air of the run-around-coil system
- **ErcIcPrt_01:** anti-icing for the heat exchanger in the exhaust air of the run-around-coil system with differential pressure switch
- **ErcVlv:** control of an analog control valve in the water/glycol circuit

The return water temperature controller **ErcRetWtr_PID**, the exhaust air minimum limiter **ErcExhA_PID** and the frost protection program **ErcIcPrt** limit the control value of the supply air temperature controller **ErcSuA_PID** of the energy recovery unit via the minimum value option, in order to prevent icing up of the heat exchanger.

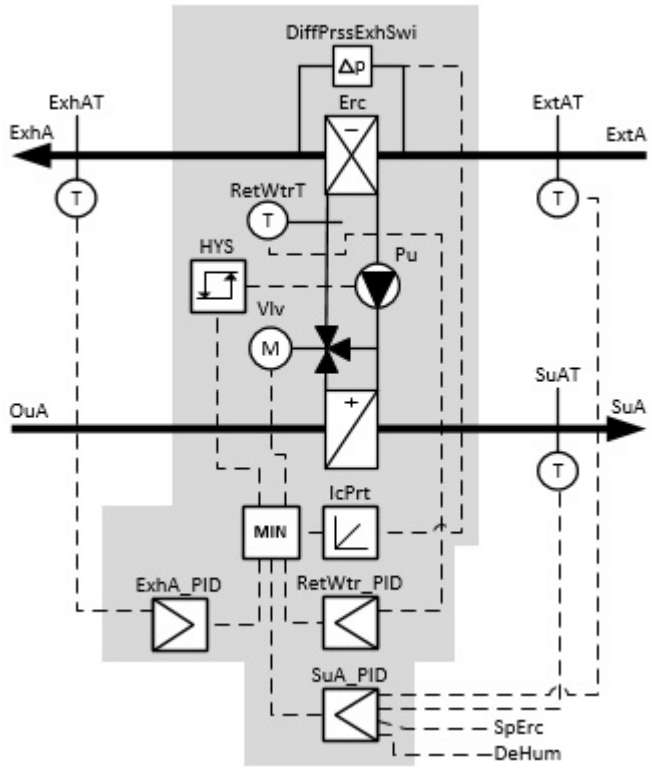
**Interface**



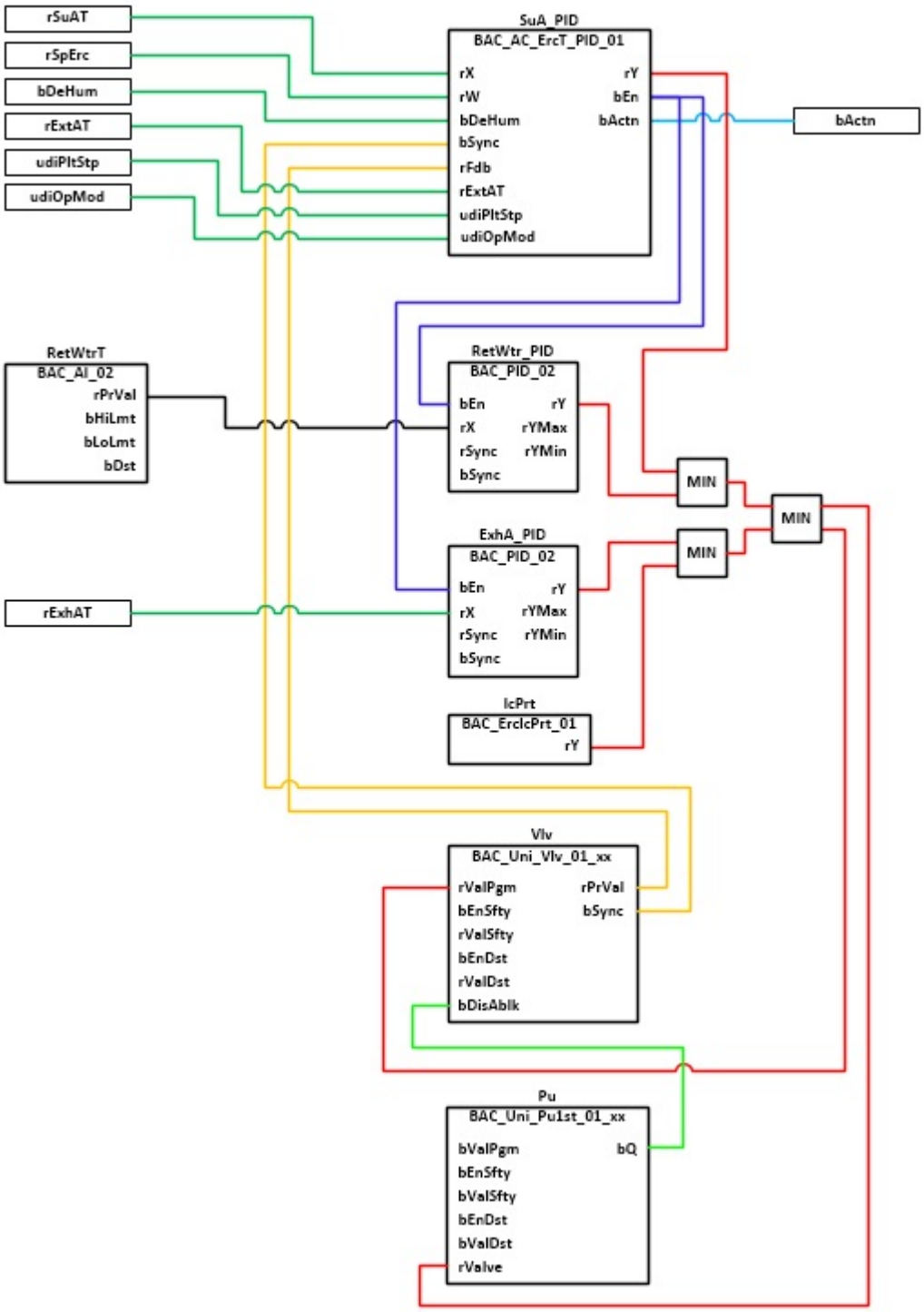
**Plant diagram 01**



**Plant diagram 02**



### Block diagram




### VAR_INPUT

```
rSuAT       : REAL;
rSpErc      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
bDeHum      : BOOL;
rExhAT      : REAL;
rExtAT      : REAL;
```

**rSuAT:** Measured value supply air temperature

**rSpErc:** Set value supply air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection program BAC_AC_OpMod_01 [▶ 515].

**bDeHum:** Status message dehumidification mode. Disabling of energy recovery in heating mode (outside temperature > outlet air temperature) .

**rExhAT:** Measured value exhaust air temperature

**rExtAT:** Measured value outlet air temperature

### VAR_OUTPUT

```
bActn       : BOOL;
```

**bActn:** control direction of the supply air temperature control **SuA_PID**. The control direction of the energy recovery is required by various setpoint programs or cascade controllers for determining setpoint for the supply air temperature of the energy recovery unit, e.g. BAC_AC_SpSuAT_02 [▶ 672], BAC_AC_CasCtrlT_02 [▶ 643].

### Program description

| Instance | Type | Task |
|---|---|---|
| RetWtrT | BAC_AI_02 [▶ 676] | Subtemplate return temperature of the run-around-coil system |
| SuA_PID | BAC_AC_ErcT_PID_01 [▶ 412] | Subtemplate supply air temperature control. The temperature control is analog via PID sequence controller |
| ExhA_PID | BAC_PID_02 [▶ 589] | Subtemplate exhaust air temperature control for anti-icing |
| RetWtrT_PID | BAC_PID_02 [▶ 589] | Subtemplate return temperature control for anti-icing |
| IcPrt | BAC_ErcIcPrt_01 [▶ 418] | Subtemplate for anti-icing of an energy recovery unit via differential pressure switch |
| Vlv | BAC_Uni_Vlv_01_13 [▶ 598] | Template for controlling an analog valve |
| Pu | BAC_Uni_Pu1st_01_189 [▶ 571] | Template for controlling a single-stage pump |
| | MIN, MIN, MIN | MIN selection of the control signals for supply air temperature control **SuA_PID**, exhaust air temperature control **ExhA_PID**, return temperature *control* **RetWtrT_PID** and frost protection ramp function |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.19 BAC_AC_ErcRot_01

### Application

The template **BAC_AC_ErcRot_01** is used for control and anti-icing of an energy recovery unit with rotary heat exchanger.

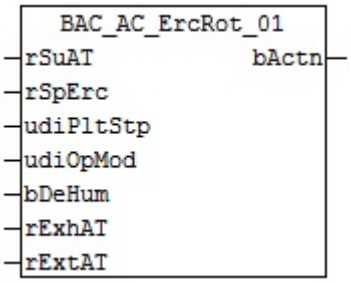
The template **BAC_AC_ErcRot_01** is a call template.
Within the call template, the following subtemplates are called and linked together:

- **ErcSuA_PID:** control of the supply air temperature
- **ErcExhA_PID:** minimum limitation of exhaust air temperature
- **ErcMot:** control of WRG control unit
- **ErcIcPrt:** anti-icing of the heat exchanger by means of a differential pressure switch
- **ErcByDmp:** control of a bypass damper system

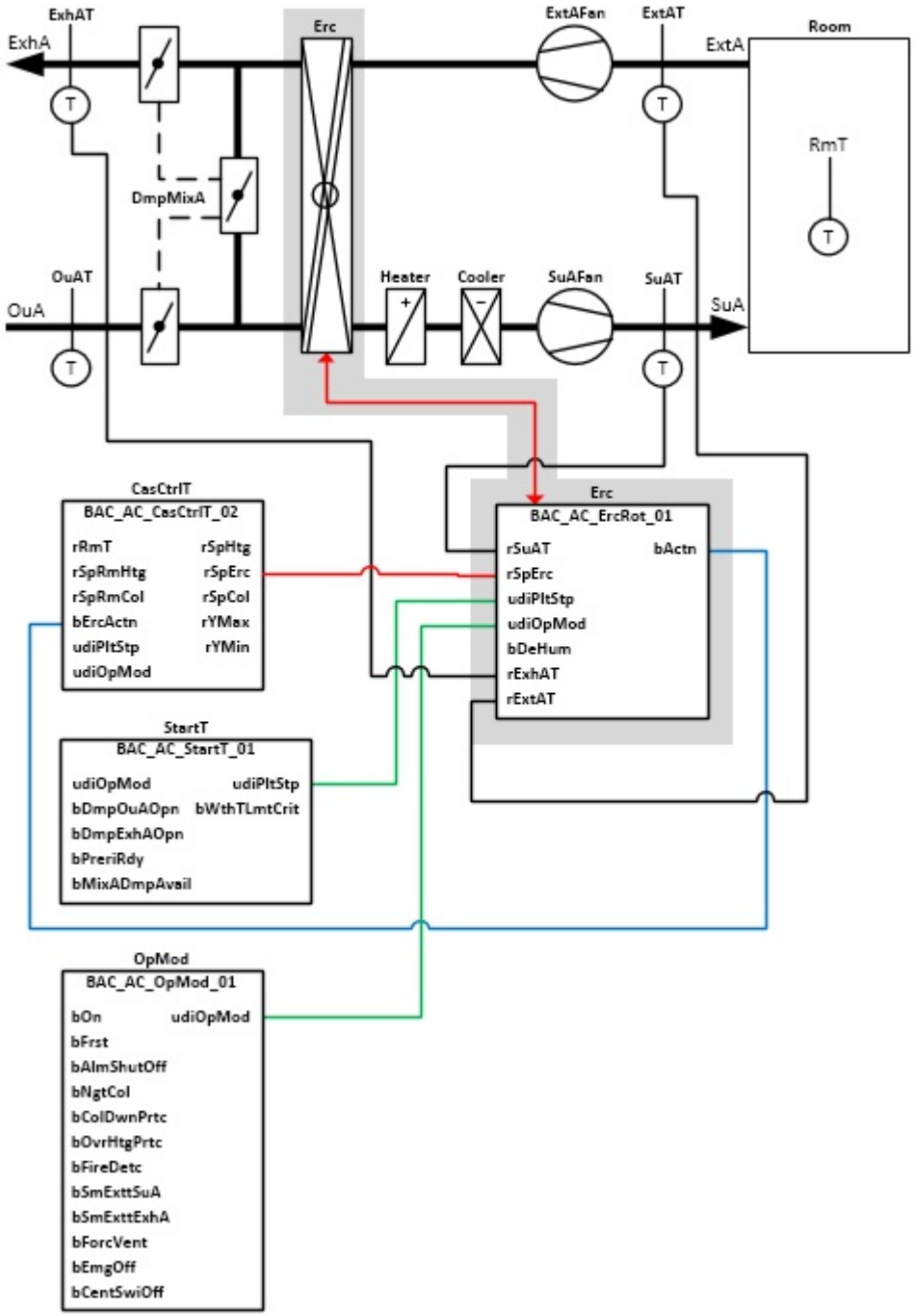
The exhaust air minimum limiter **ErcExhA_PID** and the frost protection program **ErcIcPrt** limit the control value of the supply air temperature controller **ErcSuA_PID** of the energy recovery unit via the minimum value option, in order to prevent icing up of the heat exchanger.

The supply air controller is enabled in the subtemplate BAC_AC_ErcT_PID_01 [▶ 412], depending on the plant operation mode. In operation modes such as summer night cooling, for example, the supply air controller is disabled. The hysteresis module **HysByDmp** is used to open the bypass damper when energy recovery is disabled. In this way, unnecessarily high flow resistances are avoided, and the energy consumption of the fans is reduced.

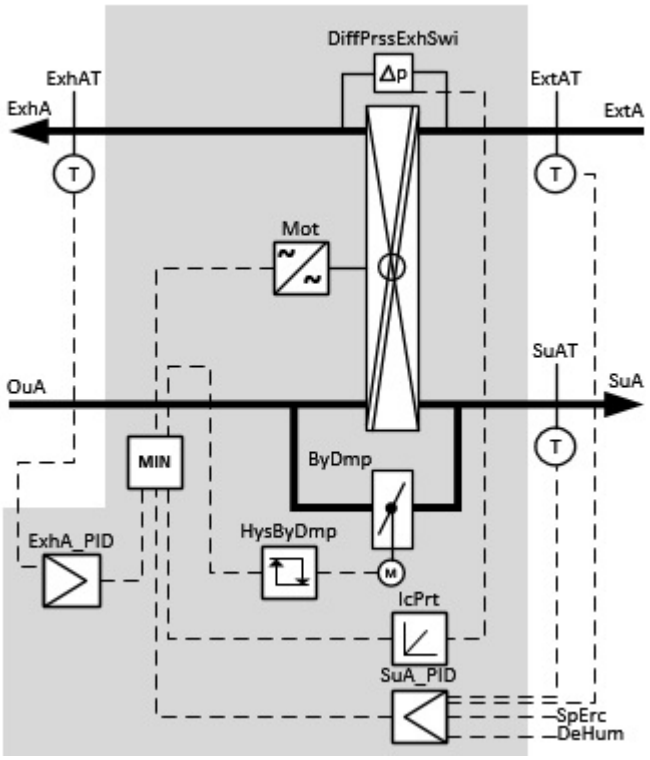
**Interface**



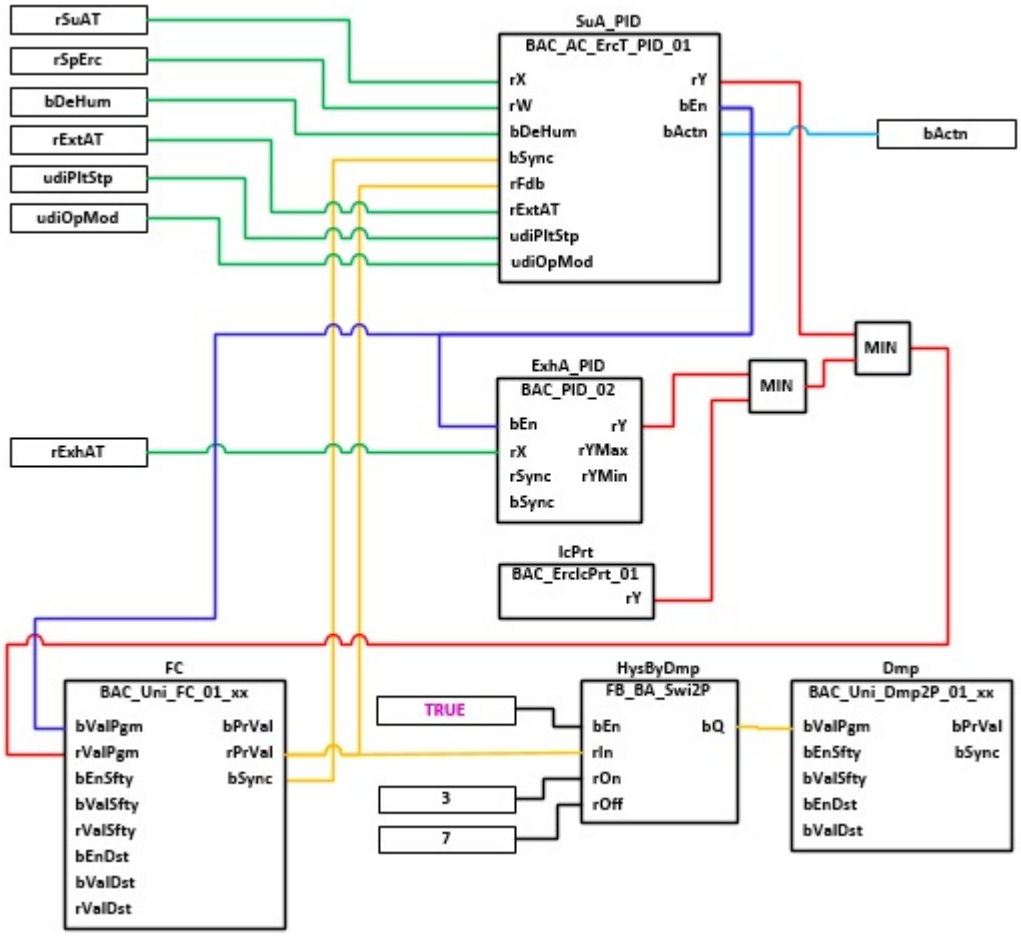
**Plant diagram 01**



### Plant diagram 02




### Block diagram



### VAR_INPUT

```
rSuAT       : REAL;
rSpErc      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
bDeHum      : BOOL;
rExhAT      : REAL;
rExtAT      : REAL;
```

**rSuAT:** Measured value supply air temperature

**rSpErc:** Set value supply air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection program BAC_AC_OpMod_01 [▶ 515].

**bDeHum:** Status message dehumidification mode. If dehumidification is active, energy recovery is disabled in heating mode. If the heat recovery unit recovers 'coolth' from the outlet air, it is switched to 100% in dehumidification mode.

**rExhAT:** Measured value exhaust air temperature

**rExtAT:** Measured value outlet air temperature

### VAR_OUTPUT

```
bActn       : BOOL;
```

**bActn:** control direction of the supply air temperature control **SuA_PID**. The control direction of the energy recovery is required by various setpoint programs or cascade controllers for determining setpoint for the supply air temperature of the energy recovery unit, e.g. BAC_AC_SpSuAT_02 [▶ 672], BAC_AC_CasCtrlT_02 [▶ 643].

### Program description

| Instance | Type | Task |
|---|---|---|
| SuA_PID | BAC_AC_ErcT_PID_01 [▶ 412] | Sub template supply air temperature control. The temperature control is analog via PID sequence controller |
| ExhA_PID | BAC_PID_02 [▶ 589] | Sub template exhaust air temperature control for anti-icing |
| IcPrt | BAC_ErcIcPrt_01 [▶ 418] | Sub template for anti-icing of an energy recovery unit via differential pressure switch |
| Mot | BAC_Uni_FC_01_223 [▶ 555] | Sub template for controlling a frequency converter, which then controls the motor of the rotary heat exchanger. |
| HysByDmp | FB_BA_Swi2P [▶ 146] | The function block **HysByDmp** is a two-point switch, which opens or closes the bypass damper as a function of a hysteresis. The control value for the motor of the rotary heat exchanger serves as input variable for the two-point switch. |
| ByDmp | BAC_Uni_Dmp2P_01_08 [▶ 563] | Sub template for controlling the bypass damper. |
| | MIN, MIN | MIN selection of the control signals for supply air temperature control **SuA_PID**,exhaust air temperature control **ExhA_PID** and the frost protection ramp function |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.20 BAC_AC_ErcRot_02

### Application

The template **BAC_AC_ErcRot_0**2 is used for control and anti-icing of an energy recovery unit with rotary heat exchanger.

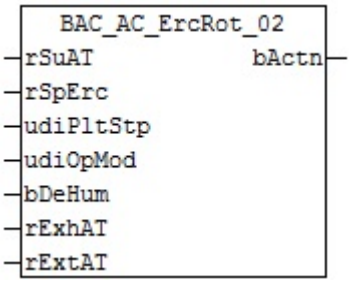
The template **BAC_AC_ErcRot_02** is a call template.
Within the call template, the following subtemplates are called and linked together:

- **ErcSuA_PID:** control of the supply air temperature
- **ErcExhA_PID:** minimum limitation of exhaust air temperature
- **ErcMot:** control of WRG control unit
- **ErcIcPrt:** anti-icing of the heat exchanger by means of a differential pressure switch

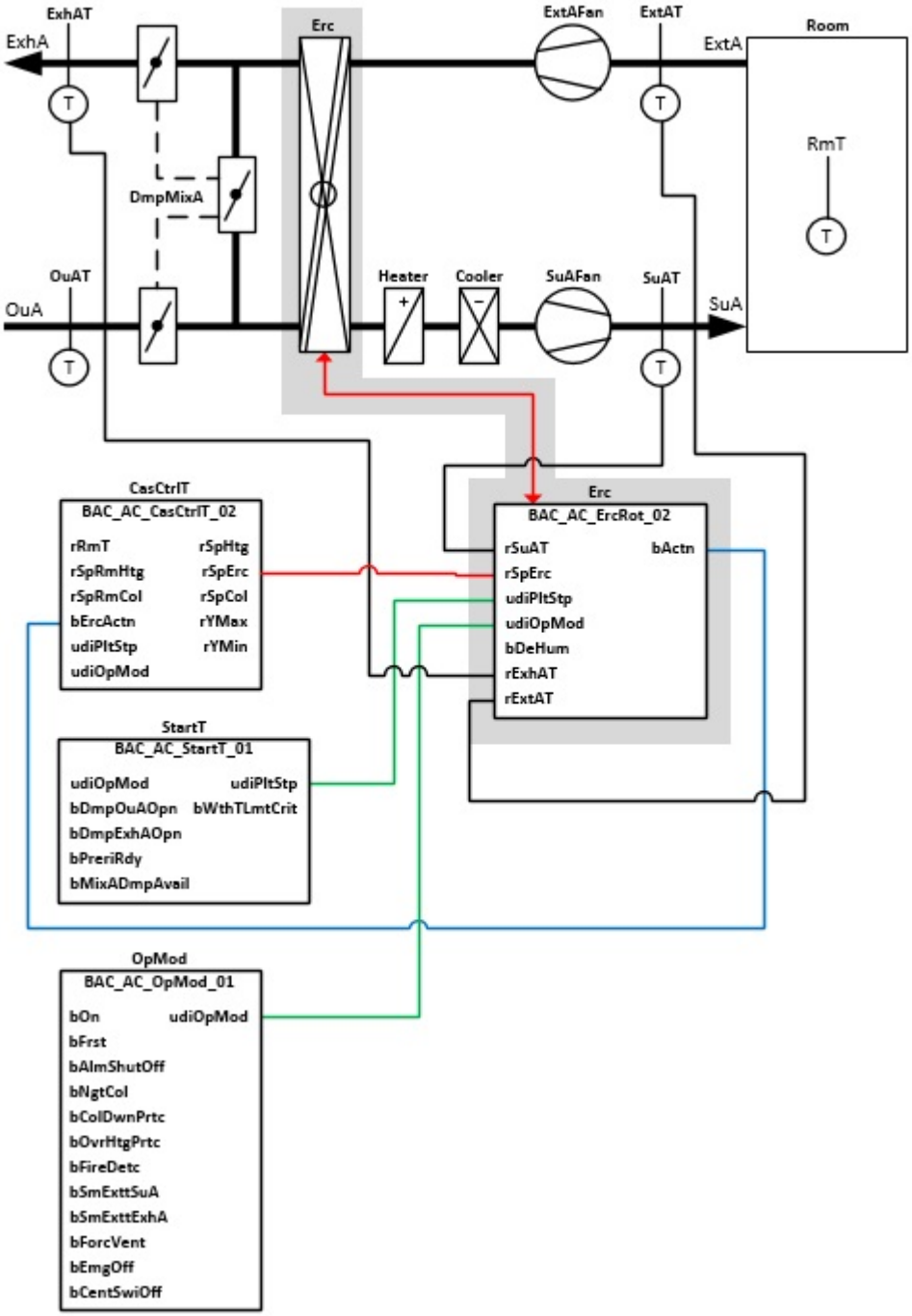
The exhaust air minimum limiter **ErcExhA_PID** and the frost protection program **ErcIcPrt** limit the control value of the supply air temperature controller **ErcSuA_PID** of the energy recovery unit via the minimum value option, in order to prevent icing up of the heat exchanger.

The supply air controller is enabled in the subtemplate BAC_AC_ErcT_PID_01 [▶ 412], depending on the plant operation mode. In operation modes such as summer night cooling, for example, the supply air controller is disabled.

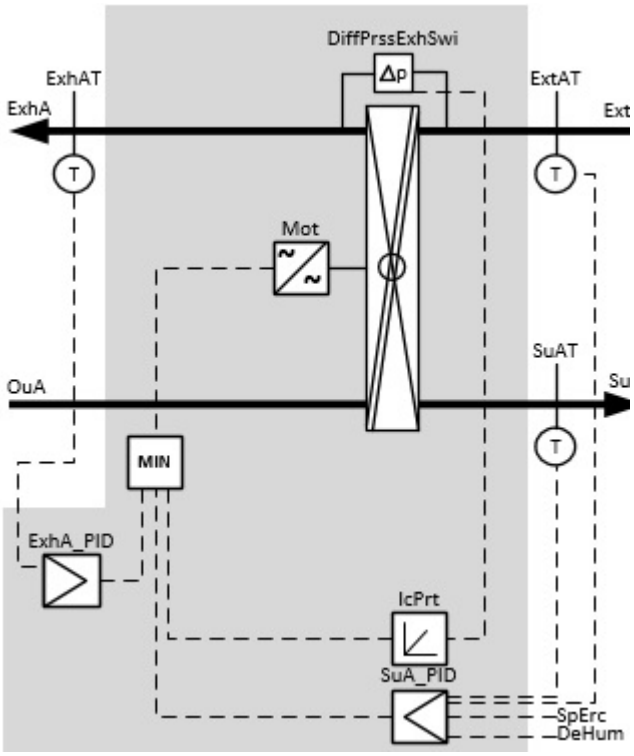
### Interface



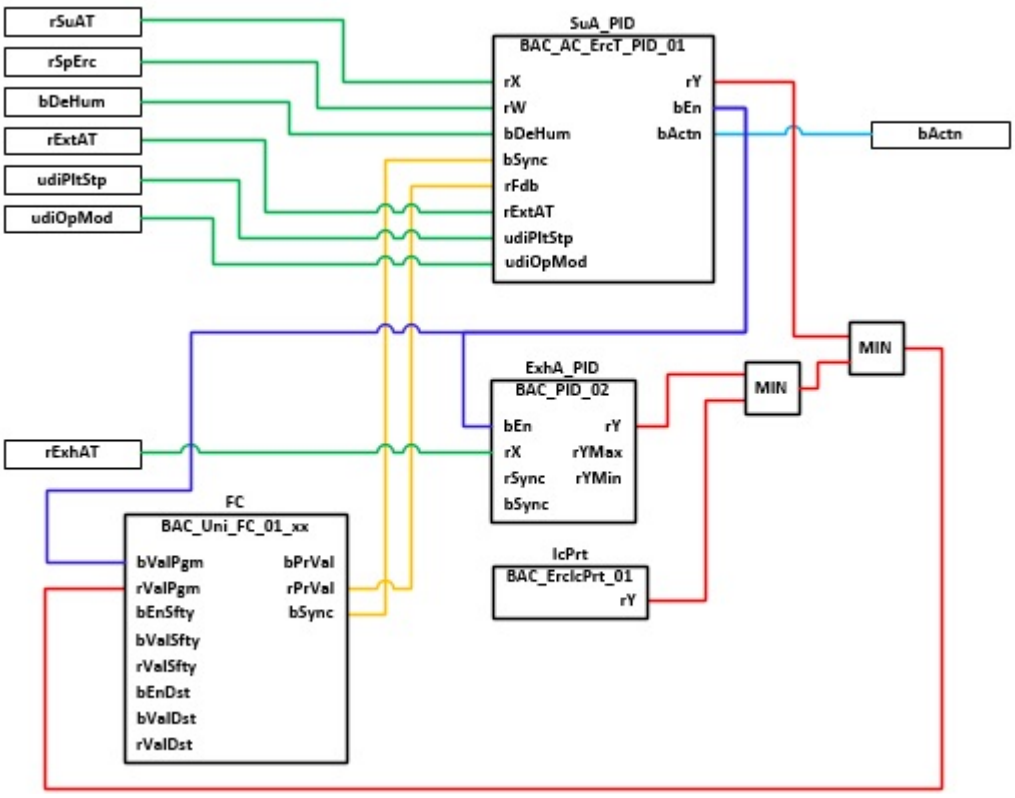
**Plant diagram 01**



### Plant diagram 02




### Block diagram




### VAR_INPUT

```
rSuAT       : REAL;
rSpErc      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
bDeHum      : BOOL;
rExhAT      : REAL;
rExtAT      : REAL;
```

**rSuAT:** Measured value supply air temperature

**rSpErc:** Set value supply air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection program BAC_AC_OpMod_01 [▶ 515].

**bDeHum:** Status message dehumidification mode. If dehumidification is active, energy recovery is disabled in heating mode. If the heat recovery unit recovers 'coolth' from the outlet air, it is switched to 100% in dehumidification mode.

**rExhAT:** Measured value exhaust air temperature

**rExtAT:** Measured value outlet air temperature

### VAR_OUTPUT

```
bActn      : BOOL;
```

**bActn:** control direction of the supply air temperature control **SuA_PID**. The control direction of the energy recovery is required by various setpoint programs or cascade controllers for determining setpoint for the supply air temperature of the energy recovery unit, e.g. BAC_AC_SpSuAT_02 [▶ 672], BAC_AC_CasCtrlT_02 [▶ 643].

### Program description

| Instance | Type | Task |
|---|---|---|
| SuA_PID | BAC_AC_ErcT_PID_01 [▶ 412] | Subtemplate supply air temperature control. The temperature control is analog via PID sequence controller |
| ExhA_PID | BAC_PID_02 [▶ 589] | Subtemplate exhaust air temperature control for anti-icing |
| IcPrt | BAC_ErcIcPrt_01 [▶ 418] | Subtemplate for anti-icing of an energy recovery unit via differential pressure switch |
| Mot | BAC_Uni_FC_01_223 [▶ 555] | Subtemplate for controlling a frequency converter, which then controls the motor of the rotary heat exchanger. |
| | MIN, MIN | MIN selection of the control signals for supply air temperature control **SuA_PID**,exhaust air temperature control **ExhA_PID** and the frost protection ramp function |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.21 BAC_AC_ErcT_PID_01

### Functional description

The subtemplate **BAC_AC_ErcT_PID_01** is an supply air temperature sequence controller for energy recovery.
The set value, actual value and control output are referenced via the BACnet value objects **X**, **W** and **Y**.

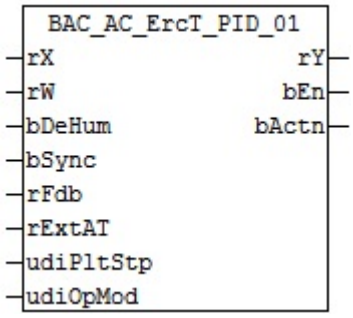
The PID sequence controller is enabled based on the plant operation mode **udiOpMod** and the global temperature communication structure **g_stSeqLinkT[PLT_NUM]**. This data and command structure is the link between the individual sequence controllers and the corresponding control function block FB_BA_SeqLink [▶ 168] of a plant.

The BACnet BV object **En** is used to display the controller enable.

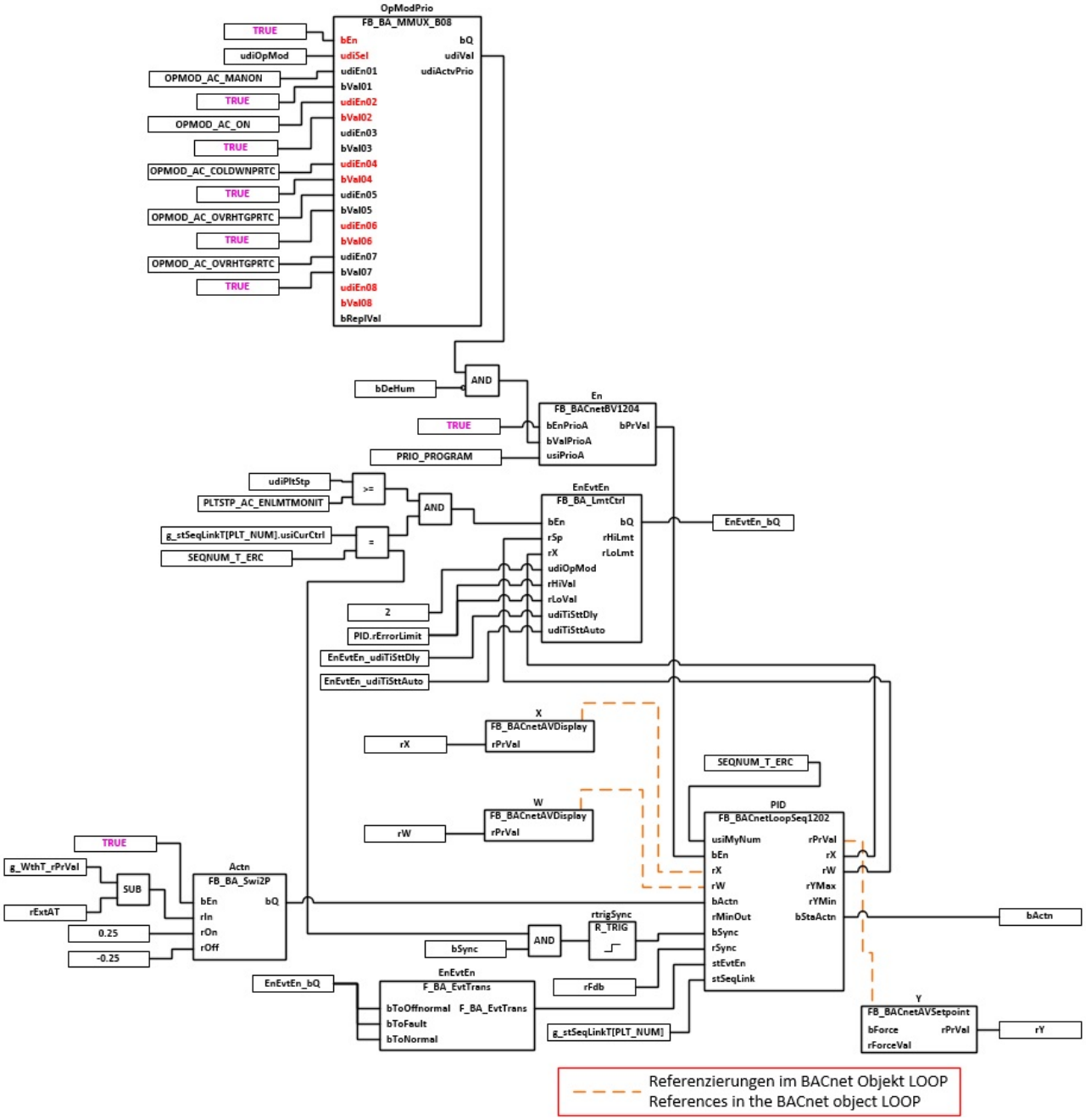
The limit value monitoring of PID controller is controlled by the function block **EnEvtEn** as a function of the plant startup process.

In dehumidification mode **bDeHum** = TRUE, the PID sequence controller is disabled.

**Interface**

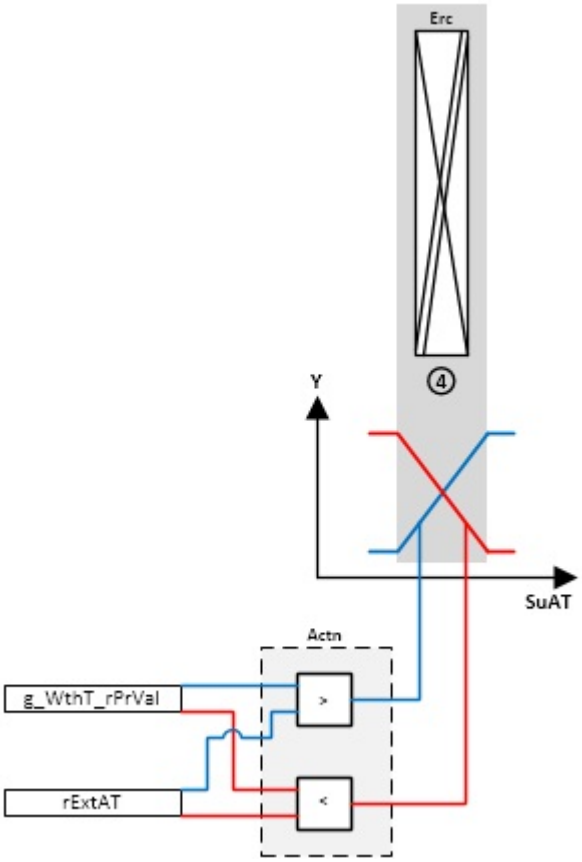


## Block diagram




## Direction of action

The direction of action of the PID sequence controller is selected based on a comparison of the outside temperature with the exhaust air temperature.
If the outside temperature is lower than the outlet air temperature, the direction of action of the PID controller is indirect (heating mode)
If the outside temperature is higher than the outlet air temperature, the direction of action of the PID controller is direct (cooling mode)




### VAR_INPUT

```
rX          : REAL;
rW          : REAL;
bDeHum      : BOOL;
bSync       : BOOL;
rFdb        : REAL;
rExtAT      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rX:** Measured value supply air temperature

**rW:** Set value of the supply air temperature

**bDeHum:** Dehumidification active, supply air temperature controller disabled. If dehumidification is active, energy recovery is disabled in heating mode. If the heat recovery unit recovers ‘coolth’ from the outlet air, it is switched to 100% in dehumidification mode.

**bSync:** Input for synchronisation of the controller

**rFdb:** Position feedback actuator

**rExtAT:** Measured value outlet air temperature

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rY          : REAL;
bEn         : BOOL;
bActn       : BOOL;
```

**rY:** Control value output

**bEn:** Energy recovery control enabled

**bActn:** Direction of action of the supply air controller, required for the set value strategy; TRUE = direct = cooling; FALSE = reverse = heating

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkT[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task | | | |
|---|---|---|---|---|---|
| X | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the actual value input of the BACnet loop object | | | |
| W | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the setpoint input of the BACnet loop object | | | |
| OpModPrio | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the enable conditions of the sequence controller depending on the plant operation mode. | | | |
| | | **udiOpMod** | | **Enable** | **Comment** |
| | | OPMOD_AC_MANON [▶ 357] | Manual on | TRUE | The plant is switched on manually via the plant selector switch |
| | | OPMOD_AC_ON [▶ 357] | On | TRUE | The plant runs in automatic mode via the timer program |
| | | OPMOD_AC_OVRHTGPRTC [▶ 357] | Overheating protection | TRUE | The plant operates in overheating protection mode |
| | | OPMOD_AC_COLDWNPRTC [▶ 357] | Support operation, cooling protection | TRUE | The plant operates in cooling protection mode |
| | | OPMOD_AC_FORCVENT [▶ 357] | Forced ventilation | TRUE | The plant operates in forced ventilation mode |

| Instance | Type | Task |
|---|---|---|
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the sequence controller enable in the MCL. |
| | AND | When OpModPrio is enabled in dehumidification mode ( bDeHum = TRUE) , energy recovery is disabled. |
| Actn | FB_BA_Swi2P [▶ 146] | The function block FB_BA_Swi2P [▶ 146] determines the control direction of the sequence controller.<br>If the subtraction of the outside temperature and the exhaust air temperature is < - 0.25, the output of the function block **Actn** is FALSE. The control direction of the loop object is thus indirect and the energy recovery is in heating mode.<br>If the subtraction of the outside temperature and the exhaust air temperature is > + 0.25, the output of the function block **Actn** is TRUE. The control direction of the loop object is thus direct and the energy recovery is in cooling mode.<br>The hysteresis between - 0.25 and + 0.25 of the Actn function block prevents frequent switching of the control direction at approximately the same outside and exhaust air temperature. |
| EnEvtEn | FB_BA_LmtCtrl [▶ 230] | The BACnet loop object PID monitors the control function by comparing the setpoint **W** with the actual value **X**. If the deviation W-X is greater than the property **ErrorLimit**, then the loop object sends a message to the MCL.<br><br>When the plant is at a standstill, at the moment of startup and until the plant is in a controlled state, the loop object is suppressed so that no incorrect messages are sent to the MCL. Reporting of the loop object should not be activated until the air-conditioning system is fully running and the control is stable. In addition, reporting is enabled if the control has not reached the range around the setpoint defined by the property **ErrorLimit** after a long time.<br><br>Enabling the object internal reporting is done by writing to the BACnetEventTransitionBits of the loop object.<br><br>**The following conditions must be met to enable reporting from the loop object:**<br><br>**1.**<br>The plant start program BAC_AC_StartT_01 [▶ 530] has enabled control monitoring and sensor limit monitoring<br>**udiPltStp** >= **PLTSTP_AC_ENLMTMONIT**<br>and<br>energy recovery is the active controller in the control sequence.<br>g_stSeqLinkT[PLT_NUM] [▶ 357]**.usiCurCtrl** = **SEQNUM_T_ERC**<br>and<br>the supply air temperature has approached the setpoint to a point where it has settled into a range between **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit**<br>and<br>the supply air temperature must have remained within the range of **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit** for at least the duration of **EnEvtEn_udiTiSttDly**.<br><br>**2.**<br>The timer **EnEvtEn_udiTiSttAuto** has expired and the control has not reached its setpoint range. |
| | F_BA_EvtTrans | writes to the BACnetEventTransitionBits to_offnormal, to_fault and to_normal of the loop object. The input bEnEvtEn must be TRUE |

| Instance | Type | Task |
|---|---|---|
| PID | FB_BACnetLoopSeq1202 [▶ 110] | Sequence controller supply air temperature energy recovery. |
| rtrigSync | R_TRIG | A rising edge at input **bSync** triggers synchronization of the loop object to the value of **lrSync**.<br>If the actuator of the energy recovery was overridden by writing a higher priority to the corresponding AO object of the MCL or by activating the local priority operation, the current position of the actuator deviates from the output of the loop object. The variables**bSync** and **rFdbVlv** can be used to restore synchronicity between the position of the actuator and the controller. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |
| | AND, SEL, MAX | Network which outputs the output value **rY** by the **MAX** selection.<br>On the one hand, the value of the supply air temperature controller PID is output at the **MAX** selection.<br>On the other hand, a value of 100 or 0 is output depending on the dehumidification operation **bDeHum** and the comparison of the outdoor temperature with the exhaust air temperature. |

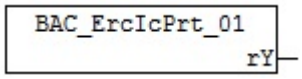
### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

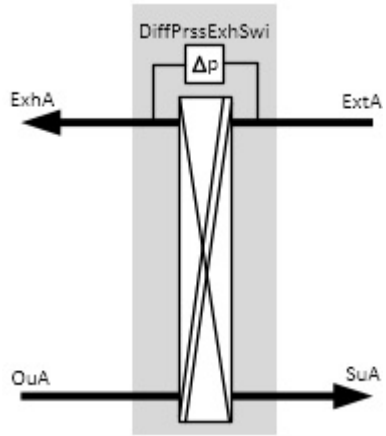
## 9.22 BAC_ErcIcPrt_01

### Functional description

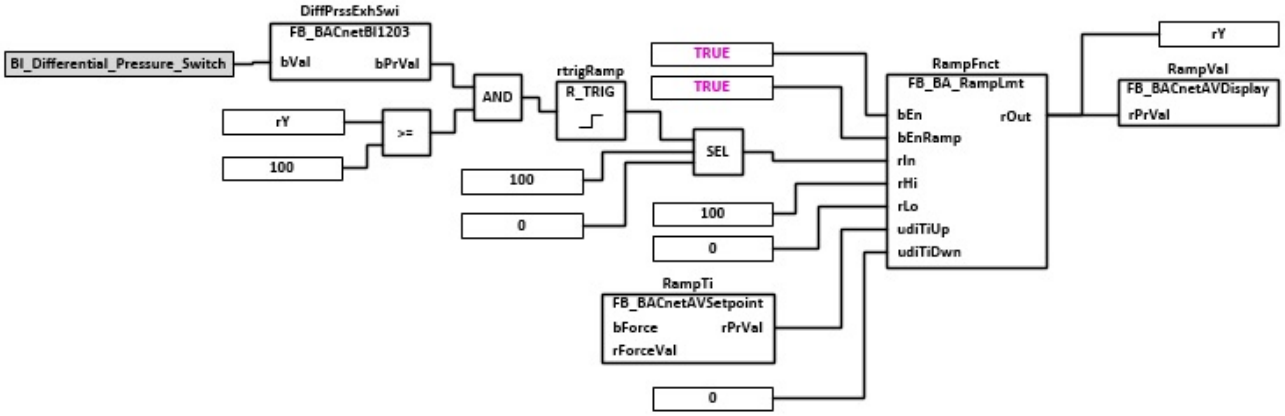
The template is used to protect an energy recovery unit from icing up.
If the differential pressure switch **DiffPrssExhSwi** is triggered, a ramp function is activated, which issues an analog limit signal.
The limitation of the energy recovery results in thawing through the warm exhaust air flow.




### System diagram



### Block diagram




### VAR_OUTPUT

```
rY          : REAL;
```

**rY:** Ramp signal output. If the differential pressure switch is not triggered, 100% is output at *rY*

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| DiffPrssExhSwi | FB_BACnetBI1203 [▶ 72] | BI object differential pressure switch via the heat exchanger on the exhaust air side |
| RampTi | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the ramp signal rise time |
| RampFnct | FB_BA_RampLmt [▶ 141] | Ramp function, which generates a ramp signal that increases from 0%, if the differential pressure monitor was triggered. |
| RampVal | FB_BACnetAVDisplay [▶ 69] | The AV object indicates the ramp signal. |
| rtrigRamp | R_TRIG<br>SEL | Specifies the starting value 0% for the ramp function for one cycle, if the differential pressure switch was triggered. |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| BI_Differential_Pressure_Switch | BOOL | | Input | Digital input - message - differential pressure switch |

### Version history

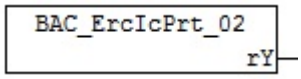
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.23 BAC_ErcIcPrt_02

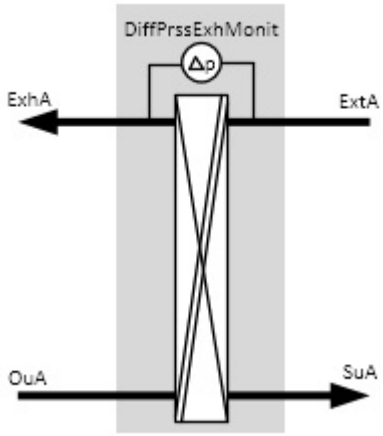
### Functional description

The template is used to protect an energy recovery unit from icing up.
On actuation of the HighLimit of the differential pressure sensor **DiffPrssExhMonit** a ramp function is activated, which generates an analog limit signal.
The limitation of the energy recovery results in thawing through the warm exhaust air flow.

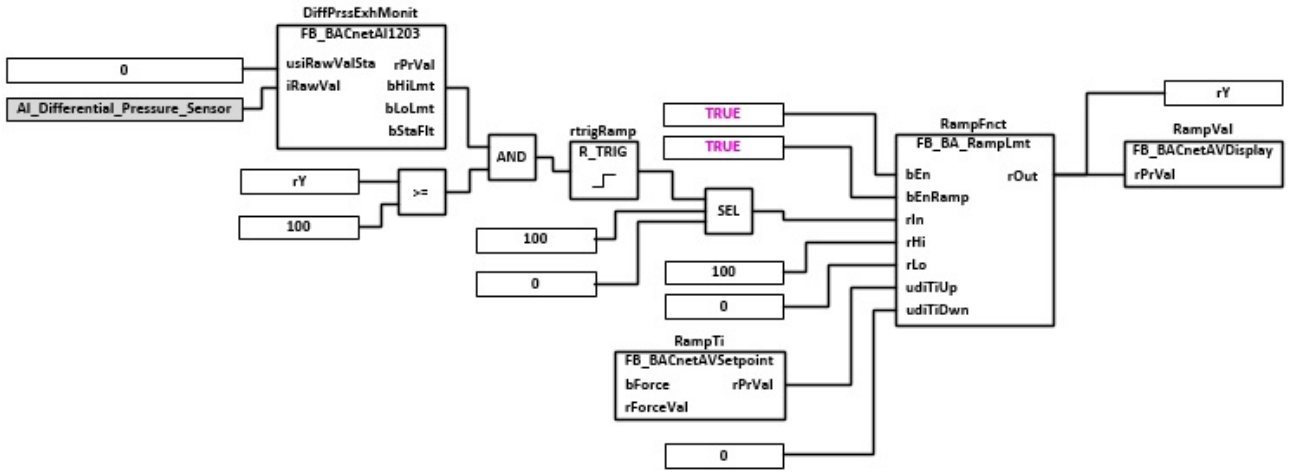
### Interface




### System diagram




### Block diagram



### VAR_OUTPUT

```
rY          : REAL;
```

**rY:** Ramp signal output. If the HighLimit of the differential pressure sensor is not triggered, 100% is output at *rY*

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| DiffPrssExhMonit | FB_BACnetAI1203 [▶ 49] | Differential pressure sensor over the heat exchanger on the exhaust air side |
| RampTi | FB_BACnetAVSetpoint [▶ 69] | Input of the rise time of the ramp signal |
| RampFnct | FB_BA_RampLmt [▶ 141] | Ramp function, which generates a ramp signal that increases from 0%, if the differential pressure sensor was triggered. |
| RampVal | FB_BACnetAVDisplay [▶ 69] | Display of the ramp signal |
| rtrigRamp | R_TRIG<br>SEL | Specifies the starting value 0% for the ramp function for one cycle, if the differential pressure switch was triggered. |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| AI_Differential_Pressure_Sensor | INT | | Input | Analog input - measured value - differential pressure exhaust air |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.24 BAC_AC_ExtAFan_FC_01

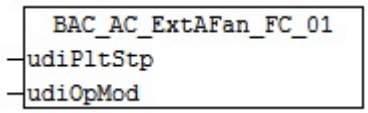
### Application

The template **BAC_AC_ExtAFan_FC_01** is used for controlling a pressure-controlled outlet air fan.

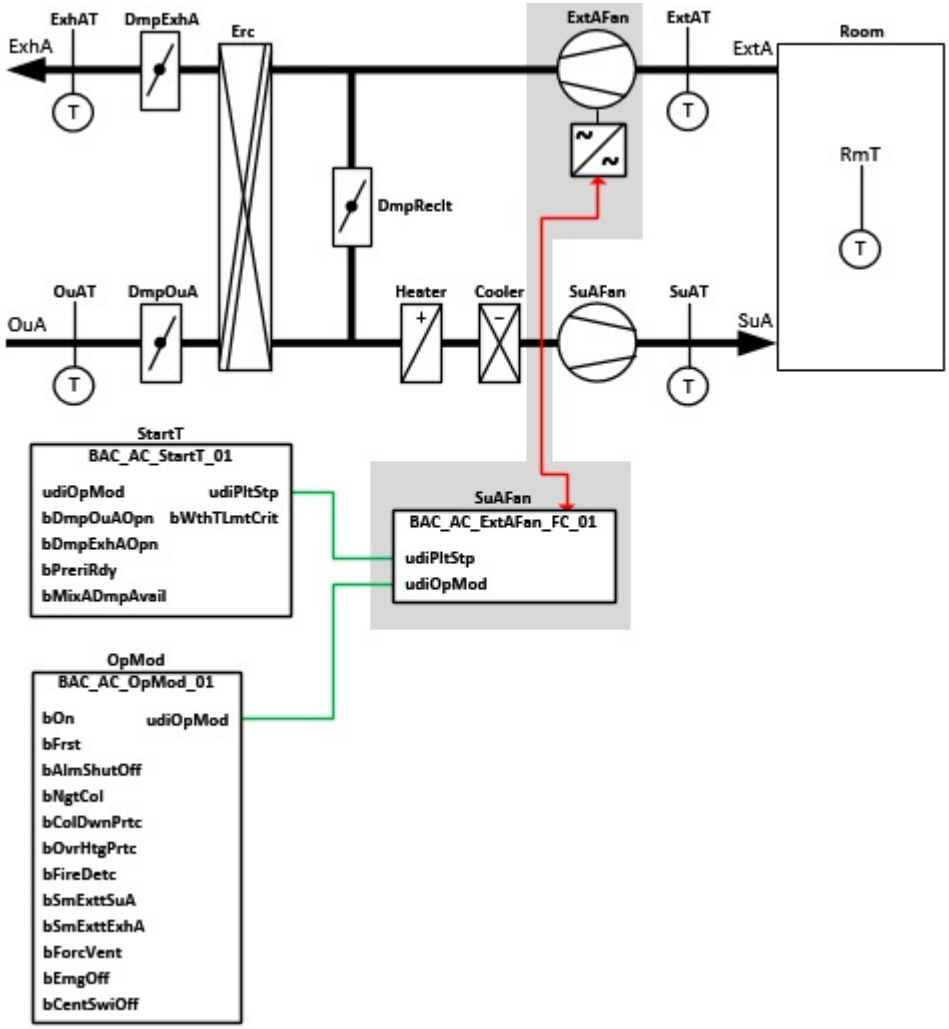
The main tasks of the template are:

- pressure control of the outlet air fan
- control of a frequency converter
- differential pressure monitoring of the fan

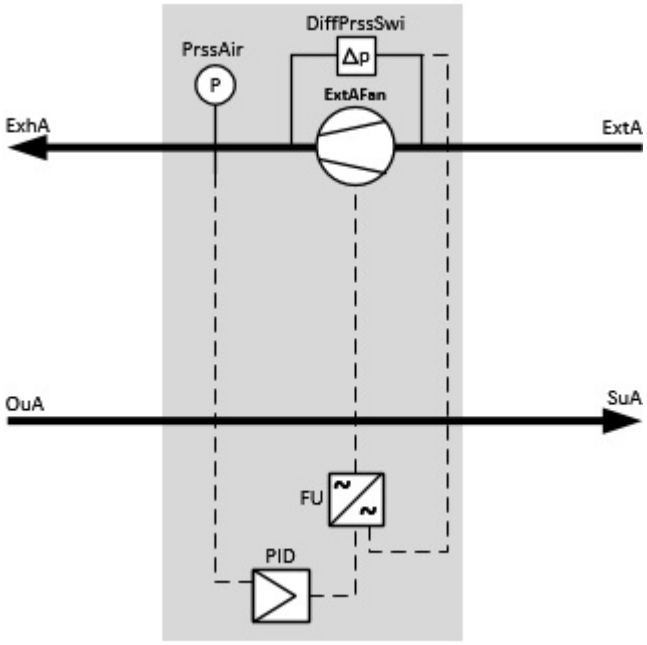
### Interface



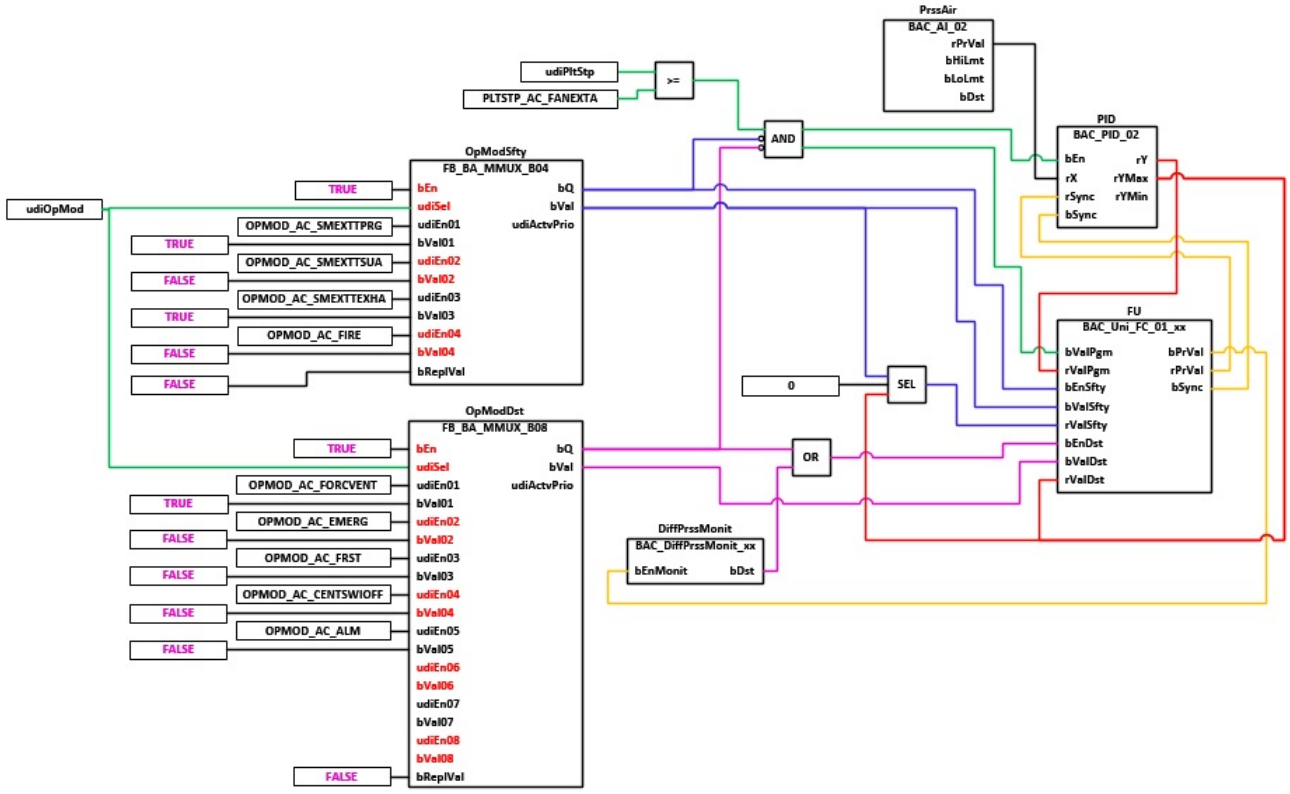

### Plant diagram 01



### Plant diagram 02




### Block diagram




### VAR_INPUT

```
udiPltStp       : UDINT;
udiOpMod        : UDINT;
```

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### Program description

| Instance | Type | Task | | |
|---|---|---|---|---|
| PrssAir | BAC_AI_02 [▶ 676] | Sub template pressure sensor **PrssAir** is the control value for the pressure regulator. | | |
| PID | BAC_PID_02 [▶ 589] | Sub template pressure control. | | |
| DiffPrssMonit | BAC_DiffPrssMonit_01 [▶ 433] | Sub template differential pressure monitoring via differential pressure switch | | |
| FC | BAC_Uni_FC_01_223 [▶ 555] | Sub template control of a frequency converter, including motor logic. | | |
| OpModSfty | FB_BA_MMUX_B04 [▶ 205] | The multiplexer defines the safety priority for the control of the outlet air fan **FC** , based on the plant operation mode **udiOpMod** | | |
| | | **udiOpMod** | | **Enable** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | TRUE |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction inlet air | FALSE |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | TRUE |
| | | OPMOD_AC_FIRE | Fire | FALSE |
| OpModDst | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the fault priority for the control of the outlet air fan **FC** , based on the plant operation mode **udiOpMod** | | |
| | | **udiOpMod** | | **Enable** |
| | | OPMOD_AC_FORCVENT | Forced ventilation | TRUE |
| | | OPMOD_AC_EMERG | emergency | FALSE |
| | | OPMOD_AC_FRST | Frost | FALSE |
| | | OPMOD_AC_CENTSWIOFF | Central shutdown | FALSE |
| | | OPMOD_AC_ALM | Fault/alarm | FALSE |
| | >=, AND | Network containing the enable for the outlet air fan as result, based on the plant steps **udiPltStp** | | |
| | SEL | SEL selection, which specifies the value for the safety priority of the template **FC** | | |
| | OR | The result of the OR operation is enable of the fault priority of template **FC**. | | |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.25 BAC_AC_ExtAFan1st_01

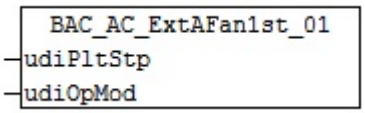
### Application

The call template **BAC_AC_ExtAFan1st_01** is used to control a single-stage extract air fan.

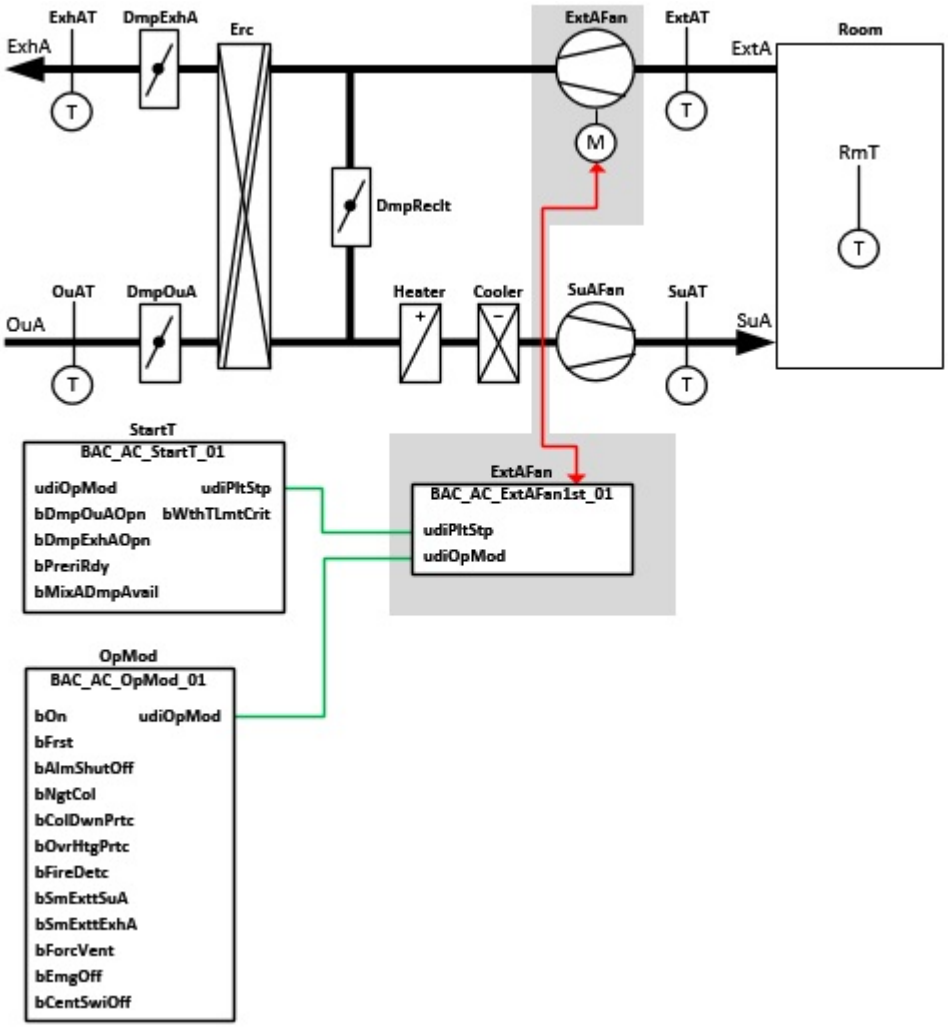
The main tasks of the template are:

- extract air fan control
- Differential pressure monitoring

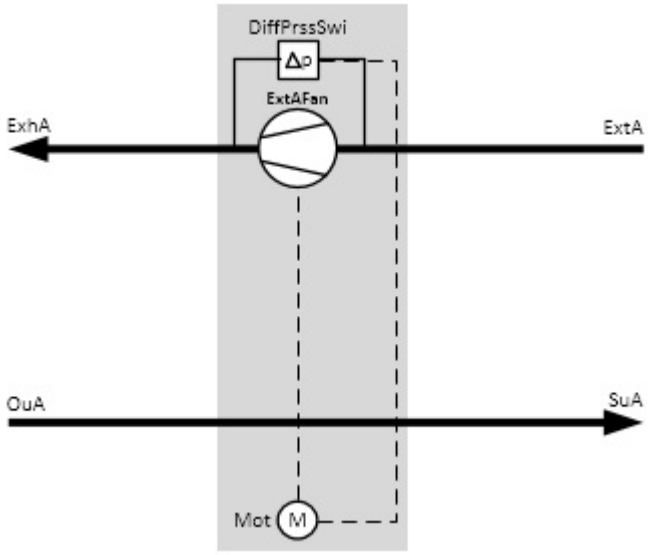
### Interface



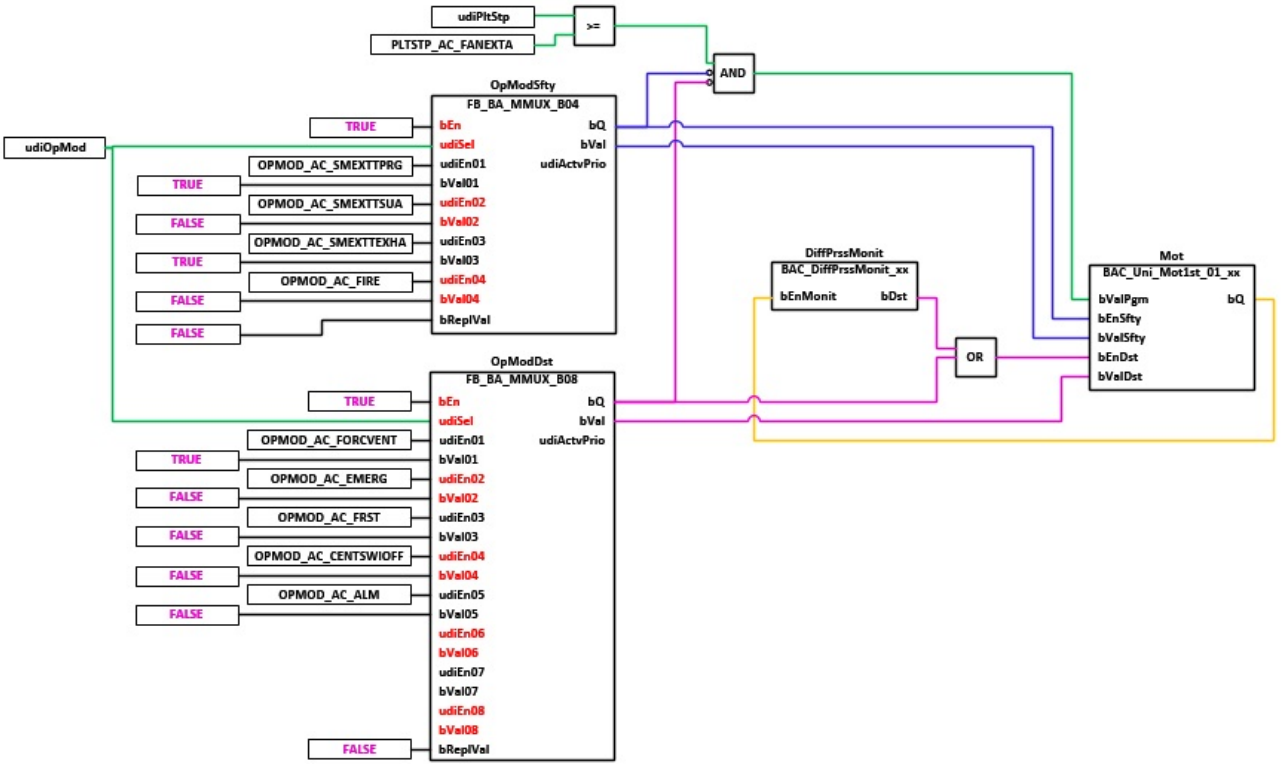

### Plant diagram 01



### Plant diagram 02




### Block diagram




### VAR_INPUT

```
udiPltStp       : UDINT;
udiOpMod        : UDINT;
```

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### Program description

| Instance | Type | Task | | |
|---|---|---|---|---|
| DiffPrssMonit | BAC_DiffPrssMonit_01 [▶ 433] | Subtemplate differential pressure monitoring via differential pressure switch | | |
| Mot | BAC_Uni_Mot1st_01_29 [▶ 567] | Subtemplate control of a single-stage motor including motor logic. | | |
| OpModSfty | FB_BA_MMUX_04 [▶ 205] | The multiplexer defines the safety priority for the control of the extractair fan **Mot**, based on the plant operation mode *udiOpMod* | | |
| | | **udiOpMod** | | **Enable** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | TRUE |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction inlet air | FALSE |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | TRUE |
| | | OPMOD_AC_FIRE | Fire | FALSE |
| OpModDst | FB_BA_MMUX_08 [▶ 205] | The multiplexer defines the fault priority for the control of the extractair fan **Mot**, based on the plant operation mode *udiOpMod* | | |
| | | **udiOpMod** | | **Enable** |
| | | OPMOD_AC_FORCVENT | Forced ventilation | TRUE |
| | | OPMOD_AC_EMERG | emergency | FALSE |
| | | OPMOD_AC_FRST | Frost | FALSE |
| | | OPMOD_AC_CENTSWIOFF | Central shutdown | FALSE |
| | | OPMOD_AC_ALM | Fault/alarm | FALSE |
| | AND >= | The result of the network is enable for the extractair fan, based on the plant steps **udiPltStp** | | |
| | OR | The result of the OR operation is enable of the fault priority of template **Mot**. | | |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.26 BAC_AC_SuAFan_FC_01

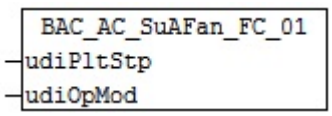
### Application

The call template BAC_AC_SuAFan_FC_01 is used for controlling a pressure-controlled supply air fan.

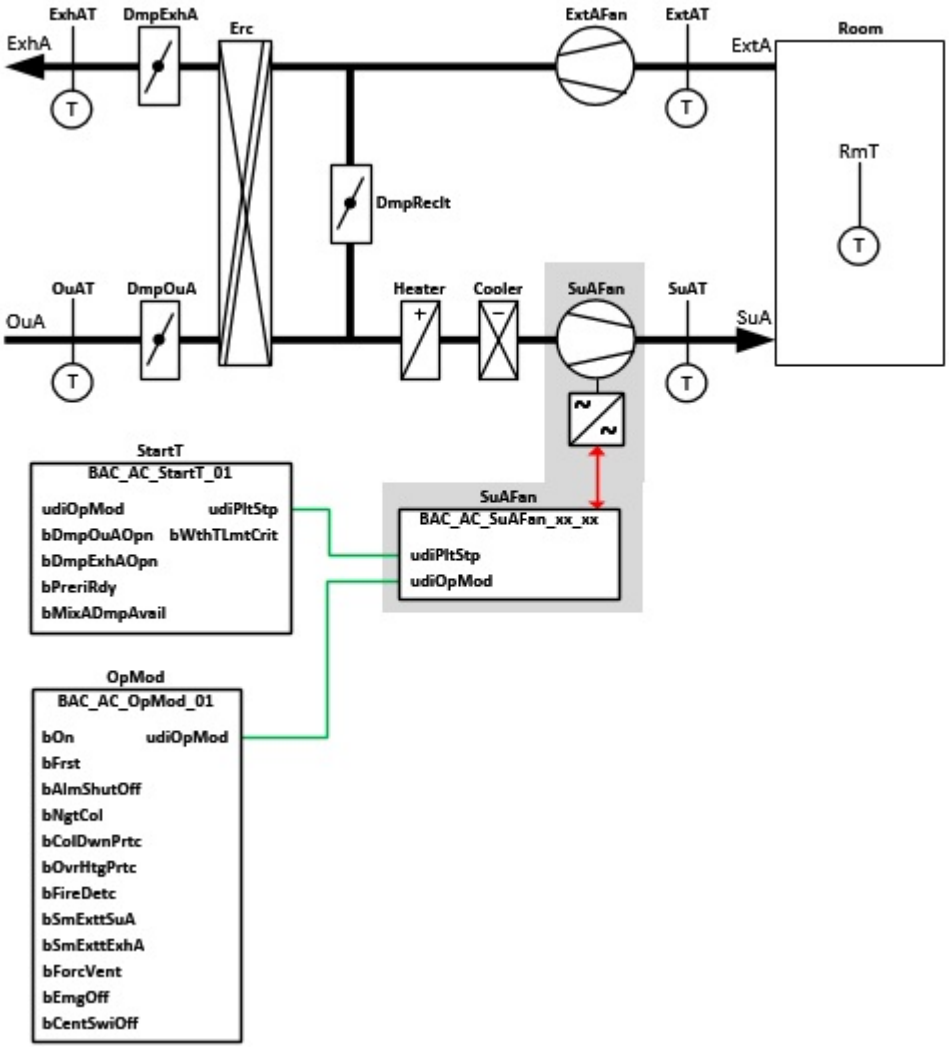
The main tasks of the template are:

- Pressure control of the supply air fan
- control of a frequency converter
- Differential pressure monitoring

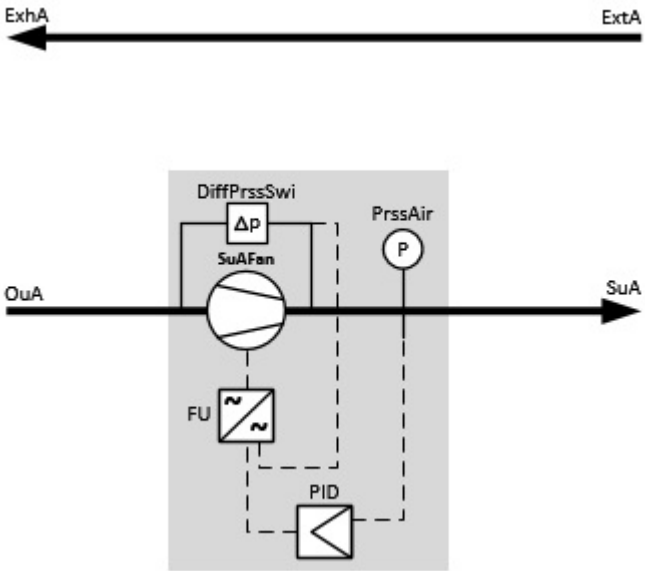
### Interface



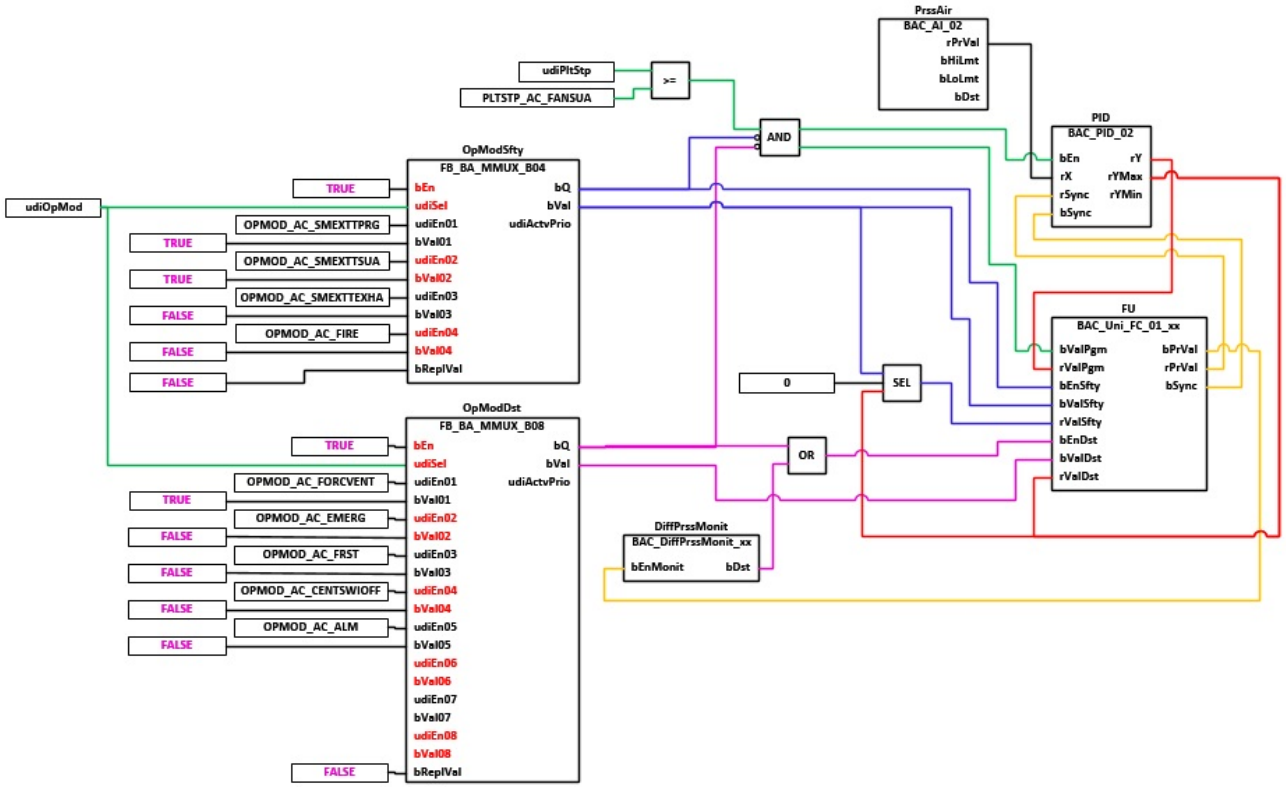
### Plant diagram 01




### Plant diagram 02



### Block diagram




### VAR_INPUT

```
udiPltStp       : UDINT;
udiOpMod        : UDINT;
```

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### Program description

| Instance | Type | Task | | |
|---|---|---|---|---|
| PrssAir | BAC_AI_02 [▶ 676] | Subtemplate pressure sensor **PrssAir**is the control value for the pressure regulator **PID**. | | |
| PID | BAC_PID_02 [▶ 589] | Subtemplate pressure control. The control is analog via PID controller | | |
| DiffPrssMonit | BAC_DiffPrssMonit_01 [▶ 433] | Subtemplate differential pressure monitoring via differential pressure switch | | |
| FU | BAC_Uni_FC_01_223 [▶ 555] | Template control of a frequency converter, including motor logic. | | |
| OpModSfty | FB_BA_MMUX_B04 [▶ 205] | The multiplexer defines the safety priority for the control of the supply air fan **FU**, based on the plant operation mode **udiOpMod** | | |
| | | **udiOpMod** | | **Enable** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | TRUE |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | TRUE |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | FALSE |
| | | OPMOD_AC_FIRE | Fire | FALSE |

| Instance | Type | Task | | |
|---|---|---|---|---|
| OpModDst | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the fault priority for the control of the supply air fan **FU**, based on the plant operation mode **udiOpMod** | | |
| | | **udiOpMod** | | **Enable** |
| | | OPMOD_AC_FORCVENT | Forced ventilation | TRUE |
| | | OPMOD_AC_EMERG | emergency | FALSE |
| | | OPMOD_AC_FRST | Frost | FALSE |
| | | OPMOD_AC_CENTSWIOFF | Central shutdown | FALSE |
| | | OPMOD_AC_ALM | Fault/alarm | FALSE |
| | >=, AND | Network containing the enable for the supply air fan as result, based on the plant steps **udiPltStp** | | |
| | SEL | SEL selection, which specifies the value for the safety priority of the template **FU** | | |
| | OR | The result of the OR operation is the enable of the disturbance priority of template **FU**. | | |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.27 BAC_AC_SuAFan1st_01

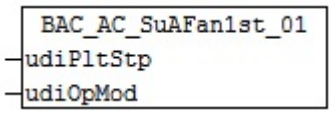
### Application

The call template **BAC_AC_SuAFan1st_01** is used for controlling a single-stage supply air fan.

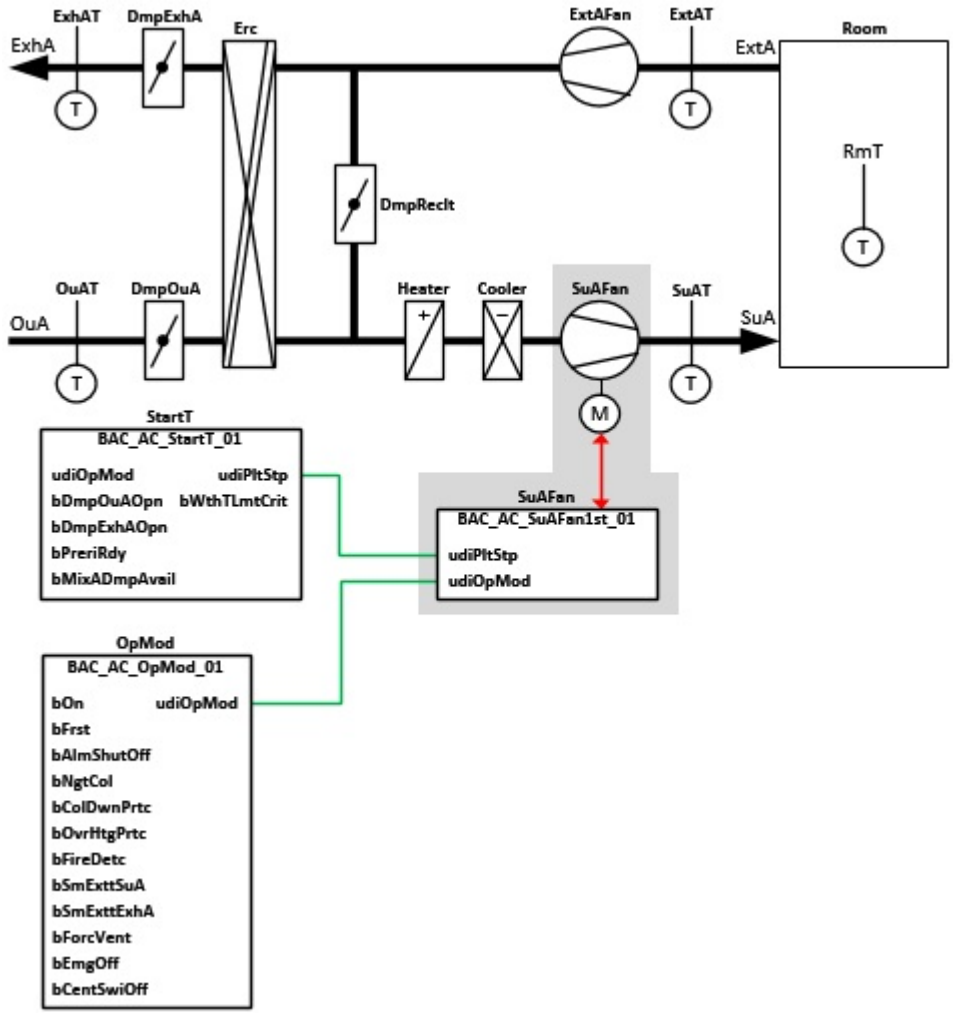
The main tasks of the template are:

- Control of supply air fan
- Differential pressure monitoring

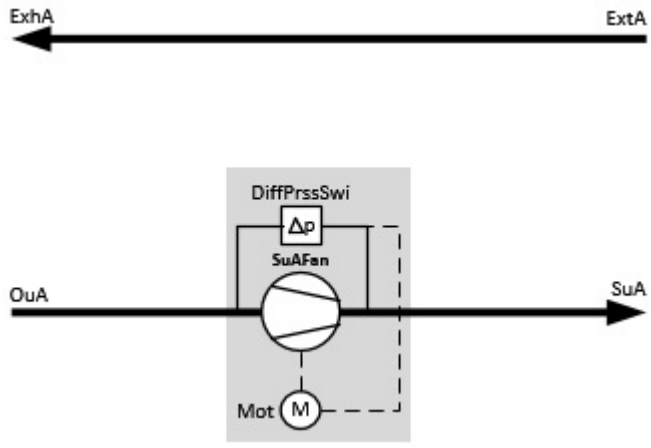
### Interface



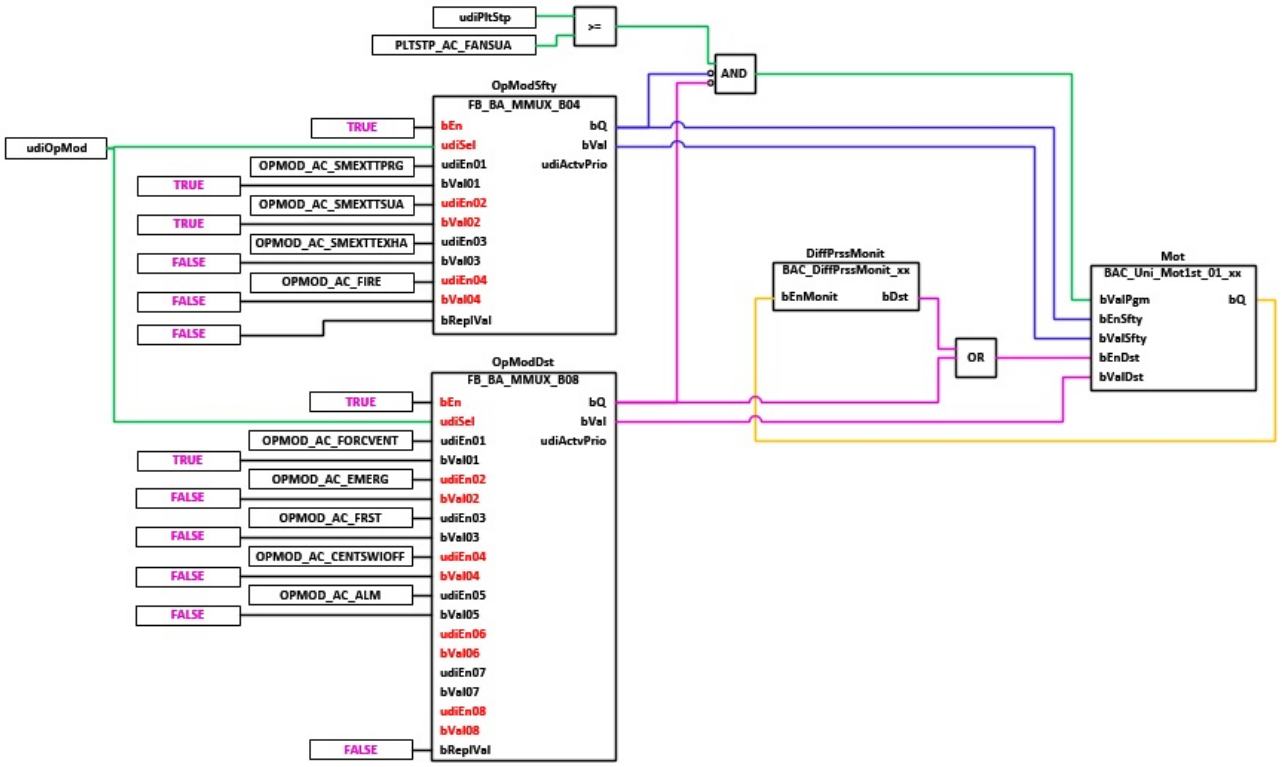
### Plant diagram 01




### Plant diagram 02



### Block diagram




### VAR_INPUT

```
udiPltStp       : UDINT;
udiOpMod        : UDINT;
```

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### Program description

| Instance | Type | Task | | |
|---|---|---|---|---|
| DiffPrssMonit | BAC_DiffPrssMonit_01 [▶ 433] | Subtemplate differential pressure monitoring via differential pressure switch | | |
| Mot | BAC_Uni_Mot1st_01_29 [▶ 567] | Subtemplate control of a single-stage motor including motor logic. | | |
| OpModSfty | FB_BA_MMUX_B04 [▶ 205] | The multiplexer defines the safety priority for the control of the supply air fan **Mot**, based on the plant operation mode **udiOpMod** | | |
| | | **udiOpMod** | | **Enable** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | TRUE |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | TRUE |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | FALSE |
| | | OPMOD_AC_FIRE | Fire | FALSE |
| OpModDst | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the fault priority for the control of the supply air fan **Mot**, based on the plant operation mode *udiOpMod*see "Extracted nested table 49" | | |
| | | **udiOpMod** | | **Enable** |

| Instance | Type | Task | | |
|---|---|---|---|---|
| | | OPMOD_AC_FORCVENT | Forced ventilation | TRUE |
| | | OPMOD_AC_EMERG | emergency | FALSE |
| | | OPMOD_AC_FRST | Frost | FALSE |
| | | OPMOD_AC_CENTSWIOFF | Central shutdown | FALSE |
| | | OPMOD_AC_ALM | Fault/alarm | FALSE |
| | >=<br>AND | The result of the network is the enable for the supply air fan based on the system steps **udiPltStp** | | |
| | OR | The result of the OR operation is the enable of the disturbance priority of template **Mot**. | | |

### Version history

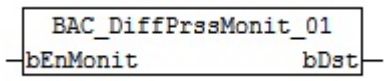
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.28 BAC_DiffPrssMonit_01

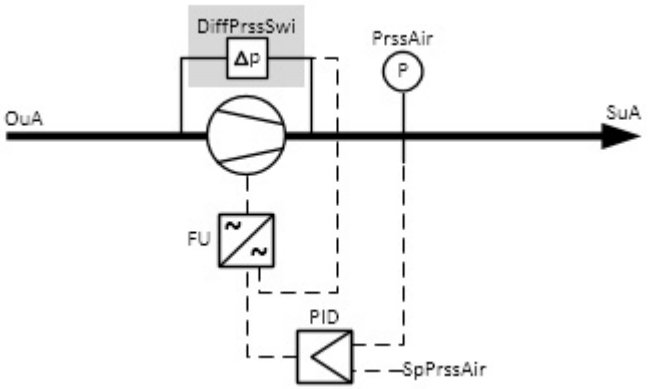
### Functional description

The template realises differential pressure monitoring of a fan or a pump via a differential pressure switch.

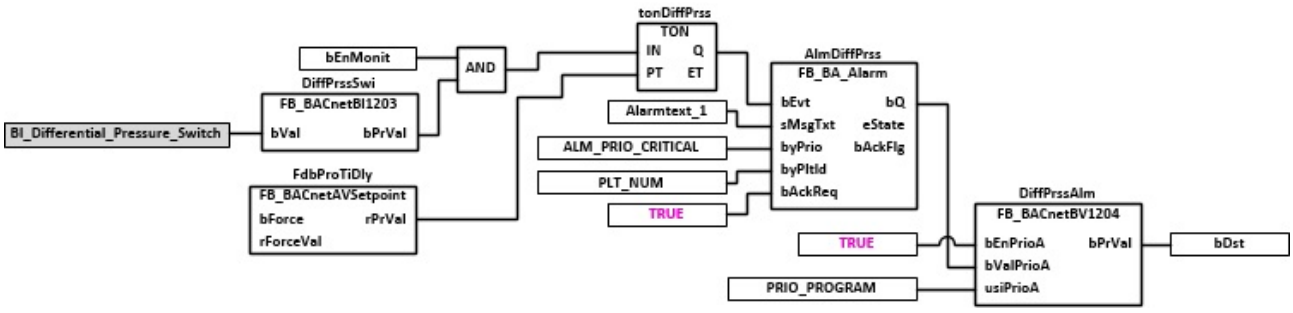
### Interface




### System diagram




### Block diagram



### VAR_INPUT

```
bEnMonit     : BOOL;
```

**bEnMonit:** Enable differential pressure monitoring. The fan or pump is switched on

### VAR_OUTPUT

```
bDst     : BOOL;
```

**bDst:** Display fault message differential pressure monitoring

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| DiffPrssSwi | FB_BACnetBI1203 [▶ 72] | BI object differential pressure monitor for differential pressure monitoring |
| FdbProTiDly | FB_BACnetAVSetpoint [▶ 69] | AV object for entering the time delay for process feedback *FdbProTiDly.*The message of the differential pressure monitor can be delayed to buffer pressure fluctuations. |
| tonDiffPrss | TON | Timing element for delayed output of the process feedback from the differential pressure monitor as alarm. The timing element can be helpful for buffering pressure fluctuations. |
| AlmDiffPrss | FB_BA_Alarm [▶ 179] | The collective alarm function block is used for integrating the differential pressure fault in the group alarm for the corresponding plant. |
| DiffPrssAlm | FB_BACnetBV1204 [▶ 94] | BV object for display of the differential pressure monitoring alarm |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_Differential_Pressure_Switch | BOOL | | Input | Digital input - message - differential pressure switch |

### Version history

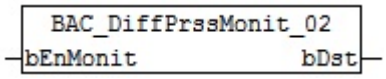
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.29 BAC_DiffPrssMonit_02

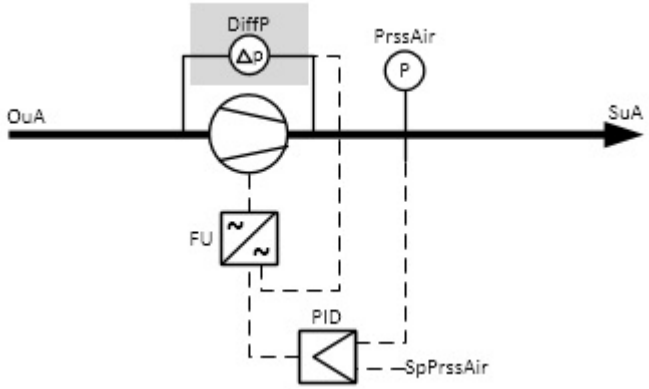
### Functional description

The template realizes differential pressure monitoring of a fan or a pump via a differential pressure sensor.

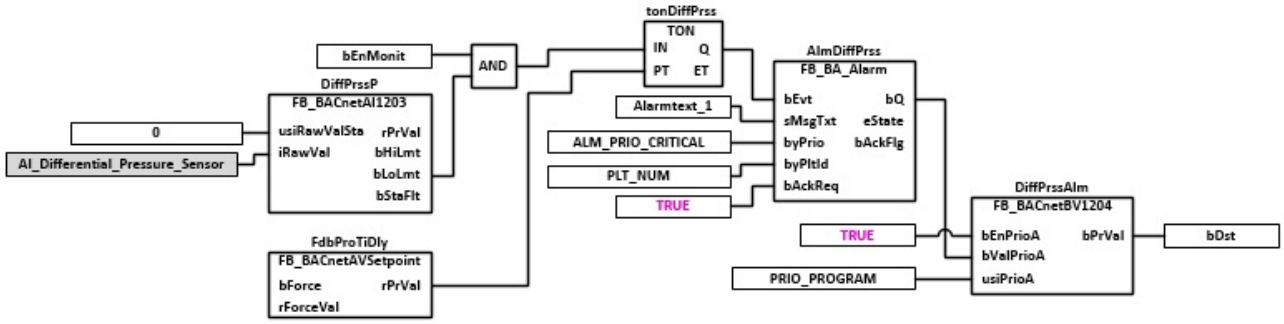
### Interface




### System diagram




### Block diagram




### VAR_INPUT

```
bEnMonit     : BOOL;
```

**bEnMonit:** enable differential pressure monitoring. The fan or pump is switched on.

### VAR_OUTPUT

```
bDst     : BOOL;
```

**bDst:** Display fault message differential pressure monitoring

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| DiffP | FB_BACnetAI1203 [▶ 49] | AI object differential pressure sensor for differential pressure monitoring. The BACnet property Low Limit triggers the differential pressure monitoring action. |
| FdbProTiDly | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the process feedback delay *FdbProTiDly*.The message from the differential pressure sensor can be delayed, to buffer pressure fluctuations. |
| tonDiffPrss | TON | Timing element for delayed output of the process feedback from the differential pressure sensor as alarm. |
| AlmDiffPrss | FB_BA_Alarm [▶ 179] | The collective alarm function block is used for integrating the differential pressure fault in the group alarm for the corresponding plant. |
| DiffPrssAlm | FB_BACnetBV1204 [▶ 94] | BV object for display of the differential pressure monitoring alarm |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| AI_Differential_Pressure_Sensor | INT | | Input | Analog input - measured value - differential pressure |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

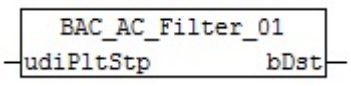
## 9.30 BAC_AC_Filter_01

### Functional description

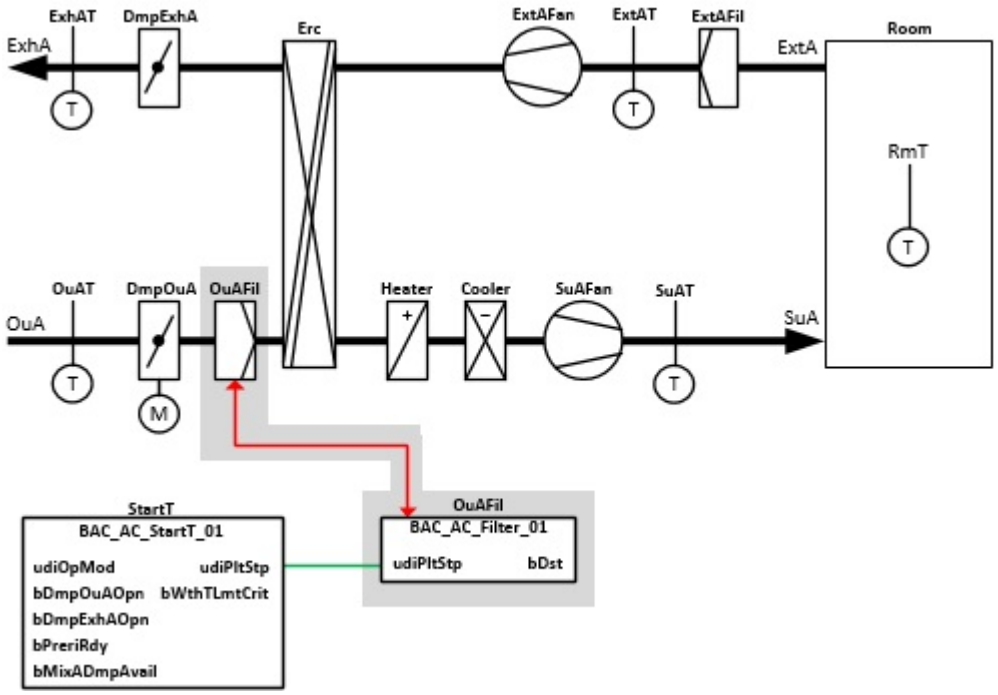
The template realises filter monitoring within an air conditioning plant via a differential pressure switch. If the differential pressure switch is triggered, filter contamination is logged.
A rising edge at the binary input of the differential pressure switch switches the RS flip-flop to latching. As a result, the filter contamination continues to be displayed, even if the plant shuts down. The RS flip-flop is reset after a plant restart without actuation of the differential pressure switch.

A BV object is used to display the filter contamination in the OWS.

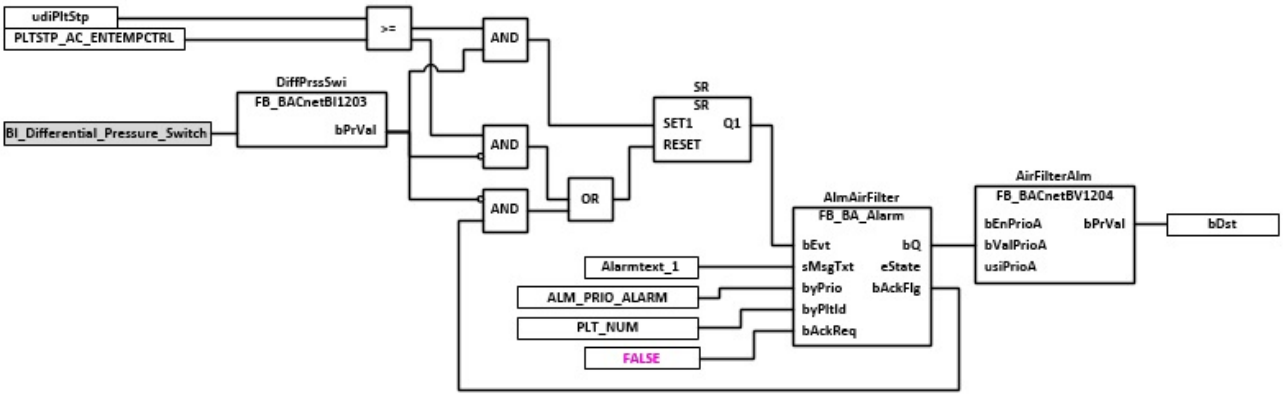
### Interface



### System diagram




### Block diagram




### VAR_INPUT

```
udiPltStp     : UDINT;
```

**udiPltStp :** The input variable is used to supply the template with the plant steps of the air-conditioning plant.BAC_StartPltStpT_01.

### VAR_OUTPUT

```
bDst      : BOOL;
```

**bDst:** The output variable indicates that filter monitoring was triggered and issues an error for further processing.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| DiffPrssSwi | FB_BACnetBI1203 [▶ 72] | BI object differential pressure monitor for filter monitoring |
| tonDiffPrss | TON | Response delay of the filter monitoring |
| tonDiffPrss | RS | Error message memory for filter monitoring |
| AlmDiffPrss | FB_BA_Alarm [▶ 179] | The collective alarm function block logs the filter monitoring fault and forwards it via an internal variable to the plant alarm collector BAC_PltIniAlm_01<br>. |
| DiffPrssAlm | FB_BACnetBV1204 [▶ 94] | BV object for displaying the filter monitoring |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| BI_Differential_Pressure_Switch | BOOL | | Input | Digital input - message - differential pressure switch |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

### Also see about this

- BAC_AC_StartT_01 [▶ 530]
- BAC_PltAlm_01 [▶ 365]

## 9.31 BAC_AC_ExhADmp2P_01_xx

### Functional description

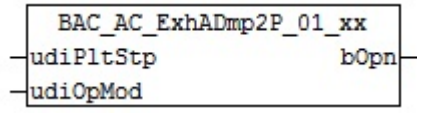
The sub template **BAC_AC_ExhADmp2P_01_xx** is used for control and monitoring of an exhaust air damper with spring return actuator and limit switch monitor. The damper is controlled via a binary output.

> The output variable bOpn outputs the actual state of the damper, if feedback from limit switch SwiOpn is available in the template used. If no end position monitoring is available, it is emulated internally. If the damper is controlled, output bOpn becomes TRUE. As a result, in the start program for a ventilation system BAC_AC_StartT_01 [▶ 530], the delay time for starting the fans (SuAFanDlyOn / ExtAFanDlyOn) has to be adjusted to the time it takes for the damper to open.

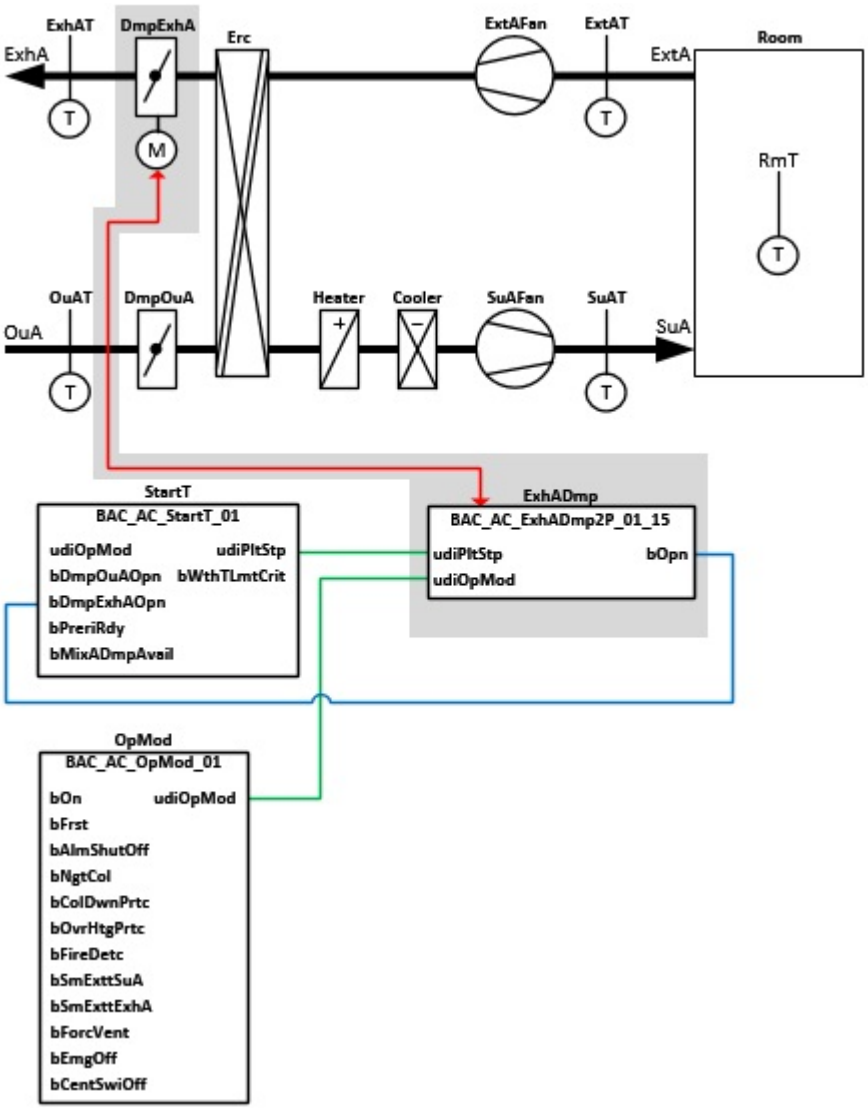
The template **BAC_AC_ExhADmp2P_01_xx** is available in different versions.
The damper versions are identified by means of a key. The identification key is derived from the table below.

| Options | mechanical priority operation Feedback hand switch | mechanical priority operation Feedback Relay output | Final position Open | Final position Close |
|---|---|---|---|---|
| **Instance** | **LocSwi** | **FdbOut** | **SwiOpn** | **SwiCls** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** |
| | **8** | **4** | **2** | **1** |
| BAC_AC_ExhADmp2P_01_00 | 0 | 0 | 0 | 0 |
| BAC_AC_ExhADmp2P_01_02 | 0 | 0 | 1 | 0 |
| BAC_AC_ExhADmp2P_01_03 | 0 | 0 | 1 | 1 |
| BAC_AC_ExhADmp2P_01_08 | 1 | 0 | 0 | 0 |
| BAC_AC_ExhADmp2P_01_10 | 1 | 0 | 1 | 0 |
| BAC_AC_ExhADmp2P_01_11 | 1 | 0 | 1 | 1 |
| BAC_AC_ExhADmp2P_01_12 | 1 | 1 | 0 | 0 |
| BAC_AC_ExhADmp2P_01_14 | 1 | 1 | 1 | 0 |
| BAC_AC_ExhADmp2P_01_15 | 1 | 1 | 1 | 1 |

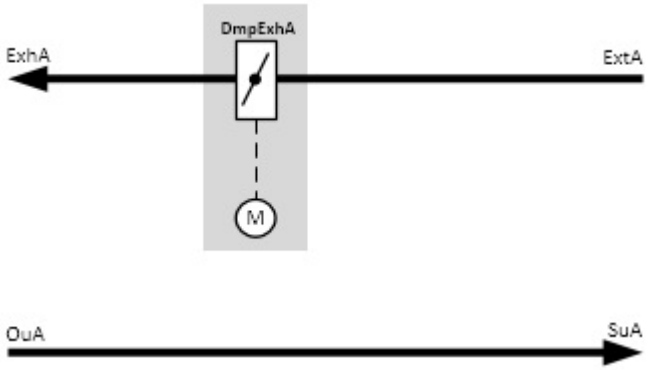
**Interface**



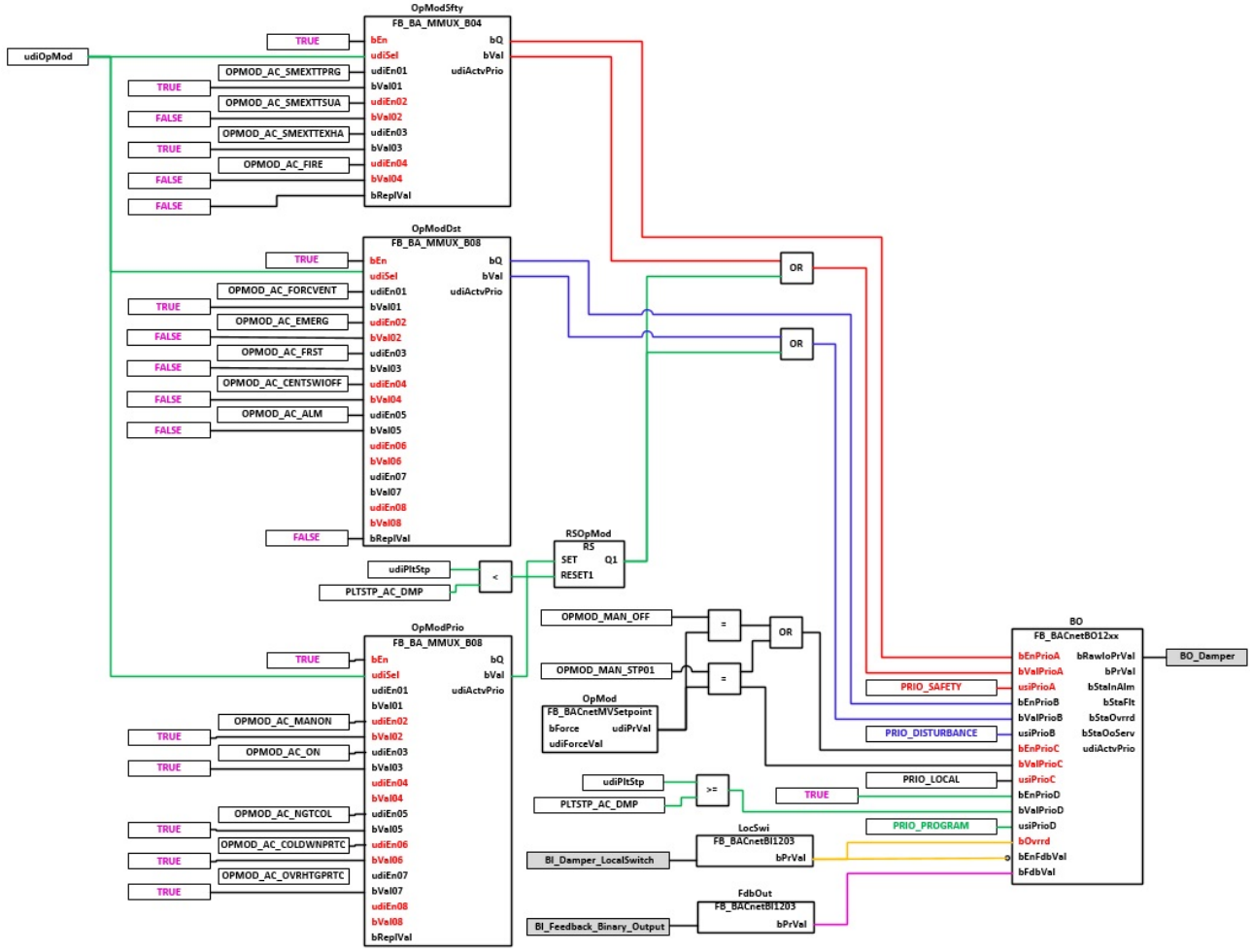
### Plant diagram 01



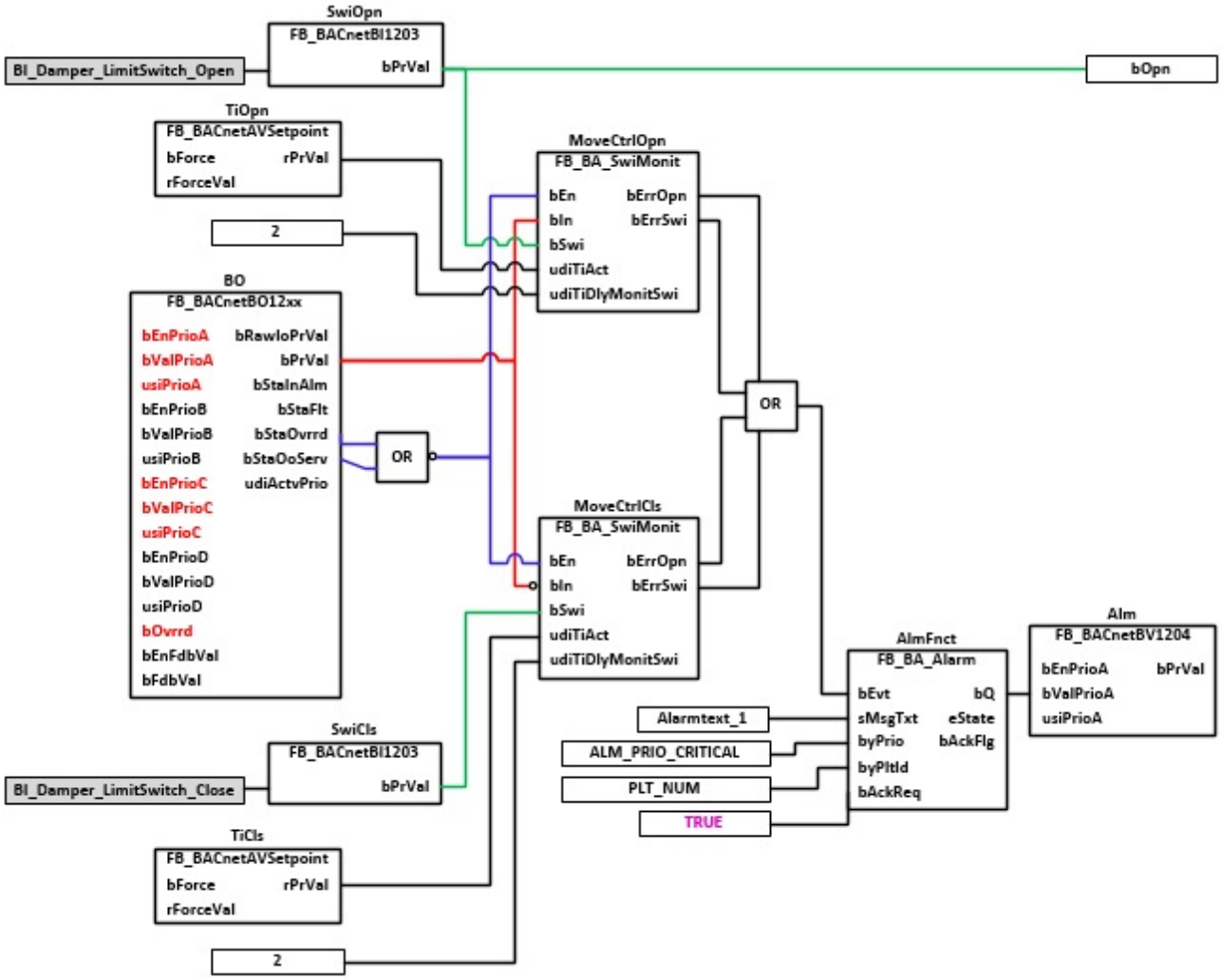

### Plant diagram 02



## Block diagram for controlling the damper



**Block diagram for monitoring the limit switches**




## VAR_INPUT

```
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**udiOpMode:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

## VAR_OUTPUT

```
bOpn        : BOOL;
```

**bOpn:** End position 'open' of the damper has been reached.
If no end position monitoring is available, it is emulated internally. If the damper is controlled, output **bOpn** becomes TRUE.

## VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task |
|---|---|---|---|
| SwiOpn | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'open' |
| SwiCls | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'closed' |
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the damper via the MCL or a local operator display |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object for feedback from mechanical priority operation. (manual/emergency operating level) |
| FdbOut | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation position feedback relay |
| TiOpn | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the opening time value |
| TiCls | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the closing time value |
| OpModSfty | FB_BA_MMUX_B04 [▶ 205] | | The multiplexer defines the safety priority of the exhaust air damper depending on the plant operation mode **udiOpMod** |
| | | | **udiOpMod** / **Damper state** |
| | | | OPMOD_AC_SMEXTTPRG / Smoke extraction program / Open |
| | | | OPMOD_AC_SMEXTTSUA / Smoke extraction supply air / Close |
| | | | OPMOD_AC_SMEXTTEXHA / Smoke extraction exhaust air / Open |
| | | | OPMOD_AC_FIRE / Fire / Close |
| OpModDst | FB_BA_MMUX_B08 [▶ 205] | | The multiplexer defines the disturbance priority of the exhaust air damper depending on the plant operation mode **udiOpMod** |
| | | | **udiOpMod** / **Damper state** |
| | | | OPMOD_AC_FORCVENT / Forced ventilation / Open |
| | | | OPMOD_AC_EMERG / emergency / Close |
| | | | OPMOD_AC_FRST / Frost / Close |
| | | | OPMOD_AC_CENTSWIOFF / Central shutdown / Close |
| | | | OPMOD_AC_ALM / Fault / Close |
| OpModPrio RSOpMod | FB_BA_MMUX_B08 [▶ 205] RS | | The multiplexer defines the plant operation mode **udiOpMod**. The function **RSOpMod** is intended to prevent the damper closing immediately if the operation mode changes while the process/fans are running. The start program is designed to shut down and restart the plant when the operation mode changes. |

| Instance | Type | optional | Task | | |
|---|---|---|---|---|---|
| | | | **udiOpMod** | | **Damper state** |
| | | | OPMOD_AC_MANON | Manual on | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_ON | On | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_NGTCOL | Night cooling | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_COLDWNPRTC | Support operation, cooling protection | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_OVRHTGPRTC | Overheating protection | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for controlling the damper | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | see multiplexer **OpModSftyudiOpMod** | see multiplexer **OpModSfty** Damper state |
| | | | PRIO_DISTURBANCE (3) | see multiplexer **OpModDstudiOpMod** | see multiplexer **OpModDst** Damper state |
| | | | PRIO_LOCAL (8) | The OR module pools events which enable writing to the priority manual override (local) of the downstream BO object.<br>Events:<br>1. The MV object has the value **OPMOD_MAN_OFF** (manual off)<br>2. The MV object has the value **OPMOD_MAN_STP01**(manual on) | Open, if **OpMod_udiPrVal** = **OPMOD_MAN_STP01** Otherwise close |
| | | | PRIO_PROGRAM (15) | TRUE | If the plant step is **udiPltStp** >= **PLTSTP_AC_DMP**, the damper should open |
| MoveCtrlOpn | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'open' end position of the damper | | |

| Instance | Type | optional | Task |
|---|---|---|---|
| MoveCtrlCls | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'closed' end position of the damper |
| AlmFnct | FB_BA_Alarm [▶ 179] | x | The AlmFnct function block records the event of the limit position switch monitoring.<br>Actions that are to take place after the limit switch fault is received can be parameterized in the template on the AlmFnct function block. |
| Alm | FB_BACnetBV1204 [▶ 94] | x | BV object for displaying the damper fault in the MCL |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_Damper_LimitSwitch_Open | BOOL | X | Input | Digital input - switch damper open - message - actuated/not actuated |
| BI_Damper_LimitSwitch_Close | BOOL | X | Input | Digital input - switch damper closed - message - actuated/not actuated |
| BI_Damper_LocalSwitch | BOOL | X | Input | Digital input - switch manual damper - message - manual/automatic |
| BI_Feedback_Binary_Output | BOOL | X | Input | Digital input - damper switching command - feedback - on/off |
| BO_Damper | BOOL | | Output | Digital output - damper - switching command - on/off |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.32 BAC_AC_OuADmp2P_01_xx

### Functional description

The sub template **BAC_AC_OuADmp2P_01_xx** is used for control and monitoring of an outside air damper with spring return actuator and limit switch monitor. The damper is controlled via a binary output.

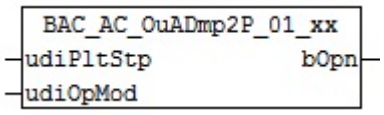
The output variable bOpn outputs the actual state of the damper, if feedback from limit switch SwiOpn is available in the template used. If no end position monitoring is available, it is emulated internally. If the damper is controlled, output bOpn becomes TRUE. As a result, in the start program for a ventilation system BAC_AC_StartT_01 [▶ 530], the delay time for starting the fans (SuAFanDlyOn / ExtAFanDlyOn) must be adjusted to the time it takes for the damper to open.

The template **BAC_AC_OuADmp2P_01_xx** is available in different versions.
The damper versions are identified by means of a key. The identification key is derived from the table below.

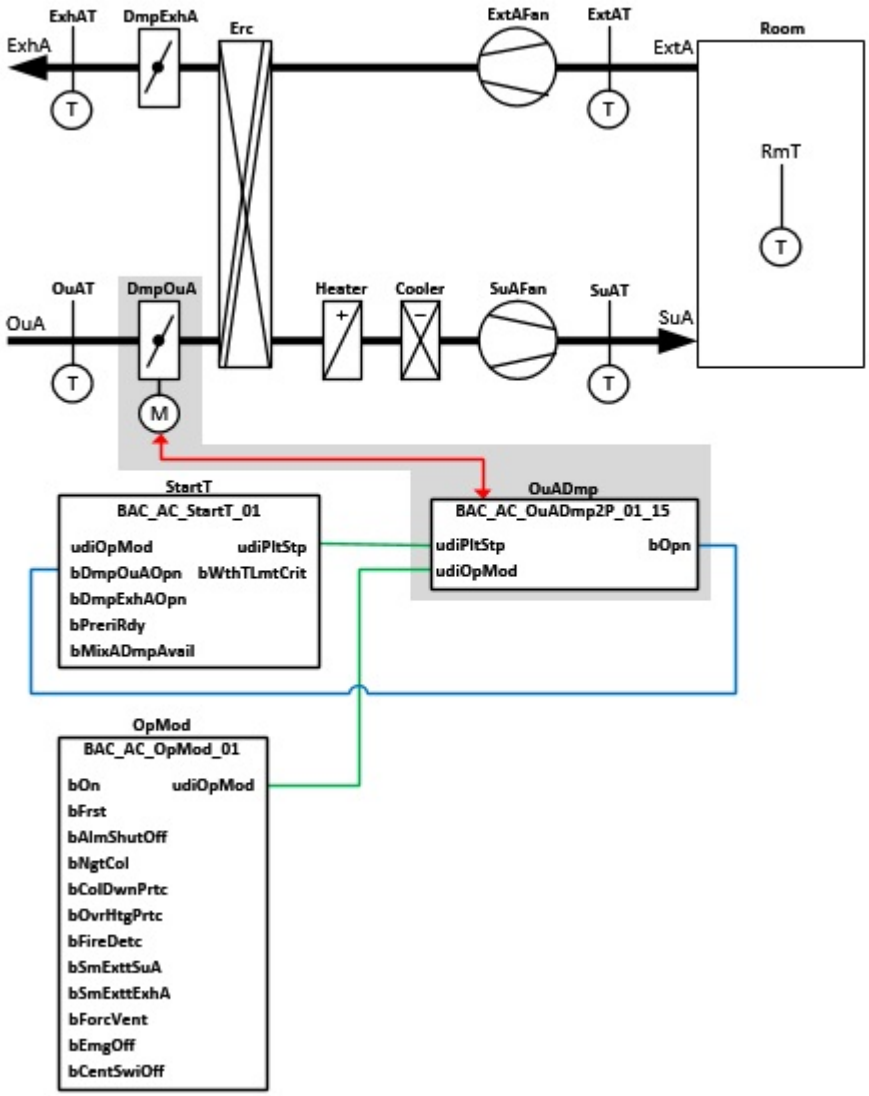
| Options | mechanical priority operation Feedback hand switch | mechanical priority operation Feedback Relay output | Final position Open | Final position Close |
|---|---|---|---|---|
| **Instance** | **LocSwi** | **FdbOut** | **SwiOpn** | **SwiCls** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** |
| | **8** | **4** | **2** | **1** |
| BAC_AC_OuADmp2P_01_00 | 0 | 0 | 0 | 0 |

| Options | mechanical priority operation Feedback hand switch | mechanical priority operation Feedback Relay output | Final position Open | Final position Close |
|---|---|---|---|---|
| **Instance** | **LocSwi** | **FdbOut** | **SwiOpn** | **SwiCls** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** |
| | **8** | **4** | **2** | **1** |
| BAC_AC_OuADmp2P_01_02 | 0 | 0 | 0 | 0 |
| BAC_AC_OuADmp2P_01_03 | 0 | 0 | 1 | 1 |
| BAC_AC_OuADmp2P_01_08 | 1 | 0 | 0 | 0 |
| BAC_AC_OuADmp2P_01_10 | 1 | 0 | 1 | 0 |
| BAC_AC_OuADmp2P_01_11 | 1 | 0 | 1 | 1 |
| BAC_AC_OuADmp2P_01_12 | 1 | 1 | 0 | 0 |
| BAC_AC_OuADmp2P_01_14 | 1 | 1 | 1 | 0 |
| BAC_AC_OuADmp2P_01_15 | 1 | 1 | 1 | 1 |

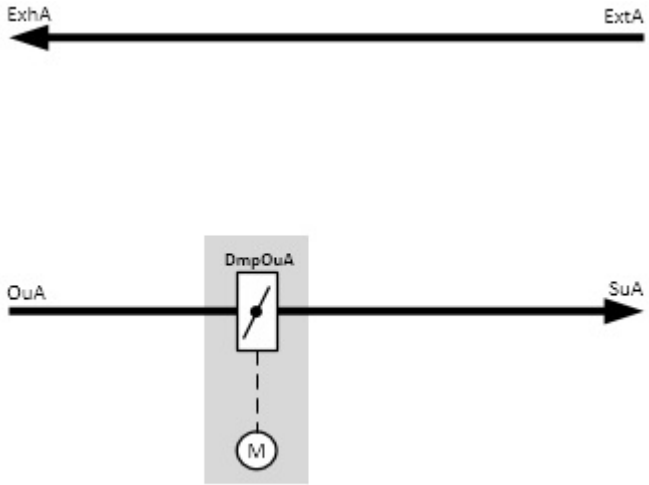
**Interface**



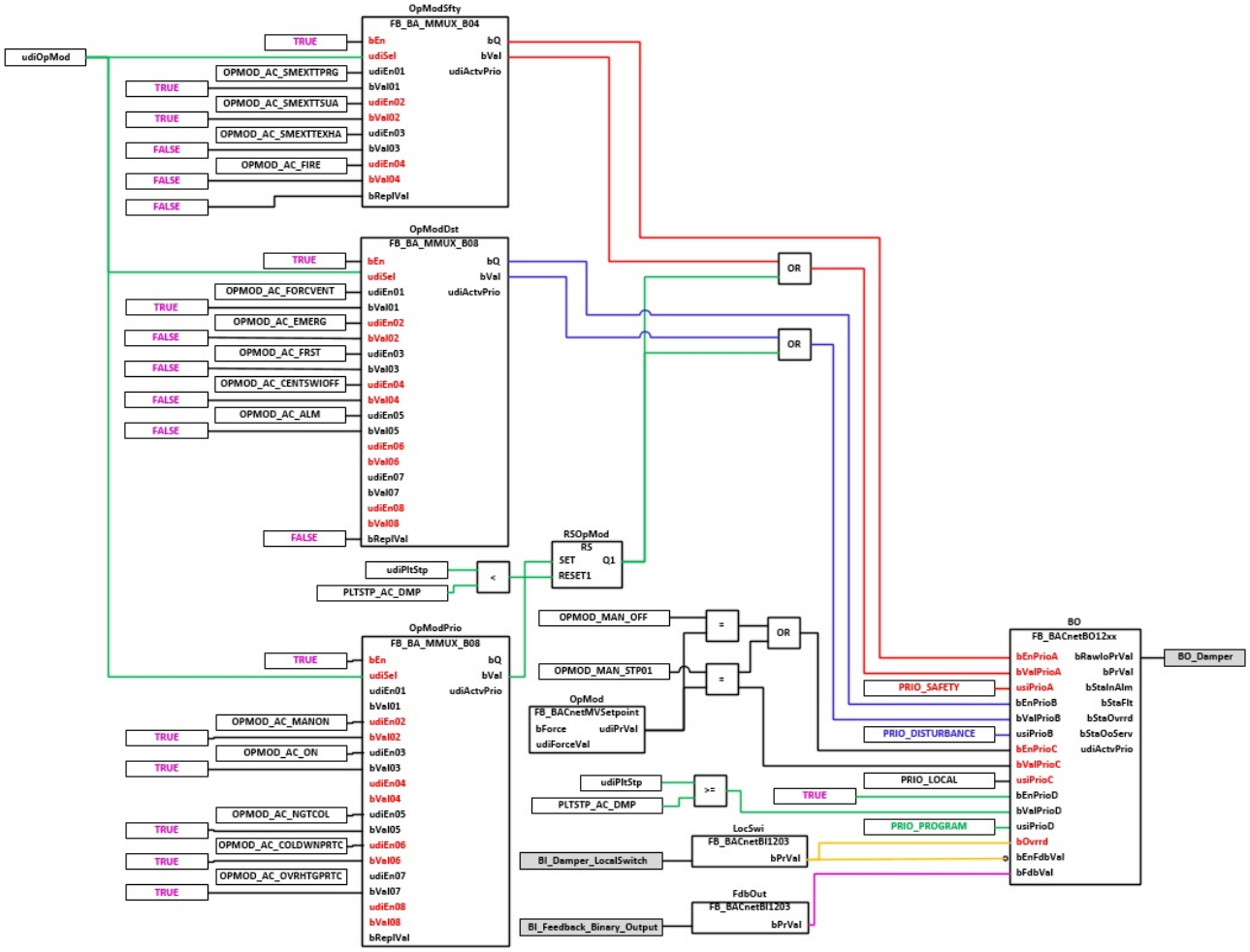
### Plant diagram 01



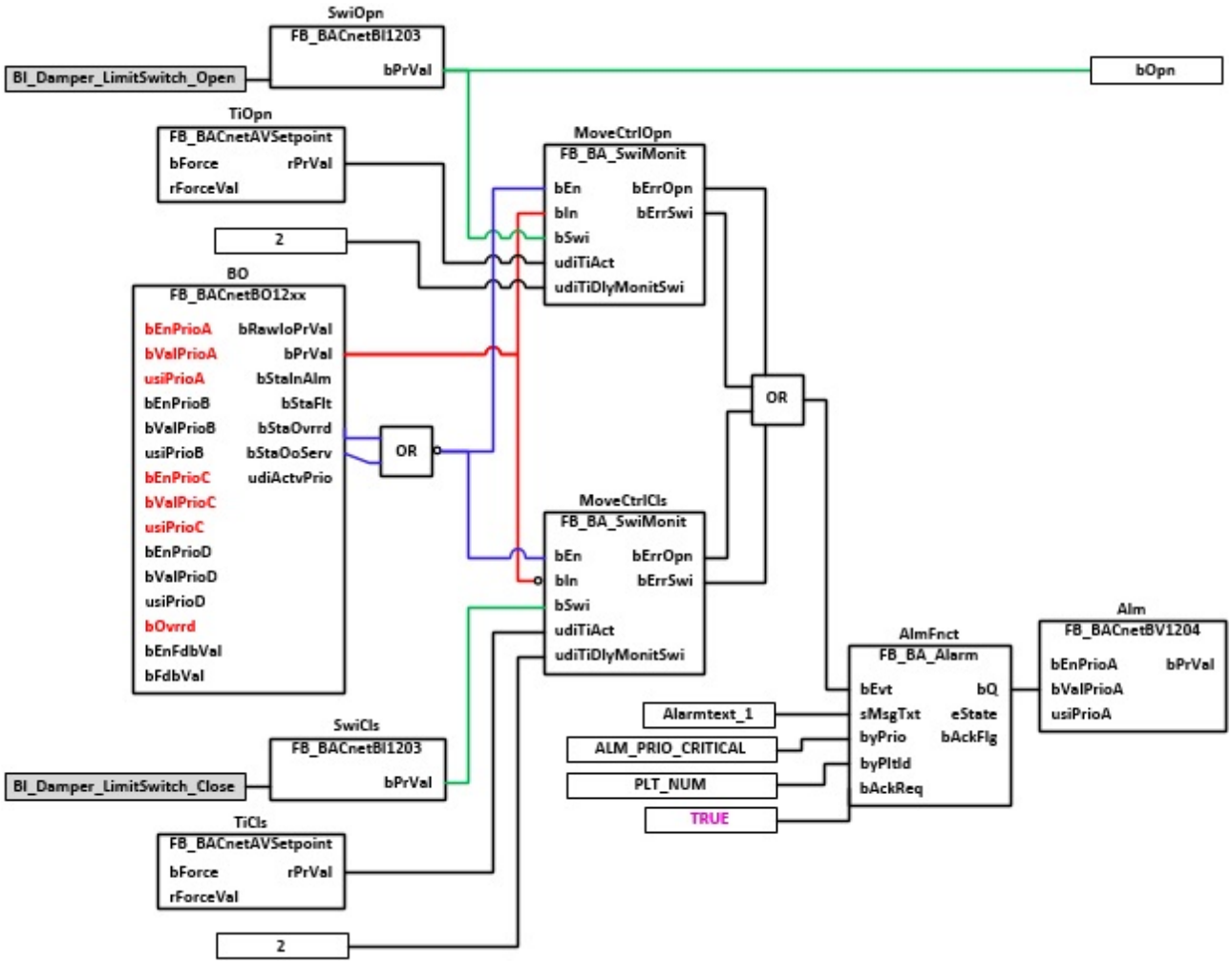

### Plant diagram 02



### Block diagram for controlling the damper



**Block diagram for monitoring the limit switches**




## VAR_INPUT

```
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**udiOpMode:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

## VAR_OUTPUT

```
bOpn        : BOOL;
```

**bOpn:** End position 'open' of the damper has been reached.
If no end position monitoring is available, it is emulated internally. If the damper is controlled, output **bOpn** becomes TRUE.

## VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| SwiOpn | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'open' |
| SwiCls | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'closed' |
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the damper via the MCL or a local operator display |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object for feedback from mechanical priority operation. (manual/emergency operating level) |
| FdbOut | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation position feedback relay |
| TiOpn | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the opening time value |
| TiCls | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the closing time value |
| OpModSfty | FB_BA_MMUX_B04 [▶ 205] | | The multiplexer defines the safety priority of the outside air damper depending on the plant operation mode **udiOpMod** |

| **udiOpMod** | | **Damper state** |
|---|---|---|
| OPMOD_AC_SMEXTTPRG | Smoke extraction program | Open |
| OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | Close |
| OPMOD_AC_SMEXTTEXHA | Smoke extraction outside air | Open |
| OPMOD_AC_FIRE | Fire | Close |

| OpModDst | FB_BA_MMUX_B08 [▶ 205] | | The multiplexer defines the disturbance priority of the outside air damper depending on the plant operation mode **udiOpMod** |
|---|---|---|---|

| **udiOpMod** | | **Damper state** |
|---|---|---|
| OPMOD_AC_FORCVENT | Forced ventilation | Open |
| OPMOD_AC_EMERG | emergency | Close |
| OPMOD_AC_FRST | Frost | Close |
| OPMOD_AC_CENTSWIOFF | Central shutdown | Close |
| OPMOD_AC_ALM | Fault | Close |

| OpModPrio RSOpMod | FB_BA_MMUX_B08 [▶ 205] RS | | The multiplexer defines the plant operation mode **udiOpMod**. The function **RSOpMod** is intended to prevent the damper closing immediately if the operation mode changes while the process/fans are running. The start program is designed to shut down and restart the plant when the operation mode changes. |
|---|---|---|---|

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| | | | **udiOpMod** | | **Damper state** |
| | | | OPMOD_AC_MANON | Manual on | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_ON | On | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_NGTCOL | Night cooling | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_COLDWNPRTC | Support operation, cooling protection | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| | | | OPMOD_AC_OVRHTGPRTC | Overheating protection | Damper does not close until the plant step udiPltStp < PLTSTP_AC_DMP if no longer required |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for controlling the damper | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | see multiplexer **OpModSftyudiOpMod** | see multiplexer **OpModSfty** Damper state |
| | | | PRIO_DISTURBANCE (3) | see multiplexer **OpModDstudiOpMod** | see multiplexer **OpModDst** Damper state |
| | | | PRIO_LOCAL (8) | The OR module pools events which enable writing to the priority manual override (local) of the downstream BO object. Events: 1. The MV object has the value **OPMOD_MAN_OFF** (manual off) 2. The MV object has the value **OPMOD_MAN_STP01**(manual on) | Open, if **OpMod_udiPrVal** = **OPMOD_MAN_STP01** Otherwise close |

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| | | | PRIO_PROGRAM (15) | TRUE | If the plant step is **udiPltStp** >= **PLTSTP_AC_DMP**, the damper should open |
| MoveCtrlOpn | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'open' end position of the damper | | |
| MoveCtrlCls | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'closed' end position of the damper | | |
| AlmFnct | FB_BA_Alarm [▶ 179] | x | The AlmFnct function block records the event of the limit position switch monitoring.<br>Actions that are to take place after the limit switch fault is received can be parameterized in the template on the **AlmFnct** function block. | | |
| Alm | FB_BACnetBV1204 [▶ 94] | x | BV object for displaying the damper fault in the MCL | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| BI_Damper_LimitSwitch_Open | BOOL | X | Input | Digital input - switch damper open - message - actuated/not actuated |
| BI_Damper_LimitSwitch_Close | BOOL | X | Input | Digital input - switch damper closed - message - actuated/not actuated |
| BI_Damper_LocalSwitch | BOOL | X | Input | Digital input - switch manual damper - message - manual/automatic |
| BI_Feedback_Binary_Output | BOOL | X | Input | Digital input - damper switching command - feedback - on/off |
| BO_Damper | BOOL | | Output | Digital output - damper - switching command - on/off |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.33 BAC_AC_ColH_PID_01

### Functional description

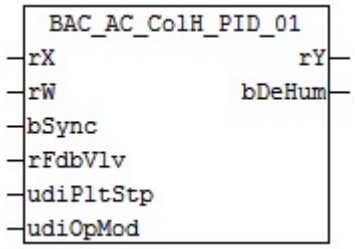
The subtemplate **BAC_AC_ColH_PID_01** is the dehumidification controller for a cooler with dehumidification.
The set value, actual value and control output are referenced via the BACnet value objects **X**, **W** and **Y**.

The PID sequence controller is enabled based on the plant operation mode **udiOpMod** and the global humidity communication structure **g_stSeqLinkH[PLT_NUM]**. This data and command structure is the link between the individual sequence controllers and the corresponding control function block FB_BA_SeqLink [▶ 168] of a plant.

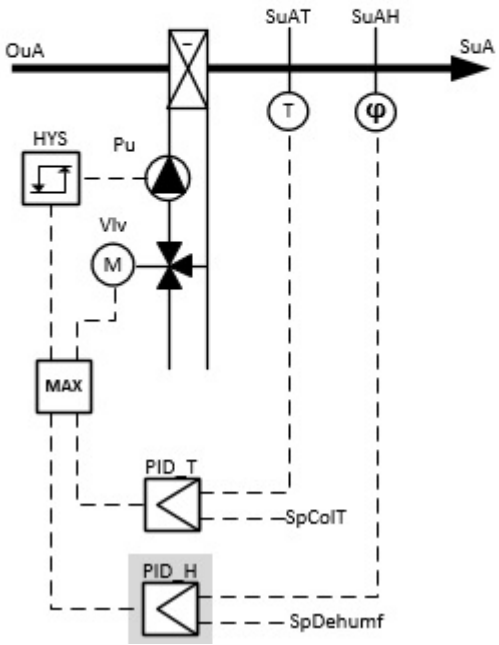
The BACnet BV object **En** is used to display the controller enable.

The limit value monitoring of PID controller is controlled by the function block **EnEvtEn** as a function of the plant startup process.

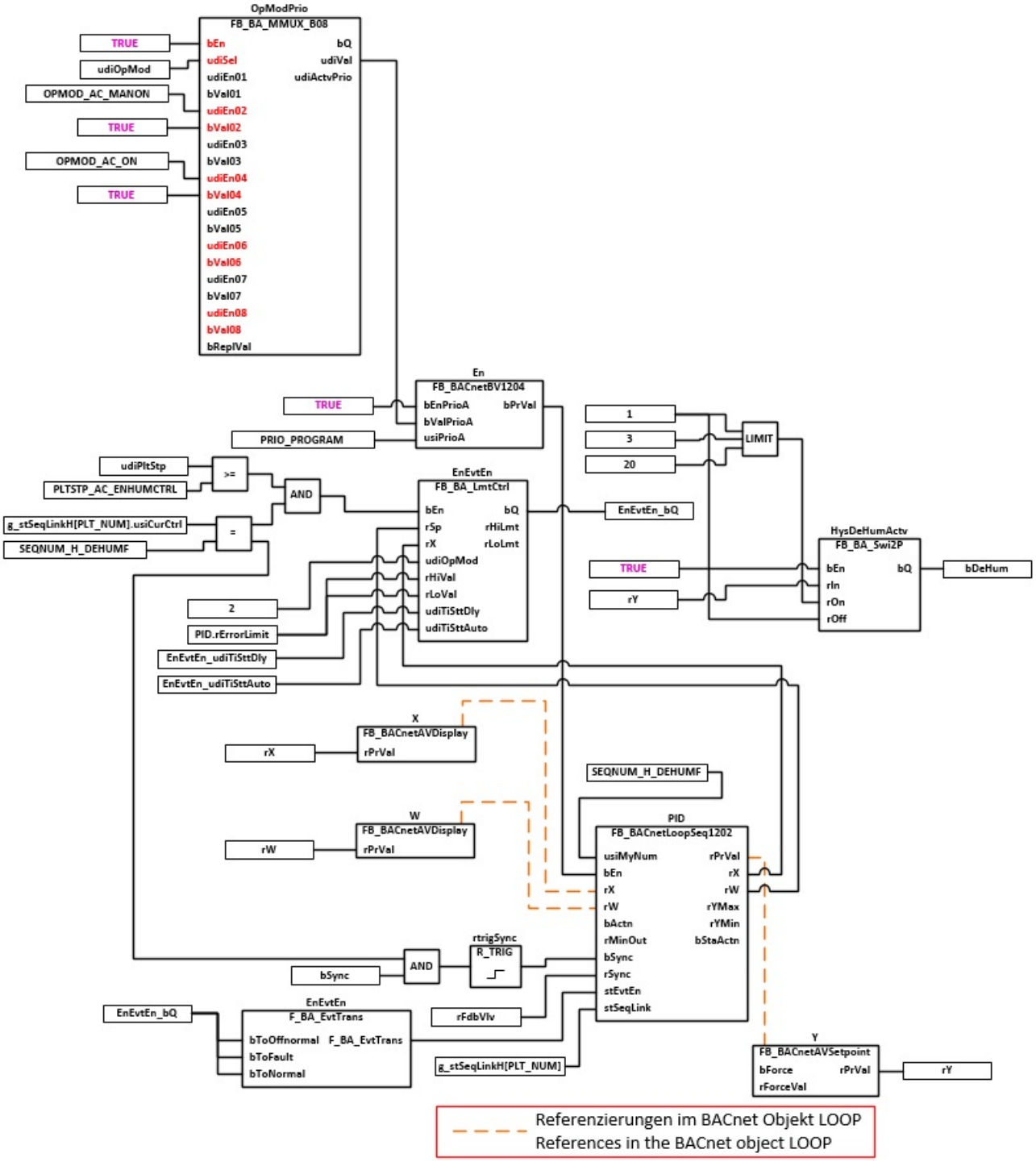
### Interface




### System diagram

### Block diagram




### VAR_INPUT

```
rX          : REAL;
rW          : REAL;
bSync       : BOOL;
rFdbVlv     : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rX:** Measured value supply air humidity

**rW:** Set value of the supply air humidity

**bSync:** Input for synchronization of the controller

**rFdbVlv:** Position feedback actuator

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartTH_01 [▶ 535].

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rY    : REAL;
```

**rY:** Output of the control value for the control valve

**bDeHum:** Dehumidification mode active.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkH[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task | | | |
|---|---|---|---|---|---|
| X | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the actual value input of the BACnet loop object | | | |
| W | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the setpoint input of the BACnet loop object | | | |
| OpModPrio | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the enable conditions of the sequence controller depending on the plant operation mode. | | | |
| | | **udiOpMod** | | **Enable** | **Comment** |
| | | OPMOD_AC_MANON [▶ 357] | Manual on | TRUE | The plant is switched on manually via the plant selector switch |
| | | OPMOD_AC_ON [▶ 357] | On | TRUE | The plant runs in automatic mode via the timer program |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display and activate the controller enable in the MCL or in a local operator display.<br>The PID sequence controller is enabled using the plant operation mode **udiOpMod** and the global communication structure **g_stSeqLinkH[PLT_NUM].** | | | |
| EnEvtEn | FB_BA_LmtCtrl [▶ 230] | The BACnet loop object PID monitors the control function by comparing the setpoint **W** with the actual value **X**. If the deviation W-X is greater than the property **ErrorLimit**, then the loop object sends a message to the MCL. | | | |

| Instance | Type | Task |
|---|---|---|
| | | When the plant is at a standstill, at the moment of startup and until the plant is in a controlled state, the loop object is suppressed so that no incorrect messages are sent to the MCL. Reporting of the loop object should not be activated until the air-conditioning system is fully running and the control is stable. In addition, reporting is enabled if the control has not reached the range around the setpoint defined by the property **ErrorLimit** after a long time.<br><br>Enabling the object internal reporting is done by writing to the BACnetEventTransitionBits of the loop object.<br><br>**The following conditions must be met to enable reporting from the loop object:**<br><br>**1.**<br>The plant start program BAC_AC_StartTH_01 [▶ 535] has enabled control monitoring and sensor limit monitoring<br>**udiPltStp** >= **PLTSTP_AC_ENHUMCTRL**<br>and<br>the dehumidifier controller is the active controller in the control sequence. g_stSeqLinkH[PLT_NUM] [▶ 357]**.usiCurCtrl** = **SEQNUM_H_DEHUMF**<br>and<br>the absolute supply air humidity has approached the setpoint so far that it has settled in a range between **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit**.<br>and<br>the absolute supply air humidity must have remained within the range of **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit** for at least the duration of **EnEvtEn_udiTiSttDly**.<br><br>**2.**<br>The timer **EnEvtEn_udiTiSttAuto** has expired and the control has not reached its setpoint range. |
| | F_BA_EvtTrans | writes to the **BACnetEventTransitionBits** to_offnormal, to_fault and to_normal of the loop object. The input **bEnEvtEn** must be TRUE |
| PID | FB_BACnetLoopSeq1202 [▶ 102] | Sequence controller absolute supply air humidity cooler. |
| rtrigSync | R_TRIG | A rising edge at input **bSync** triggers synchronization of the loop object to the value of **lrSync**.<br>If the control valve of the cooler was overridden by writing a higher priority to the corresponding AO object of the MCL or by activating the local priority operation, the current position of the control valve deviates from the output of the loop object. The variables **bSync** and **rFdbVlv** can be used to restore synchronicity between the position of the control valve and the controller. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |
| HysDeHumActv | FB_BA_Swi2P [▶ 146]<br>LIMIT | Two-point switch, which switches the output **bDeHum** on/off, depending on the control value **rY**, based on a hysteresis (5). If the control value is < 2, **bDeHum** is switched off.<br>The function **LIMIT** limits the upper limit value for the two-point switch **HysDeHumActv**. |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.34 BAC_AC_ColT_01

### Application

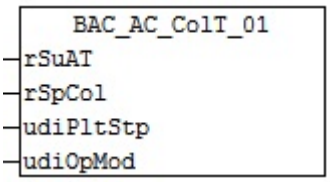
The template **BAC_AC_ColT_01** is used for control of a cold water air cooler without dehumidification control.

The template **BAC_AC_ColT_01** is a call template.

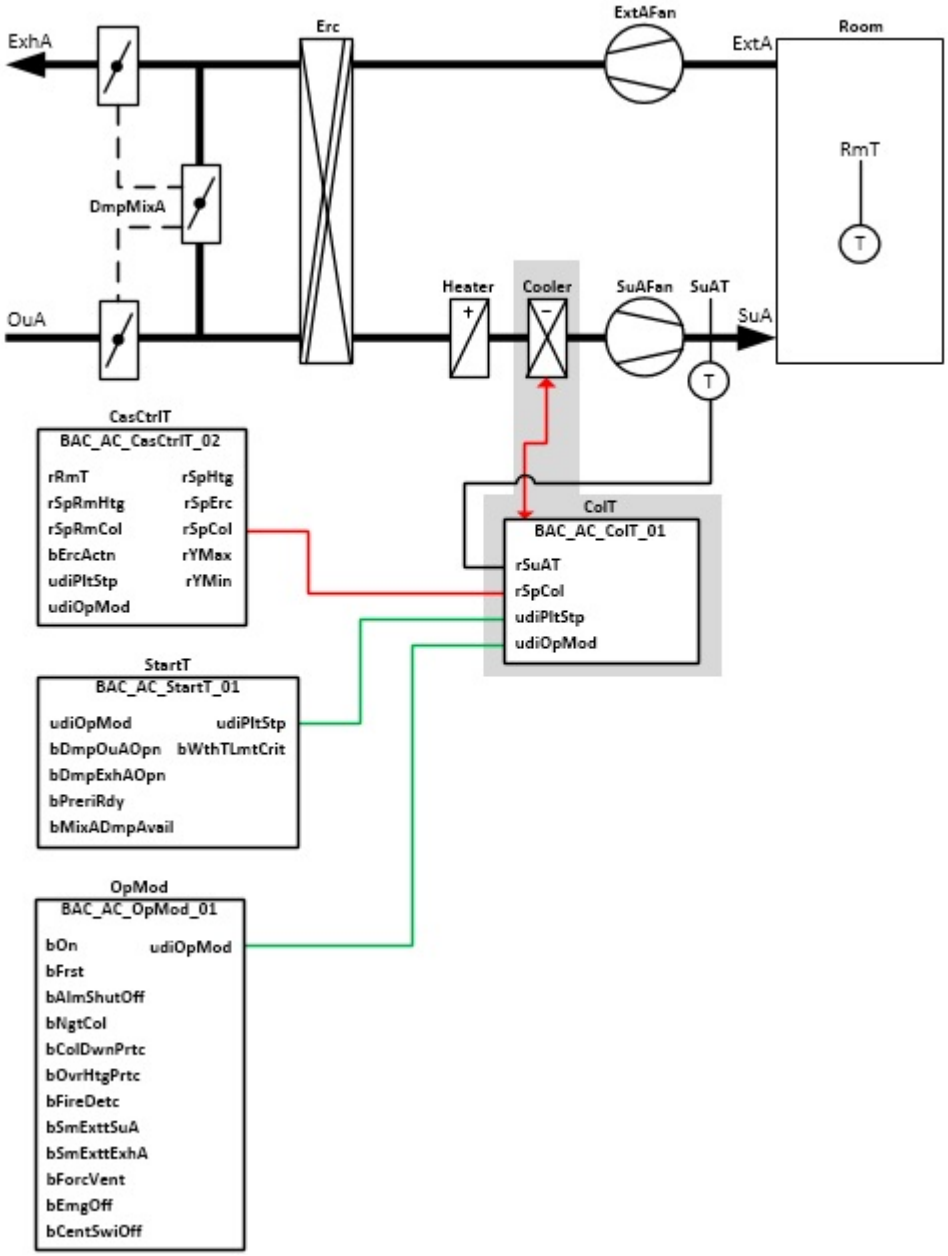
Within the call template the following subtemplates are called and linked with each other:

- **ColTPID** control of supply air temperature
- **ColTPu** control of the cooler pump
- **ColTVlv** control of an analog control valve

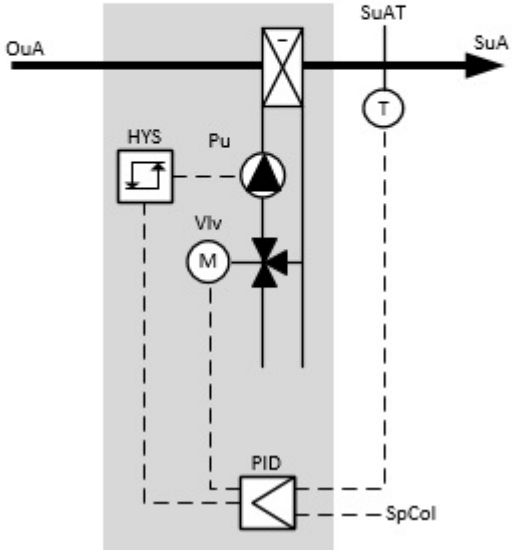
### Interface



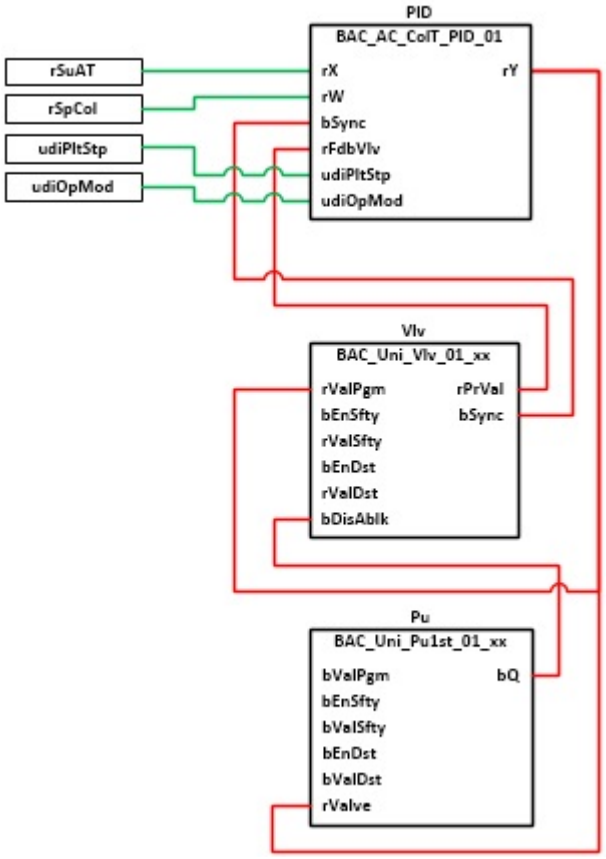
**Plant diagram 01**



### Plant diagram 02


### Block diagram



### VAR_INPUT

```
rSuAT       : REAL;
rSpCol      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rSuAT:** Measured value of the supply air temperature

**rSpCol:** Set value for the supply air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### Program description

| Instance | Type | Task |
|---|---|---|
| PID | BAC_AC_ColT_PID_01 [▶ 463] | Sub template including PID sequence controller for supply air temperature control. |
| Vlv | BAC_Uni_Vlv_01_13 [▶ 598] | Sub template for controlling an analog valve |
| Pu | BAC_Uni_Pu1st_01_189 [▶ 571] | Sub template for controlling a single-stage pump |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.35 BAC_AC_ColT_02

### Application

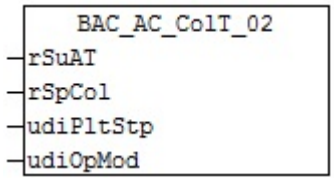
The template **BAC_AC_ColT_02** is used for control of a cold water air cooler without dehumidification control.

The template **BAC_AC_ColT_02** is a call template.

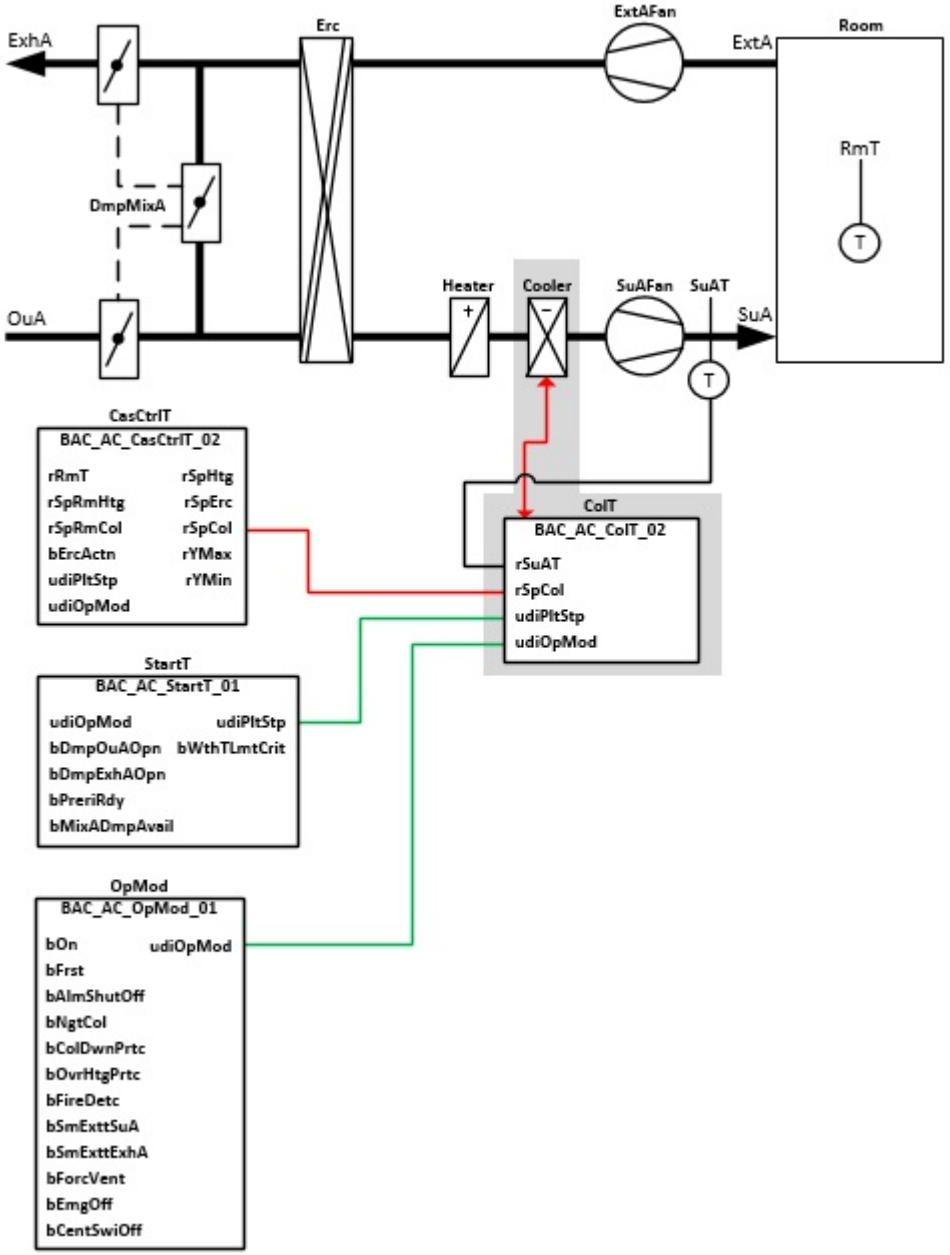
Within the call template the following subtemplates are called and linked with each other:

- **ColTPID:** control of the supply air temperature
- **ColTVlv:** control of an analog control valve

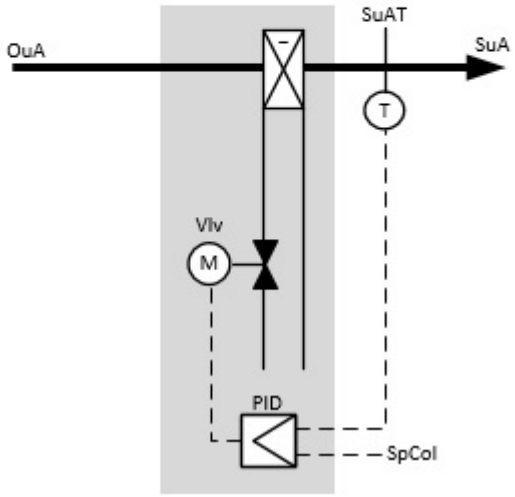
### Interface



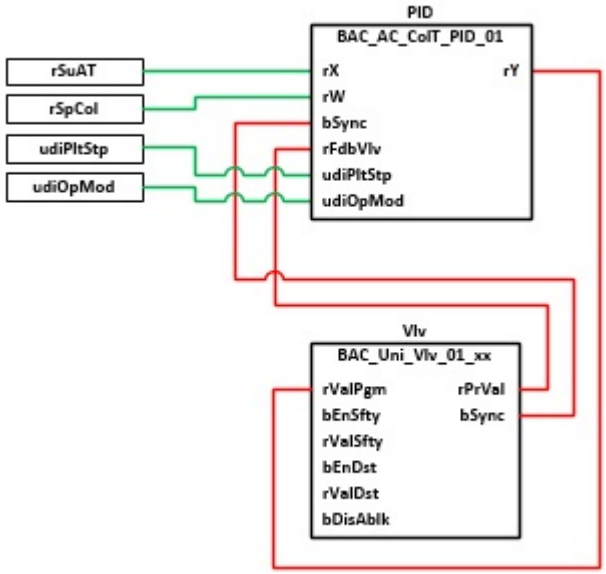
**Plant diagram 01**



### Plant diagram 02


### Block diagram




### VAR_INPUT

```
rSuAT      : REAL;
rSpCol     : REAL;
udiPltStp  : UDINT;
udiOpMod   : UDINT;
```

**rSuAT:** Measured value of the inlet air temperature

**rSpCol:** Set value for the inlet air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### Program description

| Instance | Type | Task |
|---|---|---|
| PID | BAC_AC_ColT_PID_01 [▶ 463] | Subtemplate including the PID sequence controller for the control of the supply air temperature. |
| Vlv | BAC_Uni_Vlv_01_13 [▶ 598] | Subtemplate for controlling an analog valve |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.36 BAC_AC_ColT_PID_01

### Functional description

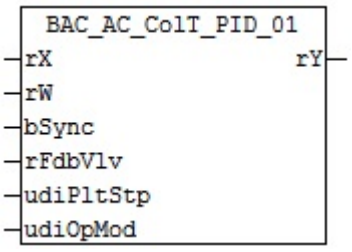
The sub template **BAC_AC_ColT_PID_01** is the sequence controller for the supply air temperature control of a cooler.
The set value, actual value and control output are referenced via the BACnet value objects **X**, **W** and **Y**.

The PID sequence controller is enabled based on the plant operation mode **udiOpMod** and the global temperature communication structure **g_stSeqLinkT[PLT_NUM]**. This data and command structure is the link between the individual sequence controllers and the corresponding control function block FB_BA_SeqLink [▶ 168] of a plant.

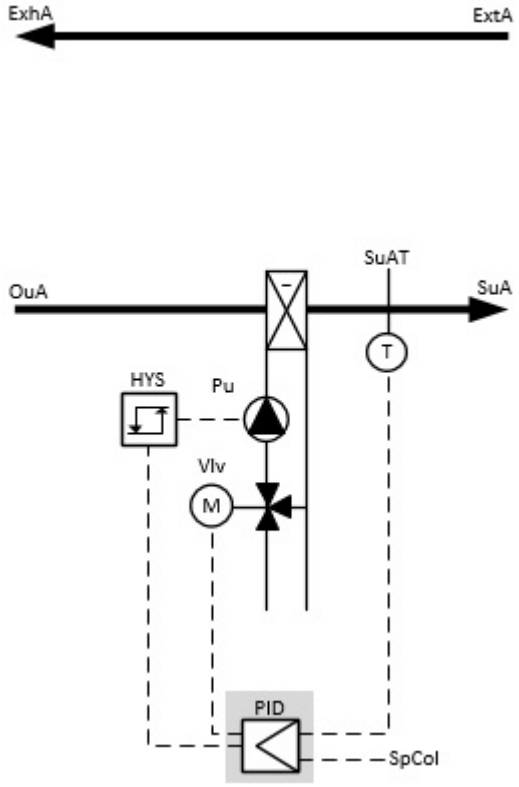
The BACnet BV object **En** is used to display the controller enable.

The limit value monitoring of PID controller is controlled by the function block **EnEvtEn** as a function of the plant startup process.

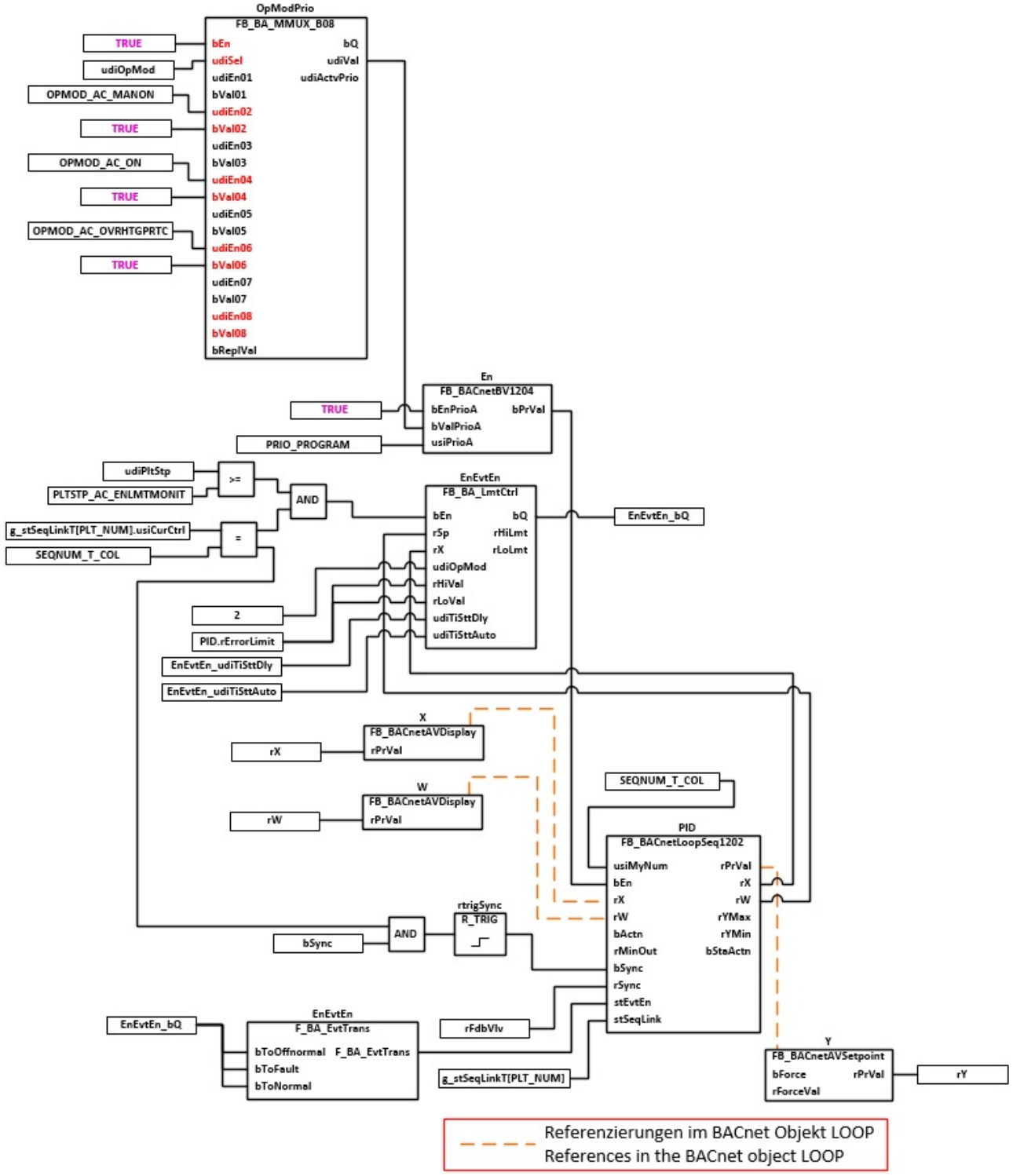
### Interface



**System diagram**



### Block diagram




### VAR_INPUT

```
rX          : REAL;
rW          : REAL;
bSync       : BOOL;
rFdbVlv     : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rX:** Measured value supply air temperature

**rW:** Set value of the supply air temperature

**bSync:** Input for synchronisation of the controller

**rFdbVlv:** Position feedback actuator

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rY    : REAL;
```

**rY:** Output of the control value for the control valve

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkT[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

<table>
<tr><th>Instance</th><th>Type</th><th colspan="3">Task</th></tr>
<tr><td>X</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="3">The AV object is referenced to the actual value input of the BACnet loop object</td></tr>
<tr><td>W</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="3">The AV object is referenced to the setpoint input of the BACnet loop object</td></tr>
<tr><td>OpModPrio</td><td>FB_BA_MMUX_B08 [▶ 205]</td><td colspan="3">The multiplexer defines the enable conditions of the sequence controller depending on the plant operation mode.</td></tr>
<tr><td rowspan="4"></td><td rowspan="4"></td><td colspan="2">udiOpMod</td><td>Enable</td><td>Comment</td></tr>
<tr><td>OPMOD_AC_MANON [▶ 357]</td><td>Manual on</td><td>TRUE</td><td>The plant is switched on manually via the plant selector switch</td></tr>
<tr><td>OPMOD_AC_ON [▶ 357]</td><td>On</td><td>TRUE</td><td>The plant runs in automatic mode via the timer program</td></tr>
<tr><td>OPMOD_AC_OVRHTGPRTC [▶ 357]</td><td>Overheating protection</td><td>TRUE</td><td>The plant operates in overheating protection mode</td></tr>
<tr><td>En</td><td>FB_BACnetBV1204 [▶ 94]</td><td colspan="3">The BV object is used to display and activate the controller enable in the MCL or in a local operator display.<br>The PID sequence controller is enabled using the plant operation mode udiOpMod and the global communication structure g_stSeqLinkT.</td></tr>
</table>

| Instance | Type | Task |
|---|---|---|
| EnEvtEn | FB_BA_LmtCtrl [▶ 230] | The BACnet loop object PID monitors the control function by comparing the setpoint **W** with the actual value **X**. If the deviation W-X is greater than the property **ErrorLimit**, then the loop object sends a message to the MCL.<br><br>When the plant is at a standstill, at the moment of startup and until the plant is in a controlled state, the loop object is suppressed so that no incorrect messages are sent to the MCL. Reporting of the loop object should not be activated until the air-conditioning system is fully running and the control is stable. In addition, reporting is enabled if the control has not reached the range around the setpoint defined by the property **ErrorLimit** after a long time.<br><br>Enabling the object internal reporting is done by writing to the **BACnetEventTransitionBits** of the loop object.<br><br>**The following conditions must be met to enable reporting from the loop object:**<br><br>**1.**<br>The plant start program BAC_AC_StartT_01 [▶ 530] has enabled control monitoring and sensor limit monitoring<br>**udiPltStp** >= **PLTSTP_AC_ENLMTMONIT**<br>and<br>the cooler controller is the active controller in the control sequence g_stSeqLinkT[PLT_NUM] [▶ 357].**usiCurCtrl** = **SEQNUM_T_COL**<br>and<br>the supply air temperature has approached the setpoint to a point where it has settled within a range between **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit**<br>and<br>the supply air temperature must have remained within the range of **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit** for at least the duration of **EnEvtEn_udiTiSttDly**.<br><br>**2.**<br>The timer **EnEvtEn_udiTiSttAuto** has expired and the control has not reached its setpoint range. |
| | F_BA_EvtTrans | writes to the **BACnetEventTransitionBits** to_offnormal, to_fault and to_normal of the loop object. The input **bEnEvtEn** must be TRUE |
| PID | FB_BACnetLoopSeq1202 [▶ 110] | Sequence controller supply air temperature cooler. |
| rtrigSync | R_TRIG | A rising edge at input **bSync** triggers synchronization of the loop object to the value of **lrSync**.<br>If the control valve of the cooler was overridden by writing a higher priority to the corresponding AO object of the MCL or by activating the local priority operation, the current position of the control valve deviates from the output of the loop object. The variables **bSync** and **rFdbVlv** can be used to restore synchronicity between the position of the control valve and the controller. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.37 BAC_AC_ColTH_01

### Application

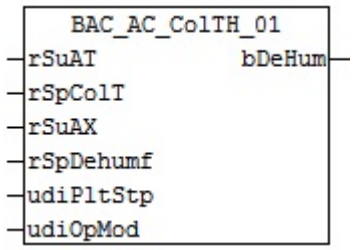
The template **BAC_AC_ColTH_01** is used for control, supply air temperature control and supply air humidity control of a cold water air cooler.

The template **BAC_AC_ColTH_01** is a call template.

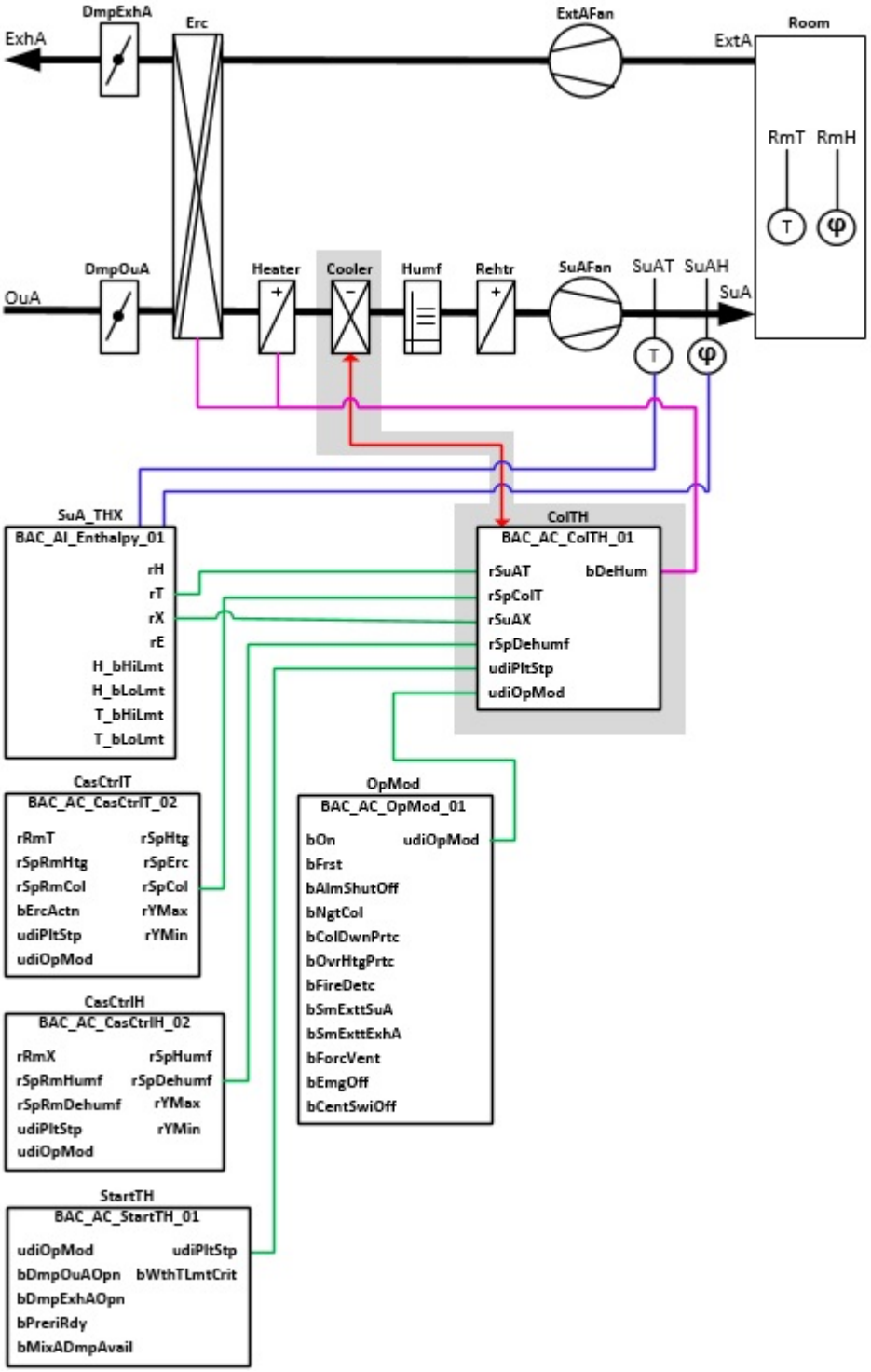
Within the call template the following subtemplates are called and linked with each other:

- **ColTHPID_T:** control of the supply air temperature
- **ColTHPID_H:** control of absolute supply air humidity
- **ColTHPu:** control of the cooler pump
- **ColTHVlv:** control of an analog control valve

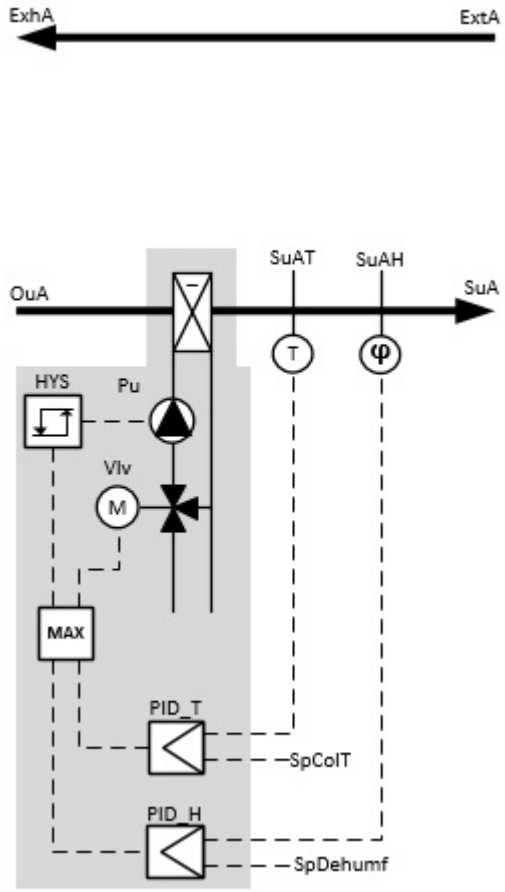
### Interface



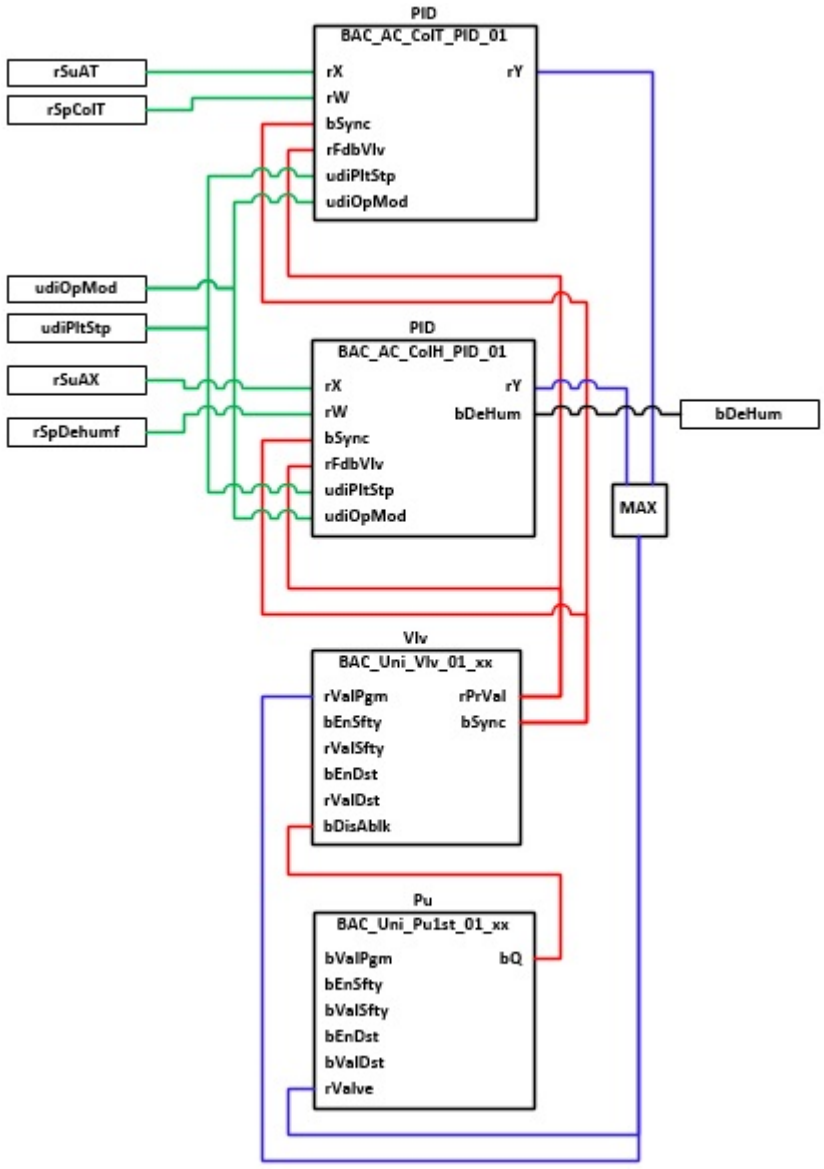
**Plant diagram 01**



**Plant diagram 02**



### Block diagram




### VAR_INPUT

```
rSuAT       : REAL;
rSpColT     : REAL;
rSuAX       : REAL;
rSpDehumf   : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rSuAT:** Measured value of the supply air temperature

**rSpColT:** Set value for the supply air temperature

**rSuAX:** Calculated value of the absolute supply air humidity. This value is generated in template BAC_AI_Enthalpy_01 [▶ 678].

**rSpDehumf:** Set value for the absolute supply air humidity.

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC [▶ 535]StartTH_01.

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### VAR_OUTPUT

```
bDeHum      : BOOL;
```

**bDeHum:** Dehumidification mode active.
If dehumidification mode is active, energy recovery in heating mode and the preheater are disabled.

### Program description

| Instance | Type | Task |
|---|---|---|
| ColTHPID_T | BAC_AC_ColT_PID_01 [▶ 463] | Subtemplate including PID sequence controller for supply air temperature control. |
| ColTHPID_H | BAC_AC_ColH_PID_01 [▶ 452] | Subtemplate including PID sequence controller for absolute supply air humidity control. |
| Vlv | BAC_Uni_Vlv_01_13 [▶ 598] | Subtemplate for controlling an analog valve |
| Pu | BAC_Uni_Pu1st_01_189 [▶ 571] | Subtemplate for controlling a single-stage pump |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.38 BAC_AC_ColTH_02

### Application

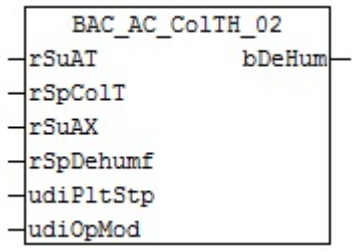
The template **BAC_AC_ColTH_0**2 is used for control, supply air temperature control and supply air humidity control of a cold water air cooler.

The template **BAC_AC_ColTH_02** is a call template.

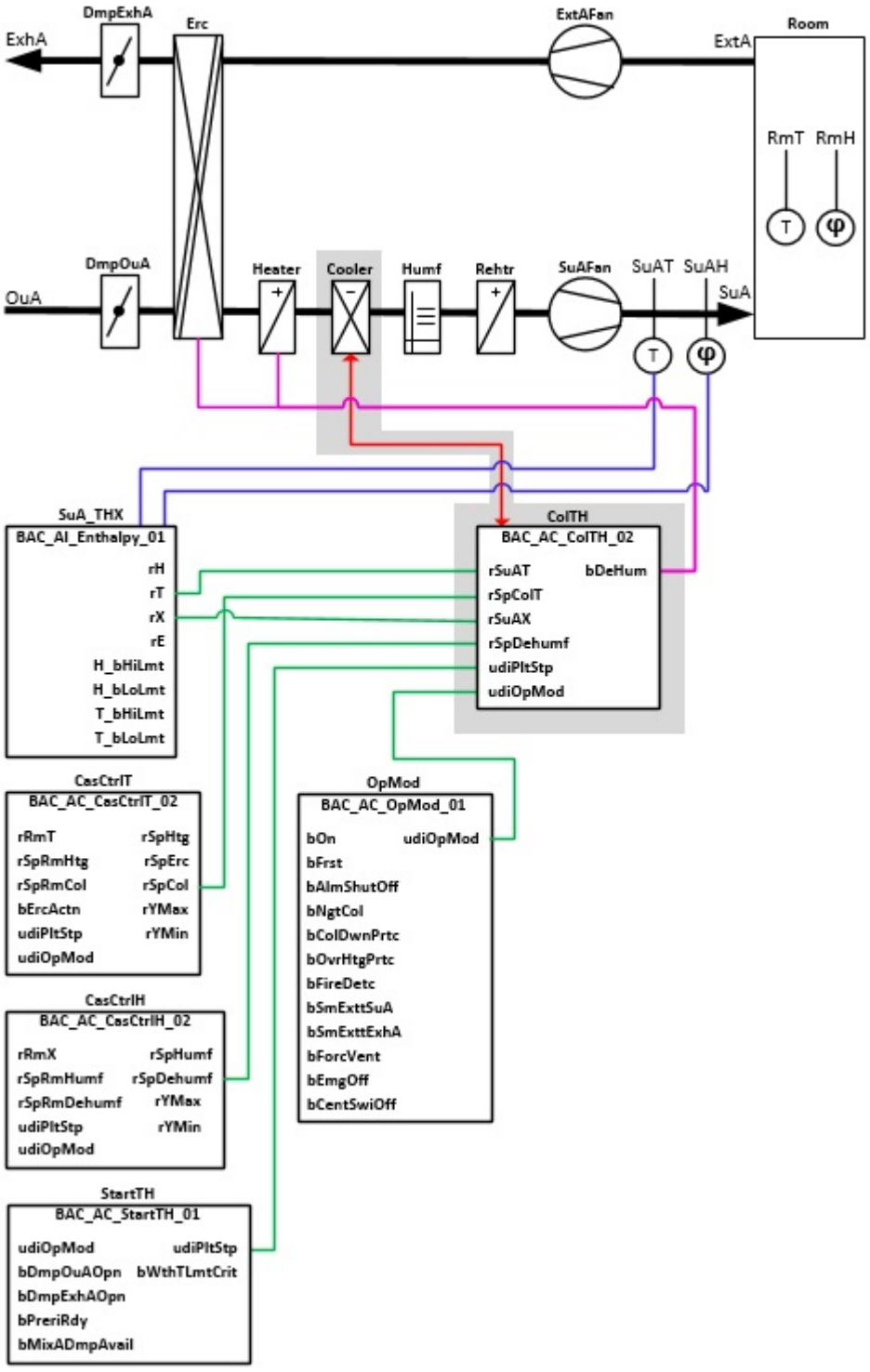
Within the call template the following subtemplates are called and linked with each other:

- **ColTHPID_T:** control of the supply air temperature
- **ColTHPID_H:** control of absolute supply air humidity
- **ColTHVlv:** control of an analog control valve

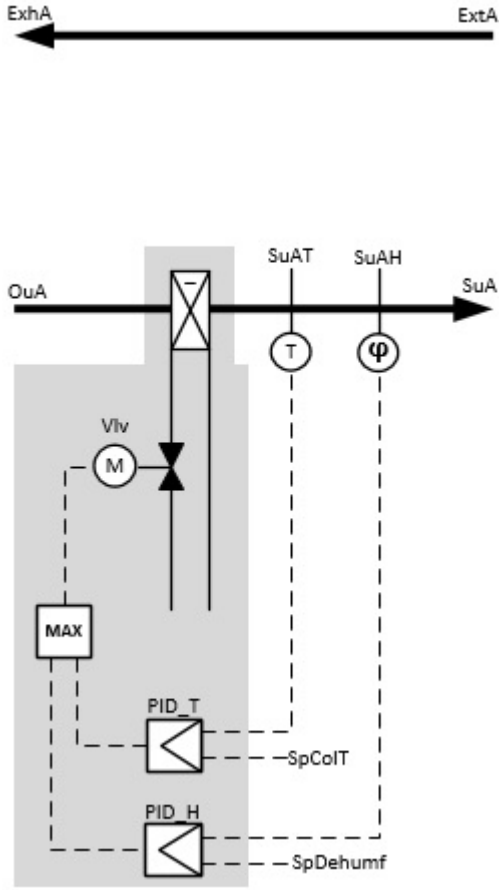
### Interface



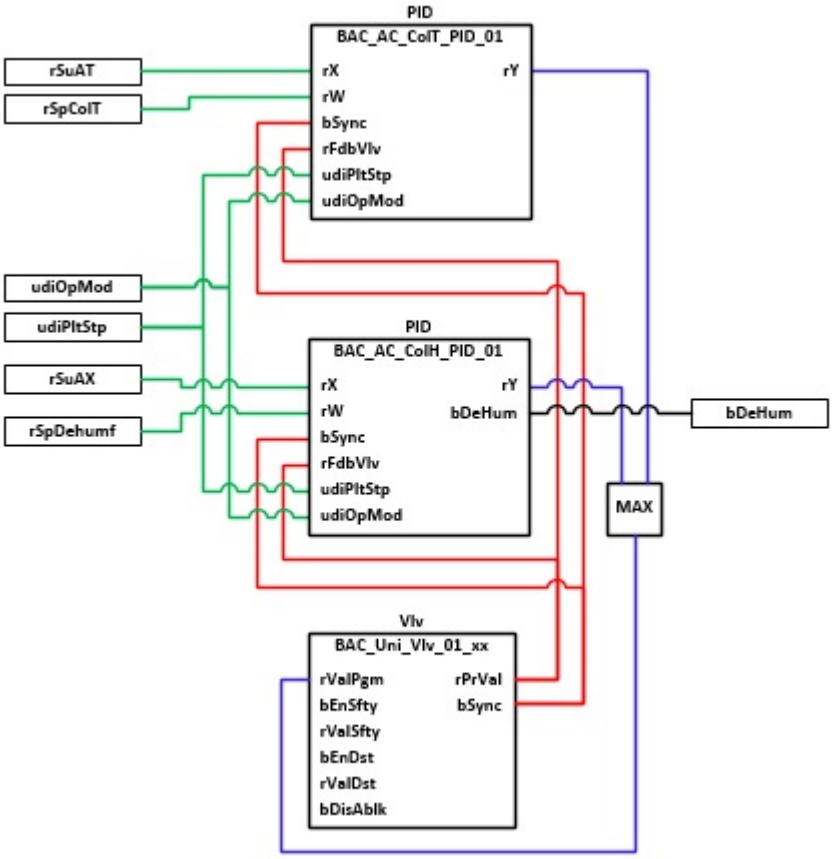
**Plant diagram 01**



### Plant diagram 02




### Block diagram



### VAR_INPUT

```
rSuAT       : REAL;
rSpColT     : REAL;
rSuAX       : REAL;
rSpDehumf   : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rSuAT:** Measured value of the inlet air temperature

**rSpColT:** Set value for the inlet air temperature

**rSuAX:** Calculated value of the absolute inlet air humidity. This value is generated in template BAC_AI_Enthalpy_01 [▶ 678].

**rSpDehumf:** Set value for the absolute inlet air humidity.

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC [▶ 535]StartTH_01.

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### VAR_OUTPUT

```
bDeHum      : BOOL;
```

**bDeHum:** Dehumidification mode active.
If dehumidification mode is active, energy recovery in heating mode and the preheater are disabled.

### Program description

| Instance | Type | Task |
|---|---|---|
| ColTHPID_T | BAC_AC_ColT_PID_01 [▶ 463] | Sub template including PID sequence controller for inlet air temperature control. |
| ColTHPID_H | BAC_AC_ColH_PID_01 [▶ 452] | Sub template including PID sequence controller for absolute inlet air humidity control. |
| Vlv | BAC_Uni_Vlv_01_13 [▶ 598] | Sub template for controlling an analog valve |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

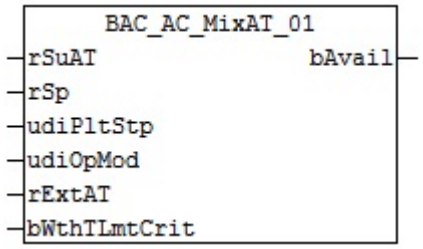
## 9.39 BAC_AC_MixAT_01

### Application

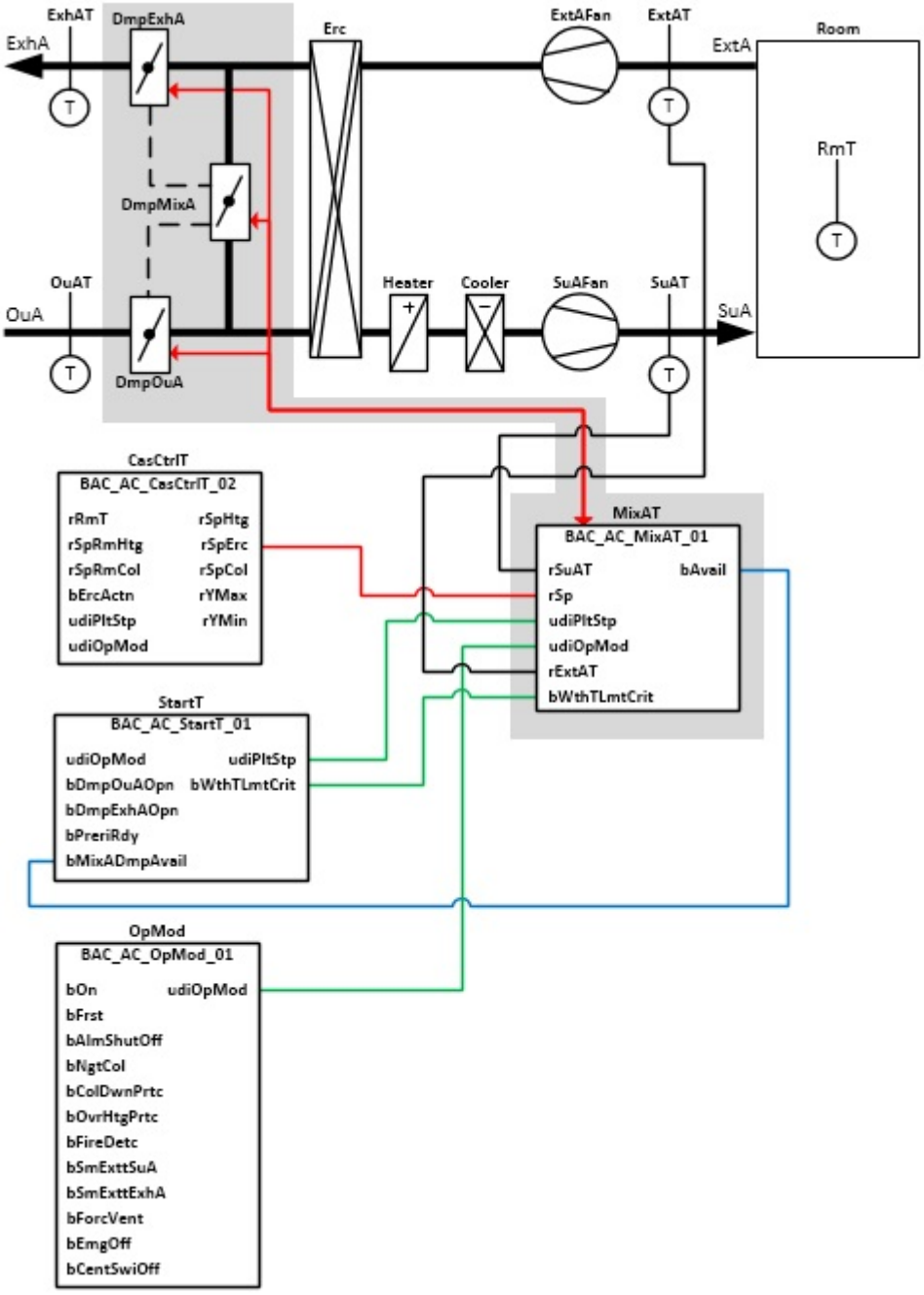
The call template **BAC_AC_MixAT_01** is used to control and regulate a mixed air system.
The template **BAC_AC_MixAT_01** is a call template.
Within the template the following subtemplates are called and linked with each other.

- **MixAT_PID:** supply air temperature controller
- **DmpMixA:** mixed air damper
- **DmpOuA:** outside air damper
- **DmpExhA:** exhaust air damper

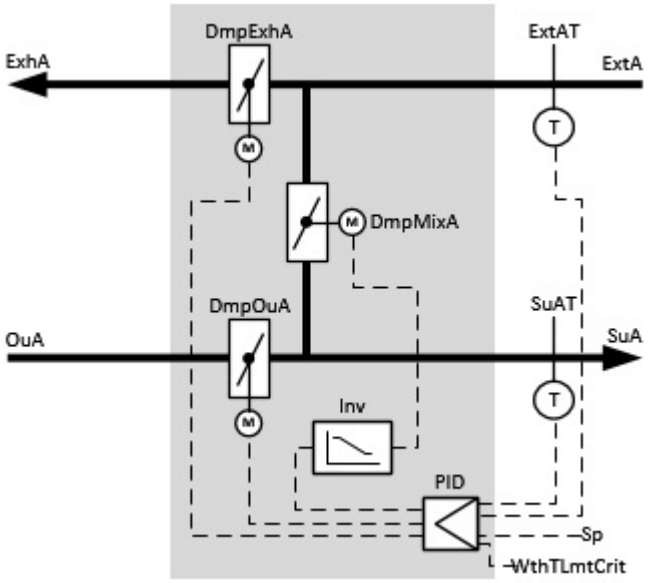
### Interface



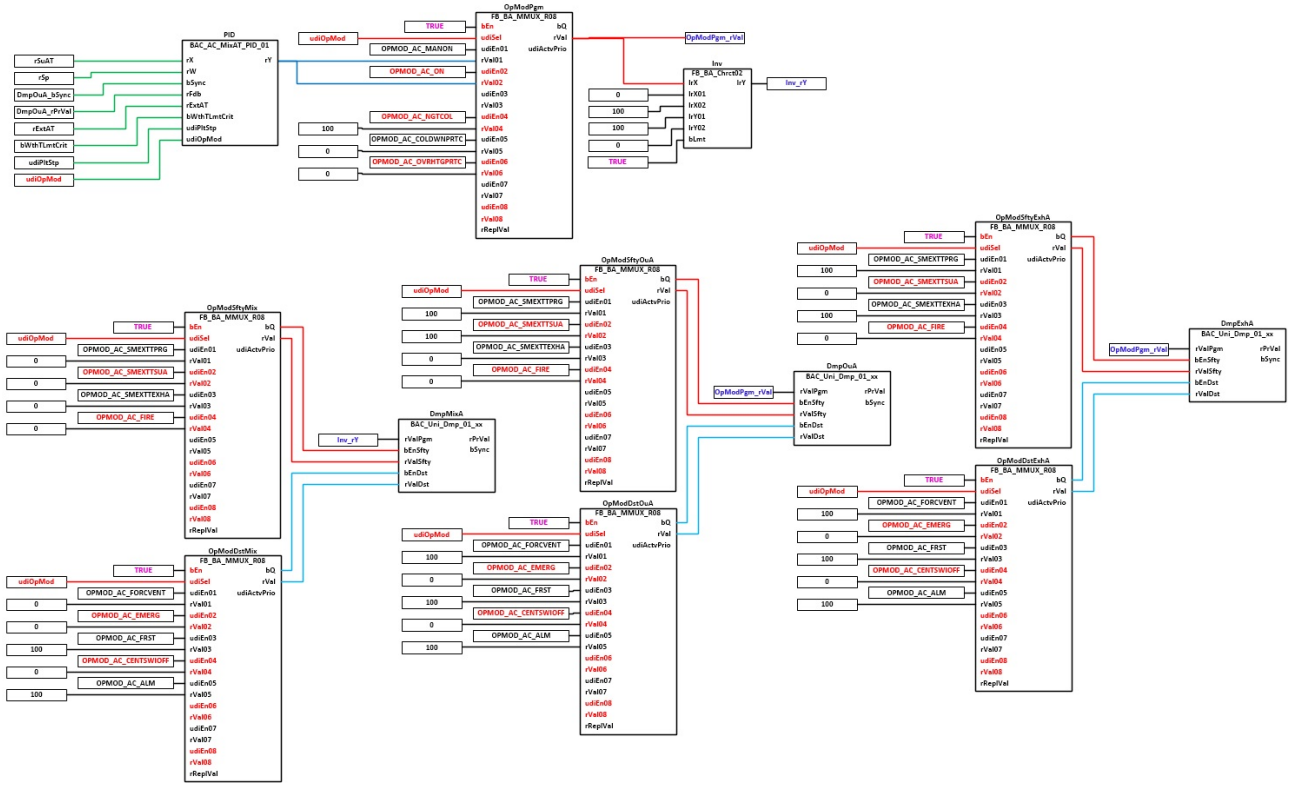

### Plant diagram 01



### Plant diagram 02




### Block diagram



### VAR_INPUT

```
rSuAT         : REAL;
rSp           : REAL;
udiPltStp     : UDINT;
udiOpMod      : UDINT;
rExtAT        : REAL;
bWthTLmtCrit  : BOOL;
```

**rSuAT:** Measured value of the supply air temperature

**rSp:** Set value for the supply air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program[BAC_AC_StartT_01 [▶ 530]].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

**rExtAT:** Measured value outlet air temperature.

**bWthTLmtCrit:** Signal from system start program. Starting in recirculated-air mode via start-up ramp.

### VAR_OUTPUT

```
bAvail      : BOOL;
```

**bAvail:** This output indicates that a mixed-air damper system is present. This variable notifies the system start program that prerinsing of the preheater is skipped, see BAC_AC_StartT_01 [▶ 530].
On startup of the ventilation system, the ramp function in the template **MixAT_PID** replaces prerinsing of the air heater.

### Program description

<table>
<tr><th>Instance</th><th>Type</th><th colspan="4">Task</th></tr>
<tr><td>PID</td><td>BAC_AC_MixAT_PID_01 [▶ 480]</td><td colspan="4">Subtemplate including PID sequence controller for supply air temperature control.</td></tr>
<tr><td>DmpMixA</td><td>BAC_Uni_Dmp_01_05 [▶ 560]</td><td colspan="4">Subtemplate recirculating air damper</td></tr>
<tr><td>DmpOuA</td><td>BAC_Uni_Dmp_01_05 [▶ 560]</td><td colspan="4">Subtemplate outside air damper</td></tr>
<tr><td>DmpExhA</td><td>BAC_Uni_Dmp_01_05 [▶ 560]</td><td colspan="4">Subtemplate exhaust air damper</td></tr>
<tr><td>Inv</td><td>FB_BA_Chrct02 [▶ 171]</td><td colspan="4">Inversion of the control signal of the outside air damper for controlling the mixed air damper.</td></tr>
<tr><td rowspan="7">OpModPgm</td><td rowspan="7">FB_BA_MMUX_R04 [▶ 205]</td><td colspan="4">The multiplexer defines the control value for the dampers in the program priority (automatic operation) via the plant operation mode udiOpMod</td></tr>
<tr><td colspan="2">udiOpMod</td><td>Value</td><td></td></tr>
<tr><td>OPMOD_AC_MANON</td><td>Manual on</td><td>Pgm_rY</td><td>Pgm_rY is a variable, which relays the result of the MIN selection of the mixed air damper controller and the ramp function to the dampers.</td></tr>
<tr><td>OPMOD_AC_AC_ON</td><td>Automatic mode on</td><td>Pgm_rY</td><td>Pgm_rY is a variable, which relays the result of the MIN selection of the mixed air damper controller and the ramp function to the dampers.</td></tr>
<tr><td>OPMOD_AC_NGTCOL</td><td>Night cooling</td><td>100</td><td></td></tr>
<tr><td>OPMOD_AC_COLDWNPRTC</td><td>Cooling protection</td><td>0</td><td></td></tr>
<tr><td>OPMOD_AC_OVRHTGPRTC</td><td>Overheating protection</td><td>0</td><td></td></tr>
<tr><td rowspan="3">OpModSftyMix</td><td rowspan="3">FB_BA_MMUX_R04 [▶ 205]</td><td colspan="4">The multiplexer defines the safety priority for the recirculating air damper via the plant operation mode udiOpMod</td></tr>
<tr><td colspan="2">udiOpMod</td><td colspan="2">Value</td></tr>
<tr><td>OPMOD_AC_SMEXTTPRG</td><td>Smoke extraction program</td><td colspan="2">0</td></tr>
</table>

| Instance | Type | Task | | |
|---|---|---|---|---|
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | 0 |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | 0 |
| | | OPMOD_AC_FIRE | Fire | 0 |
| OpModDistMix | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the fault priority for the recirculating air damper via the plant operation mode *udiOpMod* | | |
| | | **udiOpMod** | | **Value** |
| | | OPMOD_AC_FORCVENT | Forced ventilation | 0 |
| | | OPMOD_AC_EMERG | emergency | 0 |
| | | OPMOD_AC_FRST | Frost | 100 |
| | | OPMOD_AC_CENTSWIOFF | Central shutdown | 0 |
| | | OPMOD_AC_ALM | Fault/alarm | 100 |
| OpModSftyOuA | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the safety priority for the outside air damper via the plant operation mode *udiOpMod* | | |
| | | **udiOpMod** | | **Value** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | 100 |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | 100 |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | 0 |
| | | OPMOD_AC_FIRE | Fire | 0 |
| OpModDistOuA | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the fault priority for the outside air damper via the plant operation mode *udiOpMod* | | |
| | | **udiOpMod** | | **Value** |
| | | OPMOD_AC_FORCVENT | Forced ventilation | 100 |
| | | OPMOD_AC_EMERG | emergency | 0 |
| | | OPMOD_AC_FRST | Frost | 0 |
| | | OPMOD_AC_CENTSWIOFF | Central shutdown | 0 |
| | | OPMOD_AC_ALM | Fault/alarm | 0 |
| OpModSftyExhA | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the safety priority for the exhaust air damper via the plant operation mode *udiOpMod* | | |
| | | **udiOpMod** | | **Value** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | 100 |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | 0 |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | 100 |
| | | OPMOD_AC_FIRE | Fire | 0 |
| OpModDistExhA | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the fault priority for the exhaust air damper via the plant operation mode *udiOpMod* | | |
| | | **udiOpMod** | | **Value** |
| | | OPMOD_AC_FORCVENT | Forced ventilation | 100 |
| | | OPMOD_AC_EMERG | emergency | 0 |

| Instance | Type | Task | | |
|---|---|---|---|---|
| | | OPMOD_AC_FRST | Frost | 0 |
| | | OPMOD_AC_CENTSWIOFF | Central shutdown | 0 |
| | | OPMOD_AC_ALM | Fault/alarm | 0 |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.40 BAC_AC_MixAT_PID_01

### Functional description

The sub template **BAC_AC_MixAT_PID_01** is a supply air temperature sequence controller for energy recovery.

The set value, actual value and control output are referenced via the BACnet value objects **X**, **W** and **Y**.

The PID sequence controller is enabled based on the plant operation mode **udiOpMod** and the global temperature communication structure **g_stSeqLinkT[PLT_NUM]**. This data and command structure is the link between the individual sequence controllers and

the corresponding control function block FB_BA_SeqLink [▶ 168] of a plant.

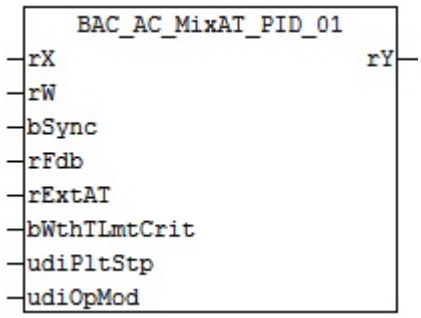
The BACnet BV object **En** is used to display the controller enable.

The limit value monitoring of PID controller is controlled by the function block **EnEvtEn** as a function of the plant startup process.

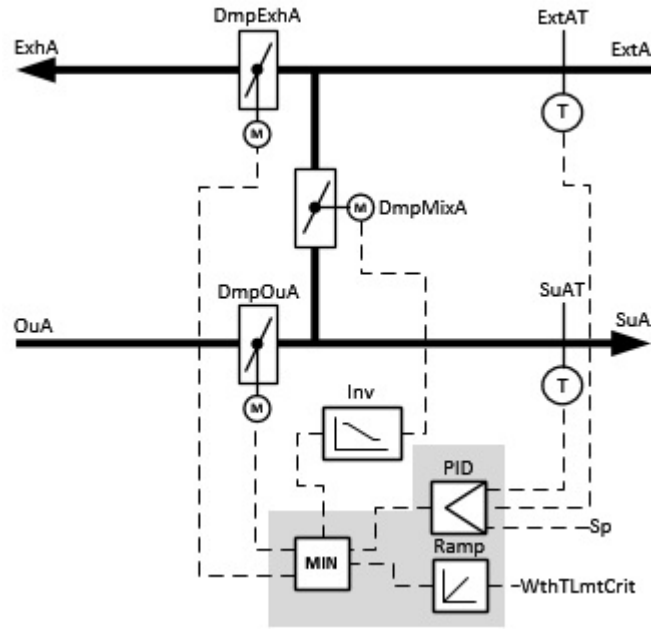
The main tasks of the template are:

- Control of the supply air temperature
- Plant startup through slow opening of the outside and exhaust air damper.

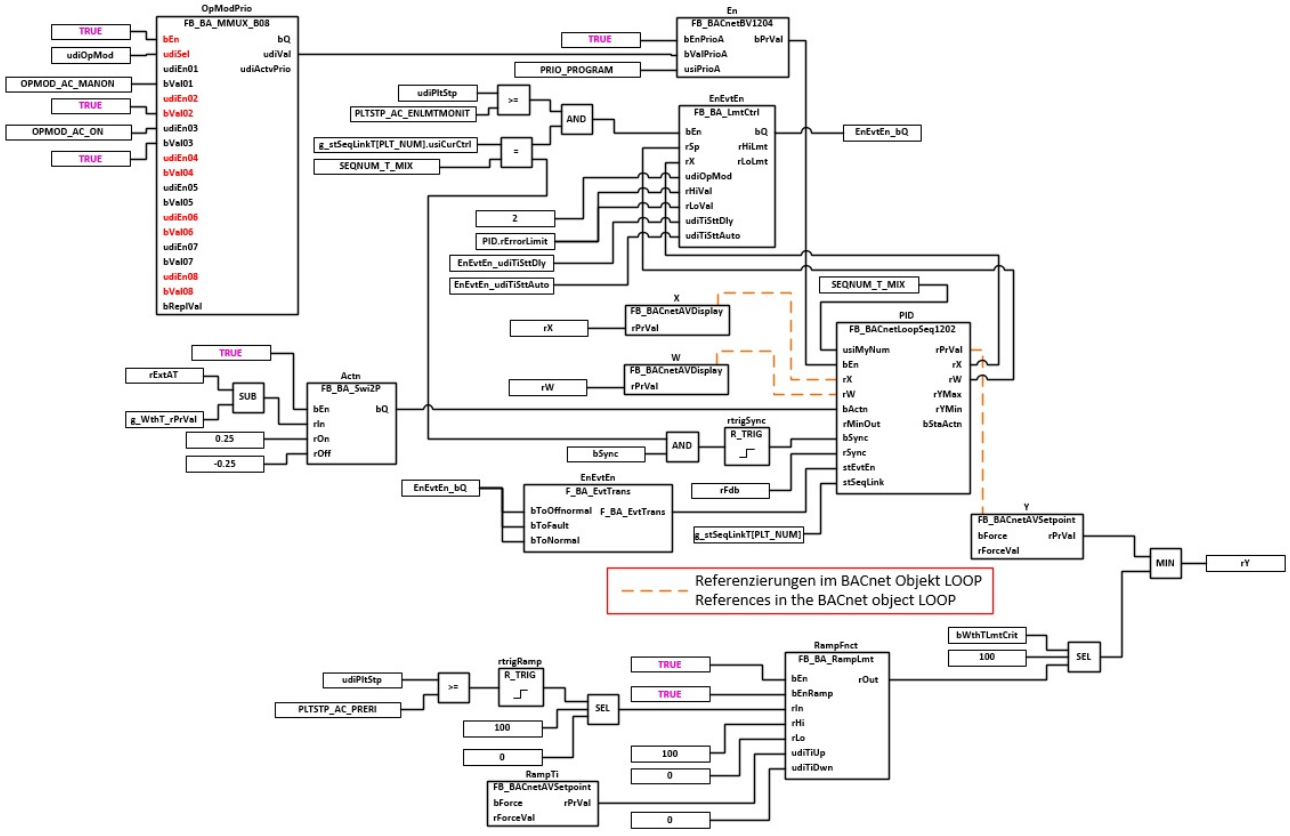
### Interface



### System diagram

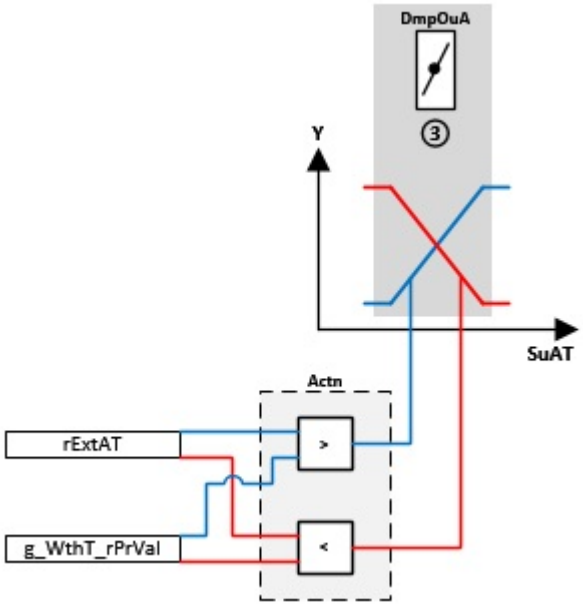



### Block diagram




### Direction of action

The direction of action of the PID sequence controller is selected based on a comparison of the outlet air temperature with the extract air temperature.
The logic of the direction of action relates to the **outside air damper DmpOuA**.
If the extract air temperature is higher than the outside air temperature, the direction of action of the PID controller is direct (cooling mode)
If the extract air temperature is lower than the outside air temperature, the direction of action of the PID controller is reverse (heating mode)




### Control information

A minimum outside air percentage must be set at the property MinimumOutput of the BACnet loop object!

### VAR_INPUT

```
rX          : REAL;
rW          : REAL;
bSync       : BOOL;
rFdb        : REAL;
rExtAT      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rX:** Measured value supply air temperature

**rW:** Set value of the supply air temperature

**bSync:** Input for synchronisation of the controller

**rFdb:** Position feedback outside air damper

**rExtAT:** Measured value outlet air temperature

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**udiOpMode:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rY    : REAL;
```

**rY:** Control value output

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkT[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

**Program description**

<table>
<tr><th>Instance</th><th>Type</th><th colspan="4">Task</th></tr>
<tr><td>X</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the actual value input of the BACnet loop object</td></tr>
<tr><td>W</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the setpoint input of the BACnet loop object</td></tr>
<tr><td>RampTi</td><td>FB_BACnetAVSetpoint [▶ 69]</td><td colspan="4">Input of the ramp time for a ramp function, during which the outside and exhaust air damper opens and the recirculating air damper closes very slowly during the start phase of the ventilation system.</td></tr>
<tr><td rowspan="3">OpModPrio</td><td rowspan="3">FB_BA_MMUX_B08 [▶ 205]</td><td colspan="4">The multiplexer defines the enable conditions of the sequence controller depending on the plant operation mode.</td></tr>
<tr><td colspan="2">udiOpMod</td><td>Enable</td><td>Comment</td></tr>
<tr><td>OPMOD_AC_MANON [▶ 357]</td><td>Manual on</td><td>TRUE</td><td>The plant is switched on manually via the plant selector switch</td></tr>
<tr><td></td><td>OPMOD_AC_ON [▶ 357]</td><td>On</td><td>TRUE</td><td>The plant runs in automatic mode via the timer program</td></tr>
<tr><td>En</td><td>FB_BACnetBV1204 [▶ 94]</td><td colspan="4">The BV object is used to display the sequence controller enable in the MCL.</td></tr>
<tr><td>Actn</td><td>FB_BA_Swi2P [▶ 146]<br>SUB</td><td colspan="4">The function block FB_BA_Swi2P [▶ 146] determines the control direction of the sequence controller.<br>If the subtraction of the exhaust air temperature and the outside temperature is < - 0.25, the output of the function block Actn is FALSE.<br>The control direction of the loop object is thus indirect and the mixed air damper control is in heating mode.<br>If the subtraction of the exhaust air temperature and the outside temperature is > + 0.25, the output of the function block Actn is TRUE.<br>The control direction of the loop object is thus direct and the mixed air damper control is in cooling mode.<br>The hysteresis between - 0.25 and + 0.25 of the Actn function block prevents frequent switching of the control direction at approximately the same outside and exhaust air temperature.</td></tr>
<tr><td>EnEvtEn</td><td>FB_BA_LmtCtrl [▶ 230]</td><td colspan="4">The BACnet loop object PID monitors the control function by comparing the setpoint W with the actual value X. If the deviation W-X is greater than the property ErrorLimit, then the loop object sends a message to the MCL.<br><br>When the plant is at a standstill, at the moment of startup and until the plant is in a controlled state, the loop object is suppressed so that no incorrect messages are sent to the MCL. Reporting of the</td></tr>
</table>

| Instance | Type | Task |
|---|---|---|
| | | loop object should not be activated until the air-conditioning system is fully running and the control is stable. In addition, reporting is enabled if the control has not reached the range around the setpoint defined by the property *ErrorLimit* after a long time.<br><br>Enabling the object internal reporting is done by writing to the BACnetEventTransitionBits of the loop object.<br><br>**The following conditions must be met to enable reporting from the loop object:**<br><br>**1.**<br>The plant startup program BAC_AC_StartT_01 [▶ 530] has enabled control monitoring and sensor limit monitoring<br>*udiPltStp* >= PLTSTP_AC_ENLMTMONIT [▶ 357]<br>and<br>The mixed air damper controller is the active controller in the control sequence. g_stSeqLinkT[PLT_NUM] [▶ 357]**.usiCurCtrl** = **SEQNUM_T_PREHTR**<br>and<br>The supply air temperature has approached the setpoint to a point where it has settled into a range between **rSp** - *ErrorLimit* and **rSp** + *ErrorLimit*.<br>and<br>the supply air temperature must have remained within the range of **rSp** - *ErrorLimit* and **rSp** + *ErrorLimit* for at least the duration of *EnEvtEn_udiTiSttDly*.<br><br>**2.**<br>The timer **EnEvtEn_udiTiSttAuto** has expired and the control has not reached its setpoint range. |
| rtrigSync | R_TRIG | A rising edge at input *bSync* triggers synchronization of the loop object to the value of *lrSync*.<br>If the outside air damper of the mixed air damper control was overridden by writing a higher priority to the corresponding AO object of the MCL or by activating the local priority operation, the current position of the control valve deviates from the output of the loop object. The variables *bSync* and *rFdb* can be used to restore synchronicity between the position of the outside air damper and the controller. |
| | F_BA_EvtTrans | writes to the BACnetEventTransitionBits to_offnormal, to_fault and to_normal of the loop object. The input bEnEvtEn must be TRUE |
| PID | FB_BACnetLoopSeq1201 [▶ 106] | Sequence controller supply air temperature mixed air damper control |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |
| rtrigRamp | R_TRIG<br>>=<br>SEL | During the start phase of the ventilation system this network of functions activates the ramp function. |
| RampFnct | FB_BA_RampLmt [▶ 141] | The ramp function is used to open the outside and exhaust air dampers very slowly during plant startup. The ramp function is started only at critical outside temperatures below 6°C .<br>This function replaces the pre-purge of the hot water air heater. |
| | SEL, MIN | The result of this network is the control value for the mixed air dampers. |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.41 BAC_AC_ReHtr_01

### Application

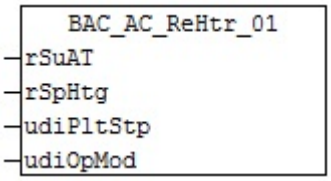
The call template **BAC_AC_ReHtr_01** is used for controlling a reheater.

The main tasks of the template are:

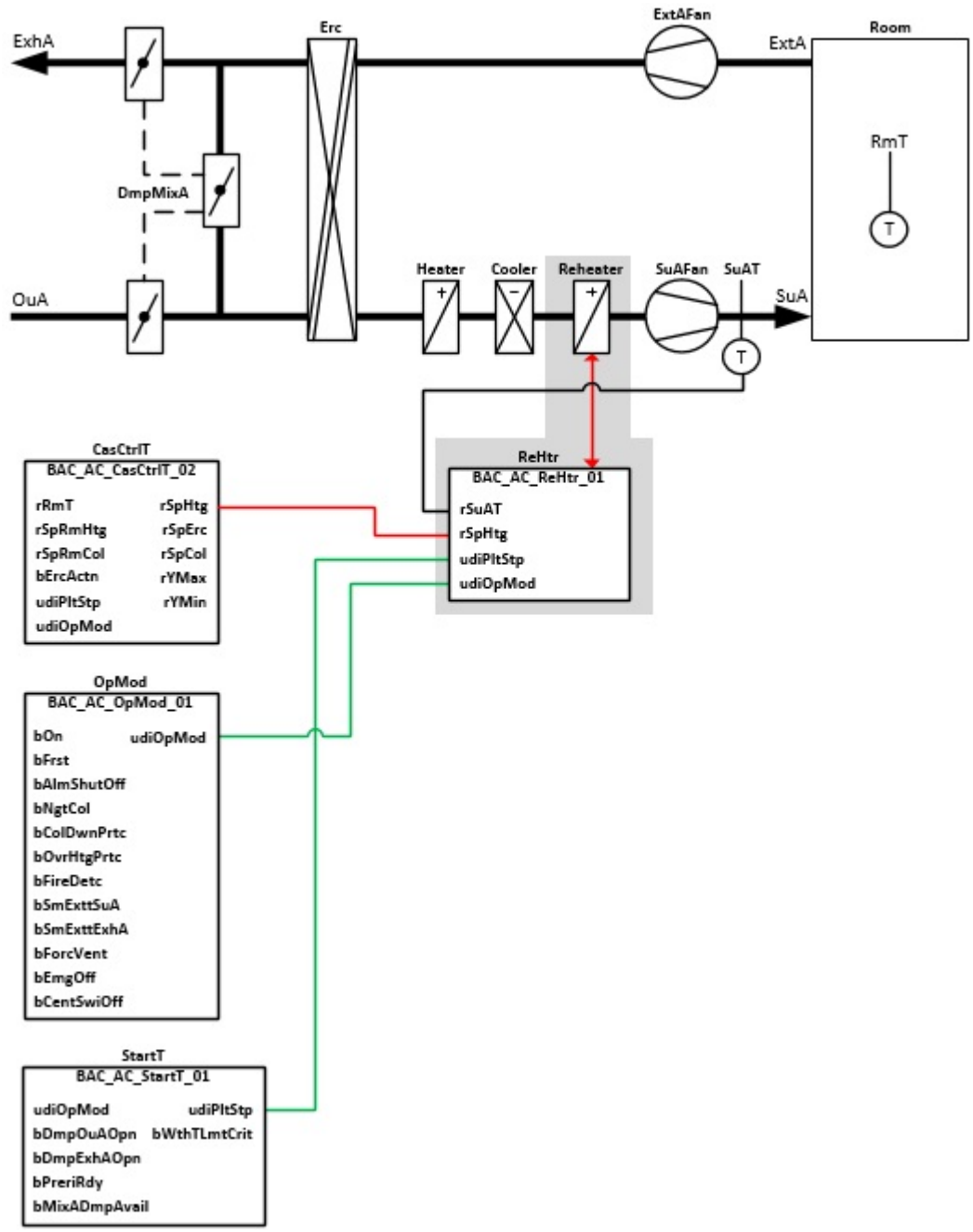
- Control of the supply air temperature
- Pump enable
- Control of the control valve

The enable of the supply air temperature controller **SuA_PID** is formed in the template via an evaluation of the plant operation mode **udiOpMod**.

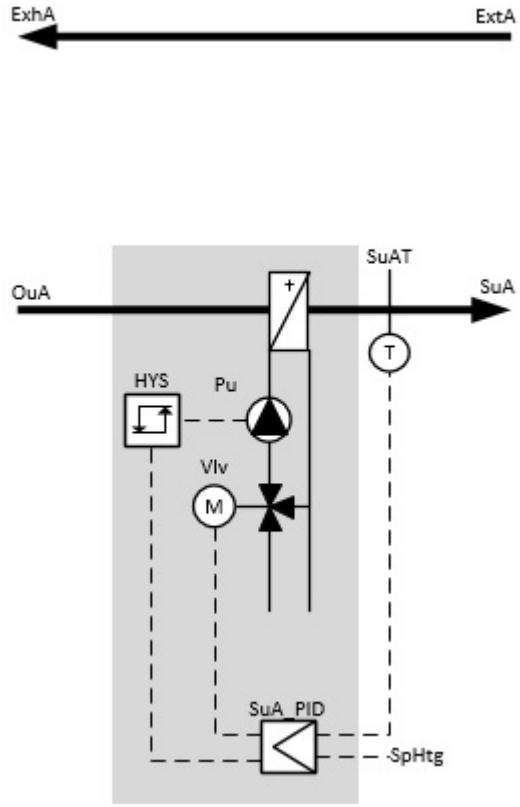
### Interface



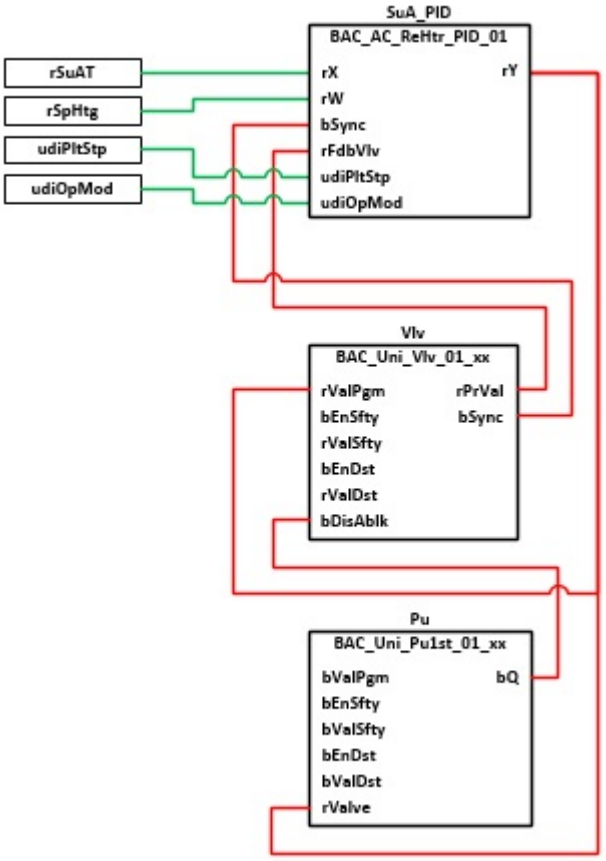
**Plant diagram 01**



### Plant diagram 02




### Block diagram



### VAR_INPUT

```
rSuAT       : REAL;
rSpHtg      : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rSuAT:** Measured value supply air temperature

**rSpHtg:** Set value of the supply air temperature

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

### Program description

| Instance | Type | Task |
|---|---|---|
| SuA_PID | BAC_AC_ReHtr_PID_01 [▶ 488] | Subtemplate supply air temperature control. The temperature control is analog via PID sequence controller. |
| Vlv | BAC_Uni_Vlv_01_13 [▶ 598] | Subtemplate for controlling an analog valve |
| Pu | BAC_Uni_Pu1st_01_189 [▶ 571] | Subtemplate for controlling a single-stage pump |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.42 BAC_AC_ReHtr_PID_01

### Functional description

The sub template **BAC_AC_ReHtr_PID_01** is the supply air temperature sequence controller for a reheater.

The set value, actual value and control output are referenced via the BACnet value objects **X**, **W** and **Y**.

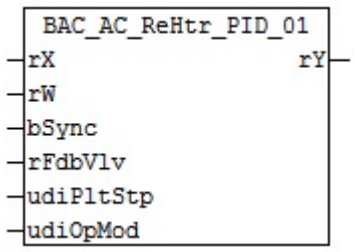
The PID sequence controller is enabled based on the plant operation mode **udiOpMod** and the global temperature communication structure **g_stSeqLinkT[PLT_NUM]**.

This data and command structure is the link between the individual sequence controllers and the corresponding control function block FB_BA_SeqLink [▶ 168] of a plant.

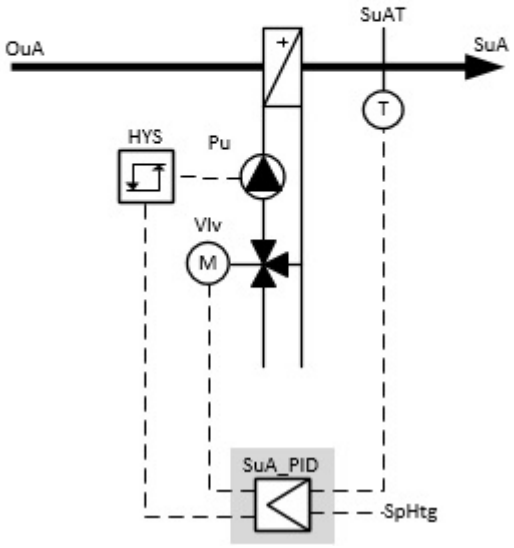
The BACnet BV object **En** is used to display the controller enable.

The limit value monitoring of PID controller is controlled by the function block **EnEvtEn** as a function of the plant startup process.

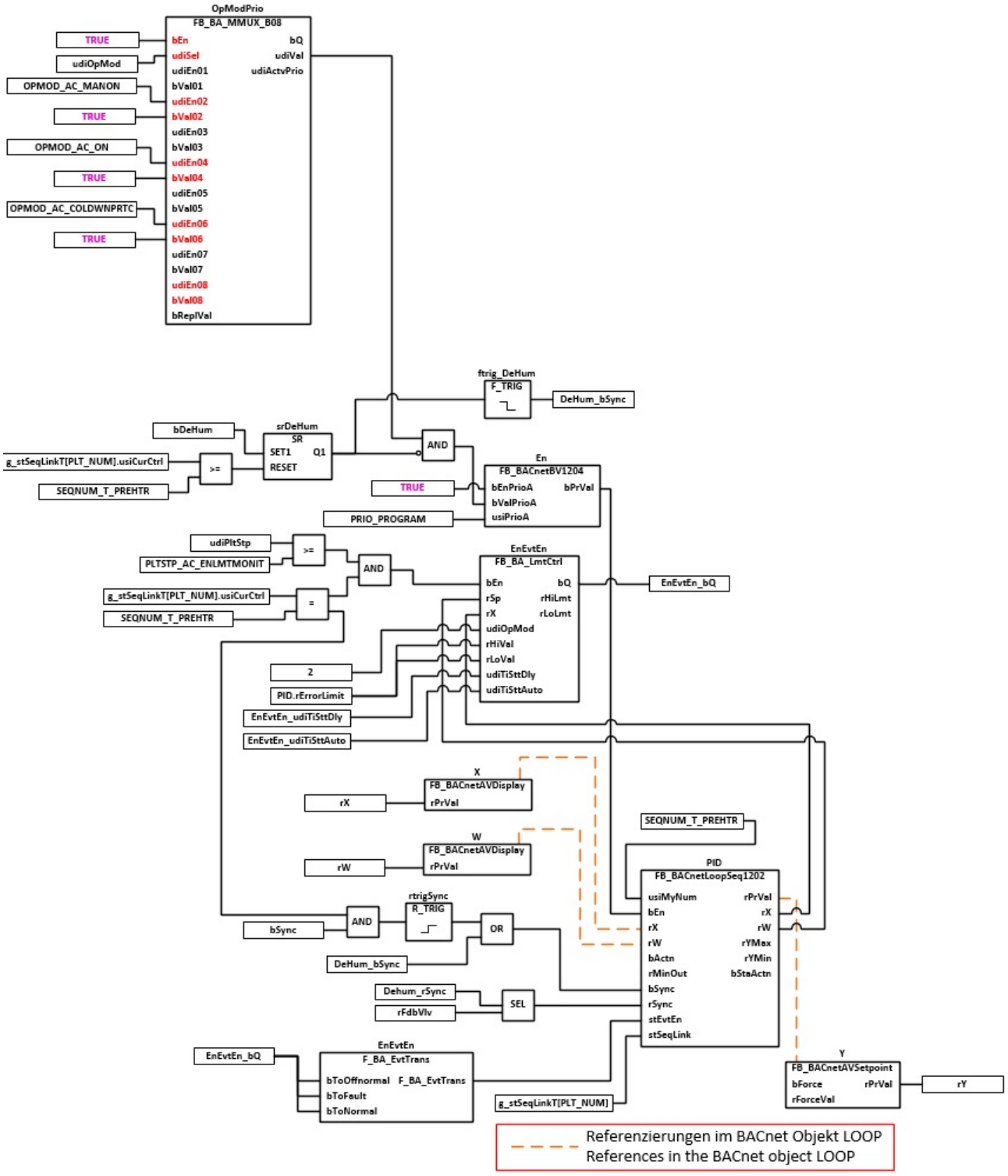
### Interface



**System diagram**


Block diagram




## VAR_INPUT

```
rX          : REAL;
rW          : REAL;
bSync       : BOOL;
rFdbVlv     : REAL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rX:** Measured value supply air temperature

**rW:** Set value of the supply air temperature

**bSync:** Input for synchronization of the controller

**rFdbVlv:** Position feedback actuator

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rY    : REAL;
```

**rY:** Output of the control value for the control valve

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkT[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

<table>
<tr><th>Instance</th><th>Type</th><th colspan="4">Task</th></tr>
<tr><td>X</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the actual value input of the BACnet loop object</td></tr>
<tr><td>W</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the setpoint input of the BACnet loop object</td></tr>
<tr><td rowspan="4">OpModPrio</td><td rowspan="4">FB_BA_MMUX_B08 [▶ 205]</td><td colspan="4">The multiplexer defines the enable conditions of the sequence controller depending on the plant operation mode.</td></tr>
<tr><td colspan="2">udiOpMod</td><td>Enable</td><td>Comment</td></tr>
<tr><td>OPMOD_AC_MANON [▶ 357]</td><td>Manual on</td><td>TRUE</td><td>The plant is switched on manually via the plant selector switch</td></tr>
<tr><td>OPMOD_AC_ON [▶ 357]</td><td>On</td><td>TRUE</td><td>The plant runs in automatic mode via the timer program</td></tr>
<tr><td>OPMOD_AC_COLDWNPRTC [▶ 357]</td><td>Support operation, cooling protection</td><td>TRUE</td><td>The plant operates in cooling protection mode</td></tr>
<tr><td>En</td><td>FB_BACnetBV1204 [▶ 94]</td><td colspan="4">The BV object is used to display and activate the controller enable in the MCL or in a local operating display.<br>The controller is enabled depending on the plant operation mode.</td></tr>
<tr><td>EnEvtEn</td><td>FB_BA_LmtCtrl [▶ 230]</td><td colspan="4">The BACnet loop object PID monitors the control function by comparing the setpoint W with the actual value X. If the deviation W-X is greater than the property ErrorLimit, then the loop object sends a message to</td></tr>
</table>

| Instance | Type | Task |
|---|---|---|
| | | the MCL.<br><br>When the plant is at a standstill, at the moment of startup and until the plant is in a controlled state, the loop object is suppressed so that no incorrect messages are sent to the MCL. Reporting of the loop object should not be activated until the air-conditioning system is fully running and the control is stable. In addition, reporting is enabled if the control has not reached the range around the setpoint defined by the property **ErrorLimit** after a long time.<br><br>Enabling the object internal reporting is done by writing to the BACnetEventTransitionBits of the loop object.<br><br>**The following conditions must be met to enable reporting from the loop object:**<br><br>**1.**<br>The plant startup program BAC_AC_StartT_01 [▶ 530] has enabled control monitoring and sensor limit monitoring<br>**udiPltStp** >= **PLTSTP_AC_ENLMTMONIT**<br>and<br>the reheater controller is the active controller in the control sequence.<br>g_stSeqLinkT[PLT_NUM] [▶ 357]**.usiCurCtrl** = **SEQNUM_T_REHTR**<br>and<br>the supply air temperature has approached the setpoint to a point where it has settled into a range between **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit**<br>and<br>the supply air temperature must have remained within the range of **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit** for at least the duration of **EnEvtEn_udiTiSttDly**.<br><br>**2.**<br>The timer **EnEvtEn_udiTiSttAuto** has expired and the control has not reached its setpoint range. |
| | F_BA_EvtTrans | writes to the BACnetEventTransitionBits to_offnormal, to_fault and to_normal of the loop object. The input **bEnEvtEn** must be TRUE |
| PID | FB_BACnetLoopSeq1202 [▶ 110] | Sequence controller supply air temperature reheater. |
| rtrigSync | R_TRIG | A rising edge at input **bSync** triggers synchronization of the loop object to the value of **lrSync**.<br>If the control valve of the reheater was overridden by writing a higher priority to the corresponding AO object of the MCL or by activating the local priority operation, the current position of the control valve deviates from the output of the loop object. The variables **bSync** and **rFdbVlv** can be used to restore synchronicity between the position of the control valve and the controller. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.43 BAC_AC_PreHtr_01

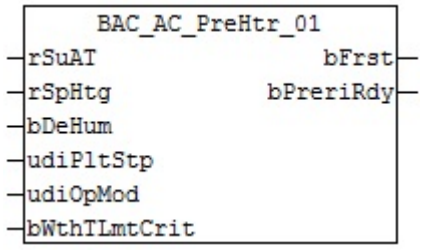
### Application

The call template **BAC_AC_PreHtr_01** is used to control a hot-water air heater.

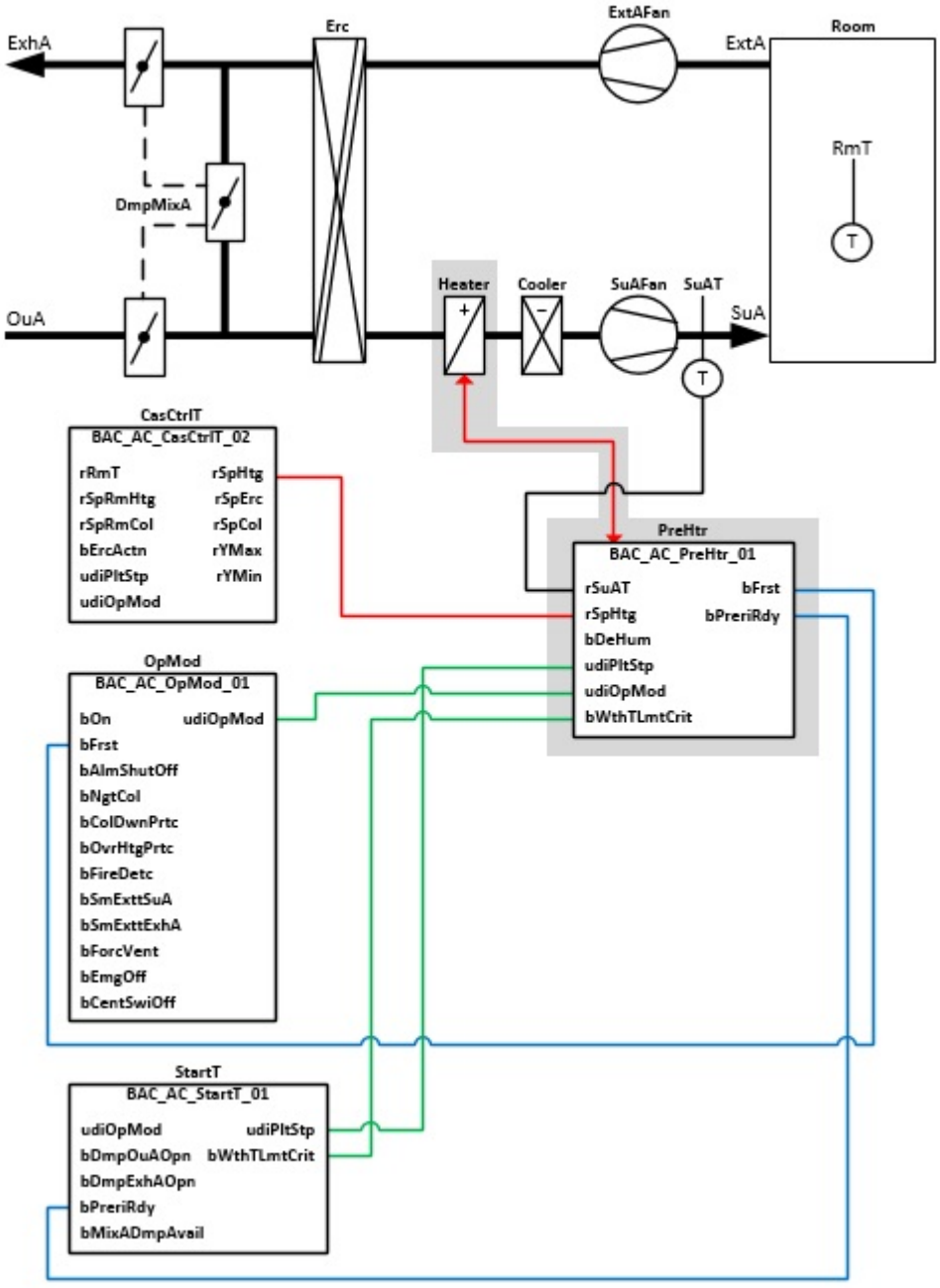
The main tasks of the template are:

- Control of the supply air temperature
- Control of the return temperature
- Frost monitoring air side with frost protection thermostat **FrstThermostat**
- Frost monitoring air side with analog frost protection sensor **FrstT**
- Frost monitoring water side with return temperature sensor **RetWtrT**
- Enable the heater pump
- Control and monitoring of the heater valve

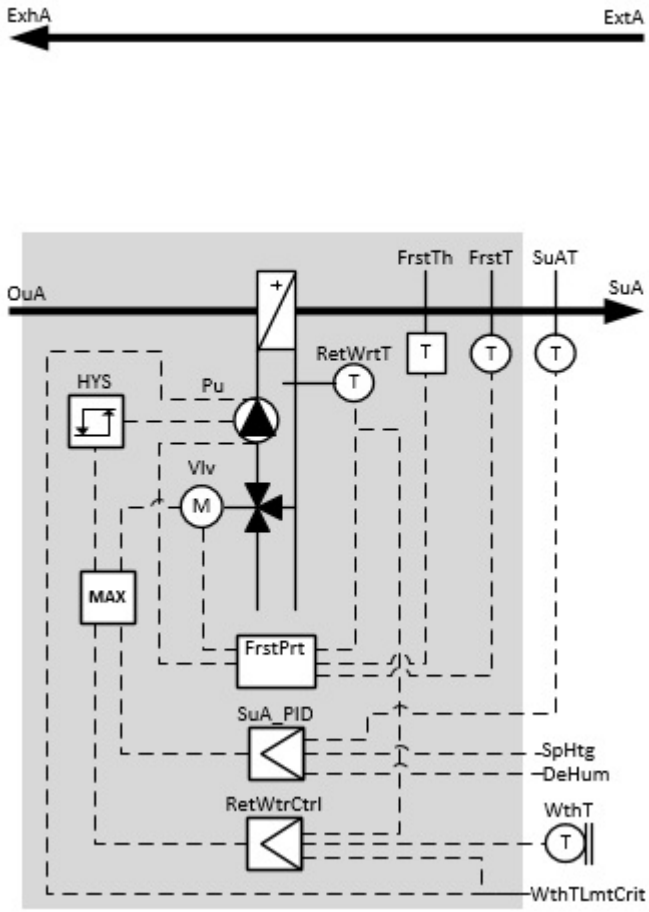
### Interface



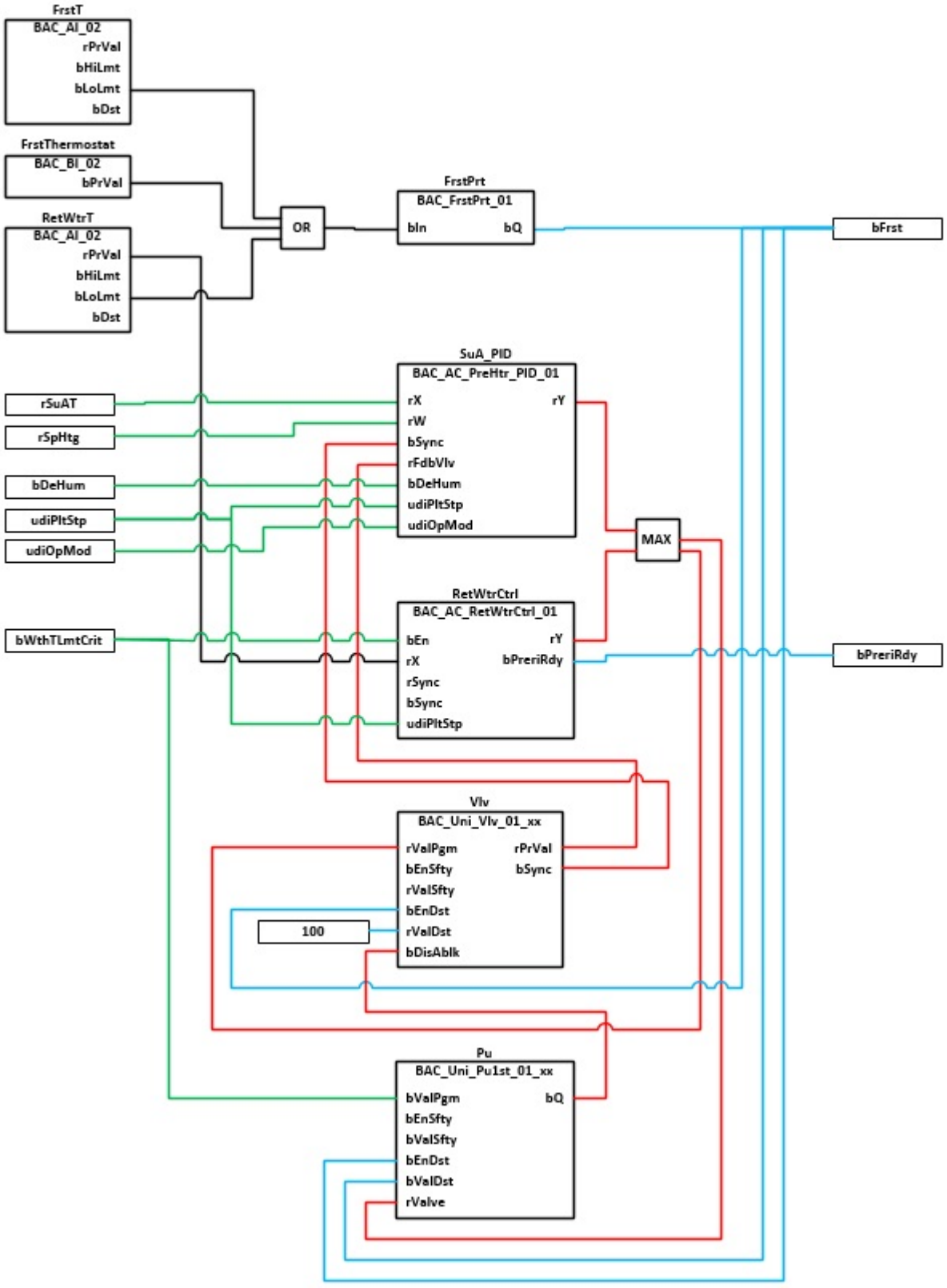
**Plant diagram 01**



**Plant diagram 02**



## Block diagram




## VAR_INPUT

```
rSuAT           : REAL;
rSpHtg          : REAL;
bDeHum          : BOOL;
udiPltStp       : UDINT;
udiOpMod        : UDINT;
bWthTLmttCrit   : BOOL;
bRst            : BOOL;
```

**rSuAT:** Measured value of the supply air temperature

**rSpHtg:** Set value for the supply air temperature

**bDeHum:** Status message dehumidification mode

If dehumidification is active, the supply air temperature controller **SuA_PID** is disabled and therefore removed from the sequence control.

**udiPltStp:** Plant start sequence steps. The plant steps are generated in the system start program BAC_AC_StartT_01 [▶ 530].

**udiOpMod:** Plant operation mode. The plant operation mode is determined in the mode selection programBAC_AC_OpMod_01 [▶ 515].

**bWthTLmtCrit:** Status message weather temperature critical. This information is generated in the system start program BAC_AC_StartT_01 [▶ 530].

The following actions are triggered when the outside temperature falls below a critical value:

- Heater pump activation, forced pump operation
- Return temperature control enable BAC_AC_RetWtrCtrl_01 [▶ 503]
- During plant startup prerinsing mode is active, see BAC_AC_RetWtrCtrl_01 [▶ 503]

**bRst:** Acknowledge frost protection message.

### VAR_OUTPUT

```
bFrst      : BOOL;
bPreriRdy  : BOOL;
```

**bFrst:** Frost protection program output active.

**bPreriRdy:** Pre-rinsing process output complete.

### Program description

| Instance | Type | Task |
|---|---|---|
| RetWtrT | BAC_AI_02 [▶ 676] | Subtemplate AI object return temperature sensor |
| FrstT | BAC_AI_02 [▶ 676] | Subtemplate AI object frost protection sensor air side |
| FrstThermostat | BAC_BI_02 [▶ 695] | Subtemplate BI object frost protection thermostat |
| SuA_PID | BAC_AC_PreHtr_PID_01 [▶ 498] | Subtemplate loop object supply air temperature control |
| RetWtrCtrl | BAC_AC_RetWtrCtrl_01 [▶ 503] | Subtemplate return temperature control of the hot water air heater |
| FrstPrt | BAC_FrstPrt_01 [▶ 507] | Subtemplate frost protection program |
| Vlv | BAC_Uni_Vlv_01_13 [▶ 598] | Subtemplate control valve |
| Pu | BAC_Uni_Pu1st_01_189 [▶ 571] | Subtemplate pump |
| | MAX | MAX selection of the control signals for supply air temperature control **SuA_PID** and return temperature control **RetWtrCtrl** |
| | OR | Collects frost events and passes them to the frost protection program.<br>The frost event occurs in the following conditions<br>• The frost protection thermostat is triggered<br>• The air temperature after the heating coil falls below the LowLimit |

| Instance | Type | Task |
|---|---|---|
| | | • The return temperature falls below the LowLimit (RetWtr.bLoLmt = TRUE) |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.44 BAC_AC_PreHtr_PID_01

**Functional description**

The subtemplate **BAC_AC_PreHtr_PID_01** is the sequence controller for a preheater.

The set value, actual value and control output are referenced via the BACnet value objects **X**, **W** and **Y**.

The PID sequence controller is enabled based on the plant operation mode **udiOpMod** and the global temperature communication structure **g_stSeqLinkT[PLT_NUM]**.

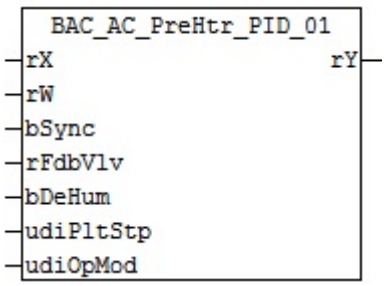
This data and command structure is the link between the individual sequence controllers and the corresponding control function block FB_BA_SeqLink [▶ 168] of a plant.

The BACnet BV object **En** is used to display the controller enable.

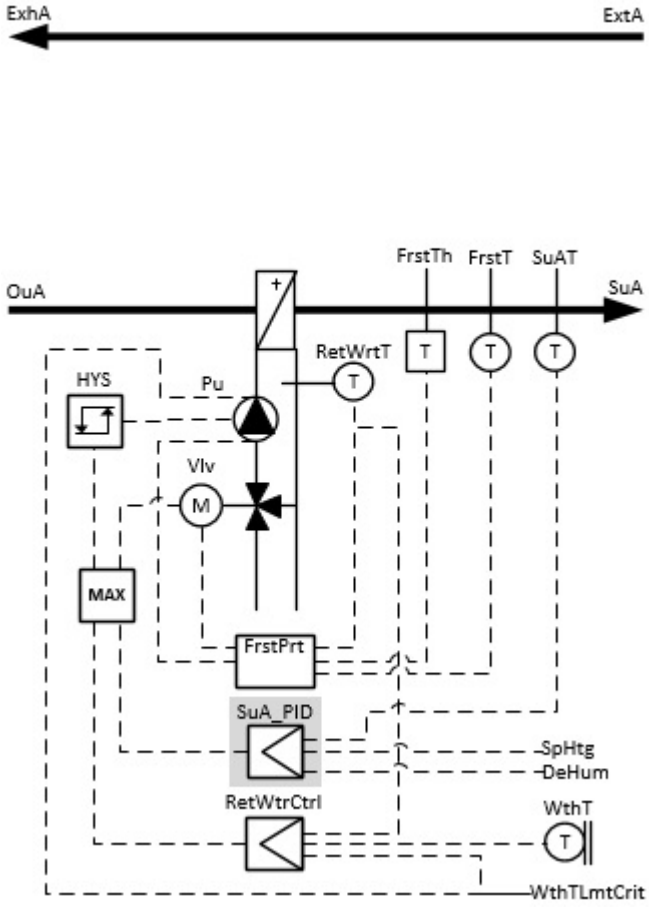
The limit value monitoring of PID controller is controlled by the function block **EnEvtEn** as a function of the plant startup process.

In dehumidification mode **bDeHum** = TRUE, the PID sequence controller is disabled.

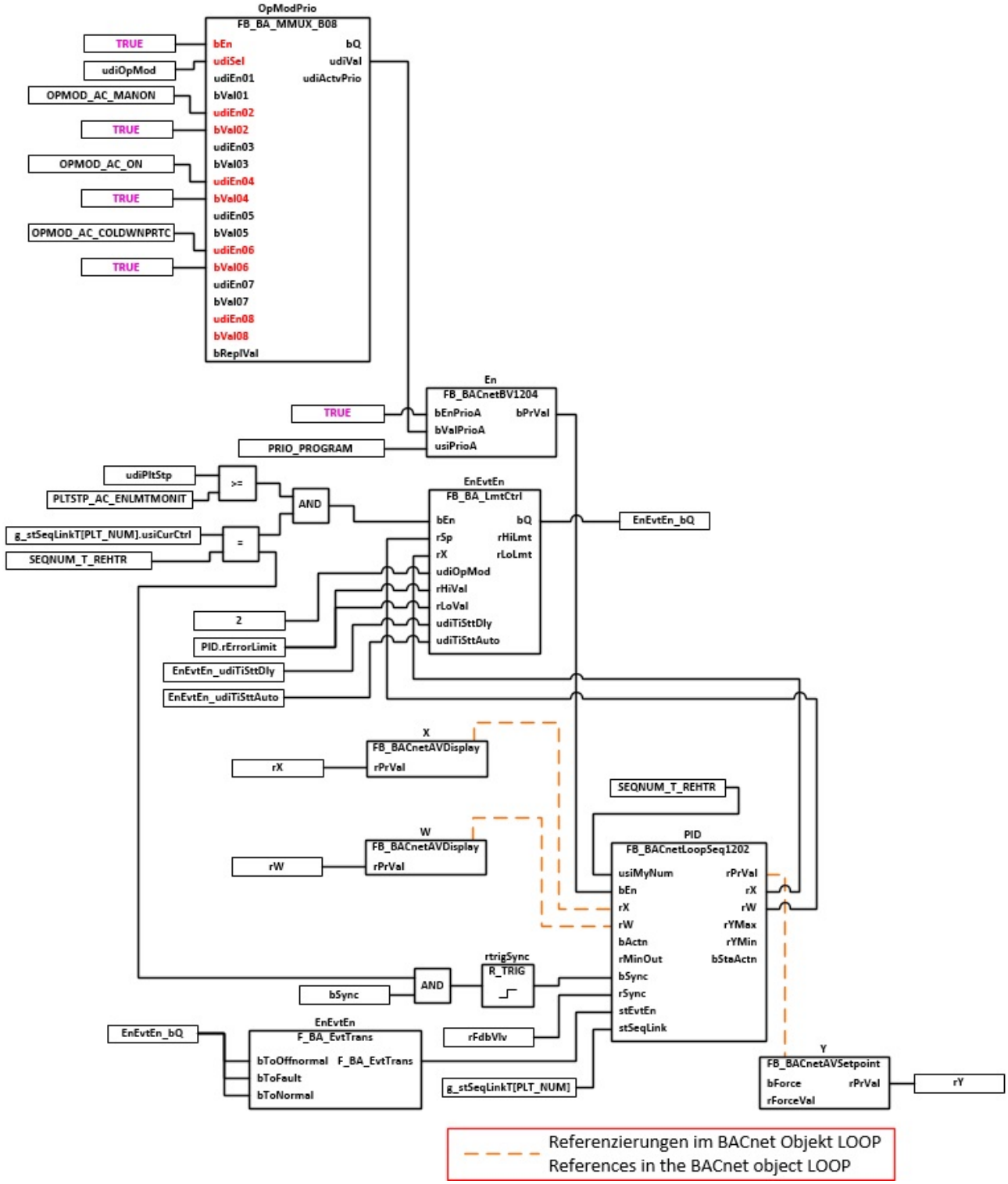
**Interface**



### System diagram



## Block diagram




## VAR_INPUT

```
rX          : REAL;
rW          : REAL;
bSync       : BOOL;
rFdbVlv     : REAL;
bDeHum      : BOOL;
udiPltStp   : UDINT;
udiOpMod    : UDINT;
```

**rX:** Measured value supply air temperature

**rW:** Set value of the supply air temperature

**bSync:** Input for synchronisation of the controller

**rFdbVlv:** Position feedback actuator

**bDeHum:** Dehumidification active, supply air temperature controller disabled.

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rY    : REAL;
```

**rY:** Control value output control valve

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkT[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

<table>
<tr><th>Instance</th><th>Type</th><th colspan="4">Task</th></tr>
<tr><td>X</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the actual value input of the BACnet loop object</td></tr>
<tr><td>W</td><td>FB_BACnetAVDisplay [▶ 69]</td><td colspan="4">The AV object is referenced to the setpoint input of the BACnet loop object</td></tr>
<tr><td rowspan="4">OpModPrio</td><td rowspan="4">FB_BA_MMUX_B08 [▶ 205]</td><td colspan="4">The multiplexer defines the enable conditions of the sequence controller depending on the plant operation mode.</td></tr>
<tr><td colspan="2">udiOpMod</td><td>Enable</td><td>Comment</td></tr>
<tr><td>OPMOD_AC_MANON [▶ 357]</td><td>Manual on</td><td>TRUE</td><td>The plant is switched on manually via the plant selector switch</td></tr>
<tr><td>OPMOD_AC_ON [▶ 357]</td><td>On</td><td>TRUE</td><td>The plant runs in automatic mode via the timer program</td></tr>
</table>

| Instance | Type | Task | | | |
|---|---|---|---|---|---|
| | | OPMOD_AC_COLDWNPRTC [▶ 357] | Support operation, cooling protection | TRUE | The plant operates in cooling protection mode |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display and activate the controller enable in the MCL or in a local operating display.<br>The controller is enabled depending on the plant operation mode. | | | |
| | AND | The AND function block locks the enable of the sequence controller in dehumidification mode. Switching from preheater to reheater is done in the template BAC_AC_SeqT_01 [▶ 524] by the sequence linker. | | | |
| EnEvtEn | FB_BA_LmtCtrl [▶ 230] | The BACnet loop object PID monitors the control function by comparing the setpoint **W** with the actual value **X**. If the deviation W-X is greater than the property **ErrorLimit**, then the loop object sends a message to the MCL.<br><br>When the plant is at a standstill, at the moment of startup and until the plant is in a controlled state, the loop object is suppressed so that no incorrect messages are sent to the MCL. Reporting of the loop object should not be activated until the air-conditioning system is fully running and the control is stable. In addition, reporting is enabled if the control has not reached the range around the setpoint defined by the property **ErrorLimit** after a long time.<br><br>Enabling the object internal reporting is done by writing to the BACnetEventTransitionBits of the loop object.<br><br>**The following conditions must be met to enable reporting from the loop object:**<br><br>**1.**<br>The plant startup program BAC_AC_StartT_01 [▶ 530] has enabled control monitoring and sensor limit monitoring<br>**udiPltStp** >= **PLTSTP_AC_ENLMTMONIT**<br>and<br>The heater controller is the active controller in the control sequence. g_stSeqLinkT[PLT_NUM] [▶ 357]**.usiCurCtrl** = **SEQNUM_T_PREHTR**<br>and<br>The supply air temperature has approached the setpoint to a point where it has settled into a range between **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit**.<br>and<br>The supply air temperature must have remained within the range of **rSp** - **ErrorLimit** and **rSp** + **ErrorLimit** for at least the duration of **EnEvtEn_udiTiSttDly**.<br><br>**2.**<br>The timer **EnEvtEn_udiTiSttAuto** has expired and the control has not reached its setpoint range. | | | |
| | F_BA_EvtTrans | writes to the BACnetEventTransitionBits to_offnormal, to_fault and to_normal of the loop object. The input bEnEvtEn must be TRUE | | | |
| PID | FB_BACnetLoopSeq1202 [▶ 110] | Sequence controller supply air temperature preheater. | | | |

| Instance | Type | Task |
|---|---|---|
| rtrigSync | R_TRIG | A rising edge at input **bSync** triggers synchronization of the loop object to the value of **lrSync**.<br>If the control valve of the preheater was overridden by writing a higher priority to the corresponding AO object of the MCL or by activating the local priority operation, the current position of the control valve deviates from the output of the loop object. The variables **bSync** and **rFdbVlv** can be used to restore synchronicity between the position of the control valve and the controller. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.45 BAC_AC_RetWtrCtrl_01

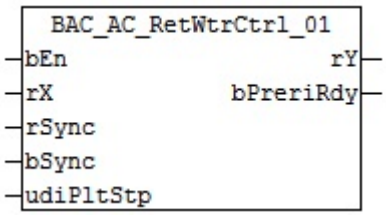
### Application

The call template BAC_AC_RetWtrCtrl_01 is used for controlling the return temperature of the hot-water air heater.

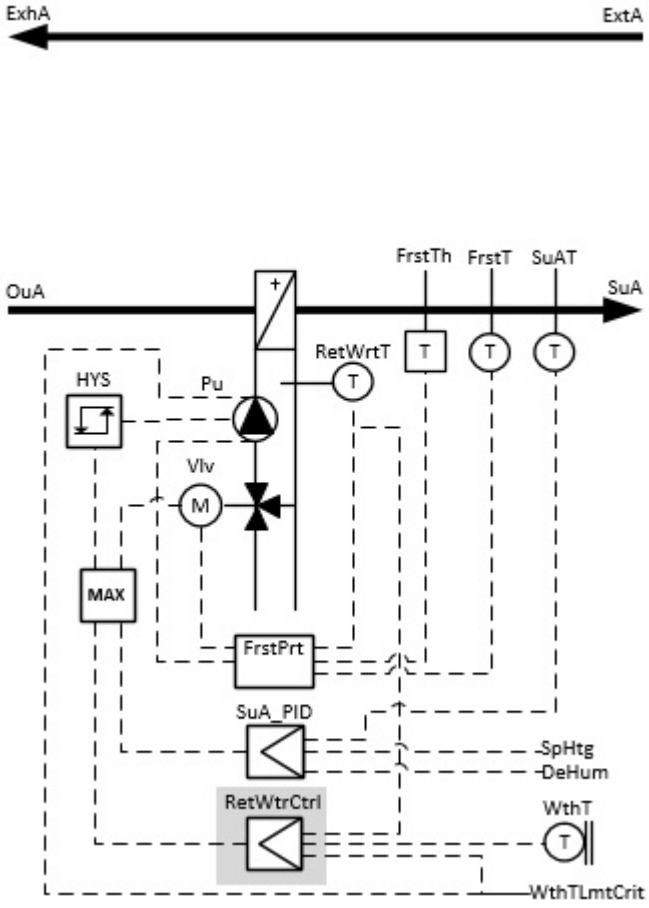
The main tasks of the template are:

- Control of the return temperature during the pre-rinsing process on plant startup.
- Control of the return temperature to a frost protection set value during plant operation and standstill.

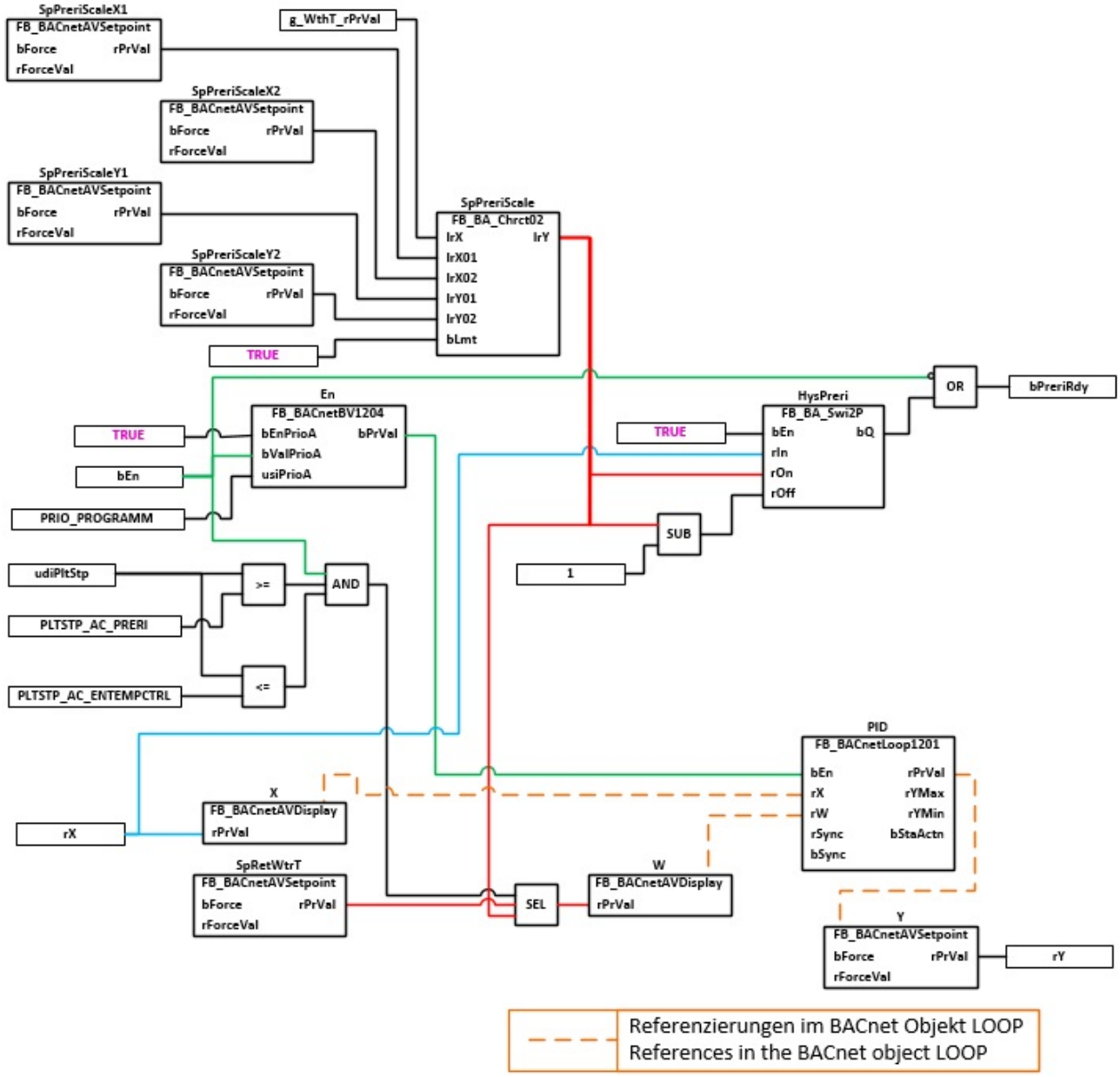
### Interface



**Diagram**


ExhA
ExtA
OuA
SuA
FrstTh
FrstT
SuAT
RetWrtT
HYS
Pu
Vlv
M
MAX
FrstPrt
SuA_PID
SpHtg
DeHum
RetWtrCtrl
WthT
WthTLmtCrit

### Block diagram




### VAR_INPUT

```
bEn         : BOOL;
rX          : REAL;
rSync       : BOOL;
bSync       : UDINT;
udiPltStp   : UDINT;
```

**bEn:** Return temperature control enable.

**rX:** Measured value of the return temperature

**rSync:** Synchronization value for the controller **PID**

**bSync:** Input for synchronization of the controller **PID**

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

### AR_OUTPUT

```
rY          : REAL;
bPreriRdy   : BOOL;
```

**rY:** Control value output

**bPreriRdy:** Pre-rinsing process message complete

### VAR CONSTANT

```
PLT_NUM       : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| SpPreriScaleX1 | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the setpoint for prerinsing scaling point X1 |
| SpPreriScaleX2 | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the setpoint for prerinsing scaling point X2 |
| SpPreriScaleY1 | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the setpoint for prerinsing scaling point Y1 |
| SpPreriScaleY2 | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the setpoint for prerinsing scaling point Y2 |
| SpRetWtrT | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the setpoint for the return temperature water |
| SpPreriScale | FB_BA_Chrct02 [▶ 171] | Function block for calculating the prerinsing temperature depending on the outside temperature for return temperature control during plant startup. |
| HysPreri | FB_BA_Swi2P [▶ 146] | When the air-conditioning plant starts up at low outside temperatures, the heating coil is pre-rinsed with warm water. Only then are the dampers opened and the fans switched on. When the prerinsing temperature **SpPreriScale_IrY** has been reached, the function block **HysPreri** switches the variable **bPreriRdy** to TRUE. This tells the plant start program BAC_AC_StartT_01 [▶ 530] that the prerinsing process is complete.<br>If the input variable **bEn** is not set to TRUE during the plant startup, **bPreriRdy** is set immediately, without waiting for the prerinsing process to complete. This means that at higher outside temperatures the air-conditioning plant starts without prerinsing of the heating coil. |
| | SUB | The value of the subtraction is the lower switching point of the two-point switch **HysPreri.** |
| | OR | Outputs the prerinsing status through **HysPreri** or **bEn**. |
| X | FB_BACnetAVDisplay [▶ 69] | AV object for displaying the return temperature of the hot water air heater.<br>The AV object is referenced to the actual value input of the BACnet loop object **PID**. |

| Instance | Type | Task |
|---|---|---|
| W | FB_BACnetAVDisplay [▶ 69] | AV object for displaying the setpoint of the return temperature controller.<br>The AV object is referenced to the setpoint input of the BACnet loop object **PID**. |
| | >=, <=, AND, SEL | The value of this networks determines the setpoint for the return temperature control of the hot water air heater **PID**. |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display and activate the controller enable in the MCL or in a local operator display.<br>The controller is enabled via the input variable **bEn**. |
| PID | FB_BACnetLoop1201 [▶ 98] | Loop object Return temperature control of the hot water air heater.<br>It is responsible for the prerinsing of the heating coil during the plant start of the ventilation system and the control of the return temperature to the setpoint **SpRetWtrT**.<br>The return temperature control is enabled at low outside temperatures by the input variable **bEn**. The enable takes place in the call template BAC_AC_PreHtr_01 [▶ 493]. |
| Y | FB_BACnetAVSetpoint [▶ 69] | AV object for displaying the control value of the return temperature control.<br>The AV object is referenced to the control value output of the BACnet loop object **PID**. |

### Version history

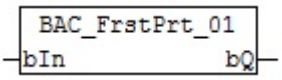
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.46 BAC_FrstPrt_01

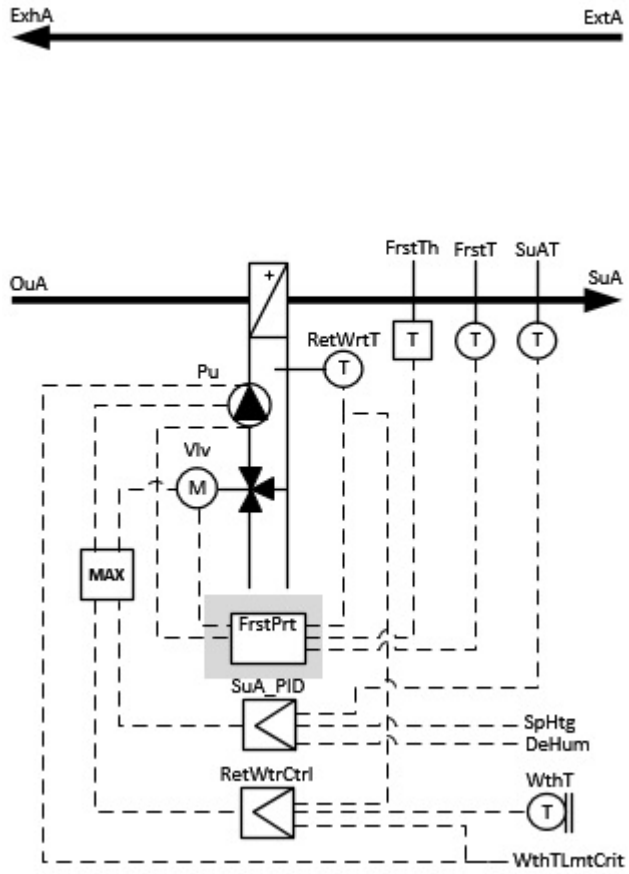
### Functional description

The template realises frost monitoring for a heating coil in an air-conditioning plant.

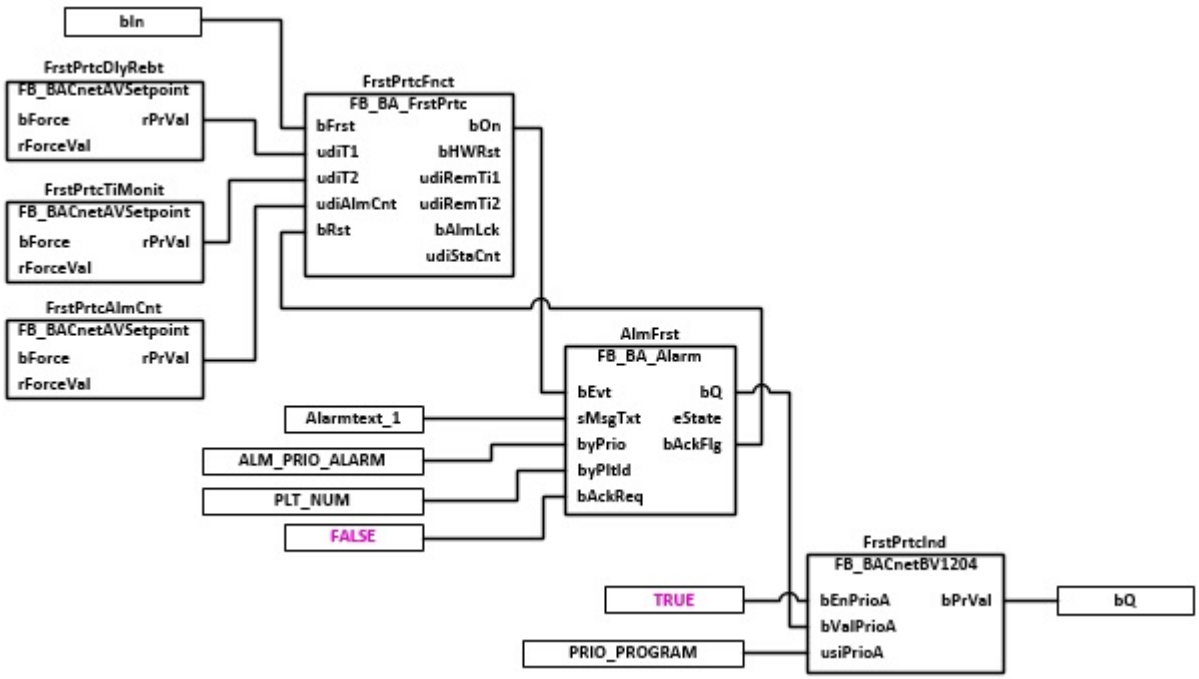
### Interface



### System diagram




### Block diagram




### VAR_INPUT

```
bIn       : BOOL;
bRst      : BOOL;
```

**bIn:** Frost event

**bRst:** Frost alarm acknowledgement

### VAR_OUTPUT

```
bQ    : BOOL;
```

**bQ:** Frost protection active message

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| FrstPrtcDlyRebt | FB_BACnetAVSetpoint [▶ 69] | AV object for input of frost protection timer T1 (time within which the frost protection alarm must have ceased) |
| FrstPrtcTiMonit | FB_BACnetAVSetpoint [▶ 69] | AV object for input of frost protection timer T2 (time interval within which no further frost alarm may occur, without requirement for manual acknowledgement) |
| FrstPrtcAlmCnt | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the maximum number of automatic plant restarts after a frost event |
| FrstPrtcFnct | FB_BA_FrstPrtc [▶ 224] | The function block is the core of the frost protection monitoring. |
| AlmFrst | FB_BA_Alarm | The function block **AlmFrst** detects the frost protection event. In addition, an acknowledgement pulse is forwarded to the frost protection monitoring **FrstPrtcFnct** via the output **bAckFlg**. Actions that are to take place after the input of the frost protection active can be parameterized in the template at the function block **AlmFrst**. |
| FrstPrtcInd | FB_BACnetBV1204 [▶ 94] | BV object for display of the frost alarm |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.47 BAC_AC_VAV_01_xx

### Functional description

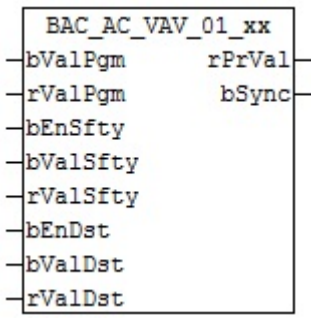
The template is used for controlling a volume flow controller via an analog output.

### Versions

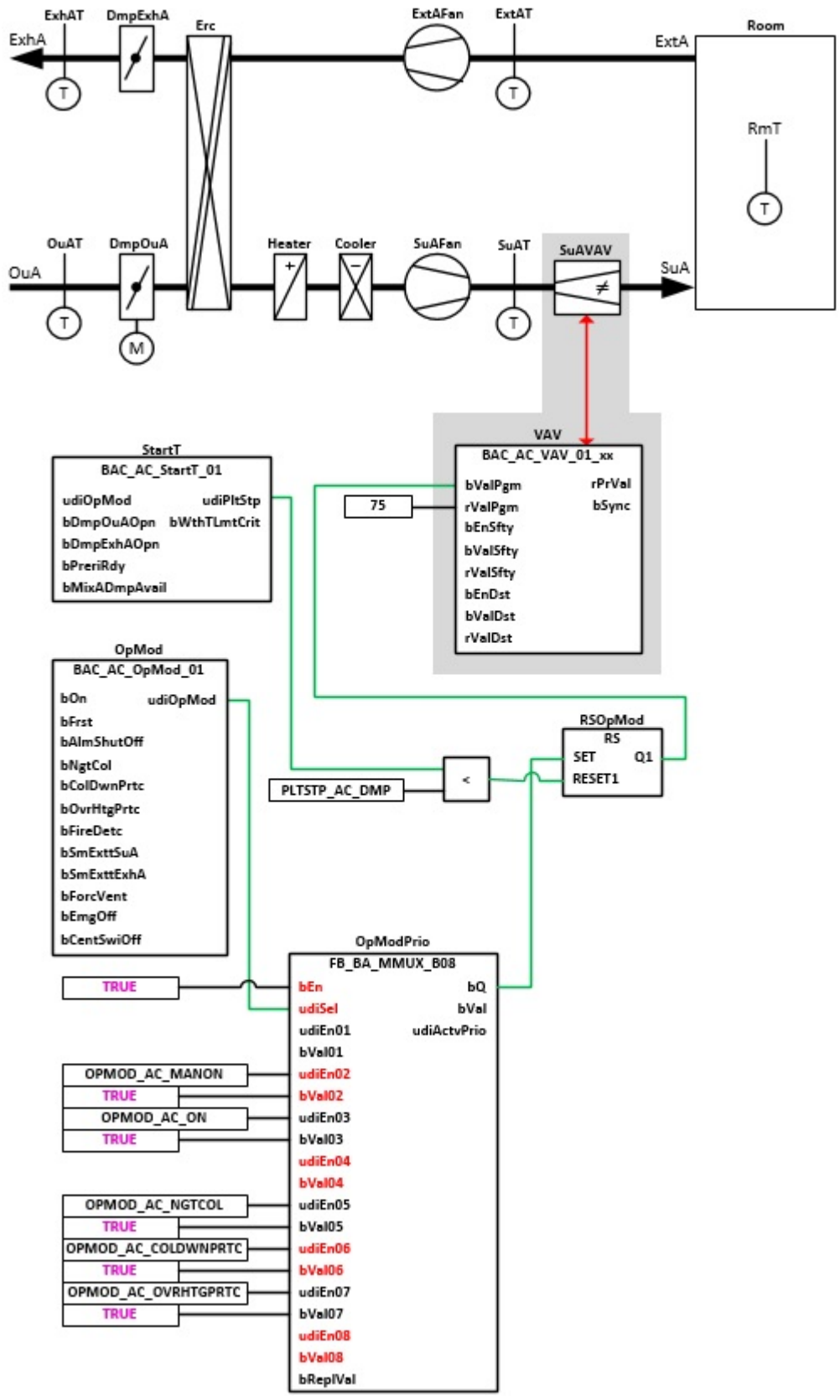
The template **BAC_AC_VAV_01_xxx** is available in different versions.
The volume flow controller versions are identified by means of a key. The identification key is derived from the table below.

| Optionen | Forced control BO | Feedback of the volume flow or the damper position | mechanical priority operation position feedback potentiometer | mechanical priority operation position feedback hand switch |
|---|---|---|---|---|
| **Instanz** | **Force** | **Fdb** | **FdbOut** | **LocSwi** |
| **Datenpunkt Typ** | **BO** | **AI** | **AI** | **BI** |
| | **8** | **4** | **2** | **1** |
| BAC_AC_VAV_01_000 | 0 | 0 | 0 | 0 |
| BAC_AC_VAV_01_003 | 0 | 0 | 1 | 1 |
| BAC_AC_VAV_01_004 | 0 | 1 | 0 | 0 |
| BAC_AC_VAV_01_007 | 0 | 1 | 1 | 1 |
| BAC_AC_VAV_01_008 | 1 | 0 | 0 | 0 |
| BAC_AC_VAV_01_011 | 1 | 0 | 1 | 1 |
| BAC_AC_VAV_01_012 | 1 | 1 | 0 | 0 |
| BAC_AC_VAV_01_015 | 1 | 1 | 1 | 1 |

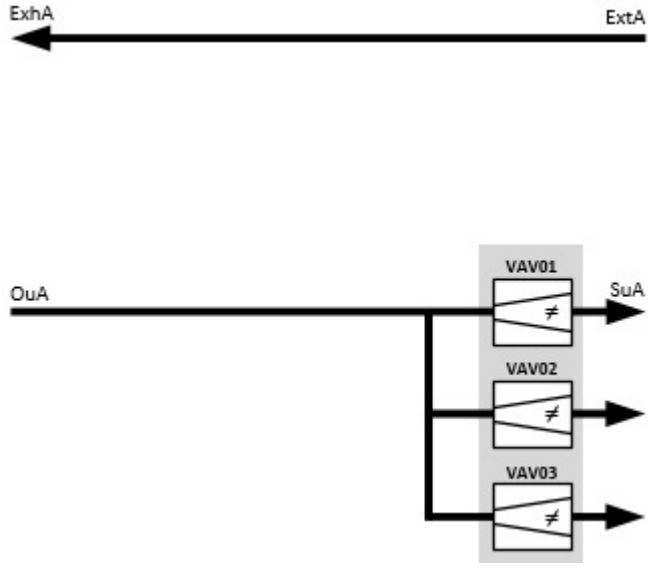
### Interface



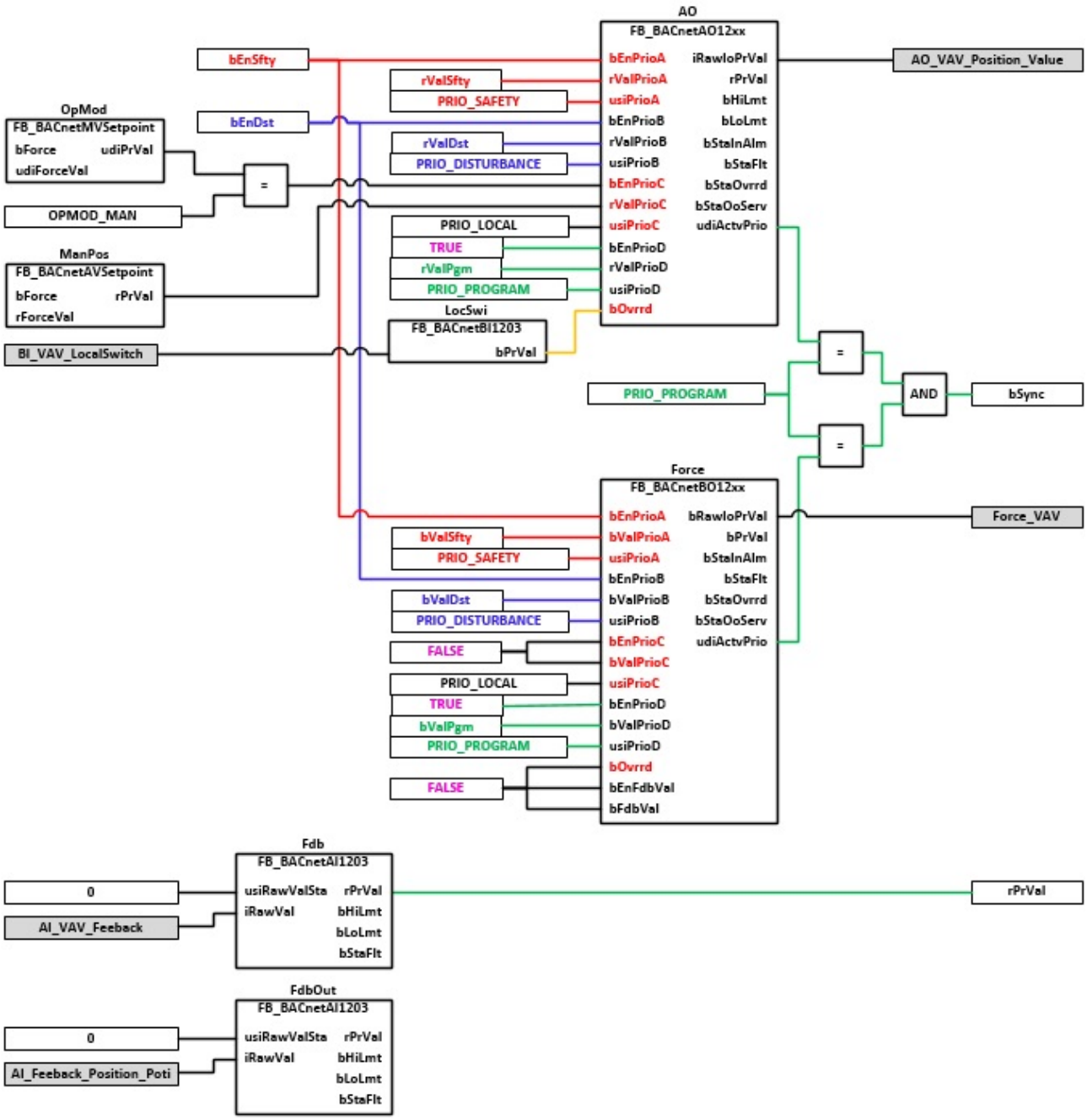
**Plant diagram 01**



**Plant diagram 02**



### Block diagram version BAC_AC_VAV_01_15




### VAR_INPUT

```
bValPgm     : BOOL;
rValPgm     : REAL;
bEnSfty     : BOOL;
bValSfty    : BOOL;
rValSfty    : REAL;
bEnDst      : BOOL;
bValDst     : BOOL;
rValDst     : REAL;
```

**bValPgm:** binary value program priority

**rValPgm:** analog value program priority

**bEnSfty:** safety priority enable

**bValSfty:** binary value safety priority

**rValSfty:** analog value safety priority

**bEnDst:** disturbance priority enable. This input can be used to connect process feedback, for example.

**bValDst:** binary value disturbance priority. This input can be used to connect process feedback, for example.

**rValDst:** analog value disturbance priority

### VAR_OUTPUT

```
rPrVal    : REAL;
bSync     : BOOL;
bForce    : BOOL;
```

**rPrVal :** Current position of the volume flow controller

**bSync:** Output of a pulse to synchronise the PID controller associated with the volume flow controller during reset from manual to automatic operation to the current valve position.
The synchronising pulse **bSync** should only be used if the template used contains the valve position feedback **Fdb**.

**bForce :** The volume flow is under forced control from the binary output.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task | | | |
|---|---|---|---|---|---|---|
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the volume flow controller via the MCL or a local operator display | | | |
| ManPos | FB_BACnetAVSetpoint [▶ 69] | | AV object for entering the control value for the volume flow controller with manual override | | | |
| Fdb | FB_BACnetAI1203 [▶ 49] | X | AI object for logging the position feedback from the volume flow controller | | | |
| FdbOut | FB_BACnetAI1203 [▶ 49] | X | AI object for logging the mechanical priority operation position feedback potentiometer | | | |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation feedback hand switch | | | |
| AO | FB_BACnetAO1203 [▶ 53] | | AO object for controlling the volume flow controller | | | |
| | | | **Priority:** | **Enable** | **Value** | **Comment** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** | |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** | |

| Instance | Type | op-tional | Task | | | |
|---|---|---|---|---|---|---|
| | | | PRIO_LOCAL (8) | **OpMod_udiPrVal** = **OPMOD_MAN** | **ManPos_rPrVal** | In manual mode, value of AV object **ManPos** |
| | | | PRIO_PROGRAM (15) | TRUE | Input **rValPgm** | Value of input **rValPgm** (e.g. from a controller) |
| BO | FB_BACnetBO1203 [▶ 81] | X | BO object for forced control of the volume flow controller | | | |
| | | | **Priority:** | **Enable** | **Value** | |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** | |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **bValDst** | |
| | | | PRIO_LOCAL (8) | | | |
| | | | PRIO_PROGRAM (15) | TRUE | Input **bValPgm** | |
| | EQ, EQ, AND | | Value of the network is TRUE, if the active priority is **PRIO_PROGRAM** (15). Can be used for synchronizing the controller on return to automatic mode | | | |
| TLogAO | FB_BACnetTLog1201 [▶ 135] | | Logs the present value of the AO object | | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process image | |
|---|---|---|---|---|
| AO | INT | X | Output | Analog output control value volume flow controller |
| AO_State | USINT | X | Input | Status byte analog output |
| AI_Feedback | INT | X | Input | Analog input position feedback |
| AI_Feedback_State | USINT | X | Input | Analog input position feedback status byte |
| BO_Force | BOOL | X | Output | Digital output forced control |
| AI_Feeback_Poti | INT | X | Input | Analog input - manual potentiometer – feedback - control value |
| BI_LocalSwitch | BOOL | X | Input | Digital input – switch manual pump - message - manual/auto |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0 | |

## 9.48 BAC_AC_OpMod_01

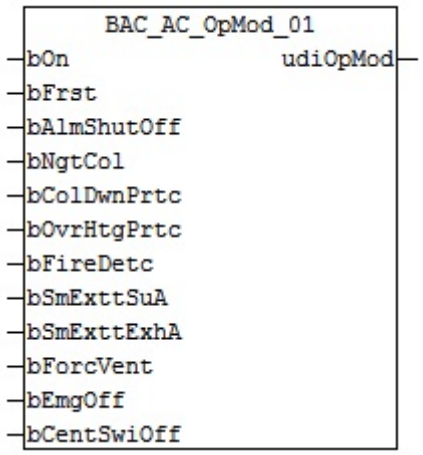
### Functional description

The template **BAC_AC_OpMod_01** prioritizes different events or commands of an air-conditioning plant, such as fire alarm, request of central switching or schedule and writes a resulting operation mode or resulting system status to the variable **udiOpMod**.
The devices and aggregates respond individually, depending on **udiOpMod**, so that the whole plant adjusts itself according to the current operation mode.

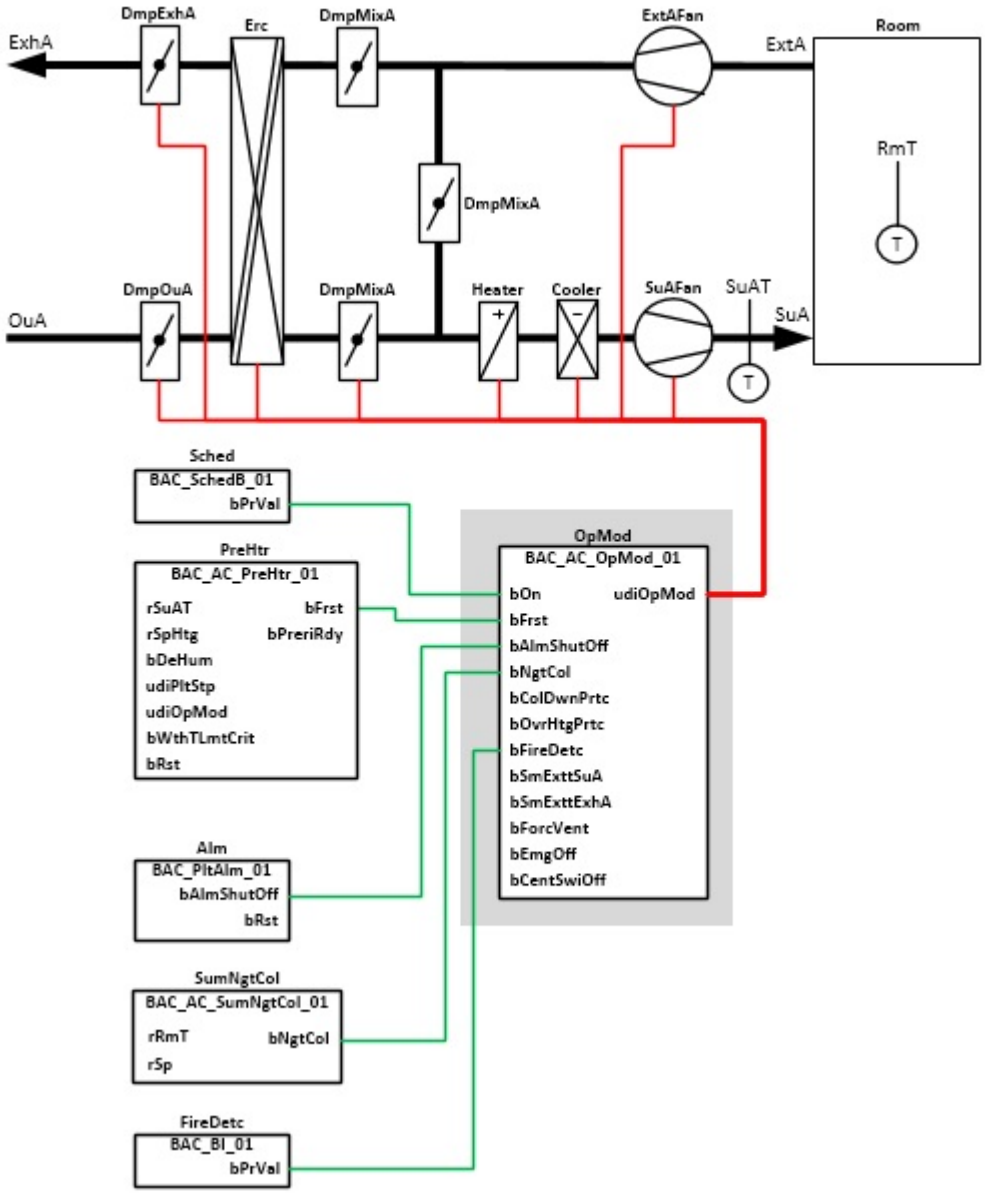
Sample: 1
The mixed-air system BAC_AC_MixAT_01 [▶ 475] switches to a fresh air rate of 100%, if **udiOpMod** is in summer night cooling mode (**udiOpMod** = 11).

Sample: 2
The supply air and extract air fan switches on if forced ventilation is enabled. Controlled fans crank up to a speed of 100%.

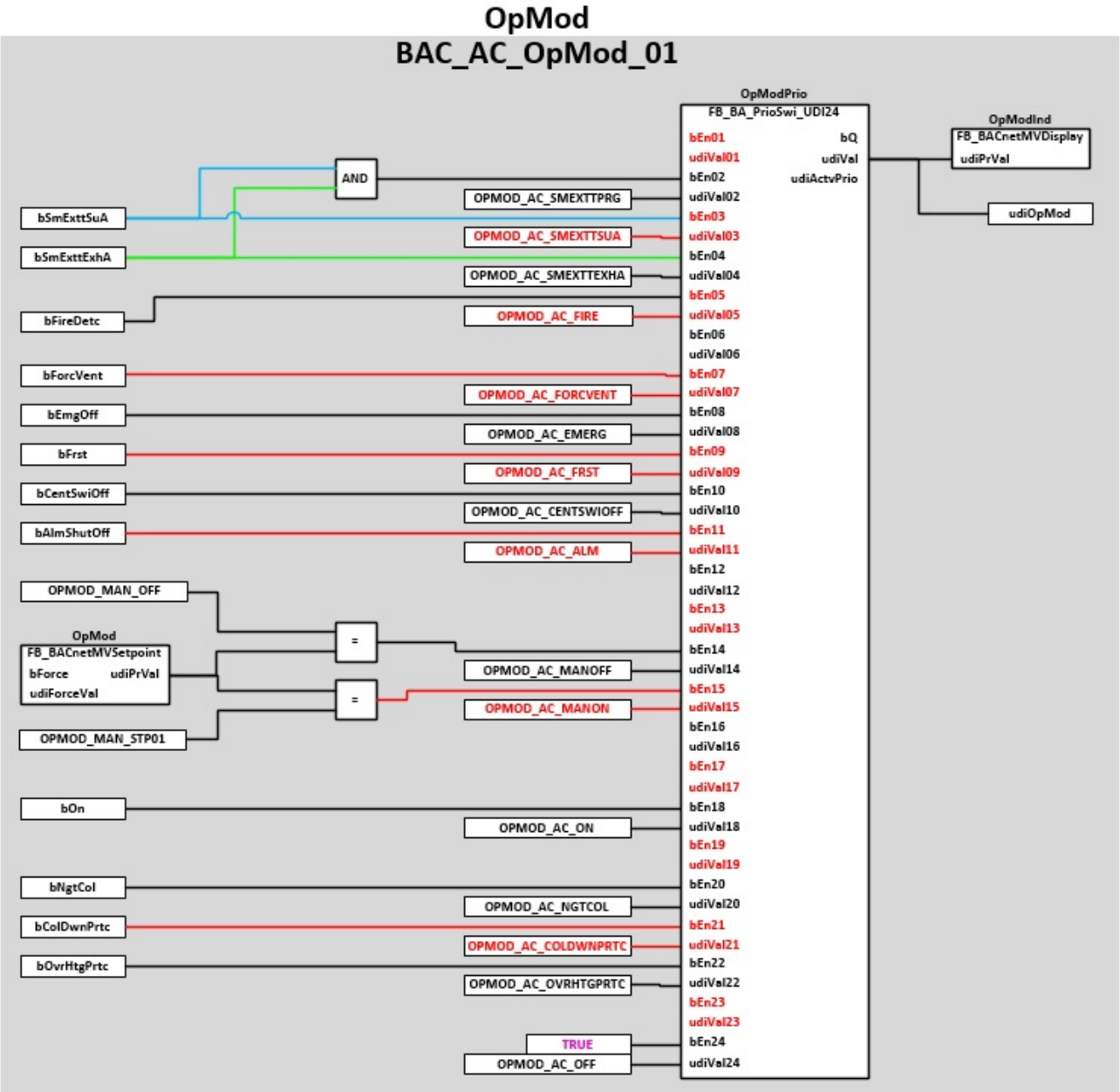
**Interface**



### System diagram



### Block diagram




**The following system states or operation modes are described via the variable :** *udiOpMod*

| udiOpMod | | | |
|---|---|---|---|
| OPMOD_AC_OFF [▶ 357] | 1 | Off | Plant shutdown |
| OPMOD_AC_ON [▶ 357] | 2 | On | Plant startup |
| OPMOD_AC_EMERG [▶ 357] | 3 | Emergency | Plant shutdown |
| OPMOD_AC_MANOFF [▶ 357] | 4 | Manual off | Plant shutdown |
| OPMOD_AC_MANON [▶ 357] | 5 | Manual on | Plant startup |
| OPMOD_AC_FRST [▶ 357] | 6 | Frost | Plant shutdown |
| OPMOD_AC_SMEXTTPRG [▶ 357] | 7 | Smoke extraction program | Plant startup |
| OPMOD_AC_SMEXTTSUA [▶ 357] | 8 | Smoke extraction supply | Plant supply air on |

| | | | |
|---|---|---|---|
| OPMOD_AC_SMEXTTEXHA [▶ 357] | 9 | Smoke extraction exhaust | Plant extract air on |
| OPMOD_AC_FIRE [▶ 357] | 10 | Fire | Plant shutdown |
| OPMOD_AC_NGTCOL [▶ 357] | 11 | Night cooling | Plant startup |
| OPMOD_AC_COLDWNPRTC [▶ 357] | 12 | Cool down protection | Plant startup |
| OPMOD_AC_OVRHTGPRTC [▶ 357] | 13 | Overheating protection | Plant startup |
| OPMOD_AC_ALM [▶ 357] | 14 | Alarm | Plant shutdown |
| OPMOD_AC_FORCVENT [▶ 357] | 15 | Forced ventilation | Plant startup |
| OPMOD_AC_CENTSWIOFF [▶ 357] | 16 | Central switch-off | Plant shutdown |

### VAR_INPUT

```
bOn             : BOOL;
bFrst           : BOOL;
bAlmShutOff     : BOOL;
bNgtCol         : BOOL;
bColDwnPrtc     : BOOL;
bOvrHtgPrtc     : BOOL;
bFireDetc       : BOOL;
bSmExttSuA      : BOOL;
bSmExttExuA     : BOOL;
bForcVent       : BOOL;
bEmgOff         : BOOL;
bCentSwiOff     : BOOL;
```

**bOn:** Request from timer program

**bFrst:** Frost protection program active

**bAlmShutOff:** Collective fault message - shut down plant

**bNgtCol:** Request from summer night cooling program

**bColDwnPrtc:** Request from cooling protection program

**bOvrHtgPrtc:** Request from overheating protection program

**bFireDetc:** Fire alarm message from fire alarm center

**bSmExttSuA:** Additional flow requested by supply air section of the plant for smoke extraction.

**bSmExttExhA:** Smoke extraction with extract air section of the plant requested.

**bForcVent:** Forced ventilation request

**bEmgOff:** Emergency Stop

**bCentSwiOff:** Central shutdown

### VAR_OUTPUT

```
udiOpMod        : UDINT;
```

**udiOpMod:** Output of the current plant operation mode.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| OpMod | FB_BACnetMVSetpoint [▶ 130] | The BACnet MV object serves as plant selector switch. The MV object can be used to switch the plant on or off, independent of the automatic programs. |
| OpModPrio | FB_BA_PrioSwi_UDI24 [▶ 213] | The function block logs all commands, prioritizes them and outputs the result at output **udiVal**. |
| OpModInd | FB_BACnetMVDisplay [▶ 129] | The BACnet MV object indicates the current plant operation mode. |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.49 BAC_AC_SeqH_01

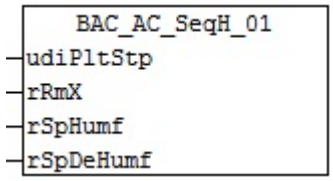
### Functional description

The template **BAC_AC_SeqH_01** is responsible for starting the supply air sequence control for humidification and dehumidification of an air-conditioning plant.

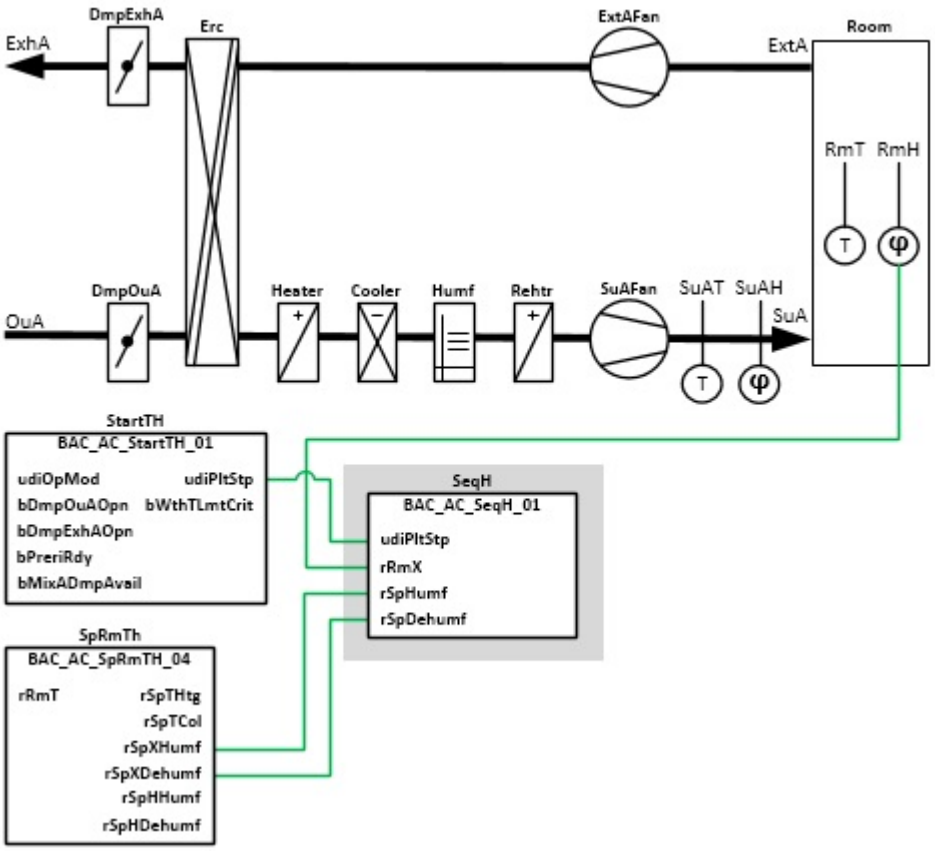
On startup of the air-conditioning plant the system determines whether to start with the humidification sequence or the dehumidification sequence. The start sequence is selected depending on the plant steps and the absolute room humidity.

Only one element of the sequence can be in control. When the output of a controlling sequence element reaches Y Min or Y Max, the control is transferred to the next sequence controller that is ready for switch-on.

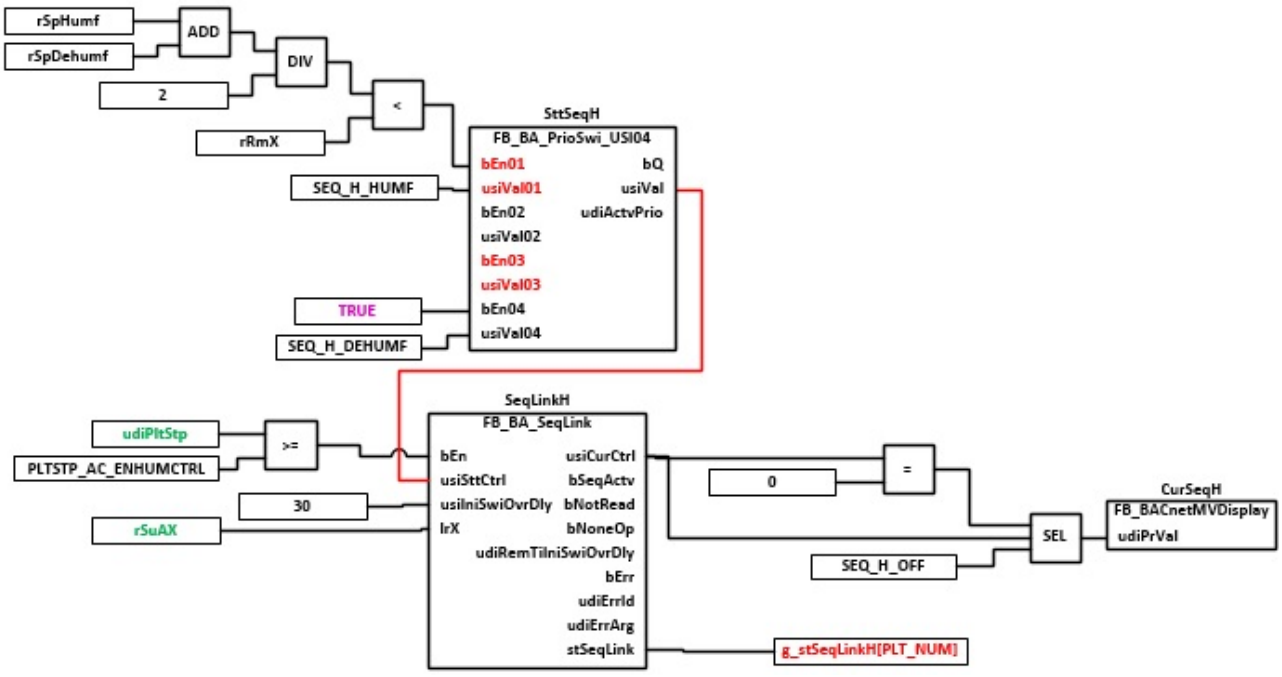
### Interface



### System diagram

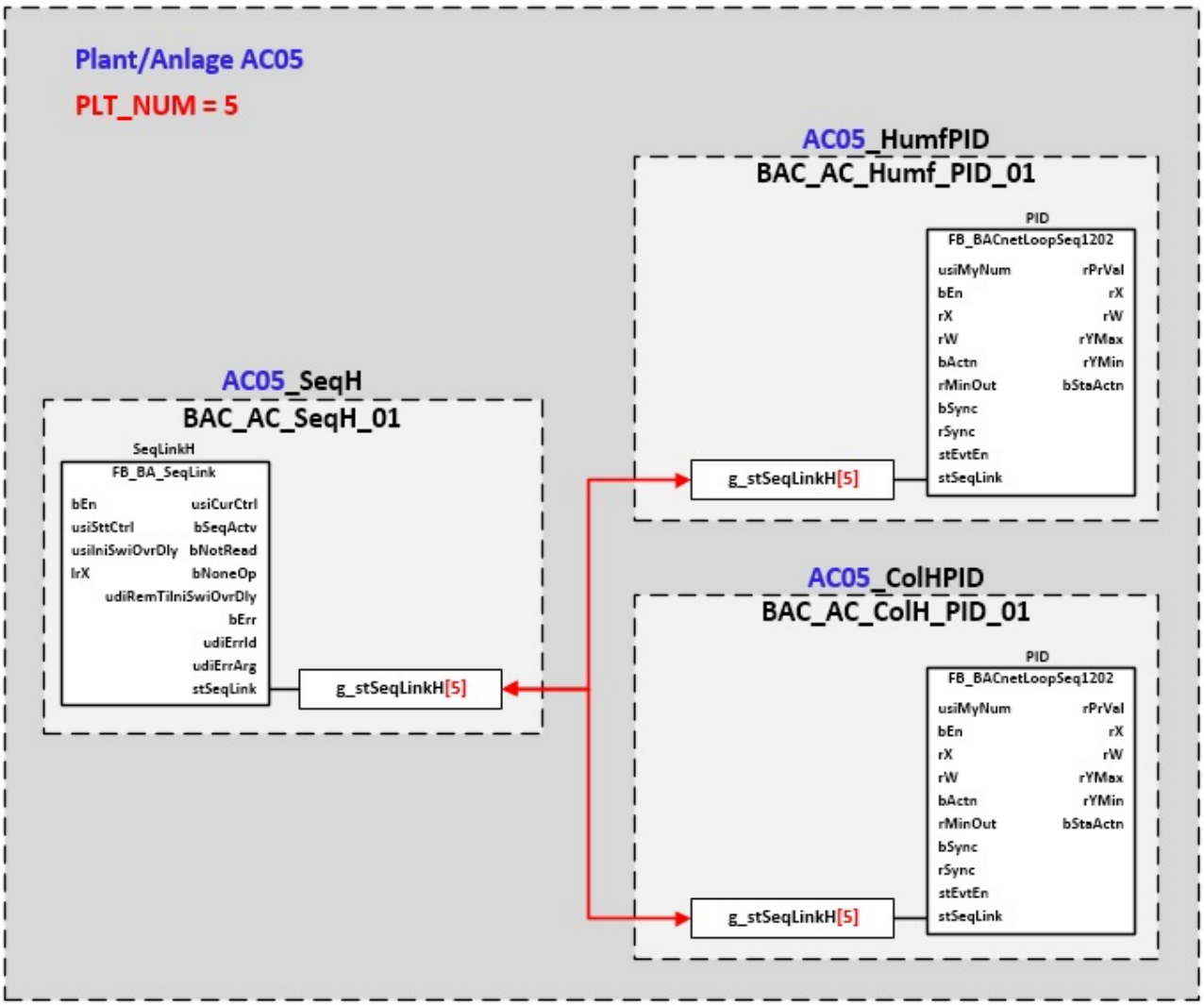



### Block diagram



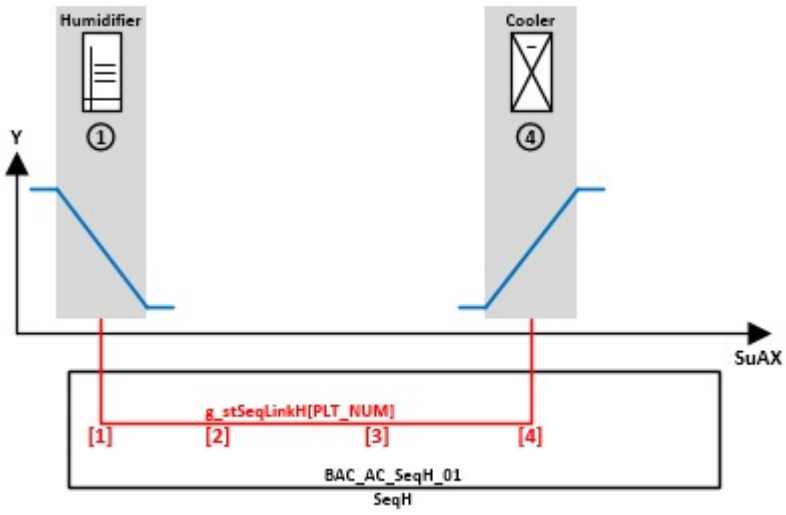

### Linking of the global sequence link structure g_stSeqLinkH[PLT_NUM]

g_stSeqLinkH[PLT_NUM] [▶ 357]




### Order of the sequences

The sequence of the sequence controllers of the individual aggregates must correspond to the sequence order.
The assignment of the sequence numbers does not have to be continuous, there may be free connections.
Example: 1 = humidifier, 4 = cooler/dehumidifier




### Globally defined sequence numbers of the sequence controllers:

| | | |
|---|---|---|
| SEQNUM_H_HUMF [▶ 357] | 1 | Sequenznummer Befeuchter//Sequence number humidifier |

| | | |
|---|---|---|
| SEQNUM_H_MIX [▶ 357] | 2 | Sequenznummer Mischluft//Sequence number mixed air |
| SEQNUM_H_ERC [▶ 357] | 3 | Sequenznummer Energierückgewinnung//Sequence number energy recovery |
| SEQNUM_H_DEHUMF [▶ 357] | 4 | Sequenznummer Entfeuchter//Sequence number dehumidifier |
| SEQNUM_H_OFF [▶ 357] | 5 | kein Sequenzregler aktiv//no sequence controller active |

### VAR_INPUT

```
udiPltStp    : UDINT;
rRmX         : REAL;
rSpHumf      : REAL;
rSpDehumf    : REAL;
```

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartTH_01 [▶ 535]

**rRmX:** Measured value room humidity **(absolute humidity**)

**rSpHumf:** Set value humidification **(absolute humidity)**

**rSpDehumf:** Set value dehumidification (**absolute humidity)**

### VAR CONSTANT

```
PLT_NUM      : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkH[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| SttSeqH<br>ADD, DIV, < | FB_BA_PrioSwi_USI04 [▶ 213] | The priority switch is used to select the start sequence.<br>Prio 1: bEN01<br>If the mean value of the setpoints **rSpHumf** / **rSpDehumf** is smaller than the absolute room humidity, the sequence starts with humidification. ( SEQNUM_H_HUMF [▶ 357]) started.<br><br>Prio 4: bEN04<br>If case Prio 1 (SEQNUM_H_HUMF [▶ 357]) does not occur, the dehumidification sequence starts (cooler) (SEQNUM_H_Dehumf) (SEQNUM_H_Dehumf [▶ 357]) |
| SeqLinkH | FB_BA_SeqLink [▶ 168] | The function block **SeqLinkH** is the core of the template BAC_SeqH_01. The sequence linker is linked with all supply air controllers of the sequence via the global data structure **g_stSeqLinkH[PLT_NUM]**. The central control element is |

| Instance | Type | Task |
|---|---|---|
| | | responsible for switching between the sequence controllers and starting the control sequence.<br>The sequence linker is enabled at input *bEn* via the steps during startup of the ventilation system if **udiPltStp**>= of the global constant PLTSTP_AC_ENHUMCTRL [▶ 357]. |
| CurSeqH | FB_BACnetMVDisplay [▶ 129] | The MV object indicates the currently active sequence controller. |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.50 BAC_AC_SeqT_01

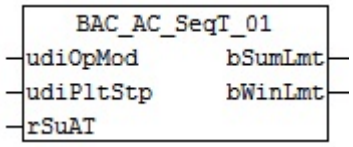
**Functional description**

The template **BAC_AC_SeqT_01** is responsible for starting the supply air temperature sequence control of an air-conditioning plant.

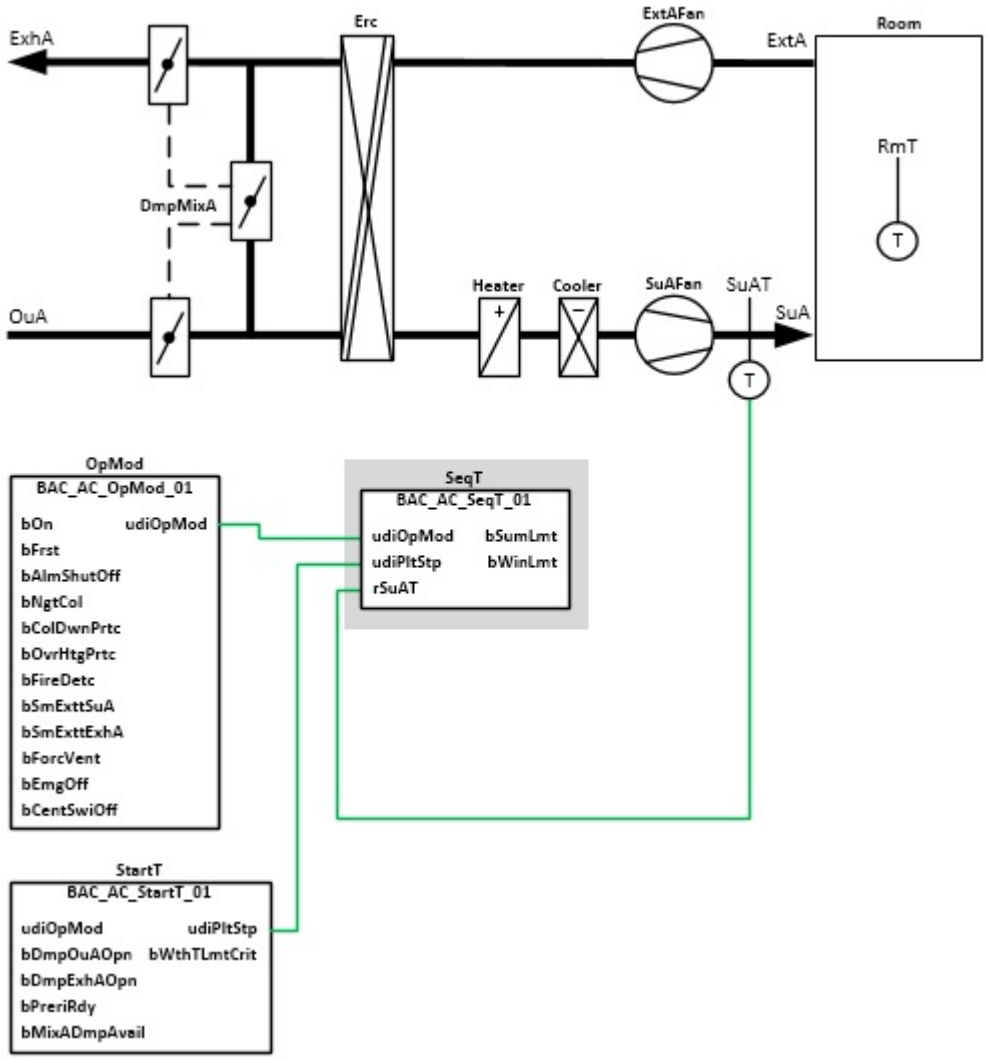
On startup of the air-conditioning plant the system determines whether to start with the heating, cooling or heat recovery sequence. The start sequence is selected depending on the plant operation mode and the outside temperature.

Only one element of the sequence can be in control. When the output of a controlling sequence element reaches Y Min or Y Max, the control is transferred to the next sequence controller that is ready for switch-on.

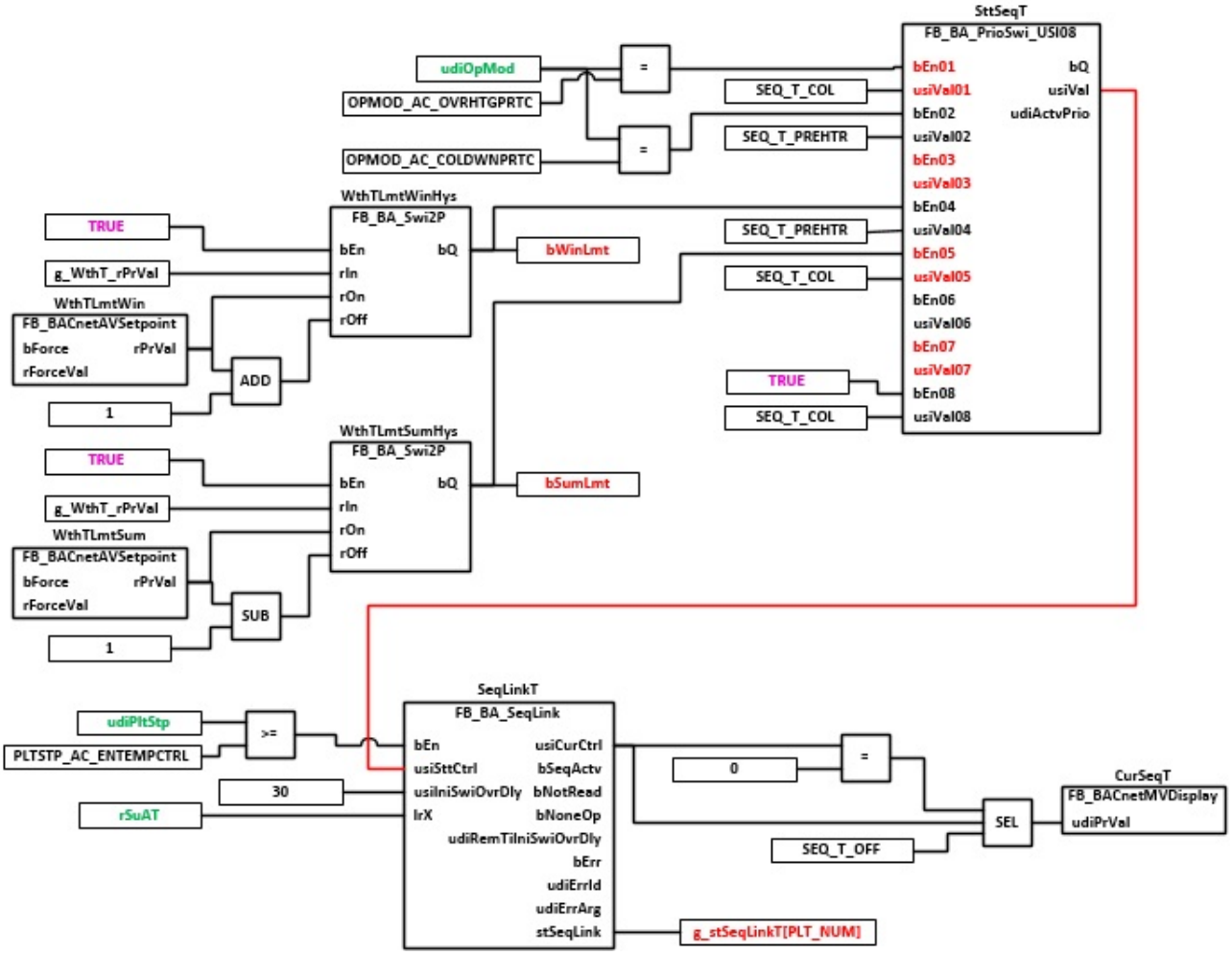
**Interface**



### System diagram

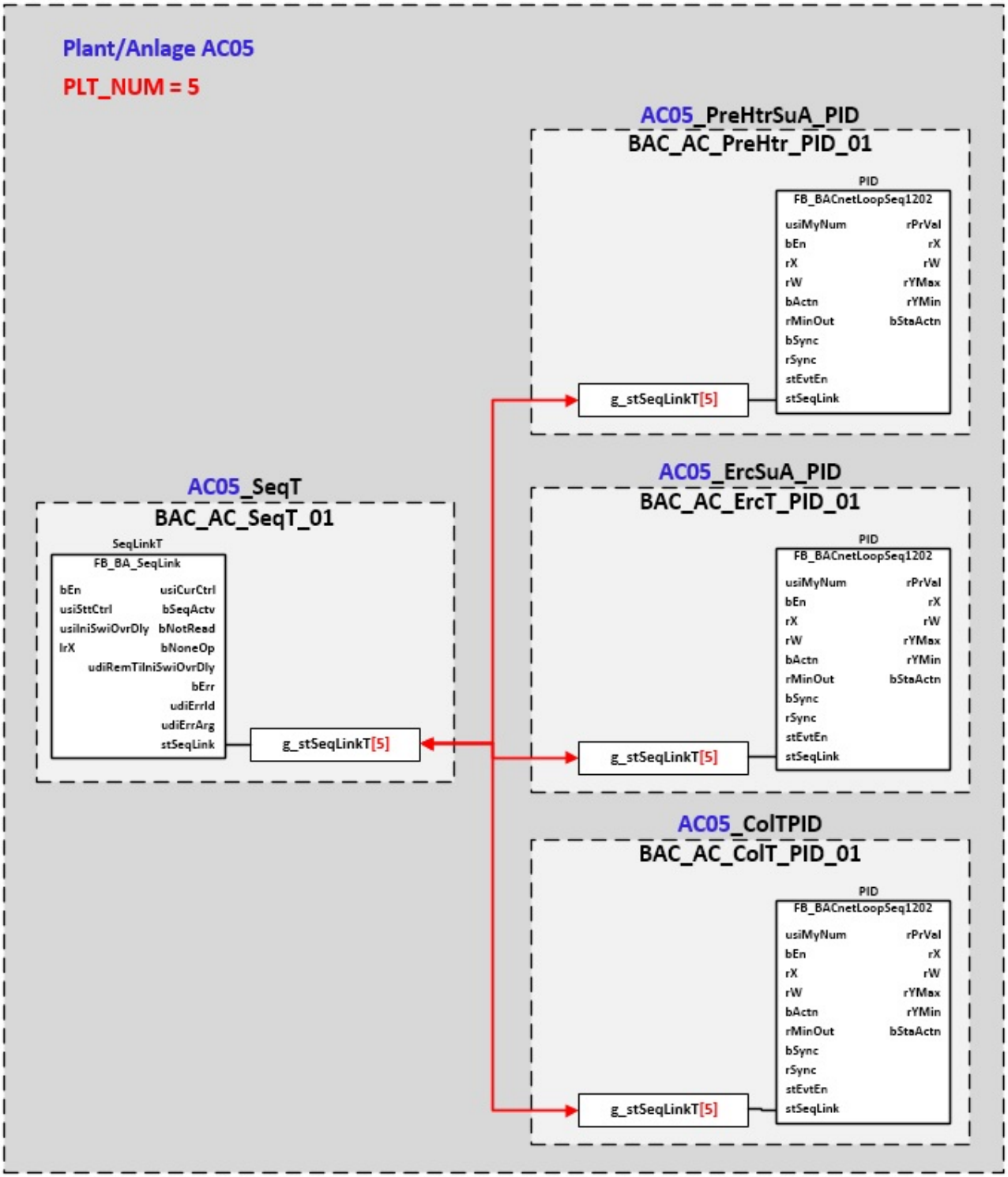


### Block diagram




### Linking of the global sequence link structure g_stSeqLinkT[PLT_NUM]

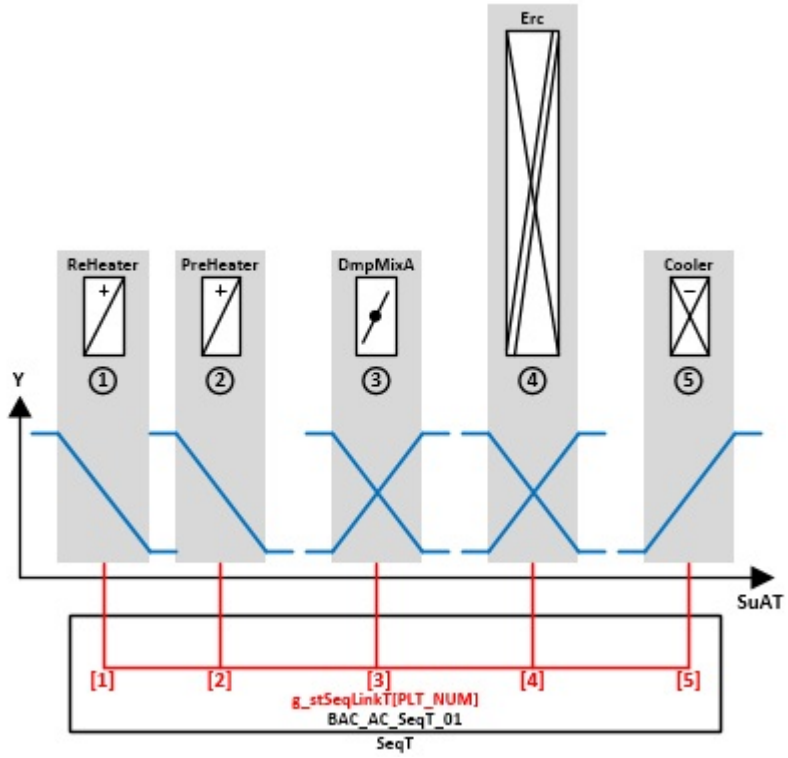
g_stSeqLinkT[PLT_NUM] [▶ 357]




### Order of the sequences

The order of the control sequence is specified by the globally defined variables. If necessary, the sequence order can be adjusted slightly.

For example if energy recovery is to take place upstream of the mixed air system.




**Globally defined sequence numbers of the sequence controllers:**

| | | |
|---|---|---|
| SEQNUM_T_REHTR [▶ 357] | 1 | Sequenznummer Nacherhitzer//Sequence number reheater |
| SEQNUM_T_PREHTR [▶ 357] | 2 | Sequenznummer Vorerhitzer//Sequence number preheater |
| SEQNUM_T_MIX [▶ 357] | 3 | Sequenznummer Mischluft//Sequence number mixed air |
| SEQNUM_T_ERC [▶ 357] | 4 | Sequenznummer Energierückgewinnung//Sequence number energy recovery |
| SEQNUM_T_COL [▶ 357] | 5 | Sequenznummer Kühler//Sequence number cooler |
| SEQNUM_T_OFF [▶ 357] | 6 | kein Sequenzregler aktiv//no sequence controller active |

### VAR_INPUT

```
udiOpMod    : UDINT;
udiPltStp   : UDINT;
rSuAT       : REAL;
```

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartT_01 [▶ 530]

**rSuAT:** Measured value supply air temperature

### VAR_OUTPUT

```
bSumLmt     : BOOL;
bWinLmt     : BOOL;
```

**bSumLmt:** Summer mode

**bWinLmt:** Winter mode

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by

means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

Within a ventilation system with sequence controller the plant number indicates which field from the global data structure **g_stSeqLinkT[PLT_NUM]** is used as link between the individual sequence controllers and the corresponding control function block**FB_BA_SeqLink**.

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| WthTLmtSum | FB_BACnetAVSetpoint [▶ 69] | AV object for input of an outside temperature value from which the air-conditioning plant starts in the cooling sequence. |
| WthTLmtWin | FB_BACnetAVSetpoint [▶ 69] | AV object for input of an outside temperature value from which the air-conditioning plant starts in the heating sequence. |
| SttSeqT | FB_BA_PrioSwi_USI08 [▶ 213] | The priority switch is used to select the start sequence.<br><br>Prio 1: bEN01<br>In overheating protection mode (**OPMOD_AC_OVRHTGPRTC**), the system invariably starts with the cooling sequence ( **SEQNUM_T_COL**).<br><br>Prio 2: bEN02<br>In cooling protection mode (**OPMOD_AC_COLDWNPRTC**), the system invariably starts with the heating sequence ( SEQNUM_T_PREHTR).<br><br>Prio 4: bEN04<br>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 plant starts with the control sequence of the preheater. (SEQNUM_T_PREHTR)<br><br>Prio 5: bEN05<br>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 plant starts with the control sequence of the cooler. (**SEQNUM_T_COL**)<br><br>Prio 8: bEN08<br>In the transitional periods between winter and summer, the inputs of **bEn04** and **bEn05** are FALSE on account of the weather. In this case priority 8 applies at the priority switch. It means that the air-conditioning plant starts in the energy recovery sequence (SEQNUM_T_ERC). |
| SeqLinkT | FB_BA_SeqLink [▶ 168] | The function block **SeqLinkT** is the core of the template **BAC_AC_SeqT_01**. The sequence linker is linked with all supply air controllers of the sequence via the global data structure **g_stSeqLinkT[PLT_NUM]** . It is the central control element, and responsible for switching between the sequence controllers and starting of the control sequence. |

| Instance | Type | Task |
|---|---|---|
| | | The sequence linker is enabled at input *bEn* via the steps during startup of the ventilation system, if **udiPltStp** is >= the global constant PLTSTP_AC_ENTEMPCTRL **.** |
| CurSeqT | FB_BACnetMVDisplay [▶ 129] | The MV object indicates the currently active sequence controller. |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

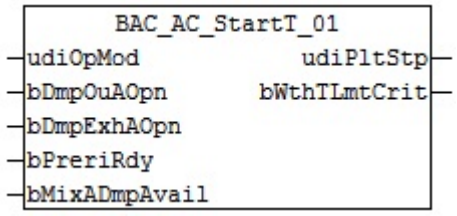
## 9.51 BAC_AC_StartT_01

### Functional description

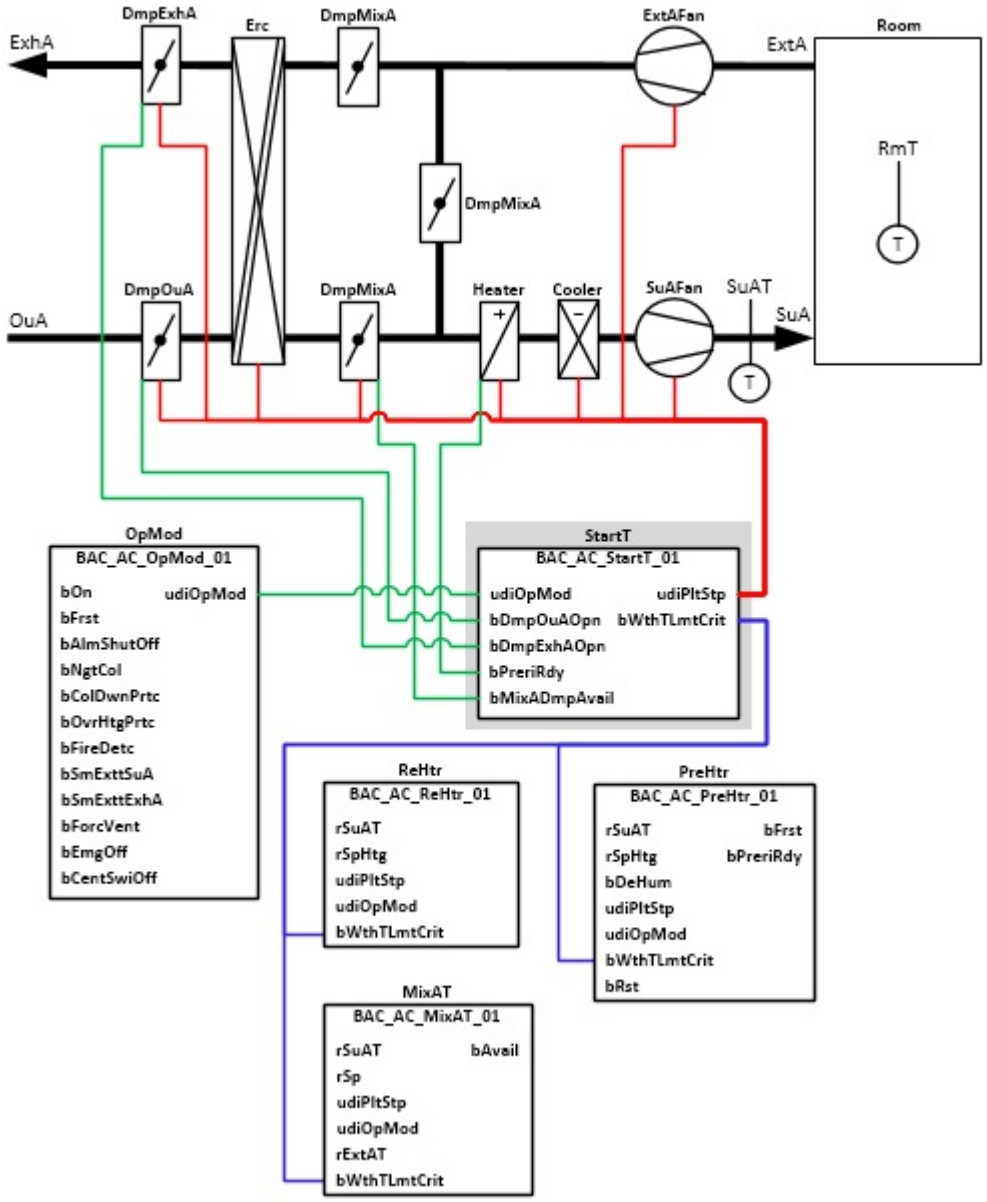
The template BAC_AC_StartT_01 is responsible for step-by-step startup of an air-conditioning plant without humidification and dehumidification.

The steps during plant startup are described through the variable **udiPltStp**. The steps are transferred to all units of the plant, so that each unit is integrated in the startup procedure of the air-conditioning plant.

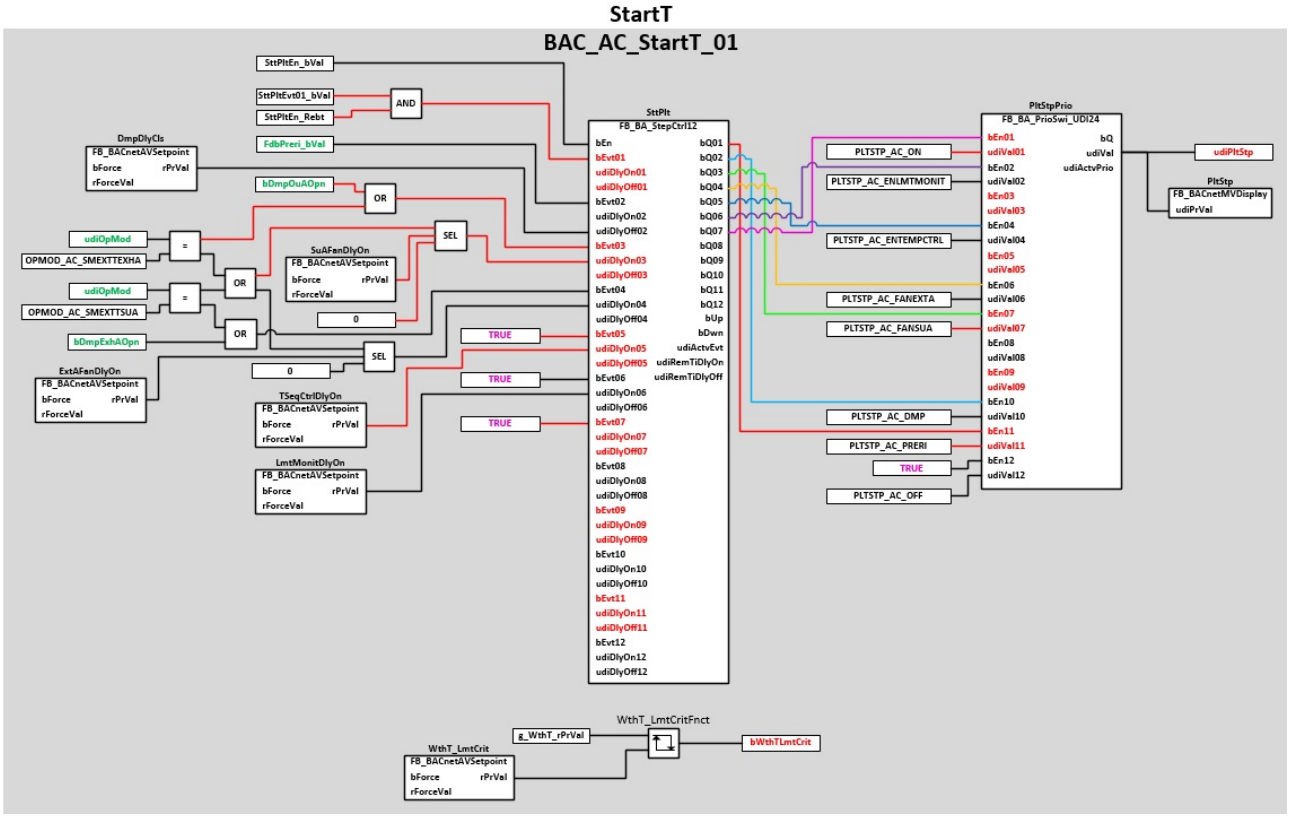
### Interface



### System diagram



### Block diagram




**The following plant steps are described via the variable udiPltStp:**

| udiPltStp | |
|---|---|
| PLTSTP_AC_OFF [▶ 357] | Off |
| PLTSTP_AC_PRERI [▶ 357] | pre-rincing heater |
| PLTSTP_AC_DMP [▶ 357] | opening damper |
| PLTSTP_AC_FANSUA [▶ 357] | supply air fan on |
| PLTSTP_AC_FANEXTA [▶ 357] | extract air fan on |
| PLTSTP_AC_ENTEMPCTRL [▶ 357] | ennabling temperature control, (sequence linker) |
| PLTSTP_AC_ENLMTMONIT [▶ 357] | limit monitoring |
| PLTSTP_AC_ON [▶ 357] | plant On (started) |

### VAR_INPUT

```
udiOpMod        : UDINT;
bExhADmpOpn     : BOOL;
bOuADmpOpn      : BOOL;
bPreriRdy       : BOOL;
bMixADmpAvail   : BOOL;
```

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

**bOuADmpOpn:** Feedback outside air damper open

**bExhADmpOpn:** Feedback exhaust air damper open

**bPreriRdy:** Prerinsing at heating coil complete

**bMixADmpAvail**: Mixed air system for startup process available

### VAR_OUTPUT

```
udiPltStp       : UDINT;
bWhtTLmtCrit    : BOOL;
```

**udiPltStp:** Output of the current system startup steps

**bWhtTLmtCrit:** The outside temperature is below a critical value, so that prerinsing of the heating coil is required on startup of the air-conditioning plant. The pumps for the preheater and the reheater are switched on. The ramp function for the the mixed air dampers is activated on plant startup.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| DmpDlyCls | FB_BACnetAVSetpoint [▶ 69] | AV object delays closing of the dampers during shutdown of the air-conditioning plant, to consider the overrun time of the fans during system shutdown. |
| SuAFanDlyOn | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for the switch-on delay of the supply air fan when the air-conditioning plant is switched on |
| ExtAFanDlyOn | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for the switch-on delay of the outlet air fan when the air-conditioning plant is switched on |
| TSeqCtrlDlyOn | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for delayed control enable |
| LmtMonitDlyOn | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for delayed enable of the temperature sensor limit value monitoring |
| WthT_LmtCrit | FB_BACnetAVSetpoint [▶ 69] | AV object for entering an outside temperature value at which the air-conditioning plant should start with prerinsing the heating coil. |
| SttPlt | FB_BA_StepCtrl12 [▶ 249] | The function block **SttPlt** is the core of the system start program.<br>Input **bEn** is used for a general enable of the function block. If **bEn** is FALSE, all outputs of the function block are also FALSE.<br><br>**Step 1: Prerinsing of the heating coil PLTSTP_AC_PRERI**<br>At input **bEvt01**, startup of the air-conditioning plant initially starts with prerinsing of the heating coil.<br><br>**Step 2: Opening the outside and exhaust air damper PLTSTP_AC_DMP**<br>The input **bEvt02** is set when the prerinsing process is complete or is not required, due to high external temperatures.<br>This triggers the next startup step, opening of the dampers. |

| Instance | Type | Task |
|---|---|---|
| | | The condition for advancing to the open dampers step is formed by the function block **FdbPreri**.<br><br>**Step 3: Start supply air fan PLTSTP_AC_FANSUA**<br>At input **bEvt03** the limit switch for the outside air damper is connected. If the damper is open, step 3 of the supply air fan startup process is triggered. The startup of the supply air fan can be delayed at input **udiDlyOn03**. In smoke extraction mode the value for the fan startup delay is set to zero by the upstream selector. The program then moves to the next step without delay.<br><br>**Step 4: Start outlet air fan PLTSTP_AC_FANEXTA**<br>At input **bEvt04** the limit switch for the exhaust air damper is connected. If the damper is open, step 4 of the exhaust air fan startup process is triggered. The startup of the outlet air fan can be delayed at input **udiDlyOn04**. In smoke extraction mode the value for the fan startup delay is set to zero by the upstream selector. The program then moves to the next step without delay.<br><br>**Step 5: Enable supply air sequence control temperature PLTSTP_AC_ENTEMPCTRL**<br>The input **bEvt05** is constantly TRUE. This means that enable of the supply air sequence control is only delayed by the value of **udiDlyOn05**.<br><br>**Step 6: Enable sensor limit value monitoring PLTSTP_AC_ENLMTMONIT**<br>The input **bEvt06** is constantly TRUE. This means that enable of the supply air sequence control is only delayed by the value of **udiDlyOn06**.<br><br>**Step 7: Indicates that the ventilation system is fully operational PLTSTP_AC_ON**<br>The input **bEvt0**7 is constantly TRUE. This means that enable of the supply air sequence control is only delayed by the value of **udiDlyOn06**.<br><br>During shutdown of the air-conditioning plant, the step sequence from step 7 to step 3 takes place without delay. Closing of the dampers after the fans are switched off is delayed by the timer **DmpDlyCls**. Step 1, prerinsing of the heating coil, is skipped. |
| SttPltEn | FB_BA_MMUX_B04 [▶ 210] | The multiplexer **SttPltEn** generates the general enable for the start-up program, depending on the plant operation modes. If the plant operation mode changes during operation, the variable **SttPltEn_Rebt** at the comparison output becomes TRUE. As a result, the plant startup function block **SttPlt** is temporarily reset. The start program is set to a defined state and restarted according to the new operation mode. |
| SttPltEvt01 | FB_BA_MMUX_B24 [▶ 205] | The multiplexer **SttPltEvt01** generates the enable for starting the prerinsing process, depending on the plant operation modes |
| FdbPreri | FB_BA_MMUX_B08 [▶ 205] | The multiplexer **FdbPreri** generates the enable for opening the dampers.<br>In operation modes **OPMOD_AC_SMEXTTPRG**, **OPMOD_AC_SMEXTTSUA**, **OPMOD_AC_SMEXTTEXHA**, **OPMOD_AC_NGTCOL** and **OPMOD_AC_OVRHTGPRTC**, |

| Instance | Type | Task |
|---|---|---|
| | | TRUE is output at output **bVal**. Step 1 of the plant startup function block **SttPlt** is skipped in these operation modes. The prerinsing process is reported as complete.<br>For normal plant operation, opening of the dampers is triggered under the following conditions:<br>1. **bMixADmpAvail** = TRUE (A mixed-air system is present. The startup mode is realised via the mixed-air system)<br>2. **bPreriRdy** = TRUE (The prerinsing process is complete)<br>3. The outside temperature is uncritical. The output of function block **WthT_LmtCritFnct** is FALSE. |
| WthT_LmtCritFnct | FB_BA_Swi2P [▶ 146] | The function block **WthT_LmtCritFnct** is a two-point switch. It sets the output variable **bWthTLmtCrit** to TRUE or FALSE, depending on the outside temperature. The switching point for the two-point switch is specified via the AV object **WhtT_LmtCrit**. |
| PltStpPrio | FB_BA_PrioSwi_UDI12 [▶ 213] | The priority switch **PltStpPrio** evaluates the current steps during the startup process and relays them via the variable **udiVal** to the BACnet MV object **PltStp** for display at the management level. |
| PltStp | FB_BACnetMVDisplay [▶ 129] | The BACnet MV object indicates the current step during the startup process. |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

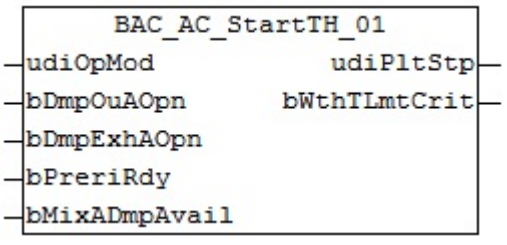
# 9.52 BAC_AC_StartTH_01

**Functional description**

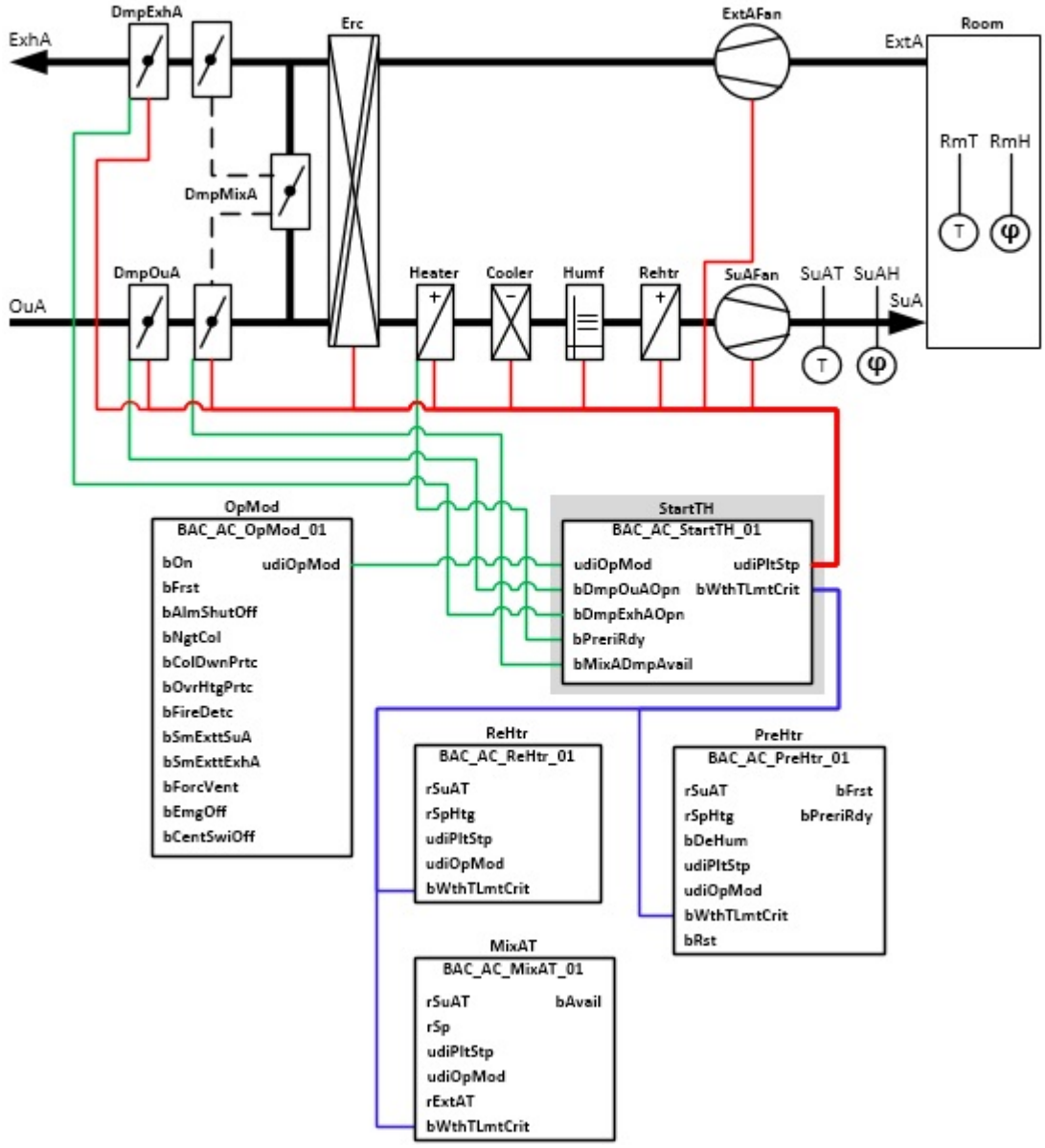
The template BAC_AC_StartTH_01 is responsible for step-by-step startup of an air-conditioning plant with humidification and dehumidification.

The steps during plant startup are described through the variable **udiPltStp**. The steps are transferred to all units of the plant, so that each unit is integrated in the startup procedure of the air-conditioning plant.

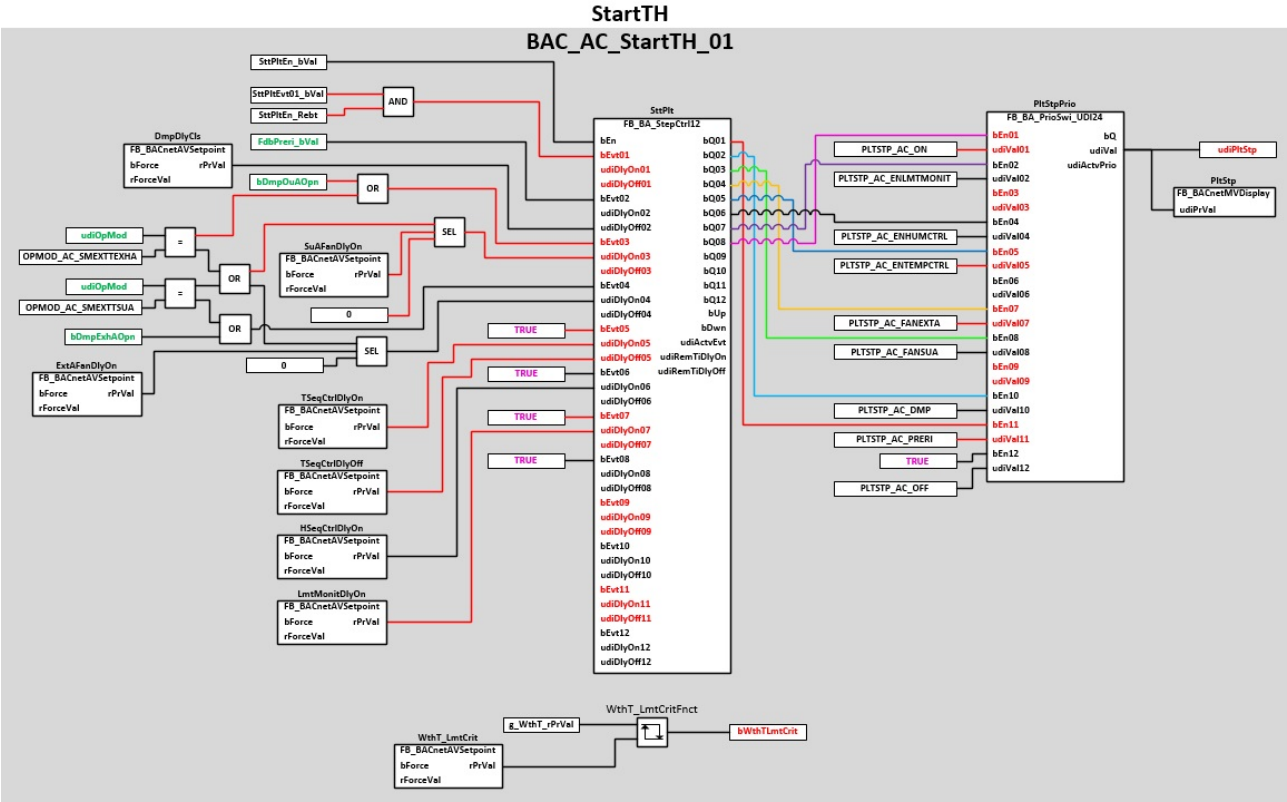
**Interface**



### System diagram



### Block diagram




**The following plant steps are described via the variable udiPltStp:**

| udiPltStp | |
|---|---|
| PLTSTP_AC_OFF [▶ 357] | Off |
| PLTSTP_AC_PRERI [▶ 357] | pre-rinse heater |
| PLTSTP_AC_DMP [▶ 357] | opening Damper |
| PLTSTP_AC_FANSUA [▶ 357] | supply air fan on |
| PLTSTP_AC_FANEXTA [▶ 357] | extract air fan on |
| PLTSTP_AC_ENTEMPCTRL [▶ 357] | enabling temperature control, (sequence link) |
| PLTSTP_AC_ENHUMCTRL [▶ 357] | enable humidity , (sequence link) |
| PLTSTP_AC_ENLMTMONIT [▶ 357] | enabling limit monitoring |
| PLTSTP_AC_ON [▶ 357] | On |

### VAR_INPUT

```
udiOpMod         : UDINT;
bExhADmpOpn      : BOOL;
bOuADmpOpn       : BOOL;
bPreriRdy        : BOOL;
bMixADmpAvail    : BOOL;
```

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

**bOuADmpOpn:** Feedback outside air damper open

**bExhADmpOpn:** Feedback exhaust air damper open

**bPreriRdy:** Prerinsing at heating coil complete

**bMixADmpAvail**: Mixed air system for startup process available

### VAR_OUTPUT

```
udiPltStp       : UDINT;
bWhtTLmtCrit    : BOOL;
```

**udiPltStp:** Output of the current system startup steps

**bWhtTLmtCrit:** The outside temperature is below a critical value, so that prerinsing of the heating coil is required on startup of the air-conditioning plant. The pumps for the preheater and the reheater are switched on. The ramp function for the mixed air dampers is activated on plant startup.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| DmpDlyCls | FB_BACnetAVSetpoint [▶ 69] | AV object delays closing of the dampers during shutdown of the air-conditioning plant, to take into account the overrun time of the fans during system shutdown. |
| SuAFanDlyOn | FB_BACnetAVSetpoint [▶ 69] | 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_BACnetAVSetpoint [▶ 69] | AV object for entering a value for the start-up delay of the exhaust air fan when the air-conditioning plant is switched on |
| TSeqCtrlDlyOn | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for delayed temperature control enable |
| TSeqCtrlDlyOff | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for delayed temperature control disable |
| HSeqCtrlDlyOn | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for delayed humidity control enable |
| LmtMonitDlyOn | FB_BACnetAVSetpoint [▶ 69] | AV object for entering a value for delayed enable of the temperature sensor limit value monitoring |
| WthT_LmtCrit | FB_BACnetAVSetpoint [▶ 69] | AV object for entering an outside temperature value at which the air-conditioning plant should start prerinsing the heating coil. |
| SttPlt | FB_BA_StepCtrl12 [▶ 249] | The function block **SttPlt** is the core of the plant start program. The function block is generally enabled at the input **bEn**. If **bEn** = FALSE, all outputs of the function block are forced to FALSE.<br><br>**Step 1: prerinsing of the heating coil** PLTSTP_AC_PRERI [▶ 357]<br>At the input **bEvt01** the startup of the air-conditioning plant is first started with the prerinsing of the heating coil. |

| Instance | Type | Task |
|---|---|---|
| | | **Step 2: opening the outside and exhaust air damper** PLTSTP_AC_DMP [▶ 357]<br>If the prerinsing process is completed or not required due to high outside temperatures, the input **bEvt02** is set.<br>This triggers the next startup step, the opening of the dampers. The condition for switching to the Open damper step is formed by the function block **FdbPreri**.<br><br>**Step 3: Starting the supply air fan** PLTSTP_AC_FANSUA [▶ 357]<br>The limit switch of the outdoor air damper is connected to the input **bEvt03**. If the damper is open, step 3 of the supply air fan startup process is triggered. The startup of the supply air fan can be delayed at input **udiDlyOn03**. In smoke extraction mode the value for the fan startup delay is set to zero by the upstream selector. Thus the program jumps to the next step without delay.<br><br>**Step 4: starting the exhaust air fan** PLTSTP_AC_FANEXTA [▶ 357]<br>The limit switch of the exhaust air damper is connected to the input **bEvt04**. If the damper is open, step 4 of the exhaust air fan startup process is triggered. The startup of the exhaust air fan can be delayed at input **udiDlyOn04**. In smoke extraction mode the value for the fan startup delay is set to zero by the upstream selector. Thus the program jumps to the next step without delay.<br><br>**Step 5: Enable supply air sequence control temperature** PLTSTP_AC_ENTEMPCTRL [▶ 357]<br>The input **bEvt05** is constantly TRUE. Thus the enable of the supply air sequence control is only delayed by the value of **udiDlyOn05**.<br><br>**Step 6: enable supply air sequence control humidity** PLTSTP_AC_ENHUMCTRL [▶ 357]<br>The input **bEvt0**6 is constantly TRUE. Thus, the enable of the supply air sequence control is only delayed by the value of **udiDlyOn0**6.<br><br>**Step 7: enable sensor limit value monitoring** PLTSTP_AC_ENLMTMONIT [▶ 357]<br>The input **bEvt0**7 is constantly TRUE. Thus the enable of the supply air sequence control is only delayed by the value of **udiDlyOn0**7.<br><br>**Step 8: indicates that the ventilation system is in full operation** PLTSTP_AC_ON [▶ 357]<br>**The** input **bEvt0**8 is constantly TRUE. Thus, the enable of the supply air sequence control is only delayed by the value of **udiDlyOn0**7.<br><br>When the air-conditioning plant is shut down, the humidity control (step 6) is switched off first and then, with a time delay, the temperature control (step 5). The switch-off delay of the temperature control prevents condensation of residual moisture in the air ducts and, as a result, the response of a hygrostat when the plant is switched off.<br><br>To bridge a overrun time of the fans, the closing of the dampers during shutdown of the plant is delayed by the timer **DmpDlyCls**. |

| Instance | Type | Task |
|---|---|---|
| SttPltEn | FB_BA_MMUX_B04 [▶ 210] | The multiplexer **SttPltEn** generates the general enable for the startup program, depending on the plant operation modes. If the plant operation mode changes during operation, the variable **SttPltEn_Rebt** at the comparison output becomes TRUE. This briefly resets the plant startup function block **SttPlt**. The start program is thus set to a defined state and restarted according to the new operation mode. |
| SttPltEvt01 | FB_BA_MMUX_B24 [▶ 205] | The multiplexer **SttPltEvt01** generates the enable for starting the prerinsing process, depending on the plant operation modes |
| FdbPreri | FB_BA_MMUX_B08 [▶ 205] | The multiplexer **FdbPreri** generates the enable for opening the dampers.<br>In the operation modes **OPMOD_AC_SMEXTTPRG**, **OPMOD_AC_SMEXTTSUA**, **OPMOD_AC_SMEXTTEXHA**, **OPMOD_AC_NGTCOL** and **OPMOD_AC_OVRHTGPRTC** TRUE is output at output **bVal**. Step 1 of the plant startup function block **SttPlt**is skipped in these operation modes. The prerinsing process is reported as completed.<br>For normal plant operation, the opening of the dampers is triggered under the following conditions:<br>1. **bMixADmpAvail** = TRUE (A mixed air system is present. Startup operation is realized by means of mixed air system)<br>2. **bPreriRdy** = TRUE (The prerinsing process is completed)<br>3. The outside temperature is uncritical. The output of function block **WthT_LmtCritFnct** is FALSE. |
| WthT_LmtCritFnct | FB_BA_Swi2P [▶ 146] | The function block **WthT_LmtCritFnct** is a two-point switch. It sets the output variable **bWthTLmtCrit** to TRUE or FALSE, depending on the outside temperature. The switching point for the two-point switch is specified via the AV object **WhtT_LmtCrit**. |
| PltStpPrio | FB_BA_PrioSwi_UDI12 [▶ 213] | The priority switch **PltStpPrio** evaluates the current steps during the startup process and relays them via the variable **udiVal** to the BACnet MV object **PltStp** for display at the management level. |
| PltStp | FB_BACnetMVDisplay [▶ 129] | The BACnet MV object indicates the current step during the startup process. |

### Version history

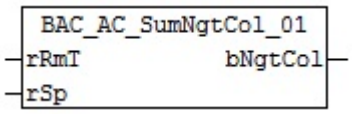
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.53 BAC_AC_SumNgtCol_01

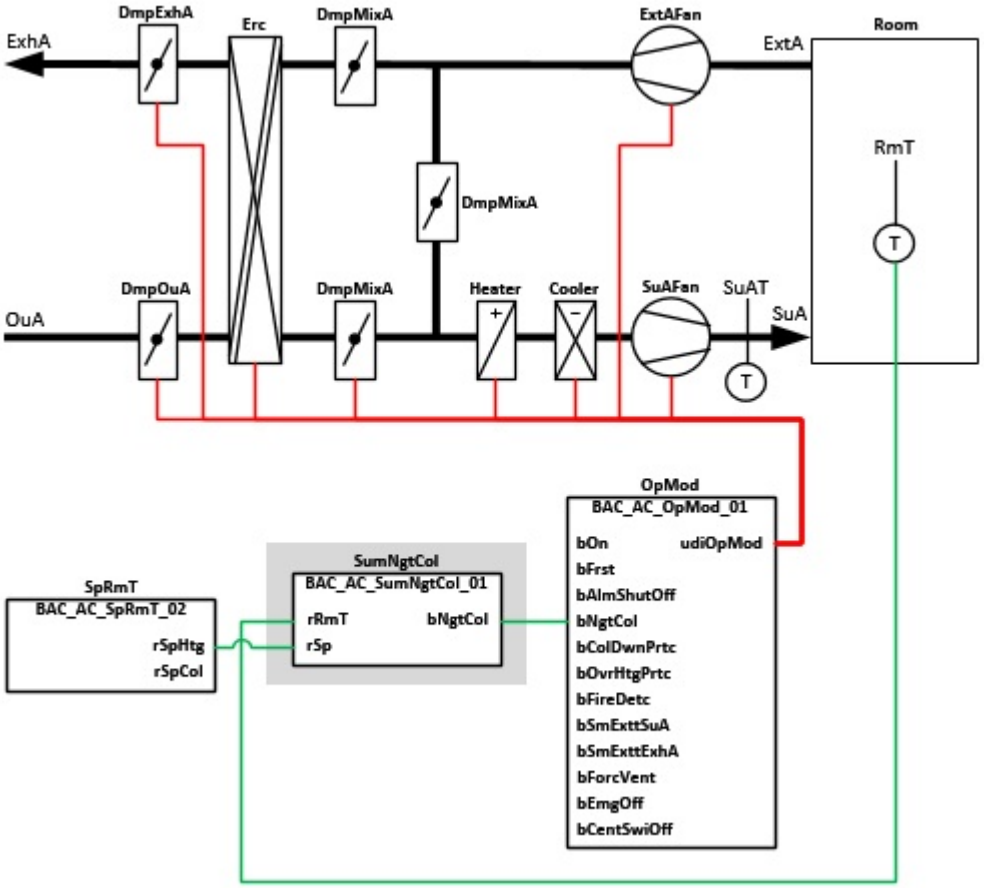
### Functional description

The template **BAC_AC_SumNgtCol_01** is used for night cooling with cool outside air of rooms that were warmed up during the previous day. 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.

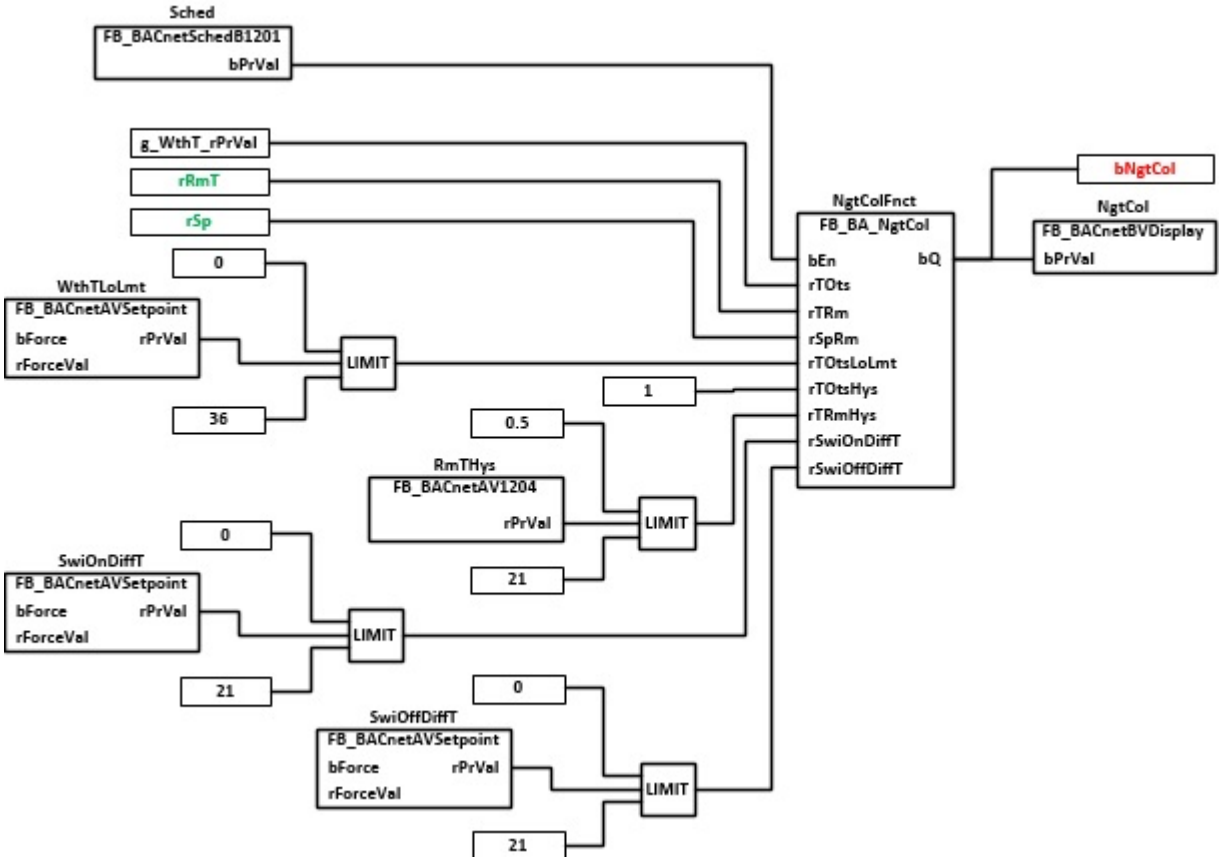
### Interface



### System diagram




### Block diagram



### VAR_INPUT

```
rRmT        : REAL;
rSp     : REAL;
```

**rRmT:** Measured value room temperature

**rSp:** Set value room temperature

### VAR_OUTPUT

```
bNgtCol     : BOOL;
```

**bNgtCol:** Summer night cooling active

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| RmTHys | FB_BACnetAVSetpoint [▶ 69] | Input of a hysteresis value around the room temperature set value. Avoids excessive switching on and off with fluctuating room temperature.<br>The switch-on point is exceeded if the room temperature *rRmT* is greater than **SpRm** + **RmTHys**.<br>The switch-off point is reached if the room temperature falls below **SpRm**. |
| Sched | FB_BACnetSchedB1201 [▶ 131] | The schedule object can be used to define periods in which night cooling is enabled. |
| WthTLoLmt | FB_BACnetAVSetpoint [▶ 69] | Input value for the lower outside temperature limit. The lower outside temperature limit prevents excessive cooling. |
| SwiOffDiffT | FB_BACnetAVSetpoint [▶ 69] | Input value for the difference between the room temperature and the outside temperature, from which summer night cooling is disabled. |
| SwiOnDiffT | FB_BACnetAVSetpoint [▶ 69] | Input value for the difference between the room temperature and the outside temperature, from which summer night cooling is enabled. |
| NgtColFnct | FB_BA_NgtCol [▶ 233] | The function block **NgtColFnct** is the core of the templates **BAC_AC_SumNgtCol_01** and includes the actual control process of the night cooling program |
| NgtCol | FB_BACnetBV1204 [▶ 94], FB_BACnetBVDisplay [▶ 95] | The BV object indicates that night cooling is active. |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.54 BAC_H_HtgCir_01

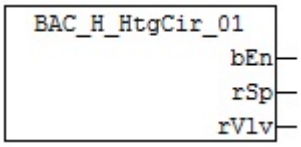
### Application

The call template **BAC_H_HtgCir_01** is a weather temperature controlled heating circuit. It is comprised of a single-stage pump, an analog valve and a heating characteristic curve with 4 interpolation points.

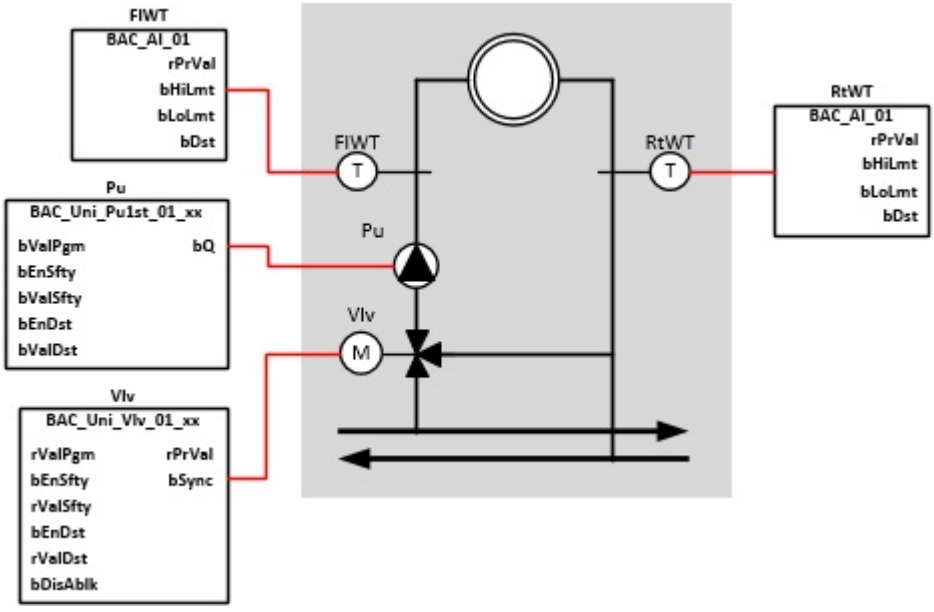
The main tasks of the template are:

- Control of the flow temperature
- Control of the pump
- Control of the control valve

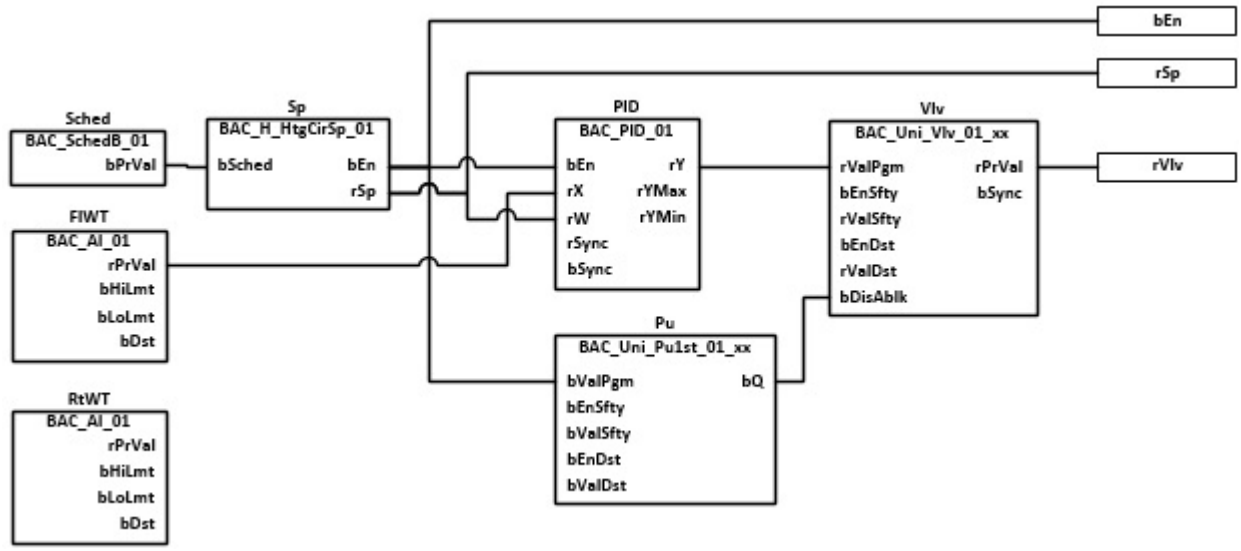
### Interface




### System diagram



### Block diagram




### VAR_OUTPUT

```
bEn     : REAL;
rSp     : REAL;
rVlv    : REAL;
```

**bEn:** Output of the enable signal for the heating circuit

**rSp:** Output of the setpoint for the heating circuit

**rVlv:** Output of the valve position

### Program description

| Instance | Type | Task |
|---|---|---|
| Sched | BAC_SchedB_01 [▶ 709] | Sub template timer program for switching between day and night mode |
| FlWT | BAC_AI_01 [▶ 675] | Sub template for displaying the flow temperature |
| RtWT | BAC_AI_01 [▶ 675] | Sub template for displaying the return temperature |
| Sp | BAC_H_HtgCirSp_01 [▶ 545] | Sub template weather temperature-controlled heating circuit control. It is comprised of a heating curve, a heating limit switch, and the operating modes Frost / Night / Day / Auto with the associated setpoint. |
| PID | BAC_PID_01 [▶ 587] | Sub template PID controller for controlling the flow temperature |
| Vlv | BAC_Uni_Vlv_01_11 [▶ 598] | Sub template for controlling an analog valve |
| Pu | BAC_Uni_Pu1st_01_61 [▶ 571] | Sub template for controlling a single-stage pump |

### Version history

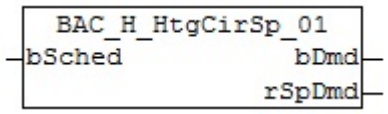
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

# 9.55 BAC_H_HtgCirSp_01

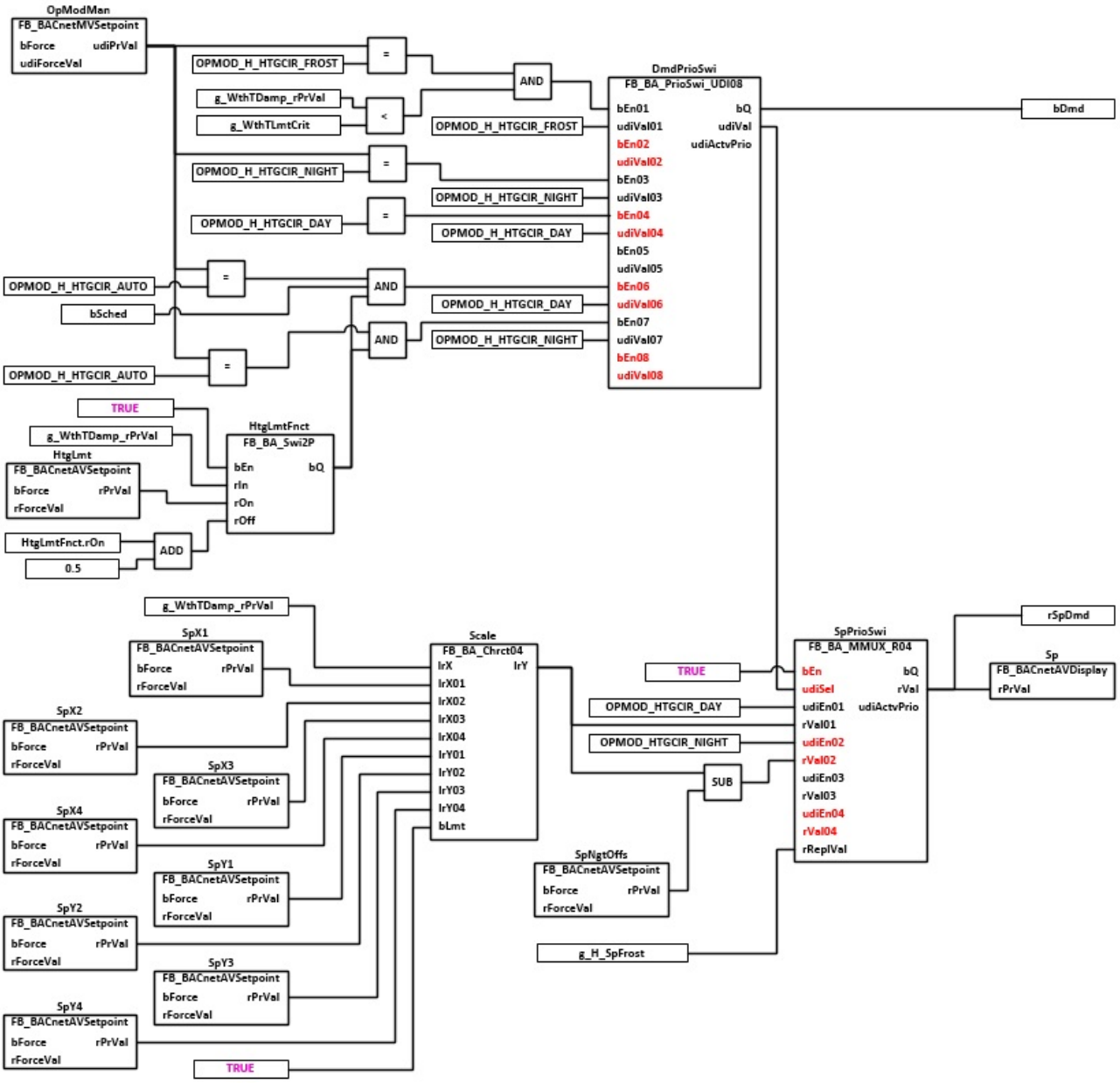
### Application

The template is an atmospheric temperature controlled heating circuit control. It is comprised of a heating curve, a heating limit switch and the operating modes Frost / Night / Day / Auto with the associated setpoint.

### Interface




### Block diagram




### VAR_INPUT

```
bSched  : BOOL;
```

**bSched:** Timer input for switching between day and night mode

### VAR_OUTPUT

```
bDmd       : BOOL;
rSpDmd     : REAL;
```

**bDmd:** Output of the heating operation request, depending on the operating mode **OpModMan**.

**rSpDmd:** Output of the setpoint for the heating circuit

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm, [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt, [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| OpMod | FB_BACnetMVSetpoint [▶ 130] | MV object for manual control of the heating circuit from the MCL or a local operating display. The following operating modes can be preset: OPMOD_H_HTGCIR_FROST [▶ 357] / OPMOD_H_HTGCIR_NIGHT [▶ 357] / OPMOD_H_HTGCIR_DAY [▶ 357] / OPMOD_H_HTGCIR_AUTO [▶ 357] |
| DmdPrioSwi | FB_BA_PrioSwi_UDI08 [▶ 213] | The priority switch prioritises the operating modes of the heating circuit. |
| HtgLmt | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the heating limit value. |
| HtgLmtFnct | FB_BA_Swi2P [▶ 146] | Depending on the damped outside temperature g_WthTDamp_rPrVal [▶ 357] and the heating limit value, the two-point switch enables the heating operation in the operating mode OPMOD_H_HTGCIR_AUTO [▶ 357]. |
| SpX1 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point X1 |
| SpX2 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point X2 |
| SpX3 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point X3 |
| SpX4 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point X4 |
| SpY1 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point Y1 |
| SpY2 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point Y2 |

| Instance | Type | Task |
|---|---|---|
| SpY3 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point Y3 |
| SpY4 | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for interpolation point Y4 |
| Scale | FB_BA_Chrct04 [▶ 172] | This function block calculates the set characteristic curve for the heating circuit, depending on the outside temperature. |
| SpPrioSwi | FB_BA_MMUX_R04 [▶ 205] | The multiplexer function block outputs the associated setpoint, depending on the present operating mode **OpModMan**. |
| SpNgtOffs | FB_BACnetAVSetpoint [▶ 69] | AV object for inputting the value for the night setback |
| Sp | FB_BACnetAVDisplay [▶ 69] | Display of the setpoint for the heating circuit. It is output at output **rSpDmd**. |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.56 BAC_DHW_01

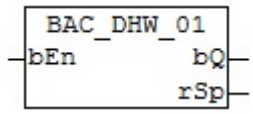
### Application

The call template **BAC_DHW_01** is used for controlling a hot water tank.

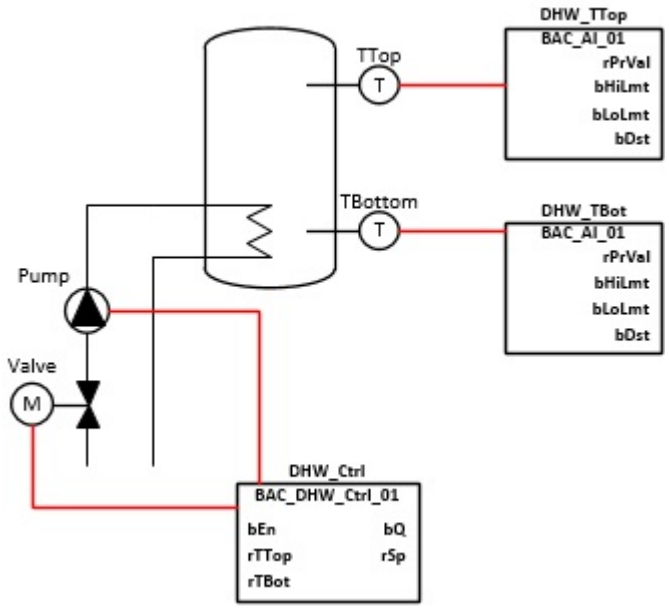
The main tasks of the template are:

- Controlling the tank temperature
- Controlling enabling of the charging process

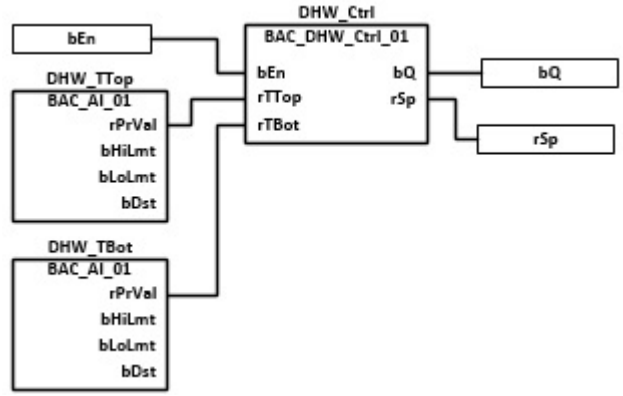
### Interface



### System diagram




### Block diagram




### VAR_INPUT

```
bEn     : BOOL;
```

**bEn:** Template enable input. If this input is FALSE, the value 0 is output at **rSp**, and **bQ** becomes FALSE.

### VAR_OUTPUT

```
bQ      : BOOL;
rSp     : REAL;
```

**bQ:** Hot water tank charging output.

**rSp:** Hot water temperature set value output.
If enable **En** of the template is FALSE, the value 0 is output at **rSp**.

### Program description

| Instance | Type | Task |
|---|---|---|
| DHW_TTop | BAC_AI_01 [▶ 675] | Subtemplate AI object tank temperature top |
| DHW_TBot | BAC_AI_01 [▶ 675] | Subtemplate AI object tank temperature bottom |
| DHW_Ctrl | BAC_DHW_Ctrl_01 [▶ 549] | Subtemplate tank temperature control |

### Version history

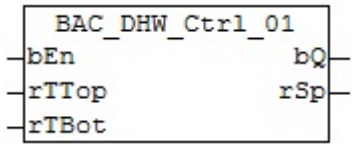
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.57 BAC_DHW_Ctrl_01

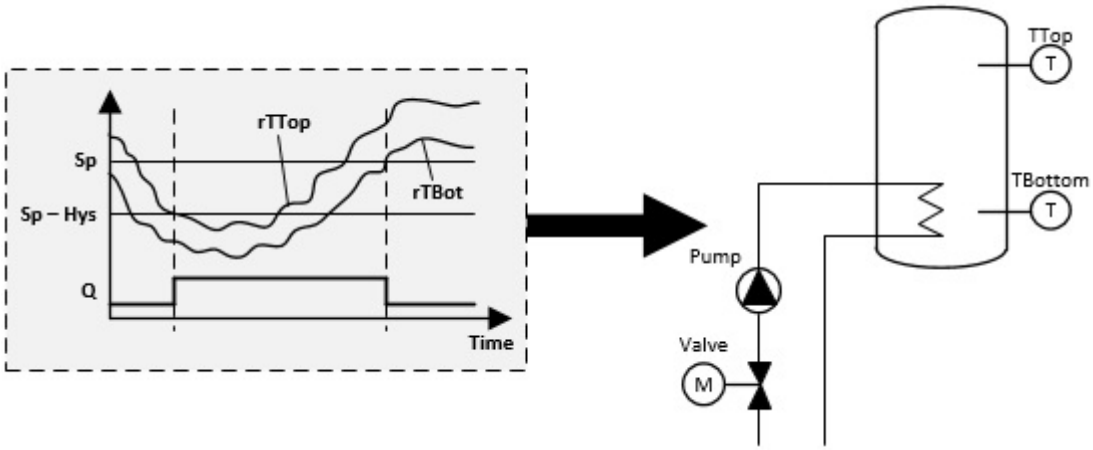
### Application

The template controls the charging of a hot water tank.
The template is enabled via the input variable **bEn**. In the template the temperature at the top of the tank **rTTop** is compared with the value of the difference **Sp - Hys**. If the temperature at the top of the tank falls below the value, charging of the hot water tank is enabled.
If the temperature at the bottom of the tank **rTBot** exceeds the set value **Sp**, charging of the hot water tank is disabled.

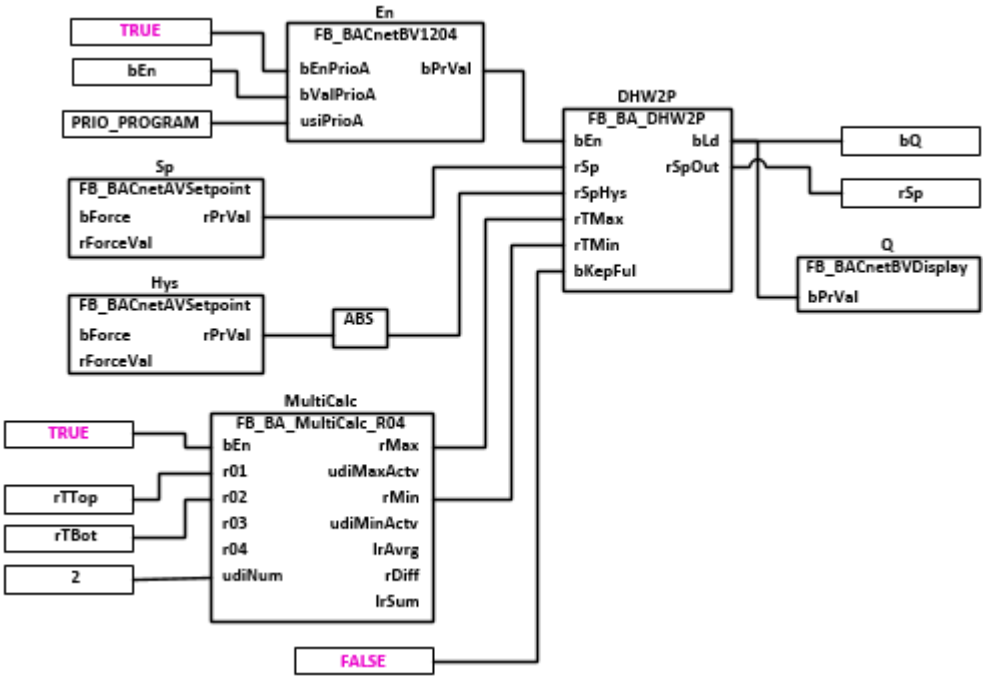
### Interface




### System diagram



### Block diagram




### VAR_INPUT

```
bEn         : BOOL;
rTTop       : REAL;
rTBot       : REAL;
```

**bEn:** Template enable input. If this input is FALSE, the value 0 is output at **rSp**, and **bQ** becomes FALSE.

**rTTop:** Input for temperature at the top of the tank

**rTBot:** Input for temperature at the bottom of the tank

### VAR_OUTPUT

```
bQ      : BOOL;
rSp     : REAL;
```

**bQ:** Hot water tank charging output.

**rSp:** Hot water temperature set value output.
If enable **En** of the template is FALSE, the value 0 is output at **rSp**.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| En | FB_BACnetBV1204 [▶ 94] | BV object for displaying the hot water tank charging template enable. |
| Sp | FB_BACnetAVSetpoint [▶ 69] | AV object for entering the set value for the hot water temperature. If the temperature at the bottom of the tank **rTBot** exceeds this value, charging of the hot water tank is disabled. The hot water tank has reached the required temperature. |
| Hys | FB_BACnetAVSetpoint [▶ 69] | AV object for entering the set value hysteresis. If the temperature at the top of the tank **rTTop** falls below the value of the difference **Sp - Hys**, charging of the hot water tank is enabled. This is indicated with BACnet object **Q**. |
| MultiCalc | FB_BA_MultiCalc_R04 [▶ 208] | The function block determines the maximum and minimum of the inputs r01-r04 and outputs the values at rMax and rMin. |
| DHW2P | FB_BA_DHW2P [▶ 216] | The function block is the main control feature for charging the hot water tank.<br>Charging of the hot water tank is controlled by means of an on-off controller. Tank heating is activated at input *bEn*. If tank heating is active the output *bLd* is TRUE. The variables *rSp* is used to transfer the set value 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.<br>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. |
| Q | FB_BACnetBVDisplay [▶ 95] | BV object for indicating whether charging of the hot water tank is enabled. |

### Version history

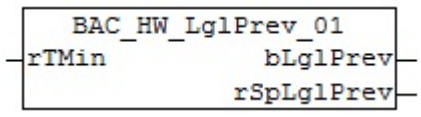
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.58 BAC_HW_LglPrev_01

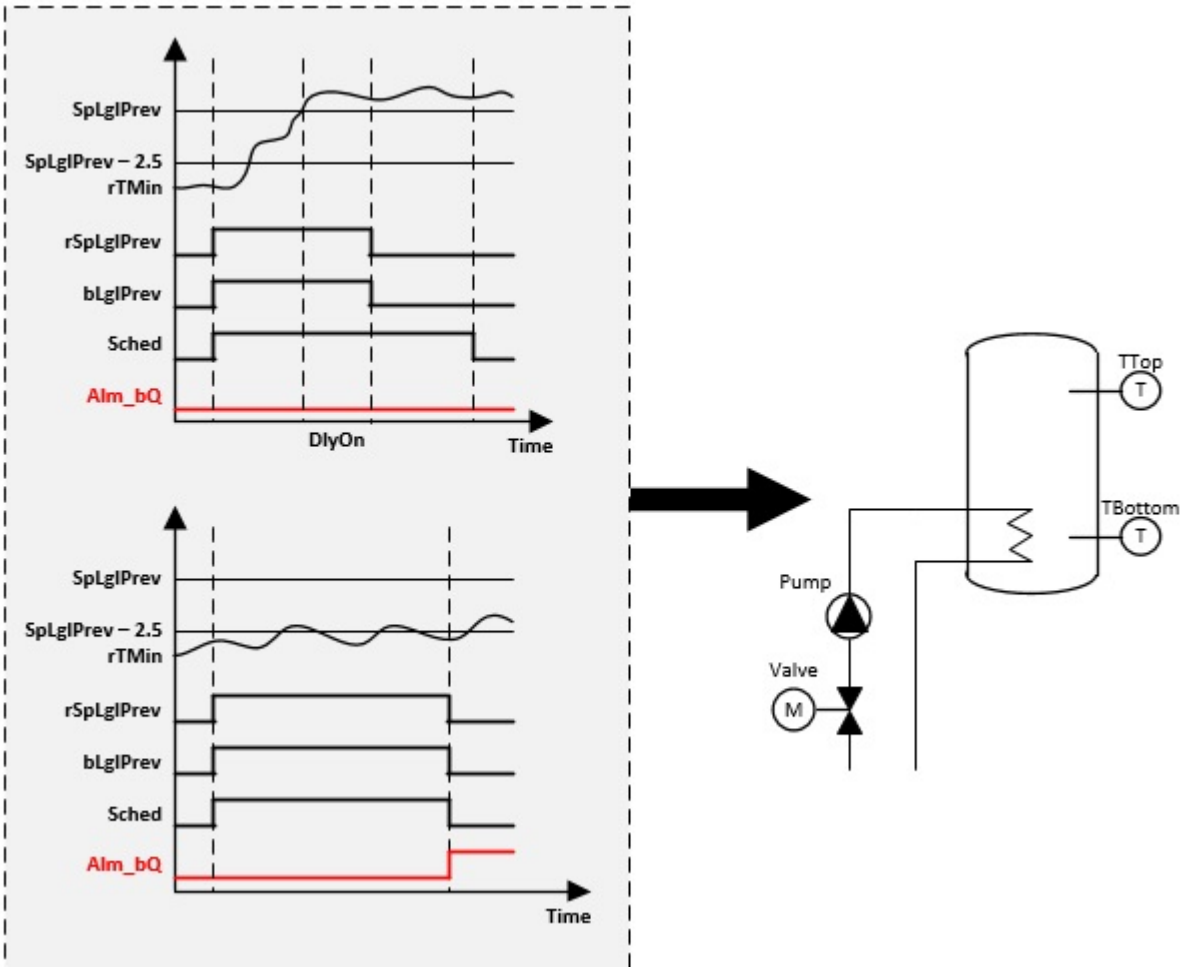
### Application

The template provides a protective function for preventing the formation of legionella in the hot water.
The protective function is enabled for a specific timeframe through the switching schedule **Sched**. In the template the minimum tank temperature **rTMin** is compared with the set value **SpLglPrev**, and a charging command **bLglPrev** and set value **rSpLglPrev** are issued.
The aim is for the tank temperature to be maintained for a certain time **TiLglPrev** within the timeframe **Sched**. If this condition was not met, an alarm is generated **AlmLglPrev**.

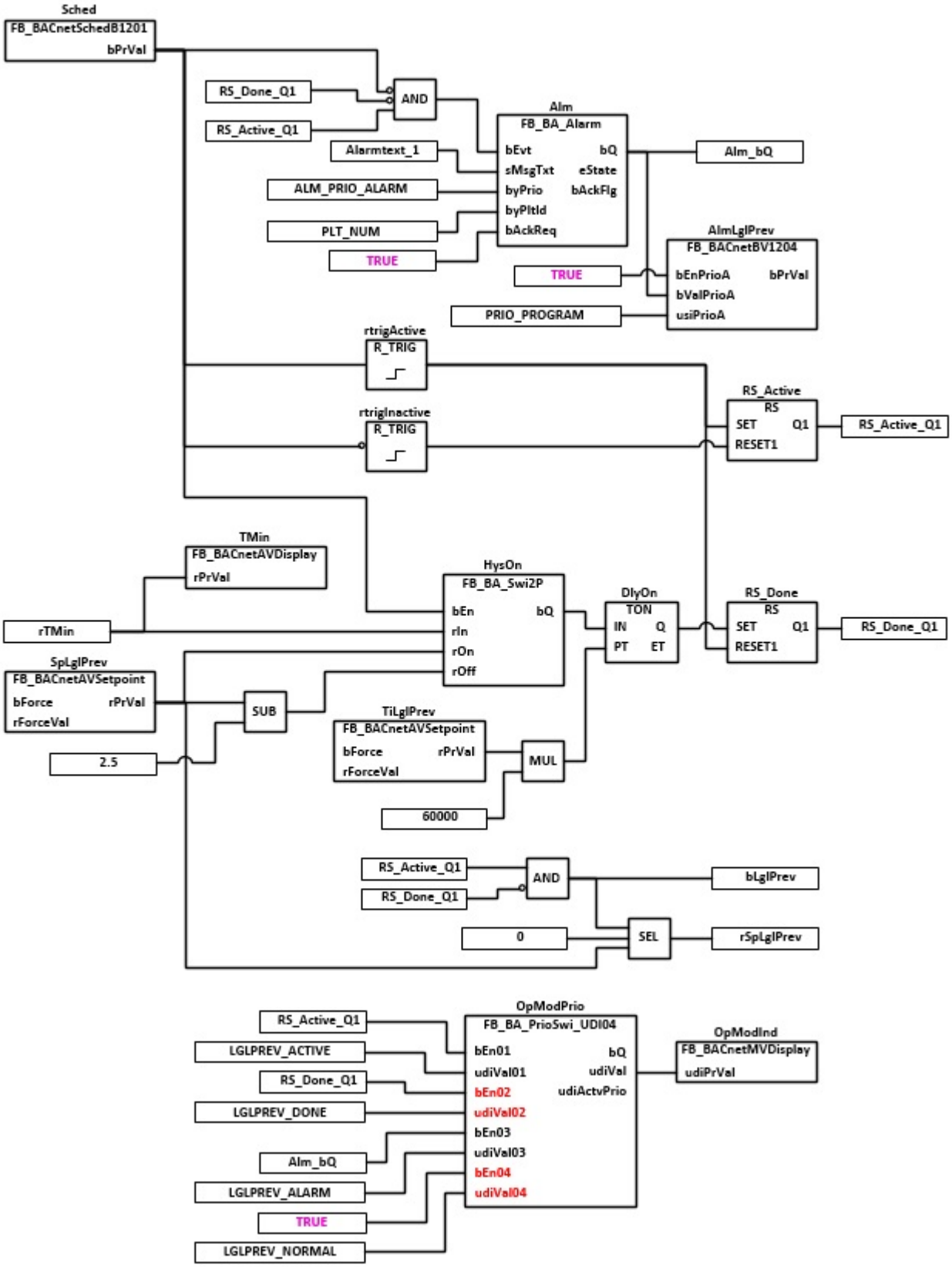
### Interface



**System diagram**



### Block diagram




### VAR_INPUT

```
rTMin    : REAL;
```

**rTMin:** Input minimum tank temperature

### VAR_OUTPUT

```
bLglPrev     : BOOL;
rSpLglPrev   : REAL;
```

**bLglPrev:** Output request legionella protection

**rSpLglPrev:** Output set value legionella protection

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm, [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt, [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| Sched | FB_BACnetSchedB1201 [▶ 131] | Input switching schedule enable legionella prevention |
| TMin | FB_BACnetAVDisplay [▶ 69] | Display of minimum temperature **rTMin** of hot water tank. |
| SpLglPrev | FB_BACnetAVSetpoint [▶ 69] | AV object for entering the setpoint for legionella prevention. |
| HysOn | FB_BA_Swi2P [▶ 146] | The two-point switch indicates that legionella prevention is active, depending on the minimum hot water tank temperature **rTMin** and the setpoint for legionella prevention **SpLglPrev**. |
| TiLglPrev | FB_BACnetAVSetpoint [▶ 69]<br>MUL | AV object for entering the duration of legionella prevention.<br>Multiplication of the input duration of legionella prevention in minutes |
| DlyOn | TON | Timer delay legionella prevention successful. |
| RS_Active | RS | Signal latch legionella prevention active |
| RS_Done | RS | Signal latch timeframe legionella prevention completed. |
| Alm | FB_BA_Alarm [▶ 179] | Logging and further processing of the legionella setpoint alarm not reached |
| AlmLglPrev | FB_BACnetBV1204 [▶ 94] | Reports legionella setpoint alarm not reached to MCL. |
| OpModPrio | FB_BA_PrioSwi_UDI04 [▶ 213] | The priority switch prioritizes the legionella prevention operating modes. |
| OpModInd | FB_BACnetMVDisplay [▶ 129] | The BACnet MV object indicates the currently valid operation mode of the legionella protection function.<br>LGLPREV_NORMAL [▶ 357] Legionella prevention normal, alarm was reset |

| Instance | Type | Task |
|---|---|---|
| | | LGLPREV_ACTIVE [▶ 357] Legionella prevention active<br>LGLPREV_DONE [▶ 357] Legionella prevention done<br>LGLPREV_ALARM [▶ 357] Legionella prevention alarm |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.59 BAC_Uni_FC_01_xx

### Functional description

The template **BAC_Uni_FC_01_xx** is used for controlling a frequency converter with binary and analog inputs and outputs.

The two output variables rPrVal / bSync are only active, if the mechanical priority operation FdbOutAO exists in the template that is used. If this is not the case, the two variables return the value zero.

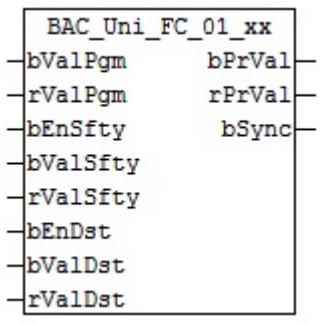
### Versions

The template **BAC_Uni_FC_01_xx** is available in different versions.
The template versions are identified by means of a key. The identification key is derived from the table below.

| Options | Repair switch (Rep) | Mechanical priority operation enable FC feedback hand switch (Enbl A-0-H) | Mechanical priority operation enable FC feedback relay (Rm-output) | Mechanical priority operation set value FC feedback hand switch (Rotary A-H) | Mechanical priority operation control value FC feedback status potentiometer (Rm-ContrVal) | Operating feedback FC (Oper-FC) | Fault message FC (Fault-FC) | Fault message motor protection switch (Therm) |
|---|---|---|---|---|---|---|---|---|
| **Instance** | **MntnSwi** | **LocSwiBO** | **FdbOutBO** | **LocSwiAO** | **FdbOutAO** | **FdbFC** | **DstFC** | **ThOvrld** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** | **AI** | **BI** | **BI** | **BI** |
| | **128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| BAC_Uni_FC_01_002 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| BAC_Uni_FC_01_006 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| BAC_Uni_FC_01_007 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| BAC_Uni_FC_01_086 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

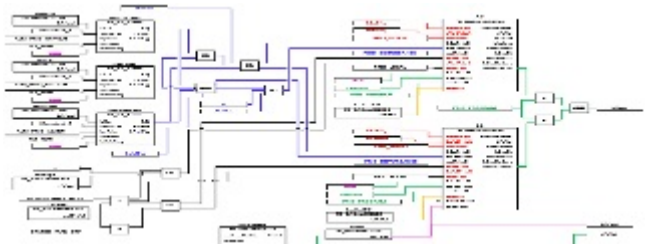
| Options | Repair switch (Rep) | Mechanical priority operation enable FC feedback hand switch (Enbl A-0-H) | Mechanical priority operation enable FC feedback relay (Rm-output) | Mechanical priority operation set value FC feedback hand switch (Rotary A-H) | Mechanical priority operation control value FC feedback status potentiometer (Rm-ContrVal) | Operating feedback FC (Oper-FC) | Fault message FC (Fault-FC) | Fault message motor protection switch (Therm) |
|---|---|---|---|---|---|---|---|---|
| **Instance** | **MntnSwi** | **Loc-SwiBO** | **Fd-bOutBO** | **Loc-SwiAO** | **Fd-bOutAO** | **FdbFC** | **DstFC** | **ThOvrld** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** | **AI** | **BI** | **BI** | **BI** |
| | **128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| BAC_Uni_FC_01_087 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| BAC_Uni_FC_01_094 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| BAC_Uni_FC_01_130 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| BAC_Uni_FC_01_134 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| BAC_Uni_FC_01_135 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| BAC_Uni_FC_01_214 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| BAC_Uni_FC_01_215 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| BAC_Uni_FC_01_222 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| BAC_Uni_FC_01_223 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

### Interface



### Block diagram version BAC_Uni_FC_01_223

https://infosys.beckhoff.com/content/1033/tcba/Resources/12269787275/.jpg



### VAR_INPUT

```
bValPgm    : BOOL;
rValPgm    : REAL;
bEnSfty    : BOOL;
bValSfty   : BOOL;
rValSfty   : REAL;
bEnDst     : BOOL;
bValDst    : BOOL;
rValDst    : REAL;
```

**bValPgm:** binary value program priority

**rValPgm:** analog value program priority

**bEnSfty:** safety priority enable

**bValSfty:** binary value safety priority

**rValSfty:** analog value safety priority

**bEnDst:** disturbance priority enable. This input can be used to connect process feedback, for example.

**bValDst:** binary value disturbance priority. This input can be used to connect process feedback, for example.

**rValDst:** analog value disturbance priority

### VAR_OUTPUT

```
bPrVal   : BOOL;
rPrVal   : REAL;
bSync    : BOOL;
```

**bPrVal:** Operating feedback FC

**rPrVal :** Current speed of the FC.

**bSync:** Output of a pulse to synchronise the controller associated with the FC during reset from manual to automatic operation to the current FC speed.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

**Program description**

| Instance | Type | op-tional | Task |
|---|---|---|---|
| ThOvrld | FB_BACnetBI1203 [▶ 72] | X | BI object thermal motor protection (PTC thermistor tripping unit, motor protection switch etc.) |
| DstFC | FB_BACnetBI1203 [▶ 72] | X | BI object frequency converter fault |
| LocSwiBO | FB_BACnetBI1203 [▶ 72] | X | BI object mechanical priority operation feedback hand switch binary |
| LocSwiAO | FB_BACnetBI1203 [▶ 72] | X | BI object mechanical priority operation feedback hand switch analog |
| FdbFC | FB_BACnetBI1203 [▶ 72] | X | BI object operating feedback FU |
| MntnSwi | FB_BACnetBI1203 [▶ 72] | X | BI object maintenance switch |
| FdbOutAO | FB_BACnetAI1203 [▶ 49] | X | Return value mechanical priority operation control value FU |
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the frequency converter via the MCL or a local operator display |
| ManSpd | FB_BACnetAVSetpoint [▶ 69] | | AV object for entering the motor speed with manual override |
| AlmThOvrld | FB_BA_Alarm [▶ 179] | x | Logging and further processing of an error for thermal protection of the motor (PTC thermistor tripping unit, motor protection switch etc.) |
| AlmDstFC | FB_BA_Alarm [▶ 179] | x | Logging and further processing of a frequency converter fault |
| AlmMntnSwi | FB_BA_Alarm [▶ 179] | x | Logging and processing of the repair switch triggered event. |
| AO | FB_BACnetAO1203 [▶ 53] | | AO object for specifying the set speed value for the frequency converter |

| Priority: | Enable | Value | Comment |
|---|---|---|---|
| PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** | |
| PRIO_DISTURBANCE (3) | Input **bEnDst** OR **ThOvrld** OR **DstFC** OR **MntnSwi** | no fault from **bEnDst**, **ThOvrld**, **DstFC** or **MntnSwi** -> value of input **rValDst**<br><br>fault from **bEnDst**, **ThOvrld**, **DstFC** or **MntnSwi** = 0 | |
| PRIO_LOCAL (8) | **OpMod_udiPrVal** = **OPMOD_MAN_OFF** OR **OPMOD_MAN_STP01** | **Selector** 0 OR **ManSpd_rPrVal** | In manual mode, value of AV object **ManSpd** |
| PRIO_PROGRAM (15) | TRUE | **rValPgm** | Value of input **rValPgm** (e.g. controller |

| Instance | Type | optional | Task | | | |
|---|---|---|---|---|---|---|
| | | | | | | speed specification) |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for specifying the frequency converter enable | | | |
| | | | **Priority:** | **Enable** | **Value** | **Comment** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** | |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** OR **ThOvrld** OR **DstFC** OR **MntnSwi** | Value of the operator AND with the inputs **bValDst**, **ThOvrld_bPrVal**, **MntnSwi_bPrVal**, **DstFC_bPrVal** | |
| | | | PRIO_LOCAL (8) | **OpMod_udiPrVal** = **OPMOD_MAN_OFF** OR **OPMOD_MAN_STP01** | TRUE, if **OpMod_udiPrVal** = **OPMOD_MAN_STP01** | In manual mode value of AV object ManSpd |
| | | | PRIO_PROGRAM (15) | TRUE | **bValPgm** | Value of input **bValPgm** |
| | EQ, EQ, AND | | Value of the network is TRUE, if the active priority is **PRIO_PROGRAM** (15). Can be used for synchronizing the controller on return to automatic mode | | | |
| TLogAO | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the AO object for specifying the set speed value for the frequency converter | | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_Thermal_Overlaod | BOOL | X | Input | Digital input - thermal motor overload - message - triggered |
| BI_FC_Disturbance | BOOL | X | Input | Digital input - FC fault - message - triggered |
| BI_FC_Feedback | BOOL | X | Input | Digital input - FC - message - operation |
| BI_Maintenance_Switch | BOOL | X | Input | Digital input – maintenance switch - message - triggered |
| BI_FC_Enable_LocalSwitch | BOOL | X | Input | Digital input - switch manual enable FC - message - manual/auto |
| BI_Speed_LocalSwitch | BOOL | X | Input | Digital input - switch speed - message - manual/auto |
| BI_Feedback_Binary_Output | BOOL | X | Input | Digital input - feedback control command- message - on/off |
| AI_Feeback_Speed_Poti | INT | X | Input | Analog input - manual potentiometer – feedback - control value |

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BO_FC_Enable | BOOL | | Output | Digital output - FC switching command - enable on/off |
| AO_FC_Speed_Value | INT | | Output | Analog output - FC control value - speed |

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.60 BAC_Uni_Dmp_01_xx

### Functional description

The template **BAC_Uni_Dmp_01_xx** is used for controlling an analog damper drive. It basically consists of an AO object including a trend object, an MV object for manual control, and the associated AV object for entering the position. The template is complemented through optional BACnet objects, see table with version overview.

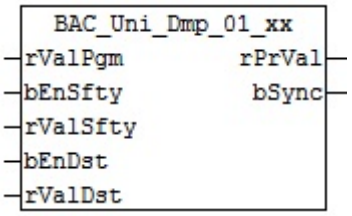
The two output variables rPrVal / bSync are only used for synchronization of controllers if the template used contains feedback of the damper position Fdb.

### Versions

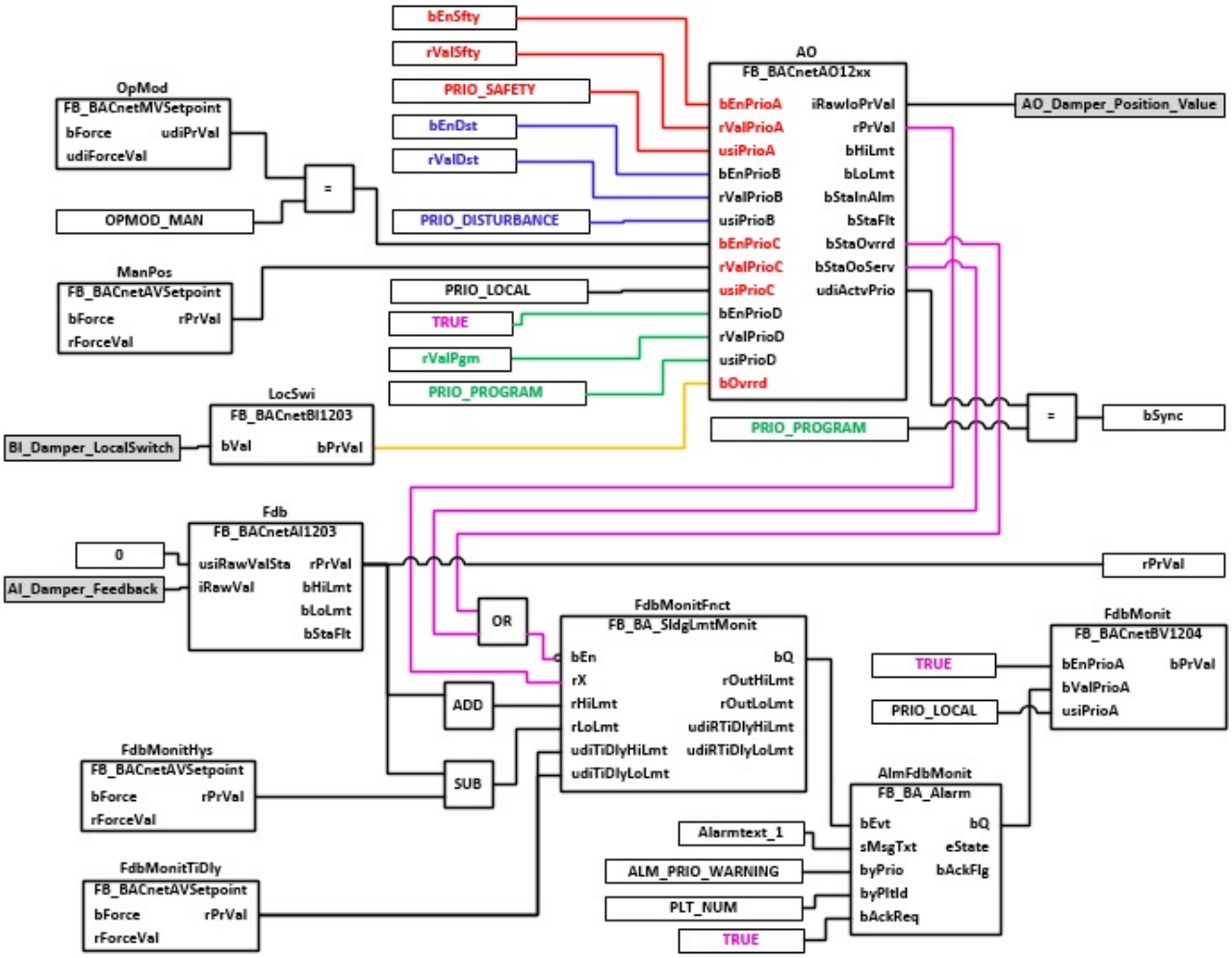
The template **BAC_Uni_Dmp_01_xx** is available in different versions.
The damper versions are identified by means of a key. The identification key is derived from the table below.

| Options | Feedback Damper position with monitoring (Rm) | Mechanical priority operation position feedback Potentiometer (Rm-output) | mechanical priority operation position feedback hand switch (A-0-H) |
|---|---|---|---|
| **Instance** | **Fdb** | **FdbOut** | **LocSwi** |
| **Data point type** | **AI** | **AI** | **BI** |
| | **4** | **2** | **1** |
| BAC_Uni_Dmp_01_00 | 0 | 0 | 0 |
| BAC_Uni_Dmp_01_01 | 0 | 0 | 1 |
| BAC_Uni_Dmp_01_03 | 0 | 1 | 1 |
| BAC_Uni_Dmp_01_04 | 1 | 0 | 0 |
| BAC_Uni_Dmp_01_05 | 1 | 0 | 1 |
| BAC_Uni_Dmp_01_07 | 1 | 1 | 1 |

### Interface



### Block diagram version BAC_Uni_Dmp_01_05




### VAR_INPUT

```
rValPgm  : REAL;
bEnSfty  : BOOL;
bValSfty : BOOL;
bEnDst   : BOOL;
bValDst  : BOOL;
```

**rValPgm:** Analog value program priority

**bEnSfty:** Safety priority enable

**rValSfty:** Analog value safety priority

**bEnDst:** Enable fault priority

**rValDst:** Analog value fault priority

### VAR_OUTPUT

```
rPrVal   : REAL;
bSync    : BOOL;
```

**rPrVal :** Current position of the damper drive.
The output variable **rPrVal** is only active if the template used contains feedback of the damper position **Fdb** .

**bSync:** Output of a pulse to synchronise the controller associated with the damper during reset from manual to automatic operation to the current damper position.
The output variable **bSync** is only active if the template used contains feedback of the damper position **Fdb** .

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the damper drive via the MCL or a local operator display |
| ManPos | FB_BACnetAVSetpoint [▶ 69] | | AV object for entering the damper position with manual override |
| FdbMonitHys | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the hysteresis for the function monitoring of the control valve via the position feedback |
| FdbOut | FB_BACnetAI1203 [▶ 49] | X | AI object for logging the mechanical priority operation position feedback potentiometer |
| Fdb | FB_BACnetAI1203 [▶ 49] | X | AI object for logging the position feedback from the damper position |
| FdbMonitTiDly | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the response delay of the function monitoring via position feedback. |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object signals mechanical priority operation position feedback hand switch |
| AO | FB_BACnetAO1203 [▶ 53] | | AO object for controlling the control valve. |

| **Priority:** | **Enable** | **Value** | **Comment** |
|---|---|---|---|
| PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** | |
| PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** | |
| PRIO_LOCAL (8) | **OpMod_udiPrVal** = **OPMOD_MAN** | **ManPos_rPrVal** | In manual mode, value of AV object **ManPos** |
| PRIO_PROGRAM (15) | TRUE | **AblkFnct_bQ** or **rValPgm** | Value of input **rValPgm** or 100% of blocking protection |

| | | | |
|---|---|---|---|
| | EQ | | TRUE if the active priority is **PRIO_PROGRAM** (15). Can be used for synchronizing the controller on return to automatic mode |

| Instance | Type | op-tional | Task |
|---|---|---|---|
| FdbMonitFnct | FB_BA_SldgLmtMonit [▶ 143] | X | Function monitoring of the damper drive through comparison of the control output and the position feedback. |
| AlmFdbMonit | FB_BA_Alarm [▶ 179] | X | Logging and further processing of an error from the position feedback monitoring |
| FdbMonit | FB_BACnetBV1204 [▶ 94] | X | Reports position feedback error to the MCL |
| TLogAO | FB_BACnetTLog1201 [▶ 135] | | Logs the present value of the AO object |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| AI_Damper_Feedback | INT | X | Input | Damper - measured value - return value position |
| AI_Feeback_Position_Poti | INT | X | Input | Damper potentiometer H-A - measured value |
| BI_Damper_LocalSwitch | BOOL | X | Input | Damper switch H-A - message - operation mode |
| AO_Damper_Position_Value | INT | | Output | Analog output - damper - control value position |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.61 BAC_Uni_Dmp2P_01_xx

### Functional description

The template **BAC_Uni_Dmp2P_01_xx** is used for control and monitoring of a two-point damper, e.g. a damper with a spring return actuator. It basically consists of a BO object for damper control and an MV object for manual override. The template is complemented by optional BACnet objects, see table with version overview.
The variables that are linked to the process image of the input and output level in the PLC can be found at **IO linking**.

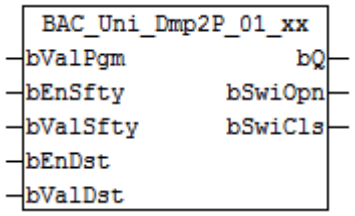
The two output variables bSwiOpn / bSwiCls only output the actual state of the damper, if the template used includes feedback from the limit switches SwiOpn/SwiCls. If no end position monitoring is available, it is emulated internally. If the damper is controlled, output bSwiOpn becomes TRUE. If the damper is not controlled, the output bSwiCls becomes TRUE. This may mean that delay time for opening the damper has to be specified in the start program for a ventilation system.

### Versions

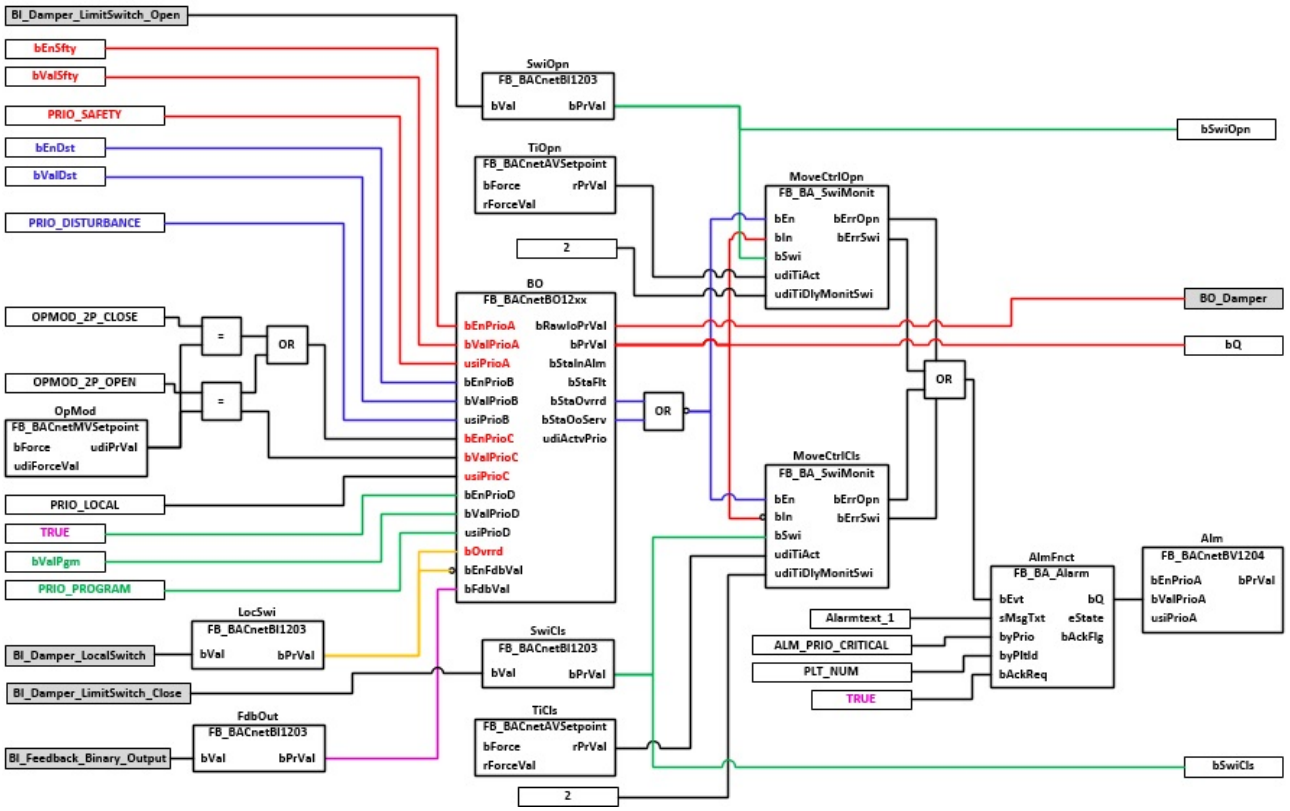
The template **BAC_Uni_Dmp2P_01_xx** is available in different versions.
The damper versions are identified by means of a key. The identification key is derived from the table below.

| Options | mechanical priority operation Feedback hand switch (A-0-H) | mechanical priority operation Feedback Relay output (Rm-output) | Final position Open (switch open) | Final position Close (switch close) |
|---|---|---|---|---|
| **Instance** | **LocSwi** | **FdbOut** | **SwiOpn** | **SwiCls** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** |
| | **8** | **4** | **2** | **1** |
| BAC_Uni_Dmp2P_01_00 | 0 | 0 | 0 | 0 |
| BAC_Uni_Dmp2P_01_02 | 0 | 0 | 1 | 0 |
| BAC_Uni_Dmp2P_01_03 | 0 | 0 | 1 | 1 |
| BAC_Uni_Dmp2P_01_08 | 1 | 0 | 0 | 0 |
| BAC_Uni_Dmp2P_01_10 | 1 | 0 | 1 | 0 |
| BAC_Uni_Dmp2P_01_11 | 1 | 0 | 1 | 1 |
| BAC_Uni_Dmp2P_01_12 | 1 | 1 | 0 | 0 |
| BAC_Uni_Dmp2P_01_14 | 1 | 1 | 1 | 0 |
| BAC_Uni_Dmp2P_01_15 | 1 | 1 | 1 | 1 |

**Interface**



### Block diagram version BAC_Uni_Dmp2P_01_15




### VAR_INPUT

```
bValPgm     : BOOL;
bEnSfty     : BOOL;
bValSfty    : BOOL;
bEnDst      : BOOL;
bValDst     : BOOL;
```

**bValPgm:** Binary value program priority

**bEnfty:** Safety priority enable

**bValSfty:** Binary value safety priority

**bEnDst:** Enable fault priority

**bValDst:** Binary value fault priority

### VAR_OUTPUT

```
bQ          : BOOL;
bSwiOpn     : BOOL;
bSwiCls     : BOOL;
```

**bQ:** Damper control output status

**bSwiOpn:** End position 'open' of the damper has been reached. If no end position monitoring is available, it is emulated internally. If the damper is controlled, output **bSwiOpn** becomes TRUE. If the damper is not controlled, the output **bSwiCls** becomes TRUE.

**bSwiCls:** End position 'closed' of the damper has been reached. If no end position monitoring is available, it is emulated internally. If the damper is controlled, output **bSwiOpn** becomes TRUE. If the damper is not controlled, the output **bSwiCls** becomes TRUE.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| SwiOpn | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'open' | | |
| SwiCls | FB_BACnetBI1203 [▶ 72] | X | BI object for connecting the limit switch 'closed' | | |
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the damper via the MCL or a local operator display | | |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object for feedback from mechanical priority operation. (manual/emergency operating level) | | |
| FdbOut | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation position feedback relay | | |
| TiOpn | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the opening time value | | |
| TiCls | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the closing time value | | |
| MoveCtrlOpn | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'open' end position of the damper | | |
| MoveCtrlCls | FB_BA_SwiMonit [▶ 153] | X | Function block, which monitors the 'closed' end position of the damper | | |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for controlling the damper | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **bValDst** |
| | | | PRIO_LOCAL (8) | The OR module pools events which enable writing to the priority manual override (local) of the downstream BO object.<br>Events:<br>1. The MV object has the value **OPMOD_2P_CLOSE** (close damper) | TRUE, if **OpMod_udiPrVal** = **OPMOD_2P_OPEN** |

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| | | | | 2. The MV object has the value **OPMOD_2P_OPEN** (open damper) | |
| | | | PRIO_PROGRAM (15) | TRUE | Input **bValPgm** |
| AlmFnct | FB_BA_Alarm [▶ 179] | x | The AlmFnct function block records the event of the limit position switch monitoring.<br>Actions that are to take place after the limit switch fault is received can be parameterized in the template on the AlmFnct function block. | | |
| Alm | FB_BACnetBV1204 [▶ 94] | x | BV object for displaying the damper fault in the MCL | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| BI_Damper_LimitSwitch_Open | BOOL | X | Input | Damper limit switch - message - open |
| BI_Damper_LimitSwitch_Close | BOOL | X | Input | Damper limit switch - message - closed |
| BI_Damper_LocalSwitch | BOOL | X | Input | Damper switch H-0-A - message - operation mode |
| BI_Feedback_Binary_Output | BOOL | X | Input | Damper switch H-0-A - message - operation |
| BO_Damper | BOOL | | Output | Damper - switching command - open |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.62 BAC_Uni_Mot1st_01_xx

### Functional description

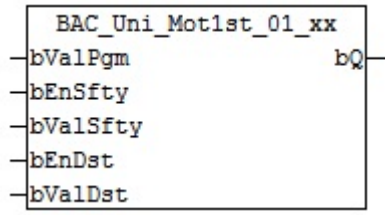
The template **BAC_Uni_Mot1st_01_xx** is used for controlling a single-stage motor, e.g. a fan with binary inputs and outputs. It basically consists of a BO object for the motor control and an MV object for manual override. The template is complemented through optional BACnet objects, see table with version overview.

### Versions

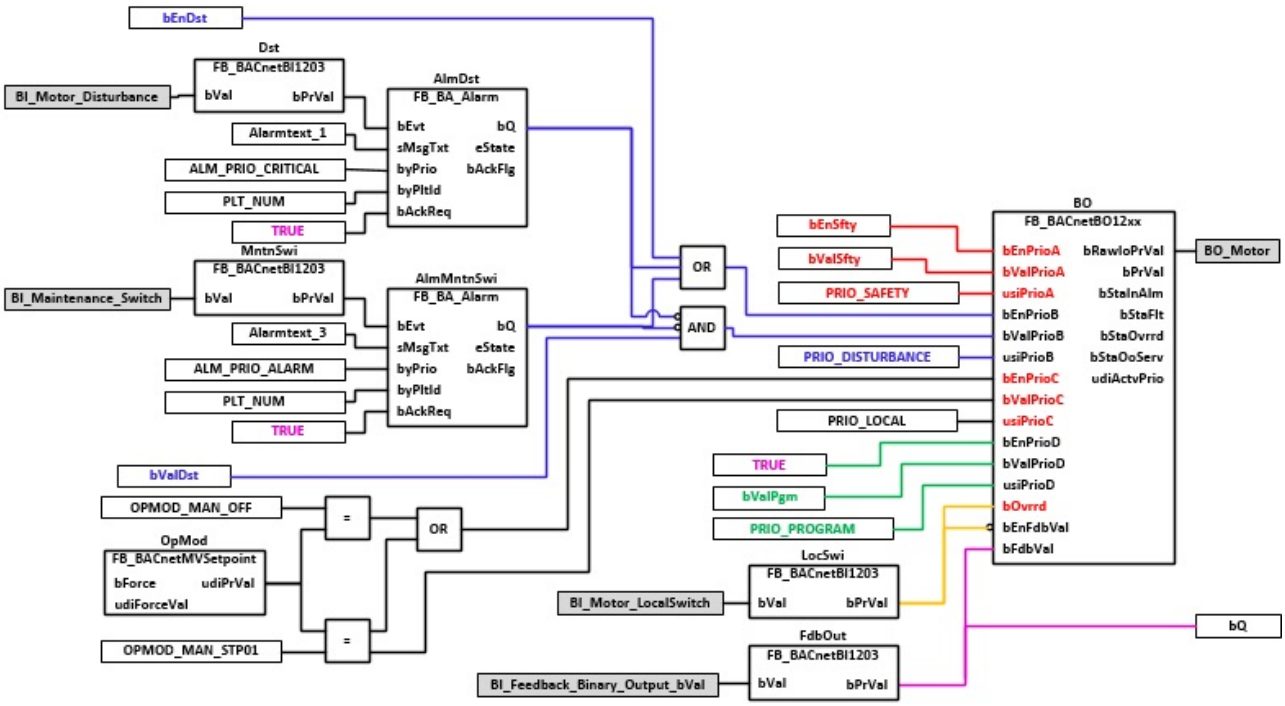
The template **BAC_Uni_Mot1st_01_xx** is available in different versions.
The template versions are identified by means of a key. The identification key is derived from the table below.

| Options | Maintenance switch (Rep) | mechanical priority operation Feedback hand switch (A-0-H) | mechanical priority operation Feedback Relay output (Rm-output) | Operating feedback (Oper) | Fault (Fault) |
|---|---|---|---|---|---|
| **Instance** | **MntnSwi** | **LocSwi** | **FdbOut** | **Fdb** | **Dst** |
| **Data point type** | **BI** | **BI** | **BI** | **BI** | **BI** |
| | **16** | **8** | **4** | **2** | **1** |
| BAC_Uni_Mot1st_01_01 | 0 | 0 | 0 | 0 | 1 |
| BAC_Uni_Mot1st_01_03 | 0 | 0 | 0 | 1 | 1 |
| BAC_Uni_Mot1st_01_11 | 0 | 1 | 0 | 1 | 1 |
| BAC_Uni_Mot1st_01_13 | 0 | 1 | 1 | 0 | 1 |
| BAC_Uni_Mot1st_01_17 | 1 | 0 | 0 | 0 | 1 |
| BAC_Uni_Mot1st_01_19 | 1 | 0 | 0 | 1 | 1 |
| BAC_Uni_Mot1st_01_27 | 1 | 1 | 0 | 1 | 1 |
| BAC_Uni_Mot1st_01_29 | 1 | 1 | 1 | 0 | 1 |

**Interface**



### Block diagram version BAC_Uni_Mot1st_01_29




### VAR_INPUT

```
bValPgm     : BOOL;
bEnSfty     : BOOL;
bValSfty    : BOOL;
bEnDst      : BOOL;
bValDst     : BOOL;
```

**bValPgm:** binary value program priority

**bEnSfty:** safety priority enable

**bValSfty:** binary value safety priority

**bEnDst:** disturbance priority enable. This input can be used to connect process feedback, for example.

**bValDst:** binary value disturbance priority. This input can be used to connect process feedback, for example.

### VAR_OUTPUT

```
bQ      : BOOL;
```

**bQ:** Operating feedback

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task | | | |
|---|---|---|---|---|---|---|
| Dst | FB_BACnetBI1203 [▶ 72] | x | BI object pump fault | | | |
| Fdb | FB_BACnetBI1203 [▶ 72] | X | BI object operating feedback | | | |
| FdbOut | FB_BACnetBI1203 [▶ 72] | X | BI object feedback mechanical priority operation feedback relay output | | | |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object feedback mechanical priority operation feedback hand switch | | | |
| MntnSwi | FB_BACnetBI1203 [▶ 72] | X | BI object maintenance switch | | | |
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the pump via the OWS or a local operator display | | | |
| AlmDst | FB_BA_Alarm [▶ 179] | | The function block **AlmDst** logs the fault event.<br>Actions, which should take place after arrival of the fault, can be parameterised in the template at function block **AlmDst**. | | | |
| AlmMntnSwi | FB_BA_Alarm [▶ 179] | X | The function block **AlmMntnSwi** logs the event repair switch triggered.<br>Actions to be taken after the input repair switch triggered can be parameterised in the template at function block **AlmMntnSwi**. | | | |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for specifying control of single-stage motor | | | |
| | | | **Priority:** | **Enable** | **Value** | **Comment** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** | |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** OR **Dst** OR **MntnSwi** | Value of the operator AND with the inputs **bValDst**, **MntnSwi_bPrVal**, **Dst_bPrVal** | |
| | | | PRIO_LOCAL (8) | **OpMod_udiPrVal** = **OPMOD_MAN_OFF**OR**OPMOD_MAN_STP01** | TRUE if **OpMod_udiPrVal** = **OPMOD_MAN_STP01** | In manual mode value of AV object ManSpd |
| | | | PRIO_PROGRAM (15) | TRUE | **bValPgm** | Value of input **bValPgm** |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process image | |
|---|---|---|---|---|
| BI_Motor_Disturbance | BOOL | | Input | Digital input - motor fault - message - triggered |
| BI_Feedback_Binary_Output | BOOL | X | Input | Digital input - motor switching command - feedback - on/off |

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_Pump_Feeback | BOOL | X | Input | Digital input - motor operation - message - on/off |
| BI_Motor_LocalSwitch | BOOL | X | Input | Digital input - switch manual motor - message - manual/auto |
| BI_Maintenance_Switch | BOOL | X | Input | Digital input – maintenance switch - message - triggered |
| BO_Motor | BOOL | | Output | Digital output - motor - switching command - on/off |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.63 BAC_Uni_Pu1st_01_xx

**Functional description**

The template **BAC_Uni_Pu1st_01_xx** is used for controlling a single-stage pump with binary inputs and outputs. It basically consists of a BO object for pump control and an MV object for manual control.

The template **BAC_Uni_Pu1st_01_xx** is available in different versions. Within these variants there are two basic variants. One version has an input for position reporting of the corresponding control valve, the other does not.

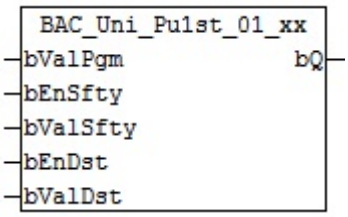
The input **rValve** is used for switching on the pump via a hysteresis module, depending on the valve position. This is required for air heaters in air-conditioning plants, for example.

| Options | Valve position (Valve) | Reserve | Maintenance switch (Rep) | Blocking protection (Block) | Mechanical priority operation feedback hand switch (A-0-H) | Mechanical priority operation feedback relay output (Rm-output) | Operating feedback (Oper) | Fault message (fault) |
|---|---|---|---|---|---|---|---|---|
| **Instance name** | **rValve** | **-** | **MntnSwi** | **AblkFnct** | **LocSwi** | **FdbOut** | **Fdb** | **Dst** |
| **Data point type** | **-** | **BI** | **BI** | **-** | **BI** | **BI** | **BI** | **BI** |
| | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| BAC_Uni_Pu1St_01_017 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| BAC_Uni_Pu1St_01_019 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| BAC_Uni_Pu1St_01_027 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| BAC_Uni_Pu1St_01_029 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

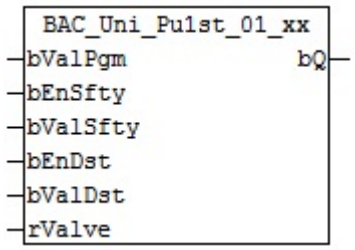
| Options | Valve po-sition (Valve) | Reserve | Mainte-nance switch (Rep) | Blocking protec-tion (Block) | Mechani-cal priority operation feedback hand switch (A-0-H) | Mechani-cal priority operation feedback relay out-put (Rm-out-put) | Operating feedback (Oper) | Fault message (fault) |
|---|---|---|---|---|---|---|---|---|
| **Instance name** | **rValve** | **-** | **MntnSwi** | **AblkFnct** | **LocSwi** | **FdbOut** | **Fdb** | **Dst** |
| **Data point type** | **-** | **BI** | **BI** | **-** | **BI** | **BI** | **BI** | **BI** |
| BAC_Uni_Pu1St_01_059 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| BAC_Uni_Pu1St_01_061 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| BAC_Uni_Pu1St_01_145 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| BAC_Uni_Pu1St_01_147 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| BAC_Uni_Pu1St_01_155 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| BAC_Uni_Pu1St_01_157 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| BAC_Uni_Pu1St_01_187 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| BAC_Uni_Pu1St_01_189 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |

**Interface1**

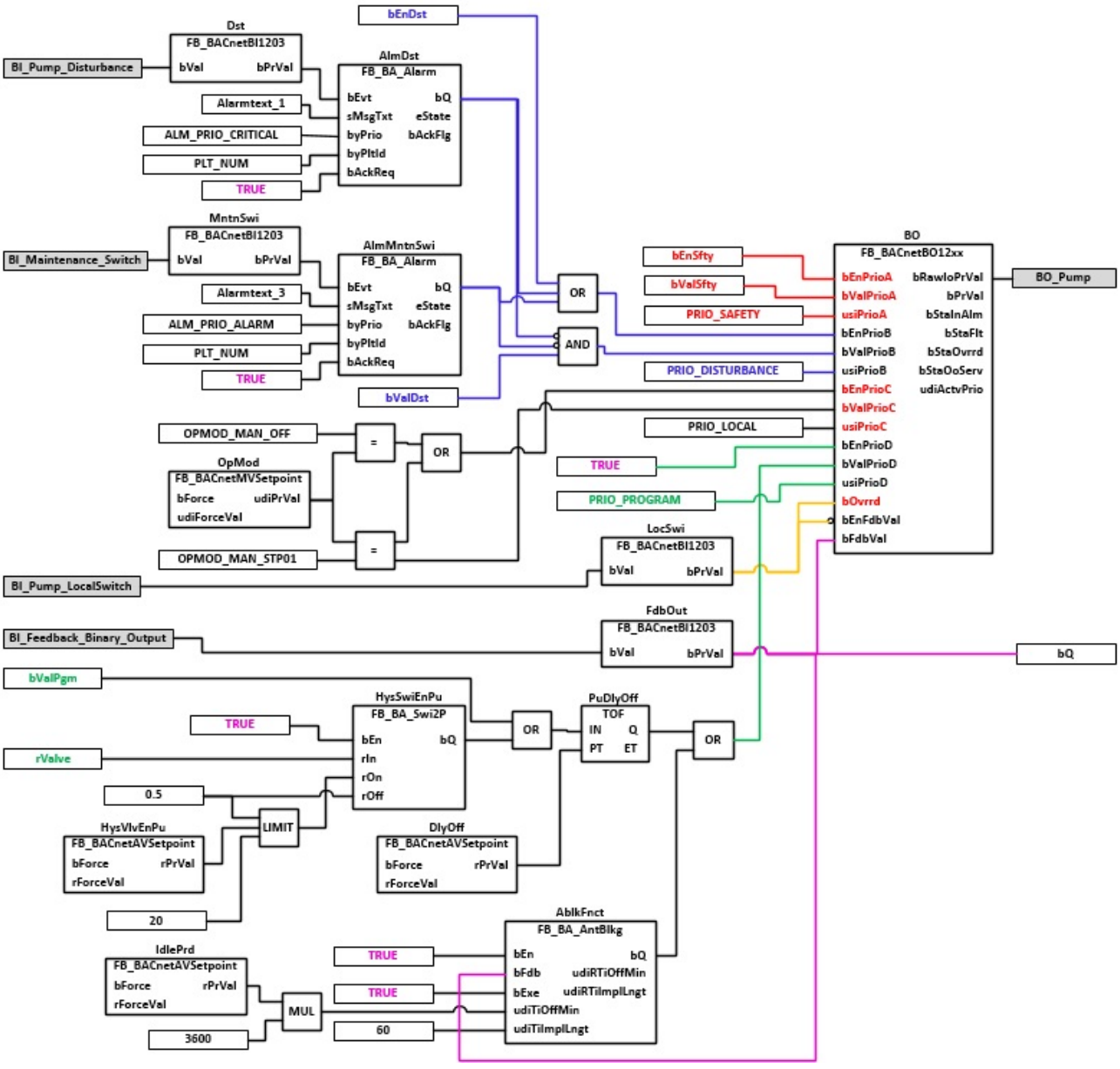
**Without** input **rValve**




**With** input **rValve**




### Block diagram version BAC_Uni_Pu1st_01_189




### VAR_INPUT

```
bValPgm     : BOOL;
bEnSfty     : BOOL;
bValSfty    : BOOL;
bEnDst      : BOOL;
bValDst     : BOOL;
rValve      : REAL;
```

**bValPgm:** binary value program priority

**bEnfty:** safety priority enable

**bValSfty:** binary value safety priority

**bEnDst:** disturbance priority enable. This input can be used to connect process feedback, for example.

**bValDst:** binary value disturbance priority. This input can be used to connect process feedback, for example.

**rValve:** input at which the valve position is connected.

### VAR_OUTPUT

```
bQ      : BOOL;
```

**bQ:** Pump switched on

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| Dst | FB_BACnetBI1203 [▶ 72] | x | BI object pump fault |
| Fdb | FB_BACnetBI1203 [▶ 72] | X | BI object operating feedback pump |
| FdbOut | FB_BACnetBI1203 [▶ 72] | X | BI object feedback mechanical priority operation feedback relay output |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object feedback mechanical priority operation feedback hand switch |
| MntnSwi | FB_BACnetBI1203 [▶ 72] | X | BI object maintenance switch |
| HysVlvEnPu | FB_BACnetAVSetpoint [▶ 69] | x | AV object for input of the hysteresis value for the two-point switch **HysSwiEnPu**, in order to switch the pump via the valve position **rValve**. If the value 0 is entered, the pump is not switched on via the valve position. |
| DlyOff | FB_BACnetAVSetpoint [▶ 69] | | AV object for input of the overrun time value. |
| IdlePrd | FB_BACnetAVSetpoint [▶ 69] | | AV object for input of the maximum pump standstill duration until a blocking protection pulse is issued. |
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the pump via the MCL or a local operator display |

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| AlmDst | FB_BA_Alarm [▶ 179] | | The **AlmDst** function block records the pump fault event. Actions that are to take place after the pump fault notification is received can be parameterized in the template at the **AlmDst** function block. | | |
| AlmMntnSwi | FB_BA_Alarm [▶ 179] | X | The function block **AlmMntnSwi** records the event repair switch triggered. Actions to be taken after the input Repair switch triggered can be parameterized in the template at function block **AlmMntnSwi**. | | |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for controlling the pump | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** |
| | | | PRIO_DISTURBANCE (3) | The OR module pools events which enable writing to the disturbance priority of the downstream BO object. Events: 1. Motor fault from the function block **AlmDst** 2. Input template **bEnDst** 3. Repair switch of function block **AlmMntnSwi** | At the input of the templates **bEnDst** the pump can be forced to switch on or off, e.g. when there is risk of frost at the air heater. However, if a pump fault is pending or the repair switch was triggered, forced switch-on is blocked at the AND module. |
| | | | PRIO_LOCAL (8) | The OR module pools events which enable writing to the priority manual override (local) of the downstream BO object. Events: 1. The MV object has the value **OPMOD_MAN_OFF** (manual off) 2. The MV object has the value **OPMOD_MAN_STP01**(manual on) | TRUE, if **OpMod_udiPrVal** = **OPMOD_MAN_STP01** |

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| | | | PRIO_PROGRAM (15) | TRUE | The OR module pools events which enable writing to the priority Pgm (program) of the downstream BO object. Events: 1. External request of the pump from the inputs **bValPgm** or **rValve**. The pump delay time **PuDlyOff** may still be active. 2. Blocking protection function of function block **AblkFnct** is active |
| HysSwiEnPu | FB_BA_Swi2P [▶ 146] | x | Two-point switch, which switches the pump on or off by means of a hysteresis **HysVlvEnPu**, depending on the valve position **rValve**. If the valve position is < 0.5, the pump is switched off. | | |
| PuDlyOff | TOF | | Timing element for pump overrun | | |
| AblkFnct | FB_BA_AntBlkg [▶ 215] | | Function block for output of a blocking protection pulse | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| BI_Pump_Disturbance | BOOL | | Input | Pump - display - fault |
| BI_Feedback_Binary_Output | BOOL | X | Input | Pump switch H-0-A - message - operation |
| BI_Pump_Feeback | BOOL | X | Input | Pump - message - operation |
| BI_Pump_LocalSwitch | BOOL | X | Output | Pump switch H-0-A - message - operation mode |
| BI_Maintenance_Switch | BOOL | X | Input | Pump - display - maintenance switch |
| BO_Pump | BOOL | | Output | Pump - switching command - on |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

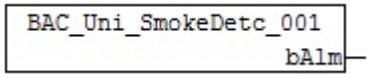
## 9.64 BAC_Uni_SmokeDetc_001

### Functional description

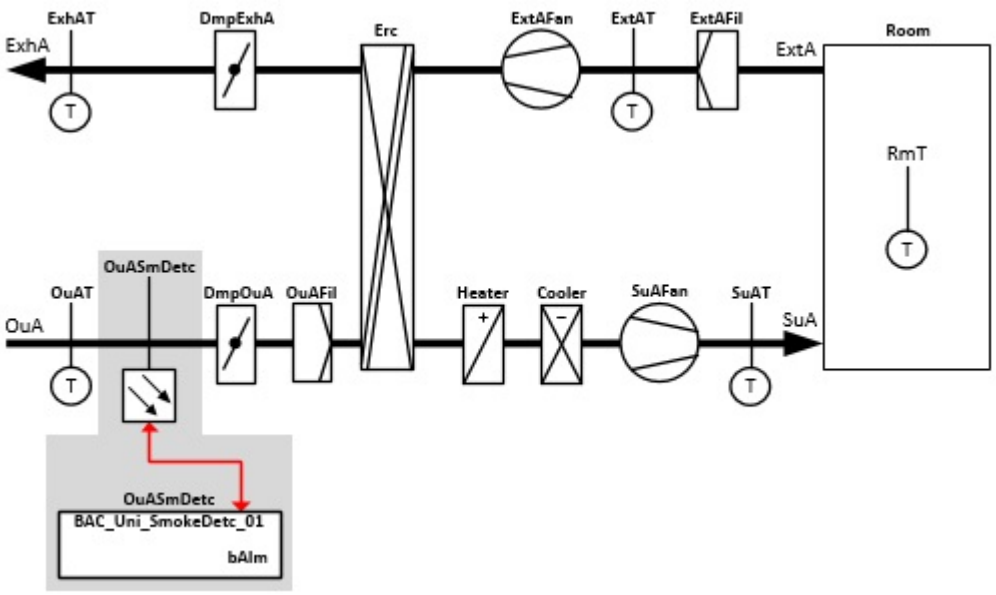
The template **BAC_Uni_SmokeDetc_001** is used to control and monitor a duct smoke detector.
It consists of two binary input objects for pollution and smoke alarm and one binary output object for resetting the duct smoke detector.

**The BACnet property MinimumOntime of the BO object is set to a default value for the acknowledgment of the duct smoke detector. The acknowledgment pulse is extended by this time setting.**

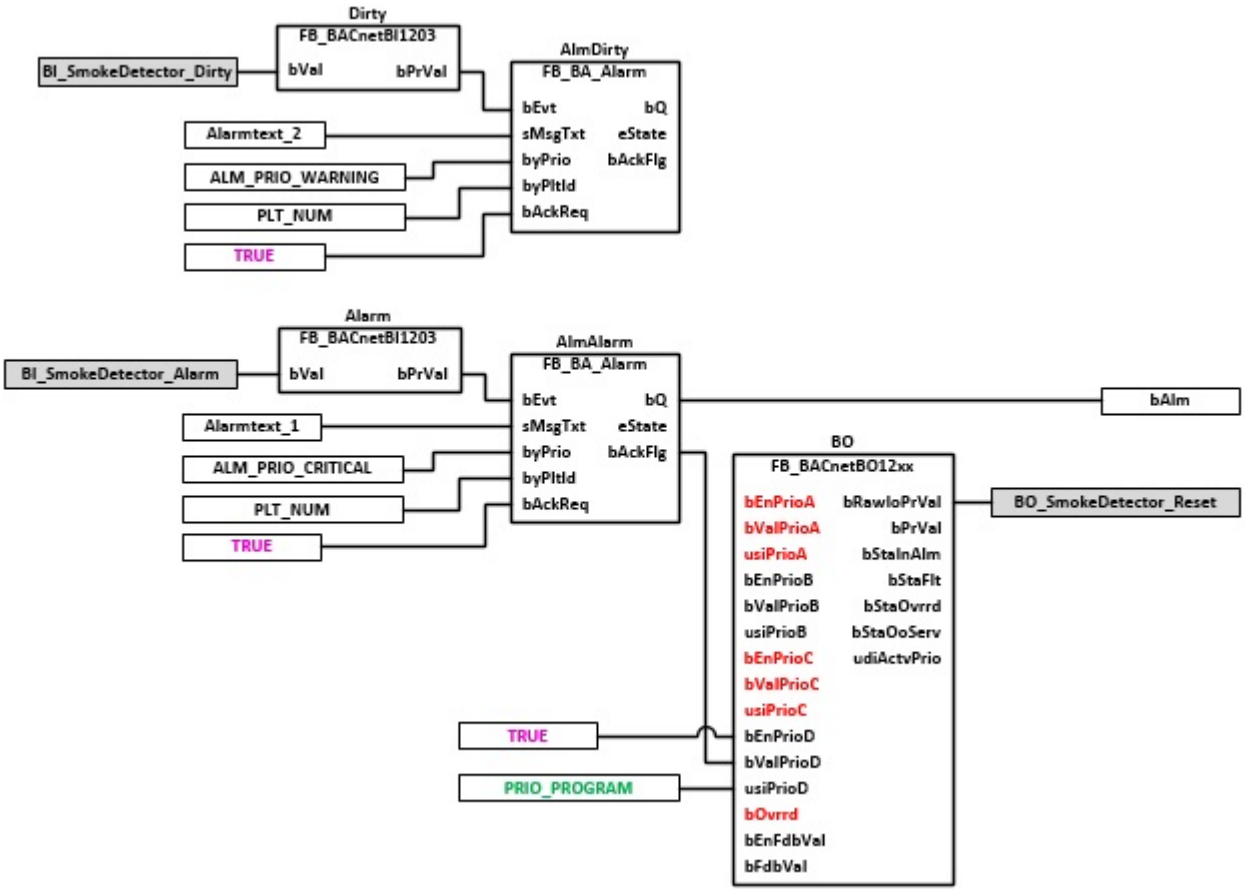
### Interface




### Plant diagram




### Block diagram




### VAR_OUTPUT

```
bAlm     : BOOL;
```

**bAlm:** this output indicates that the smoke detector has been triggered.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task | | |
|---|---|---|---|---|---|
| Dirty | FB_BACnetBI1203 [▶ 72] | | BI object dirty smoke detector | | |
| AlmDirty | FB_BA_Alarm [▶ 179] | | The function block **AlmDirty** detects the dirty smoke detector event.<br>Actions that should take place after receipt of the dirty smoke detector notification can be parameterized in the template on the function block **AlmDirty**. | | |
| Alarm | FB_BACnetBI1203 [▶ 72] | | BI object smoke detector alarm | | |
| AlmAlarm | FB_BA_Alarm [▶ 179] | | The function block **AlmAlarm** detects the smoke detector alarm event.<br>Actions that should take place after receipt of the smoke detector alarm can be parameterized in the template on the function block **AlmAlarm**. | | |
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for the acknowledge pulse of the duct smoke detector. | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_PROGRAM (15) | TRUE | The system acknowledge pulse is passed through to the BO object via the alarm logging **AlmAlarm**. |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_SmokeDetector_Dirty | BOOL | | Input | Digital input – smoke detector – message – dirty |
| BI_SmokeDetector_Alarm | BOOL | | Input | Digital input – smoke detector – message – alarm |

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| BO_SmokeDetector_Reset | BOOL | | Output | Digital output – smoke detector – switching command – acknowledge on/off |

### Version history

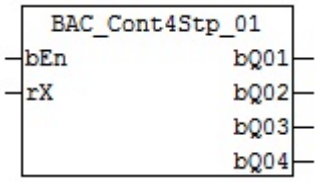
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.65 BAC_Cont4Stp_01

### Functional description

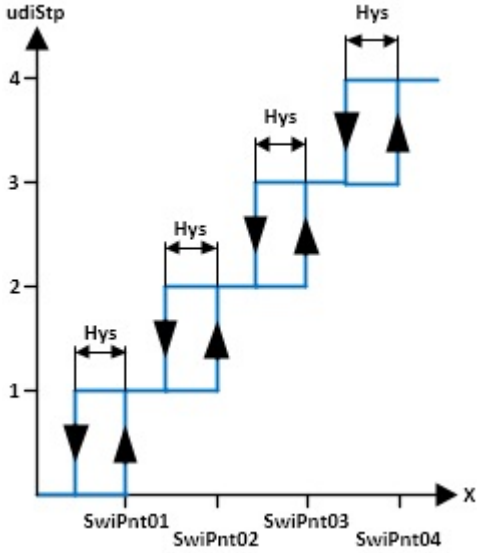
The template determines the resulting switching stages of a multi-level unit, depending on the input signal. Four switch-on thresholds and a hysteresis can be parameterized.

### Interface




### Diagram 01

Direction of action *Cont4Stp_bActn* = FALSE = reverse = heating

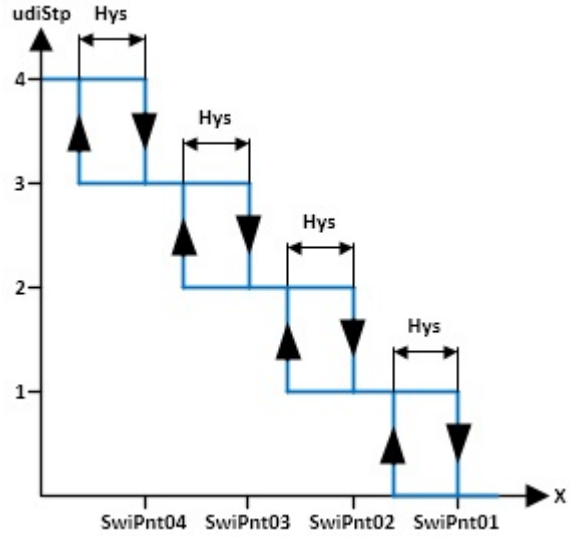



| DspStp | Cont4Stp_udiStp | SwiOn | SwiOff | Cont4Stp_udiRemTiDylOn | Cont4Stp_udiRemTiDylOff | bQ01 | bQ02 | bQ03 | bQ04 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | SwiPnt01 | SwiPnt01 - Hys | DlyOn | 0 | FALSE | FALSE | FALSE | FALSE |
| 1 | >= 1 | SwiPnt02 | SwiPnt01 - Hys | DlyOn | DlyOff | TRUE | FALSE | FALSE | FALSE |
| 2 | >= 2 | SwiPnt03 | SwiPnt02 - Hys | DlyOn | DlyOff | TRUE | TRUE | FALSE | FALSE |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | >= 3 | SwiPnt04 | SwiPnt03 - Hys | DlyOn | DlyOff | TRUE | TRUE | TRUE | FALSE |
| 4 | >= 4 | SwiPnt04 | SwiPnt04 - Hys | 0 | DlyOff | TRUE | TRUE | TRUE | TRUE |

**Diagram 02**

Direction of action *Cont4Stp_bActn* =TRUE = direct = cooling



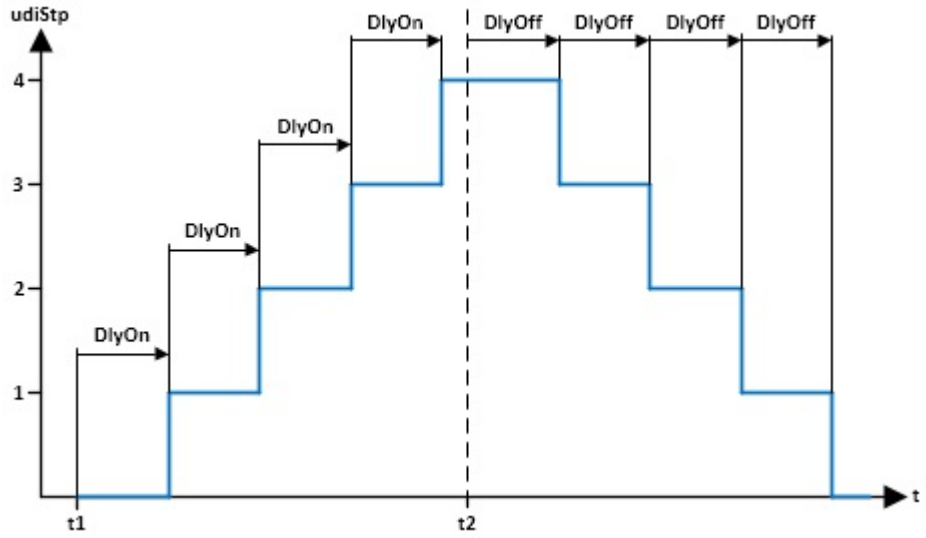

| DspStp | Cont4Stp_udiStp | SwiOn | SwiOff | Cont4Stp_udiRemTiDlyOn | Cont4Stp_udiRemTiDlyOff | bQ01 | bQ02 | bQ03 | bQ04 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | SwiPnt01 | SwiPnt01 + Hys | DlyOn | 0 | FALSE | FALSE | FALSE | FALSE |
| 1 | >= 1 | SwiPnt02 | SwiPnt01 +Hys | DlyOn | DlyOff | TRUE | FALSE | FALSE | FALSE |
| 2 | >= 2 | SwiPnt03 | SwiPnt02 + Hys | DlyOn | DlyOff | TRUE | TRUE | FALSE | FALSE |
| 3 | >= 3 | SwiPnt04 | SwiPnt03 + Hys | DlyOn | DlyOff | TRUE | TRUE | TRUE | FALSE |
| 4 | 4 | SwiPnt04 | SwiPnt04 + Hys | 0 | DlyOff | TRUE | TRUE | TRUE | TRUE |

**Diagram 03**

Timing of the switch-on and switch-off delays

At time t1 *r*X changes from £*SwiPnt01* to *SwiPnt04*

At time t2 *r*X changes from *SwiPnt04* to £*SwiPnt01* – *Hys*




**Inputs/outputs**

### VAR_INPUT

```
bEn       : BOOL;
rX        : REAL;
```

**bEn: G**eneral enable of the function block. If *bEn* is FALSE, all outputs are set to 0.

**rX:** Input value, from which the switching state is derived.

### VAR_OUTPUT

```
bQ01        : BOOL;
bQ02        : BOOL;
bQ03        : BOOL;
bQ04        : BOOL;
```

**bQ01:** Shows status level 01
TRUE = ON; FALSE = OFF
udiStp >= 1

**bQ02:** Shows status level 02
TRUE = ON; FALSE = OFF
udiStp >= 2

**bQ03:** Shows status level 03
TRUE = ON; FALSE = OFF
udiStp >= 3

**bQ04:** Shows status level 04
TRUE = ON; FALSE = OFF
udiStp >= 4

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| En | FB_BACnetBV1204 [▶ 94] | General enable of the function block. If *bEn* is FALSE, all outputs are set to 0 |
| X | FB_BACnetAVSetpoint [▶ 69] | Input value, from which the switching state is derived |
| Hys | FB_BACnetAVSetpoint [▶ 69] | Input absolute value hysteresis |
| DlyOn | FB_BACnetAVSetpoint [▶ 69] | Start-up delay of the stages in seconds |
| DlyOff | FB_BACnetAVSetpoint [▶ 69] | Switch-off delay of the stages in seconds |
| SwiPnt01 | FB_BACnetAVSetpoint [▶ 69] | Input switch-on point step 01, internally limited to between 0 and 100 in the template |
| SwiPnt02 | FB_BACnetAVSetpoint [▶ 69] | Input switch-on point step 02, internally limited to between 0 and 100 in the template |
| SwiPnt03 | FB_BACnetAVSetpoint [▶ 69] | Input switch-on point step 03, internally limited to between 0 and 100 in the template |
| SwiPnt04 | FB_BACnetAVSetpoint [▶ 69] | Input switch-on point step 04, internally limited to between 0 and 100 in the template |
| Cont4Stp | FB_BA_Cont4Stp01 [▶ 137] | The function block Cont4Stp is the core of the step switch template. |
| DspStp | FB_BACnetMVDisplay [▶ 129] | Display active step |
| SwiOn | FB_BACnetAVDisplay [▶ 69] | Display active switch-on point |
| SwiOff | FB_BACnetAVDisplay [▶ 69] | Display active switch-off point |
| | LIMIT, LIMIT, LIMIT, LIMIT | Limitation of the switch-on points step 1 - 4 to 0 - 100 |

### Versionshistorie

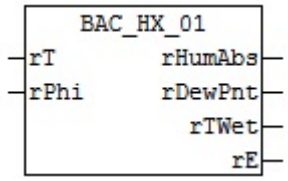
| Versionsnummer | Bemerkungen |
|---|---|
| 1.0.1 | First release |

## 9.66 BAC_HX_01

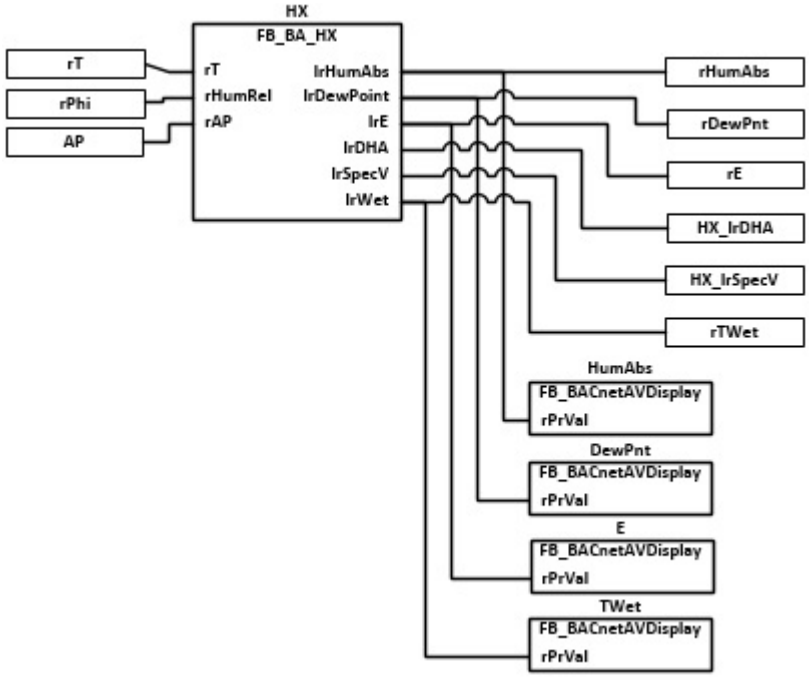
### Functional description

This template is used to calculate the **dew point temperature**, the **specific enthalpy**, the **wet bulb temperature,** and the **absolute humidity**.
The temperature, the relative humidity and the barometric pressure are required for calculating these parameters.

### Interface



### Block diagram




### VAR_INPUT

```
rT        : REAL;
rPhi      : REAL;
```

**rT:** Current temperature value [°C]

**rPhi:** Current relative humidity value [%]

### VAR_OUTPUT

```
rHumAbs    : REAL;
rDewPnt    : REAL;
rTWet      : REAL;
rE         : REAL;
```

**rHumAbs:** Calculated value **Absolute humidity g water per kg dry air [g/kg]**

**rDewPnt:** Calculated value **Dew point temperature [°C]**

**rTWet:** Calculated value **Wet bulb temperature [°C]**

**rE:** Calculated value **Enthalpy [kJ/kg]**

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| HX | FB_BA_HX [▶ 227] | | This function block calculates the dew point temperature, the specific enthalpy, the wet bulb temperature and the absolute humidity.<br>For the calculation of the quantities, the temperature, the relative humidity and the barometric air pressure are required. |
| HumAbs | FB_BACnetAVDisplay [▶ 69] | | Display of the calculated value of absolute humidity g water per kg of dry air [g/kg] |
| DewPnt | FB_BACnetAVDisplay [▶ 69] | | Display of the calculated dew point temperature value [°C] |
| TWet | FB_BACnetAVDisplay [▶ 69] | | Display of the calculated wet bulb temperature value [°C] |
| E | FB_BACnetAVDisplay [▶ 69] | | Display of the calculated enthalpy value [kJ/kg] |
| TLogHum Abs | FB_BACnetTLog1201 [▶ 135] | | Trend logging for the AV object **HumAbs** absolute humidity |

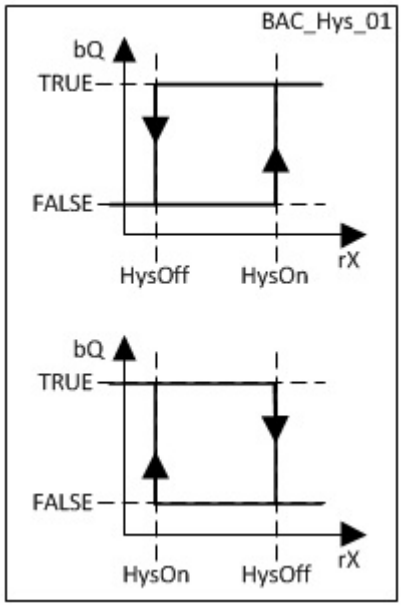
### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

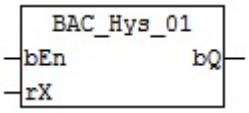
## 9.67 BAC_Hys_01

### Functional description

The template represents a hysteresis function with fixed switching points.



### Interface




### VAR_INPUT

```
bEn    : BOOL;
rX     : REAL;
```

**bEn:** Enable

**rX:** Actual value

### VAR_OUTPUT

```
bQ     : BOOL;
```

**bQ:** Output current state of the hysteresis-function

### Program description

| Instance | Type | Task |
|---|---|---|
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the hysteresis function enable |
| X | FB_BACnetAVDisplay [▶ 69] | The AV object indicates the actual value, which is linked to the template via the input variable rX |
| HysOn | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the upper limit of the hysteresis function |
| HysOff | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the lower limit of the hysteresis function |
| Hys | FB_BA_Swi2P [▶ 146] | The function block Hys is the core of the hysteresis function |
| Q | FB_BACnetBVDisplay [▶ 95] | The BV object indicates the current state of the hysteresis function |

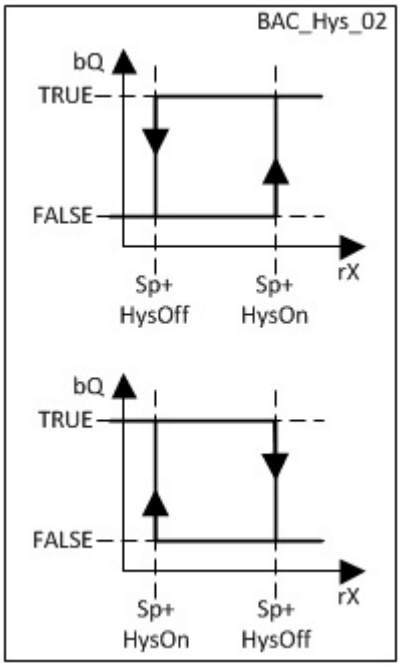
### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

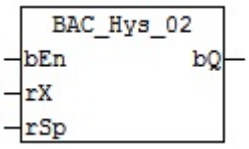
## 9.68 BAC_Hys_02

### Functional description

The template represents a hysteresis function with dynamic switching points.




### Interface




### VAR_INPUT

```
bEn    : BOOL;
rX     : REAL;
```

**bEn:** Enable

**rX:** Actual value

**rSp:** Set value

### VAR_OUTPUT

```
bQ     : BOOL;
```

**bQ:** Output current state of the hysteresis-function

### Program description

| Instance | Type | Task |
|---|---|---|
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the hysteresis function enable |
| X | FB_BACnetAVDisplay [▶ 69] | The AV object indicates the actual value, which is linked to the template via the input variable rX |
| Sp | FB_BACnetAVDisplay [▶ 69] | The AV object indicates the set value, which is linked to the template via the input variable rSp |
| HysOn | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the upper limit of the hysteresis function |
| HysOff | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the lower limit of the hysteresis function |
| Hys | FB_BA_Swi2P [▶ 146] | The function block Hys is the core of the hysteresis function |

| Instance | Type | Task |
|---|---|---|
| Q | FB_BACnetBVDisplay [▶ 95] | The BV object indicates the current state of the hysteresis function |

### Version history

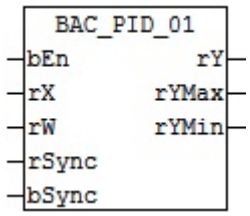
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.69 BAC_PID_01

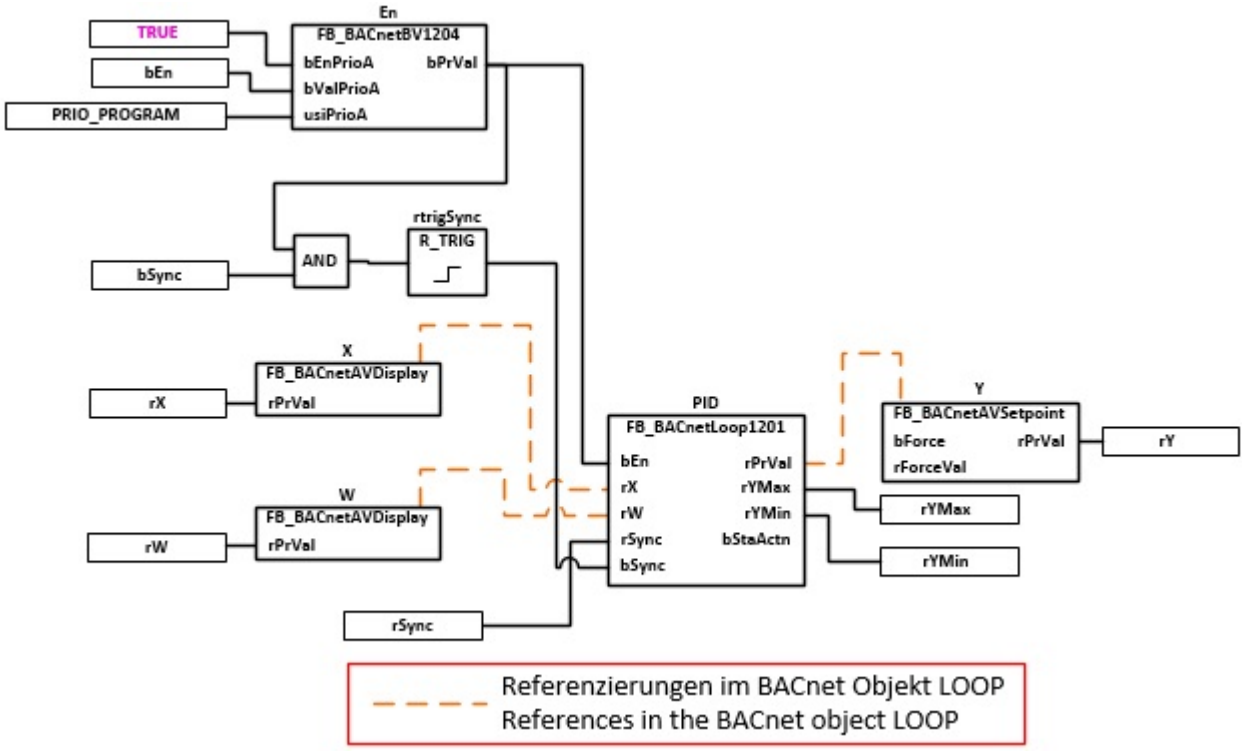
### Functional description

The template BAC_PID_01 is a universal PID controller.
The set value, actual value and control output are referenced via the BACnet value objects X, W and Y.
The PID controller is enabled via the input variable **bEn**.

### Interface




### Block diagram




### VAR_INPUT

```
bEn    : BOOL;
rX     : REAL;
rW     : REAL;
rSync  : REAL;
bSync  : BOOL;
```

**bEn:** Enable

**rX:** Actual value

**rW:** Set value

**rSync:** Synchronisation value

**bSync:** Start synchronisation

### VAR_OUTPUT

```
rY          : REAL;
rYMax       : REAL;
rYMin       : REAL;
```

**rY:** Control value output

**rYMax:** Maximum controller value

**rYMin:** Minimum controller value

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| X | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the actual value input of the BACnet loop object |
| W | FB_BACnetAVDisplay [▶ 69] | The AV object displays the setpoint of the input **rW** and is referenced to the setpoint input of the BACnet loop object |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the controller enable in the MCL or in a local operator display |
| PID | FB_BACnetLoop1201 [▶ 98] | PID controller |
| rtrigSync | R_TRIG | A rising edge at input bSync triggers synchronization of the loop object to the value of rSync. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |

### Version history

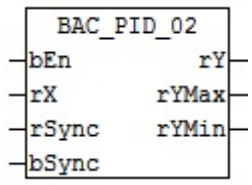
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.70 BAC_PID_02

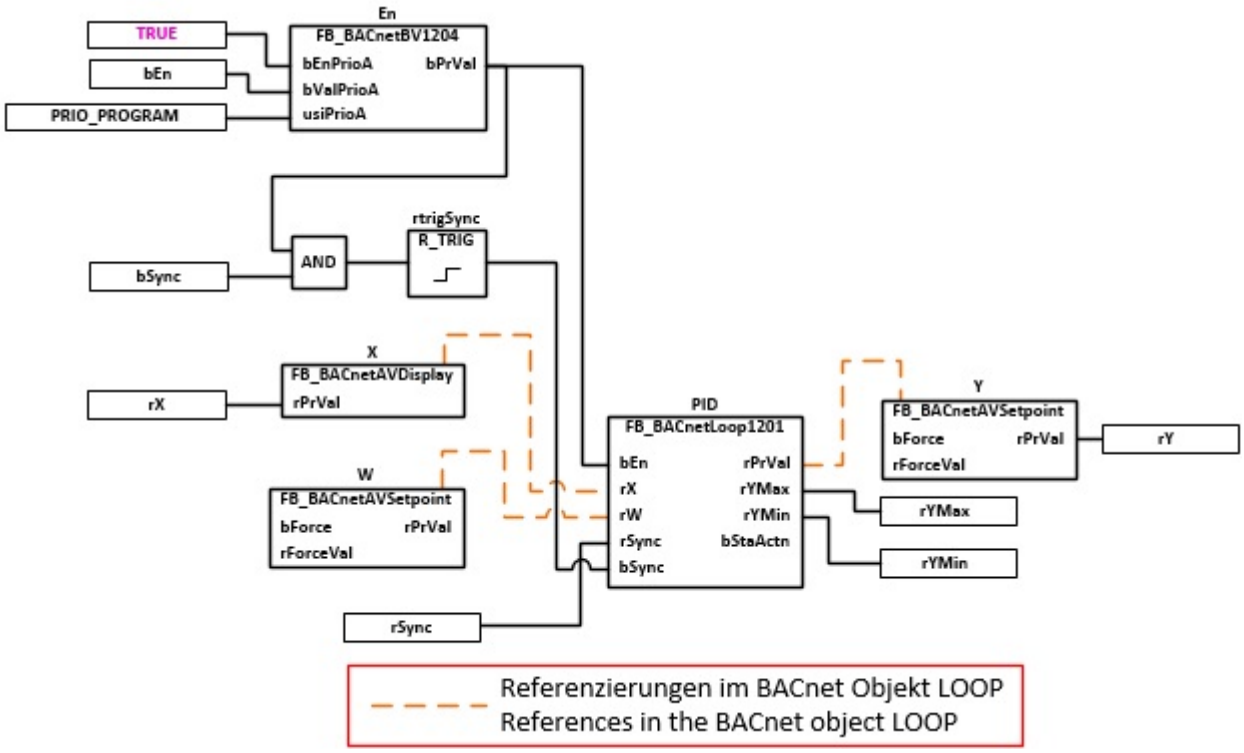
### Functional description

The template BAC_PID_02 is a universal PID controller.
The set value, actual value and control output are referenced via the BACnet value objects X, W and Y.
The PID controller is enabled via the input variable **bEn**.

### Interface




### Block diagram




### VAR_INPUT

```
bEn       : BOOL;
rX        : REAL;
rSync     : REAL;
bSync     : BOOL;
```

**bEn:** Enable

**rX:** Actual value

**rW:** Set value

**rSync:** Synchronization value

### VAR_OUTPUT

```
rY          : REAL;
rYMax       : REAL;
rYMin       : REAL;
```

**rY:** Control value output

**rYMax:** Maximum controller value

**rYMin:** Minimum controller value

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| X | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the actual value input of the BACnet loop object |
| W | FB_BACnetAVSetpoint [▶ 69] | The setpoint is entered via the AV object. It is referenced to the setpoint input of the BACnet loop object |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the controller enable in the MCL or in a local operator display |
| PID | FB_BACnetLoop1201 [▶ 98] | PID controller |
| rtrigSync | R_TRIG | A rising edge at input bSync triggers synchronization of the loop object to the value of rSync. |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

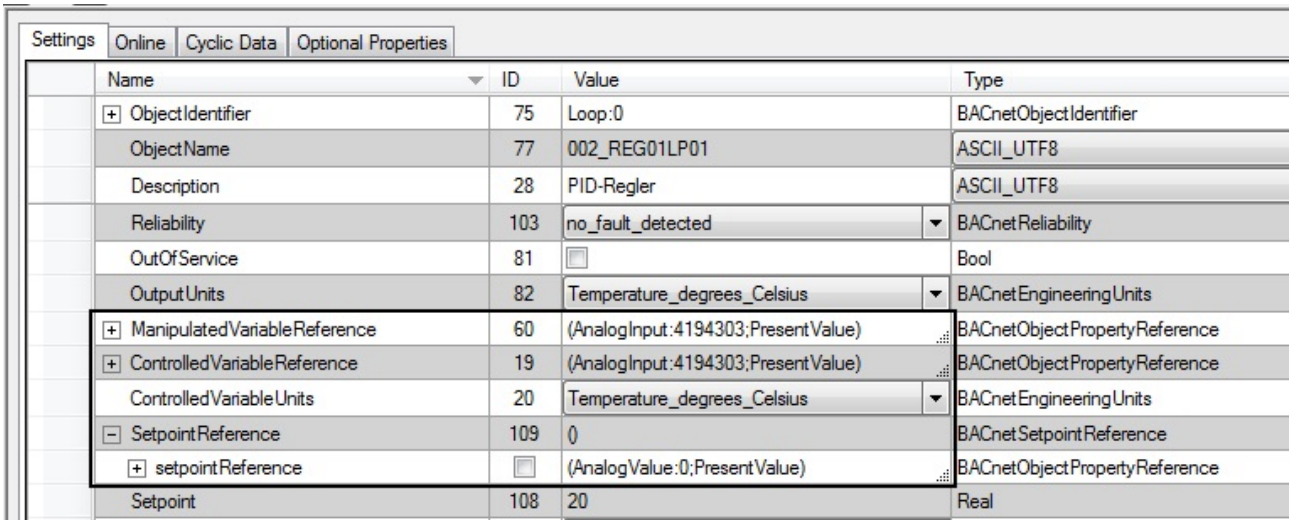
## 9.71 BAC_PID_03

### Functional description

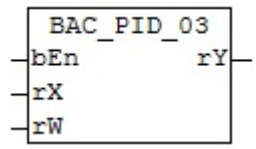
The template BAC_PID_03 is a universal PID controller.
The PID controller is enabled using the input variable **bEn**.

The LOOP object does not require any referencing objects for the setpoint, actual value and control output. For this the PLC comment syntax must be specified as follows:
**(BACnet_ManipulatedVariableReference : undefined : )**
**(BACnet_ControlledVariableReference : undefined : )**
**(BACnet_SetpointReference : undefined : )**

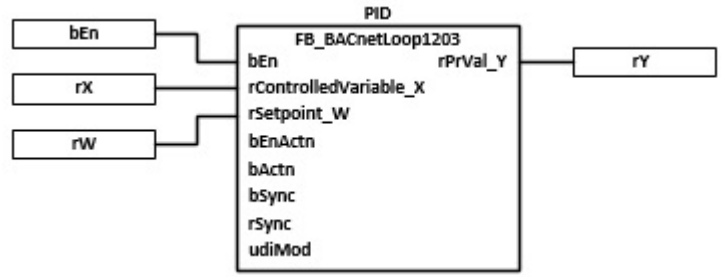
In the System Manager the undefined comments are converted as shown in the picture:




### Interface




### Block diagram




### VAR_INPUT

```
bEn    : BOOL;
rX     : REAL;
rW     : REAL;
```

**bEn:** enable

**rX:** actual value

**rW:** setpoint

### VAR_OUTPUT

```
rY     : REAL;
```

**rY:** control value output

### Program description

| Instance | Type | Task |
|---|---|---|
| PID | FB_BACnetLoop1203 | PID controller |

### Version history

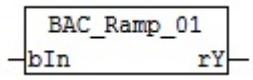
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.72 BAC_Ramp_01

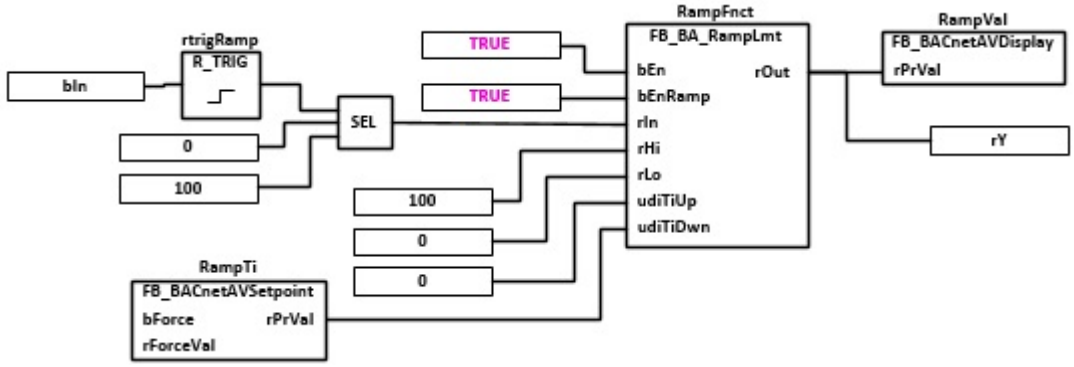
### Functional description

The template represents a falling ramp limitation. A rising edge at input **bIn** triggers the ramp function. The output variable **rY** assumes the value 100. The value decreases linearly to 0, based on the fall time **RampTi**.

### Interface




### Block diagram




### VAR_INPUT

```
bIn       : BOOL;
```

**bIn:** A rising edge at this input triggers the ramp function.

### VAR_OUTPUT

```
rY        : REAL;
```

**rY:** Output of the ramp function

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| RampTi | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the fall time |
| RampFnct | FB_BA_RampLmt [▶ 141] | The function block RampFnct is the core of the ramp function. |
| RampVal | FB_BACnetAVDisplay [▶ 69] | The AV object indicates the output of the ramp function |
| rtrigRamp | R_TRIG<br>SEL | Through a rising edge at input **bIn** this network supplies the function block **RampFnct** with the value 100 for one cycle, thereby triggering the ramp function. |

### Version history

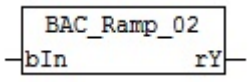
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.73 BAC_Ramp_02

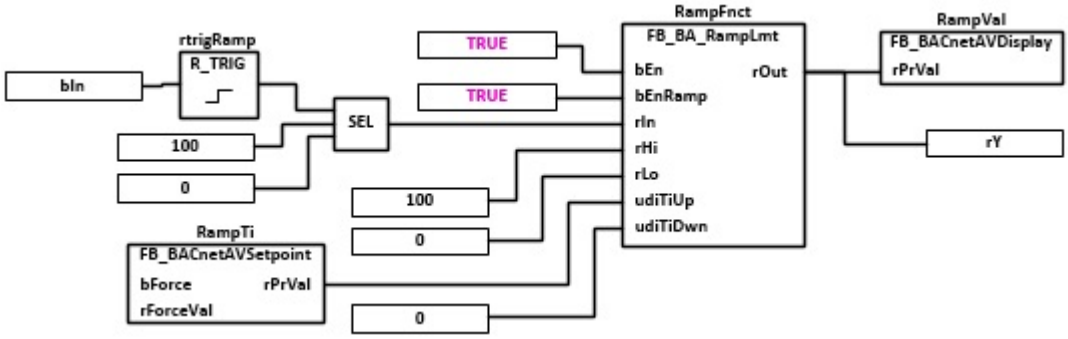
### Functional description

The template represents a rising ramp limitation. A rising edge at input **bIn** triggers the ramp function. The output variable **rY** assumes the value 0. The value increases linearly to 100, based on the rise time **RampTi**.

### Interface




### Block diagram




### VAR_INPUT

```
bIn       : BOOL;
```

**bIn:** A rising edge at this input triggers the ramp function.

### VAR_OUTPUT

```
rY          : REAL;
```

**rY:** Output of the ramp function

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| RampTi | FB_BACnetAVSetpoint [▶ 69] | AV object for input of the rise time |
| RampFnct | FB_BA_RampLmt [▶ 141] | The function block RampFnct is the core of the ramp function. |
| RampVal | FB_BACnetAVDisplay [▶ 69] | The AV object indicates the output of the ramp function |
| rtrigRamp | R_TRIG<br>SEL | Through a rising edge at input **bIn** this network supplies the function block **RampFnct** with the value 0 for one cycle, thereby triggering the ramp function. |

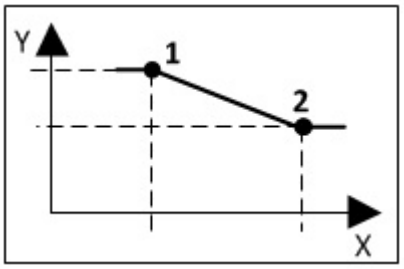
### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

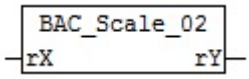
## 9.74 BAC_Scale_02

### Functional description

The template *BAC_Scale_02* represents a linear interpolation with two interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [*X1/Y1*] to [*X2/Y2*].




### Interface




### VAR_INPUT

```
rX      : REAL;
```

**rX:** Input value of the characteristic curve

### VAR_OUTPUT

```
rY      : REAL;
```

**rY:** Calculated output value of the characteristic curve

### Program description

| Instance | Type | Task |
|---|---|---|
| X1 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X1 |
| X2 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X2 |
| Y1 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y1 |
| Y2 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y2 |
| Scale | FB_BA_Chrct02 [▶ 171] | The function block FB_BA_Chrct02 represents a linear interpolation with two interpolation points and can be used to generate a characteristic curve |
| Y | FB_BACnetAVDisplay [▶ 69] | Output of the calculated output value of the characteristic curve |

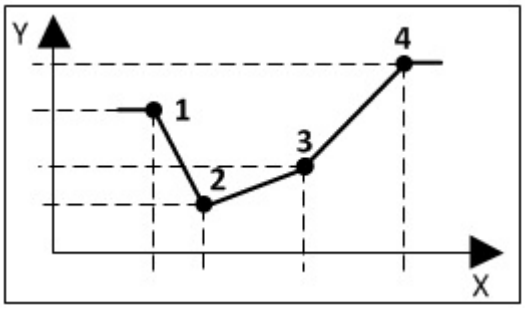
### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

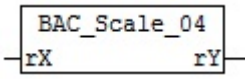
## 9.75 BAC_Scale_04

### Functional description

The template *BAC_Scale_04* represents a linear interpolation with 4 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [*X1/Y1*] to [*X4/Y4*].




### Interface




### VAR_INPUT

```
rX      : REAL;
```

**rX:** Input value of the characteristic curve

### VAR_OUTPUT

```
rY      : REAL;
```

**rY:** Calculated output value of the characteristic curve

### Program description

| Instance | Type | Task |
|---|---|---|
| X1 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X1 |
| X2 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X2 |
| X3 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X3 |
| X4 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X4 |
| Y1 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y1 |
| Y2 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y2 |
| Y3 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y3 |
| Y4 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y4 |
| Scale | FB_BA_Chrct04 [▶ 172] | The function block FB_BA_Chrct04 represents a linear interpolation with four interpolation points and can be used to generate a characteristic curve |
| Y | FB_BACnetAVDisplay [▶ 69] | Output of the calculated output value of the characteristic curve |

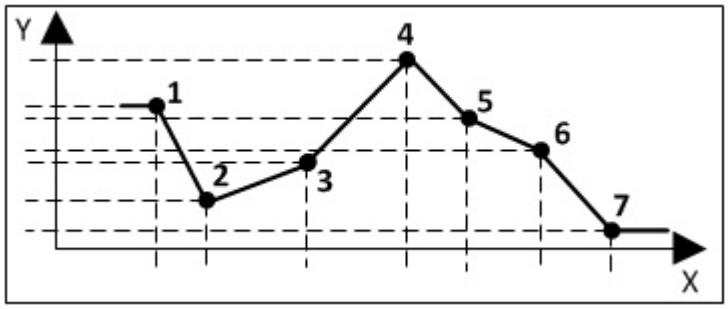
### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

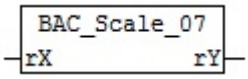
# 9.76 BAC_Scale_07

### Functional description

The template *BAC_Scale_07* represents a linear interpolation with 7 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points [*X1/Y1*] to [*X7/Y7*].




### Interface




### VAR_INPUT

```
rX      : REAL;
```

**rX:** Input value of the characteristic curve

### VAR_OUTPUT

```
rY      : REAL;
```

**rY:** Calculated output value of the characteristic curve

### Program description

| Instance | Type | Task |
|---|---|---|
| X1 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X1 |
| X2 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X2 |
| X3 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X3 |
| X4 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X4 |
| X5 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X5 |
| X6 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X6 |
| X7 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point X7 |
| Y1 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y1 |
| Y2 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y2 |

| Instance | Type | Task |
|---|---|---|
| Y3 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y3 |
| Y4 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y4 |
| Y5 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y5 |
| Y6 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y6 |
| Y7 | FB_BACnetAVSetpoint [▶ 69] | The AV object is used to specify the value for interpolation point Y7 |
| Scale | FB_BA_Chrct07 [▶ 174] | The function block FB_BA_Chrct07 represents a linear interpolation with seven interpolation points and can be used to generate a characteristic curve |
| Y | FB_BACnetAVDisplay [▶ 69] | Output of the calculated output value of the characteristic curve |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.77 BAC_Uni_Vlv_01_xx

### Functional description

The template **BAC_Uni_Vlv_01_xx** is used for controlling an analog control valve. It basically consists of an AO object including a trend object, an MV object for manual control, and the associated AV object for entering the manual position. The template is complemented through optional BACnet objects, see table with version overview.

The two output variables rPrVal / bSync are only active, if the valve position feedback Fdb exists in the template that is used. If this is not the case, the two versions return the value zero. The interface of the templates BAC_Uni_Vlv3P_01_xx / BAC_Uni_Vlv_01_xx is the same. For this reason, in the Project Builder in a call template a three-point valve can be replaced with an analog valve, without having to adjust the PLC code.

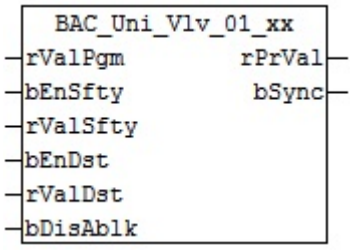
### Versions

The template **BAC_Uni_Vlv_01_xx** is available in different versions.
The control valve versions are identified by means of a key. The identification key is derived from the table below.

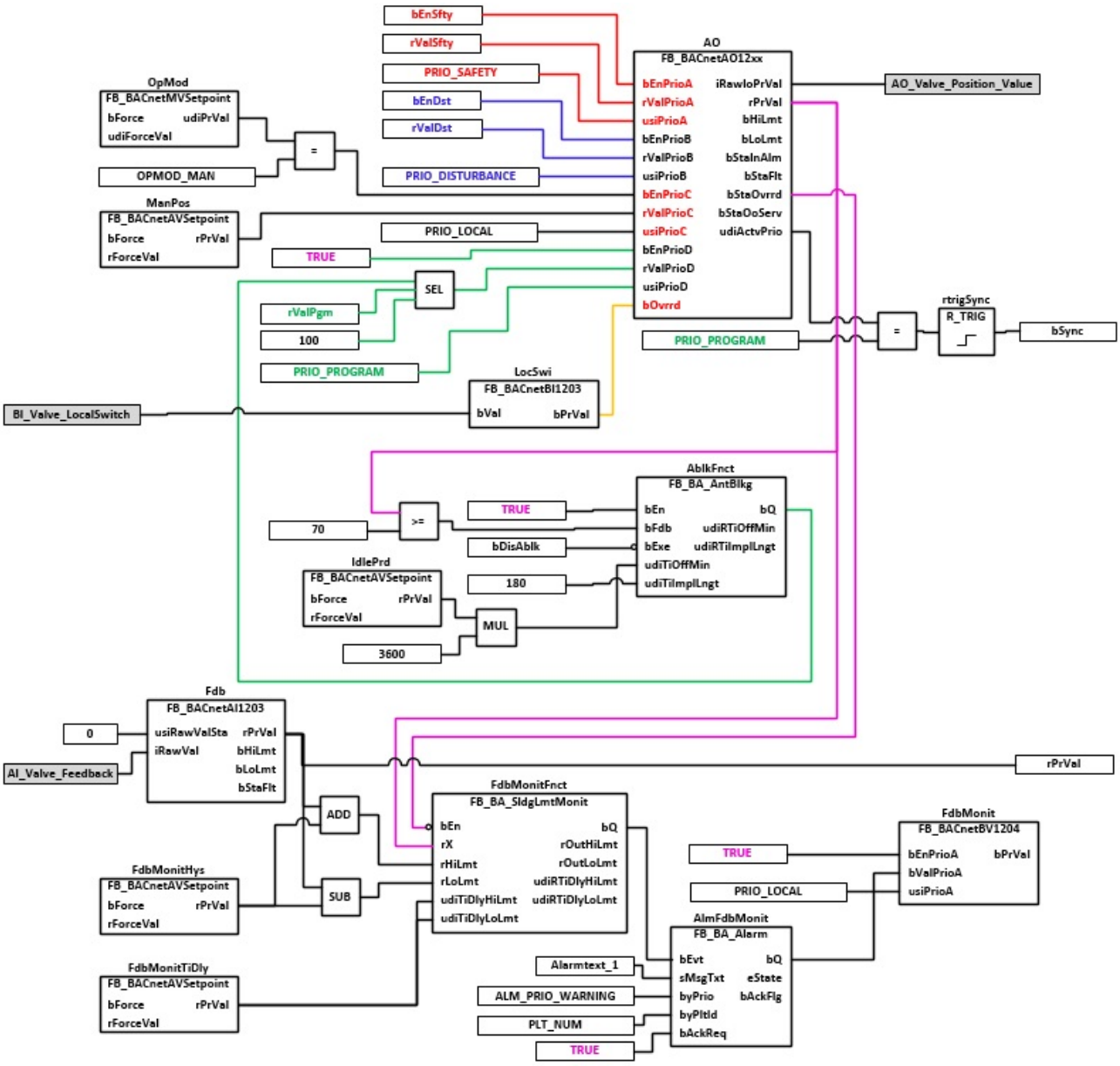
| Options | Blocking protection (Block) | Feedback Valve position with monitoring (Rm) | mechanical priority operation position feedback potentiometer (Rm-output) | mechanical priority operation position feedback hand switch (A-H) |
|---|---|---|---|---|
| **Instance** | **AblkFnct** | **Fdb** | **FdbOut** | **LocSwi** |
| **Data point type** | **-** | **AI** | **AI** | **BI** |
| | **8** | **4** | **2** | **1** |
| BAC_Uni_Vlv_01_00 | 0 | 0 | 0 | 0 |
| BAC_Uni_Vlv_01_01 | 0 | 0 | 0 | 1 |

| Options | Blocking protection<br><br>(Block) | Feedback Valve position with monitoring<br><br>(Rm) | mechanical priority operation position feedback potentiometer (Rm-output) | mechanical priority operation position feedback hand switch (A-H) |
|---|---|---|---|---|
| **Instance** | **AblkFnct** | **Fdb** | **FdbOut** | **LocSwi** |
| **Data point type** | **-** | **AI** | **AI** | **BI** |
| | **8** | **4** | **2** | **1** |
| BAC_Uni_Vlv_01_03 | 0 | 0 | 1 | 1 |
| BAC_Uni_Vlv_01_04 | 0 | 1 | 0 | 0 |
| BAC_Uni_Vlv_01_05 | 0 | 1 | 0 | 1 |
| BAC_Uni_Vlv_01_08 | 1 | 0 | 0 | 0 |
| BAC_Uni_Vlv_01_09 | 1 | 0 | 0 | 1 |
| BAC_Uni_Vlv_01_11 | 1 | 0 | 1 | 1 |
| BAC_Uni_Vlv_01_12 | 1 | 1 | 0 | 0 |
| BAC_Uni_Vlv_01_13 | 1 | 1 | 0 | 1 |

**Interface**



### Block diagram version BAC_Uni_Vlv_01_13




### VAR_INPUT

```
rValPgm  : REAL;
bEnSfty  : BOOL;
bValSfty : BOOL;
bEnDst   : BOOL;
bValDst  : BOOL;
```

**rValPgm:** analog value program priority

**bEnSfty:** safety priority enable

**rValSfty:** analog value safety priority

**bEnDst:** disturbance priority enable

**rValDst:** analog value disturbance priority

**bDisAblk:** blocking protection function locked. Prevents simultaneous activation of the blocking protection of the pump and the control valve, e.g. at a heat exchanger

### VAR_OUTPUT

```
rPrVal   : REAL;
bSync    : BOOL;
```

**rPrVal :** Current position of the control valve.

**bSync:** Output of a pulse to synchronize the controller associated with the valve during reset from manual to automatic operation to the current valve position.
The synchronizing pulse **bSync** should only be used if the template used contains the valve position feedback **Fdb**.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| OpMod | FB_BACnetMVSetpoint [▶ 130] | | MV object for manual control of the control valve via the MCL or a local operator display |
| IdlePrd | FB_BACnetAVSetpoint [▶ 69] | X | AV object for input of the maximum valve standstill duration until a blocking protection pulse is issued. |
| ManPos | FB_BACnetAVSetpoint [▶ 69] | | AV object for entering the valve position with manual override |
| FdbMonitHys | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the hysteresis for the function monitoring of the control valve via the position feedback |
| Fdb | FB_BACnetAI1203 [▶ 49] | X | AI object for logging the position feedback from the valve. |
| FdbOut | FB_BACnetAI1203 [▶ 49] | X | AI object for logging the mechanical priority operation position feedback potentiometer |
| FdbMonitTiDly | FB_BACnetAVSetpoint [▶ 69] | X | AV object for entering the response delay of the function monitoring via position feedback. |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation feedback hand switch |
| AO | FB_BACnetAO1203 [▶ 53] | | AO object for controlling the control valve. |

| Priority: | Enable | Value | Comment |
|---|---|---|---|
| PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** | |
| PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** | |
| PRIO_LOCAL (8) | **OpMod_udiPrVal = OPMOD_MAN** | **ManPos_rPrVal** | In manual mode, value of AV object **ManPos** |

| Instance | Type | op-tional | Task | | | |
|---|---|---|---|---|---|---|
| | | | PRIO_PROGRAM (15) | TRUE | **AblkFnct_bQ** or **rValPgm** | Value of input **rValPgm** or 100% of blocking protection |
| | EQ | | TRUE if the active priority is **PRIO_PROGRAM** (15). Can be used for synchronizing the controller on return to automatic mode | | | |
| FdbMonitFnct | FB_BA_SldgLmtMonit [▶ 143] | X | Function monitoring of the control valve through comparison of the control output and the position feedback. | | | |
| AlmFdbMonit | FB_BA_Alarm [▶ 179] | X | Logging and further processing of an error from the position feedback monitoring | | | |
| FdbMonit | FB_BACnetBV1204 [▶ 94] | X | Reports position feedback error to the MCL | | | |
| AblkFnct | FB_BA_AntBlkg [▶ 215] | X | Generates a blocking protection pulse if in the last time interval of **IdlePrd** the valve has moved no more than 70% of its travel path. Pulse duration 180 seconds. | | | |
| TLogAO | FB_BACnetTLog1201 [▶ 135] | | Logs the present value of the AO object | | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| BI_Valve_LocalSwitch | BOOL | X | Input | Digital input - switch manual valve - message - manual/auto |
| AI_Valve_Feedback | INT | X | Input | Analog input - valve - measured value - position feedback |
| AI_Feeback_Position_Poti | INT | X | Input | Analog input - manual potentiometer - feedback - control value |
| AO_Valve_Position_Value | INT | | Output | Analog output - valve - control value position |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.78 BAC_Uni_Vlv3P_01_xx

### Functional description

The template **BAC_Uni_Vlv3P_01_xx** is used for controlling a three-point valve. It basically consists of two BO objects for the opening and closing and an MV object for manual control. The template is complemented through optional BACnet objects, see table with version overview.
The variables, which are linked with the process image of the input and output level in the PLC, can be found under **IO linking**.

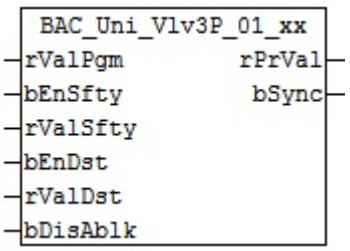
The two output variables rPrVal / bSync are only active, if the valve position feedback Fdb exists in the template that is used. If this is not the case, the two versions return the value zero. The interface of the templates BAC_Uni_Vlv3P_01_xx / BAC_Uni_Vlv_01_xx is the same. For this reason, in the Project Builder in a call template a three-point valve can be replaced with an analog valve, without having to adjust the PLC code.

### Versions

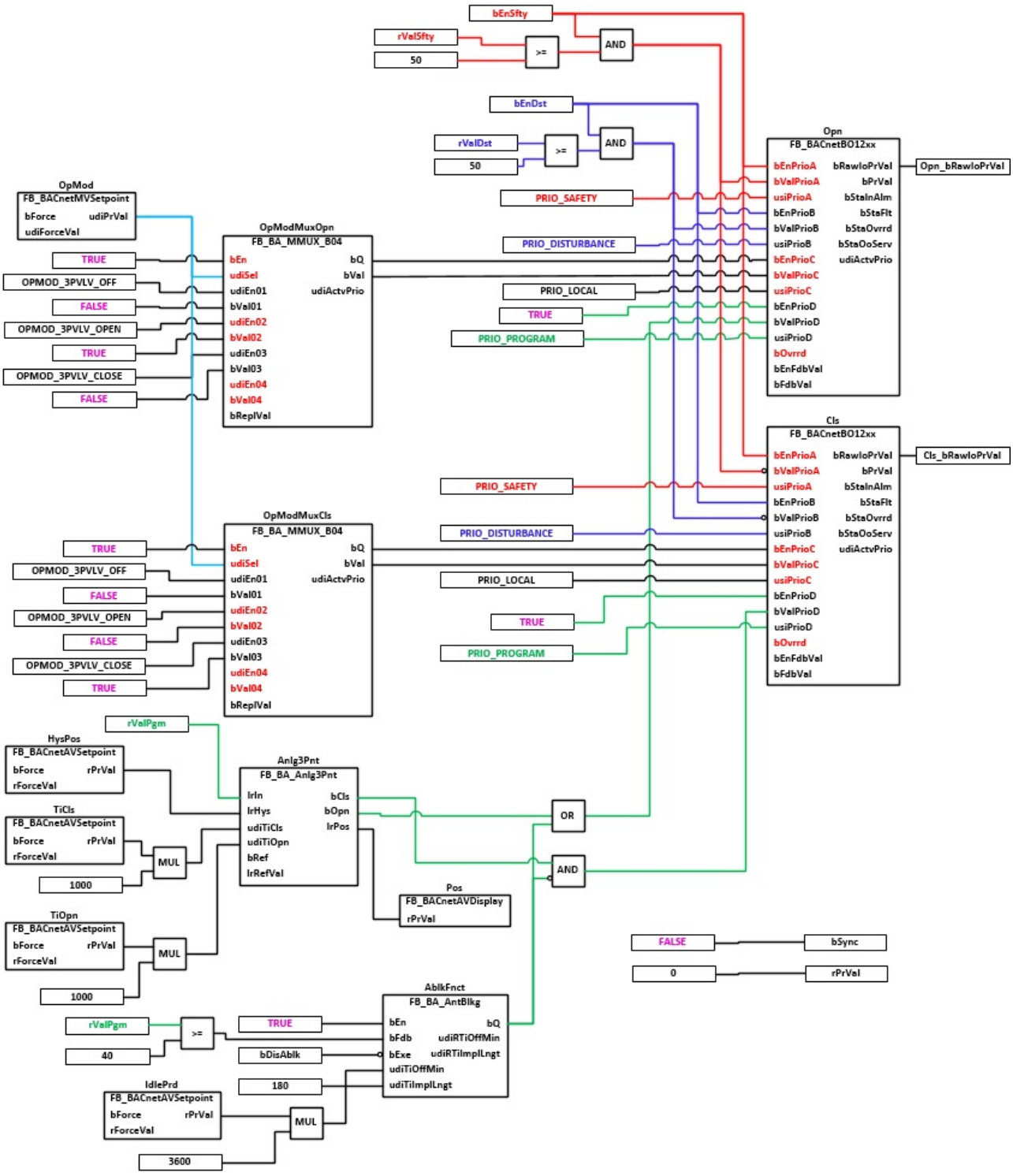
The template BAC_Uni_Vlv3P_xx is available in different versions.
The valve versions are identified by means of a key. The identification key is derived from the table below.

| Options | Feedback Valve position | Blocking protection | Mechanical priority operation position feedback closed | Mechanical priority operation position feedback open | mechanical priority operation position feedback hand switch | Final position Close | Final position Open |
|---|---|---|---|---|---|---|---|
| **Instance** | **Fdb** | **AblkFnct** | **FdbOutCls** | **FdbOutOpn** | **LocSwi** | **SwiCls** | **SwiOpn** |
| **Data point type** | **AI** | | **BI** | **BI** | **BI** | **BI** | **BI** |
| | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| BAC_Uni_Vlv3P_01_32 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

### Interface



### Block diagram version BAC_Uni_Vlv3P_01_32




### VAR_INPUT

```
bValPgm    : BOOL;
bEnSfty    : BOOL;
bValSfty   : BOOL;
bEnDst     : BOOL;
bValDst    : BOOL;
```

**rValPgm:** analog value program priority

**bEnSfty:** safety priority enable

**bEnDst:** disturbance priority enable

**bDisAblk:** blocking protection function locked. Prevents simultaneous activation of the blocking protection of the pump and the control valve, e.g. at a heat exchanger

### VAR_OUTPUT

```
rPrVal      : REAL;
bSync       : BOOL;
```

**rPrVal :** Current position of the control valve.

**bSync:** Output of a pulse to synchronize the controller associated with the valve during reset from manual to automatic operation to the current valve position.
The synchronising pulse **bSync** should only be used if the template used contains the valve position feedback **Fdb**.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| SwiOpn | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the open end position of the valve |
| SwiCls | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the closed end position of the valve |
| LocSwi | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation of the hand switch position feedback |
| FdbOutOpn | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation of the open position feedback |
| FdbOutCls | FB_BACnetBI1203 [▶ 72] | X | BI object for logging the mechanical priority operation of the closed position feedback |
| FdbVlv | FB_BACnetAI1203 [▶ 49] | X | AI object for logging the position feedback from the valve. |
| HysPos | FB_BACnetAVSet point [▶ 69] | | AV object for input of the hysteresis value for starting the change in position |
| IdlePrd | FB_BACnetAVSet point [▶ 69] | X | AV object for input of the blocking protection idle time |
| OpMod | FB_BACnetMVSet point [▶ 130] | | MV object for manual valve control via the MCL or a local operator display |

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| TiOpn | FB_BACnetMVSetpoint [▶ 130] | | AV object for entering the opening time value | | |
| TiCls | FB_BACnetMVSetpoint [▶ 130] | | AV object for entering the closing time value | | |
| OpModMuxOpn | FB_BA_MMUX_B04 [▶ 205] | | Multiplexer evaluation operation mode for opening | | |
| OpModMuxCls | FB_BA_MMUX_B04 [▶ 205] | | Multiplexer evaluation operation mode for closing | | |
| Anlg3Pnt | FB_BA_Anlg3Pnt [▶ 136] | | The function block **Anlg3Pnt** is the core of the templates and intended for controlling the three-point valve. It converts an analog positioning signal into the binary open/close commands | | |
| Pos | FB_BACnetAVDisplay [▶ 69] | | Current calculated actuator position | | |
| Opn | FB_BACnetBO1203 [▶ 81] | | BO object for the opening of the actuator | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | FALSE |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | FALSE |
| | | | PRIO_LOCAL (8) | **OpModMuxOpn** pools events which enable writing to the priority manual override (local) of the downstream BO object.<br>Events:<br>1. The MV object has the value **OPMOD_3PVLV_OFF** (manual off)<br>2. The MV object has the value **OPMOD_3PVLV_OPEN** (manual open)<br>3. The MV object has the value **OPMOD_3PVLV_CLOSE** (manual close) | **OpModMuxOpn_bVal** = TRUE, if **OpMod_udiPrVal** = **OPMOD_3PVLV_OPEN** |
| | | | PRIO_PROGRAM (15) | TRUE | **Anlg3Pnt_bOpn**OR**AblkFnct_bQ** |
| Cls | FB_BACnetBO1203 [▶ 81] | | BO object for the opening of the actuator | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | TRUE |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | TRUE |
| | | | PRIO_LOCAL (8) | **OpModMuxCls** pools events which enable writing to the priority manual override (local) of the downstream BO object.<br>Events: | **OpModMuxCls_bVal** = TRUE, if **OpMod_udiPrVal** = **OPMOD_3PVLV_CLOSE** |

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| | | | | 1. The MV object has the value **OPMOD_3PVLV_OFF** (manual off)<br>2. The MV object has the value **OPMOD_3PVLV_OPEN** (manual open)<br>3. The MV object has the value **OPMOD_3PVLV_CLOSE** (manual close) | |
| | | | PRIO_PROGRAM (15) | TRUE | **AnIg3Pnt_bCls**AND NOT**AblkFnct_bQ** |
| AblkFnct | FB_BA_AntBlkg [▶ 215] | X | Generates a blocking protection pulse if in the last time interval of IdlePrd the valve has moved no more than 40% of its travel path. Pulse duration 180 seconds. | | |
| TLogPos | FB_BACnetTLog1201 [▶ 135] | | Logs the present value of the Pos object | | |

### IO linking

In the XML description associated with the template, variables with the ID **Input** or **Output** are declared in the **Parameter** section. These parameters can be linked with the process image of the input and output level in the PLC in the Project Builder or via the Excel import interface.

| Parameter | Type | Instance | Type | Process image | |
|---|---|---|---|---|---|
| SwiOpn_bVal | BOOL | SwiOpn | FB_BACnetBI1203 [▶ 72] | Input | |
| SwiCls_bVal | BOOL | SwiCls | FB_BACnetBI1203 [▶ 72] | Input | |
| LocSwi_bVal | BOOL | LocSwi | FB_BACnetBI1203 [▶ 72] | Output | |
| FdbOutOpn_bVal | BOOL | FdbOutOpn | FB_BACnetBI1203 [▶ 72] | Input | |
| FdbOutCls_bVal | BOOL | FdbOutCls | FB_BACnetBI1203 [▶ 72] | Input | |
| FdbVlv_usiRawValSta | USINT | FdbVlv | FB_BACnetAI1203 [▶ 49] | Input | |
| FdbVlv_iRawVal | INT | FdbVlv | FB_BACnetAI1203 [▶ 49] | Input | |
| Opn_bRawIoPrVal | BOOL | Opn | FB_BACnetBO1203 [▶ 81] | Output | |
| Cls_bRawIoPrVal | BOOL | Cls | FB_BACnetBO1203 [▶ 81] | Output | |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.79 Ventilating system

## 9.79.1 BAC_AC_Identification_System_Plant_Key

**Identification system for HVAC plant templates**

**Ventilation systems with inlet and outlet air fan and thermal air treatment**

**Type: BAC_AC_SE**

| **Strategy** | | **Fan** | | **Preheater** | | **Cooler** | | **Energy recovery** | | **Mixed air** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inlet air control (simple) | 1 | 1-stage | 1 | No | 0 | No | 0 | No | 0 | No | 0 | |
| Inlet air control | 2 | 2-stage | 2 | Yes | 1 | Yes | 1 | plate | 1 | Yes | 1 | |
| Outlet air/ inlet air cascade (simple) | 3 | Speed | 3 | | | | | Recuperative | 2 | | | |
| Outlet air/ inlet air cascade | 4 | Pressure | 4 | | | | | Rotation | 3 | | | |
| | | Volume rate of flow | 5 | | | | | | | | | |

**Ventilation systems with inlet air fan and thermal air treatment**

**Type: BAC_AC_S**

| **Strategy** | | **Fan** | | **Preheater** | | **Cooler** | | **Energy recovery** | | **Mixed air** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inlet air control (simple) | 1 | 1-stage | 1 | No | 0 | No | 0 | No | 0 | No | 0 | |
| Inlet air control | 2 | 2-stage | 2 | Yes | 1 | Yes | 1 | Recuperative | 2 | Yes | 1 | |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Outlet air/ inlet air cascade (simple) | 3 | Speed | 3 | | | | | | | | | |
| Outlet air/ inlet air cascade | 4 | Pressure | 4 | | | | | | | | | |
| | | Volume rate of flow | 5 | | | | | | | | | |

**Ventilation systems with inlet and outlet air fan, thermal air treatment and humidification/ dehumidification**

**Type: BAC_AC_SEH**

| **Strategy** | | **Fan** | | **Preheater** | | **Cooler** | | **Energy recovery** | | **Mixed air** | | **Humidifier** | | **Reheater** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Supply air (single) | 1 | 1-stage | 1 | No | 0 | No | 0 | No | 0 | No | 0 | No | 0 | No | 0 | |
| Supply air | 2 | 2-stage | 2 | Yes | 1 | Yes | 1 | plate | 1 | Yes | 1 | Steam | 1 | Yes | 1 | |
| Cascade (single) | 3 | Speed | 3 | | | | | Recuperative | 2 | | | | | | | |
| Cascade | 4 | Pressure | 4 | | | | | Rotation | 3 | | | | | | | |
| | | Volume rate of flow | 5 | | | | | | | | | | | | | |

**Outlet air fan without air treatment**

**Type: BAC_AC_E**

| **Control** | | **Fan** | | |
|---|---|---|---|---|
| Request of: switching schedule, switch or external request | 1 | 1-stage | 1 | |
| | 2 | 2-stage | 2 | |
| | 3 | Speed | 3 | |
| | 4 | Pressure | 4 | |

## 9.79.2 BAC_AC_SE_3_4_1_1_1_0

### Application

Plant call template for a temperature-controlled ventilation system.

The main tasks of the template are:

- Plant control with the different operation modes and set values
- Temperature control for the selected sequence elements

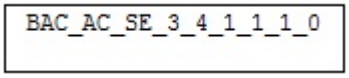
### Plant key

**Ventilation system with inlet and outlet air fan and thermal air treatment**

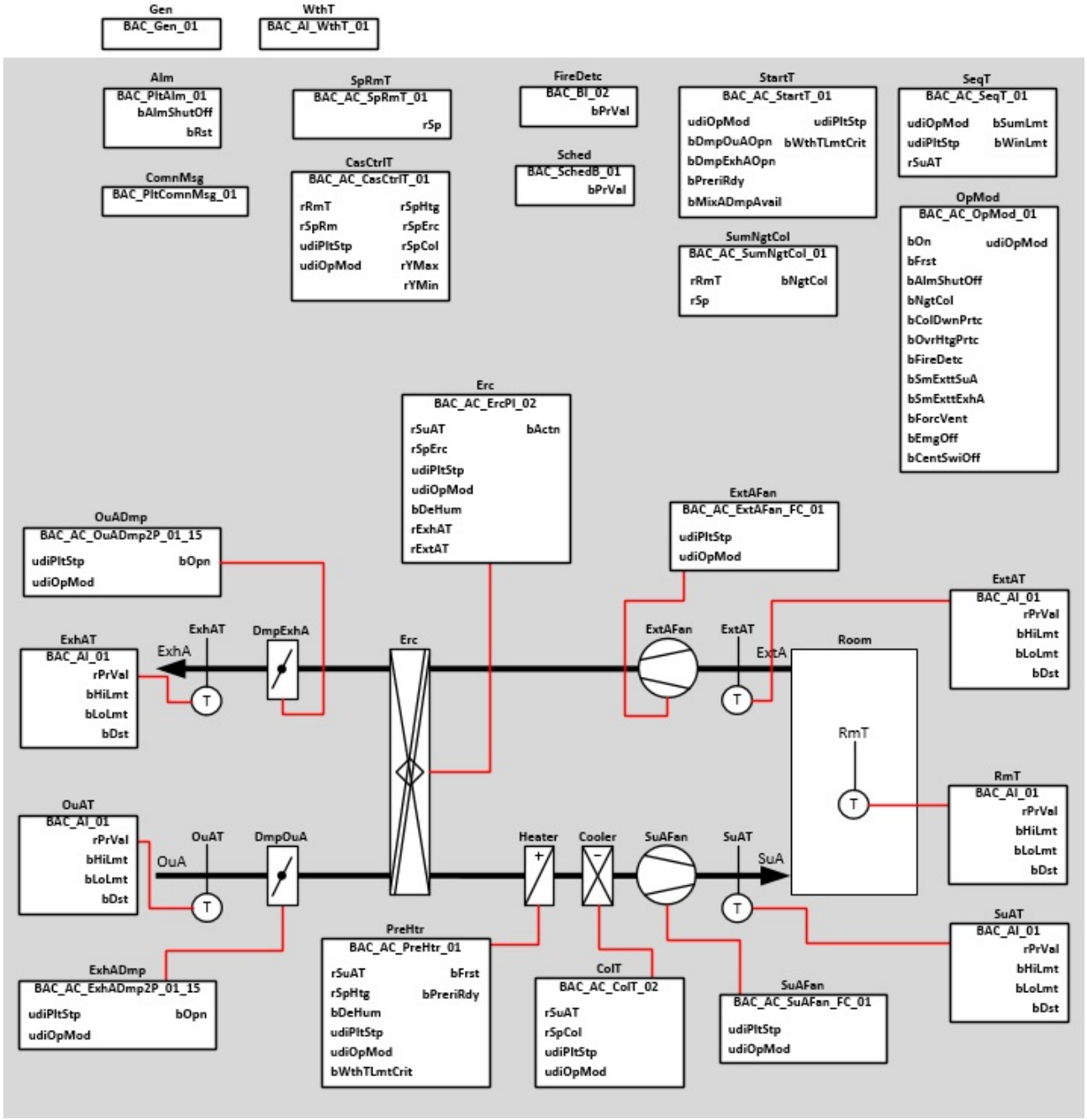
**Type: BAC_AC_SE**

| **Strategy** | | **Fan** | | **Preheater** | | **Cooler** | | **Energy recovery** | | **Mixed air** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inlet air control (simple) | 1 | 1-stage | 1 | No | 0 | No | 0 | No | 0 | No | 0 | |
| Inlet air control | 2 | 2-stage | 2 | Yes | 1 | Yes | 1 | plate | 1 | Yes | 1 | |
| Outlet air/ inlet air cascade (simple) | 3 | Speed | 3 | | | | | Recuperative | 2 | | | |
| Outlet air/ inlet air cascade | 4 | Pressure | 4 | | | | | Rotation | 3 | | | |
| | | Volume rate of flow | 5 | | | | | | | | | |
| | | | | | | | | | | | | |
| Outlet air/ inlet air cascade (simple) | 3 | Pressure | 4 | Preheater | 1 | Cooler | 1 | Energy recovery | 1 | Mixed air | 0 | |

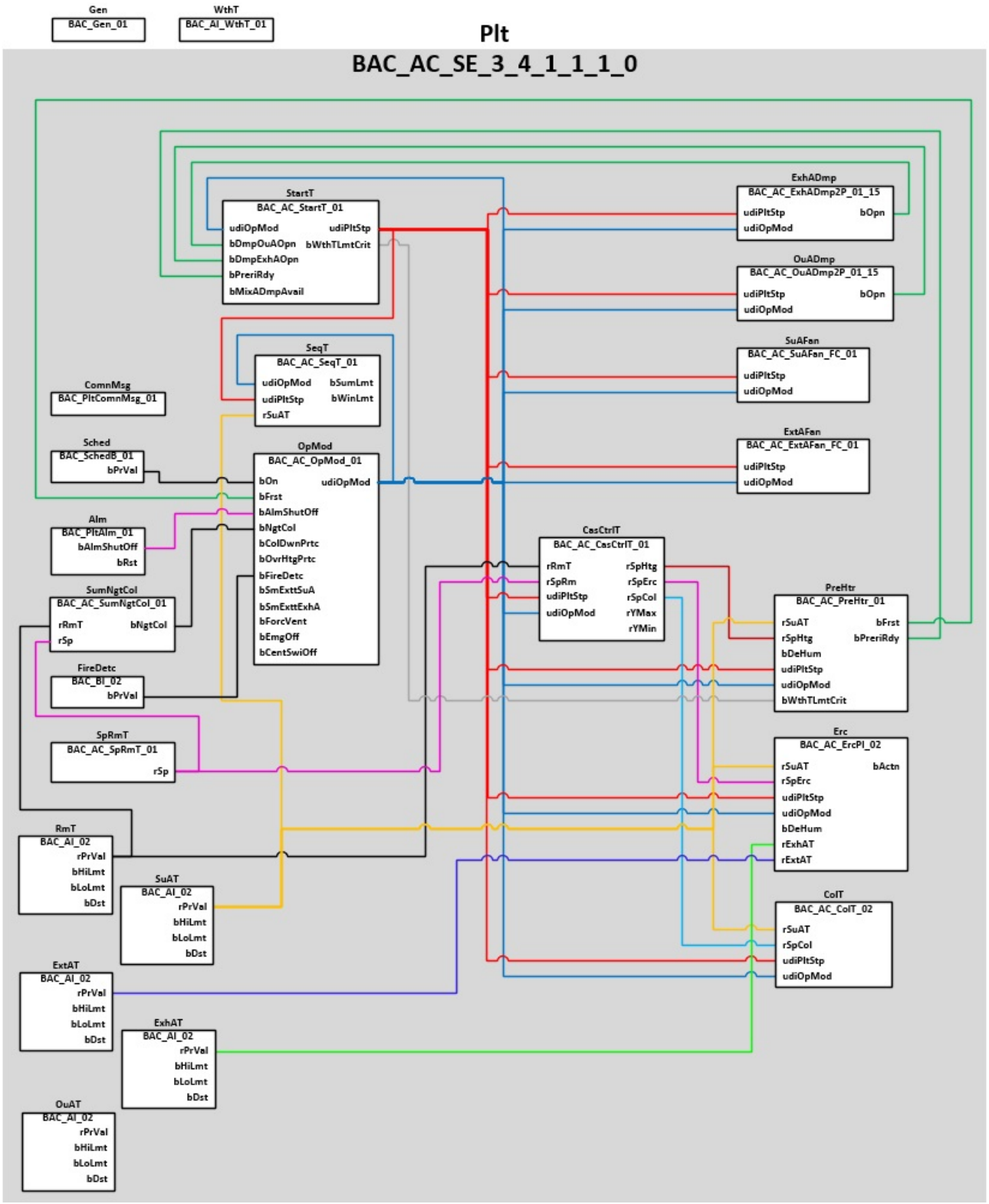
### Interface



## System diagram



### Block diagram




### Operation modes

The plant is switched on via a timer program or a manual system switch (software).

The plant can assume the operation modes listed in the table.

| **udiOpMod** | | |
|---|---|---|
| OPMOD_AC_OFF | 1 | Aus//Off |
| OPMOD_AC_ON | 2 | Ein//On |

| | | |
|---|---|---|
| OPMOD_AC_EMERG | 3 | Notfall//Emergency |
| OPMOD_AC_MANOFF | 4 | Hand aus//Manual off |
| OPMOD_AC_MANON | 5 | Hand ein//Manual on |
| OPMOD_AC_FRST | 6 | Frost//Frost |
| OPMOD_AC_SMEXTTPRG | 7 | Entrauchung Programm//Smoke extraction programm |
| OPMOD_AC_SMEXTTSUA | 8 | Entrauchung Zuluft//Smoke extraction supply |
| OPMOD_AC_SMEXTTEXHA | 9 | Entrauchung Fortluft//Smoke extraction exhaust |
| OPMOD_AC_FIRE | 10 | Feuer//Fire |
| OPMOD_AC_NGTCOL | 11 | Nachtkühlung//Night cooling |
| OPMOD_AC_COLDWNPRTC | 12 | Stützbetrieb,Auskühlschutz//Cool down protection |
| OPMOD_AC_OVRHTGPRTC | 13 | Überhitzungsschutz//Over heating protection |
| OPMOD_AC_ALM | 14 | Störung//Alarm |
| OPMOD_AC_FORCVENT | 15 | Zwangsbelüftung//Forced ventilation |
| OPMOD_AC_CENTSWIOFF | 16 | Zentralabschaltung//Central switch-off |

### Fire alarm

In case of a fire alarm, all switch-on conditions are overridden, and the system switches off. The smoke extraction functions are enabled. The two binary inputs *SmExttSuA* / *SmExttExhA* can be used for the 3 smoke extraction scenarios listed above.

### Fault shutdown

The plant switches off in the event of the following faults or status messages:

- Status message fire alarm (BAC_AC_StartT_01 [▶ 530])
- Status message frost alarm (BAC_FrstPrt_01 [▶ 507])
- Fan fault (BAC_AC_ExtAFan_FC_01 [▶ 422] / BAC_AC_SuAFan_FC_01 [▶ 427])
- Fault in the outside or exhaust air damper (BAC_AC_OuADmp2P_01_15 [▶ 445] / BAC_AC_ExhADmp2P_01_15 [▶ 438])

### Plant startup

If the plant is switched on in cold weather, control of the return temperature of the hot-water air heater is activated in the prerinsing plant step PLTSTP_AC_PRERI [▶ 357]. This procedure is intended to prevent the freezing of the hot-water air heater during the plant start-up.

### Temperature control

In this plant template a room air inlet air cascade is realized as control strategy.

In the templates BAC_AC_SpRmT_01 [▶ 648] / BAC_AC_CasCtrlT_01 [▶ 639] , the set values for the plant are generated and relayed to the sequence controllers.

### Program description

| Instance | Type | Task |
|---|---|---|
| Alm | BAC_PltAlm_01 [▶ 365] | Plant alarms |
| ComnMsg | BAC_PltComnMsg_01 [▶ 369] | Collective plant messages |
| ExhAT | BAC_AI_01 [▶ 675] | Call template exhaust air temperature |
| ExtAT | BAC_AI_01 [▶ 675] | Call template outlet air temperature |
| OuAT | BAC_AI_01 [▶ 675] | Call template outside air temperature |
| RmT | BAC_AI_01 [▶ 675] | Call template room temperature |

| Instance | Type | Task |
|---|---|---|
| SuAT | BAC_AI_01 [▶ 675] | Call template inlet air temperature |
| Sched | BAC_SchedB_01 [▶ 709] | Call template plant timer program |
| FireDetc | BAC_BI_01 [▶ 694] | BI object for logging the fire alarm event |
| SuAFan | BAC_AC_SuAFan_FC_01 [▶ 427] | Call template with integrated pressure regulator for a speed-controlled inlet air fan |
| ExtAFan | BAC_AC_ExtAFan_FC_01 [▶ 422] | Call template with integrated pressure regulator for a speed-controlled outlet air fan |
| ExhADmp | BAC_AC_ExhADmp2P_01_15 [▶ 438] | Call template exhaust air damper with spring return actuator and limit switch monitor |
| OuADmp | BAC_AC_OuADmp2P_01_15 [▶ 445] | Call template outside air damper with spring return actuator and limit switch monitor |
| SpRmT CasCtrlT | BAC_AC_SpRmT_01 [▶ 648] BAC_AC_CasCtrlT_01 [▶ 639] | Set value room air/inlet air cascade |
| SeqT | BAC_AC_SeqT_01 [▶ 524] | Call template sequence control |
| OpMod | BAC_AC_OpMod_01 [▶ 515] | Plant operation mode |
| Start | BAC_AC_StartT_01 [▶ 530] | Call template system start program for temperature-controlled ventilation system |
| SumNgtCol | BAC_AC_SumNgtCol_01 [▶ 540] | Night cooling |
| PreHtr | BAC_AC_PreHtr_01 [▶ 493] | Call template for a hot-water air heater |
| Erc | BAC_AC_ErcPl_01 [▶ 391] | Call template for the control of a plate heat exchanger for the recovery of energy in a ventilation system. |
| ColT | BAC_AC_ColT_02 | Call template of a temperature controlled cold-water air cooler (valve) |

### Plant alarm logging

The alarm priority ALM_PRIO_CRITICAL is a fault that results in plant shutdown.

| Template | Subtemplate | Instance | Type | Alarm priority | |
|---|---|---|---|---|---|
| ExhAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| ExtAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| OuAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| RmT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| SuAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |

| Template | Subtemplate | Instance | Type | Alarm priority | |
|---|---|---|---|---|---|
| SuAFan | SuAFanFU | AlmThOvrld | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmDstFC | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | SuAFanDiffPrssMonit | AlmDiffPrss | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| ExtAFan | ExtAFanFU | AlmThOvrld | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmDstFC | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | ExtAFanDiffPrssMonit | AlmDiffPrss | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| ExhADmp | | AlmFnct | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| OuADmp | | AlmFnct | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| PreHtr | PreHtrRetWtrT | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstThermostat | AlmBI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstPrt | AlmFrst | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstPrtT | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrPu | AlmDst | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrVlv | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |
| Erc | ErcDmp | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |
| ColT | ColTVlv | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1.0 | First release |

## 9.79.3 BAC_AC_SE_4_4_1_1_0_1

**Application**

Plant call template for a temperature-controlled ventilation system.

The main tasks of the template are:

- Plant control with the different operation modes and set values
- Temperature control for the selected sequence elements

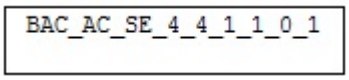
**Plant key**

**Ventilation system with inlet and outlet air fan and thermal air treatment**

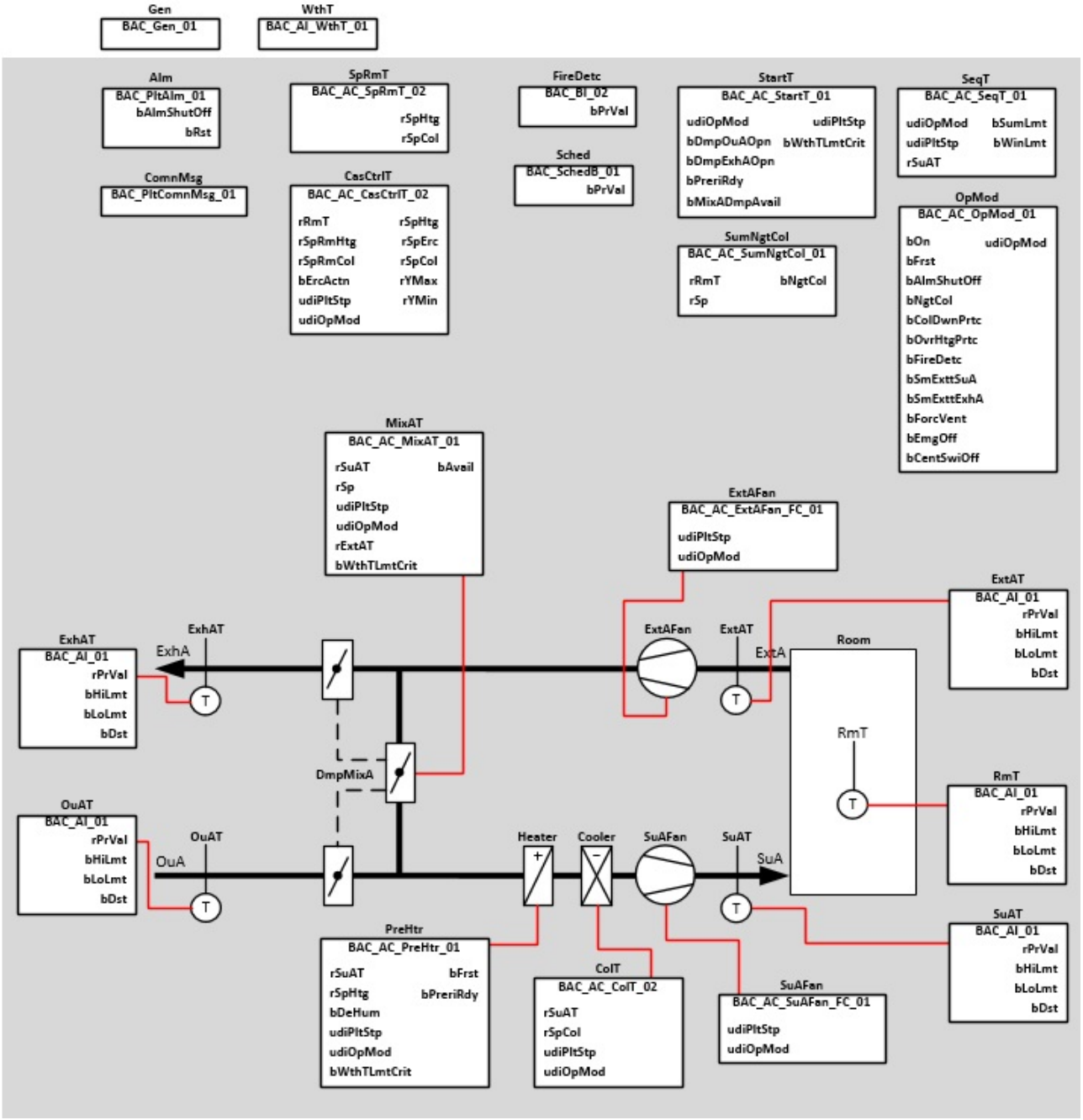
**Type: BAC_AC_SE**

| **Strategy** | | **Fan** | | **Preheater** | | **Cooler** | | **Energy recovery** | | **Mixed air** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inlet air control (simple) | 1 | 1-stage | 1 | No | 0 | No | 0 | No | 0 | No | 0 | |
| Inlet air control | 2 | 2-stage | 2 | Yes | 1 | Yes | 1 | plate | 1 | Yes | 1 | |
| Outlet air/ inlet air cascade (simple) | 3 | Speed | 3 | | | | | Recuperative | 2 | | | |
| Outlet air/ inlet air cascade | 4 | Pressure | 4 | | | | | Rotation | 3 | | | |
| | | Volume rate of flow | 5 | | | | | | | | | |
| | | | | | | | | | | | | |
| Outlet air/ inlet air cascade (simple) | 4 | Pressure | 4 | Preheater | 1 | Cooler | 1 | Energy recovery | 0 | Mixed air | 1 | |

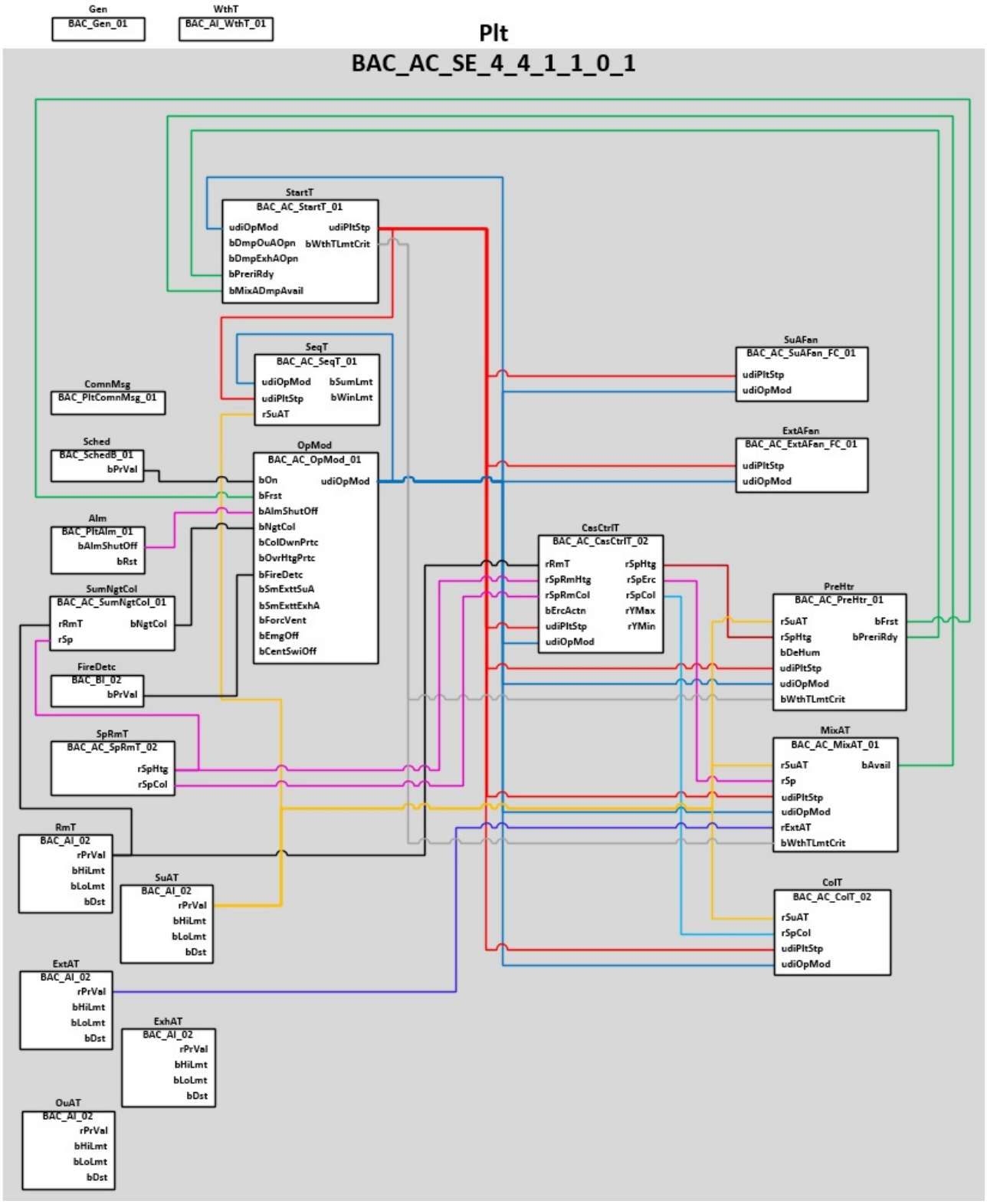
**Interface**



## System diagram



### Block diagram




### Operation modes

The plant is switched on via a timer program or a manual system switch (software).

The plant can assume the operation modes listed in the table.

| **udiOpMod** | | |
|---|---|---|
| OPMOD_AC_OFF | 1 | Aus//Off |
| OPMOD_AC_ON | 2 | Ein//On |

| | | |
|---|---|---|
| OPMOD_AC_EMERG | 3 | Notfall//Emergency |
| OPMOD_AC_MANOFF | 4 | Hand aus//Manual off |
| OPMOD_AC_MANON | 5 | Hand ein//Manual on |
| OPMOD_AC_FRST | 6 | Frost//Frost |
| OPMOD_AC_SMEXTTPRG | 7 | Entrauchung Programm//Smoke extraction programm |
| OPMOD_AC_SMEXTTSUA | 8 | Entrauchung Zuluft//Smoke extraction supply |
| OPMOD_AC_SMEXTTEXHA | 9 | Entrauchung Fortluft//Smoke extraction exhaust |
| OPMOD_AC_FIRE | 10 | Feuer//Fire |
| OPMOD_AC_NGTCOL | 11 | Nachtkühlung//Night cooling |
| OPMOD_AC_COLDWNPRTC | 12 | Stützbetrieb,Auskühlschutz//Cool down protection |
| OPMOD_AC_OVRHTGPRTC | 13 | Überhitzungsschutz//Over heating protection |
| OPMOD_AC_ALM | 14 | Störung//Alarm |
| OPMOD_AC_FORCVENT | 15 | Zwangsbelüftung//Forced ventilation |
| OPMOD_AC_CENTSWIOFF | 16 | Zentralabschaltung//Central switch-off |

### Fire alarm

In case of a fire alarm, all switch-on conditions are overridden, and the system switches off. The smoke extraction functions are enabled. The two binary inputs *SmExttSuA* / *SmExttExhA* can be used for the 3 smoke extraction scenarios listed above.

### Fault shutdown

The plant switches off in the event of the following faults or status messages:

- Status message fire alarm (BAC_AC_StartT_01 [▶ 530])
- Status message frost alarm (BAC_FrstPrt_01 [▶ 507])
- Fan fault (BAC_AC_ExtAFan_FC_01 [▶ 422] / BAC_AC_SuAFan_FC_01 [▶ 427])

### Plant startup

The plant start program **StartT** is informed by the variable **bAvail** of the mixed air system **MixAT** that the prerinsing of the preheater will be skipped.
A ramp function in the subtemplate **PID** (BAC_AC_MixAT_PID_01 [▶ 480]) of the mixed air system **MixAT** (BAC_AC_MixAT_01 [▶ 475]) replaces the prerinsing of the air heater during the start-up of the ventilation system.

### Temperature control

In this plant template a room air inlet air cascade is realized as control strategy.

In the templates BAC_AC_SpRmT_02 [▶ 651] / BAC_AC_CasCtrlT_02 [▶ 643] , the set values for the plant are generated and relayed to the sequence controllers.

### Program description

| Instance | Type | Task |
|---|---|---|
| Alm | BAC_PltAlm_01 [▶ 365] | Plant alarms |
| ComnMsg | BAC_PltComnMsg_01 [▶ 369] | Collective plant messages |
| ExhAT | BAC_AI_01 [▶ 675] | Call template exhaust air temperature |
| ExtAT | BAC_AI_01 [▶ 675] | Call template exhaust air temperature |
| OuAT | BAC_AI_01 [▶ 675] | Call template outside air temperature |
| RmT | BAC_AI_01 [▶ 675] | Call template room temperature |
| SuAT | BAC_AI_01 [▶ 675] | Call template supply air temperature |

| Instance | Type | Task |
|---|---|---|
| Sched | BAC_SchedB_01 [▶ 709] | Call template plant timer program |
| FireDetc | BAC_BI_01 [▶ 694] | BI object for logging the fire alarm event |
| SuAFan | BAC_AC_SuAFan_FC_01 [▶ 427] | Call template with integrated pressure regulator for a speed-controlled supply air fan |
| ExtAFan | BAC_AC_ExtAFan_FC_01 [▶ 422] | Call template with integrated pressure regulator for a speed-controlled exhaust air fan |
| SpRmT CasCtrlT | BAC_AC_SpRmT_02 [▶ 651] BAC_AC_CasCtrlT_02 [▶ 643] | Setpoint room air/supply air cascade |
| SeqT | BAC_AC_SeqT_01 [▶ 524] | Call template sequence control |
| OpMod | BAC_AC_OpMod_01 [▶ 515] | Plant operation mode |
| StartT | BAC_AC_StartT_01 [▶ 530] | Call template plant start program for temperature-controlled ventilation system |
| SumNgtCol | BAC_AC_SumNgtCol_01 [▶ 540] | Night cooling |
| PreHtr | BAC_AC_PreHtr_01 [▶ 493] | Call template for a hot water air heater |
| MixAT | BAC_AC_MixAT_01 [▶ 475] | Call template for the control and regulation of a mixed air system |
| ColT | BAC_AC_ColT_02 | Call template of a temperature controlled cold-water air cooler (valve) |

### Plant alarm logging

The alarm priority ALM_PRIO_CRITICAL is a fault that results in plant shutdown.

| Template | Subtemplate | Instance | Type | Alarm priority | |
|---|---|---|---|---|---|
| ExhAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| ExtAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| OuAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| RmT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| SuAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| SuAFan | SuAFanFU | AlmThOvrld | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmDstFC | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | SuAFanDiffPrssMonit | AlmDiffPrss | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |

| Template | Subtemplate | Instance | Type | Alarm priority | |
|---|---|---|---|---|---|
| ExtAFan | ExtAFanFU | AlmThOvrld | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmDstFC | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | ExtAFanDiffPrssMonit | AlmDiffPrss | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| PreHtr | PreHtrRetWtrT | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstThermostat | AlmBI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstPrt | AlmFrst | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstPrtT | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrPu | AlmDst | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrVlv | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |
| MixAT | DmpMixA | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |
| | DmpOuA | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |
| | DmpExhA | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |
| ColT | ColTVlv | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1.0 | First release |

## 9.79.4 BAC_AC_SE_4_4_1_1_3_0

**Application**

Plant call template for a temperature-controlled ventilation system.

The main tasks of the template are:

- Plant control with the different operation modes and set values
- Temperature control for the selected sequence elements

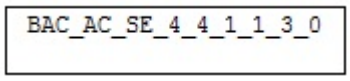
**Plant key**

**Ventilation system with inlet and outlet air fan and thermal air treatment**

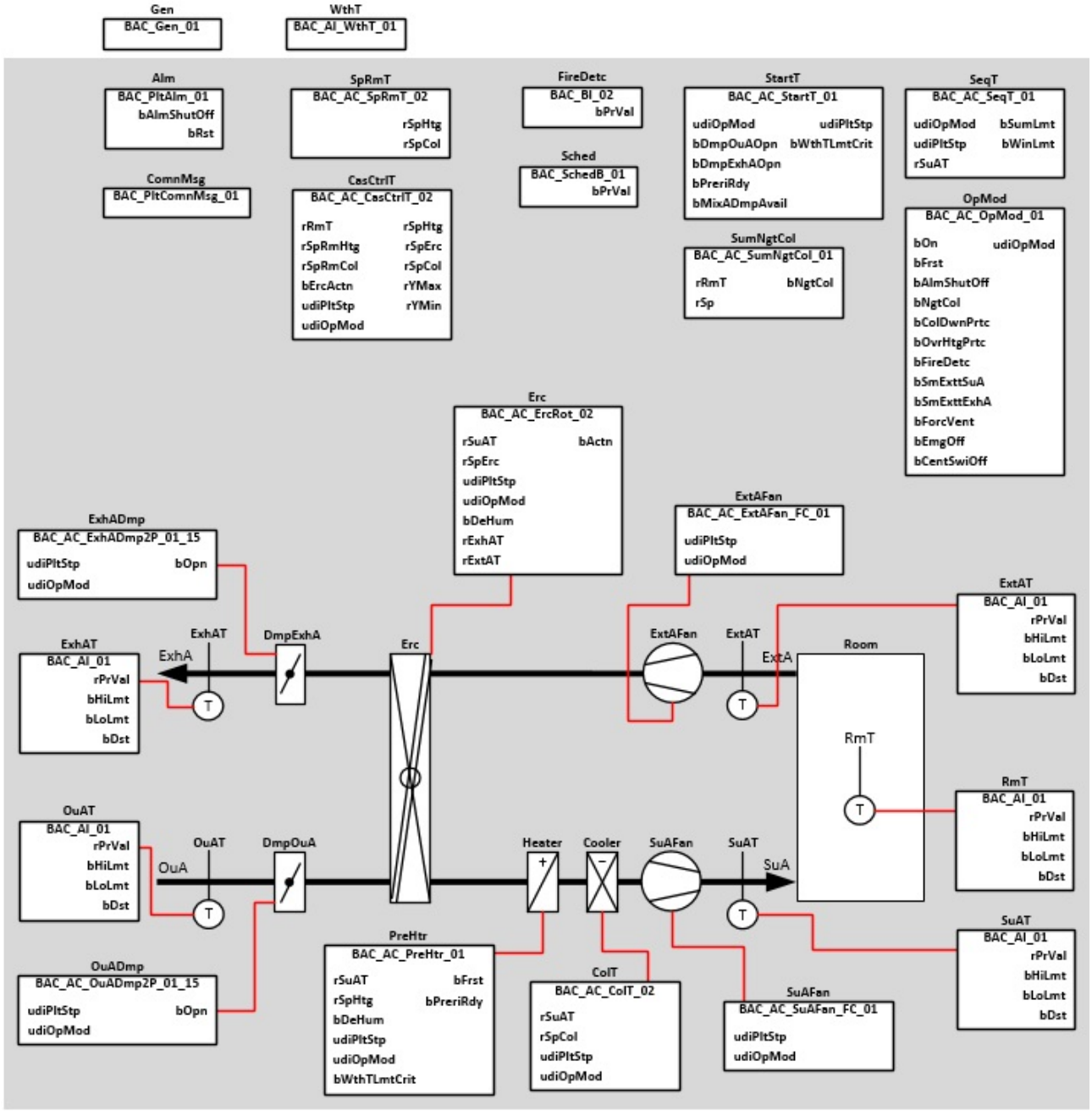
**Type: BAC_AC_SE**

| **Strategy** | | **Fan** | | **Preheater** | | **Cooler** | | **Energy recovery** | | **Mixed air** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inlet air control (simple) | 1 | 1-stage | 1 | No | 0 | No | 0 | No | 0 | No | 0 | |
| Inlet air control | 2 | 2-stage | 2 | Yes | 1 | Yes | 1 | plate | 1 | Yes | 1 | |
| Outlet air/ inlet air cascade (simple) | 3 | Speed | 3 | | | | | Recuperative | 2 | | | |
| Outlet air/ inlet air cascade | 4 | Pressure | 4 | | | | | Rotation | 3 | | | |
| | | Volume rate of flow | 5 | | | | | | | | | |
| | | | | | | | | | | | | |
| Outlet air/ inlet air cascade (simple) | 4 | Pressure | 4 | Preheater | 1 | Cooler | 1 | Energy recovery | 3 | Mixed air | 0 | |

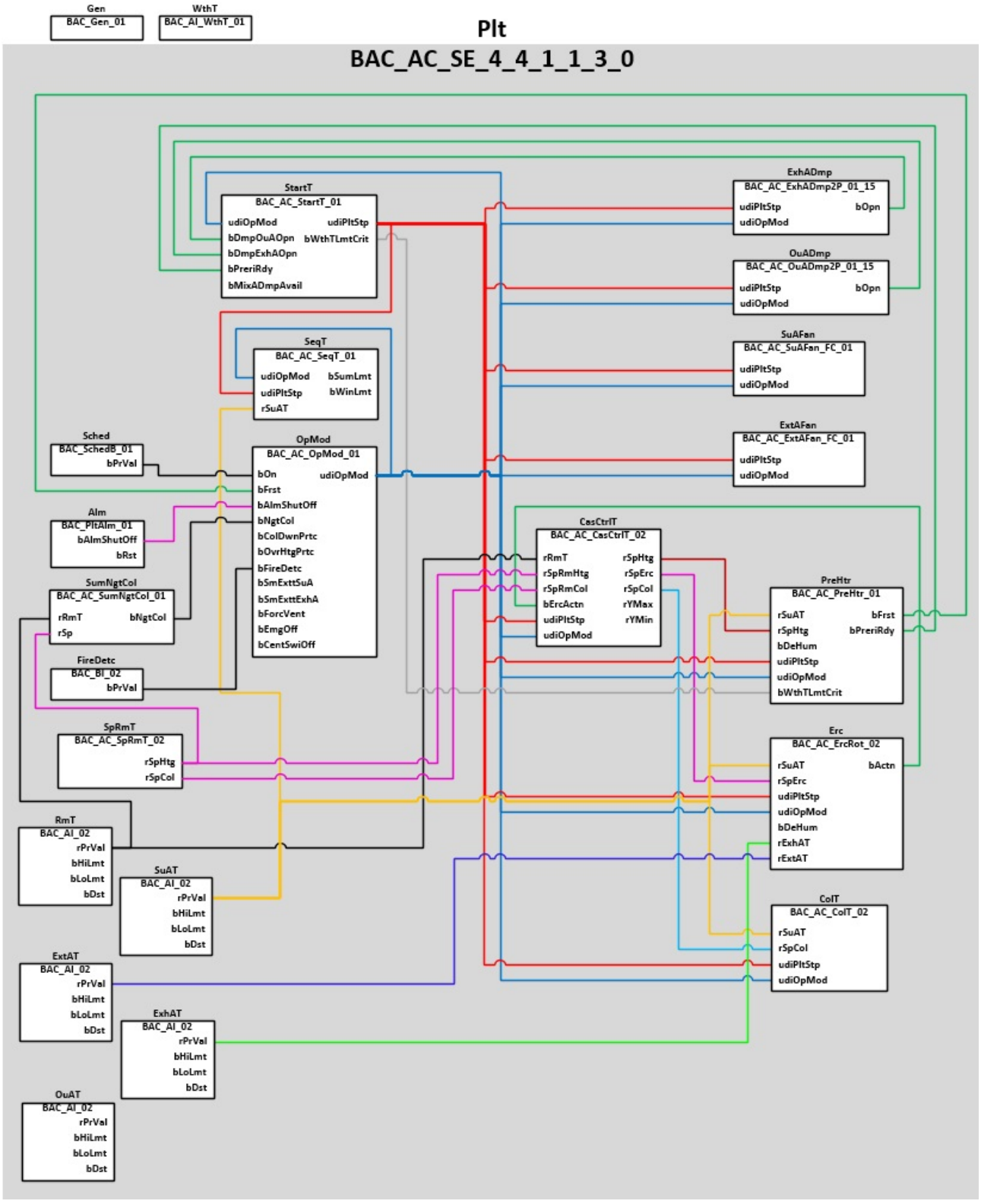
**Interface**



### System diagram



### Block diagram




### Operation modes

The plant is switched on via a timer program or a manual system switch (software).

The plant can assume the operation modes listed in the table.

| **udiOpMod** | | |
|---|---|---|
| OPMOD_AC_OFF | 1 | Aus//Off |
| OPMOD_AC_ON | 2 | Ein//On |

| | | |
|---|---|---|
| OPMOD_AC_EMERG | 3 | Notfall//Emergency |
| OPMOD_AC_MANOFF | 4 | Hand aus//Manual off |
| OPMOD_AC_MANON | 5 | Hand ein//Manual on |
| OPMOD_AC_FRST | 6 | Frost//Frost |
| OPMOD_AC_SMEXTTPRG | 7 | Entrauchung Programm//Smoke extraction programm |
| OPMOD_AC_SMEXTTSUA | 8 | Entrauchung Zuluft//Smoke extraction supply |
| OPMOD_AC_SMEXTTEXHA | 9 | Entrauchung Fortluft//Smoke extraction exhaust |
| OPMOD_AC_FIRE | 10 | Feuer//Fire |
| OPMOD_AC_NGTCOL | 11 | Nachtkühlung//Night cooling |
| OPMOD_AC_COLDWNPRTC | 12 | Stützbetrieb,Auskühlschutz//Cool down protection |
| OPMOD_AC_OVRHTGPRTC | 13 | Überhitzungsschutz//Over heating protection |
| OPMOD_AC_ALM | 14 | Störung//Alarm |
| OPMOD_AC_FORCVENT | 15 | Zwangsbelüftung//Forced ventilation |
| OPMOD_AC_CENTSWIOFF | 16 | Zentralabschaltung//Central switch-off |

### Fire alarm

In case of a fire alarm, all switch-on conditions are overridden, and the system switches off. The smoke extraction functions are enabled. The two binary inputs *SmExttSuA* / *SmExttExhA* can be used for the 3 smoke extraction scenarios listed above.

### Fault shutdown

The plant switches off in the event of the following faults or status messages:

- Status message fire alarm (BAC_AC_StartT_01 [▶ 530])
- Status message frost alarm (BAC_FrstPrt_01 [▶ 507])
- Fan fault (BAC_AC_ExtAFan_FC_01 [▶ 422] / BAC_AC_SuAFan_FC_01 [▶ 427])
- Fault in the outside or exhaust air damper (BAC_AC_OuADmp2P_01_15 [▶ 445] / BAC_AC_ExhADmp2P_01_15 [▶ 438])
- Error of the rotary heat exchanger (BAC_AC_ErcRot_02 [▶ 409])

### Plant startup

If the plant is switched on in cold weather, control of the return temperature of the hot-water air heater is activated in the prerinsing plant step PLTSTP_AC_PRERI [▶ 357]. This procedure is intended to prevent the freezing of the hot-water air heater during the plant start-up.

### Temperature control

In this plant template a room air inlet air cascade is realized as control strategy.

In the templates BAC_AC_SpRmT_02 [▶ 651] / BAC_AC_CasCtrlT_02 [▶ 643] , the set values for the plant are generated and relayed to the sequence controllers.

### Program description

| Instance | Type | Task |
|---|---|---|
| Alm | BAC_PltAlm_01 [▶ 365] | Plant alarms |
| ComnMsg | BAC_PltComnMsg_01 [▶ 369] | Collective plant messages |
| ExhAT | BAC_AI_01 [▶ 675] | Call template exhaust air temperature |
| ExtAT | BAC_AI_01 [▶ 675] | Call template outlet air temperature |
| OuAT | BAC_AI_01 [▶ 675] | Call template outside air temperature |

| Instance | Type | Task |
|---|---|---|
| RmT | BAC_AI_01 [▶ 675] | Call template room temperature |
| SuAT | BAC_AI_01 [▶ 675] | Call template inlet air temperature |
| Sched | BAC_SchedB_01 [▶ 709] | Call template plant timer program |
| FireDetc | BAC_BI_01 [▶ 694] | BI object for logging the fire alarm event |
| SuAFan | BAC_AC_SuAFan_FC_01 [▶ 427] | Call template with integrated pressure regulator for a speed-controlled inlet air fan |
| ExtAFan | BAC_AC_ExtAFan_FC_01 [▶ 422] | Call template with integrated pressure regulator for a speed-controlled outlet air fan |
| ExhADmp | BAC_AC_ExhADmp2P_01_15 [▶ 438] | Call template exhaust air damper with spring return actuator and limit switch monitor |
| OuADmp | BAC_AC_OuADmp2P_01_15 [▶ 445] | Call template outside air damper with spring return actuator and limit switch monitor |
| SpRmT CasCtrlT | BAC_AC_SpRmT_02 [▶ 651] BAC_AC_CasCtrlT_02 [▶ 643] | Set value room air/inlet air cascade |
| SeqT | BAC_AC_SeqT_01 [▶ 524] | Call template sequence control |
| OpMod | BAC_AC_OpMod_01 [▶ 515] | Plant operation mode |
| Start | BAC_AC_StartT_01 [▶ 530] | Call template system start program for temperature-controlled ventilation system |
| SumNgtCol | BAC_AC_SumNgtCol_01 [▶ 540] | Night cooling |
| PreHtr | BAC_AC_PreHtr_01 [▶ 493] | Call template for a hot-water air heater |
| Erc | BAC_AC_ErcRot_02 [▶ 409] | Call template for controlling a rotary heat exchanger for energy recovery in a ventilation system. |
| ColT | BAC_AC_ColT_02 | Call template of a temperature controlled cold-water air cooler (valve) |

### Plant alarm logging

The alarm priority ALM_PRIO_CRITICAL is a fault that results in plant shutdown.

| Template | Subtemplate | Instance | Type | Alarm priority | |
|---|---|---|---|---|---|
| ExhAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| ExtAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| OuAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| RmT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| SuAT | | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |

| Template | Subtemplate | Instance | Type | Alarm priority | |
|---|---|---|---|---|---|
| SuAFan | SuAFanFU | AlmThOvrld | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmDstFC | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | SuAFanDiffPrssMonit | AlmDiffPrss | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| ExtAFan | ExtAFanFU | AlmThOvrld | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmDstFC | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | ExtAFanDiffPrssMonit | AlmDiffPrss | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| ExhADmp | | AlmFnct | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| OuADmp | | AlmFnct | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| PreHtr | PreHtrRetWtrT | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstThermostat | AlmBI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstPrt | AlmFrst | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrFrstPrtT | AlmAI | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrPu | AlmDst | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| | PreHtrVlv | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |
| Erc | ErcMot | AlmDstFC | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmThOvrld | FB_BA_Alarm [▶ 179] | ALM_PRIO_CRITICAL | |
| | | AlmMntnSwi | FB_BA_Alarm [▶ 179] | ALM_PRIO_ALARM | |
| ColT | ColTVlv | AlmFdbMonit | FB_BA_Alarm [▶ 179] | ALM_PRIO_WARNING | |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1.0 | First release |

## 9.79.5 BAC_AC_Sx_001

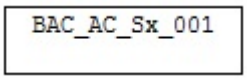
### Application

The call template is a basic system program for a ventilation system.
It contains basic elements for the following types of ventilation system:

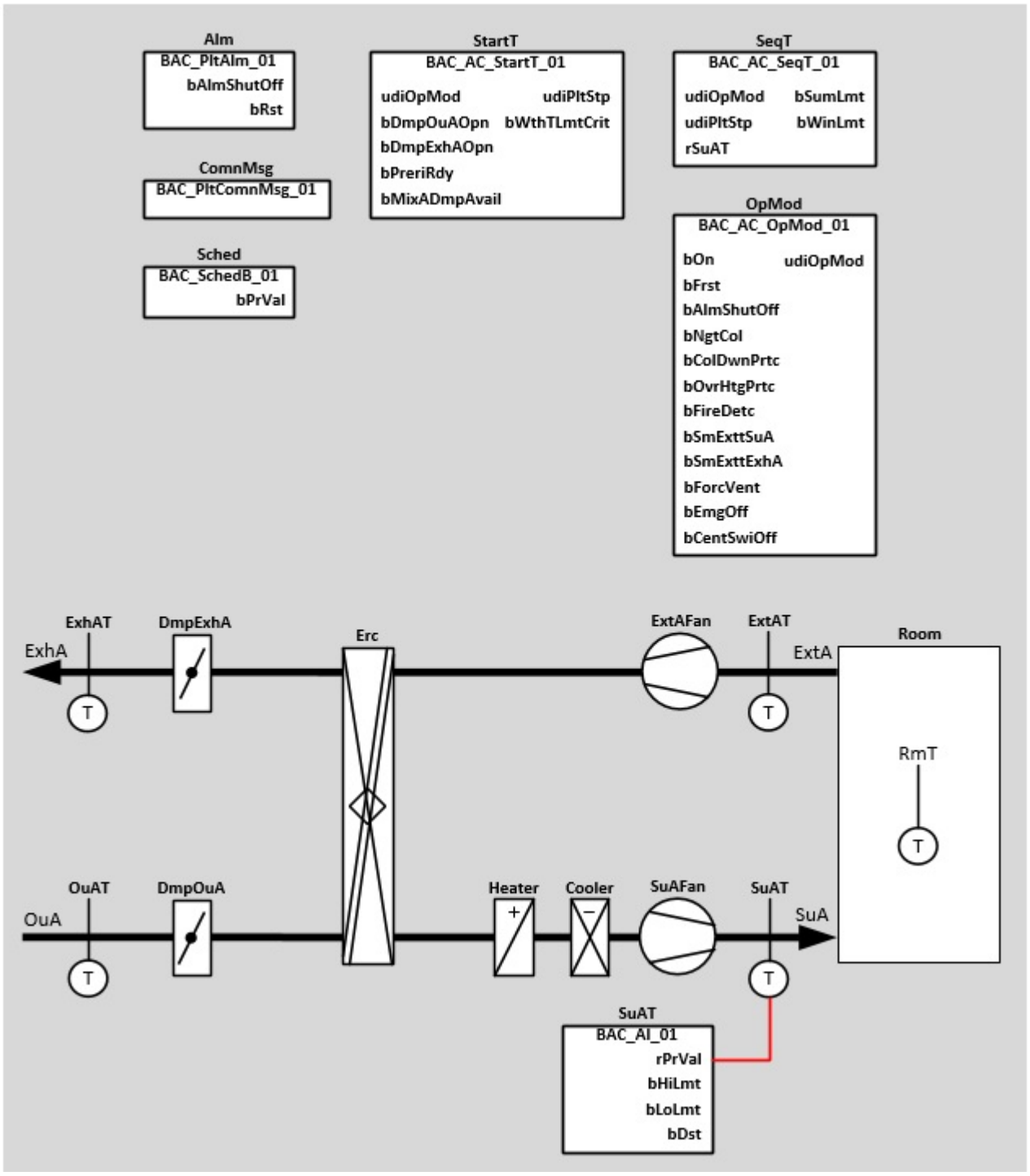
- **Ventilation systems with inlet and outlet air fan and thermal air treatment**
- **Ventilation systems with inlet air fan and thermal air treatment**

The missing units of a ventilation system must be added to the basic system program, called and linked; see the system sample BAC_AC_SE_3_4_1_1_1_0 [▶ 610].

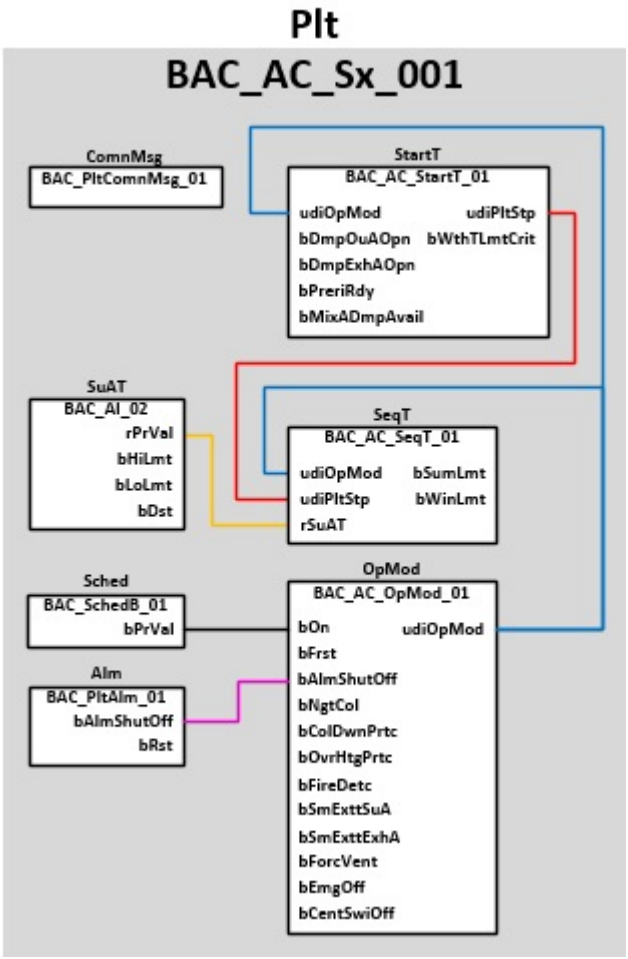
### Interface



### System diagram



### Block diagram




### Operation modes

The plant is switched on via a timer program or a manual system switch (software).

The plant can assume the operation modes listed in the table.

| **udiOpMod** | | |
|---|---|---|
| OPMOD_AC_OFF | 1 | Aus//Off |
| OPMOD_AC_ON | 2 | Ein//On |
| OPMOD_AC_EMERG | 3 | Notfall//Emergency |
| OPMOD_AC_MANOFF | 4 | Hand aus//Manual off |
| OPMOD_AC_MANON | 5 | Hand ein//Manual on |
| OPMOD_AC_FRST | 6 | Frost//Frost |
| OPMOD_AC_SMEXTTPRG | 7 | Entrauchung Programm//Smoke extraction programm |
| OPMOD_AC_SMEXTTSUA | 8 | Entrauchung Zuluft//Smoke extraction supply |
| OPMOD_AC_SMEXTTEXHA | 9 | Entrauchung Fortluft//Smoke extraction exhaust |
| OPMOD_AC_FIRE | 10 | Feuer//Fire |
| OPMOD_AC_NGTCOL | 11 | Nachtkühlung//Night cooling |
| OPMOD_AC_COLDWNPRTC | 12 | Stützbetrieb,Auskühlschutz//Cool down protection |
| OPMOD_AC_OVRHTGPRTC | 13 | Überhitzungsschutz//Over heating protection |
| OPMOD_AC_ALM | 14 | Störung//Alarm |
| OPMOD_AC_FORCVENT | 15 | Zwangsbelüftung//Forced ventilation |
| OPMOD_AC_CENTSWIOFF | 16 | Zentralabschaltung//Central switch-off |

**Program description**

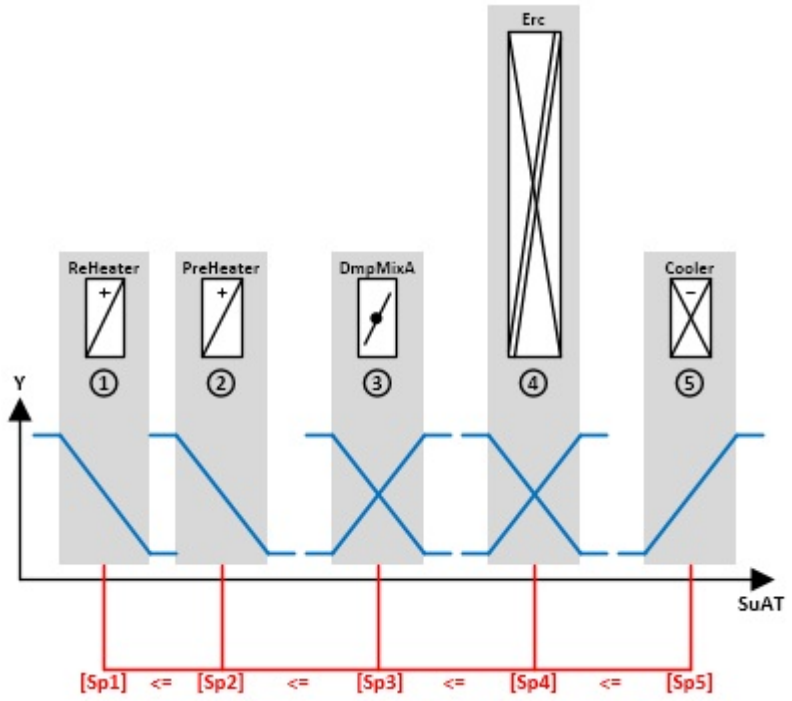
| Instance | Type | Task |
|---|---|---|
| Alm | BAC_PltAlm_01 [▶ 365] | Plant alarms |
| ComnMsg | BAC_PltComnMsg_01 [▶ 369] | Collective plant messages |
| SuAT | BAC_AI_01 [▶ 675] | Call template supply air temperature |
| Sched | BAC_SchedB_01 [▶ 709] | Call template plant timer program |
| SeqT | BAC_AC_SeqT_01 [▶ 524] | Call template sequence control |
| OpMod | BAC_AC_OpMod_01 [▶ 515] | Plant operation mode |
| Start | BAC_AC_StartT_01 [▶ 530] | Call template plant start program for temperature-controlled ventilation system |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1.0 | First release |

# 9.80 Set values of the sequence controllers

In the sequence controllers the setpoints must be monotonically increasing: [Sp1] <= [Sp2] <= [Sp3] <= [Sp4] <= [Sp5] <=...<= [Spn]



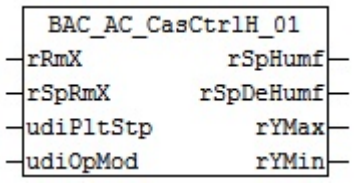

| | | |
|---|---|---|
| [Sp1] | rPrSpHtg | Heating setpoint |
| [Sp2] | rPrSpHtg | Heating setpoint |
| [Sp3] | rPrSpErc | Setpoint for mixed air dampers |
| [Sp4] | rPrSpErc | Setpoint for energy recovery |
| [Sp5] | rPrSpCol | Cooling setpoint |

# 9.80.1 BAC_AC_CasCtrlH_01

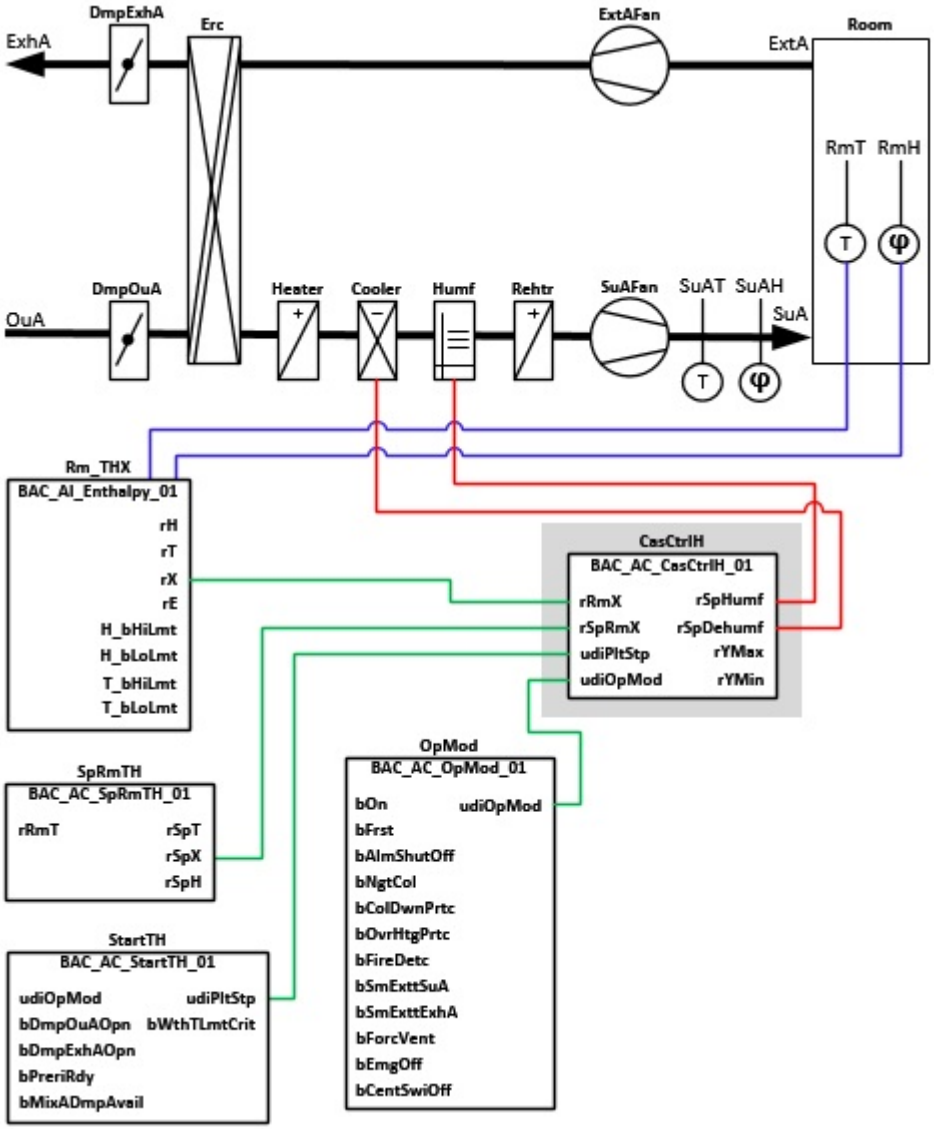
### Application

The template is a cascade controller for the supply air humidity, consisting of a master controller for calculating the set values for humidification and dehumidification.

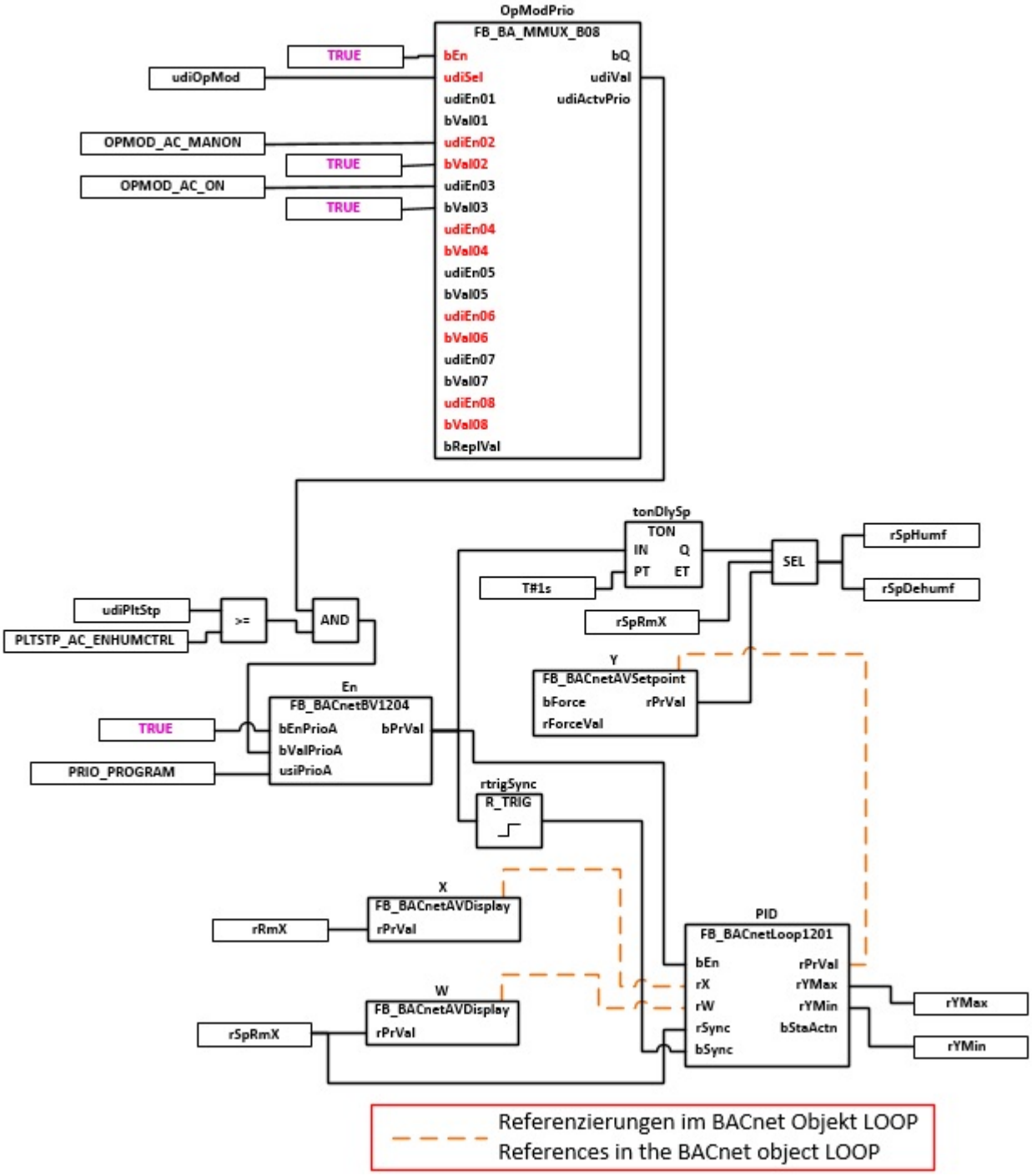
### Interface




### Plant diagram 01



### Block diagram




### VAR_INPUT

```
rRmX          : REAL;
rSpRmX        : REAL;
udiPltStp     : UDINT;
udiOpMod      : UDINT;
```

**rRmX:** Actual value of absolute room humidity in g/kg

**rSpRmX:** Room set value (dehumidify) in g/kg

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_StartTH_01 [▶ 535]

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rSpHumf      : REAL;
rSpDeHumf    : REAL;
rYMax        : REAL;
rYMin        : REAL;
```

**rSpHumf:** Supply air set value humidification g/kg

**rSpDehumf:** Supply air set value dehumidification g/kg

**rYMax:** Upper value of the controller output limitation

**rYMin:** Lower value of the controller output limitation

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task | | |
|---|---|---|---|---|
| X | FB_BACnetAVDisplay [▶ 69] | The input variable **rRmX** is connected to the AV object.<br>It is referenced to the actual value input of the BACnet loop object **PID**. | | |
| W | FB_BACnetAVDisplay [▶ 69] | The input variable **rSpRmX** is connected to the AV object.<br>It is referenced to the setpoint input of the BACnet loop object **PID**. | | |
| OpModPrio | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the enable conditions of the PID master controller depending on the plant operation mode **udiOpMod**. | | |
| | | **udiOpMod** | | **Enable cascade control** |
| | | OPMOD_AC_MANON | Manual on | TRUE |
| | | OPMOD_AC_ON | On | TRUE |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the controller enable in the MCL or in a local operator display | | |
| | >=, AND | The result of this network is the PID master controller enable. The enable depends on the plant operation mode **udiOpMod** and the plant steps **udiPltStp** during startup of the ventilation system. | | |
| PID | FB_BACnetLoop1201 [▶ 98] | Master controller for a room temperature control via room supply air cascade. Supplies a simple supply air temperature setpoint. | | |
| rtrigSync | R_TRIG | Upon receipt of the controller enable **En**, **rtrigSync** triggers synchronization of the master controller **PID** with its input value **lrSync**. | | |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object PID. It indicates the supply air humidity setpoint. | | |

| Instance | Type | Task |
|---|---|---|
| TLogY | FB_BACnetTLog1201 [▶ 135] | Trend logging of the supply air humidity setpoint **Y** |
| tonDlySp | TON<br>SEL | Upon receipt of the controller enable **En**, this network outputs the supply air humidity setpoint with a delay. |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.2 BAC_AC_CasCtrlH_02

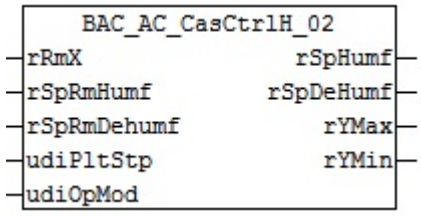
**Functional description**

The template is the master controller for a room or return air/supply air cascade humidity control.

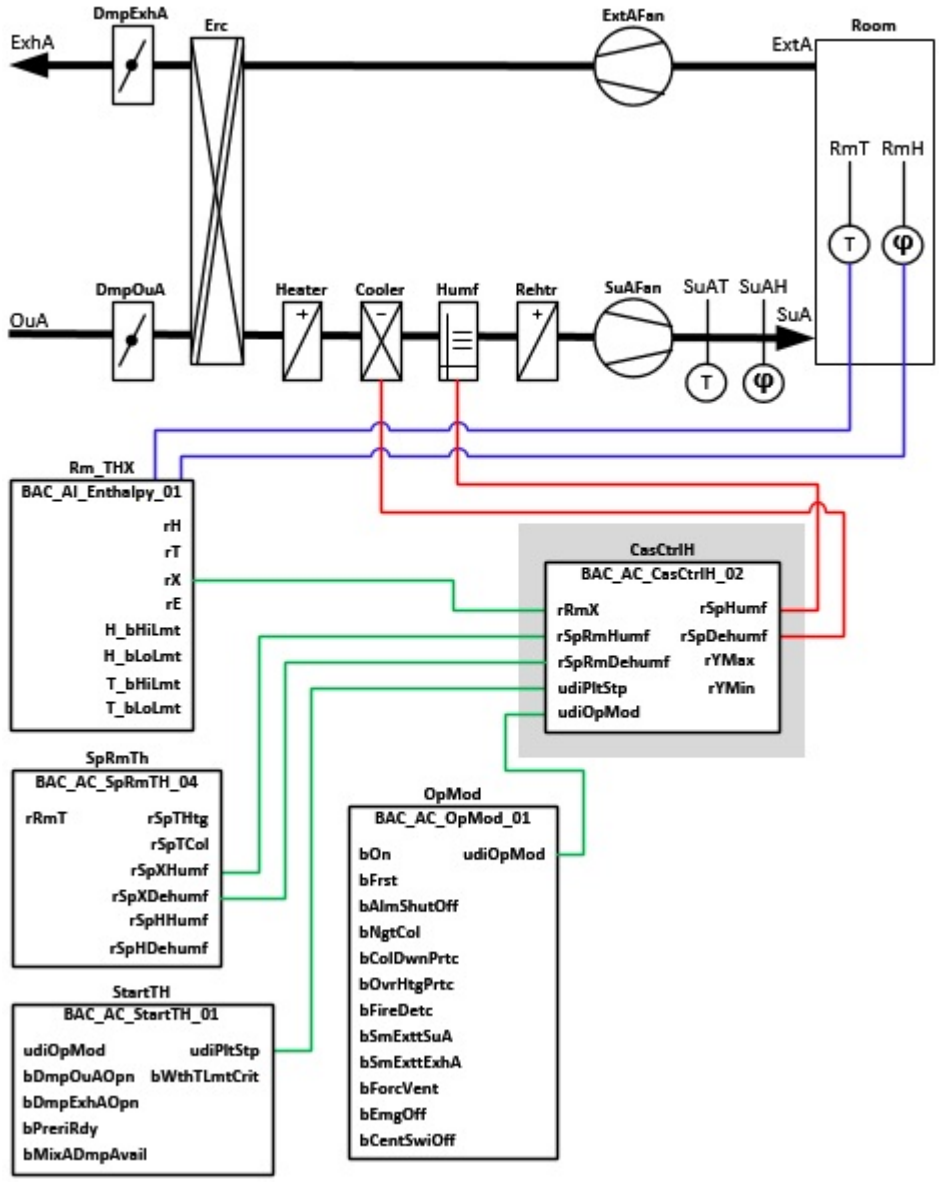
The purpose of the master control is to maintain the room humidity within the comfort zone between an upper and lower set value.
The BACnet loop object **PID_Humf** is used to control the lower room humidity limit. It calculates the supply air humidity set value for humidification.
The BACnet loop object **PID_Dehumf** is used to control the upper room humidity limit. It calculates the supply air humidity set value for dehumidification.
Both controllers receive the room humidity set values from the upstream set value template.
The set values for the room humidity and the set values for the supply air humidity calculated by the loop objects are specified in g/kg, i.e. as absolute humidity.

The parameters YMin, YMax, gain and integral time for the two loop objects are continuously synchronised by the **PID_Sync**. Due to the difference between lower and upper set value, the characteristic output curves of the two PI controllers always show a parallel offset. An overlap of the supply air set values for humidification and dehumidification is thus avoided.
It is important to ensure that **rSpRmHumf** is always less than or equal to **rSpRmDehumf**. This is done in the upstream set value template.

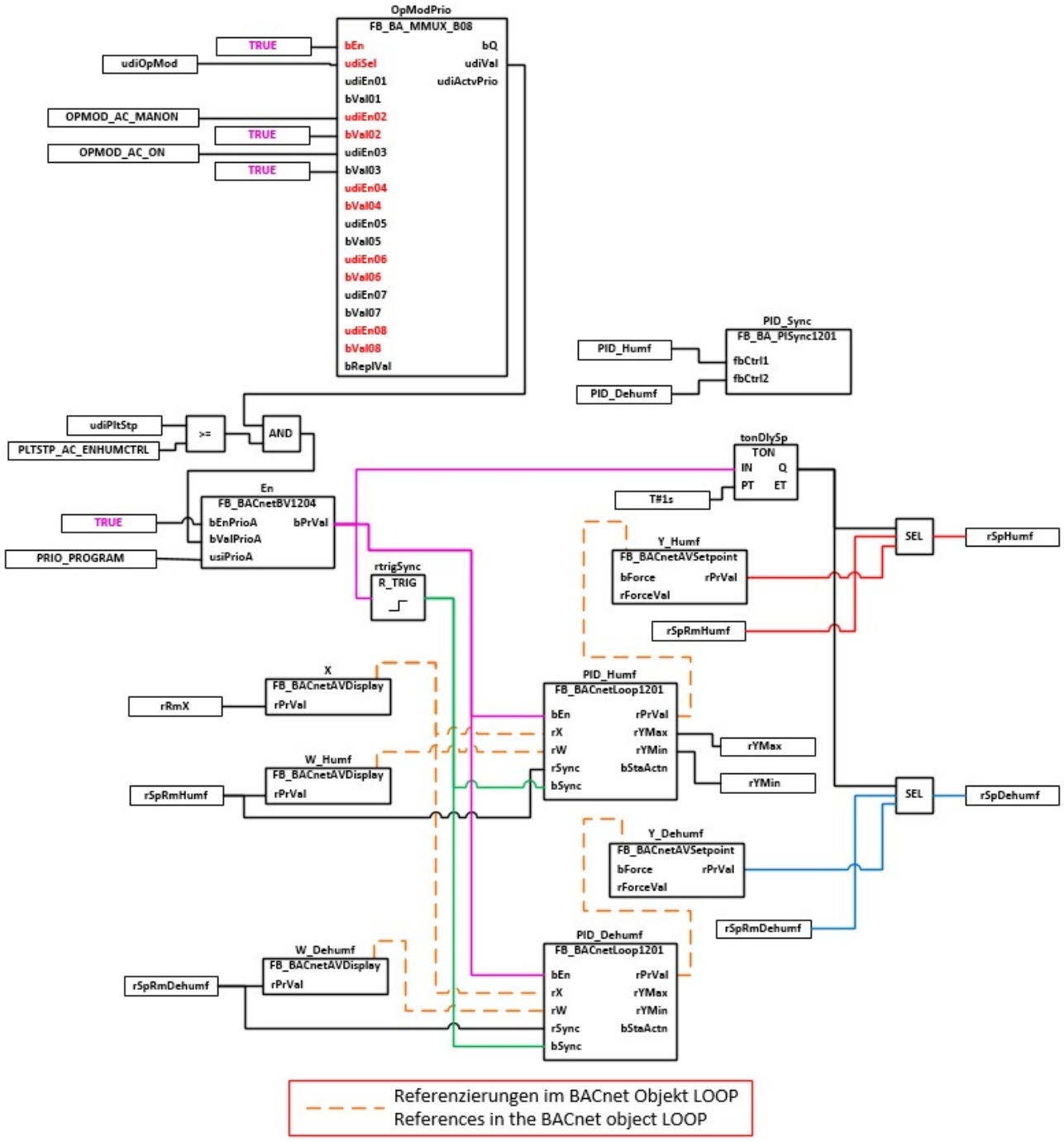
**Interface**



### Plant diagram



### Block diagram




### VAR_INPUT

```
rRmX          : REAL;
rSpRmHumf     : Real;
rSpRmDehumf   : Real;
udiPltStp     : REAL;
udiOpMod      : REAL;
```

**rRmX:** Actual value of absolute room humidity in g/kg

**rSpRmHumf:** Lower room set value (humidification) in g/kg

**rSpRmDehumf:** Upper room set value (dehumidification) in g/kg

**udiPltStp:** Stages of the air-conditioning plant startup. See also BAC_AC_StartTH_01 [▶ 535].

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515].

### VAR_OUTPUT

```
rY_Hum      : REAL;
rY_DeHum    : REAL;
```

**rY_Hum:** Supply air set value humidification g/kg

**rY_DeHum:** Supply air set value dehumidification g/kg

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm, [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt, [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task |
|---|---|---|
| X | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the actual value input of the BACnet loop objects |
| W_Humf | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the setpoint input of the BACnet loop object **PID_Humf** |
| W_Dehumf | FB_BACnetAVDisplay [▶ 69] | The AV object is referenced to the setpoint input of the BACnet loop object **PID_Dehumf** |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used for display and activation of the controller enable in the MCL or in a local operator display. |
| | >=, AND | The result of this network is the enable of the two PID master controllers **PID_Humf/PID_Dehumf**. The enable depends on the plant operation mode **udiOpMod** and the plant steps **udiPltStp** during startup of the ventilation system. |
| PID_Humf | FB_BACnetLoop1201 [▶ 98] | Master controller humidification |
| PID_Dehumf | FB_BACnetLoop1201 [▶ 98] | Master controller dehumidification |
| rtrigSync | R_TRIG | Upon receipt of the controller enable **bEn**, the two master controllers are synchronized with the input setpoints **W_Humf_rPrVal** / **W_Dehumf_rPrVal** triggered by a rising edge and **rtrigSync**. |
| Y_Humf | FB_BACnetAVSetpoint [▶ 69] | Supply air setpoint humidification. The AV object is referenced to the control value output of the BACnet loop object **PID_Humf** |
| Y_Dehumf | FB_BACnetAVSetpoint [▶ 69] | Supply air setpoint dehumidification. The AV object is referenced to the control value output of the BACnet loop object **PID_Dehumf** |

| Instance | Type | Task |
|---|---|---|
| TLogY_Humf | FB_BACnetTLog1201 [▶ 135] | Trend logging of the supply air setpoint for heating **Y_Humf** |
| TLogY_Dehumf | FB_BACnetTLog1201 [▶ 135] | Trend logging of the supply air setpoint for heating **Y_Humf** |
| PID_Sync | FB_BA_PISync1201 [▶ 162] | The function block synchronizes the parameters for the two master controllers **PID_Humf/PID_Dehumf** |
| tonDlySp | TON<br>SEL<br>SEL | Network for output of setpoints to the outputs of the template<br>After receiving the enable **bEn**, the setpoints of the master controllers are output at the outputs of the template with a delay of one second **tonDlySp** **rSpHumf** / **rSpDehumf**.<br>In the non-enabled state of the system, the input setpoints of the template are output. |

### Version history

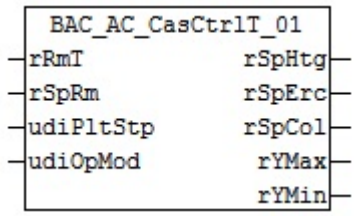
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.80.3 BAC_AC_CasCtrlT_01

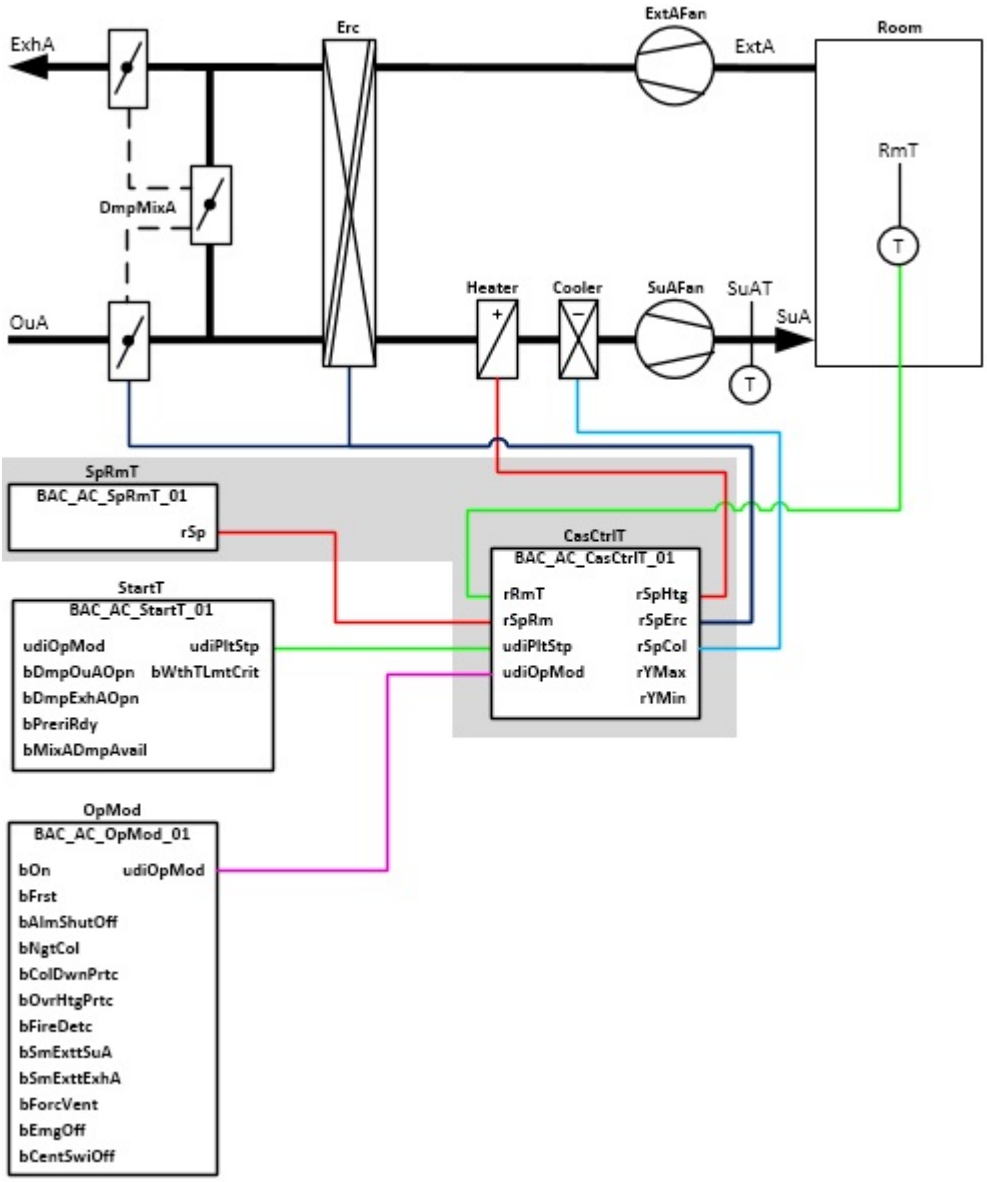
### Application

The template is a cascade controller for the supply air temperature, consisting of a master controller for set value calculation for heating, cooling and energy recovery.

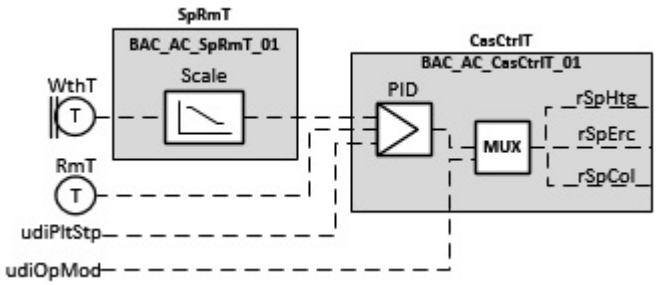
### Interface



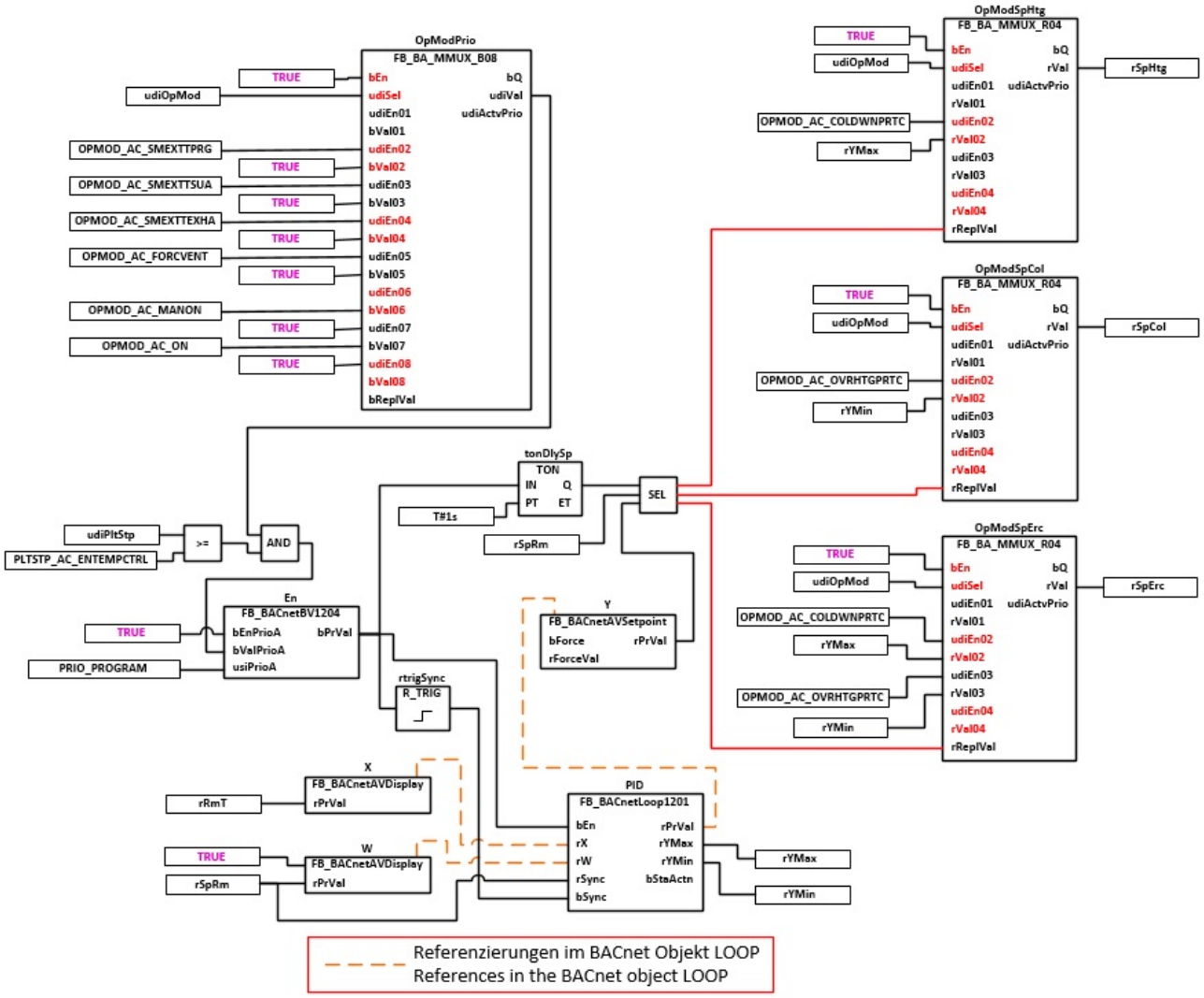
### Plant diagram 01




### Plant diagram 02



### Block diagram




### VAR_INPUT

```
rRmT          : REAL;
rSpRm         : REAL;
udiPltStp     : UDINT;
udiOpMod      : UDINT;
```

**rRmT:** Input variable to which the room temperature is applied. The room temperature is the control value of the PID master controller. If no room temperature is available, the outlet air temperature of a ventilation system can be used as control value.

**rSpRm:** Input variable for the room temperature set value, see template BAC_AC_SpRmT_01 [▶ 648].

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_Start_01 [▶ 530]

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rSpHtg      : REAL;
rSpErc      : REAL;
rSpCol      : REAL;
rYMax       : REAL;
rYMin       : REAL;
```

**rSpHtg:** Calculated set value of the supply air temperature for heating

**rSpErc:** Calculated set value of the supply air temperature for energy recovery

**rSpCol:** Calculated set value of the supply air temperature for cooling

**rYMax:** Upper value of the controller output limitation

**rYMin:** Lower value of the controller output limitation

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task | | |
|---|---|---|---|---|
| X | BACnetAVDisplay [▶ 69] | The input variable **rRmT** is connected to the AV object.<br>It is referenced to the actual value input of the BACnet loop object **PID**. | | |
| W | BACnetAVDisplay [▶ 69] | The input variable **rSpRm** is connected to the AV object.<br>It is referenced to the setpoint input of the BACnet loop object **PID**. | | |
| OpModPrio | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the enable conditions of the PID master controller depending on the plant operation mode **udiOpMod**. | | |
| | | **udiOpMod** | | **Enable cascade control** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | TRUE |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | TRUE |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | TRUE |
| | | OPMOD_AC_FORCVENT | Forced ventilation | TRUE |
| | | OPMOD_AC_MANON | Manual on | TRUE |
| | | OPMOD_AC_ON | On | TRUE |
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the controller enable in the MCL or in a local operator display | | |
| | >=, AND | The result of this network is the PID master controller enable. The enable depends on the plant operation mode and the plant steps **udiPltStp** during startup of the ventilation system. | | |
| PID | FB_BACnetLoop1201 [▶ 98] | Master controller for a room temperature control via room supply air cascade. Supplies a simple supply air temperature setpoint. | | |
| rtrigSync | R_TRIG | Upon receipt of the controller enable **En**, **rtrigSync** triggers synchronization of the master controller **PID** with its input value **lrSync**. | | |
| Y | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object PID. It shows the supply air temperature setpoint. | | |

| Instance | Type | Task | | |
|---|---|---|---|---|
| TLogY | FB_BACnetTLog1201 [▶ 135] | Trend logging of the supply air temperature setpoint **Y** | | |
| tonDlySp | TON<br>SEL | Upon receipt of the controller enable **En**, this network forwards the supply air temperature setpoint to the downstream multiplexers with a delay. | | |
| OpModSp Htg | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the heating setpoint **rSpHtg** depending on the plant operation mode **udiOpMod** in certain cases. | | |
| | | **udiOpMod** | **Setpoint** | **udiOpMod** |
| | | OPMOD_AC_COLDWNPRTC | Support operation, cooling protection | rYMax, upper setpoint of the controller output limitation |
| OpModSp Col | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the cooling setpoint **rSpCol** depending on the plant operation mode **udiOpMod** in certain cases. | | |
| | | **udiOpMod** | | **Setpoint** |
| | | OPMOD_AC_OVRHTGPRTC | Overheating protection | rYMin, lower setpoint of the controller output limitation |
| OpModSp Erc | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the energy recovery setpoint **rSpErc** depending on the plant operation mode **udiOpMod** in certain cases. see "Extracted nested table 121" | | |
| | | **udiOpMod** | | **Setpoint** |
| | | OPMOD_AC_COLDWNPRTC | Support operation, cooling protection | rYMax, upper setpoint of the controller output limitation |
| | | OPMOD_AC_OVRHTGPRTC | Overheating protection | rYMin, lower setpoint of the controller output limitation |

### Version history

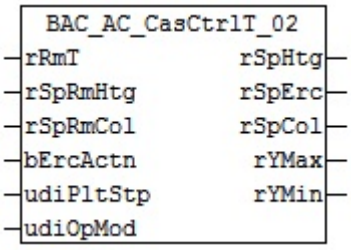
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.4 BAC_AC_CasCtrlT_02

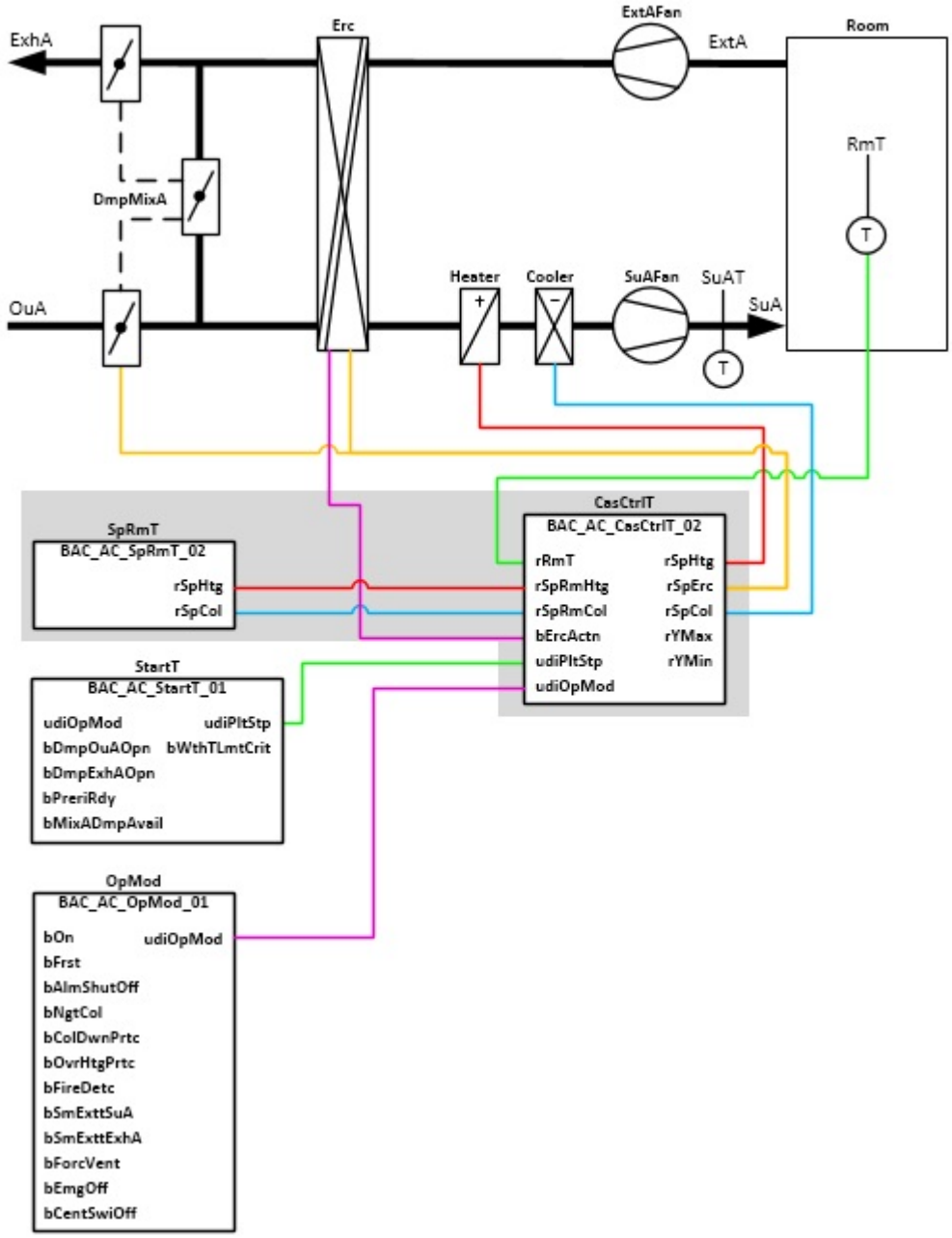
### Application

The template is a cascade controller for the supply air temperature, consisting of two master controllers for set value calculation for heating, cooling and energy recovery.

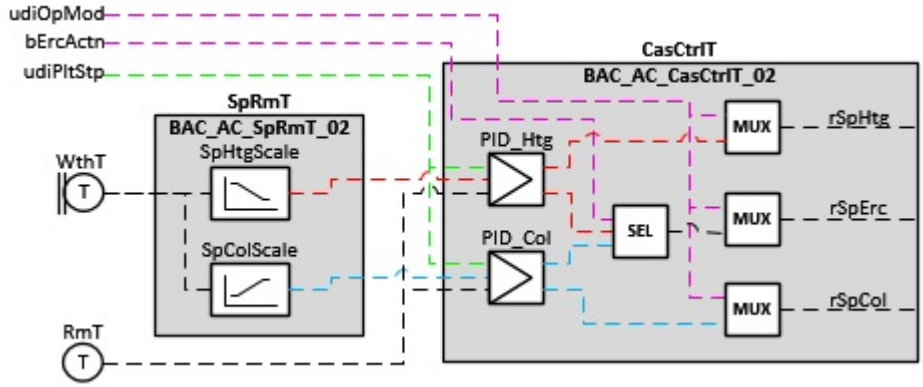
### Interface



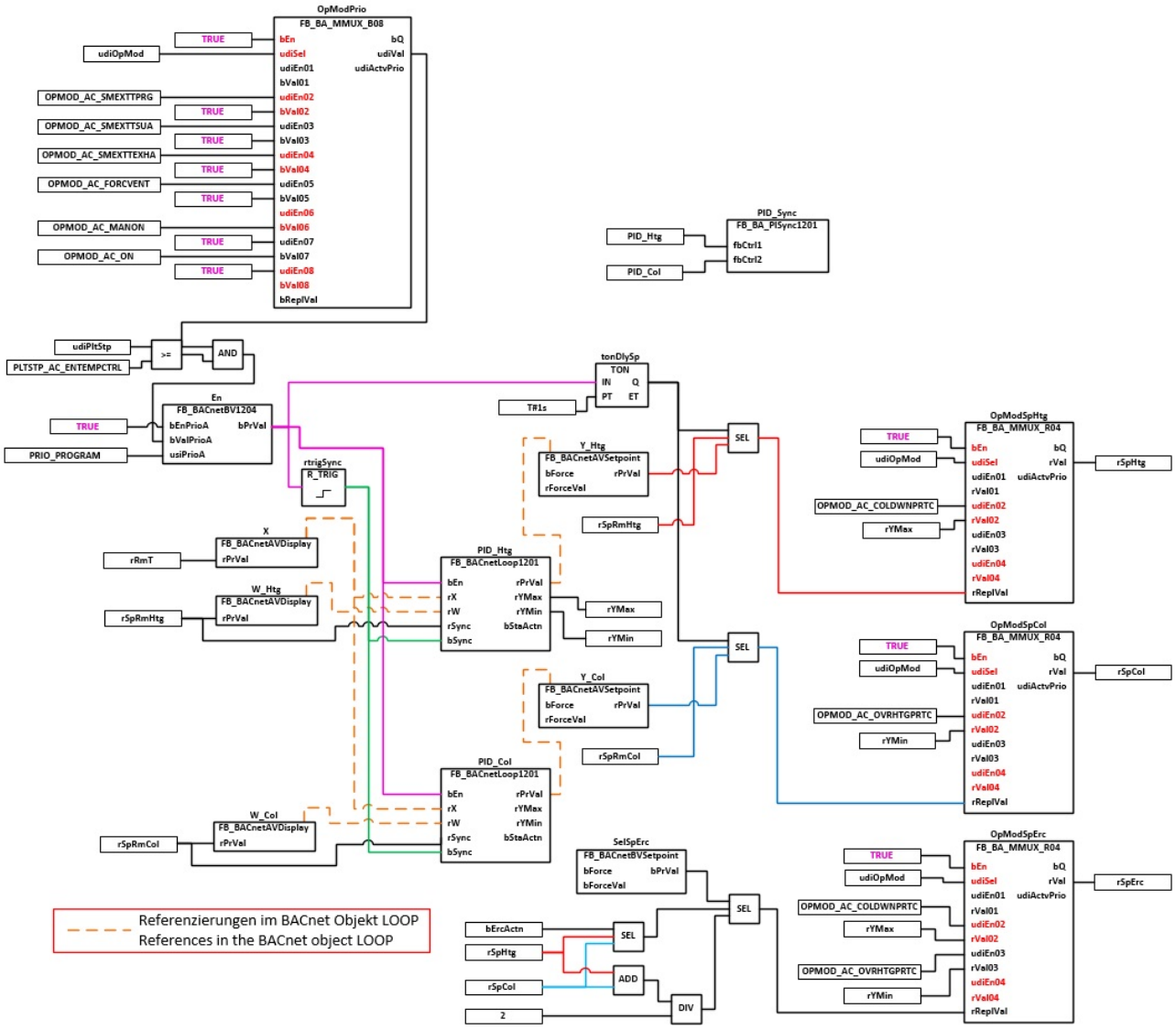
### Plant diagram 01




### Plant diagram 02



### Block diagram




### VAR_INPUT

```
rRmT          : REAL;
rSpRmHtg      : REAL;
rSpRmCol      : REAL;
udiPltStp     : UDINT;
udiOpMod      : UDINT;
```

**rRmT:** Input variable to which the room temperature is applied. The room temperature is the control value of the PID master controllers. If no room temperature is available, the outlet air temperature of a ventilation system can be used as control value.

**rSpRmHtg:** Input variable for the room temperature set value for heating, see template BAC_AC_SpRmT_02 [▶ 651].

**rSpRmCol:** Input variable for the room temperature set value for cooling, see template BAC_AC_SpRmT_02 [▶ 651].

**bErcActn:** Input variable to which the direction of action for the energy recovery is applied. The set value for the energy recovery is determined depending on the direction of action.
TRUE = direct = cooling; FALSE = reverse = heating

**udiPltStp:** Steps during startup of the air-conditioning plant. See also BAC_AC_Start_01 [▶ 530]

**udiOpMod:** Plant operation mode. See also BAC_AC_OpMod_01 [▶ 515]

### VAR_OUTPUT

```
rSpHtg      : REAL;
rSpErc      : REAL;
rSpCol      : REAL;
rYMax       : REAL;
rYMin       : REAL;
```

**rSpHtg:** Calculated set value of the supply air temperature for heating

**rSpErc:** Calculated set value of the supply air temperature for energy recovery

**rSpCol:** Calculated set value of the supply air temperature for cooling

**rYMax:** Upper value of the controller output limitation

**rYMin:** Lower value of the controller output limitation

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | Task | | |
|---|---|---|---|---|
| X | FB_BACnetAVDisplay [▶ 69] | The input variable **rRmT** is connected to the AV object.<br>It is referenced to the actual value inputs of the BACnet loop objects **PID_Htg** and **PID_Col**. | | |
| W_Htg | FB_BACnetAVDisplay [▶ 69] | The input variable **rSpRmHtg** is connected to the AV object.<br>It is referenced to the setpoint input of the BACnet loop object **PID_Htg**. | | |
| W_Col | FB_BACnetAVDisplay [▶ 69] | The input variable **rSpRmCol** is connected to the AV object.<br>It is referenced to the setpoint input of the BACnet loop object **PID_Col**. | | |
| OpModPrio | FB_BA_MMUX_B08 [▶ 205] | The multiplexer defines the enable conditions of the two PID master controllers depending on the plant operation mode **udiOpMod**. | | |
| | | **udiOpMod** | | **Enable cascade control** |
| | | OPMOD_AC_SMEXTTPRG | Smoke extraction program | TRUE |
| | | OPMOD_AC_SMEXTTSUA | Smoke extraction supply air | TRUE |
| | | OPMOD_AC_SMEXTTEXHA | Smoke extraction exhaust air | TRUE |
| | | OPMOD_AC_FORCVENT | Forced ventilation | TRUE |
| | | OPMOD_AC_MANON | Manual on | TRUE |
| | | OPMOD_AC_ON | On | TRUE |

| Instance | Type | Task |
|---|---|---|
| En | FB_BACnetBV1204 [▶ 94] | The BV object is used to display the controller enable in the MCL or in a local operator display. |
| | >=, AND | The result of this network is the enable of the two PID master controllers. The enable depends on the plant operation mode **udiOpMod** and the plant steps **udiPltStp** during startup of the ventilation system. |
| PID_Htg | FB_BACnetLoop1201 [▶ 98] | Master controller for a room temperature control via room supply air cascade. It provides the supply air temperature setpoint for heating. |
| PID_Col | FB_BACnetLoop1201 [▶ 98] | Master controller for a room temperature control via room supply air cascade. It provides the supply air temperature setpoint for cooling. |
| rtrigSync | R_TRIG | Upon receipt of the controller enable **En**, **rtrigSync** triggers synchronization of the two master controllers **PID_Htg** / **PID_Col** with the input value **lrSync**. |
| Y_Htg | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object **PID_Htg**. It shows the supply air temperature setpoint for heating. |
| TLogY_Htg | FB_BACnetTLog1201 [▶ 135] | Trend logging of the supply air temperature setpoint for heating **Y_Htg**. |
| Y_Col | FB_BACnetAVSetpoint [▶ 69] | The AV object is referenced to the control value output of the BACnet loop object **PID_Col**. It shows the supply air temperature setpoint for cooling. |
| TLogY_Col | FB_BACnetTLog1201 [▶ 135] | Trend logging of the supply air temperature setpoint for cooling **Y_Col**. |
| PID_Sync | FB_BA_PISync1201 [▶ 162] | The function block synchronizes the parameters for the two master controllers **PID_HTG** / **PID_COL**. |
| tonDlySp | TON<br>SEL | Upon receipt of the controller enable **En**, this network forwards the supply air temperature setpoints to the downstream multiplexers with a delay. |
| OpModSpHtg | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the heating setpoint **rSpHtg** depending on the plant operation mode **udiOpMod** in certain cases. |
| | | **udiOpMod** / **Setpoint** |
| | | OPMOD_AC_COLDWNPRTC / Support operation, cooling protection / rYMax, upper setpoint of the controller output limitation |
| OpModSpCol | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the cooling setpoint **rSpCol** depending on the plant operation mode **udiOpMod** in certain cases. |
| | | **udiOpMod** / **Setpoint** |
| | | OPMOD_AC_OVRHTGPRTC / Overheating protection / rYMin, lower setpoint of the controller output limitation |
| SelSpErc | FB_BACnetBV1204 [▶ 94] | Selection strategy setpoint energy recovery. **TRUE:** average of **rSpHtg** and **rSpCol**; **FALSE:** depends on the control direction, defined by input **bErcActn** |
| | SEL, ADD, DIV, SEL | The result of this network supplies two setpoints for energy recovery. Which value is used depends on the energy recovery setpoint strategy, see **SelSpErc** |
| OpModSpErc | FB_BA_MMUX_R04 [▶ 205] | The multiplexer defines the energy recovery setpoint **rSpErc** depending on the plant operation mode **udiOpMod** in certain cases. |
| | | **udiOpMod** / **Setpoint** |
| | | OPMOD_AC_COLDWNPRTC / Support operation, cooling protection / rYMax, upper setpoint of the controller output limitation |

| Instance | Type | Task | | |
|---|---|---|---|---|
| | | OPMOD_AC_OVRHTGPRTC | Overheating protection | rYMin, lower setpoint of the controller output limitation |

**Version history**

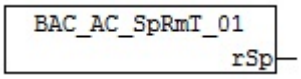
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.5 BAC_AC_SpRmT_01

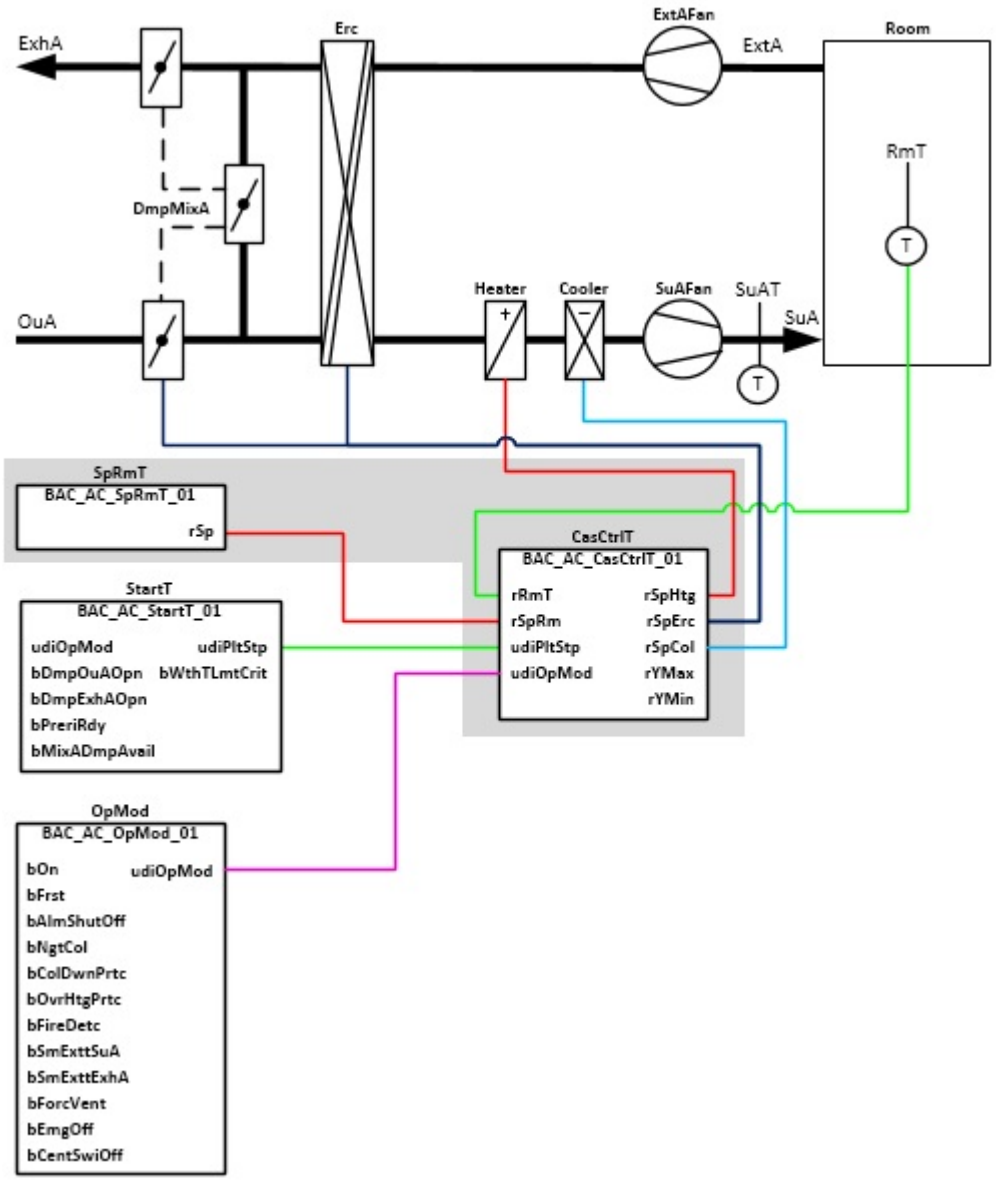
**Application**

The template is a set value program for an room air/supply air cascade with only one room temperature set value, including summer and winter compensation.

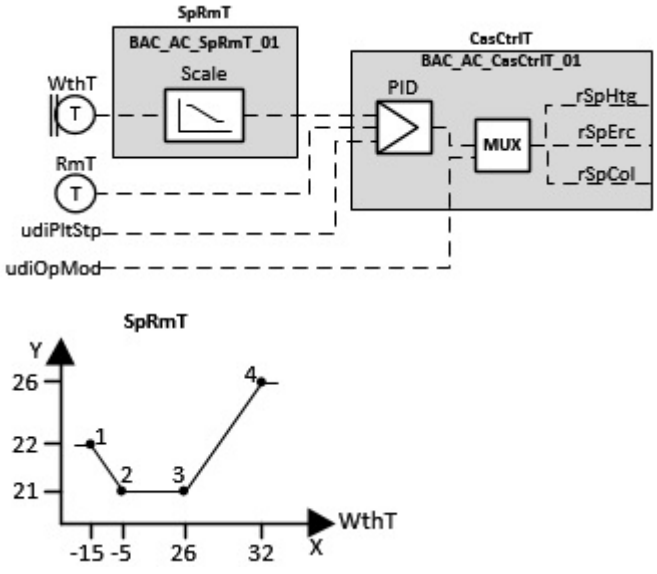
**Interface**



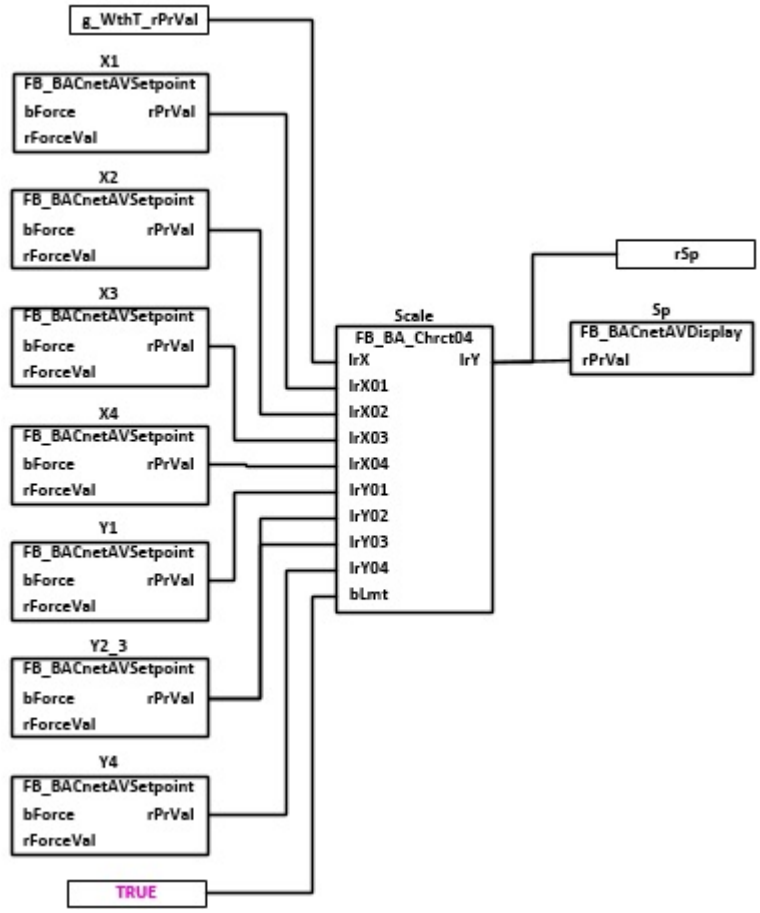
### Plant diagram 01




### Plant diagram 02



### Block diagram




### VAR_OUTPUT

```
rSp     : REAL;
```

**rSp:** Calculated set value for the room temperature

### Program description

| Instance | Type | Task |
|---|---|---|
| X1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 |
| X2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 |
| X3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points X3 |
| X4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X4 |
| Y1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 |
| Y2_3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points Y2/Y3 |
| Y4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y4 |
| Scale | FB_BA_Chrct04 [▶ 172] | The function block calculates the set value characteristic curve for the current room temperature, depending on the outside temperature. |
| Sp | FB_BACnetAVDisplay [▶ 69] | Output of the calculated, simple room temperature set value. It is output at output **rSp**. |

### Version history

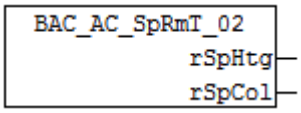
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.6 BAC_AC_SpRmT_02

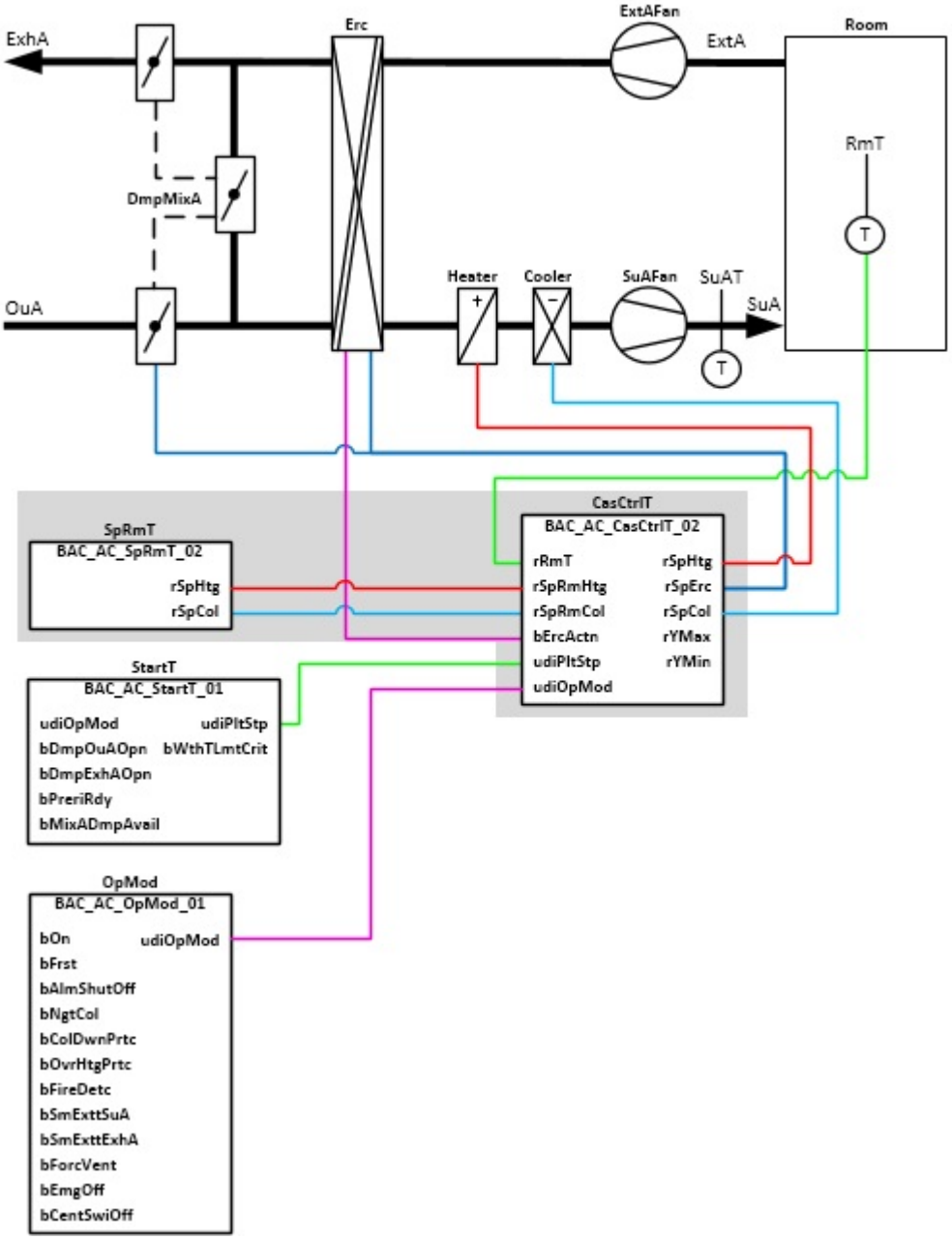
### Application

Set value program for an outlet air/inlet air cascade with a separate room temperature set value for heating and cooling mode, including summer and winter compensation

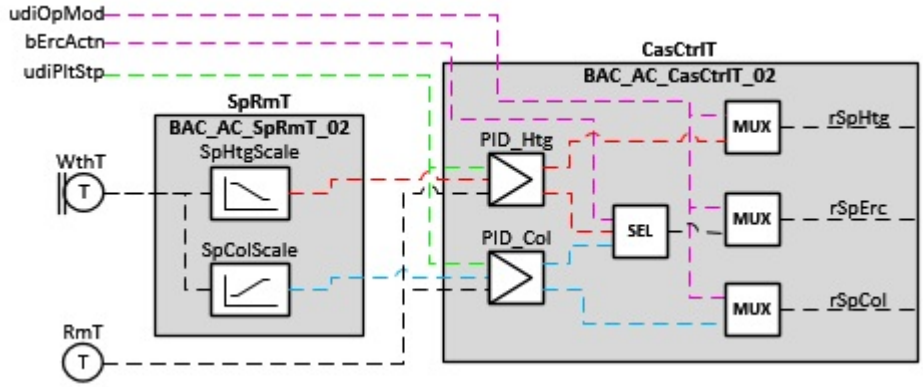
### Interface



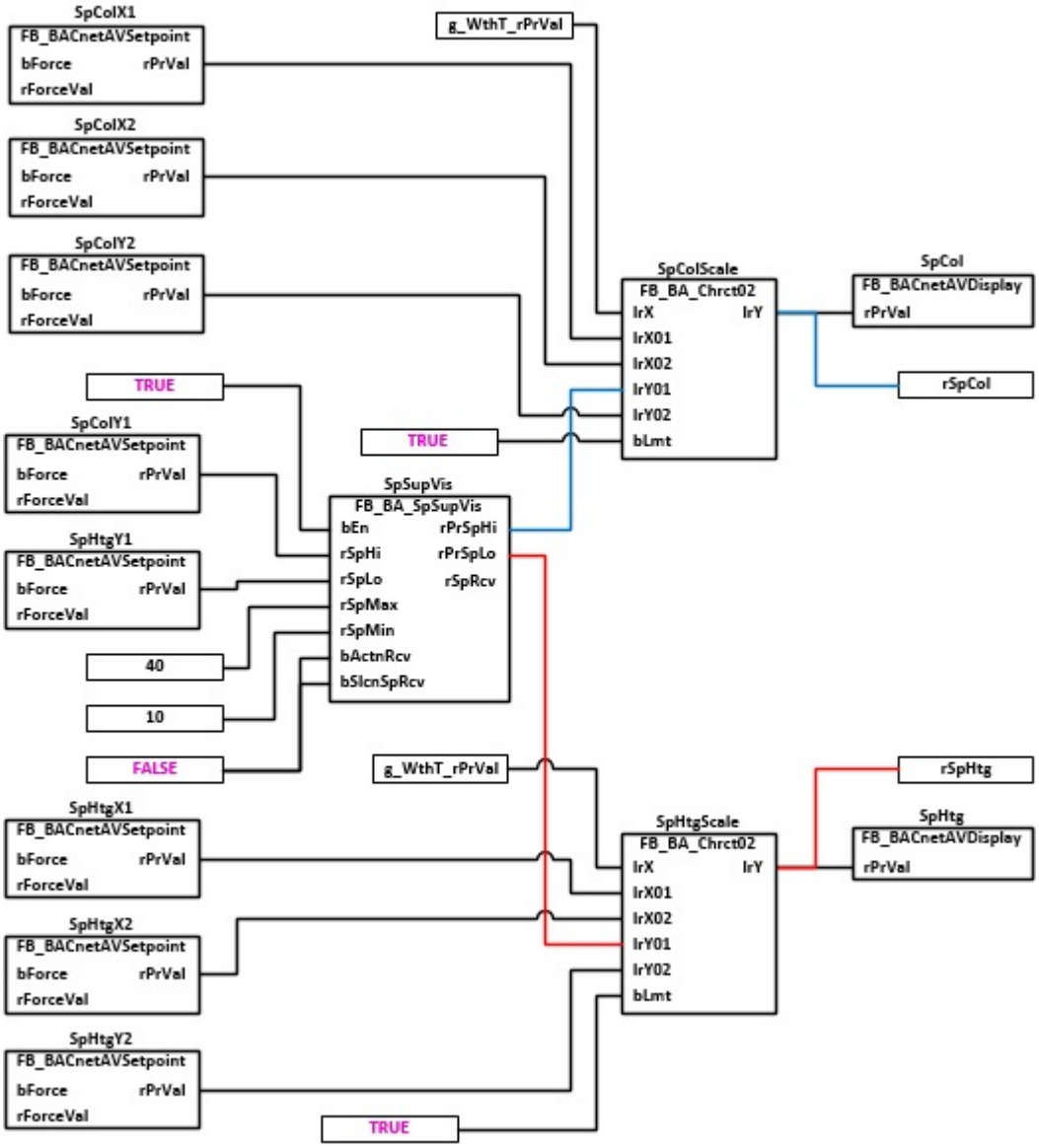
### Plant diagram 01




### Plant diagram 02



### Block diagram




### VAR_OUTPUT

```
rSp     : REAL;
```

**rSpHtg:** Calculated set value of the room temperature for heating

**rPrCol:** Calculated set value of the room temperature for cooling

### Program description

| Instance | Type | Task |
|---|---|---|
| SpColX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 of the summer compensation for the cooling setpoint. |
| SpColX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 of the summer compensation for the cooling setpoint. |
| SpColY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 of the summer compensation for the cooling setpoint. SpColY1 is the base cooling setpoint. |
| SpColY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y2 of the summer compensation for the cooling setpoint. |
| SpColScale | FB_BA_Chrct02 [▶ 171] | The function block determines the summer compensation value for the cooling setpoint. |

| Instance | Type | Task |
|---|---|---|
| SpHtgX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 of the winter compensation for the heating setpoint. |
| SpHtgX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 of the winter compensation for the heating setpoint. |
| SpHtgY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 of the winter compensation for the heating setpoint. SpHtgY1 is the base heating setpoint. |
| SpHtgY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y2 of the winter compensation for the heating setpoint. |
| SpHtgScale | FB_BA_Chrct02 [▶ 171] | The function block determines the winter compensation for the heating setpoint. |
| SpSuVis | FB_BA_SpSupVis [▶ 243] | The function block is used to control the two interpolation points SpHtgY1 and SpColY1. |
| SpHtg | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature heating setpoint |
| SpCol | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature cooling setpoint |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.80.7 BAC_AC_SpRmTH_01

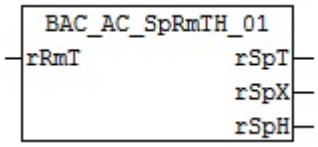
### Application

The template **BAC_AC_SpRmTH_01** is a simple set value program for an air conditioning plant with room or extractair cascade control for temperature and humidity.
There is a basic room temperature set value **SpT_Y2_3**. There is no energy-neutral zone between a lower set value (heating mode) and an upper set value (cooling mode). An outside temperature-dependent offset of the basic room temperature set value is realised via a curve function (summer/winter compensation).

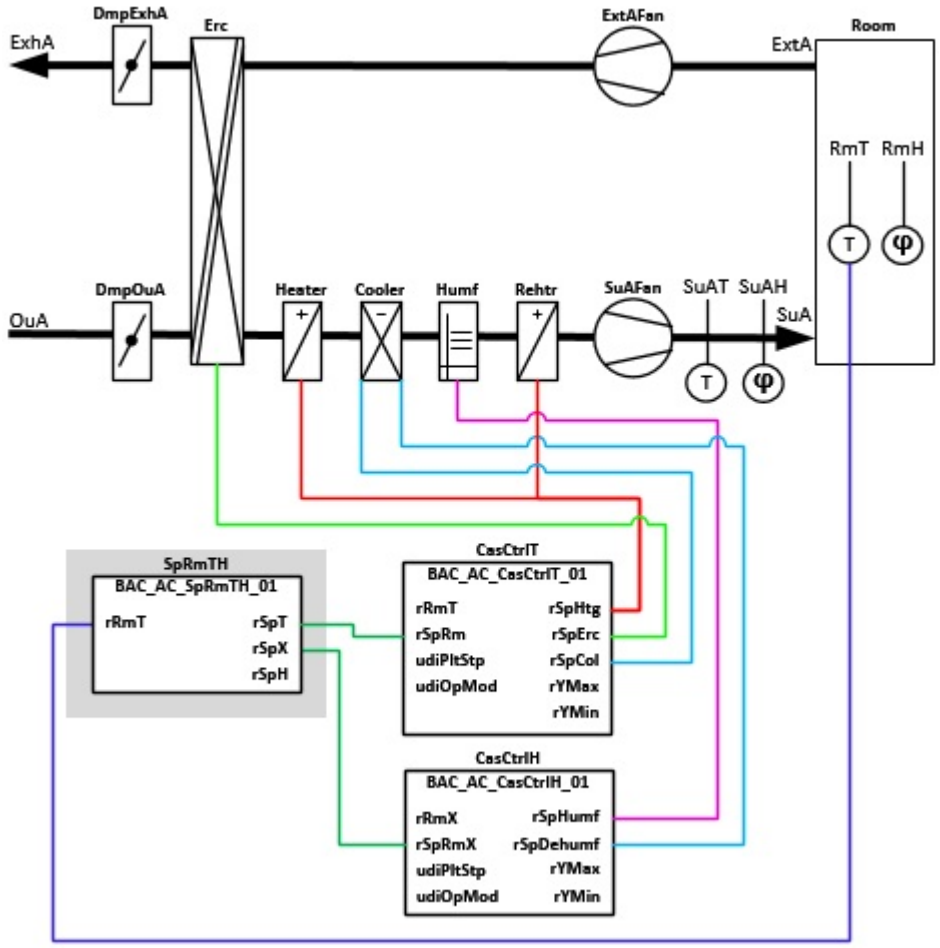
The relative room humidity set value (%) is entered with BACnet AV object **SpH**.
There is no energy-neutral zone between a lower set value (humidification) and an upper set value (dehumidification) for humidity control.

The set value for the relative room humidity is converted into a set value for the absolute room humidity (g/kg) **SpX** within the template via the current room temperature.

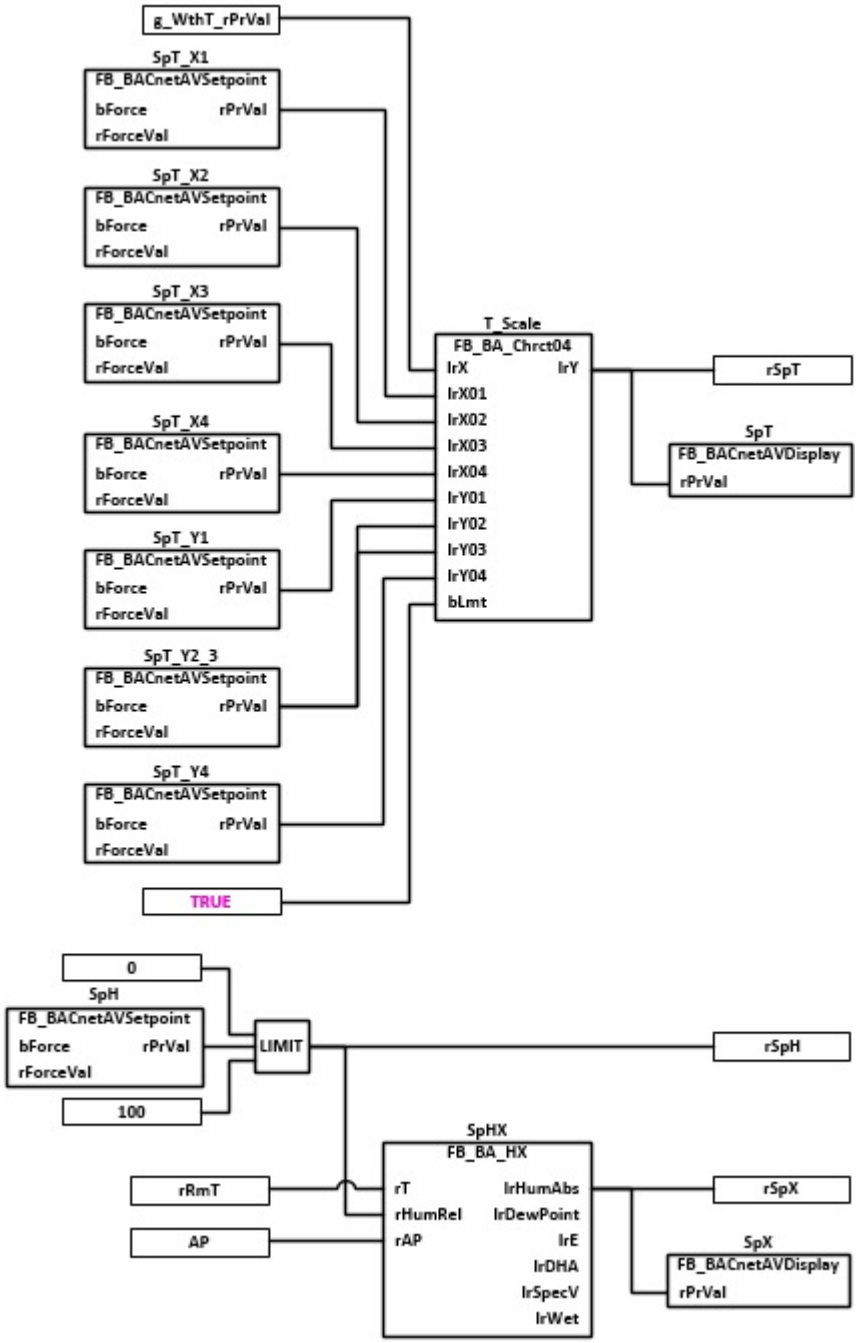
### Interface



### System diagram



### Block diagram




### VAR_Input

```
rRmT      : REAL;
```

**rRmT:** Measured value of the room temperature

### VAR_OUTPUT

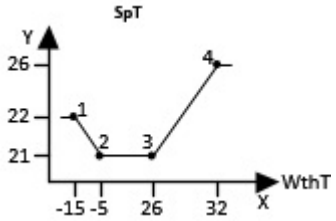
```
rSpT      : REAL;
rSpX      : REAL;
rSpH      : Real;
```

**rSpT:** Calculated set value for the room temperature

**rSpX:** Calculated set value of the absolute room humidity

**rSpH**: Set value for the relative room humidity

**Program description**

| Instance | Type | Task |
|---|---|---|
| SpT_X1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 |
| SpT_X2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 |
| SpT_X3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points X3 |
| SpT_X4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X4 |
| SpT_Y1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 |
| SpT_Y2_3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points Y2/Y3. **The value is the base setpoint value** |
| SpT_Y4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y4 |
| T_Scale | FB_BA_Chrct04 [▶ 172] | The function block calculates the set value characteristic curve for the current room temperature, depending on the outside temperature.   |
| SpT | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature set value. |
| SpH | FB_BACnetAVSetpoint [▶ 69] | Input of the set value for the relative room humidity (%). |
| SpHX | FB_BA_HX [▶ 227] | This function block is used to calculate the set value for the absolute room humidity. For calculating the value, the room temperature, the set value for the relative humidity **SpH** and the barometric air pressure are required. |
| SpX | FB_BACnetAVDisplay [▶ 69] | Output of the calculated absolute room humidity set value (g/kg). |

**Version history**

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.8 BAC_AC_SpRmTH_02

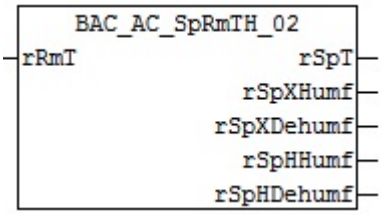
**Application**

The template **BAC_AC_SpRmTH_02** is a simple set value program for an air conditioning plant with room or extract air cascade control for temperature and humidity.
There is a basic room temperature set value **SpT_Y2_3**. There is no energy-neutral zone between a lower set value (heating mode) and an upper set value (cooling mode). An outside temperature-dependent offset of the basic room temperature set value is realized via a curve function (summer/winter compensation).

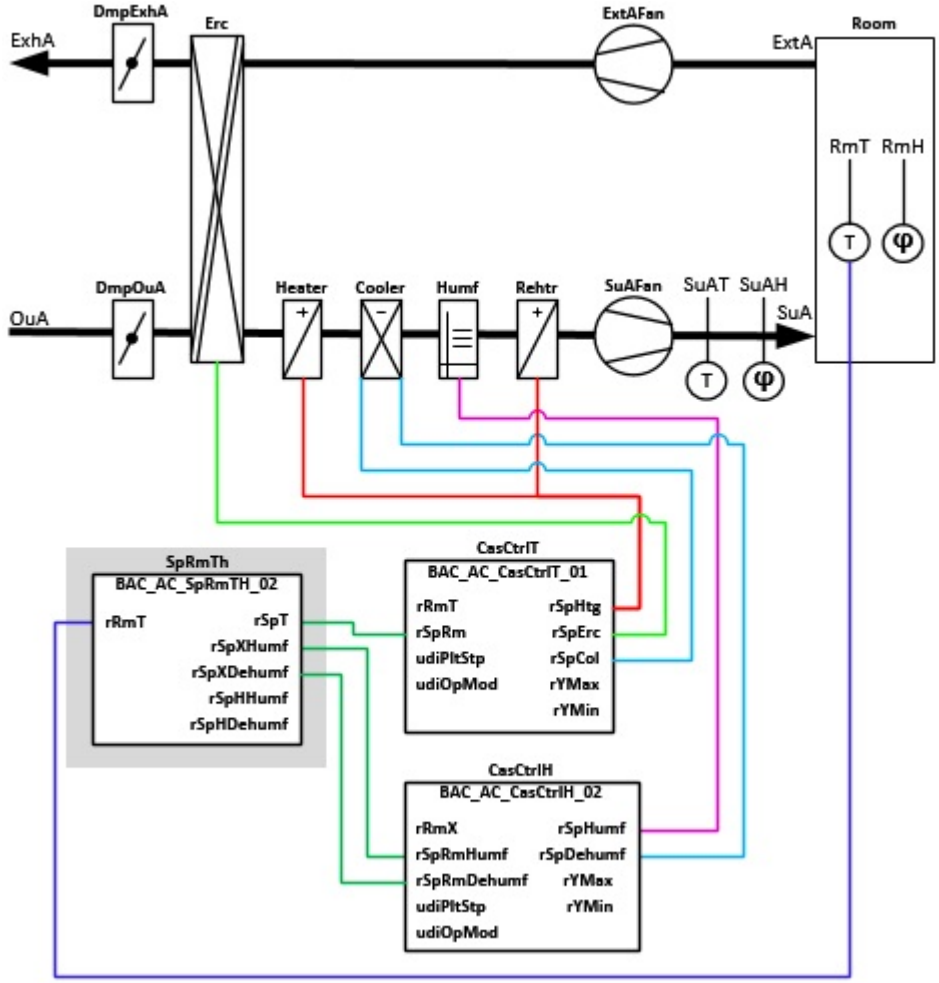
The relative room humidity set values (%) are entered via the BACnet AV objects **SpHHumf (humidification)** and **SpHDehumf (dehumidification)**. There is an energy-neutral zone between the lower set value (**humidification**) and the upper set value (**dehumidification**), which is monitored via the function **SpHSupVis**.

The two set values for the relative room humidity are converted within the template via the current room temperature to two set values for the absolute room humidity (g/kg), **SpXHumf (humidification)** and **SpXDehumf (dehumidification)**.

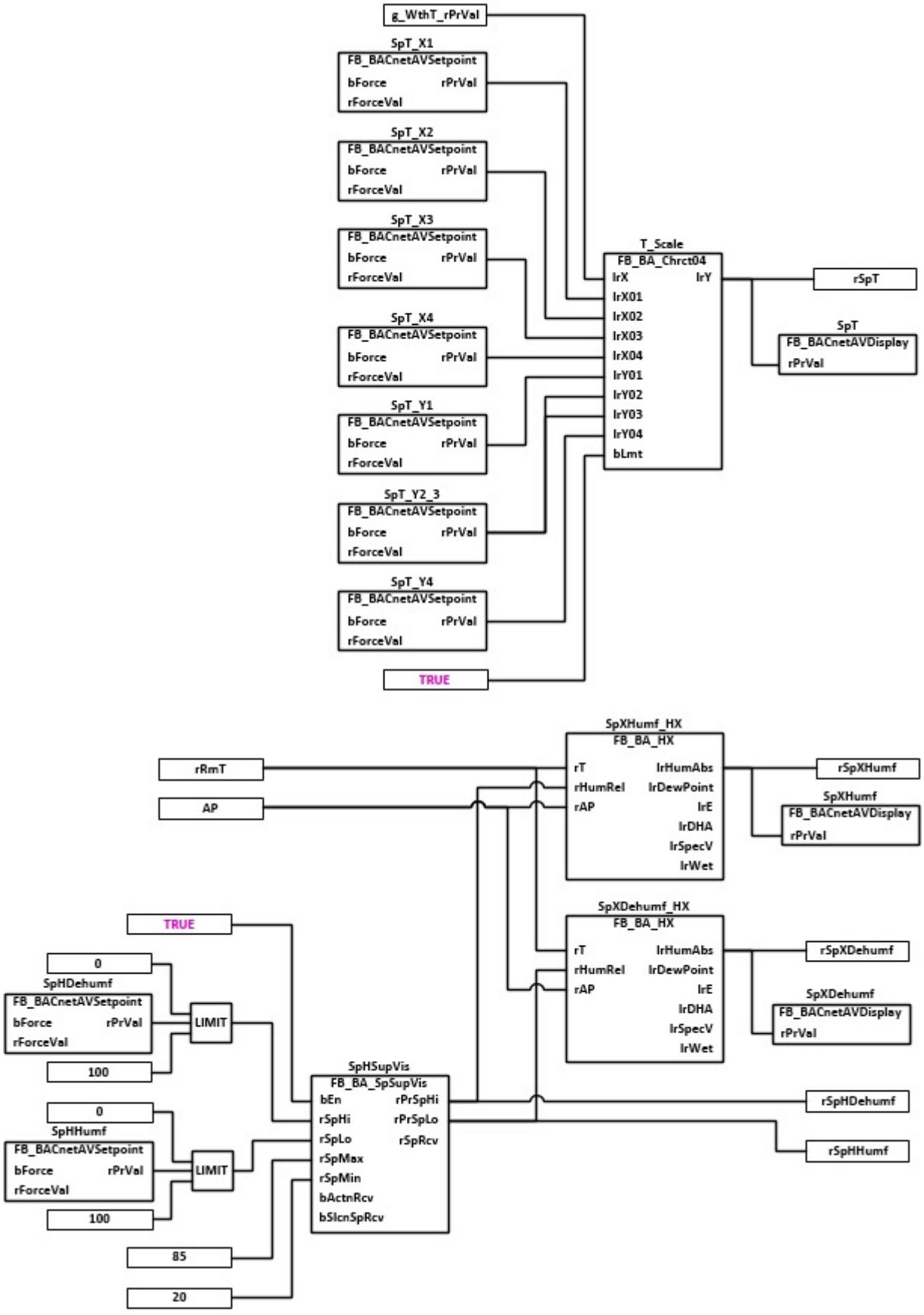
### Interface




### System diagram



## Block diagram




## VAR_Input

```
rRmT     : REAL;
```

**rRmT:** Measured value of the room temperature

### VAR_OUTPUT

```
rSpT           : REAL;
rSpXHumf       : REAL;
rSpXDehumf     : REAL;
rSpHHumf       : REAL;
rSpHDehumf     : REAL;
```

**rSpT:** Calculated set value for the room temperature

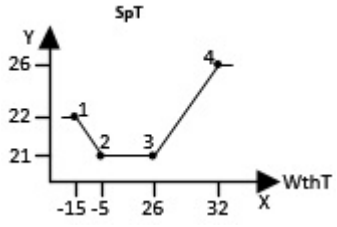
**rSpXHumf:** Set value absolute humidity; humidification

**rSpXDehumf:** Set value absolute humidity; dehumidification

**rSpHHumf:** Set value relative humidity; humidification

**rSpHDehumf:** Set value relative humidity; dehumidification

### Program description

| Instance | Type | Task |
|---|---|---|
| SpT_X1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 |
| SpT_X2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 |
| SpT_X3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points X3 |
| SpT_X4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X4 |
| SpT_Y1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 |
| SpT_Y2_3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points Y2/Y3. **The value is the base setpoint value** |
| SpT_Y4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y4 |
| T_Scale | FB_BA_Chrct04 [▶ 172] | The function block calculates the set value characteristic curve for the current room temperature, depending on the outside temperature.   |
| SpT | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature set value. |
| SpHHumf | FB_BACnetAVSetpoint [▶ 69] | Input of the relative room humidity set value for humidification (%). |
| SpHDehumf | FB_BACnetAVSetpoint [▶ 69] | Input of the relative room humidity set value for dehumidification (%). |
| SpHSupVis | FB_BA_SpSupVis [▶ 243] | The function block is used for controlling the energy-neutral zone between the two set values **SpHDehumf** and **SpHHumf** |

| Instance | Type | Task |
|---|---|---|
| SpXHumf_HX | FB_BA_HX [▶ 227] | This function block is used to calculate the set value for the absolute room humidity (humidification). For calculating the value, the room temperature, the set value for the relative humidity **SpHHumf** and the barometric air pressure are required. |
| SpXHumf | FB_BACnetAVDisplay [▶ 69] | Output of the calculated absolute room humidity set value for humidification (g/kg). |
| SpXDehumf_HX | FB_BA_HX [▶ 227] | This function block is used to calculate the set value for the absolute room humidity (dehumidification). For calculating the value, the room temperature, the set value for the relative humidity **SpHDehumf** and the barometric air pressure are required. |
| SpXDehumf | FB_BACnetAVDisplay [▶ 69] | Output of the calculated absolute room humidity set value for dehumidification (g/kg). |

### Version history

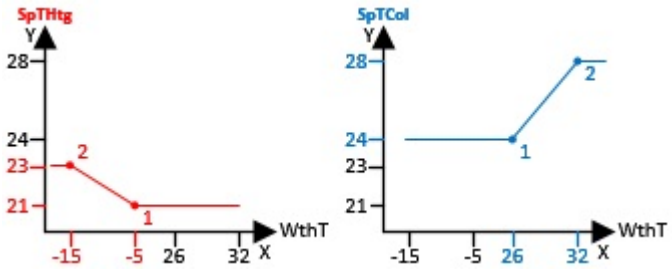
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.9 BAC_AC_SpRmTH_03

### Application

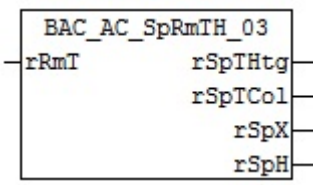
The template **BAC_AC_SpRmTH_02** is a set value program for an air conditioning plant with room or extract air cascade control for temperature and humidity.

There are two basic room temperature set values (**SpTHtgY1, SpTColY1**), with an energy-neutral zone between the lower set value (heating mode) and the upper set value (cooling mode). The energy-neutral zone is monitored via the function **SpHSupVis**.
An outside temperature-dependent offset of the basic room temperature set values is realised via two curve functions (summer/winter compensation).

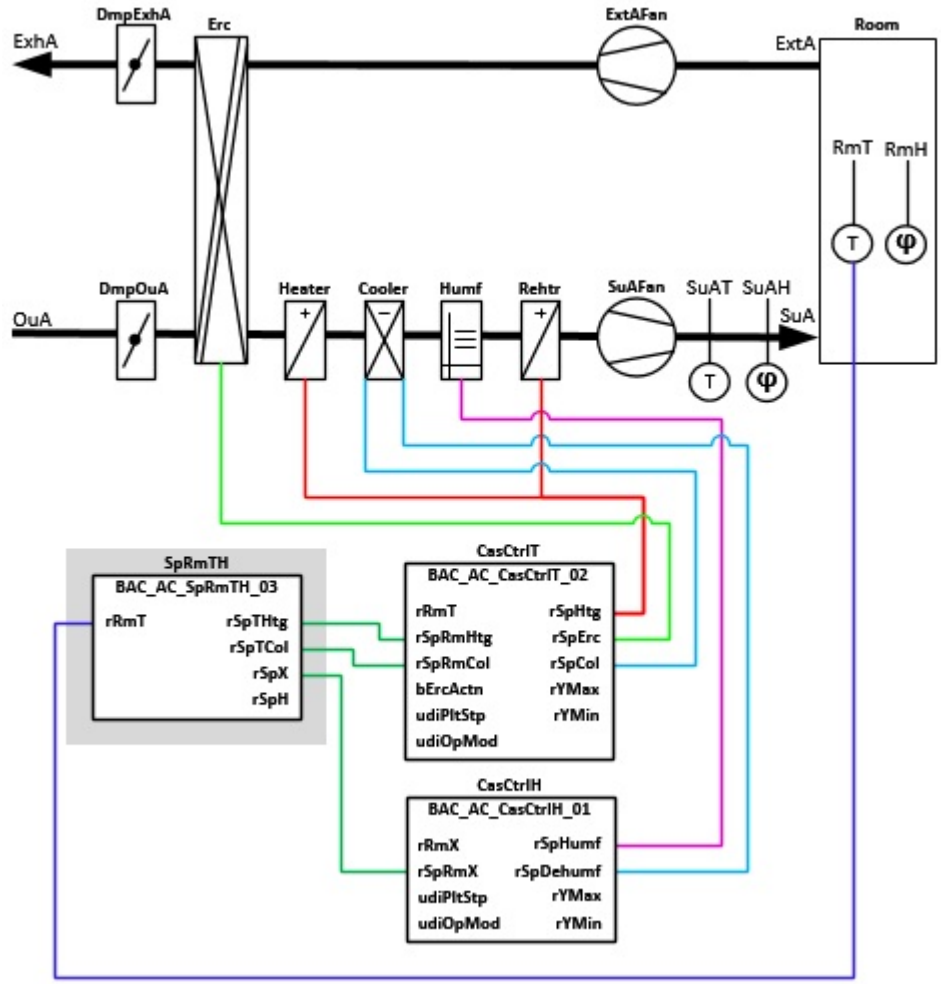



The relative room humidity set value (%) is entered with BACnet AV object **SpH**.
There is no energy-neutral zone between a lower set value (humidification) and an upper set value (dehumidification) for humidity control.
The set value for the relative room humidity **SpH** is converted into a set value for the absolute room humidity (g/kg) **SpX** within the template via the current room temperature.

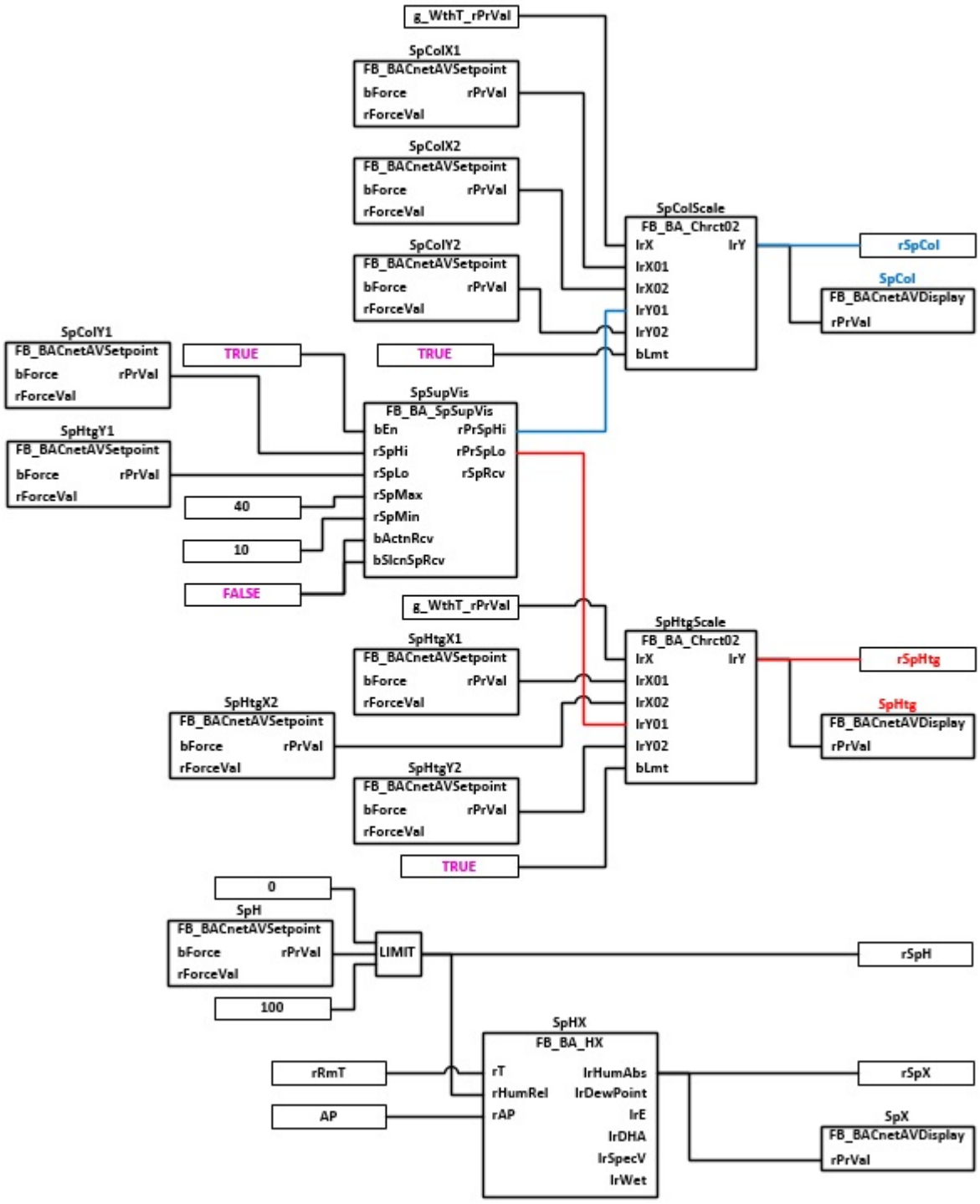
### Interface



### System diagram



### Block diagram




### VAR_Input

```
rRmT      : REAL;
```

**rRmT:** Measured value of the room temperature

### VAR_OUTPUT

```
rSpTHtg     : REAL;
rSpTCol     : REAL;
rSpX        : REAL;
rSpH        : Real;
```

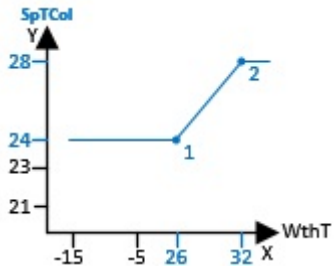
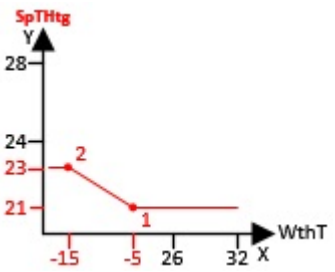
**rSpTHtg:** Calculated heating set value for the room temperature

**rSpTCol:** Calculated cooling set value for the room temperature

**rSpX:** Calculated set value of the absolute room humidity

**rSpH**: Set value for the relative room humidity

**Program description**

| Instance | Type | Task |
|---|---|---|
| SpTColX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColX1** cooling |
| SpTColX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColX2** cooling |
| SpTColY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColY1** cooling. **The value is the base cooling setpoint value** |
| SpTColY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColY2** cooling |
| SpTColScale | FB_BA_Chrct04 [▶ 172] | The function block calculates the characteristic cooling set value curve for the current room temperature, depending on the outside temperature.   |
| SpTCol | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature cooling set value |
| SpTHtgX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgX1** heating |
| SpTHtgX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgX2** heating |
| SpTHtgY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgY1** heating. **The value is the base heating setpoint value** |
| SpTHtgY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgY2** heating |
| SpTHtgScale | FB_BA_Chrct04 [▶ 172] | The function block calculates the characteristic heating set value curve for the current room temperature, depending on the outside temperature.   |
| SpTHtg | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature heating set value |
| SpTSupVis | FB_BA_SpSupVis [▶ 243] | The function block is used for controlling the energy-neutral zone between the two set values **SpTCol** and **SpTHtg** |

| Instance | Type | Task |
|---|---|---|
| SpH | FB_BACnetAVSetpoint [▶ 69] | Input of the set value for the relative room humidity (%). |
| SpHX | FB_BA_HX [▶ 227] | This function block is used to calculate the set value for the absolute room humidity. For calculating the value, the room temperature, the set value for the relative humidity **SpH** and the barometric air pressure are required. |
| SpX | FB_BACnetAVDisplay [▶ 69] | Output of the calculated absolute room humidity set value (g/kg). |

**Version history**

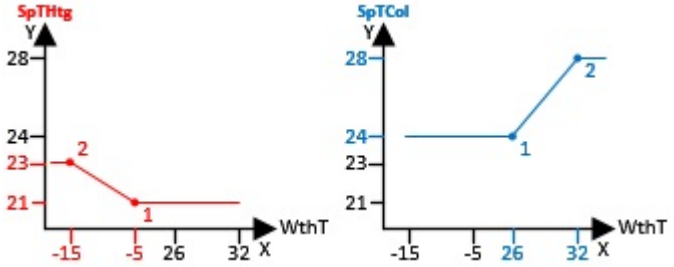
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.10 BAC_AC_SpRmTH_04

**Application**

The template **BAC_AC_SpRmTH_04** is a set value program for an air conditioning plant with room or extract air cascade control for temperature and humidity.

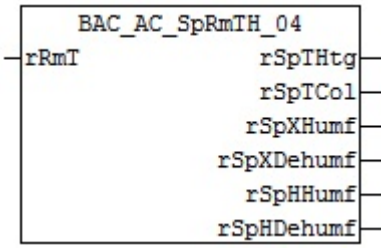
There are two basic room temperature set values (**SpTHtgY1, SpTColY1**), with an energy-neutral zone between the lower set value (heating mode) and the upper set value (cooling mode). The energy-neutral zone is monitored via the function **SpHSupVis**.
An outside temperature-dependent offset of the basic room temperature set values is realised via two curve functions (summer/winter compensation).



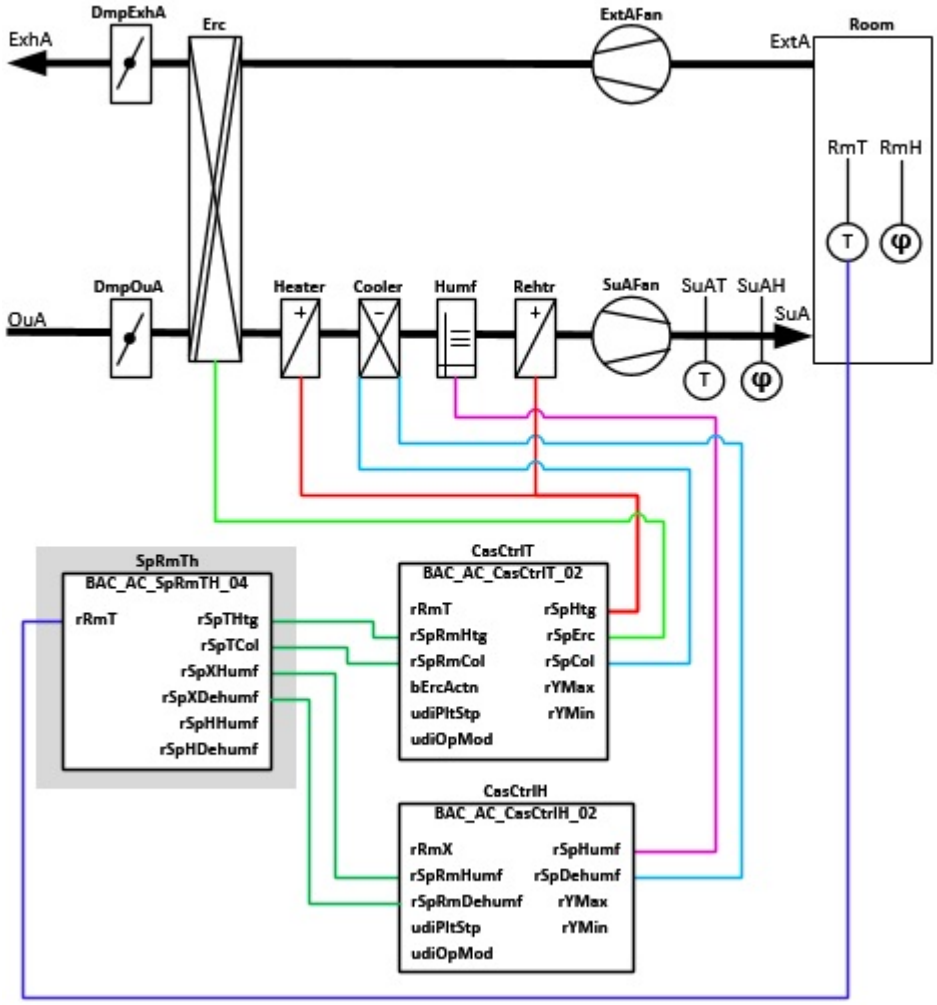

The relative room humidity set values (%) are entered via the BACnet AV objects **SpHHumf (humidification)** and **SpHDehumf (dehumidification)**. There is an energy-neutral zone between the lower set value (**humidification**) and the upper set value (**dehumidification**), which is monitored via the function **SpHSupVis**.

The two set values for the relative room humidity are converted within the template via the current room temperature to two set values for the absolute room humidity (g/kg), **SpXHumf (humidification)** and **SpXDehumf (dehumidification)**.

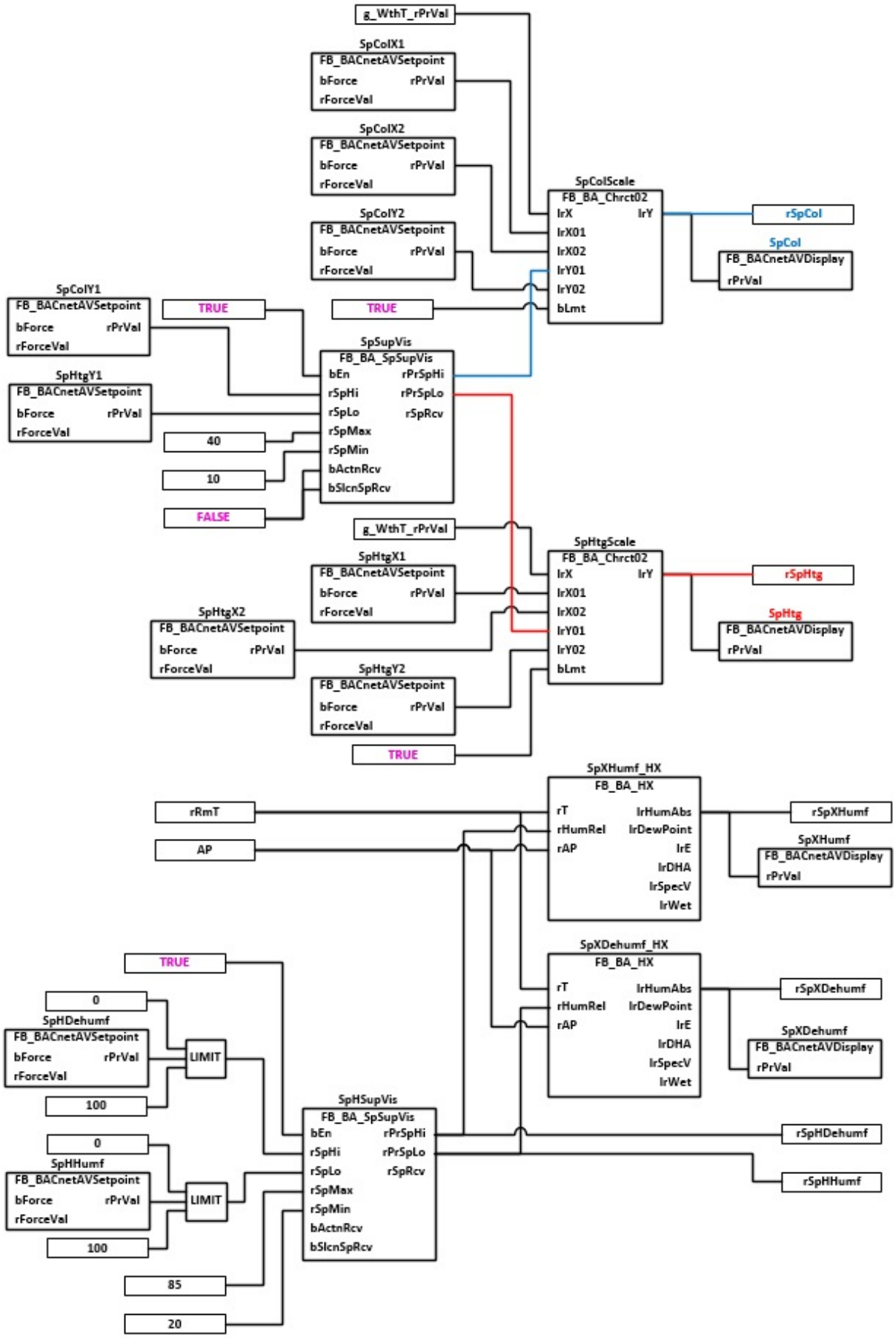
**Interface**



### System diagram



### Block diagram



### VAR_Input

```
rRmT      : REAL;
```

**rRmT:** Measured value of the room temperature

### VAR_OUTPUT

```
rSpTHtg         : REAL;
rSpTCol         : REAL;
rSpXHumf        : REAL;
rSpXDehumf      : REAL;
rSpHHumf        : Real;
rSpHDehumf      : Real;
```

**rSpTHtg:** Calculated heating set value for the room temperature

**rSpTCol:** Calculated cooling set value for the room temperature

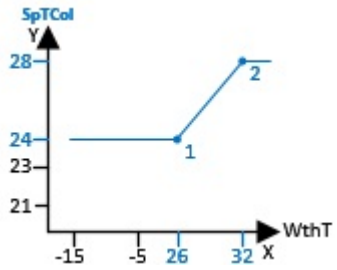
**rSpXHumf:** Calculated set value absolute humidity; humidification

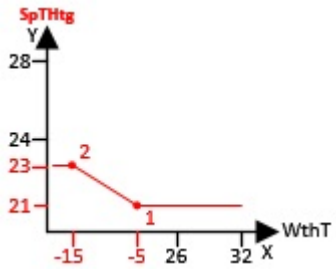
**rSpXDehumf:** Calculated set value absolute humidity; dehumidification

**rSpHHumf:** Set value relative humidity; humidification

**rSpHDehumf:** Set value relative humidity; dehumidification

### Program description

| Instance | Type | Task |
|---|---|---|
| SpTColX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColX1** cooling |
| SpTColX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColX2** cooling |
| SpTColY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColY1** cooling. **The value is the base cooling setpoint value** |
| SpTColY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTColY2** cooling |
| SpTColScale | FB_BA_Chrct04 [▶ 172] | The function block calculates the characteristic cooling set value curve for the current room temperature, depending on the outside temperature.   |
| SpTCol | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature cooling set value |
| SpTHtgX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgX1** heating |
| SpTHtgX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgX2** heating |
| SpTHtgY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgY1** heating. **The value is the base heating setpoint value** |
| SpTHtgY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point **SpTHtgY2** heating |

| Instance | Type | Task |
|---|---|---|
| SpTHtgScale | FB_BA_Chrct04 [▶ 172] | The function block calculates the characteristic heating set value curve for the current room temperature, depending on the outside temperature.   |
| SpTHtg | FB_BACnetAVDisplay [▶ 69] | Output of the calculated room temperature heating set value |
| SpTSupVis | FB_BA_SpSupVis [▶ 243] | The function block is used for controlling the energy-neutral zone between the two set values **SpTCol** and **SpTHtg** |
| SpHHumf | FB_BACnetAVSetpoint [▶ 69] | Input of the relative room humidity set value for humidification (%). |
| SpHDehumf | FB_BACnetAVSetpoint [▶ 69] | Input of the relative room humidity set value for dehumidification (%). |
| SpHSupVis | FB_BA_SpSupVis [▶ 243] | The function block is used for controlling the two set values **SpHDehumf** and **SpHHumf** |
| SpXHumf_HX | FB_BA_HX [▶ 227] | This function block is used to calculate the set value for the absolute room humidity (humidification). For calculating the value, the room temperature, the set value for the relative humidity **SpHHumf** and the barometric air pressure are required. |
| SpXHumf | FB_BACnetAVDisplay [▶ 69] | Output of the calculated absolute room humidity set value for humidification (g/kg). |
| SpXDehumf_HX | FB_BA_HX [▶ 227] | This function block is used to calculate the set value for the absolute room humidity (dehumidification). For calculating the value, the room temperature, the set value for the relative humidity **SpHDehumf** and the barometric air pressure are required. |
| SpXDehumf | FB_BACnetAVDisplay [▶ 69] | Output of the calculated absolute room humidity set value for dehumidification (g/kg). |

### Version history

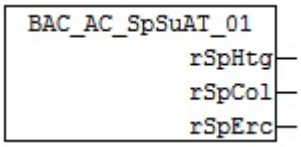
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.11 BAC_AC_SpSuAT_01

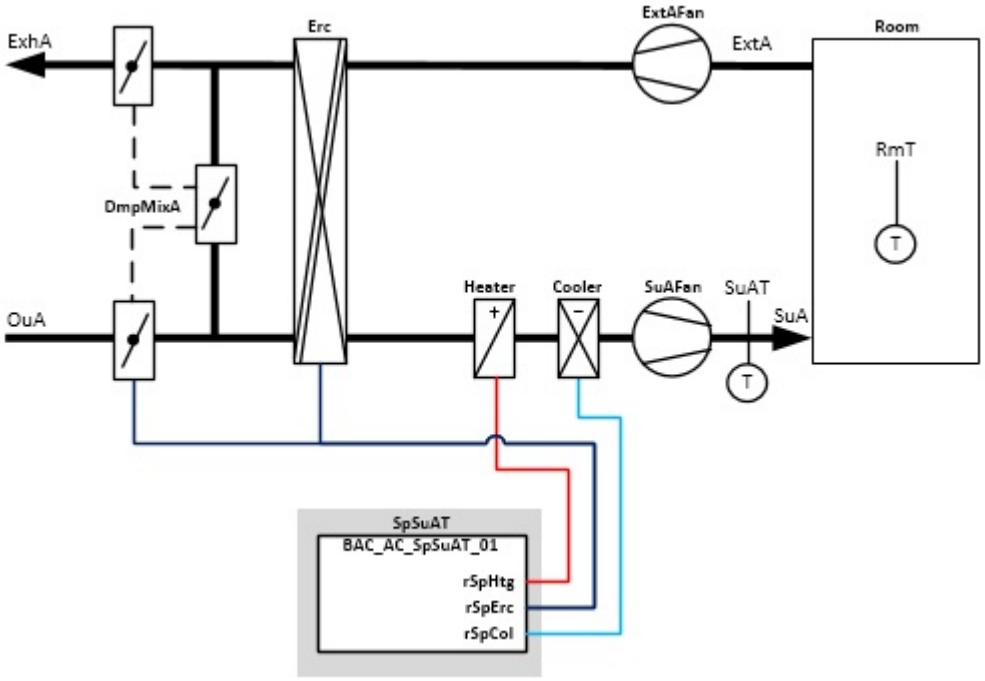
### Application

Set value program for supply air temperature control with an supply air temperature set value, including summer/winter compensation via a characteristic curve.

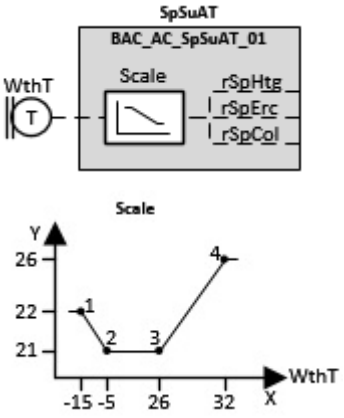
### Interface



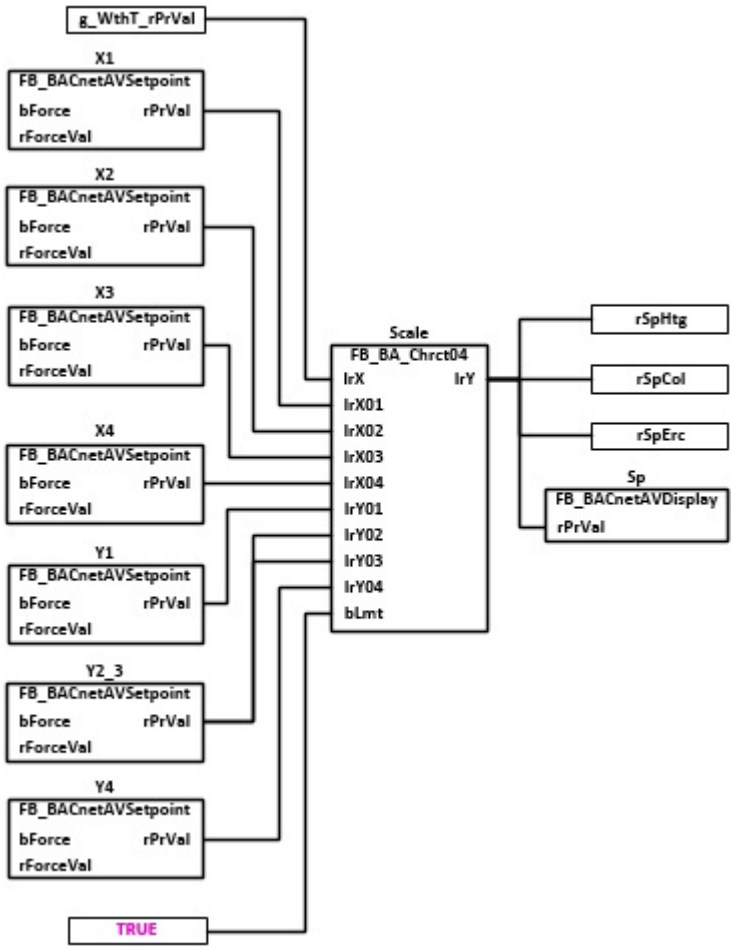

### Plant diagram 01




### Plant diagram 02



### Block diagram




### VAR_OUTPUT

```
rSpHtg      : REAL;
rSpErc      : REAL;
rSpCol      : REAL;
```

**rSpHtg:** Calculated set value of the supply air temperature for heating

**rSpErc:** Calculated set value of the supply air temperature for energy recovery

**rSpCol:** Calculated set value of the supply air temperature for cooling

### Program description

| Instance | Type | Task |
|---|---|---|
| X1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 |
| X2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 |
| X3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points X3 |
| X4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X4 |
| Y1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 |
| Y2_3 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation points Y2/Y3 |

| Instance | Type | Task |
|---|---|---|
| Y4 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y4 |
| Scale | FB_BA_Chrct04 [▶ 172] | The function block calculates the set value characteristic curve for the current room temperature, depending on the outside temperature. |
| Sp | FB_BACnetAVDisplay [▶ 69] | Output of the calculated, simple room temperature set value. It is output at the outputs **rSpHtg**, **rSpCol** and **rSpErc**. |

### Version history

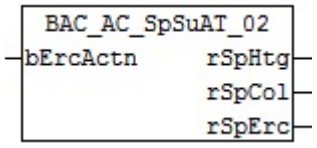
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.80.12 BAC_AC_SpSuAT_02

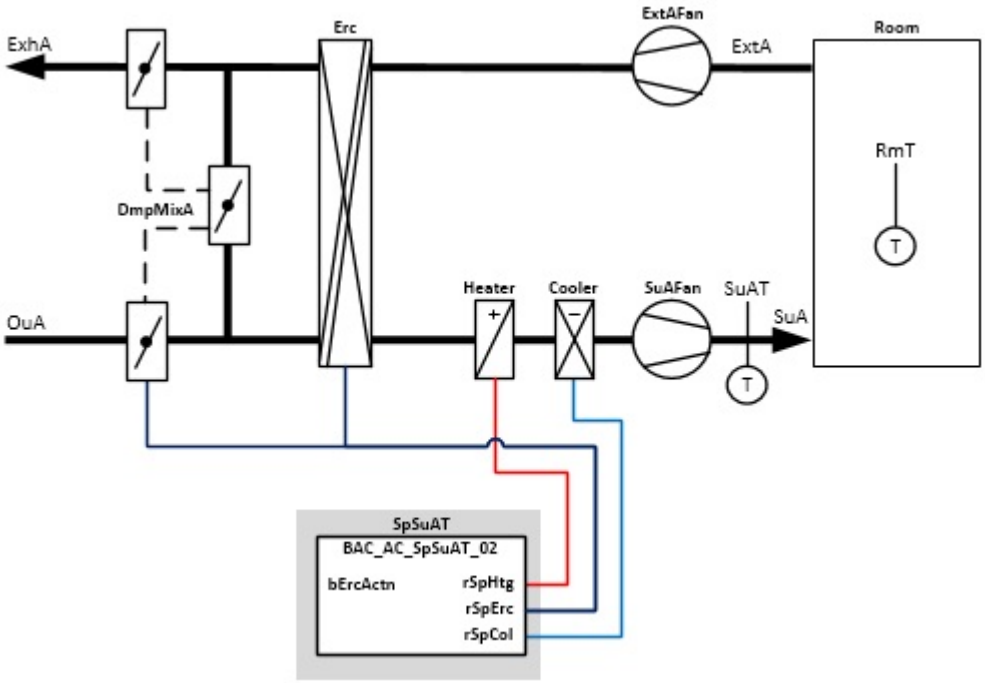
### Application

Set value program for supply air temperature control with separate supply air set values for heating, cooling and energy recovery, including two separate characteristic set value curves for summer and winter compensation.

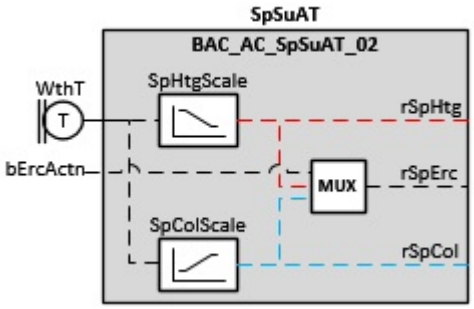
### Interface



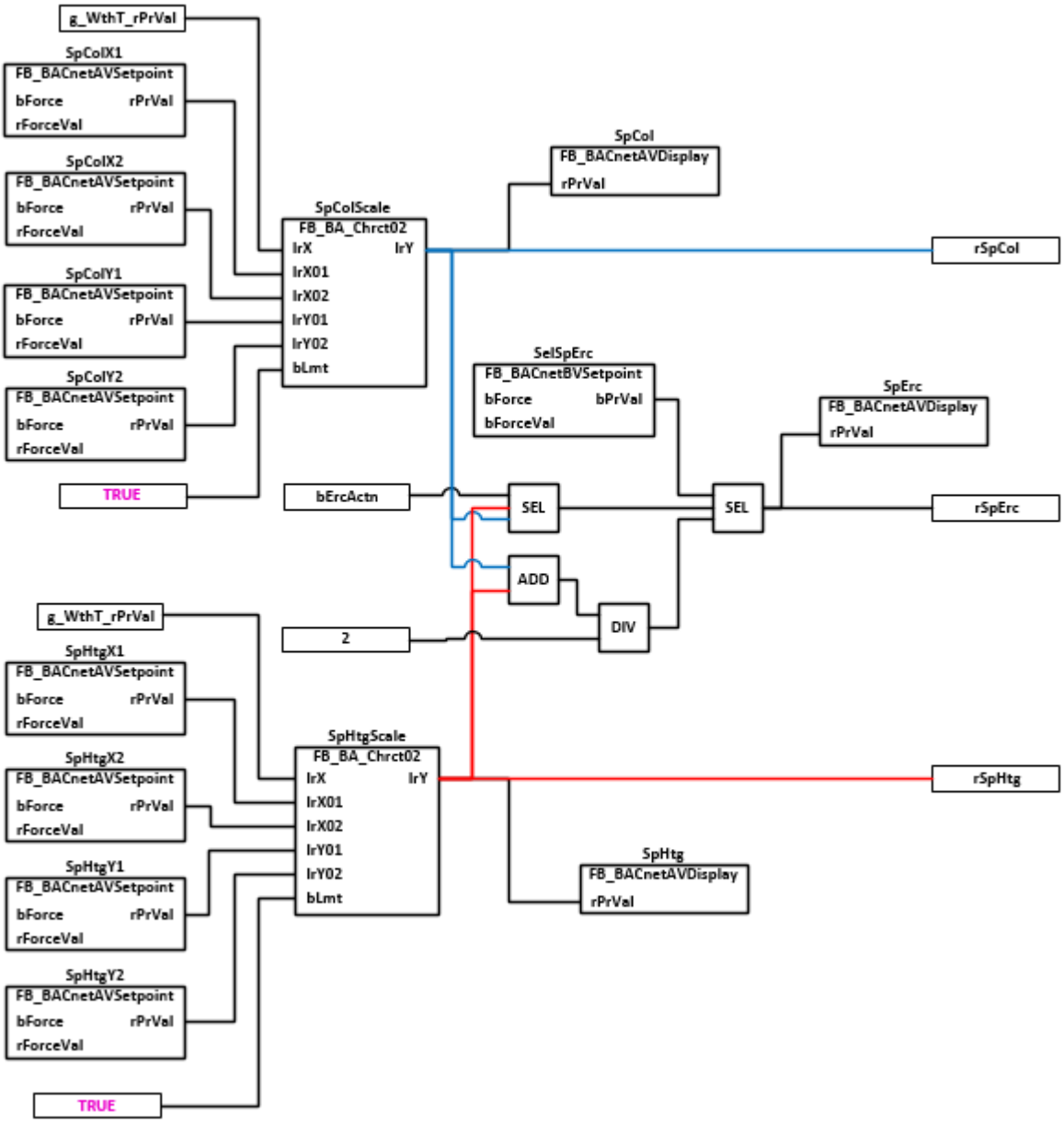

### Plant diagram 01



### Plant diagram 02




### Block diagram




### VAR_OUTPUT

```
bErcActn     : BOOL;
```

**bErcActn:** Input variable to which the direction of action for the energy recovery is applied. The set value for the energy recovery is determined depending on the direction of action.
TRUE = direct = cooling; FALSE = reverse = heating

### VAR_OUTPUT

```
rSpHtg       : REAL;
rSpErc       : REAL;
rSpCol       : REAL;
```

**rSpHtg:** Calculated set value of the supply air temperature for heating

**rSpErc:** Calculated set value of the supply air temperature for energy recovery

**rSpCol:** Calculated set value of the supply air temperature for cooling

### Program description

| Instance | Type | Task |
|---|---|---|
| SpColX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 of the summer compensation for the cooling set value. |
| SpColX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 of the summer compensation for the cooling set value. |
| SpColY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 of the summer compensation for the cooling set value. SpColY1 is the basic cooling set value. |
| SpColY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y2 of the summer compensation for the cooling set value. |
| SpColScale | FB_BA_Chrct02 [▶ 171] | The function block determines the summer compensation value for the cooling set value. |
| SpHtgX1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X1 of the winter compensation for the heating set value. |
| SpHtgX2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point X2 of the winter compensation for the heating set value. |
| SpHtgY1 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y1 of the winter compensation for the heating set value. SpHtgY1 is the basic heating set value. |
| SpHtgY2 | FB_BACnetAVSetpoint [▶ 69] | Input of the value for interpolation point Y2 of the winter compensation for the heating set value. |
| SpHtgScale | FB_BA_Chrct02 [▶ 171] | The function block determines the winter compensation for the heating set value. |
| SpSuVis | FB_BA_SpSupVis [▶ 243] | The function block is used to control the two interpolation points SpHtgY1 and SpColY1. |
| SelSpErc | FB_BACnetBV1204 | Selection strategy set value energy recovery. **TRUE:** Mean value of **PrSpHtg** and **PrSpCol**; **FALSE:** Depends on the direction of action, defined by input **bErcActn** |
| | SEL, ADD, DIV, SEL | The result of this network supplies two set values for energy recovery. Which value is used depends on the energy recovery set value strategy, see *SelSpErc* |
| SpHtg | FB_BACnetAVDisplay [▶ 69] | Output of the calculated supply air temperature heating set value |
| SpCol | FB_BACnetAVDisplay [▶ 69] | Output of the calculated supply air temperature cooling set value |
| SpErc | FB_BACnetAVDisplay [▶ 69] | Output of the calculated supply air temperature set value for energy recovery |

### Version history

| Version number | Comments |
|---|---|
| 1.0.1 | First release |

**Also see about this**

FB_BACnetBVSetpoint [▶ 96]

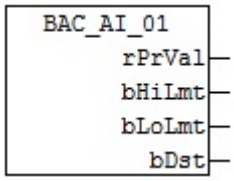
# 9.81 BACnet objects

## 9.81.1 BAC_AI_01

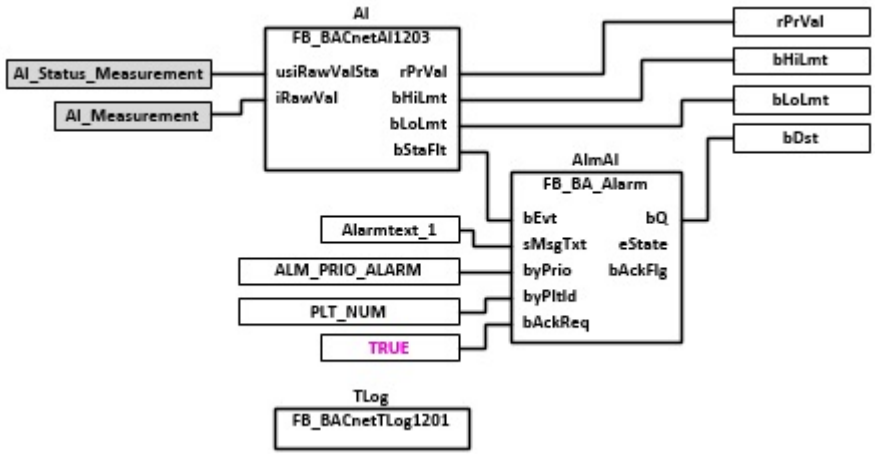
### Functional description

The template logs an analog input value from a Bus Terminal (see **I/O linking**) and converts it to a real process value. In addition, the **status flag Fault** of the **AI** object is evaluated and detected and processed by the alarm function block **AlmAI**.
The template is used so that an analog input signal (e.g. measured value) can be processed in the program as a real process value.

### Interface




### Block diagram




### VAR_OUTPUT

```
rPrVal    : REAL;
bHiLmt    : BOOL;
bLoLmt    : BOOL;
bDst      : BOOL;
```

**rPrVal:** current value of the analog input object.

**bHiLim:** message upper limit reached.

**bLoLmt:** message lower limit reached.

**bDst:** indicates the state of the status flag "Fault" of the analog input object. Once the fault has been rectified, the message may have to be acknowledged. This depends on the parameterization of input **bAckReq** of the function block **AlmAI**.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| AI | FB_BACnetAI1203 [▶ 49] | | Analog input object for displaying the measured value |
| AlmAI | FB_BA_Alarm [▶ 179] | | Logging and further processing of the status flag Fault of the measured value<br>Actions to be taken after the input Fault status flag was triggered can be parameterized in the template at function block **AlmAI**. |
| TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the measured value of the AI object |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| AI_Status_Measurement | USINT | | Input | Analog input - status byte Bus Terminal |
| AI_Measurement | INT | | Input | Analog input - measured value Bus Terminal |

### Version history

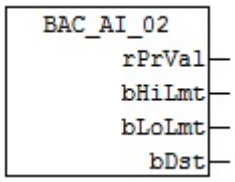
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.2 BAC_AI_02

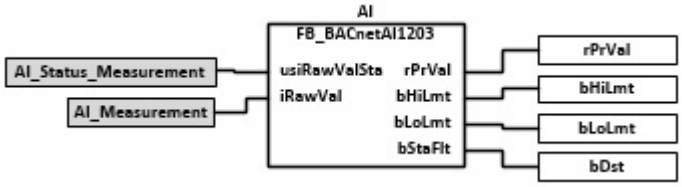
### Functional description

The template logs an analog input value from a Bus Terminal (see **I/O link**) and converts it into a real process value.
The template is used so that an analog input signal (e.g. measured value) can be processed in the program as a real process value.

### Interface




### Block diagram




### VAR_OUTPUT

```
rPrVal    : REAL;
bHiLmt    : BOOL;
bLoLmt    : BOOL;
bDst      : BOOL;
```

**rPrVal:** current value of the analog input object.

**bHiLim:** message upper limit reached.

**bLoLmt:** message lower limit reached.

**bDst:** indicates the state of the status flag "Fault" of the analog input object. Once the fault has been rectified, the message may have to be acknowledged. This depends on the parameterization of input **bAckReq** of the function block **AlmAI**.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| AI | FB_BACnetAI1203 [▶ 49] | | Analog input object for displaying the measured value |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| AI_Status_Measurement | USINT | | Input | Analog input - status byte Bus Terminal |
| AI_Measurement | INT | | Input | Analog input - measured value Bus Terminal |

### Version history

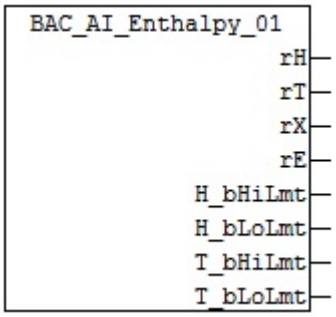
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.3 BAC_AI_Enthalpy_01

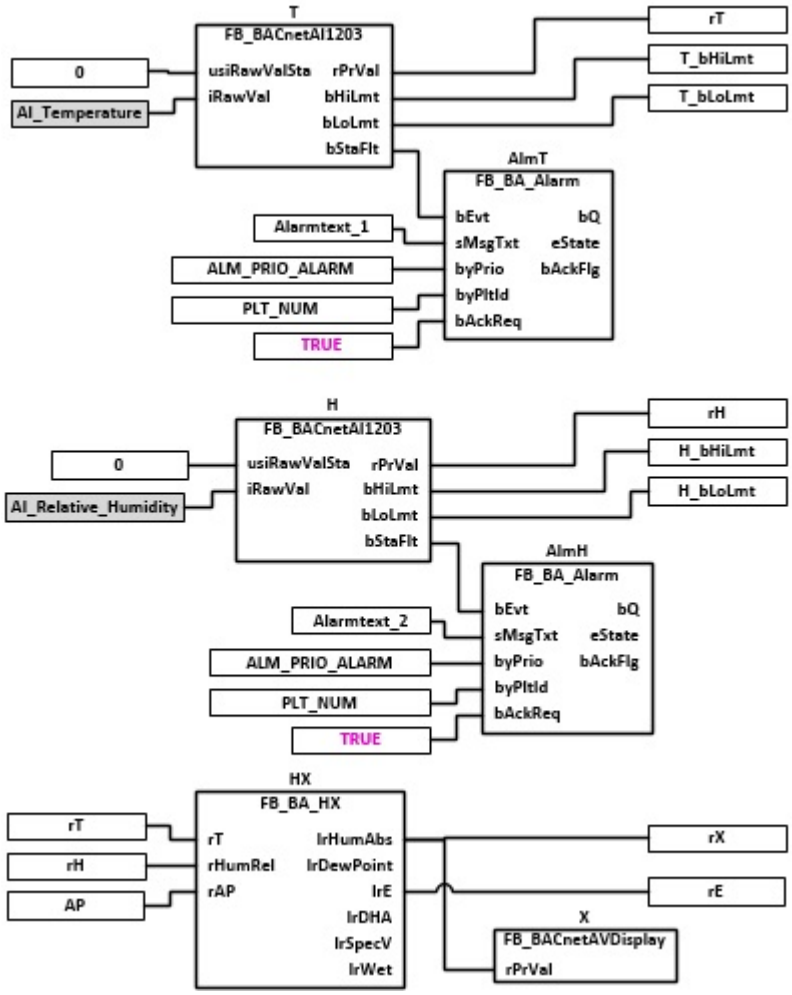
### Functional description

The template logs the two analog input values **temperature** and **relative humidity** of a Bus Terminal (see IO linking) and converts them to real process values.
The function block **HX** is used to calculate the **dew point temperature**, the specific **enthalpy**, the **wet bulb temperature** and the **absolute humidity**.

### Interface



### Block diagram




### VAR_OUTPUT

```
rH        : REAL;
rT        : REAL;
rX        : REAL;
rE        : REAL;
H_bHiLmt  : BOOL;
H_bLoLmt  : BOOL;
T_bHiLmt  : BOOL;
T_bLoLmt  : BOOL;
```

**rH:** current value of the analog input object **H - relative humidity [%]**

**rT:** current value of the analog input object **T - temperature [°C]**

**rX:** current value of the analog value object **X - absolute humidity g water in kg of dry air [g/kg]**

**rE:** calculated value **enthalpy [kJ/kg]**

**H_bHiLmt:** relative humidity High Limit

**H_bLoLmt:** relative humidity Low Limit

**T_bHiLmt:** temperature High Limit

**T_bLoLmt:** temperature Low Limit

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| T | FB_BACnetAI1203 [▶ 49] | | Analog input object for temperature display |
| AlmT | FB_BA_Alarm [▶ 179] | | Logging and further processing of the status flag Fault of the temperature<br>Actions to be taken after the input Fault status flag was triggered can be parameterized in the template at function block **AlmT**. |
| T_TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the measured value of the AI object **T** |
| H | FB_BACnetAI1203 [▶ 49] | | Analog input object for relative humidity display |
| AlmH | FB_BA_Alarm [▶ 179] | | Logging and further processing of the status flag Fault of the relative humidity<br>Actions to be taken after the input Fault status flag was triggered can be parameterized in the template at function block **AlmH**. |
| H_TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the measured value of the AI object **H** |
| HX | FB_BA_HX [▶ 227] | | 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 air pressure are required for the calculation of these parameters. Enthalpy is a measure of the energy of a thermodynamic system. |
| X | FB_BACnetAVDisplay [▶ 69] | | Analog value object for absolute humidity display |
| X_TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the measured value of the AV object **X** |

### IO linking

Variables for linking with the terminals

| Parameter | Type | op-tional | Process im-age | |
|---|---|---|---|---|
| AI_Temperature | INT | | Input | Analog input - temperature |
| AI_Relative_Humidity | INT | | Input | Analog input - relative humidity |

### Version history

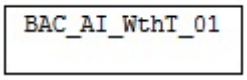
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.4 BAC_AI_WthT_01

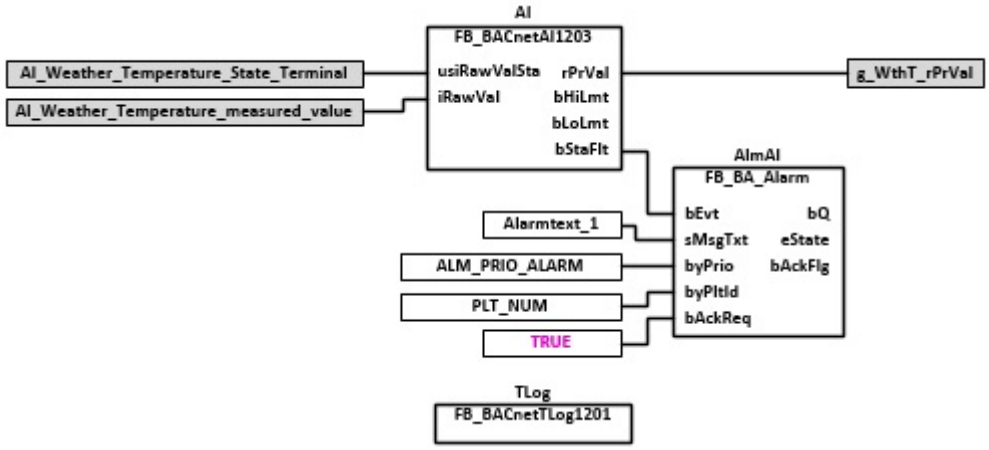
### Functional description

The template maps the weather temperature. It logs the analog input value from a Bus Terminal (see **I/O linking**) and converts it to a real process value. This value is then written to the global variable g_WthT_rPrVal [▶ 357].
In addition, the **status flag Fault** of the **AI** object is evaluated and detected and processed by the alarm function block **AlmAI**.

### Interface




### Block diagram




### VAR CONSTANT

```
PLT_NUM    : BYTE := 0;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| AI | FB_BACnetAI1203 [▶ 49] | | Analog input object for displaying the measured outside temperature |
| AlmAI | FB_BA_Alarm [▶ 179] | | Logging and further processing of the status flag Fault of the measured value outside temperature<br>Actions to be taken after the input Fault status flag was triggered can be parameterized in the template at function block **AlmAI**. |
| TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the measured value of the AI object |

### IO linking

In the XML description associated with the template, variables with the ID **Input** or **Output** are declared in the **Parameter** section. These parameters can be linked with the process image of the input and output level in the PLC in the Project Builder or via the Excel import interface.

| Parameter | Type | Instance | Type | Process image | |
|---|---|---|---|---|---|
| AI_Weather_Temperature_State_Terminal | USINT | AI | FB_BACnetAI1203 [▶ 49] | Input | Analog input - weather temperature - status Bus Terminal |
| AI_Weather_Temperature_measured_value | INT | AI | FB_BACnetAI1203 [▶ 49] | Input | Analog input - weather temperature - measured value |

### Version history

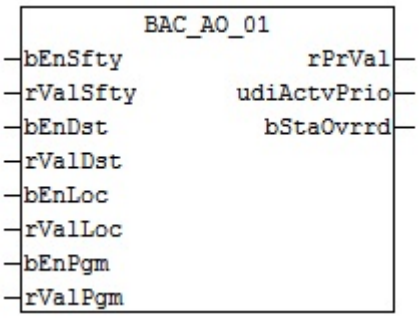
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.5 BAC_AO_01

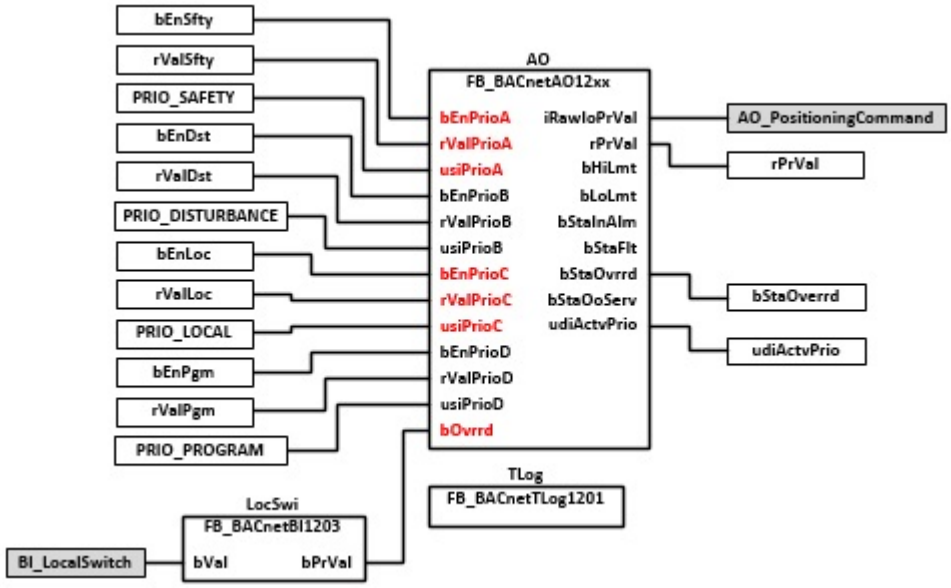
### Functional description

The template determines the current control value from several, prioritised set values (priority matrix, template inputs) and transfers the control value to a Bus Terminal (see **IO linking**).
In addition, the manual intervention is logged via a digital input **LocSwi**.

### Interface



### Block diagram




### VAR_INPUT

```
bEnSfty     : BOOL;
rValSfty    : REAL;
bEnDst      : BOOL;
rValDst     : REAL;
bEnLoc      : BOOL;
rValLoc     : REAL;
bEnPgm      : BOOL;
rValPgm     : REAL;
```

**bEnSfty:** Safety priority enable

**rValSfty:** Analog value safety priority

**bEnDst:** Enable fault priority.

**rValDst:** Analog value fault priority.

**bEnLoc:** Enable priority manual intervention

**rValLoc:** Analog value priority manual intervention

**bEnPgm:** Enable program priority

**rValPgm:** Analog value program priority

### VAR_OUTPUT

```
rPrVal         : REAL;
udiActvPrio    : UDINT;
bStaOverrd     : BOOL;
```

**rPrVal:** current value of the analog output object.

**udiActvPrio:** display of the current priority of the AO object

**bStaOverrd:** indicates that the AO object is overridden by a local mechanism, see **LocSwi**

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]

The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task | | |
|---|---|---|---|---|---|
| AO | FB_BACnetAO1203 [▶ 53] | | AO object for outputting the control value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **rValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **rValPgm** |
| LocSwi | FB_BACnetBI1203 [▶ 72] | | Digital input object for displaying the switch for the local override of the control value for the AO object | | |
| TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the control value for the AO object | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_LocalSwitch | BOOL | | Input | Digital input - switch manual - message - manual/auto |
| AO_PositioningCommand | INT | | Output | Analog output - control command |

### Version history

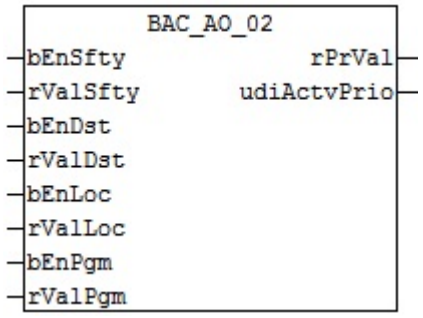
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.6 BAC_AO_02

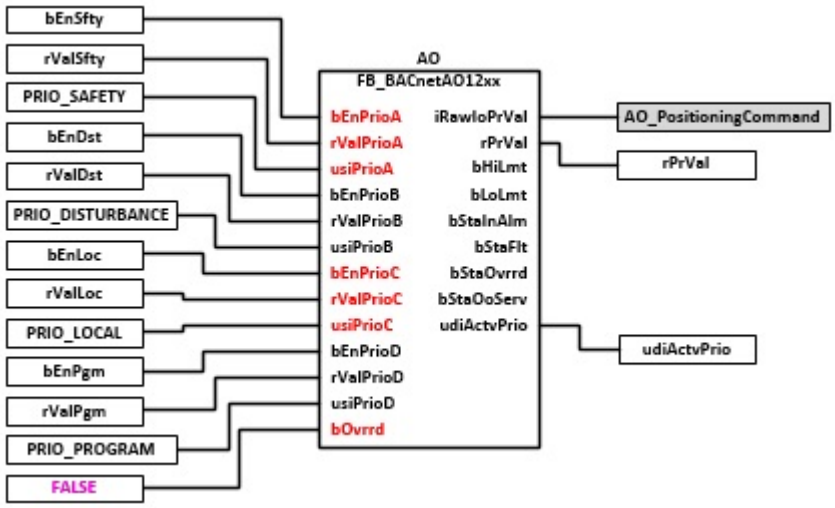
### Functional description

The template determines the current control value from several, prioritised set values (priority matrix, template inputs) and transfers the control value to a Bus Terminal (see **IO linking**).

### Interface




### Block diagram




### VAR_INPUT

```
bEnSfty    : BOOL;
rValSfty   : REAL;
bEnDst     : BOOL;
rValDst    : REAL;
bEnLoc     : BOOL;
rValLoc    : REAL;
bEnPgm     : BOOL;
rValPgm    : REAL;
```

**bEnSfty:** Safety priority enable

**rValSfty:** Analog value safety priority

**bEnDst:** Enable fault priority.

**rValDst:** Analog value fault priority.

**bEnLoc:** Enable priority manual intervention

**rValLoc:** Analog value priority manual intervention

**bEnPgm:** Enable program priority

**rValPgm:** Analog value program priority

### VAR_OUTPUT

```
rPrVal    : REAL;
udiActvPrio   : UDINT;
```

**rPrVal:** current value of the analog output object.

**udiActvPrio:** display of the current priority of the AO object

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task | | |
|---|---|---|---|---|---|
| AO | FB_BACnetAO1203 [▶ 53] | | AO object for outputting the control value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **rValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **rValPgm** |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| AO_PositioningCommand | INT | | Output | Analog output - control command |

### Version history

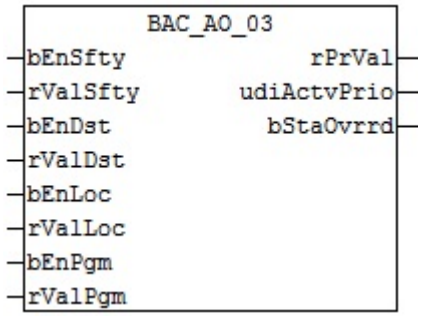
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.7 BAC_AO_03

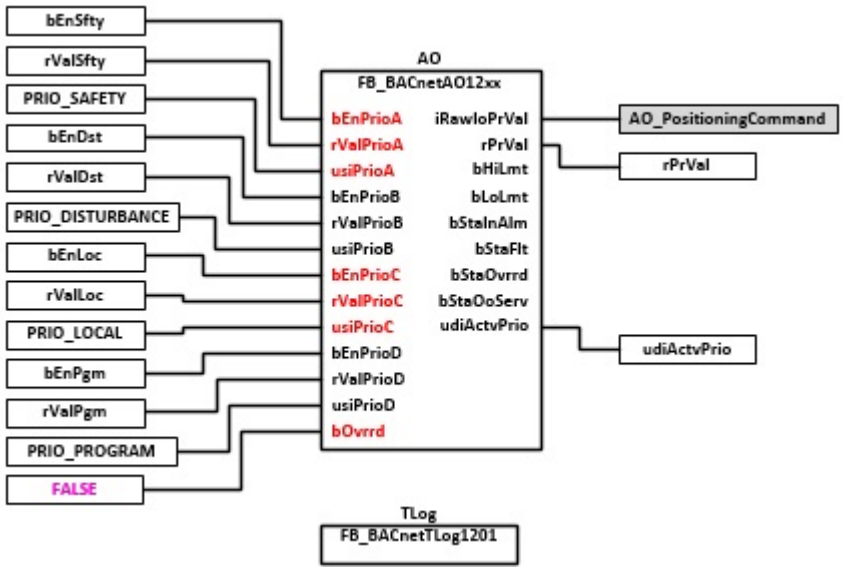
### Functional description

The template determines the current control value from several, prioritised set values (priority matrix, template inputs) and transfers the control value to a Bus Terminal (see **IO linking**).

### Interface




### Block diagram




### VAR_INPUT

```
bEnSfty     : BOOL;
rValSfty    : REAL;
bEnDst      : BOOL;
rValDst     : REAL;
bEnLoc      : BOOL;
rValLoc     : REAL;
bEnPgm      : BOOL;
rValPgm     : REAL;
```

**bEnSfty:** Safety priority enable

**rValSfty:** Analog value safety priority

**bEnDst:** Enable fault priority.

**rValDst:** Analog value fault priority.

**bEnLoc:** Enable priority manual intervention

**rValLoc:** Analog value priority manual intervention

**bEnPgm:** Enable program priority

**rValPgm:** Analog value program priority

### VAR_OUTPUT

```
rPrVal         : REAL;
udiActvPrio    : UDINT;
```

**rPrVal:** current value of the analog output object.

**udiActvPrio:** display of the current priority of the AO object

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task | | |
|---|---|---|---|---|---|
| AO | FB_BACnetAO1203 [▶ 53] | | AO object for outputting the control value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **rValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **rValPgm** |
| TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the control value for the AO object | | |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| AO_PositioningCommand | INT | | Output | Analog output - control command |

### Version history

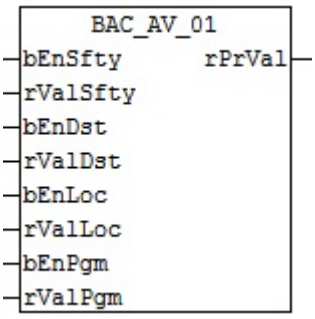
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.81.8 BAC_AV_01

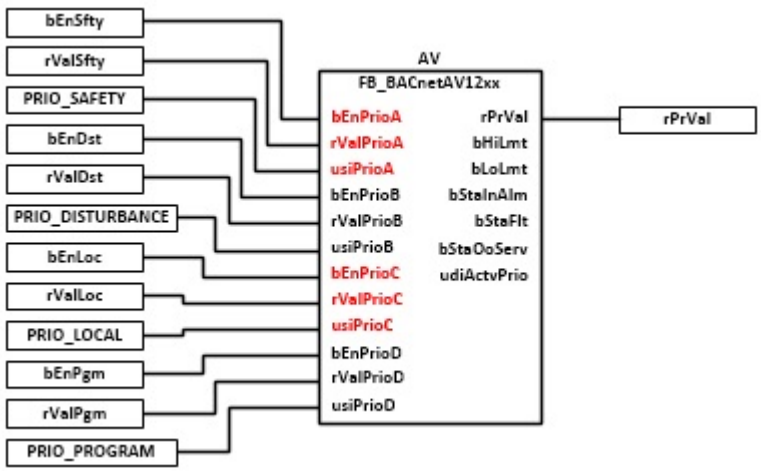
### Functional description

The template forms a REAL process value from several, prioritised set values (priority matrix, template inputs).

### Interface




### Block diagram




### VAR_INPUT

```
bEnSfty    : BOOL;
rValSfty   : REAL;
bEnDst     : BOOL;
rValDst    : REAL;
bEnLoc     : BOOL;
rValLoc    : REAL;
bEnPgm     : BOOL;
rValPgm    : REAL;
```

**bEnSfty:** Safety priority enable

**rValSfty:** Analog value safety priority

**bEnDst:** Enable fault priority.

**rValDst:** Analog value fault priority.

**bEnLoc:** Enable priority manual intervention

**rValLoc:** Analog value priority manual intervention

**bEnPgm:** Enable program priority

**rValPgm:** Analog value program priority

### VAR_OUTPUT

```
rPrVal    : REAL;
```

**rPrVal:** current value of the Analog Value object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| AV | FB_BACnetAV1204 [▶ 67] | | AV object for outputting the control value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **rValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **rValPgm** |

### Version history

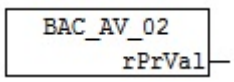
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.9 BAC_AV_02

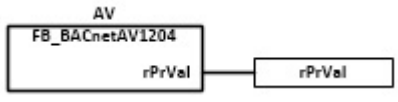
### Functional description

The template maps a REAL process value in BACnet. The template is used for enabling input of a set value or parameter.

### Interface




### Block diagram




### VAR_OUTPUT

```
rPrVal    : REAL;
```

**rPrVal:** current value of the Analog Value object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| AV | FB_BACnetAV1204 [▶ 67] | | AV object for outputting the control value |

### Version history

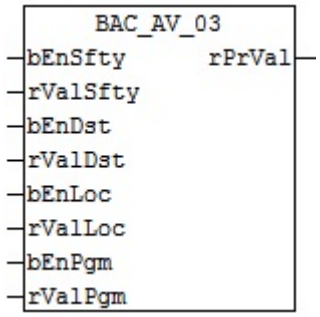
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.10 BAC_AV_03

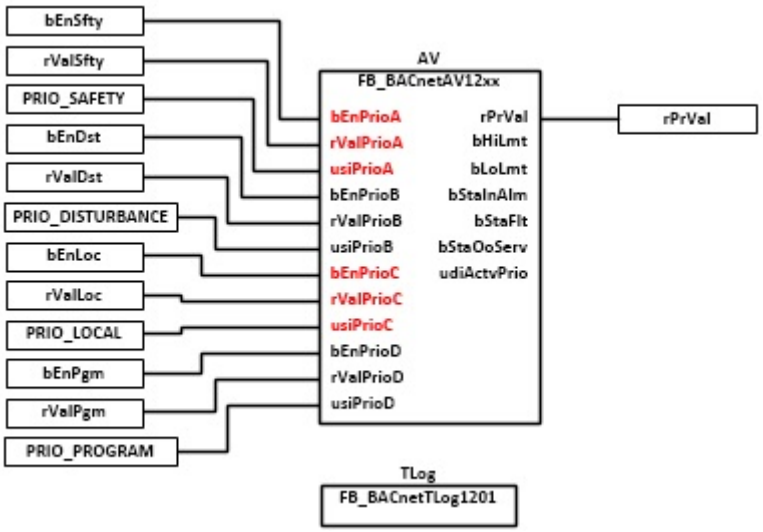
### Functional description

The template forms a REAL process value from several, prioritised set values (priority matrix, template inputs).
The process value is logged by a TrendLog object.

### Interface



### Block diagram




### VAR_INPUT

```
bEnSfty    : BOOL;
rValSfty   : REAL;
bEnDst     : BOOL;
rValDst    : REAL;
bEnLoc     : BOOL;
rValLoc    : REAL;
bEnPgm     : BOOL;
rValPgm    : REAL;
```

**bEnSfty:** Safety priority enable

**rValSfty:** Analog value safety priority

**bEnDst:** Enable fault priority.

**rValDst:** Analog value fault priority.

**bEnLoc:** Enable priority manual intervention

**rValLoc:** Analog value priority manual intervention

**bEnPgm:** Enable program priority

**rValPgm:** Analog value program priority

### VAR_OUTPUT

```
rPrVal    : REAL;
```

**rPrVal:** current value of the Analog Value object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| AV | FB_BACnetAV1204 [▶ 67] | | AV object for outputting the control value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **rValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **rValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **rValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **rValPgm** |
| TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the process value of the AV object | | |

### Version history

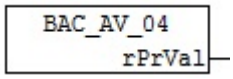
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.11 BAC_AV_04

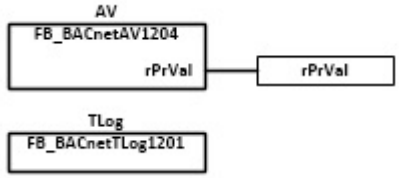
### Functional description

The template maps a REAL process value in BACnet. The template is used for enabling input of a set value or parameter.
The process value is logged by a TrendLog object.

### Interface




### Block diagram




### VAR_OUTPUT

```
rPrVal    : REAL;
```

**rPrVal:** current value of the Analog Value object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| AV | FB_BACnetAV1204 [▶ 67] | | AV object for outputting the control value |
| TLog | FB_BACnetTLog1201 [▶ 135] | | Trend logging of the process value of the AV object |

### Version history

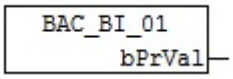
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.12 BAC_BI_01

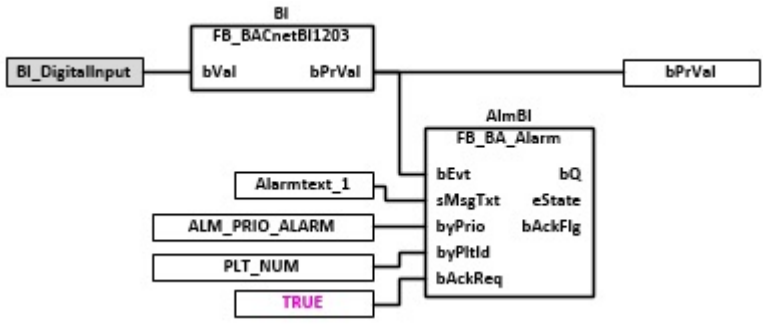
### Functional description

The template logs a binary input value from a Bus Terminal and outputs it as a boolean process value. In addition, the process value **bPrVal** of the **BI** object is evaluated and detected and processed by the alarm function block **AlmBI**.
The template is used so that a binary input signal can be processed in the program as a Boolean process value.

### Interface




### Block diagram



### VAR_OUTPUT

```
bPrVal   : BOOL;
```

**bPrVal:** current value of the binary input object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task |
|---|---|---|---|
| BI | FB_BACnetBI1203 [▶ 72] | | Binary Input Object |
| AlmBI | FB_BA_Alarm [▶ 179] | | Logging and further processing of the process value **bPrVal** |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_DigitalInput | BOOL | | Input | Digital input Bus Terminal |

### Version history

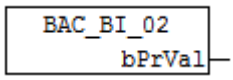
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.13 BAC_BI_02

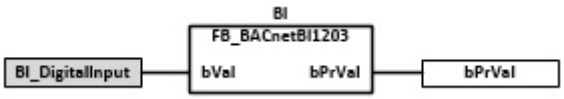
### Functional description

The template logs a binary input value from a Bus Terminal and outputs it as a boolean process value.
The template is used to enable processing of a binary input signal in the program as a boolean process value.

### Interface




### Block diagram




### VAR_OUTPUT

```
bPrVal   : BOOL;
```

**bPrVal:** current value of the binary input object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task |
|---|---|---|---|
| BI | FB_BACnetBI1203 [▶ 72] | | Binary Input Object |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_DigitalInput | BOOL | | Input | Digital input Bus Terminal |

### Version history

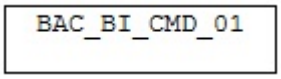
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

# 9.81.14 BAC_BI_CMD_01

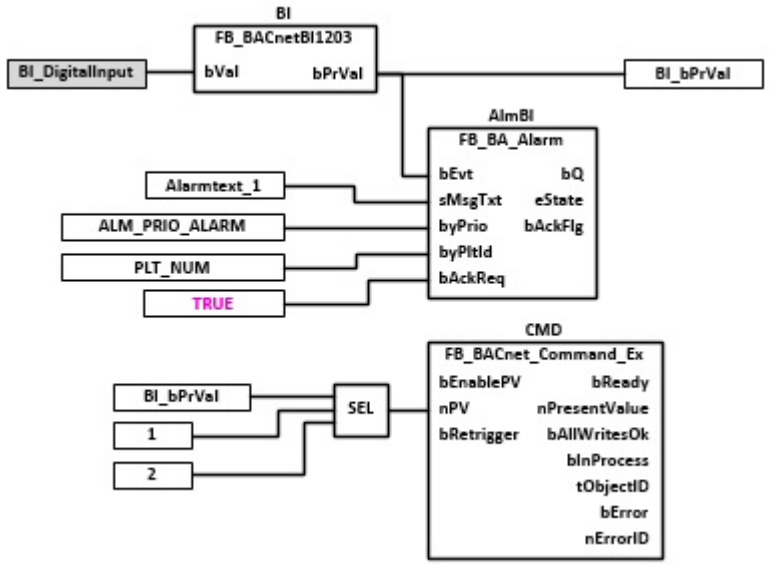
### Functional description

The template captures a binary input value of a Bus Terminal. In addition, the process value **bPrVal** of the **BI** object is evaluated and recorded and processed by the alarm function block **AlmBI**.
Based on the change of state at the **BI** object, specified properties of other objects can be written via the group order object **CMD**.

### Interface




### Block diagram




### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| BI | FB_BACnetBI1203 [▶ 72] | | Binary Input Object |

| Instance | Type | op-tional | Task |
|---|---|---|---|
| AlmBI | FB_BA_Alarm [▶ 179] | | Logging and further processing of the process value **BI_bPrVal** |
| CMD | FB_BACnet_Command_Ex | | Group order object |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process im-age | |
|---|---|---|---|---|
| BI_DigitalInput | BOOL | | Input | Digital input Bus Terminal |

### Version history

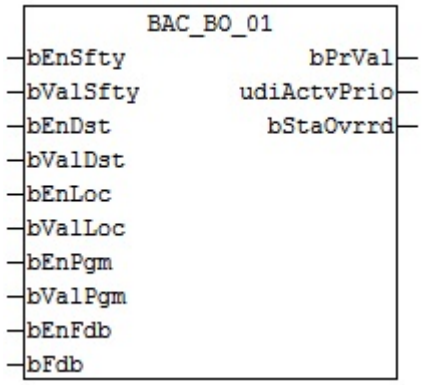
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.81.15 BAC_BO_01

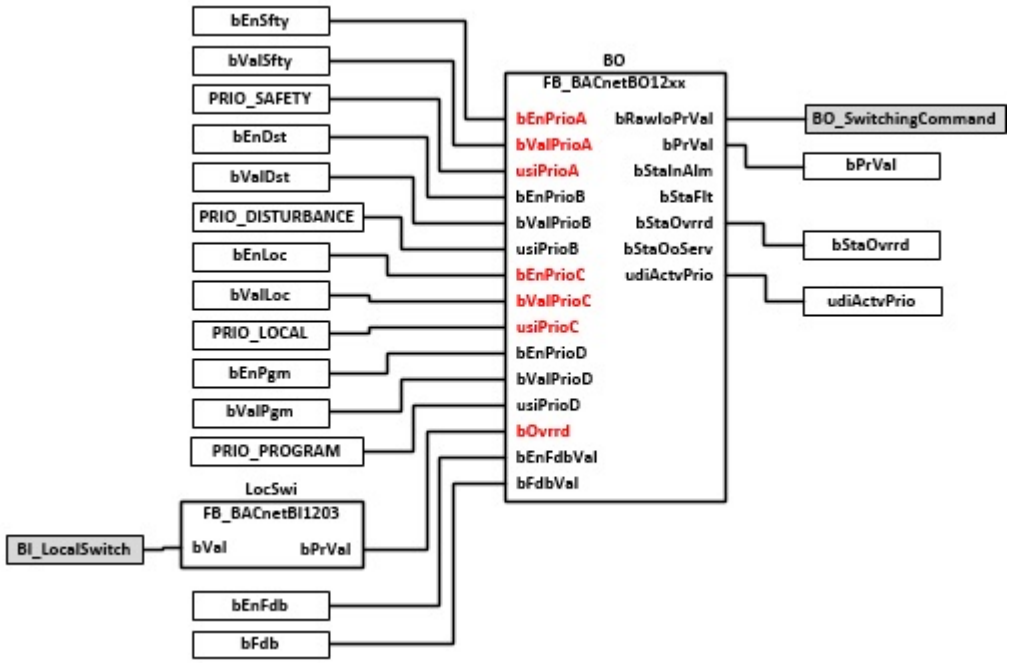
### Functional description

The template determines the current switch value from several prioritized setpoints (priority matrix, inputs of the template) and transmits the switch value to a Bus Terminal (see **I/O link**).
On the one hand, the template records the manual intervention by means of a digital input **LocSwi** and the feedback of the actuator via the two inputs **bEnFdb**/**bFdb** of the template.
The template is used to output and monitor switching commands or pulses (e.g. 1-stage motors/pumps, dampers).

### Interface



### Block diagram




### VAR_INPUT

```
bEnSfty     : BOOL;
bValSfty    : BOOL;
bEnDst      : BOOL;
bValDst     : BOOL;
bEnLoc      : BOOL;
bValLoc     : BOOL;
bEnPgm      : BOOL;
bValPgm     : BOOL;
```

**bEnSfty:** Safety priority enable

**bValSfty:** Digital value safety priority

**bEnDst:** Enable fault priority.

**bValDst:** Digital value fault priority.

**bEnLoc:** Enable priority manual intervention

**bValLoc:** Digital value priority manual intervention

**bEnPgm:** Enable program priority

**bValPgm:** Digital value program priority

### VAR_OUTPUT

```
bPrVal          : BOOL;
udiActvPrio     : UDINT;
bStaOverrd      : BOOL;
```

**bPrVal:** current value of the binary output object.

**udiActvPrio:** display of the current priority of the BO object

**bStaOverrd:** indicates that the BO object is overridden by a local mechanism, see **LocSwi**

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.

The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task | | |
|---|---|---|---|---|---|
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for output of the switching value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **bValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **bValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **bValPgm** |
| LocSwi | FB_BACnetBI1203 [▶ 72] | | Digital input object for displaying the switch for the local override of the switch value for the BO object | | |

### IO linking

In the XML description associated with the template, variables with the ID **Input** or **Output** are declared in the **Parameter** section. These parameters can be linked with the process image of the input and output level in the PLC in the Project Builder or via the Excel import interface.

| Parameter | Type | Instance | Type | Process image |
|---|---|---|---|---|
| LocSwi_bVal | USINT | LocSwi | FB_BACnetBI1203 [▶ 72] | Input |
| BO_bRawIoPrVal | BOOL | BO | FB_BACnetBO1203 [▶ 81] | Output |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process image | |
|---|---|---|---|---|
| BI_LocalSwitch | BOOL | | Input | Digital input - switch manual - message - manual/auto |
| BO_SwitchingCommand | BOOL | | Output | Digital output - switching command - on/off |

### Version history

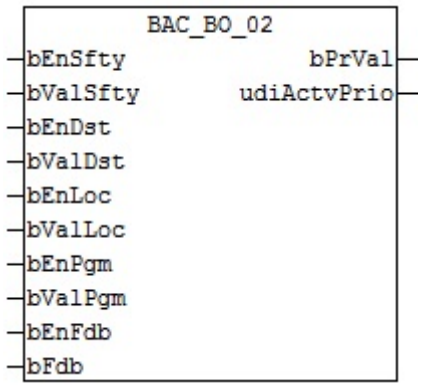
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.16 BAC_BO_02

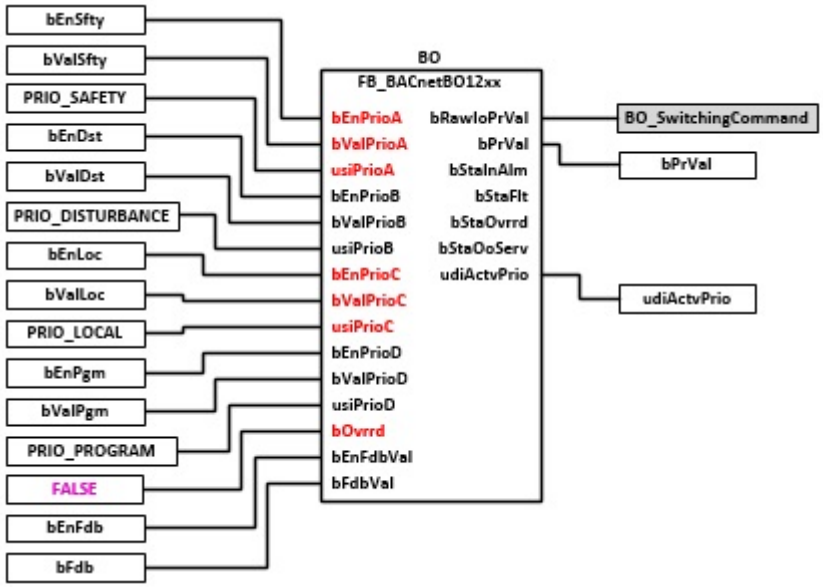
### Functional description

The template determines the current switching value from several prioritized setpoints (priority matrix, inputs of the template) and transmits the switching value to a Bus Terminal (see **I/O link**).
The template records the feedback of the actuator via the two inputs **bEnFdb**/**bFdb** of the template.
The template is used to output and monitor switching commands or pulses (e.g. 1-stage motors/pumps, dampers).

### Interface




### Block diagram




### VAR_INPUT

```
bEnSfty    : BOOL;
bValSfty   : BOOL;
bEnDst     : BOOL;
bValDst    : BOOL;
bEnLoc     : BOOL;
bValLoc    : BOOL;
bEnPgm     : BOOL;
bValPgm    : BOOL;
```

**bEnSfty:** Safety priority enable

**bValSfty:** Digital value safety priority

**bEnDst:** Enable fault priority.

**bValDst:** Digital value fault priority.

**bEnLoc:** Enable priority manual intervention

**bValLoc:** Digital value priority manual intervention

**bEnPgm:** Enable program priority

**bValPgm:** Digital value program priority

### VAR_OUTPUT

```
bPrVal    : BOOL;
udiActvPrio   : UDINT;
```

**bPrVal:** current value of the binary output object.

**udiActvPrio:** display of the current priority of the BO object

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| BO | FB_BACnetBO1203 [▶ 81] | | BO object for output of the switching value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **bValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **bValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **bValPgm** |

### IO linking

Variables for linking with the terminals

| Parameter | Type | optional | Process im-age | |
|---|---|---|---|---|
| BO_SwitchingCommand | BOOL | | Output | Digital output - switching command - on/off |

### Version history

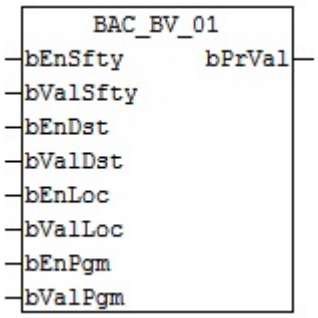
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.17 BAC_BV_01

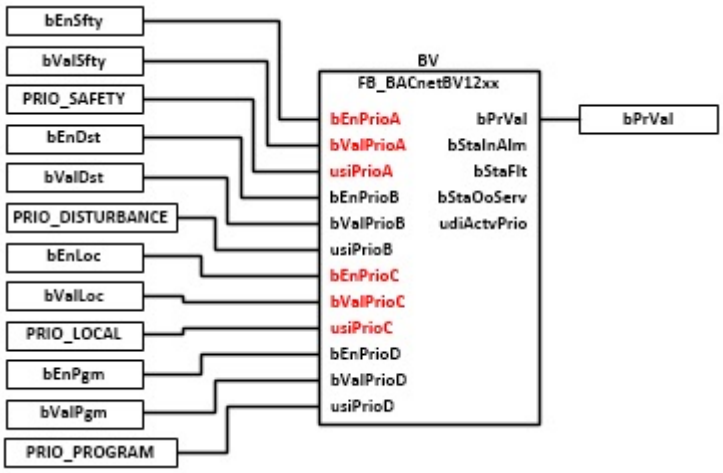
### Functional description

The template determines a process value from several, prioritised set values (priority matrix, template inputs).
The template is used for output of a binary process value or display of the state of a unit.

### Interface




### Block diagram




### VAR_INPUT

```
bEnSfty    : BOOL;
bValSfty   : BOOL;
bEnDst     : BOOL;
bValDst    : BOOL;
bEnLoc     : BOOL;
bValLoc    : BOOL;
bEnPgm     : BOOL;
bValPgm    : BOOL;
```

**bEnSfty:** Safety priority enable

**bValSfty:** Digital value safety priority

**bEnDst:** Enable fault priority.

**bValDst:** Digital value fault priority.

**bEnLoc:** Enable priority manual intervention

**bValLoc:** Digital value priority manual intervention

**bEnPgm:** Enable program priority

**bValPgm:** Digital value program priority

### VAR_OUTPUT

```
bPrVal    : BOOL;
```

**bPrVal:** current value of the binary value object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task | | |
|---|---|---|---|---|---|
| BV | FB_BACnetBV1205 [▶ 90] | | BV object for output or display of a process value | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_SAFETY (1) | Input **bEnSfty** | Input **bValSfty** |
| | | | PRIO_DISTURBANCE (3) | Input **bEnDst** | Input **bValDst** |
| | | | PRIO_LOCAL (8) | Input **bEnLoc** | Input **bValLoc** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **bValPgm** |

### Version history

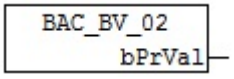
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.18 BAC_BV_02

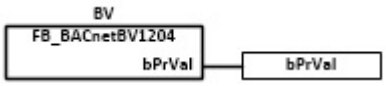
### Functional description

The template is used for output of a binary process value or display of the state of a unit.

### Interface




### Block diagram




### VAR_OUTPUT

```
bPrVal    : BOOL;
```

**bPrVal:** current value of the binary value object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| BV | FB_BACnetBV1206 [▶ 94] | | BV object for output or display of a process value |

### Version history

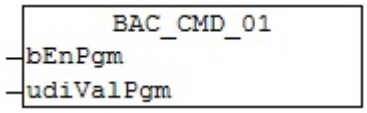
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.19 BAC_CMD_01

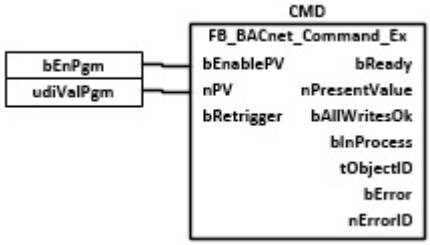
### Functional description

The template captures the two input variables **bEnPgm** / **udiValPgm**. Based on the change of state of the input value **udiValPgm**, the group order object **CMD** can be used to describe specified properties of other objects.

### Interface




### Block diagram




### VAR_INPUT

```
bEnPgm      : BOOL;
udiValPgm   : UDINT;
```

**bEnPgm:** The input variable releases the input value **udiValPgm**, so that it can be written to the group order object.

**udiValPgm:** Input value that is written to the group order object.

### Program description

| Instance | Type | optional | Task |
|---|---|---|---|
| CMD | FB_BACnet_Command_Ex | | Group order object |

### Version history

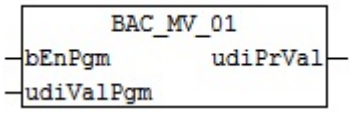
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.81.20 BAC_MV_01

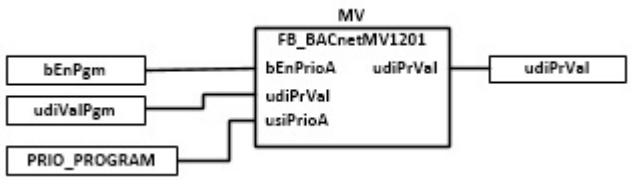
### Functional description

The template determines a multistate process value from a simple prioritisation.
It is used to operate and monitor a multistate value.

### Interface




### Block diagram



### VAR_INPUT

```
bEnPgm      : BOOL;
udiValPgm   : UDINT;
```

**bEnPgm:** Enable program priority

**udiValPgm:** Multistate value program priority

### VAR_OUTPUT

```
udiPrVal    : UDINT;
```

**udiPrVal:** current value of the Multistate Value object.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task | | |
|---|---|---|---|---|---|
| MV | FB_BACnetMV1201 [▶ 122] | | MV object for output of the multistate value<br>see "Extracted nested table 84" | | |
| | | | **Priority:** | **Enable** | **Value** |
| | | | PRIO_PROGRAM (15) | Input **bEnPgm** | Input **udiValPgm** |

### Version history

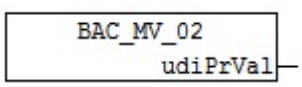
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.21 BAC_MV_02

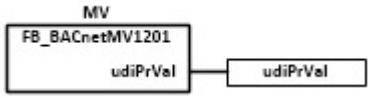
### Functional description

The template maps a multistate process value.
It is used to operate and monitor a multistate value.

### Interface



### Block diagram




### VAR_OUTPUT

```
udiPrVal    : UDINT;
```

**udiPrVal:** current value of the Multistate Value object.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| MV | FB_BACnetMV1201 [▶ 122] | | MV object for output of the multistate value |

### Version history

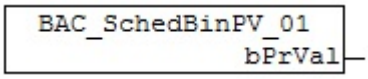
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.22 BAC_SchedBinPV_01

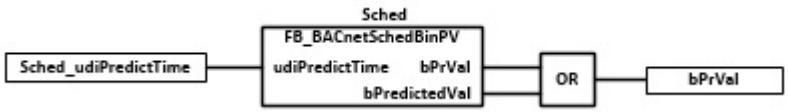
### Functional description

The template includes a BACnet scheduler of type binary present value. The PLC output variable is of type BOOL. In addition, the function block contains the function "Pre-calculating switch-on/switch-off time"
The object is parameterized either from BACnet or via comment lines in the PLC declaration part.

### Interface



### Block diagram




### VAR_OUTPUT

```
bPrVal    : BOOL;
```

**bPrVal:** Output value of the schedule depending on the currently selected date/time and the schedule entries.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| Sched | FB_BACnetSchedBinPV [▶ 132] | | BACnet scheduler of output type BOOL |

### Version history

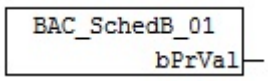
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

## 9.81.23 BAC_SchedB_01

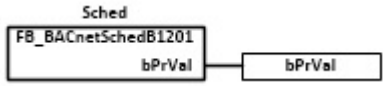
### Functional description

The template contains a BACnet scheduler of the output type BOOL.
The parameterization of the object takes place either from the BACnet or via comment lines in the PLC declaration part.

### Interface




### Block diagram




### VAR_OUTPUT

```
bPrVal    : BOOL;
```

**bPrVal:** Output value of the schedule depending on the currently selected date/time and the schedule entries.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown

in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task |
|---|---|---|---|
| Sched | FB_BACnetSchedB1201 [▶ 131] | | BACnet scheduler of output type BOOL |

### Version history

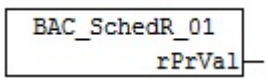
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.24 BAC_SchedR_01

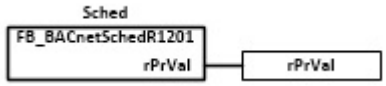
### Functional description

The template contains a BACnet scheduler of the output type REAL.
The parameterization of the object takes place either from the BACnet or via comment lines in the PLC declaration part.

### Interface




### Block diagram




### VAR_OUTPUT

```
rPrVal   : REAL;
```

r**PrVal:** Output value of the schedule depending on the currently selected date/time and the schedule entries.

### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | op-tional | Task |
|---|---|---|---|
| Sched | FB_BACnetSchedR1201 [▶ 133] | | BACnet scheduler of output type REAL |

### Version history

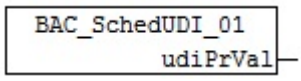
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.25 BAC_SchedUdi_01

### Functional description

The template contains a BACnet scheduler of the output type UDINT.
The parameterization of the object takes place either from the BACnet or via comment lines in the PLC declaration part.

### Interface




### Block diagram


### VAR_OUTPUT

```
udiPrVal    : UDINT;
```

udi**PrVal:** Output value of the schedule depending on the currently selected date/time and the schedule entries.

### VAR CONSTANT

```
PLT_NUM     : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function

block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task |
|---|---|---|---|
| Sched | FB_BACnetSchedUdi1201 [▶ 134] | | BACnet scheduler of output type UDINT |

### Version history

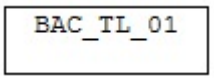
| Version number | Comments |
|---|---|
| 1.0.1 | First release |

## 9.81.26 BAC_TL_01

### Functional description

The template is used for capturing the values of a data source and logging them in the trend log memory. Data sources include all integer, real, Boolean and multistate data, which can be addressed via a BACnet reference address.

### Interface




### VAR CONSTANT

```
PLT_NUM    : BYTE := 1;
```

**PLT_NUM:** all alarms and events of all plant components within a controller are included in a global alarm and event list. The assignment of events and alarms to a plant is defined by the assignment of a PLT_NUM plant number.
The recording and processing of an alarm from an aggregate or a device takes place within the templates by means of the alarm function block FB_BA_Alarm. [▶ 179]
The evaluation of the alarms of a plant, e.g. for the generation of a collective message or for plant shutdown in case of relevant faults, takes place within the template BAC_PltAlm_01 [▶ 365] by means of the function block FB_BA_AlarmPlt. [▶ 183]
The evaluation of different plant events within the templates of a plant, takes place within the template **BAC_PltComnMsg_01** by means of the function block FB_BA_ComnMsg [▶ 197].

**Important! The assignment and evaluation of the alarms and events of a plant can only be done correctly if all templates of a plant have the same plant number!**

The plant number can be assigned in the Project Builder in the parameter menu for the templates or via a column in the Excel import.

### Program description

| Instance | Type | optional | Task |
|---|---|---|---|
| TLog | FB_BACnetTLog1201 [▶ 135] | | BACnet trend log object |

**Version history**

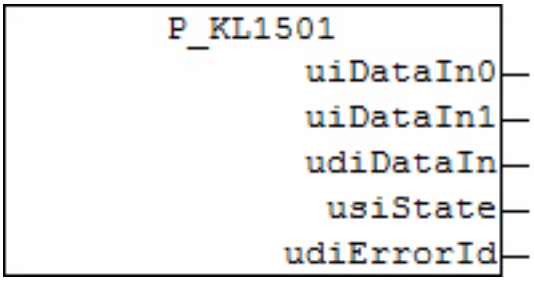
| Version number | Comments |
|---|---|
| 1.0.0.1 | First release |

# 9.82 IO

## 9.82.1 P_KL1501

IO template for parameterizing a KL1501: 1-channel up/down counter. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL1501Config.

**Interface**




**VAR_OUTPUT**

```
uiDataIn0     : UINT;
uiDataIn1     : UINT;
udiDataIn     : UDINT;
usiState      : USINT;
udiErrorId    : UDINT;
```

**uiDataIn0:** Corresponds to the data variable 0 of the input process data. This output is suitable for direct processing in process data mode of the terminal.

**uiDataIn1:** Corresponds to the data variable 1 of the input process data. This output is suitable for direct processing in process data mode of the terminal.

**udiDataIn:** This variable of type UDINT is used for improved evaluation, if a 32-bit counter is selected. It consists of the two variables referred to above, *uiDataIn0* and *uiDataIn1*. *iDataIn0* is used for the low-order part, *iDataIn1* for the high-order part.

**usiState:** Corresponds to the status variable of the input process data. This output is suitable for status assessment in process data mode of the terminal.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

**Parameter**

```
iSetCounterType   : INT;
```

**iSetCounterType:** This input is used to set the counter type. The setting is based on the table below.

| iSetCounterType | Counter type |
|---|---|
| 0 | 32 bit up/down counter |
| 1 | 2 *16-bit forward counter |
| 2 | 32-bit gated counter, gate input Low disables the counter |
| 3 | 32-bit gated counter, gate input High disables the counter |

### Development information

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

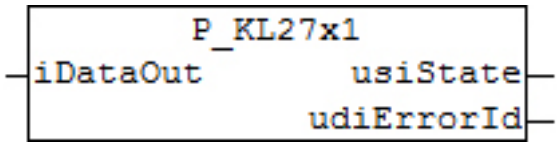
### Version history

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.2 P_KL27x1

IO template for parameterizing a KL2751 / KL2761: Single-channel dimmer terminal. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL27x1Config.

### Interface




### VAR_INPUT

```
iDataOut        : INT;
```

**iDataOut:** output data to the terminal, i.e. light value. Values less than or equal to zero result in the lamp switching off.

### VAR_OUTPUT

```
usiState        : USINT;
udiErrorId      : UDINT;
```

**usiState:** Corresponds to the status variable of the input process data. This output is suitable for status assessment in process data mode of the terminal.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
bSetDimRampAbsolute     : BOOL := FALSE;
iSetRampTime            : INT  := 3;
bSetWatchdogDisable     : BOOL := FALSE;
uiSetWatchdogTimeout    : UINT := 3000;
uiSetTimeoutOnValue     : UINT := 16383;
uiSetTimeoutOffValue    : UINT := 0;
iSetDimmerMode          : INT  := 0;
bSetOnAfterShortCircuit : BOOL := TRUE;
bSetLineFrequency60Hz   : BOOL := FALSE;
```

**bSetDimRampAbsolute:** FALSE: The set ramp time *iSetRampTime* refers to the complete data area (0 - 32767). The smaller the discontinuity, the shorter the ramp time. TRUE: Each switching step, whatever the size, requires the same time, as entered under *iSetRampTime*.

**iSetRampTime:** Ramp time input. The setting is based on the table below.

**bSetWatchdogDisable:** The internal watchdog is disabled.

**uiSetWatchdogTimeout:** Watchdog time as multiple of 10 ms.

**uiSetTimeoutOnValue:** This input specifies the light value that is output when a fieldbus error occurs with current process data > 0. The input range is limited to 32767.

**uiSetTimeoutOffValue:** This input specifies the light value that is output when a fieldbus error occurs with current process data = 0. The input range is limited to 32767.

**iSetDimmerMode:** This input is used to set the dimmer mode. The setting is based on the table below.

**bSetOnAfterShortCircuit:** FALSE: After a short circuit the light remains switched off. TRUE: The light is switched on again after a short circuit.

**bSetLineFrequency60Hz:** FALSE: mains frequency = 50 Hz. TRUE: mains frequency = 60 Hz.

| iSetRampTime | Element |
|---|---|
| 0 | 50 ms |
| 1 | 100 ms |
| 2 | 200 ms |
| 3 | 500 ms |
| 4 | 1 s |
| 5 | 2 s |
| 6 | 5 s |
| 7 | 10 s |
| **iSetDimmerMode** | Element |
| 0 | Automatic detection |
| 1 | Trailing edge phase control |
| 2 | Leading edge phase control |
| 3 | Rectifier mode, positive (positive half-wave with leading edge phase control) |
| 4 | Rectifier mode, negative (negative half-wave with leading edge phase control) |

### Development information

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

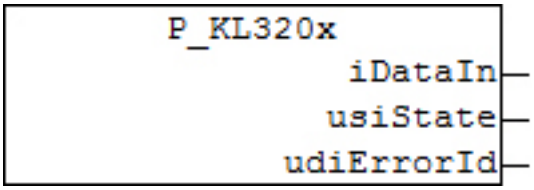
### Version history

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.3 P_KL320x

IO template for parameterizing a KL3201 or KL3202: Input terminal for resistance sensors. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL320xConfig.

### Interface



### VAR_OUTPUT

```
iDataIn     : INT;
usiState    : USINT;
udiErrorId  : UDINT;
```

**iDataIn:** Corresponds to the data variable of the input process data. This output is suitable for direct processing in process data mode of the terminal.

**usiState:** Corresponds to the status variable of the input process data. This output is suitable for status assessment in process data mode of the terminal.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
iSetSensorType  : INT;
```

**iSetSensorType:** This input is used to set the sensor. The setting is based on the table below.

| iSetSensorType | Element |
|---|---|
| 0 | PT100 |
| 1 | NI100 |
| 2 | PT1000 |
| 3 | PT500 |
| 4 | PT500 |
| 5 | NI1000 |
| 6 | NI120 |
| 7 | Output 10.0 Ω – 5000.0 Ω |
| 8 | Output 10.0 Ω – 1200.0 Ω |
| 9 | PT1000 - two-wire connection |

### Development information

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

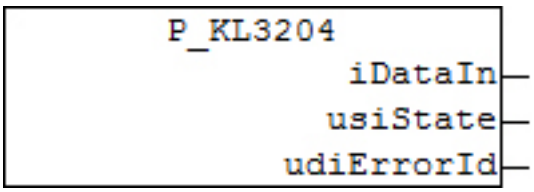
### Version history

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.4 P_KL3204

IO template for parameterizing a KL3204: 4-channel input terminal for resistance sensors. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL320xConfig. This function block configures only one channel of the KL3204. The corresponding number of function blocks is automatically declared via the Project Builder.

### Interface




### VAR_OUTPUT

```
iDataIn      : INT;
usiState     : USINT;
udiErrorId   : UDINT;
```

**iDataIn:** Corresponds to the data variable of the input process data. This output is suitable for direct processing in process data mode of the terminal.

**usiState:** Corresponds to the status variable of the input process data. This output is suitable for status assessment in process data mode of the terminal.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
iSetSensorType   : INT;
```

**iSetSensorType:** This input is used to set the sensor. The setting is based on the table below.

| iSetSensorType | Element |
|---|---|
| 0 | PT100 |
| 1 | NI100 |
| 2 | PT1000 |
| 3 | PT500 |
| 4 | PT500 |
| 5 | NI1000 |
| 6 | NI120 |
| 7 | Output 10.0 Ω – 5000.0 Ω |
| 8 | Output 10.0 Ω – 1200.0 Ω |

### Development information

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

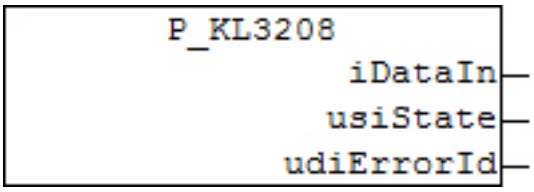
### Version history

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.5 P_KL3208

IO template for parameterizing a KL3208-0010: 8-channel input terminal for resistance sensors. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL3208Config. This function block configures only one channel of the KL3208. The corresponding number of function blocks is automatically declared via the Project Builder.

### Interface




### VAR_OUTPUT

```
iDataIn     : INT;
usiState    : USINT;
udiErrorId  : UDINT;
```

**iDataIn:** Corresponds to the data variable of the input process data. This output is suitable for direct processing in process data mode of the terminal.

**usiState:** Corresponds to the status variable of the input process data. This output is suitable for status assessment in process data mode of the terminal.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
iSetSensorType  : INT;
```

**iSetSensorType:** This input is used to set the sensor. The setting is based on the table below.

| iSetSensorType | Element |
|---|---|
| 0 | PT1000 |
| 1 | NI1000 |
| 2 | NI1000 according to Landis&Staefa characteristic: 1000Ω at 0°C and 1500Ω at 100°C. |
| 3 | NTC1K8 |
| 4 | NTC1K8_TK |
| 5 | NTC2K2 |
| 6 | NTC3K |
| 7 | NTC5K |
| 8 | NTC10K |
| 9 | NTC10KPRE |
| 10 | NTC10K_3204 |
| 11 | NTC10KTYP2 |
| 12 | NTC10KTYP3 |
| 13 | NTC10KDALE |
| 14 | NTC10K3A221 |
| 15 | NTC20K |
| 16 | Potentiometer, resolution 0.1 Ω |
| 17 | Potentiometer, resolution 1 Ω |
| 18 | NTC100K |

### Development information

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

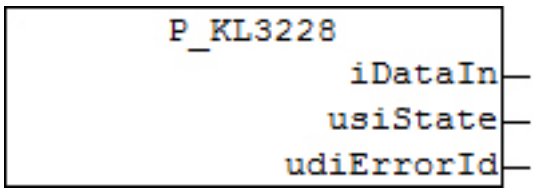
### Version history

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.6 P_KL3228

IO template for parameterizing a KL3228: 8-channel input terminal for resistance sensors. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL3228Config. This function block configures only one channel of the KL3228. The corresponding number of function blocks is automatically declared via the Project Builder.

### Interface




### VAR_OUTPUT

```
iDataIn     : INT;
usiState    : USINT;
udiErrorId  : UDINT;
```

**iDataIn:** Corresponds to the data variable of the input process data. This output is suitable for direct processing in process data mode of the terminal.

**usiState:** Corresponds to the status variable of the input process data. This output is suitable for status assessment in process data mode of the terminal.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
iSetSensorType  : INT;
```

**iSetSensorType:** This input is used to set the sensor. The setting is based on the table below.

| iSetSensorType | Element |
|---|---|
| 0 | PT1000 |
| 1 | NI1000 |
| 2 | NI1000 according to Landis&Staefa characteristic: 1000Ω at 0°C and 1500Ω at 100°C. |

**Development information**

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

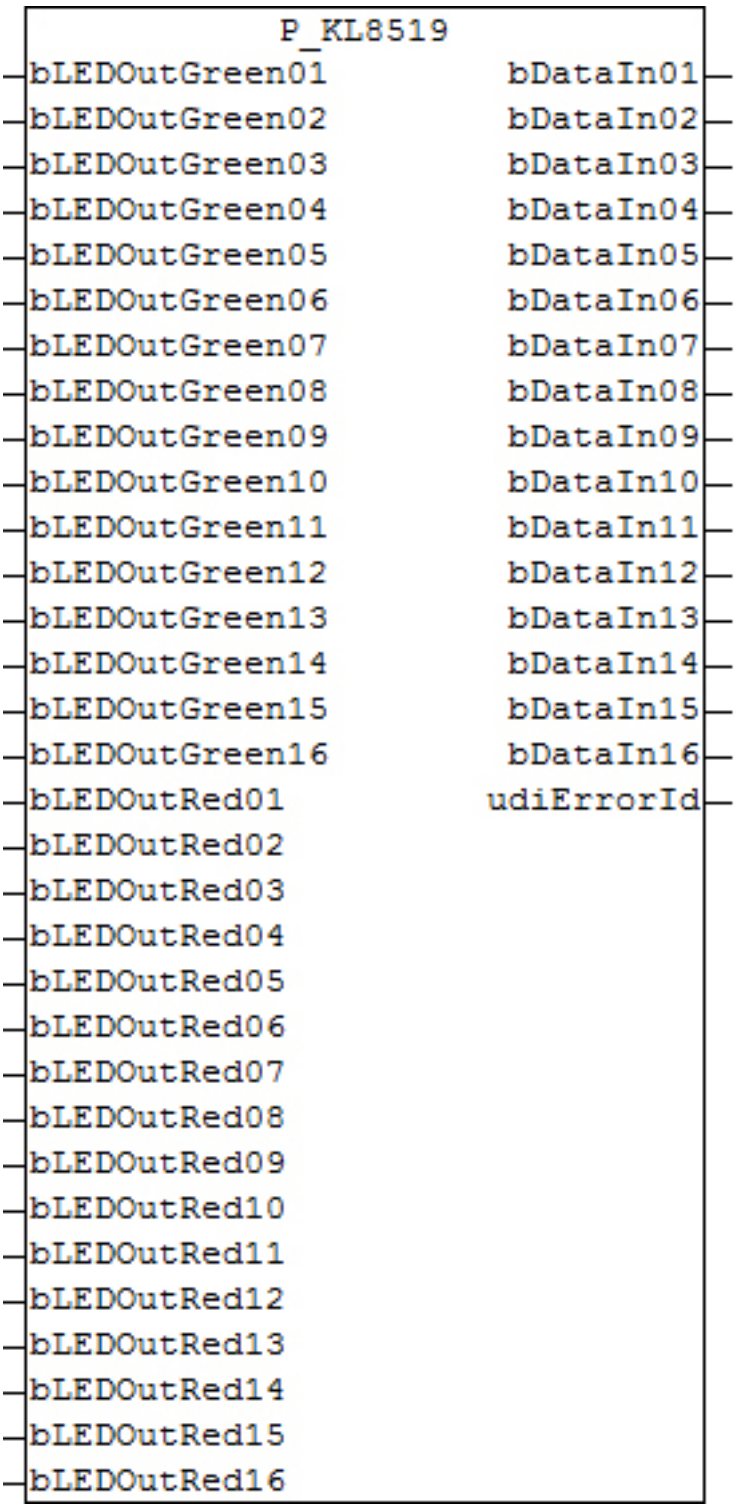
**Version history**

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.7 P_KL8519

IO template for parameterizing a KL8519: 16-channel digital input signal module. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL8519.

### Interface




### VAR_INPUT

```
bLEDOutGreen01 .. bLEDOutGreen16  : BOOL;
bLEDOutRed01 .. bLEDOutRed16      : BOOL;
```

**bLEDOutGreen01 .. bLEDOutGreen16:** Switches the green LED of the respective channel on, if switching of the LED from the PLC via the respective parameter *bLEDModePLC01* to *bLEDModePLC16* is enabled.

**bLEDOutRed01 .. bLEDOutRed16:** Switches the red LED of the respective channel on, if switching of the LED from the PLC via the respective parameter *bLEDModePLC01* to *bLEDModePLC16* is enaled.

### VAR_OUTPUT

```
bDataIn01 .. bDataIn16    : BOOL;
udiErrorId                : UDINT;
```

**bDataIn01 .. bDataIn16:** Data input channel 1 to 16.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
bLEDModePLC01 .. bLEDModePLC16        : BOOL;
bLEDColourAuto01 .. bLEDColourAuto16  : BOOL;
```

**bLEDModePLC01 .. bLEDModePLC16:** If this parameter is TRUE, the standard function of the LEDs for this channel is deselected. That is, it is no longer set via a high-signal of the channel set, but directly via the inputs *bLEDOutGreen01* to *bLEDOutGreen16* for the green colour scheme and *bLEDOutRed01* to *bLEDOutRed16* for the red colour scheme. It is possible to set both colours simultaneously.

**bLEDColourAuto01 .. bLEDColourAuto16:** If the standard mode for a channel is active (*bLEDModePLCxx* = FALSE), these parameters can be used to select the signal colour for the channel: *bLEDColourAutoxx* = FALSE: the green LED is selected for the signal status "High", *bLEDColourAutoxx* = FALSE: the red LED is selected.

**bDualColourAuto01 .. bDualColourAuto16:** If this parameter is set to TRUE, the respective channel is in two-colour mode. A high signal level at the input activates the colour selected under *bLEDColourAutoxx*, a low state the other colour.

**bLEDInvertAuto01 .. bLEDInvertAuto16:** If one of these parameters is set to TRUE, the respective low and high signal behaviour, as defined with the parameters *bLEDColourAutoxx* and *bLEDDualAutoxx*, is reversed.

**iKBusOffMode:** This parameter can be used to define the behaviour of the LEDs in the event of a K-bus error. The setting only applies for the LEDs, for which the standard mode was deselected through *bLEDModePLCxx* = TRUE. 0: LED remains on, if it was set via the PLC. 1: If the LED was switched on, it flashes with 500 ms clock frequency. 2: If the LED was switched on, it flashes with 1000 ms clock frequency. All other inputs are interpreted as "0".

**Parameter assignment example channel 1 for the standard mode *bLEDModePLC01* = FALSE:**

In the interest of clarity, the example lists the parameter combinations for the three parameters *bLEDColourAuto01, bLEDDualAuto01 and bLEDInvertAuto01* together with the LED switching characteristics for channel 1. All other channels can be parameterized differently.

If the default behaviour is deselected through *bLEDModePLC01* = TRUE, the three auto-parameters are no longer relevant, only the inputs *bLEDOutGreen01* and *bLEDOutRed01*.

| bLED-ColourAuto01 | bDual-ColourAuto01 | bLEDInvertAuto01 | Signal level at input 1 | LED colour channel 1 |
|---|---|---|---|---|
| FALSE | FALSE | FALSE | Low | OFF |
| FALSE | FALSE | FALSE | High | green |
| FALSE | FALSE | TRUE | Low | green |
| FALSE | FALSE | TRUE | High | OFF |
| FALSE | TRUE | FALSE | Low | red |
| FALSE | TRUE | FALSE | High | green |
| FALSE | TRUE | TRUE | Low | green |
| FALSE | TRUE | TRUE | High | red |
| TRUE | FALSE | FALSE | Low | OFF |
| TRUE | FALSE | FALSE | High | red |
| TRUE | FALSE | TRUE | Low | red |
| TRUE | FALSE | TRUE | High | OFF |
| TRUE | TRUE | FALSE | Low | green |
| TRUE | TRUE | FALSE | High | red |

| bLED-ColourAuto01 | bDual-ColourAuto01 | bLEDInvertAuto01 | Signal level at input 1 | LED colour channel 1 |
|---|---|---|---|---|
| TRUE | TRUE | TRUE | Low | red |
| TRUE | TRUE | TRUE | High | green |

**Development information**

| Entwicklungsumgebung | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

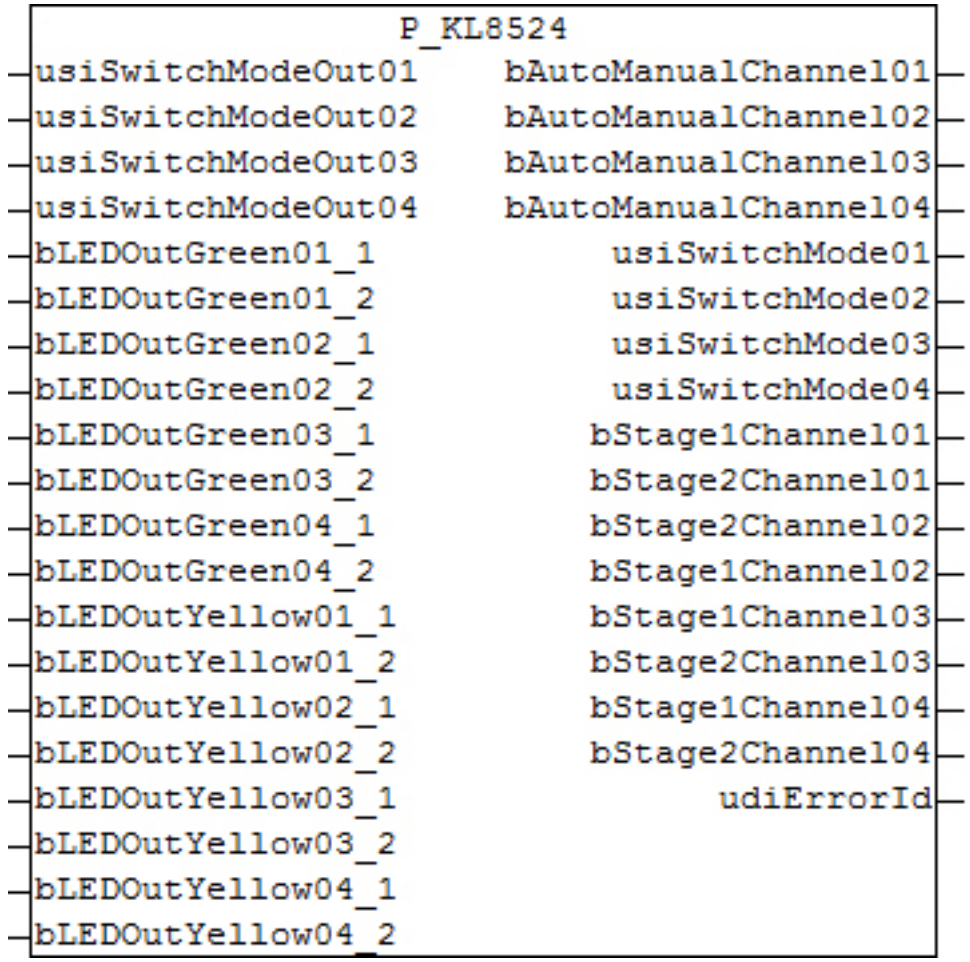
**Version history**

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.8 P_KL8524

IO template for parameterizing a KL8524: 4 x 2-channel digital output module. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL8524.

**Interface**



### VAR_INPUT

```
usiSwitschModeOut01 .. usiSwitchModeOut04    : USINT;
bLEDOutGreen01_1 .. bLEDOutGreen04_2         : BOOL;
bLEDOutYellow01_1 .. bLEDOutYellow04_2       : BOOL;
```

**usiSwitchModeOut01 .. usiSwitchModeOut04:** Control of stages channel 1 to 4. Permissible values: "0", "1" and "2". All other inputs are interpreted as "0".

**bLEDOutGreen01_1 .. bLEDOutGreen04_2:** Control of the green LEDs channels 1 to 4, step 1 and 2, if switching of the LED from the PLC via the respective parameter *bLEDModePLC01_1* to *bLEDModePLC04_2* is enabled.

**bLEDOutYellow01_1 .. bLEDOutYellow04_2:** Control of the yellow LEDs channels 1 to 4, step 1 and 2, if switching of the LED from the PLC via the respective parameter *bLEDModePLC01_1* to *bLEDModePLC04_2* is enabled.

### VAR_OUTPUT

```
bAutoManualChannel01 .. bAutoManualChannel04 : BOOL;
usiSwitchMode01 .. usiSwitchMode04           : USINT;
usiOnOffChannel01 .. usiOnOffChannel04       : USINT;
udiErrorId                                   : UDINT;
```

**bAutoManualChannel01 .. bAutoManualChannel04:** Status auto/manual selector switch channel 1 to 4: FALSE: Switch is set to "man", TRUE: Switch is set to "auto".

**usiSwitchMode01 .. usiSwitchMode04:** Numerical status (0, 1, 2) of the three-stage switch channels 1 to 4.

**usiOnOffChannel01 .. usiOnOffChannel04:** Active step channels 1 to 4: in manual mode (selector switch = man) that of the three-stage switch, in automatic mode (selector switch = auto) that of the respective input *usiSwitchmodeOut01* to *usiSwitchmodeOut04.*

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
bLEDModePLC01_1 .. bLEDModePLC04_2                : BOOL;
iKBusOffModeChannel01 .. iKBusOffModeChannel04    : INT;
bOutModeChannel01 .. bOutModeChannel04            : BOOL;
usiSwitchDelayChannel01 .. usiSwitchDelayChannel04  : USINT;
```

**bLEDModePLC01_1 .. bLEDModePLC04_2:** If one of these parameters is set to TRUE, the standard function for the respective LED is deselected. That is, it is no longer influenced by the respective signal stage, but directly via the inputs *bLEDOutGreen01_1* to *bLEDOutGreen*4_2 for the green colour scheme and *bLEDOutYellow01* to *bLEDOutYellow*04_2 for the yellow colour scheme. It is possible to set both colours simultaneously. If the parameter is FALSE, the following applies: step not active = yellow, step active = green.

**iKBusOffModeChannel01 .. iKBusOffModeChannel04:** Selection of the behaviour on K-bus error: 0: no output is set, 1: step 1 is set, 2: step 2 is set. All other inputs are interpreted as "0".

**bOutModeChannel01 .. bOutModeChannel04:** Output mode channels 1 to 4: FALSE: output mode 1, TRUE: output mode 2, see guide FB_KL8524.

**usiSwitchDelayChannel01 .. usiSwitchDelayChannel04:** Input of the delay time for step 2 for the respective channel as multiple of 10 ms.

### Development information

| Entwicklungsumgebung | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

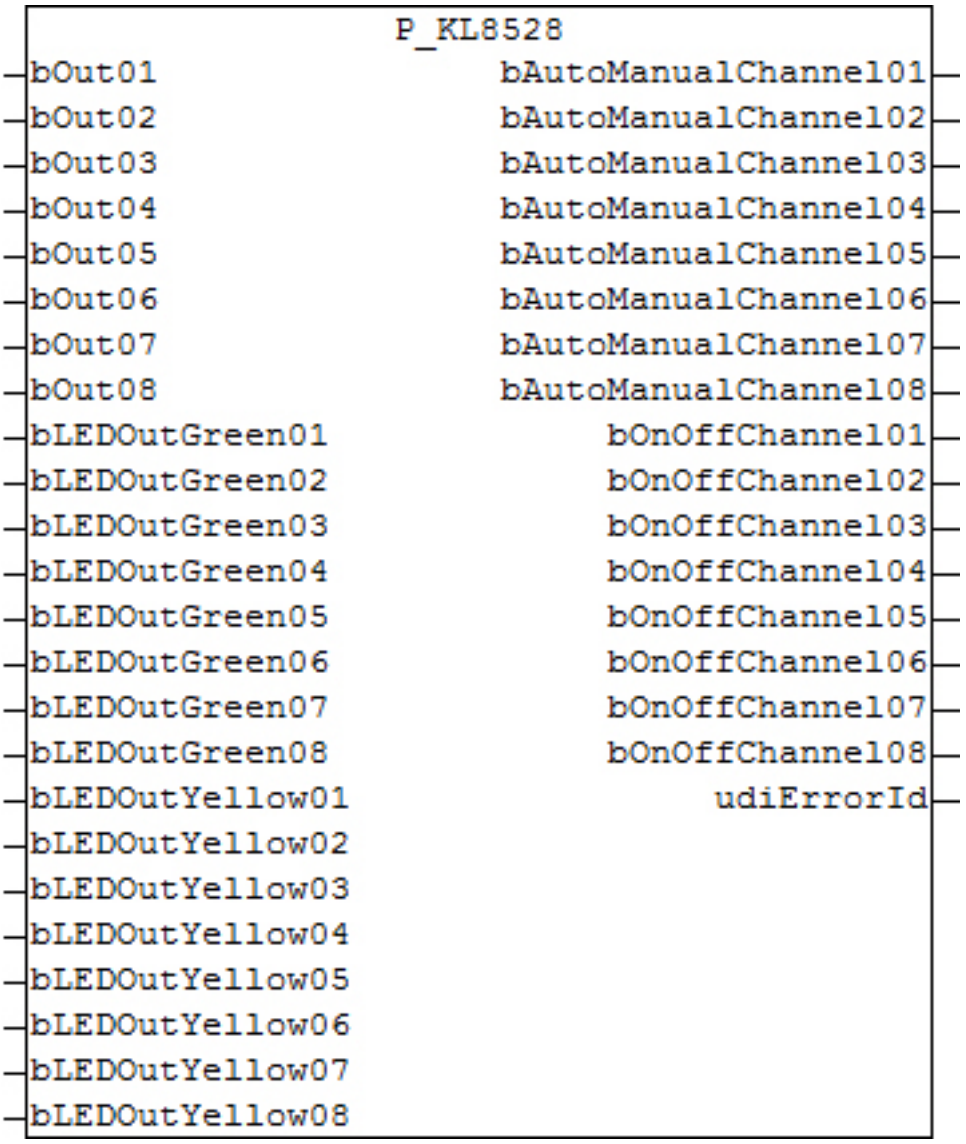
**Version history**

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.9 P_KL8528

IO template for parameterizing a KL8528: 8-channel digital output module. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL8528.

**Interface**




**VAR_INPUT**

```
bOut01 .. bOut08                    : BOOL;
bLEDOutGreen01 .. bLEDOutGreen08    : BOOL;
bLEDOutYellow01 .. bLEDOutYellow08  : BOOL;
```

**bOut01 .. bOut04:** Control outputs channel 1 to 8.

**bLEDOutGreen01 .. bLEDOutGreen08:** Control of the green LEDs channels 1 to 8, if switching of the LED from the PLC via the respective parameter *bLEDModePLC01* to *bLEDModePLC08* is enabled.

**bLEDOutYellow01 .. bLEDOutYellow08:** Control of the yellow LEDs channels 1 to 8, if switching of the LED from the PLC via the respective parameter *bLEDModePLC01* to *bLEDModePLC08* is enabled.

### VAR_OUTPUT

```
bAutoManualChannel01 .. bAutoManualChannel04 : BOOL;
bOnOffChannel01 .. bOnOffChannel04           : BOOL;
udiErrorId                                   : UDINT;
```

**bAutoManualChannel01 .. bAutoManualChannel04:** Status auto/manual selector switch channel 1 to 8: FALSE: Switch not set to "auto", TRUE: Switch is set to "auto".

**bOnOffChannel01 .. bOnOffChannel08:** Status auto/manual selector switch channel 1 to 8: FALSE: Switch not set to "on", TRUE: Switch set to "on".

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
bLEDModePLC01 .. bLEDModePLC08                 : BOOL;
bKBusOffModeChannel01 .. bKBusOffModeChannel08 : BOOL;
```

**bLEDModePLC01 .. bLEDModePLC08:** If one of these parameters is set to TRUE, the standard function for the respective LED is deselected. That is, it is no longer influenced by the respective signal stage, but directly via the inputs *bLEDOutGreen01* to *bLEDOutGreen08* for the green colour scheme and *bLEDOutYellow01* to *bLEDOutYellow08* for the yellow colour scheme. It is possible to set both colours simultaneously. If the parameter is FALSE, the following applies: step not active = yellow, step active = green.

**bKBusOffModeChannel01 .. bKBusOffModeChannel08:** Output behaviour on K-bus error. FALSE: Output not set, TRUE: Output set, if switch set = "auto".

### Development information

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

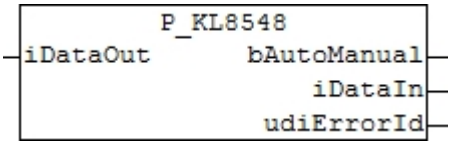
### Version history

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

## 9.82.10 P_KL8548

IO template for parameterizing a KL8548: 8-channel analog output module 0…10 V. On PLC restart this template configures the terminal with the parameters entered in the Project Builder and then switches to process data mode. This template is based on the function block FB_KL8548. This function block configures only one channel of the KL8548. The corresponding number of function blocks is automatically declared via the Project Builder.

### Interface



### VAR_INPUT

```
iDataOut      : INT;
```

**iDataOut:** Analog output value.

### VAR_OUTPUT

```
bAutoManual   : BOOL;
iDataIn       : INT;
udiErrorId    : UDINT;
```

**bAutoManual:** Switch position: FALSE = "man". TRUE = "auto".

**iDataIn:** Position of the control potentiometer.

**udiErrorId:** contains the command-specific error code of the most recently executed command. See Error codes.

### Parameter

```
bBarGraphEcoMode  : BOOL;
iKBusOffMode      : INT;
iKBusOffValue     : INT;
```

**bBarGraphEcoMode:** Bar chart display mode: FALSE: full-scale mode (solid bar), TRUE: ECO mode (only one point).

**iKBusOffMode:** Behaviour of the terminal output on K-Bus error: 0: Output becomes 0, 1: Output retains the last value, when the PLC stops it changes to 0, 2: Output assumes the value *iKBusOffValue*, if switch position = "auto". All other inputs are interpreted as "0".

**iKBusOffValue:** Value, which the output is to assume in the event of a K-Bus error (or PLC stopped) in mode 2.

### Development information

| Development environment | BACnet Revision | Target system | required supplement |
|---|---|---|---|
| TwinCAT 2.11 R3/x64 from build 2254 | n/a | PC/CX | TS8040 \| TwinCAT Building Automation from V1.1.0 |

### Version history

| Version number | Comments |
|---|---|
| 1.0.0.0 | First release |

# 10 TwinCAT BA Project Builder

The basic prerequisite for meeting the high demands placed on building automation, such as comfort, the saving of energy, low investment and running costs and a fast return on investment, is an integrated, co-ordinated control system for the automation of all building technical systems. In TwinCAT Building Automation Beckhoff has developed a software product that reduces the engineering time and integrates all essential functions for each system of a modern building automation. Extensive software libraries and supplements continue the concept of the modular Beckhoff automation construction kit at software level as well. The new software suite essentially encompasses three basic functions:

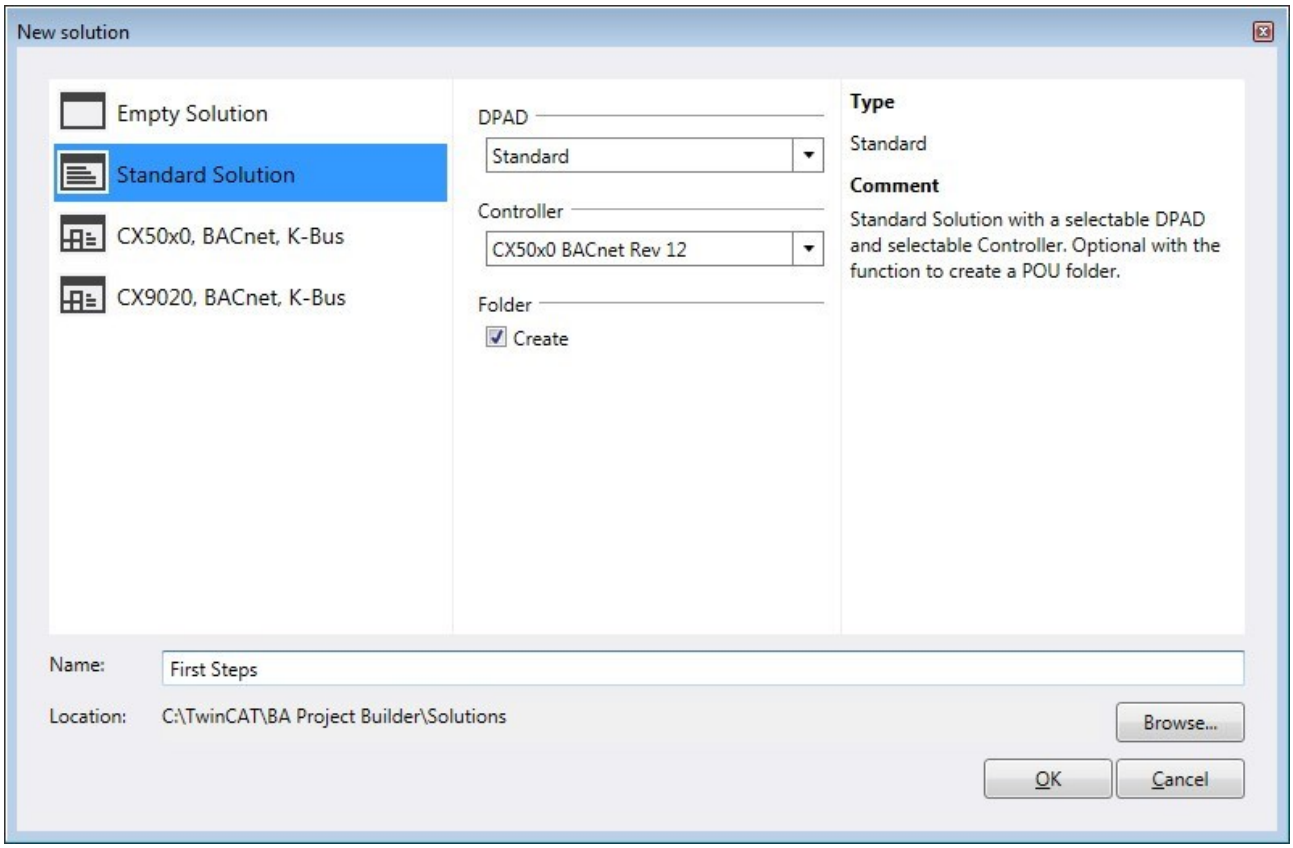
1. TwinCAT BA PLC Libraries: basic functions for all building systems.
2. TwinCAT BA PLC Templates: function templates for all building systems.
3. TwinCAT BA Project Builder: configuration tool that links templates, hardware and BACnet objects with one another.

## 10.1 First steps

The steps necessary in order to create a solution with the TwinCAT BA Project Builder are shown in the following.

### Creating a project

Call the dialog to create a new solution via the menu *Solution -> New*.

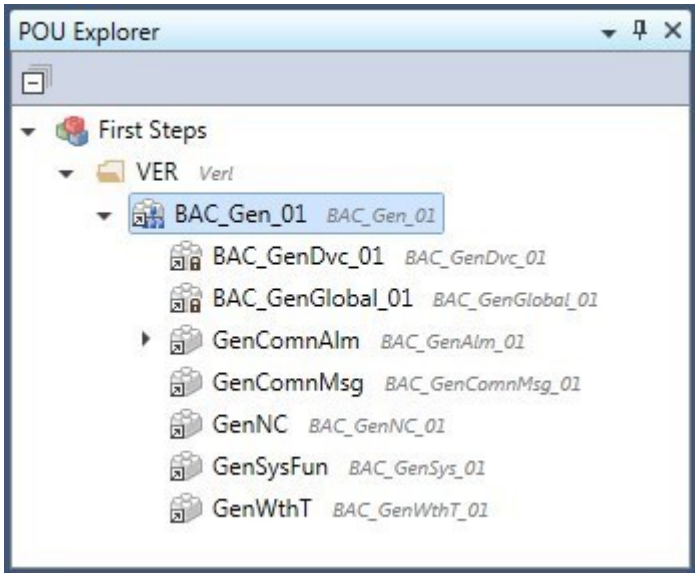



In this dialog you can specify the name of the solution and the path via which it should be saved. Furthermore, you must select the **D**ata **P**oint **A**ddressing **D**escription (DPAD) which will be used as the basis for the project engineering of the solution. The BA Project Builder offers the option of selecting an empty data point addressing description (DPAD) and defining the DPAD yourself.

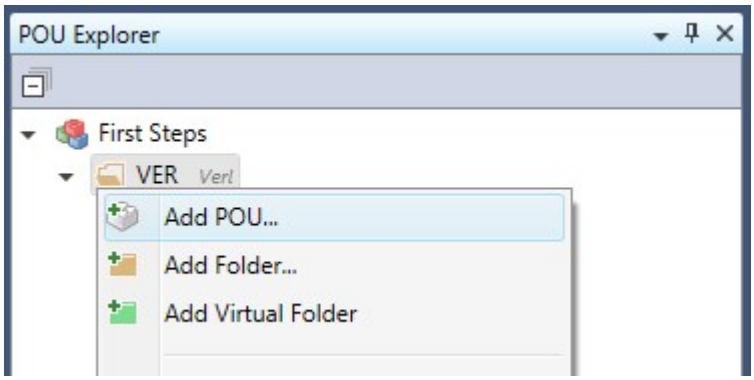
### Adding templates

The *POU Explorer* defines the system structure on the software side. Folders in the *POU Explorer* define the individual levels (DPAD level). The possible identifiers (DPAD key) of a level are preset by the DPAD. The desired DPAD key can be selected via the *Properties* tool window.

Open the *Templates* tool window and drag the template *BAC_Gen_01* onto the folder in the *POU Explorer* by drag & drop. This is a call template that inserts the basic PLC function blocks for the BACnet server into the PLC program. This call template is linked with further templates that are added automatically to the PLC program. All the POUs that have been added to the PLC project are displayed in the *POU Explorer*.




Alternatively, you can also add a POU via the context menu of the folder:

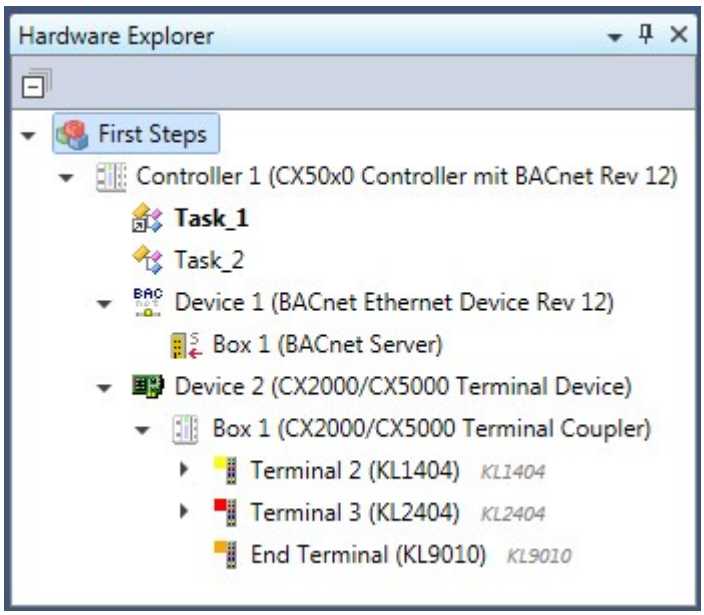



A double-click on a POU opens a dialog via which further parameters or BACnet properties can be edited.

**Adding hardware**

All controllers and their devices and terminals are assembled in the *Hardware Explorer*. When the new solution is created, a CX50x0 is automatically created with the necessary PLC tasks, BACnet server and I/O devices.

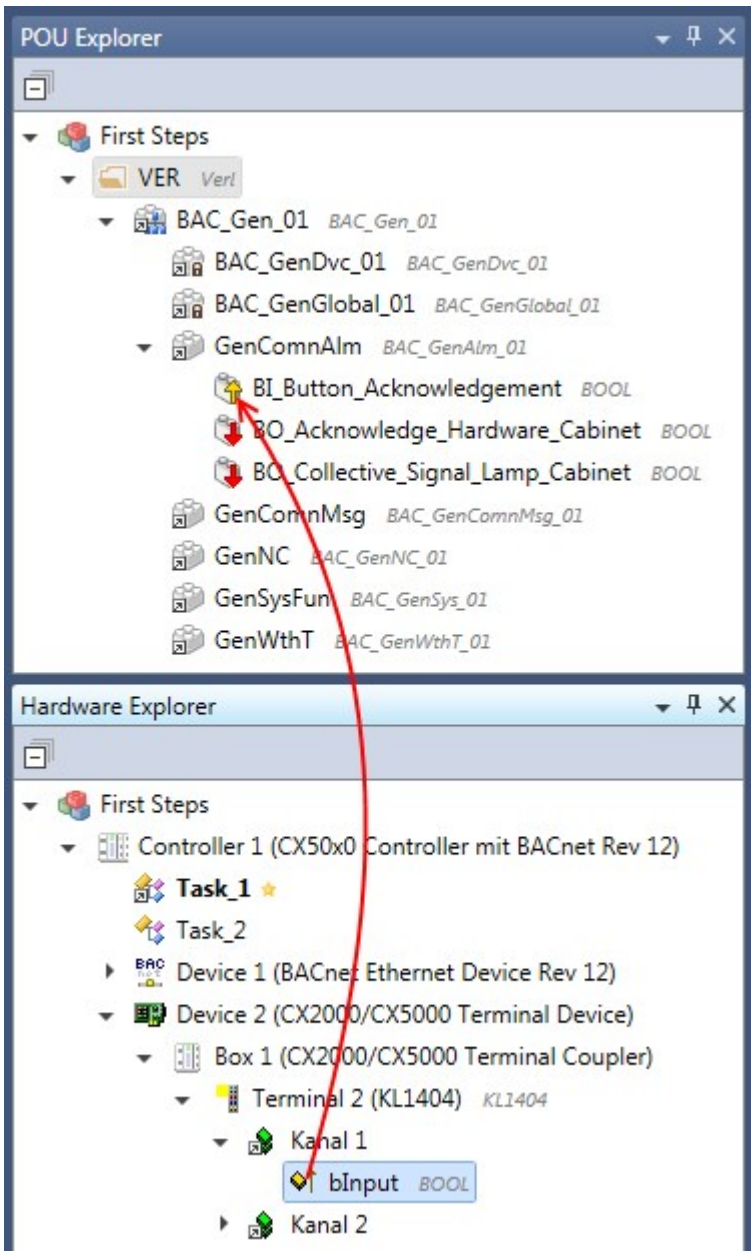
The terminal coupler *Box 1* is located below *Device* 2. Now drag the terminals *KL1404* and *KL*2404 out of the *Terminals* tool window into *Box 1* by drag & drop. The *Hardware Explorer* should then look like this:




Here, too, you can alternatively add the terminals via the context menu.

**Linking POUs with terminals.**

If POUs have inputs and outputs they can be linked to the inputs and outputs of Bus Terminals.

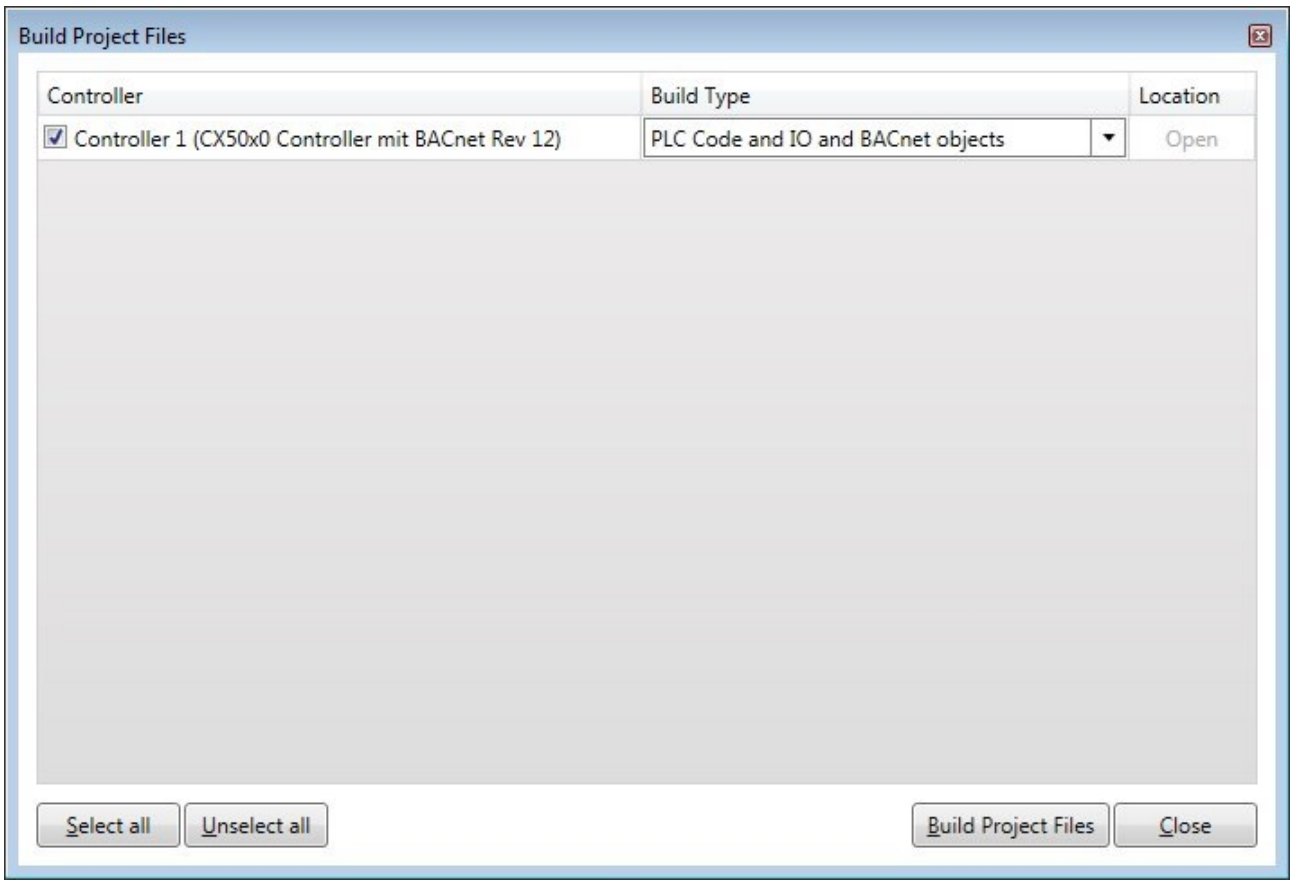



The links can be created by drag & drop or via the context menu.

**Generating TwinCAT project files.**

Once all POUs have been created and the parameters and BACnet properties have been set as desired and linked to the hardware, the TwinCAT project files can be generated. Call the menu *Action -> Build Project Files*.

Select the controller for which you want to generate the TwinCAT project files.



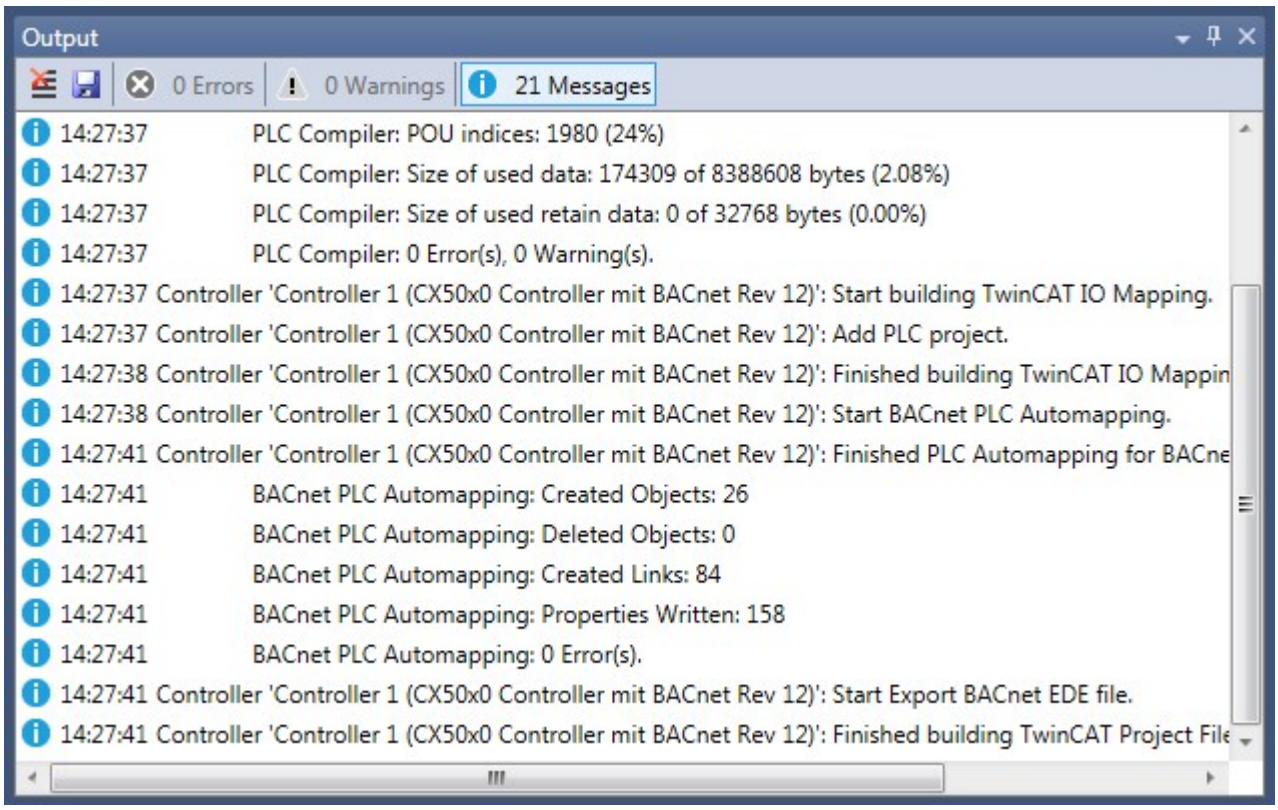

The column *Build Type* specifies which project files are created. This has a considerable influence on the time taken for the generation.

**Requirements**

| | |
|---|---|
| PLC Code and IO and BACnet objects | The pro file for the TwinCAT PLC Control is generated and compiled. Furthermore, the tsm file is created for the TwinCAT System Manager with all I/O devices, BACnet objects and links. |
| PLC Code and IO (without BACnet objects) | The pro file for the TwinCAT PLC Control is generated and compiled. Furthermore, the tsm file is created for the TwinCAT System Manager with all I/O devices and links. No BACnet objects are added. |
| PLC Code (only pro file, no tpy file) | Only the pro file for the TwinCAT PLC Control is generated. The program is not compiled. |

Various status messages are displayed in the *Output* tool window after the start of the generation.

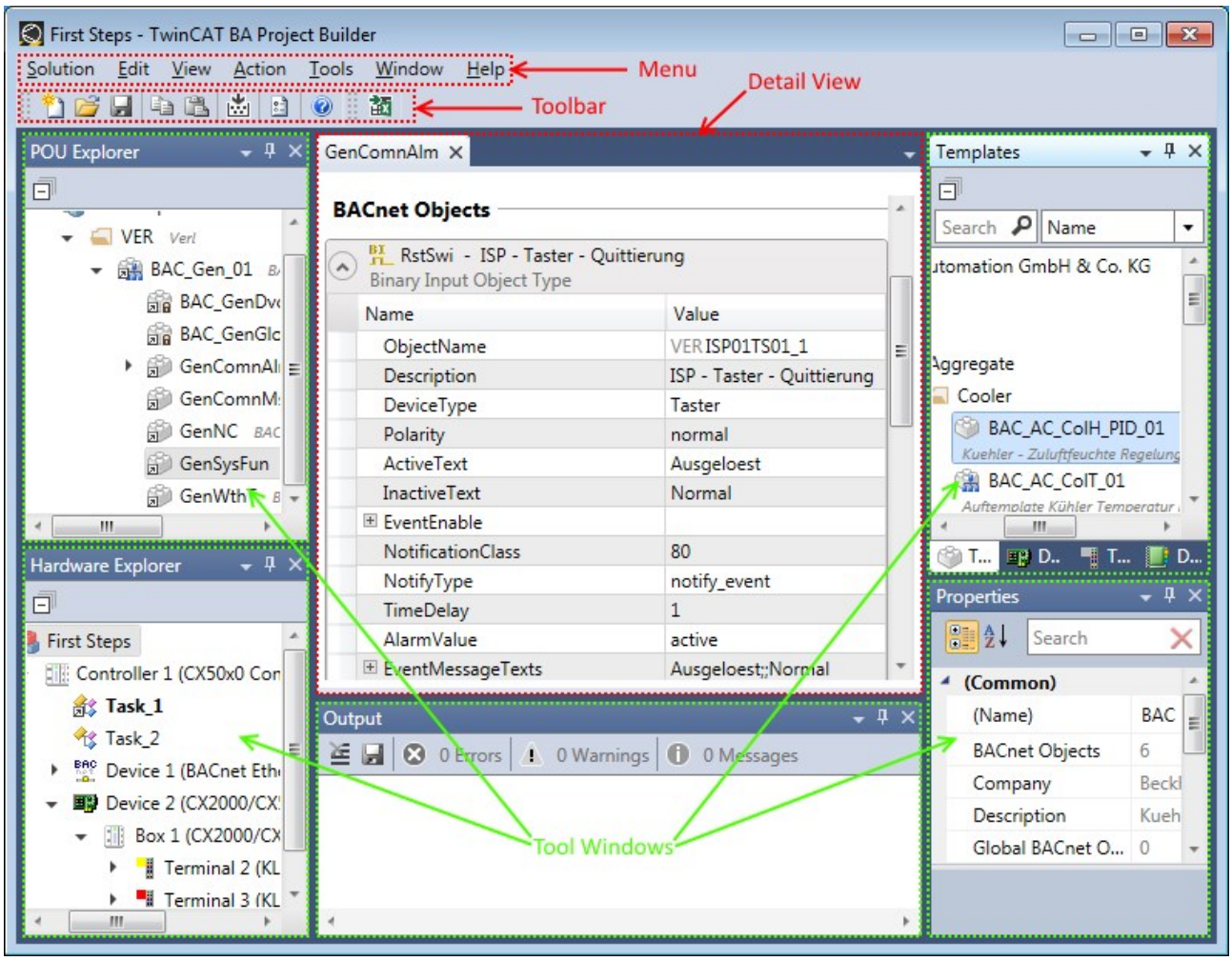
These messages indicate whether all project files have been generated successfully or whether errors occurred.




A directory is created for each controller in the same folder in which the solution is saved. This directory has the same name as the controller in the BA Project Builder and has the subdirectory *~TwinCAT*, where you will find the generated TwinCAT project files.

## 10.2 Main Window

The BA Project Builder is divided into several sections and windows.

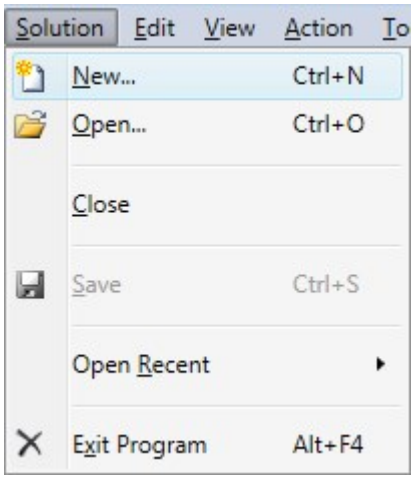



**Menu**

The individual actions and commands can be executed via the menu. Each menu item can be reached via the keyboard.

After pressing the Alt key, a letter of each menu item is underlined.


Now press the underlined letter on the keyboard.




Letters are underlined once more in the menu; you can execute the commands with these letters.

To execute the *Close* command in the *Solution* menu, you must press the <Alt> key and then 's' and 'c' individually.

Some important menu items are directly accessible via shortcuts, however. The corresponding shortcut, if one exists, is shown in the menu behind the respective command.

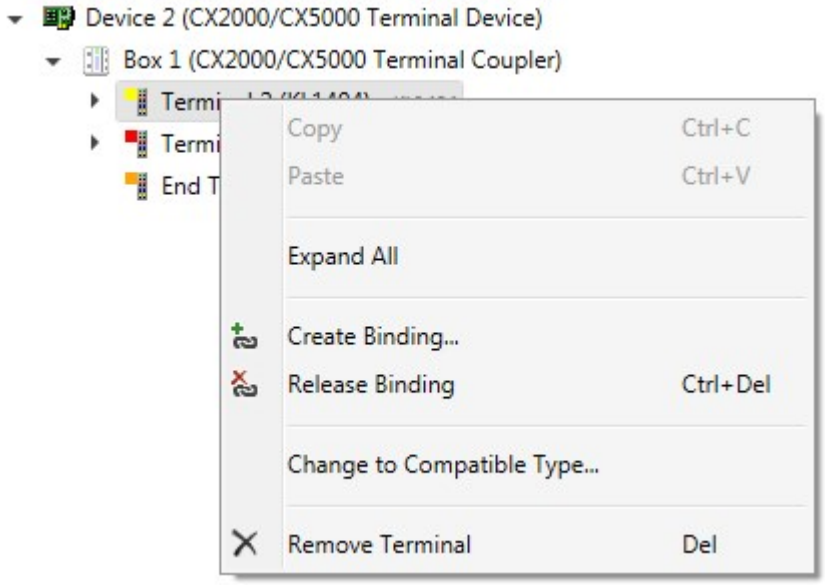
To call the *Open* command with a shortcut, you must press and hold the <Ctrl> key and then press the 'O' key.

### Toolbar

Important commands can be accessed directly via the toolbar.

### Context menus

Several elements in the tool windows offer context menus. Certain actions such as Add or Delete can be directly executed via these menus.




### Detail View

In the middle of the BA Project Builder is the *Detail View*. Further dialogs are opened in this section by double-clicking on certain elements in the tool windows.

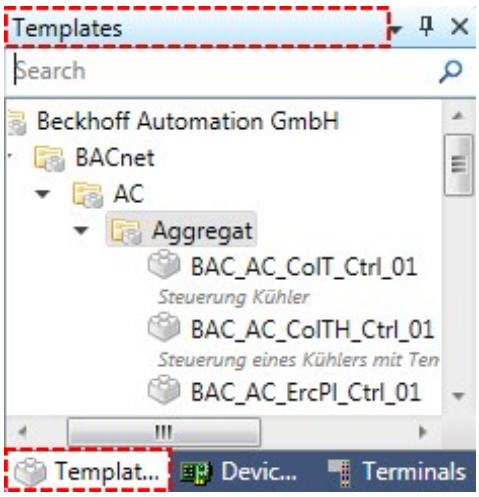
### Tool Windows

The tool windows are grouped around the *Detail View*. A tool window is a self-contained window that can be freely moved or even completely hidden. In addition to operation with the mouse, the *View* and *Window* menus also offer appropriate commands.

### Adjust position

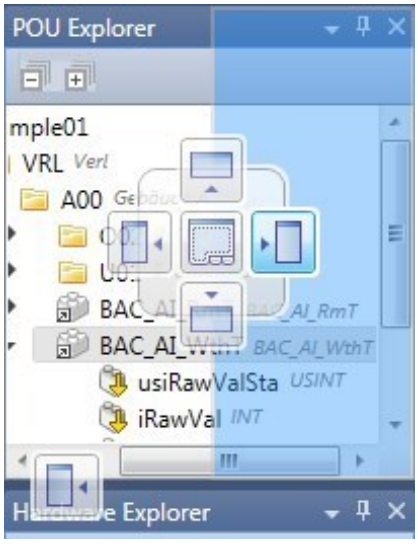
The size and position of the tool windows are freely adjustable.

To do this, left-click on the upper area of a tool window. If several tool windows are positioned on top of one another in an area, you must click on the corresponding tab.




Keep the left mouse button pressed and drag the window to the desired position. For better orientation the tool window is displayed as a blue rectangle. Small icons are also shown, which indicate possible positions.

As soon as you drag the tool window onto one of these icons, it is docked to the corresponding point. Release the left mouse button when the desired position has been reached.



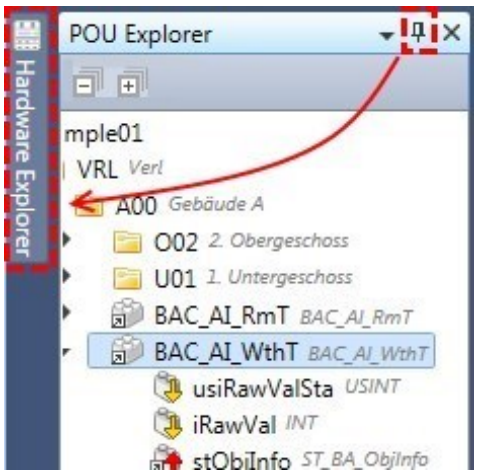

The windows can also be placed outside of the BA Project Builder, which can be very helpful in the case of workstations with several monitor screens.

**Auto Hide**

Each tool window has an *Auto Hide* function. This is switched on or off via a small symbol in the upper area of the tool window.

If this is switched on, the tool window collapses to the edge of the BA Project Builder. If you move the mouse over a hidden tool window, it will be shown again.

If the mouse leaves the tool window it will automatically be collapsed to the edge again. The command *Window -> Auto Hide All* collapses all tool windows simultaneously.




A tool window can be closed with the *X* in the upper area. Closed tool windows can be opened again via the menu item *View*. All tool windows can be closed simultaneously with the command *Window -> Close All*.

The command *Window -> Reset Window Layout* resets all windows to their standard positions.

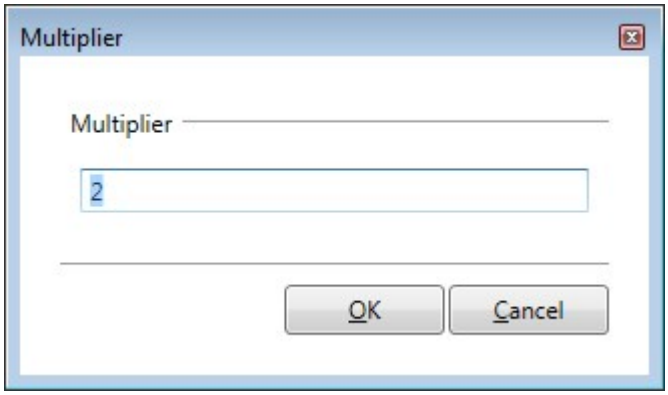
### Drag & Drop

Certain elements can be copied from one tool window to another by drag & drop. Drag & drop is also used within a tool window, for example for changing the position.

Hover with the mouse over the desired element, then press and hold the left mouse button. Keeping the mouse button pressed, drag the element to the desired position. The icon below the mouse pointer indicates whether the element can be deposited at the current position.

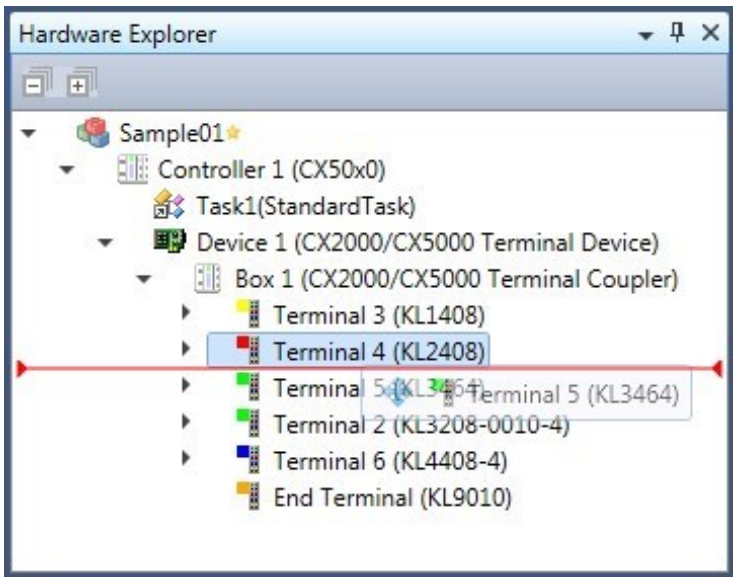
If you press and hold the shift key before releasing the left mouse button, you can insert several elements at the same time.

In the dialog, specify the number of elements that you wish to add.




In some tool windows the position of the elements can be changed by drag & drop.

If the position is valid, a red line marks the position to which the element is moved.

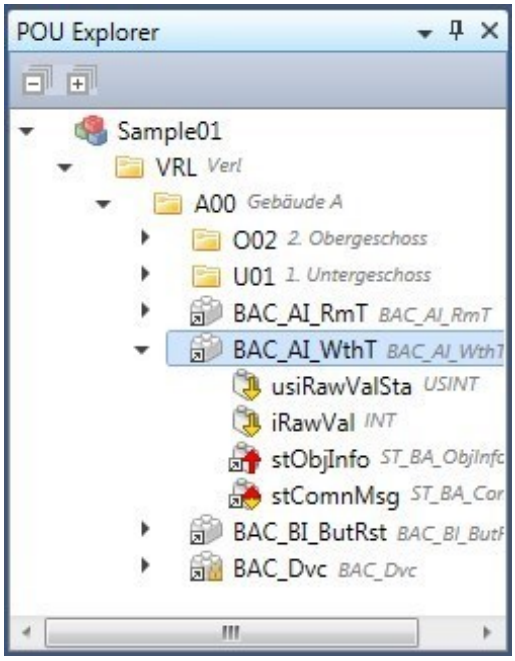



**POU Explorer**

The *POU Explorer* structures the individual POUs according to the data point addressing description (DPAD). Each folder level represents a certain area of the DPAD.

The POUs are dragged out of the *Templates* tool window and onto the desired folder by drag & drop or via the context menu.

The full name of the PLC function block is derived from the position inside the *POU Explorer* when generating the TwinCAT project files.

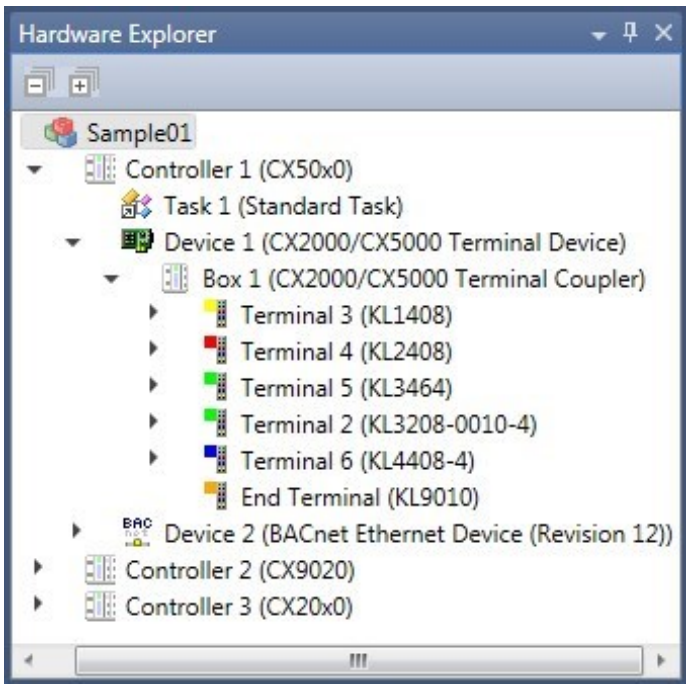



A double-click on a POU opens a further dialog in the *Detail View*. Additional parameters such as BACnet properties can be edited via this dialog.

Further information is displayed in the *Properties* tool window for each marked element in the *POU Explorer*.

**Hardware Explorer**

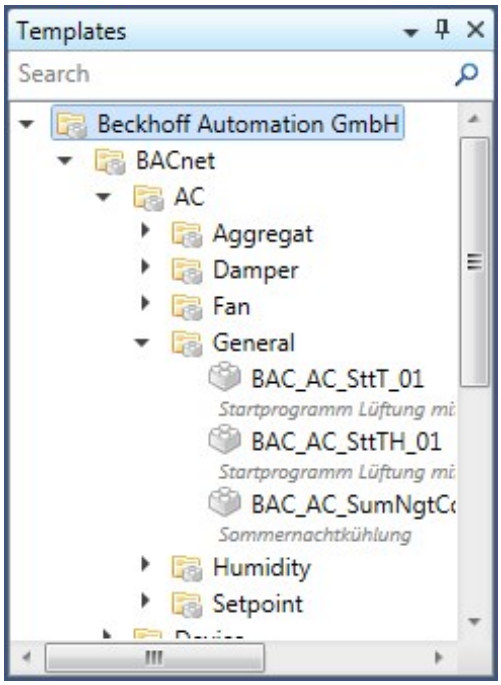
The employed hardware is entered in the *Hardware Explorer*. This is added from the *Devices* and *Terminals* tool windows by drag & drop or by context menu.




Here, too, a double-click opens further dialogs in the *Detail View*. Further information is displayed in the *Properties* tool window for each selected element.

### Templates

All templates that can be added to the *POU Explorer* are displayed in this tool window.

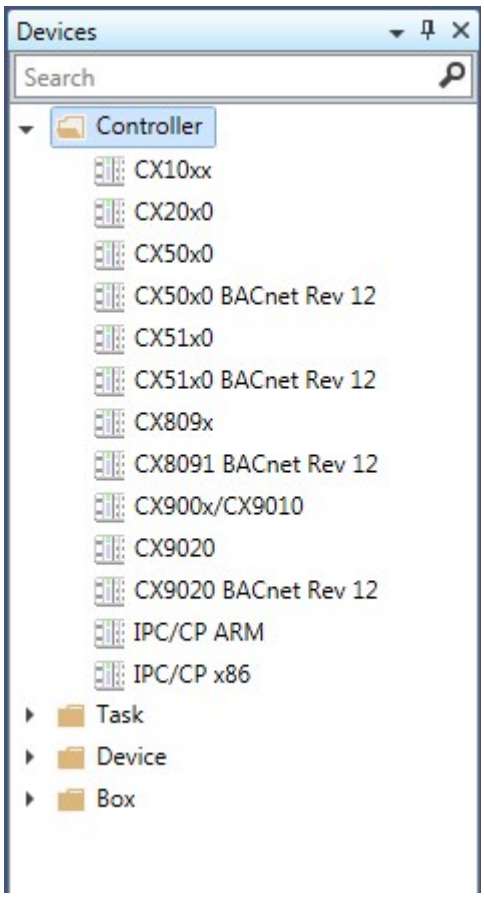



The display in the tool window can be filtered by entering a name in the search field in the upper area.

The *Properties* tool window displays further information about each marked element.

### Devices

The elements that can be added to the *Hardware Explorer* are displayed in the *Devices* tool window.

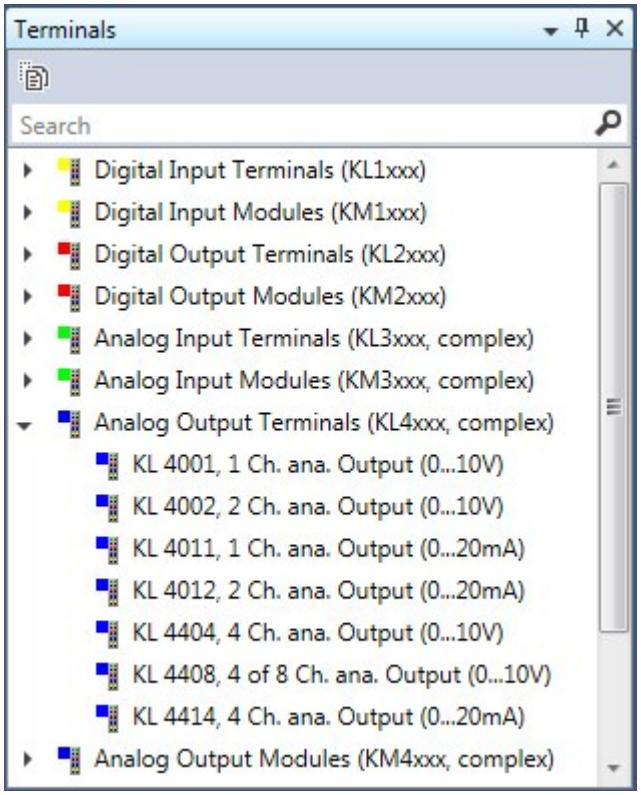



The display in the tool window can be filtered by entering a name in the search field in the upper area.

The *Properties* tool window displays further information about each marked element.

**Terminals**

All terminals that can be added to the *Hardware Explorer* are located in the *Terminals* tool window.



To speed up the choice of terminal, only the most frequently used terminals are displayed. The display filter is deactivated by the button in the upper area and all available terminals are then displayed.

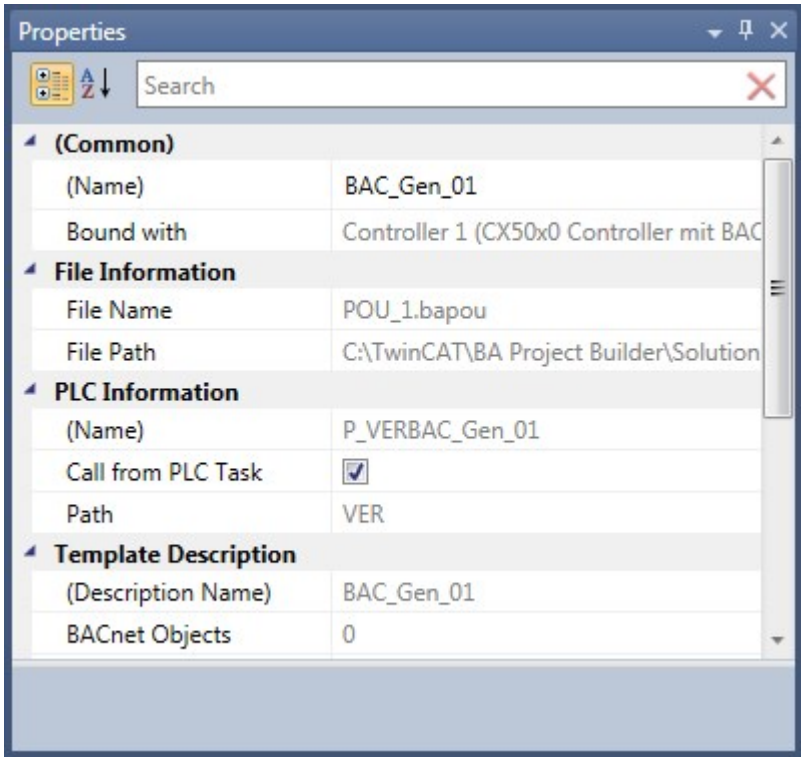
The display in the tool window can be filtered by entering a name in the search field in the upper area.

The *Properties* tool window displays further information about each marked element.

### Properties

The *Properties* tool window displays further information about the currently marked element in another window.

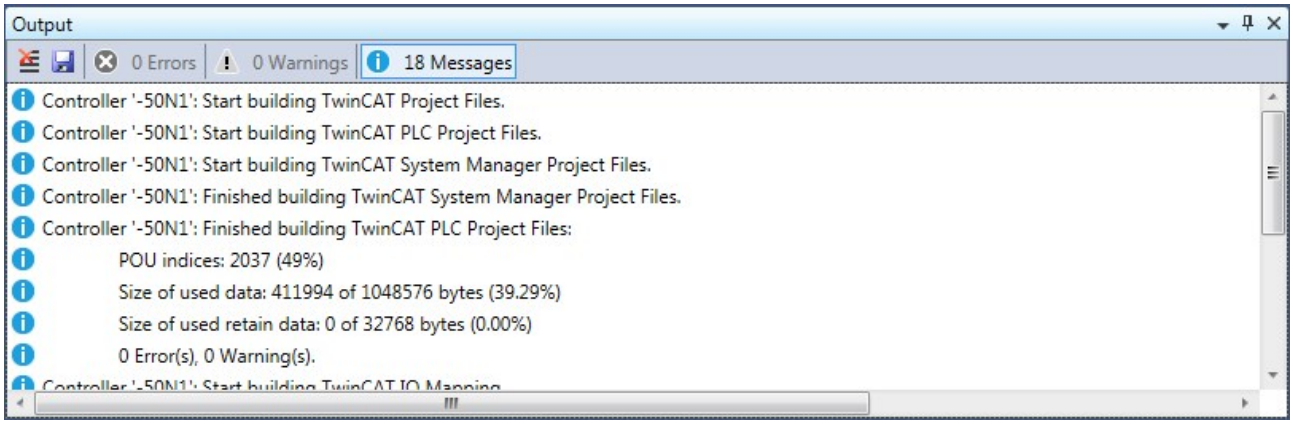
The sorting method (alphabetical or by category) can be changed with the two buttons in the upper area.




The display in the tool window can be filtered by entering a name in the search field in the upper area.

### Output

Various messages are displayed in the *Output* tool window. Messages are thus displayed during the generation of the TwinCAT project files or if errors are detected in the project engineering.




The *Output* tool window can be deleted using the button in the top left-hand area. Furthermore, the messages can be saved in a text file.

The messages are divided into three categories: *Errors*, *Warnings* and *Messages*. Each category can be hidden by actuating the respective button in the upper area.

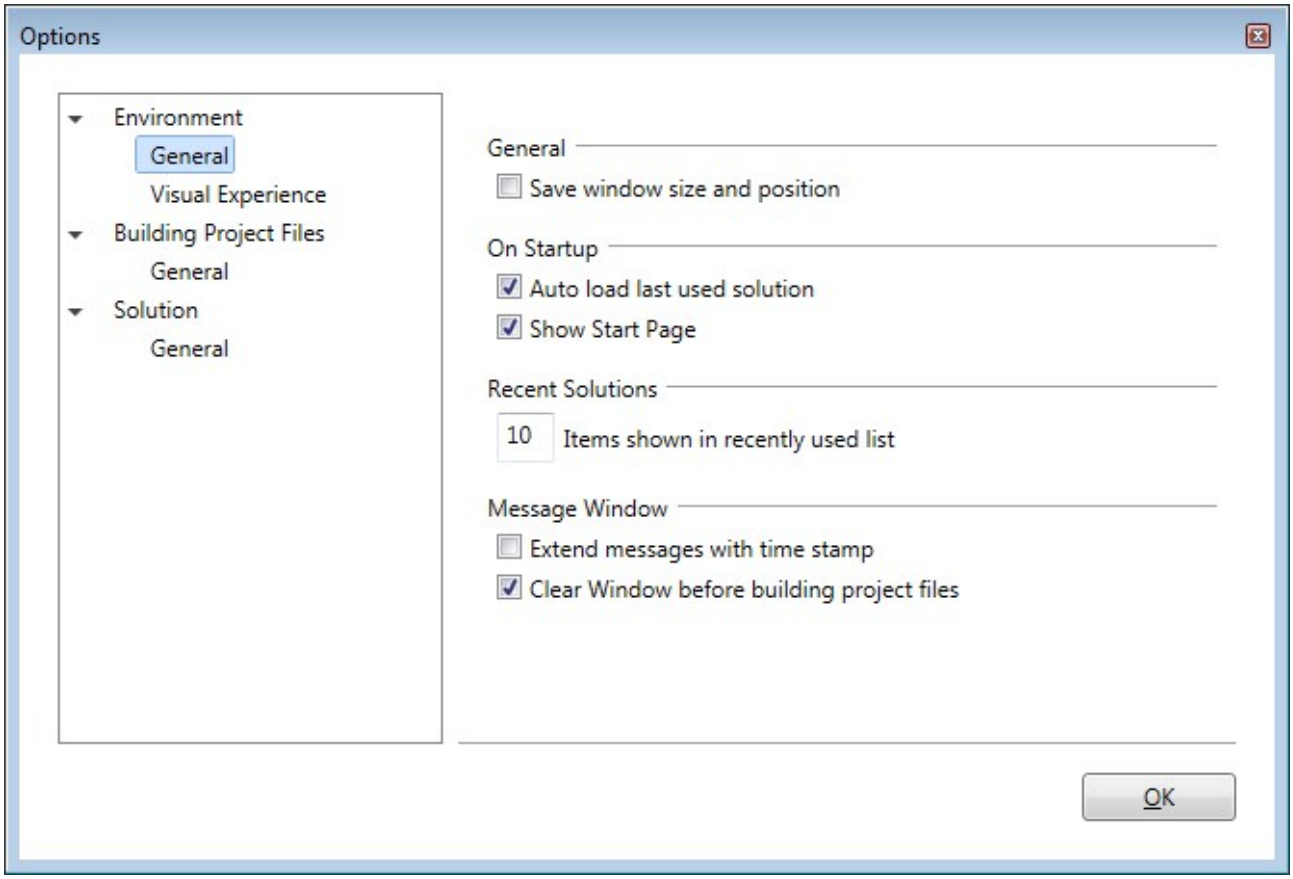
The next error message (Error) is marked by pressing the F4 key.

## 10.3 Options

The *Options* dialog can be called via the menu item *Tools --> Options*. This dialog offers various parameters that influence both the appearance and the behavior of the BA Project Builder.

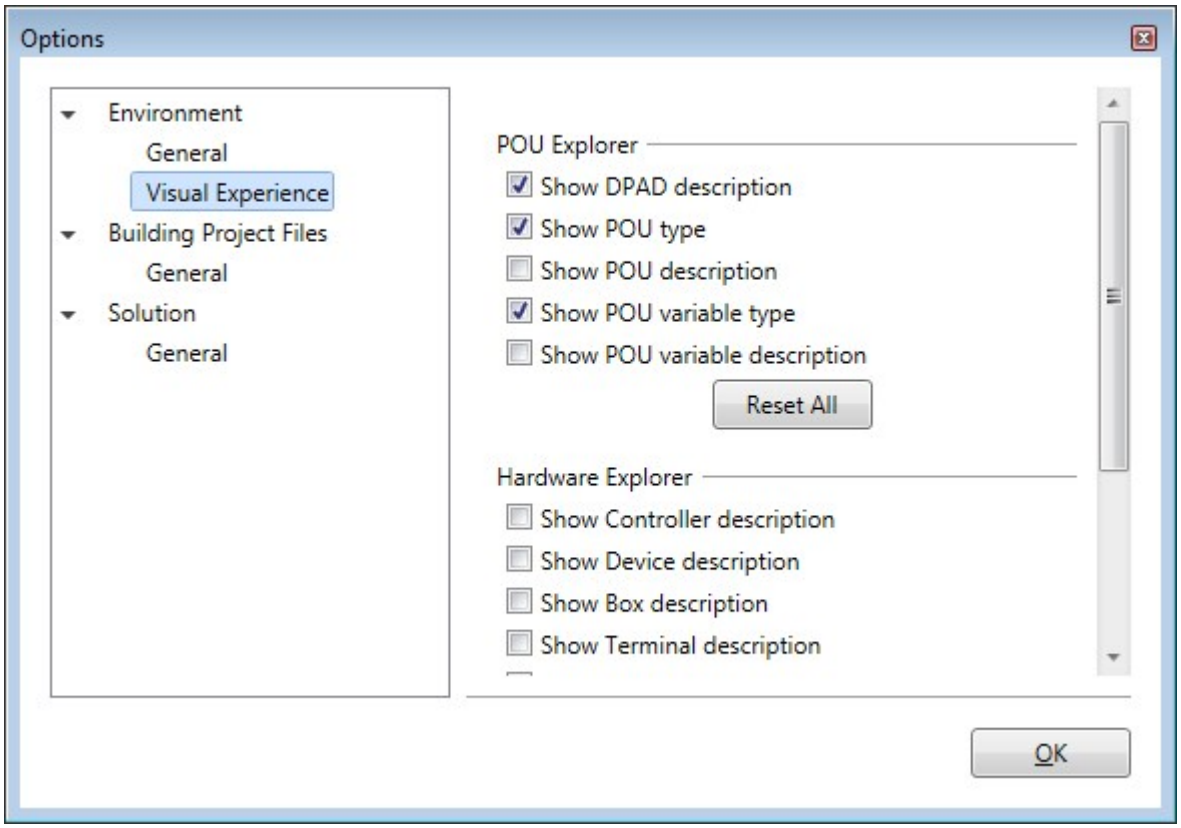
### Environment

### General




| Options section | Function |
|---|---|
| General | On exiting, the BA Project Builder saves the current positions and sizes of the individual tool windows. These are restored to their previous state on restarting. |
| On Startup | You can specify whether the last-used solution is loaded automatically and whether the start page is displayed when starting the BA Project Builder. |
| Recent Solutions | Here you can specify how many of the recently loaded solutions can be reopened directly via the menu item *Solution -> Open Recent*. |
| Message Window | The behavior of the *Output* tool window can be adjusted with these options. |

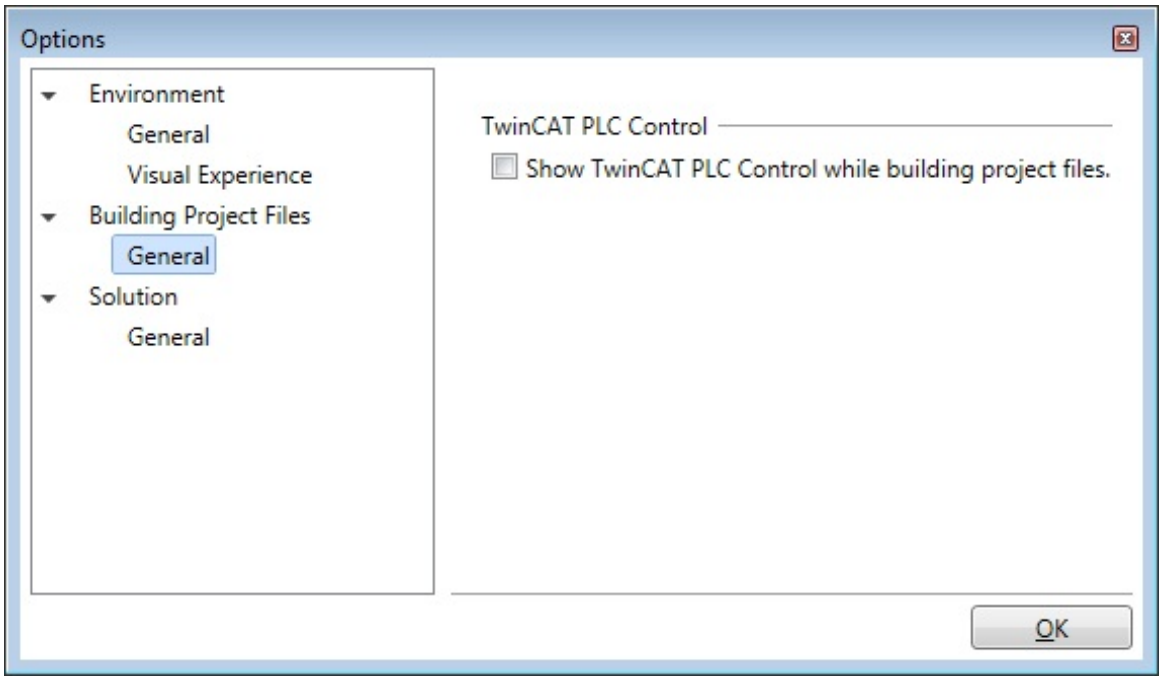
### Visual Experience




| Options section | Function |
|---|---|
| POU Explorer | Additional information on the individual elements can be shown or hid in the *POU Explorer* tool window. |
| Hardware Explorer | Additional information on the individual elements can be shown or hid in the *Hardware Explorer* tool window. |
| Templates Tool Window | Additional information on the individual elements can be shown or hid in the *Templates* tool window. |

### Building Project Files

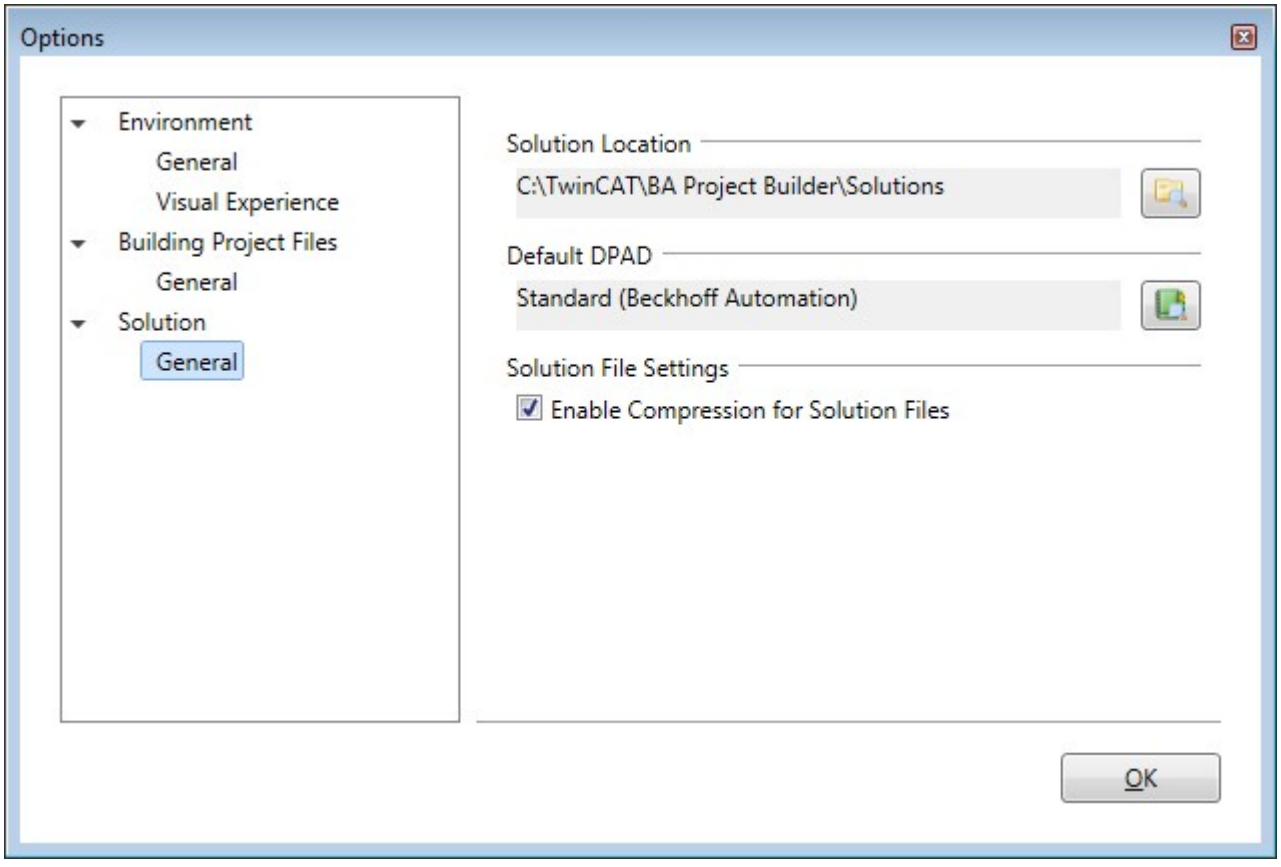
#### General




| Options section | Function |
|---|---|
| TwinCAT PLC Control | The TwinCAT PLC Control is displayed during the generation of the TwinCAT project files. |

#### Solution

#### General



| Options section | Function |
|---|---|
| Solution Location | This folder is preset when generating a new solution. |
| Default DPAD | This DPAD is preset when generating a new solution. |
| Solution File Settings | The solution files are saved in the most space-saving manner possible. This makes the files much smaller, but they can no longer be viewed directly (e.g. in a text editor). |

## 10.4 Creating AddIns

The TwinCAT BA Project Builder offers you the option of creating your own AddIns. AddIns have the possibility to access the internal data management of the TwinCAT BA Project Builder via an object model. This may be a read or a write access.

Detailed documentation on the AddIn API can be obtained here: https://infosys.beckhoff.com/content/1033/tcba/Resources/12269790091/.zip

### 10.4.1 First steps - WPF

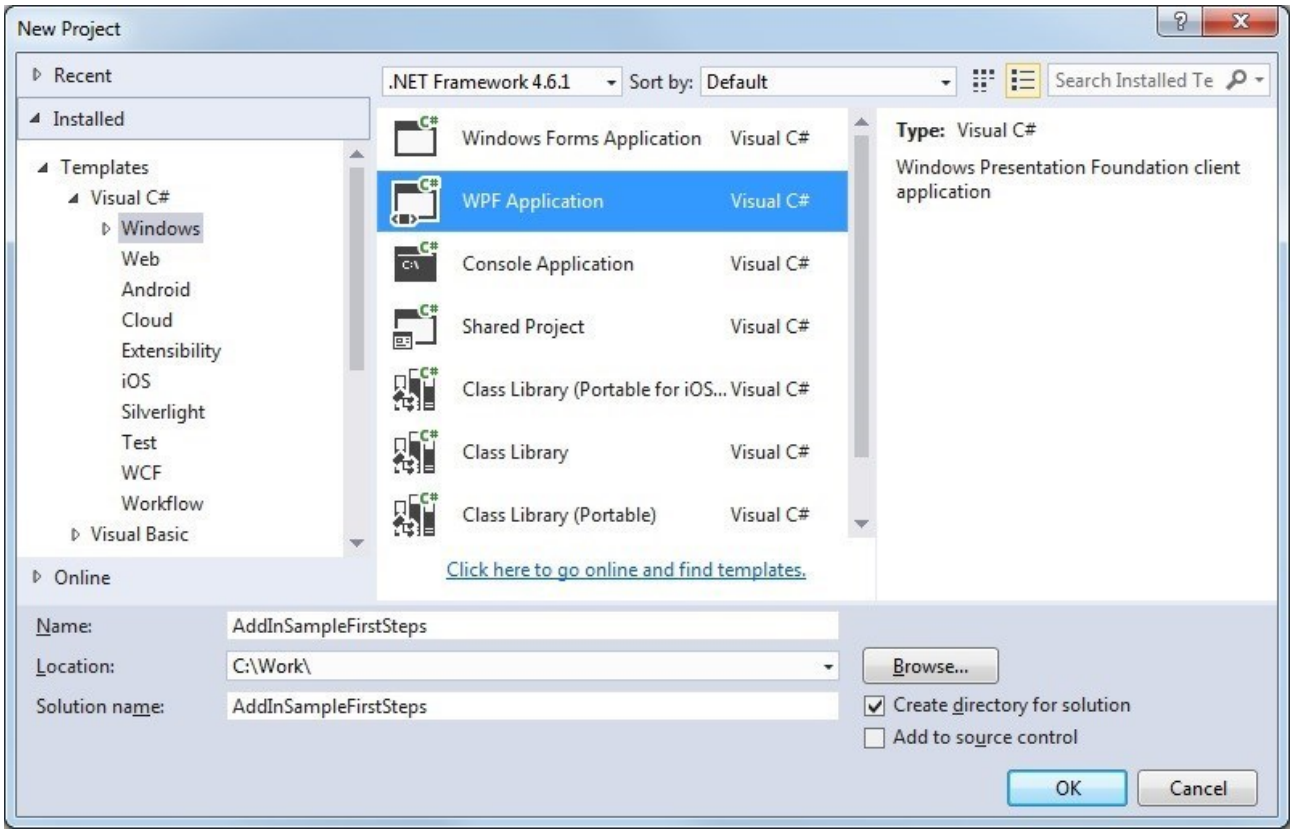
The following executions show all steps necessary to create a basis for the actual add-in development. The complete Microsoft Visual Studio project can be obtained here: https://infosys.beckhoff.com/content/1033/tcba/Resources/12269791499/.zip.

**Prerequisites**

- Microsoft Visual Studio 2013 or higher
- Microsoft .NET Framework 4.6.1 or higher
- Programming skills in C#

**Creating a project**

Start Microsoft Visual Studio and create a new WPF project. Additionally select *.NET Framework 4.6.1*.




Now open the project properties and change the *Output type* setting in the *Application* section to *Class Library*. Check that *Target framework* has been set to *.NET Framework 4.6.1*.

Switch to the *Build* section and enter the path to the AddIn folder of the TwinCAT BA Project Builder under *Output path* (standard is *TwinCAT -> BA Project Builder -> AddIns*). Select the value *x86* under *Platform target*.

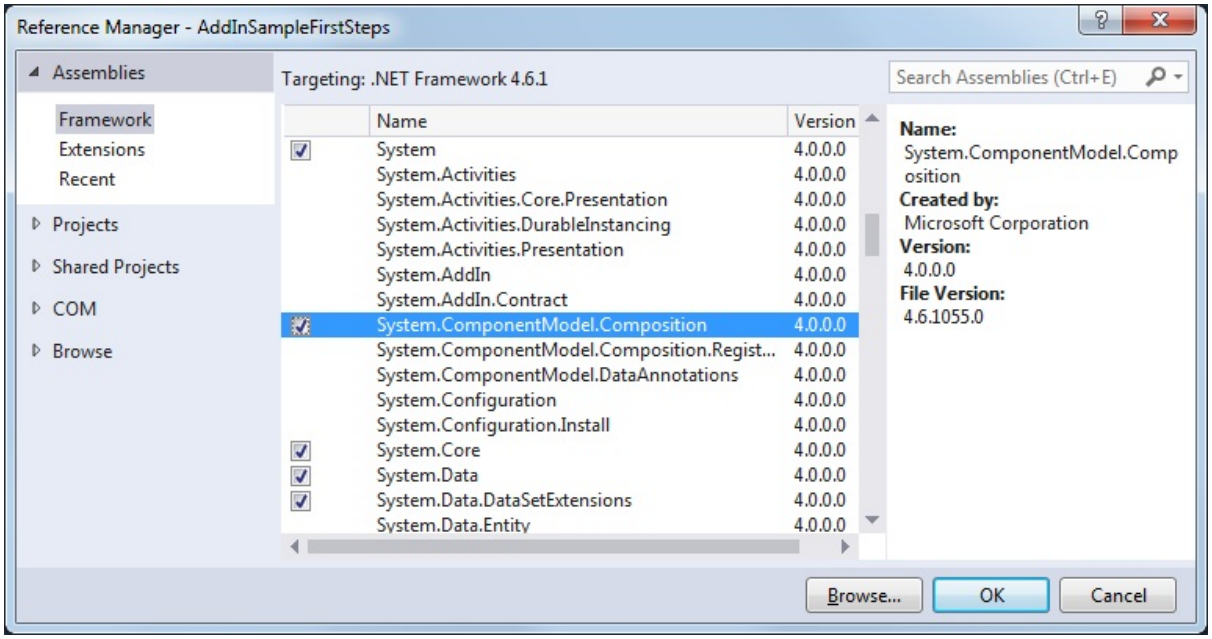
Since this project concerns a class library, you must first define an external program with which the AddIn is to be started in order to start and debug. In the *Debug* section, enter the path to the TwinCAT BA Project Builder next to *Start external program* (standard is *TwinCAT -> BA Project Builder -> TwinCAT.BA.ProjectBuilder.UI.exe*).

Remove the *App.xml* file, which is created as standard, from the project.

### Adding references

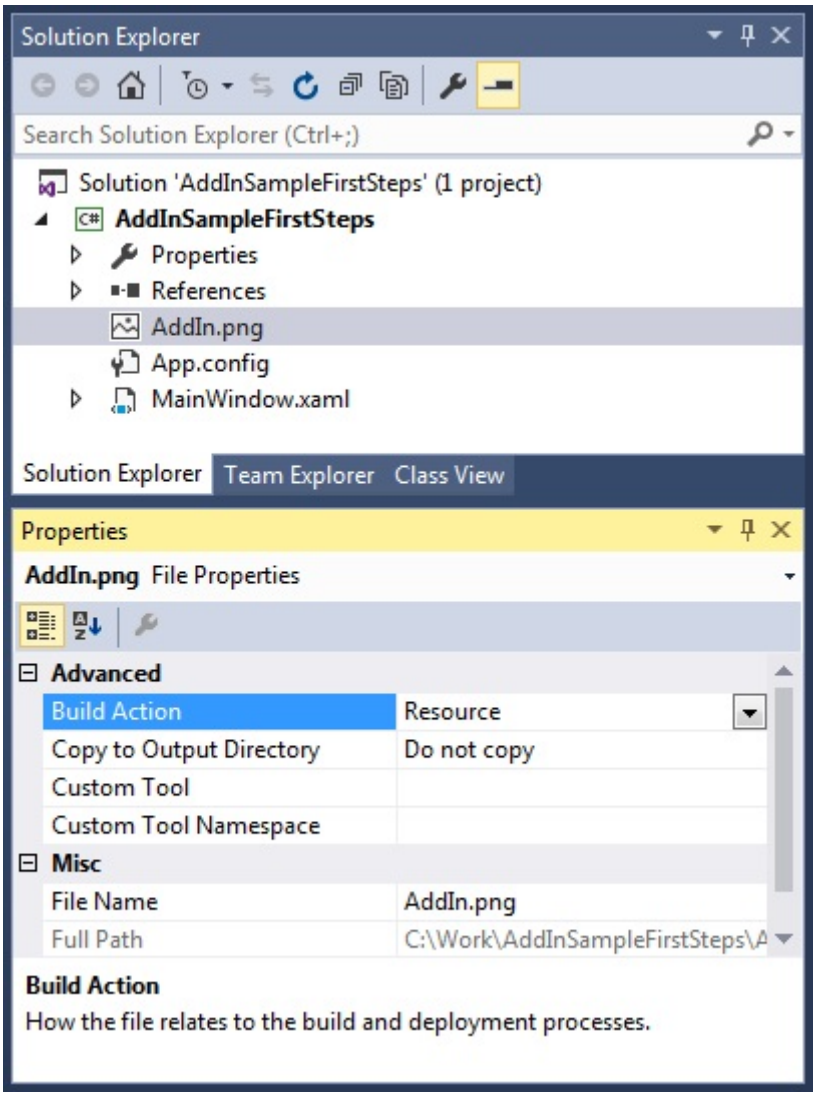
The next step covers the adding of three references.

- **System.ComponentModel.Composition:** The TwinCAT BA Project Builder uses MEF (Managed Extensibility Framework) as the basic technology for extendability by AddIns. All necessary classes for the use of MEF are contained in this DLL.
- **TwinCAT.BA.ProjectBuilder.AddIn.Contract:** This reference provides interfaces that must implement the AddIn so that it can be instanced by the TwinCAT BA Project Builder. This is contained in the file *TwinCAT -> BA Project Builder -> TwinCAT.BA.ProjectBuilder.AddIn.Contract.dll*. For this reference, set the *Copy Local* option to *False*.
- **TwinCAT.BA.ProjectBuilder.AddIn.IL:** This reference provides classes for access to the internal object model of the TwinCAT BA Project Builder. This is contained in the file *TwinCAT -> BA Project Builder -> TwinCAT.BA.ProjectBuilder.AddIn.IL.dll*. For this reference, set the *Copy Local* option to *False*.



### Adding an icon

If desired, add an icon (16 x 16 pixels) to the project and set the *Build Action* property to *Resource*.




### Placing at disposal

Two classes need to be added so that the AddIn can be displayed and loaded in the TwinCAT BA Project Builder. Firstly, a class that is derived from *ContractAttribute* and overwrites several properties. These display the AddIn in the TwinCAT BA Project Builder. Secondly, a class must be created containing the *IContractWpf* interface. The AddIn is loaded via the TwinCAT BA Project Builder using the *GetWindow* method.

Create a new class with the name *AddInEntryMetadata* and derive it from *ContractAttribute*. Add the properties *Description*, *ShortDescription*, *Name, Version* and *Icon*. This information is displayed in the TwinCAT BA Project Builder when selecting the AddIn.

```
using TwinCAT.BA.ProjectBuilder.AddIn.Contract;

namespace AddInSampleFirstSteps
{
    public class AddInEntryMetadata : ContractAttribute
    {
    // Gets the description of the AddIn.
    public override string Description
    {
        get
        {
        return "AddIn Sample FirstSteps of the TwinCAT BA Project Builder for WPF";
        }
    }
```

```
    // Gets the short description of the AddIn.
    public override string ShortDescription
    {
        get
        {
        return "AddIn Sample FirstSteps for WPF";
        }
    }

    // Gets the name of the AddIn.
    public override string Name
    {
        get
        {
        return "Sample FirstSteps WPF";
        }
    }

    // Gets the version of the AddIn.
    public override string Version
    {
        get
        {
        return "1.0.0";
        }
    }

    // Gets the contract metadata icon path.
    // "pack://application:,,,/<- Assembly name ->/;component/<- Subfolder or image/icon file name"
    public override string Icon
    {
        get
        {
        return "pack://application:,,,/AddInSampleFirstSteps-WPF;component/AddIn.png";
        }
    }
    }
}
```

Now create the *AddInEntryPoint* class and implement the *IContractWpf* interface. In addition, the class must be decorated with the attributes *Export* and *AddInEntryMetadata*.

The interface contains the *GetWindow()* method. This is called by the BA Project Builder when the AddIn is to be displayed. Furthermore, two events must be implemented which are necessary for accessing the *Output* window of the BA Project Builder.

```
using System;
using System.ComponentModel.Composition;
using System.Windows;
using TwinCAT.BA.ProjectBuilder.AddIn.Contract;

namespace AddInSampleFirstSteps
{
    [Export(typeof(IContractWpf))] // Marks this class as an entry point for the TwinCAT BA Project
Builder
    [AddInEntryMetadata] // Adds necessary information about the AddIn to this class
    public class AddInEntryPoint : IContractWpf
    {
    private AddInEntryMetadata addInEntryMetadata = null;
    public event EventHandler<ContractLogEventArgs> LogEvent;
    public event EventHandler<ContractClearLogEventArgs> ClearLogEvent;

    public Window GetWindow()
    {
        this.addInEntryMetadata = new AddInEntryMetadata();
        // Passing the main class of the AddIn
        return new MainWindow(this);
    }

    public void RaiseLogEvent(LogLevel level, string message)
    {
        OnLog(new ContractLogEventArgs(this.addInEntryMetadata, level, message));
    }

    // The event-invoking method that derived classes can override.
    private void OnLog(ContractLogEventArgs e)
    {
        // Make a temporary copy of the event to avoid possibility of
        // a race condition if the last subscriber unsubscribes
```

```
        // immediately after the null check and before the event is raised.
        EventHandler<ContractLogEventArgs> handler = this.LogEvent;
        if (handler != null)
        {
        handler(this, e);
        }
    }

    public void RaiseClearLogEvent()
    {
        OnClearLog(new ContractClearLogEventArgs(this.addInEntryMetadata));
    }

    // The event-invoking method that derived classes can override.
    private void OnClearLog(ContractClearLogEventArgs e)
    {
        // Make a temporary copy of the event to avoid possibility of
        // a race condition if the last subscriber unsubscribes
        // immediately after the null check and before the event is raised.
        EventHandler<ContractClearLogEventArgs> handler = this.ClearLogEvent;
        if (handler != null)
        {
        handler(this, e);
        }
    }
    }
}
```

### Programming

The main functionalities are implemented in the *MainWindow* class.

```
using System.Windows;
using TwinCAT.BA.ProjectBuilder.AddIn.IL;
using TwinCAT.BA.ProjectBuilder.AddIn.IL.Hardware;

namespace AddInSampleFirstSteps
{
    public partial class MainWindow : Window
    {
    AddInEntryPoint addInEntryPoint = null;
    BAProjectBuilder projectBuilder = null;

    public MainWindow(AddInEntryPoint addInEntryPoint)
    {
        InitializeComponent();
        // Code here
        this.addInEntryPoint = addInEntryPoint;
        this.projectBuilder = new BAProjectBuilder();
    }
    }
}
```

### Debugging

The TwinCAT BA Project Builder is started automatically with the project. You can execute the AddIn both via the *Tools* menu of the TwinCAT BA Project Builder ...


... and via the toolbar.


In both cases the TwinCAT BA Project Builder creates an instance of the AddIn and calls the *GetWindow* method.

An instance of the *AddInEntryMetadata* class is already created when starting the BA Project Builder and all information necessary for the representation in the TwinCAT BA Project Builder is read out via the Properties.

## 10.4.2 First steps - WinForms

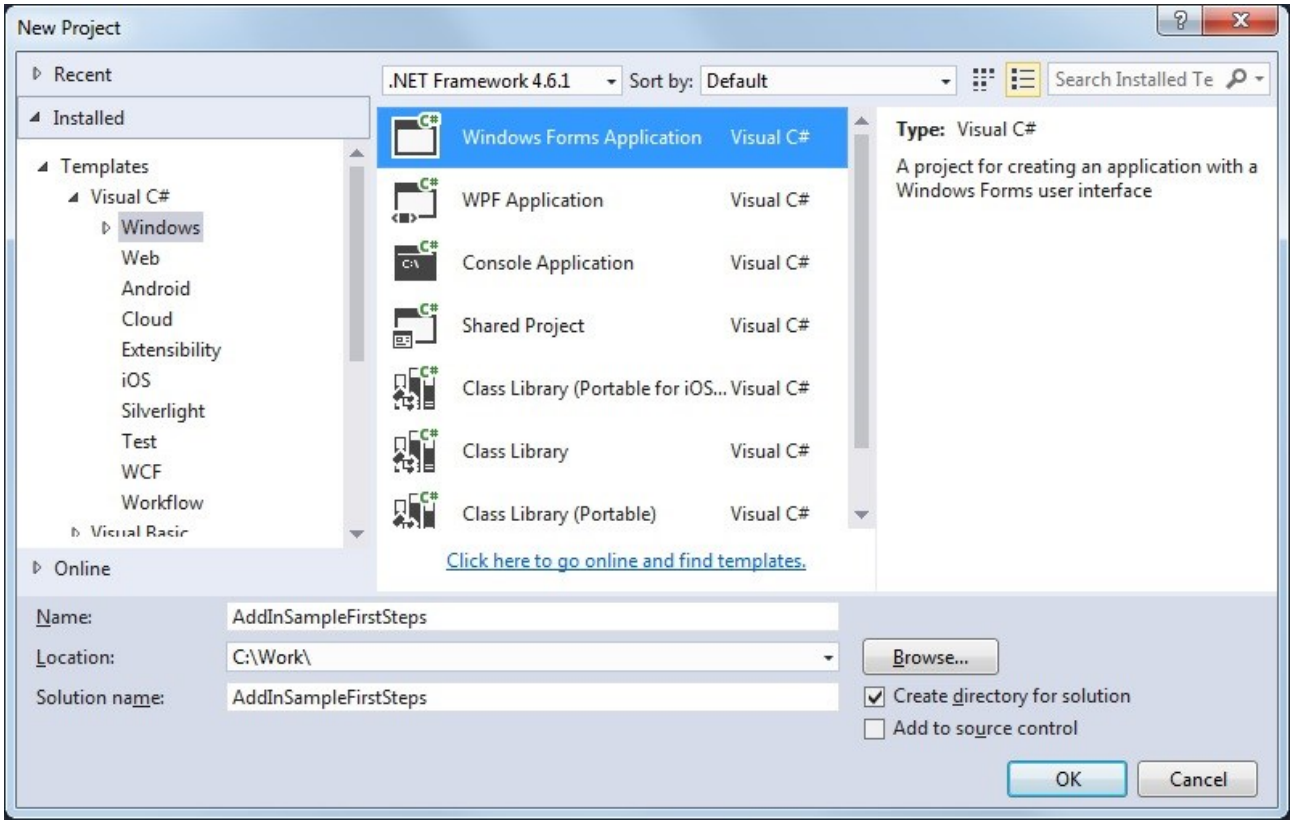
The following executions show all steps necessary to create a basis for the actual add-in development. The complete Microsoft Visual Studio project can be obtained here: https://infosys.beckhoff.com/content/1033/tcba/Resources/12269792907/.zip.

### Prerequisites

- Microsoft Visual Studio 2013 or higher
- Microsoft .NET Framework 4.6.1 or higher
- Programming skills in C#

### Creating a project

Start Microsoft Visual Studio and create a new Windows Forms project. Additionally select *.NET Framework 4.6.1*.




Now open the project properties and change the *Output type* setting in the *Application* section to *Class Library*. Check that *Target framework* has been set to *.NET Framework 4.6.1*.

Switch to the *Build* section and enter the path to the AddIn folder of the TwinCAT BA Project Builder under *Output path* (standard is *TwinCAT -> BA Project Builder -> AddIns*). Select the value *x86* under *Platform target*.

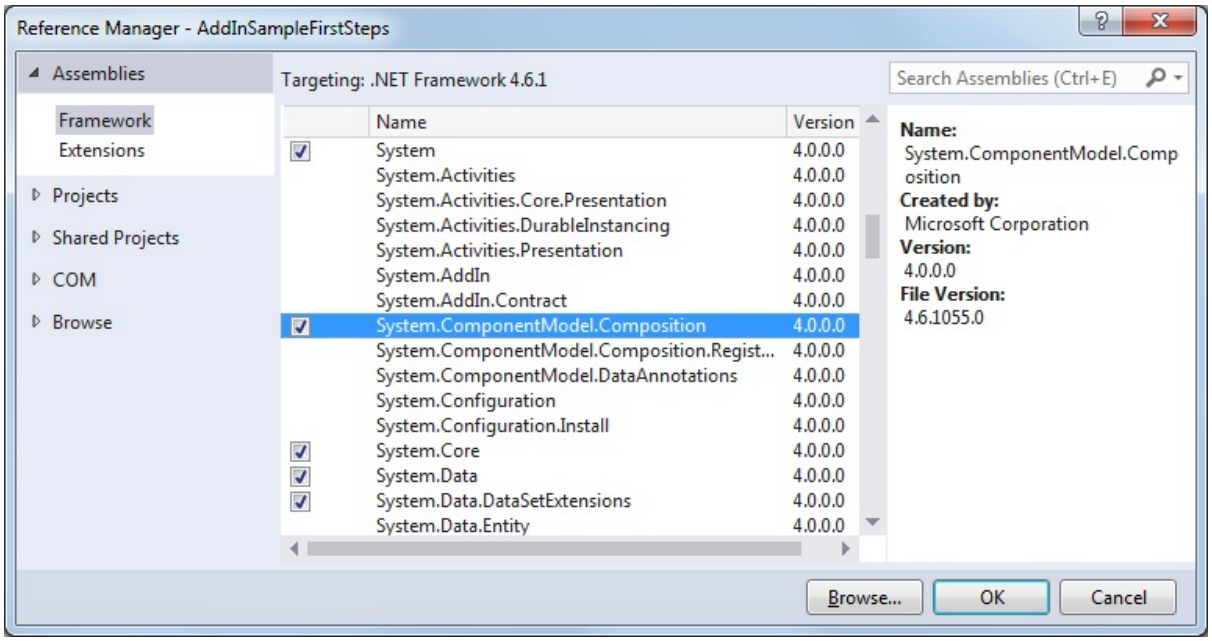
Since this project concerns a class library, you must first define an external program with which the AddIn is to be started to start and debug. In the *Debug* section, enter the path to the TwinCAT BA Project Builder next to *Start external program* (standard is *TwinCAT -> BA Project Builder -> TwinCAT.BA.ProjectBuilder.UI.exe*).

Remove the *Program.cs* file, which is created as standard, from the project.

### Adding references

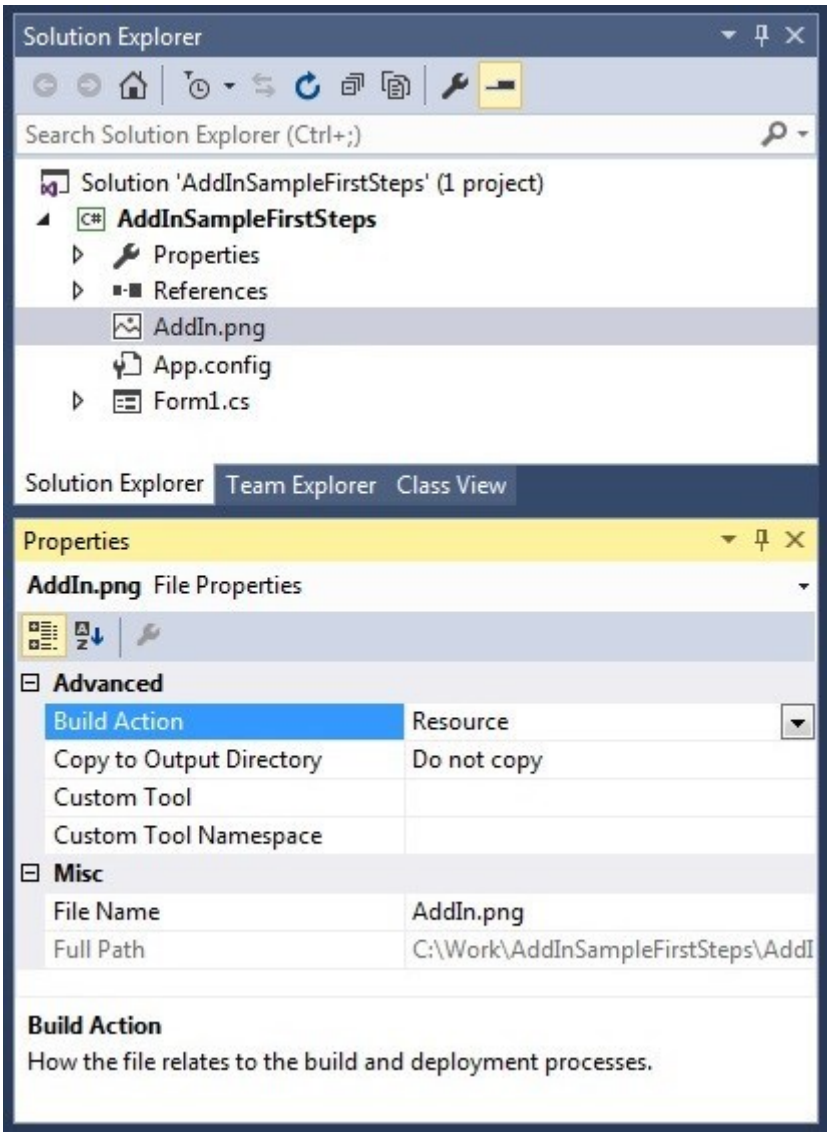
The next step covers the adding of three references.

- **System.ComponentModel.Composition:** The TwinCAT BA Project Builder uses MEF (Managed Extensibility Framework) as the basic technology for extendability by AddIns. All necessary classes for the use of MEF are contained in this DLL.
- **TwinCAT.BA.ProjectBuilder.AddIn.Contract:** This reference provides interfaces that must implement the AddIn so that it can be instanced by the TwinCAT BA Project Builder. This is contained in the file *TwinCAT -> BA Project Builder -> TwinCAT.BA.ProjectBuilder.AddIn.Contract.dll*. For this reference, set the *Copy Local* option to *False*.
- **TwinCAT.BA.ProjectBuilder.AddIn.IL:** This reference provides classes for access to the internal object model of the TwinCAT BA Project Builder. This is contained in the file *TwinCAT -> BA Project Builder -> TwinCAT.BA.ProjectBuilder.AddIn.IL.dll*. For this reference, set the *Copy Local* option to *False*.



### Adding an icon

If desired, add an icon (16 x 16 pixels) to the project and set the *Build Action* property to *Resource*.




### Placing at disposal

Two classes need to be added so that the AddIn can be displayed and loaded in the TwinCAT BA Project Builder. Firstly, a class that is derived from *ContractAttribute* and overwrites several properties. These display the AddIn in the TwinCAT BA Project Builder. Secondly, a class must be created containing the *IContractWinForms* interface. The AddIn is loaded via the TwinCAT BA Project Builder using the *GetForm* method.

Create a new class with the name *AddInEntryMetadata* and derive it from *ContractAttribute*. Add the properties *Description*, *ShortDescription*, *Name, Version* and *Icon*. This information is displayed in the TwinCAT BA Project Builder when selecting the AddIn.

```
using TwinCAT.BA.ProjectBuilder.AddIn.Contract;

namespace AddInSampleFirstSteps
{
    public class AddInEntryMetadata : ContractAttribute
    {
    // Gets the description of the AddIn.
    public override string Description
    {
        get
        {
        return "AddIn Sample FirstSteps of the TwinCAT BA Project Builder for WinForms";
        }
```

```
    }

    // Gets the short description of the AddIn.
    public override string ShortDescription
    {
        get
        {
        return "AddIn Sample FirstSteps for WinForms";
        }
    }

    // Gets the name of the AddIn.
    public override string Name
    {
        get
        {
        return "Sample FirstSteps WinForms";
        }
    }

    // Gets the version of the AddIn.
    public override string Version
    {
        get
        {
        return "1.0.0";
        }
    }

    // Gets or sets the contract metadata icon path.
    // "pack://application:,,,/<- Assembly name ->/;component/<- Subfolder or image/icon file name"
    public override string Icon
    {
        get
        {
        return "pack://application:,,,/AddInSampleFirstSteps;component/AddIn.png";
        }
    }
    }
}
```

Now create the *AddInEntryPoint* class and implement the *IContractWinForms* interface. In addition, the class must be decorated with the attributes *Export* and *AddInEntryMetadata*.

The interface contains the *GetForm()* method. This is called by the BA Project Builder when the AddIn is to be displayed. Furthermore, two events must be implemented which are necessary for accessing the *Output* window of the BA Project Builder.

```
using System;
using System.ComponentModel.Composition;
using System.Windows.Forms;
using TwinCAT.BA.ProjectBuilder.AddIn.Contract;

namespace AddInSampleFirstSteps
{
    [Export(typeof(IContractWinForms))] // Marks this class as an entry point for the TwinCAT BA Pro
ject Builder
    [AddInEntryMetadata] // Adds necessary information about the AddIn to this class
    public class AddInEntryPoint : IContractWinForms
    {
    private AddInEntryMetadata addInEntryMetadata = null;
    public event EventHandler<ContractLogEventArgs> LogEvent;
    public event EventHandler<ContractClearLogEventArgs> ClearLogEvent;

    public Form GetForm()
    {
        this.addInEntryMetadata = new AddInEntryMetadata();
        // Passing the main class of the AddIn
        return new Form1(this);
    }

    public void RaiseLogEvent(LogLevel level, string message)
    {
        OnLog(new ContractLogEventArgs(this.addInEntryMetadata, level, message));
    }

    // The event-invoking method that derived classes can override.
    private void OnLog(ContractLogEventArgs e)
    {
```

```
        // Make a temporary copy of the event to avoid possibility of
        // a race condition if the last subscriber unsubscribes
        // immediately after the null check and before the event is raised.
        EventHandler<ContractLogEventArgs> handler = this.LogEvent;
        if (handler != null)
        {
        handler(this, e);
        }
    }

    public void RaiseClearLogEvent()
    {
        OnClearLog(new ContractClearLogEventArgs(this.addInEntryMetadata));
    }

    // The event-invoking method that derived classes can override.
    private void OnClearLog(ContractClearLogEventArgs e)
    {
        // Make a temporary copy of the event to avoid possibility of
        // a race condition if the last subscriber unsubscribes
        // immediately after the null check and before the event is raised.
        EventHandler<ContractClearLogEventArgs> handler = this.ClearLogEvent;
        if (handler != null)
        {
        handler(this, e);
        }
    }
    }
}
```

### Programming

The main functionalities are implemented in the *Form1* class.

```
using System.Windows.Forms;
using TwinCAT.BA.ProjectBuilder.AddIn.IL;

namespace AddInSampleFirstSteps
{
    public partial class Form1 : Form
    {
    AddInEntryPoint addInEntryPoint = null;
    BAProjectBuilder projectBuilder = null;

    public Form1(AddInEntryPoint addInEntryPoint)
    {
        InitializeComponent();
        // Code here
        this.addInEntryPoint = addInEntryPoint;
        this.projectBuilder = new BAProjectBuilder();
    }
    }
}
```

### Debugging

The TwinCAT BA Project Builder is started automatically with the project. You can execute the AddIn both via the *Tools* menu of the TwinCAT BA Project Builder ...


... and via the toolbar.


In both cases the TwinCAT BA Project Builder creates an instance of the AddIn and calls the *GetForm* method.

An instance of the *AddInEntryMetadata* class is already created when starting the BA Project Builder and all information necessary for the representation in the TwinCAT BA Project Builder is read out via the Properties.

## 10.4.3 Samples

| How To |
|---|
| Sample 01 - access to the output window [▶ 755] |
| Sample 02 - readout of a template description [▶ 755] |
| Sample 03 - list of all terminal descriptions [▶ 756] |
| Sample 04 - creating hardware [▶ 757] |

### 10.4.3.1 Sample 01: access to the output window

The IContract interface provides event handlers with which messages can be written into the output window or the display can be cleared. IContract is handed down to the IContractWpf interface and implemented by the AddInEntryPoint class.

**Sample C#**

```
public partial class MainWindow : Window
{
  AddInEntryPoint addInEntryPoint = null;

  public MainWindow(AddInEntryPoint addInEntryPoint)
  {
    InitializeComponent();
    // Code here
    this.addInEntryPoint = addInEntryPoint;
  }

  private void buttonClear_Click(object sender, RoutedEventArgs e)
  {
    this.addInEntryPoint.RaiseClearLogEvent();
  }

  private void buttonLogEvent_Click(object sender, RoutedEventArgs e)
  {
    this.addInEntryPoint.RaiseLogEvent((LogLevel)comboBoxLogLevelCombo.SelectedIndex, textBoxMessage
.Text);
  }
}
```

**Download**

https://infosys.beckhoff.com/content/1033/tcba/Resources/12269794315/.zip

### 10.4.3.2 Sample 02: readout of a template description

General information about a template is made available by the TemplateDescription class. The TemplateDescriptionProvider class and its static method GetTemplateDecription return the template description for the desired template.

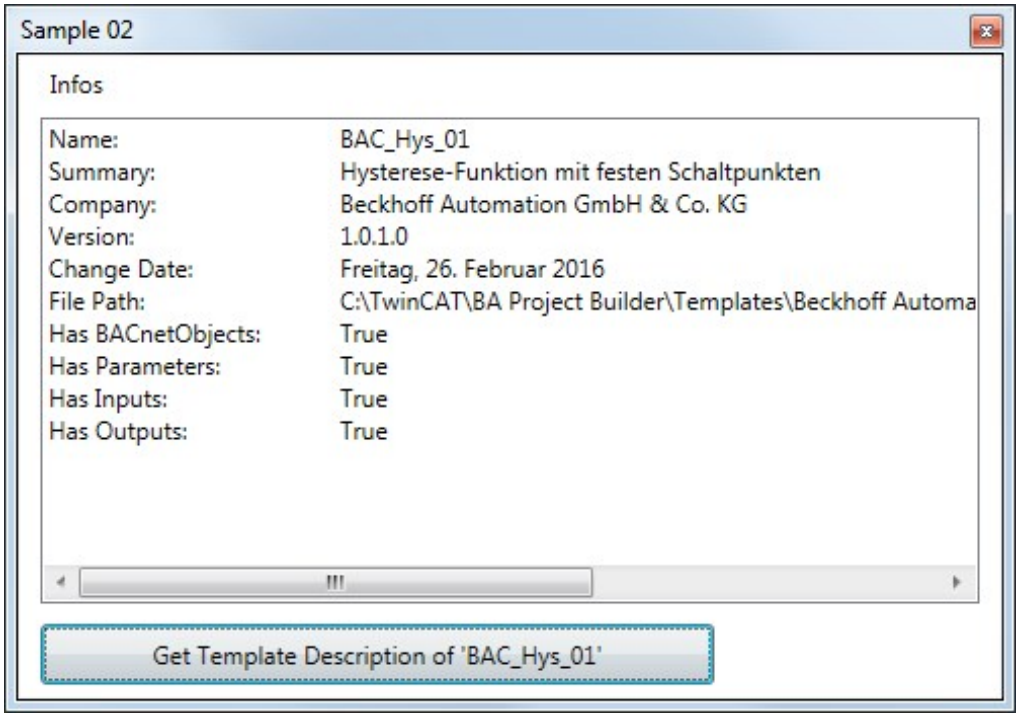
**Sample C#**

```
 public partial class MainWindow : Window
{
  BAProjectBuilder projectBuilder = null;

  public MainWindow(AddInEntryPoint addInEntryPoint)
  {
    InitializeComponent();
    // Code here
    this.projectBuilder = new BAProjectBuilder();
  }

  private void buttonGetInfos_Click(object sender, RoutedEventArgs e)
```

```
  {
    TemplateDescription td = this.projectBuilder.TemplateDescriptionProvider.GetTemplateDescription(
"BAC_Hys_01");
    List<string> items = new List<string>();
    items.Add("Name:\t\t\t" + td.Name);
    items.Add("Summary:\t\t" + td.Summary);
    items.Add("Company:\t\t" + td.Company);
    items.Add("Version:\t\t\t" + td.Version.ToString());
    items.Add("Change Date:\t\t" + td.ChangeDate.ToLongDateString());
    items.Add("File Path:\t\t\t" + td.FilePath);
    items.Add("Has BACnetObjects:\t" + td.HasBACnetObjects);
    items.Add("Has Parameters:\t\t" + td.HasParameters);
    items.Add("Has Inputs:\t\t" + td.HasInputs);
    items.Add("Has Outputs:\t\t" + td.HasOutputs);
    listBoxInfos.ItemsSource = items;
  }
}
```




**Download**

https://infosys.beckhoff.com/content/1033/tcba/Resources/12269795723/.zip

### 10.4.3.3 Sample 03: list of all terminal descriptions

The HardwareDescriptionProvider class makes all terminal descriptions available via the read-only collection TerminalDescriptions. The desired terminal description can be read out using the static method GetTerminalDecription.

**Sample C#**

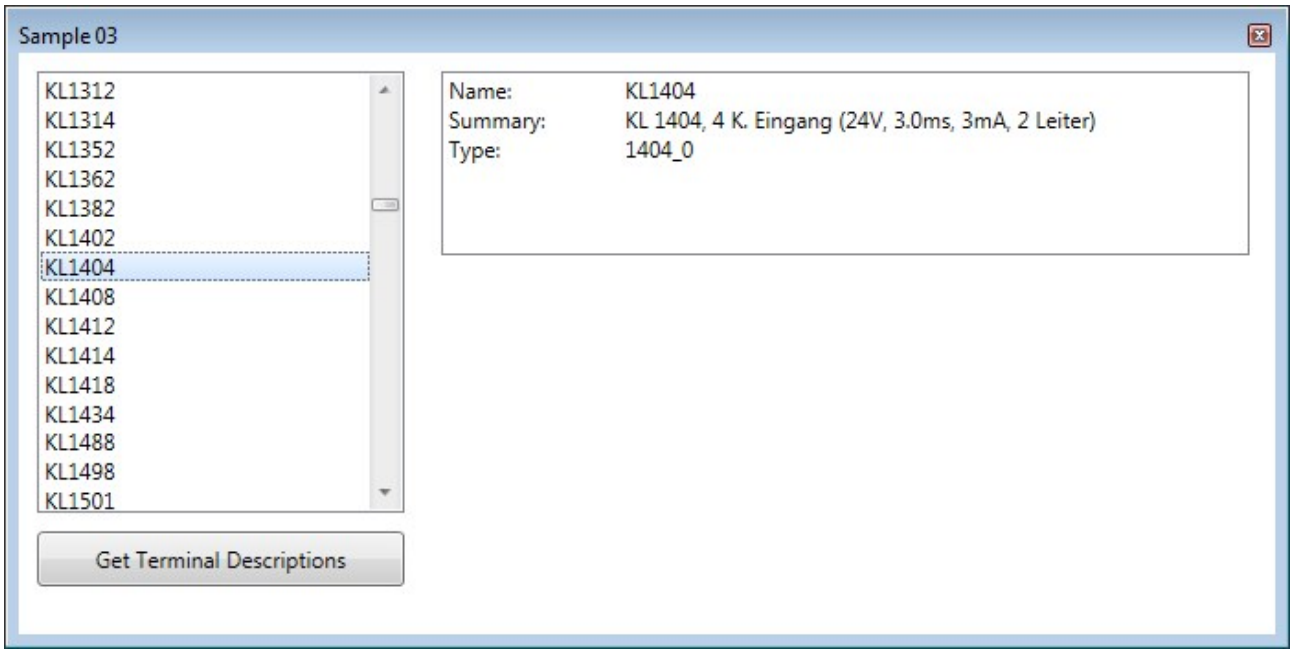
```
public partial class MainWindow : Window
{
  BAProjectBuilder projectBuilder = null;

  public MainWindow(AddInEntryPoint addInEntryPoint)
  {
    InitializeComponent();
    // Code here
    this.projectBuilder = new BAProjectBuilder();
  }

  private void buttonGetDescriptions_Click(object sender, RoutedEventArgs e)
  {
    List<string> items = new List<string>();
```

```
    foreach (TerminalDescription td in this.projectBuilder.HardwareDescriptionProvider.TerminalDescr
iptions)
    {
      items.Add(td.Abbreviation);
    }
    listBoxDescriptions.ItemsSource = items;
  }

  private void listBoxDescriptions_SelectionChanged(object sender, System.Windows.Controls.Selection
ChangedEventArgs e)
  {
    TerminalDescription td = this.projectBuilder.HardwareDescriptionProvider.GetTerminalDescription(
(string)listBoxDescriptions.SelectedItem);
    List<string> items = new List<string>();
    items.Add("Name:\t" + td.Name);
    items.Add("Summary:\t" + td.Summary);
    items.Add("Type:\t\t" + td.Type);
    listBoxInfo.ItemsSource = items;
  }
}
```




**Download**

https://infosys.beckhoff.com/content/1033/tcba/Resources/12269797131/.zip

### 10.4.3.4 Sample 04: creating hardware

The following sample creates a hardware configuration with controllers, PLC tasks, bus couplers and Bus Terminals.

**Sample C#**

```
public partial class MainWindow : Window
{
  BAProjectBuilder projectBuilder = null;

  public MainWindow(AddInEntryPoint addInEntryPoint)
  {
    InitializeComponent();
    // Code here
    this.projectBuilder = new BAProjectBuilder();
  }

  private void buttonCreateHardware_Click(object sender, RoutedEventArgs e)
  {
    // add a CX2000 with the IO devices of the controller description to the solution
    ////////////////////////////////////////////////////////////////////////////////
```

```
    ControllerDescription descriptionControllerCX2000 = this.projectBuilder.HardwareDescriptionProvi
der.GetControllerDescription("CX20x0");
    IController controllerCX2000 = this.projectBuilder.Solution.HardwareConfig.CreateController(desc
riptionControllerCX2000);
    controllerCX2000.Name = "my CX2020";

    // the CX20x0 comes with a terminal device, terminal coupler and two tasks
    // get the task and change the name and the cycle time
    ITask task01 = controllerCX2000.Tasks[0];
    task01.Name = "Main";
    task01.Interval = 45;

    ITask task02 = controllerCX2000.Tasks[1];
    task02.Name = "Background";
    task02.Interval = 15;

    // get the terminal device
    IBoxDevice terminalDeviceCX2000 = controllerCX2000.Devices[0] as IBoxDevice;
    terminalDeviceCX2000.Name = "my CX2000/CX5000 Terminal Device";

    // get the terminal coupler
    ITerminalCouplerBox terminalCouplerBoxCX2000 = terminalDeviceCX2000.Boxes[0] as ITerminalCoupler
Box;
    terminalCouplerBoxCX2000.Name = "my CX2000/CX5000 Terminal Coupler";

    // add terminals to the terminal coupler
    ITerminal terminalKL1012 = terminalCouplerBoxCX2000.CreateTerminal(this.projectBuilder.HardwareD
escriptionProvider.GetTerminalDescription("KL1012"));
    ITerminal terminalKL1404 = terminalCouplerBoxCX2000.CreateTerminal(this.projectBuilder.HardwareD
escriptionProvider.GetTerminalDescription("KL1404"));
    ITerminal terminalKL2404 = terminalCouplerBoxCX2000.CreateTerminal(this.projectBuilder.HardwareD
escriptionProvider.GetTerminalDescription("KL2404"));

    // link the channels of the terminals to the Main task
    task01.CreateLink(terminalKL1012.Channels[0]);
    task01.CreateLink(terminalKL1012.Channels[1]);

    task01.CreateLink(terminalKL1404.Channels[0]);
    task01.CreateLink(terminalKL1404.Channels[1]);
    task01.CreateLink(terminalKL1404.Channels[2]);
    task01.CreateLink(terminalKL1404.Channels[3]);

    // link the channels of the KL2404 to the Background task
    task02.CreateLink(terminalKL2404.Channels[0]);
    task02.CreateLink(terminalKL2404.Channels[1]);
    task02.CreateLink(terminalKL2404.Channels[2]);
    task02.CreateLink(terminalKL2404.Channels[3]);

    // add a BACnet Device
    BACnetEthernetDevice bacnetDevice = controllerCX2000.CreateDevice(this.projectBuilder.HardwareDe
scriptionProvider.GetDeviceDescription("BACnet Ethernet Device Rev 12")) as BACnetEthernetDevice;
    bacnetDevice.Name = "my BACnet Device";

    // add a CP without(!) the IO devices of the controller description to the solution
    ////////////////////////////////////////////////////////////////////////////////
    ControllerDescription controllerDescriptionCP = this.projectBuilder.HardwareDescriptionProvider.
GetControllerDescription("IPC/CP ARM");
    IController controllerCP = this.projectBuilder.Solution.HardwareConfig.CreateController(controll
erDescriptionCP, false);
    controllerCP.Name = "my CP";

    // add the task and change the name and the cycle time
    ITask task03 = controllerCP.CreateTask(this.projectBuilder.HardwareDescriptionProvider.GetTaskDe
scription("Standard"));
    task03.Name = "Main";
    task03.Interval = 20;

    // add the RT ethernet device
    IBoxDevice ethernetRTDeviceCP = controllerCP.CreateDevice(this.projectBuilder.HardwareDescriptio
nProvider.GetDeviceDescription("Real Time Ethernet Protocol")) as IBoxDevice;
    ethernetRTDeviceCP.Name = "my RT Ethernet Device";

    // add the BK9000 coupler
    ITerminalCouplerBox terminalCouplerBK9000 = ethernetRTDeviceCP.CreateBox(this.projectBuilder.Har
dwareDescriptionProvider.GetBoxDescription("BK9000 Ethernet Fieldbus Coupler")) as ITerminalCouplerB
ox;
    terminalCouplerBK9000.Name = "my BK9000 Ethernet Coupler";

    // add terminals to the terminal coupler
    ITerminal terminalKL3204 = terminalCouplerBK9000.CreateTerminal(this.projectBuilder.HardwareDesc
```

```
riptionProvider.GetTerminalDescription("KL3204"));

    // link the channels of the terminals to the Main task
    task03.CreateLink(terminalKL3204.Channels[0]);
    task03.CreateLink(terminalKL3204.Channels[1]);
    task03.CreateLink(terminalKL3204.Channels[2]);
    task03.CreateLink(terminalKL3204.Channels[3]);

    this.Close();
  }
}
```

### Download

https://infosys.beckhoff.com/content/1033/tcba/Resources/12269798539/.zip

# 11 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

### Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

### Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

### Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

| | |
|---|---|
| Hotline: | +49 5246 963-157 |
| e-mail: | support@beckhoff.com |

### Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

| | |
|---|---|
| Hotline: | +49 5246 963-460 |
| e-mail: | service@beckhoff.com |

### Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

| | |
|---|---|
| Phone: | +49 5246 963-0 |
| e-mail: | info@beckhoff.com |
| web: | www.beckhoff.com |

More Information:
**www.beckhoff.com/ts8040**

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com