Manual

TC3 Building Automation

TwinCAT 3

Version: 1.0
Date: 2018-11-15
Order No.: TF8040
# Table of contents

1 Foreword .......................................................................................................................... 5  
   1.1 Notes on the documentation ....................................................................................... 5  
   1.2 Safety instructions ...................................................................................................... 6  
2 Introduction ....................................................................................................................... 7  
   2.1 Target groups ............................................................................................................. 7  
   2.2 Requirement profile .................................................................................................... 7  
   2.3 Hardware requirements ............................................................................................... 7  
3 General Information .......................................................................................................... 8  
4 Integration in TwinCAT ..................................................................................................... 9  
   4.1 System requirements .................................................................................................. 9  
   4.2 Installation .................................................................................................................. 9  
   4.3 Licensing ................................................................................................................... 12  
5 Programming ................................................................................................................... 18  
   5.1 POUs .......................................................................................................................... 18  
      5.1.1 Air conditioning equipment .................................................................................. 18  
      5.1.2 Provision of hot water ....................................................................................... 27  
      5.1.3 Room automation .............................................................................................. 31  
      5.1.4 System ............................................................................................................... 110  
      5.1.5 Universal ........................................................................................................... 119  
   5.2 DUTs .......................................................................................................................... 179  
      5.2.1 Enums ............................................................................................................... 179  
      5.2.2 Structures ......................................................................................................... 180  
   5.3 GVLs .......................................................................................................................... 186  
      5.3.1 Constants .......................................................................................................... 186  
      5.3.2 Parameter .......................................................................................................... 187  
6 Appendix .......................................................................................................................... 188  
   6.1 Support and Service ................................................................................................... 188
1 Foreword

1.1 Notes on the documentation

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

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

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

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

Disclaimer

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

We reserve the right to revise and change the documentation at any time and without prior announcement. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.

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

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com/danger.png" alt="DANGER" /></td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="https://example.com/warning.png" alt="WARNING" /></td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="https://example.com/caution.png" alt="CAUTION" /></td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td><img src="https://example.com/note.png" alt="NOTE" /></td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer

This symbol indicates information that contributes to better understanding.
2 Introduction

By using the function TC3 BA (TwinCAT3 Building Automation), all PLC programs, including the central heating plant, the air conditioning plant and the room automation functions can be programmed with TwinCAT PLC Control and are then available as function blocks within the building automation libraries.

The PID controllers, the sequence controllers and the sequence linkers required for the TwinCAT 3 Building Automation library can be found in the pre-installed library TC3_BA_Common.

2.1 Target groups

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

2.2 Requirement profile

The user requires basic knowledge of the following.

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

2.3 Hardware requirements

The software 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.
3 General Information

Further libraries required

For PC systems and Embedded PCs (CXxxxx):

- Tc2_IoFunctions
- Tc2_Math
- Tc2_Standard
- Tc2_System
- Tc2.Utilities
- Tc3_BACommon
4 Integration in TwinCAT

4.1 System requirements

<table>
<thead>
<tr>
<th>Technical data</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Windows 7/10, Windows Embedded Standard 7, Windows CE7</td>
</tr>
<tr>
<td>Target platform</td>
<td>PC architecture (x86, x64 or ARM)</td>
</tr>
<tr>
<td>TwinCAT version</td>
<td>TwinCAT 3.1 build 4022.16 or higher</td>
</tr>
<tr>
<td>Required TwinCAT setup level</td>
<td>TwinCAT 3 XAE, XAR</td>
</tr>
<tr>
<td>Required TwinCAT license</td>
<td>TF8040 TC3 Building Automation</td>
</tr>
</tbody>
</table>

4.2 Installation

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

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

1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.
   - The installation dialog opens.

2. Accept the end user licensing agreement and click **Next**.
3. Enter your user data.

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

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

6. Confirm the dialog with **Yes**.
7. Select **Finish** to exit the setup.

![Beckhoff Setup Completed]

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

### 4.3 Licensing

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

The licensing of a TwinCAT 3 Function is described below. The description is divided into the following sections:

- Licensing a 7-day test version [12]
- Licensing a full version [14]

Further information on TwinCAT 3 licensing can be found in the “Licensing” documentation in the Beckhoff Information System (TwinCAT 3 > Licensing).

#### Licensing a 7-day test version

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.

⇒ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.

   ![Solution Explorer](image)

   - The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF6420: TC3 Database Server").

   ![Manage Licenses](image)

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

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

   ![Order Information](image)
7. Click 7-Day Trial License... to activate the 7-day trial license.

![Image of license activation dialog]

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

8. Enter the code exactly as it appears, confirm it and acknowledge the subsequent dialog indicating successful activation.

⇒ In the tabular overview of licenses, the license status now indicates the expiration date of the license.

9. Restart the TwinCAT system.

⇒ The 7-day trial version is enabled.

Licensing a full version

1. Start the TwinCAT 3 development environment (XAE).

2. Open an existing TwinCAT 3 project or create a new project.

3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the Choose Target System drop-down list in the toolbar.

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

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

![Image of Solution Explorer with License selected]

⇒ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. “TE1300: TC3 Scope View Professional”).

![Manage Licenses Tab](image1)

6. Open the **Order Information** tab.

   ![Order Information Tab](image2)

   - In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

   ![License Overview](image3)

A TwinCAT 3 license is generally linked to two indices describing the platform to be licensed:
- **System ID**: Uniquely identifies the device
- **Platform level**: Defines the performance of the device

The corresponding **System Id** and **Platform** fields cannot be changed.
7. Enter the order number (License Id) for the license to be activated and optionally a separate order number (Customer Id), plus an optional comment for your own purposes (Comment). If you do not know your Beckhoff order number, please contact your Beckhoff sales contact.

8. Click the Generate File... button to create a License Request File for the listed missing license.

   - A window opens, in which you can specify where the License Request File is to be stored. (We recommend accepting the default settings.)

9. Select a location and click Save.

   - A prompt appears asking whether you want to send the License Request File to the Beckhoff license server for verification:

   ![Send license request to Beckhoff]

   - Click Yes to send the License Request File. A prerequisite is that an email program is installed on your computer and that your computer is connected to the internet. When you click Yes, the system automatically generates a draft email containing the License Request File with all the necessary information.

   - Click No if your computer does not have an email program installed on it or is not connected to the internet. Copy the License Request File onto a data storage device (e.g. a USB stick) and send the file from a computer with internet access and an email program to the Beckhoff license server (tclicense@beckhoff.com) by email.

10. Send the License Request File.

    - The License Request File is sent to the Beckhoff license server. After receiving the email, the server compares your license request with the specified order number and returns a License Response File by email. The Beckhoff license server returns the License Response File to the same email address from which the License Request File was sent. The License Response File differs from the License Request File only by a signature that documents the validity of the license file content. You can view the contents of the License Response File with an editor suitable for XML files (e.g. “XML Notepad”). The contents of the License Response File must not be changed, otherwise the license file becomes invalid.

11. Save the License Response File.
12. To import the license file and activate the license, click **License Response File...** in the **Order Information** tab.

13. Select the License Response File in your file directory and confirm the dialog.

![Image of License Response File selection](image)

- The License Response File is imported and the license it contains is activated. Existing demo licenses will be removed.

14. Restart the TwinCAT system.

- The license becomes active when TwinCAT is restarted. The product can be used as a full version. During the TwinCAT restart the license file is automatically copied to the directory `\TwinCAT\3.1\Target\License` on the respective target system.
Programming

5 Programming

5.1 POUs

5.1.1 Air conditioning equipment

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_FrstPrtc [18]</td>
<td>Monitoring of frost alarm and emergency heating</td>
</tr>
<tr>
<td>FB_BA_HX [20]</td>
<td>Calculation of dew point temperature, specific enthalpy and absolute humidity</td>
</tr>
<tr>
<td>FB_BA_NgtCol [21]</td>
<td>Summer night cooling</td>
</tr>
<tr>
<td>FB_BA_RcvMonit [22]</td>
<td>Function block for calculating the efficiency of an energy recovery system</td>
</tr>
<tr>
<td>FB_BA_SPSupvis [25]</td>
<td>Function block for processing and checking the lower and upper setpoint of a supply air humidity or temperature control</td>
</tr>
</tbody>
</table>

5.1.1.1 FB_BA_FrstPrtc

The function block is used for frost monitoring of a heating coil in an air conditioning system.

A frost risk is present, if the input \( b\text{Frst} \) is TRUE. The frost alarm must be linked in the plant program such that the plant is switched off immediately, the heater valve opens, and the heater pump is switched on.

If there is risk of frost, the output \( b\text{On} \) is set, and \( udi\text{T1}\text{_sec} \) (seconds) is started. If the frost risk remains \( (b\text{Frst}=\text{TRUE}) \) after \( udi\text{T1\_sec} \) has elapsed, \( b\text{On} \) remains set. It can only be reset at input \( b\text{Rst} \).

If the frost alarm ceases due to activation of the heating coil within the time \( udi\text{T1}\text{\_sec} \) \( (b\text{Frst}=\text{FALSE}) \), the plant automatically restarts. For the plant restart \( b\text{On} \) becomes FALSE, and at output \( b\text{HWRst} \) a pulse for acknowledgement of a latching circuit in the control cabinet is issued. With the restart a second monitoring period \( udi\text{T2}\text{\_sec} \) (seconds) is initiated. If another frost alarm occurs within this period, the plant is permanently locked. \( b\text{On} \) remains set until the frost alarm has been eliminated and \( b\text{Rst} \) has been acknowledged.

In a scenario where frost alarms recur with time offsets that are greater than \( udi\text{T2}\text{\_sec} \), theoretically the plant would keep restarting automatically. In order to avoid this, the restarts within the function block are counted. The parameter \( udi\text{AlmCnt} \) can be used to set the number of possible automatic restart between 0 and 4.

An acknowledgement at input \( b\text{Rst} \) resets the alarm memory within the function block to zero.
Example:

t0 = frost alarm at input bFrst, alarm message at output bOn, start of timer T1 (udiT1_sec [s])
t1 = frost alarm off, resetting of bOn, output of hardware pulse, start of timer T2 (udiT2_sec [s]), plant restart

t2 = further frost alarm within T2, alarm message at bOn, start of timer T1, locking of the frost alarm

t3 = frost alarm off.
t4 = acknowledgement of the alarm at bRst, resetting of bOn.

VAR_INPUT

bFrst : BOOL;
udiT1_sec : UDINT;
udiT2_sec : UDINT;
udiAlmCnt : UDINT;
bRst : BOOL;

bFrst: Connection for frost events on the air and water side.

udiT1_sec: Timer for restart delays [s]. Internally limited to a minimum value of 0.

udiT2_sec: Timer monitoring time [s]. Internally limited to a minimum value of 0.

udiAlmCnt: Maximum number of automatic plant restarts without reset. Internally limited to values between 0 and 4.

bRst: Resetting and acknowledgement of the frost alarm.

VAR_OUTPUT

bOn : BOOL;
bHWRst : BOOL;
udiRemTi1_sec : UDINT;
udiRemTi2_sec : UDINT;
bAlmLck : BOOL;
udiStaCnt : UDINT;

bOn: Frost alarm active.

bHWRst: Output of a pulse for acknowledgement of the frost protection hardware.

udiRemTi1_sec: Time remaining to plant restart after frost alarm.

udiRemTi2_sec: Remaining monitoring time.

bAlmLck: Alarm lock - stored alarm.
udiStaCnt: Status counter – current number of unacknowledged false starts.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.1.2 FB_BA_HX

This function block is used to calculate the dew point temperature, the specific enthalpy and the absolute humidity. The temperature, the relative humidity and the barometric pressure are required for calculating these parameters. The enthalpy is a measure for the energy of a thermodynamic system.

VAR_INPUT

- rT : REAL;
- rHumRel : REAL;
- rAP : REAL;

rT: Temperature [°C].

rHumRel: Relative humidity [%].

rAP: Hydrostatic air pressure at 1013.25 hPa.

VAR_OUTPUT

- lrHumAbs : LREAL;
- lrDewPnt : LREAL;
- lrE : LREAL;
- lrDHA : LREAL;
- lrSpecV : LREAL;
- lrTWet : LREAL;

lrHumAbs: Absolute humidity g water per kg dry air [g/Kg].

lrDewPnt: Dew point temperature [°C].

lrE: Enthalpy [kJ/kg].

lrDHA: Density of moist air ρ [kg mixture/m³].

lrSpecV: Specific volume [m³/kg].

lrTWet: Wet bulb temperature [°C].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.1.3 FB_BA_NgtCol

With this function block, rooms that were heated up on the day before can be cooled down during the night using cool outside air. The summer night cooling function serves to improve the quality of the air and to save electrical energy. Electrical energy for cooling is saved during the first hours of the next summer day.

The start conditions for the summer night cooling are defined by parameterizing the FB_BA_NgtCol function block. The function block can be used to open motor-driven windows or to switch air conditioning systems to summer night cooling mode outside their normal hours of operation.

The following conditions must be met for activation of summer night cooling:

- The function block itself is enabled (bEn=TRUE).
- The outside temperature is not too low (rTOts > rTOtsLoLmt).
- The outside temperature is sufficiently low compared with the room temperature (rTRm - rTOts) > rSwiOnDiffT.
- The room temperature is high enough to justify activating summer night cooling. rTRm > rSpRm + rTRmHys.

Under the following conditions the summer night cooling is disabled:

- The function block itself is disabled (bEn = FALSE).
- The outside temperature is too low (rTOts < rTOtsLoLmt).
- The outside temperature is too high compared with the room temperature (rTRm - rTOts) < rSwiOffDiffT.
- The room temperature is lower than the setpoint. rTRm ≤ rSpRm.

VAR_INPUT

<table>
<thead>
<tr>
<th>bEn</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>rTOts</td>
<td>: REAL;</td>
</tr>
<tr>
<td>rTRm</td>
<td>: REAL;</td>
</tr>
<tr>
<td>rSpRm</td>
<td>: REAL;</td>
</tr>
<tr>
<td>rTOtsLoLmt</td>
<td>: REAL;</td>
</tr>
<tr>
<td>rTOtsHys</td>
<td>: REAL;</td>
</tr>
<tr>
<td>rTRmHys</td>
<td>: REAL;</td>
</tr>
<tr>
<td>rSwiOnDiffT</td>
<td>: REAL;</td>
</tr>
<tr>
<td>rSwiOffDiffT</td>
<td>: REAL;</td>
</tr>
</tbody>
</table>

bEn: Enable function block.

rTOts: Outside temperature [°C].

rTRm: Outside temperature [°C].

rSpRm: Room temperature setpoint.

rTOtsLoLmt: Lower outside temperature limit [°C]; prevents excessive cooling.

rTOtsHys: Hysteresis for minimum outside temperature [K]. This hysteresis, which at the lower end is internally limited to 0.5 K, is intended to prevent jitter in bQ, if the outside temperature fluctuates precisely around the value of rTOtsLoLmt.
**rTRmHys**: Hysteresis for the room temperature [K]. This hysteresis, which at the lower end is internally limited to 0.5 K, is intended to prevent unnecessary fluctuation of bQ, if the room temperature fluctuates precisely around the setpoint rSpRm.

**rSwiOnDiffT**: Difference between the room temperature and the outside temperature, from which summer night cooling is enabled [K].

**rSwiOffDiffT**: Difference between the room temperature and the outside temperature, from which summer night cooling is locked [K].

**VAR_OUTPUT**

bQ : BOOL;

bQ: Summer night cooling on.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.1.4 FB_BA_RcvMonit**

The function block is used for calculating the efficiency of an energy recovery system.

The function block requires the following measured temperature values for calculating the efficiency (heat recovery rate):

- Outside air temperature rTOts
- Exhaust air temperature rTExh
- Air temperature of the energy recovery system in the inlet air duct (alternatively: in the outlet air duct) rTAftRcv
The function block logs the temperature values every 10 seconds and forms minutely averages from 6 consecutive values. The results are used to check whether the plant has reached a "stable" state.

- This is the case when the recorded temperatures of outside air, exhaust air and air after energy recovery are almost constant, i.e. none of the 6 individual values deviate by more than 0.5 K from the respective mean value.
- The temperature difference between outside air and exhaust air is at least 5 K.

If this is the case, this measuring cycle is acknowledged with a TRUE signal at output bStblOp, and the calculated efficiency is output at rEffc. If the state is not "stable", a FALSE signal appears at output bStblOp, and rEffc is set to 0.

In any case, each measuring and analysis cycle is marked as completed with a trigger (a TRUE signal lasting one PLC cycle) at bNewVal.

**Enable (bEn) and Reset (bRst)**

The function block is only active if a TRUE signal is present at bEn. Otherwise its execution stops, and all outputs are set to FALSE or 0.0.

An active measuring and evaluation cycle can be terminated at any time by a TRUE signal at bRst. All outputs are set to FALSE or 0.0, and the measuring cycle restarts automatically.

**Selection of the temperature value "after recovery" (bSnsRcvTExh)**

A FALSE entry at bSnsRcvTExh means that the temperature measurement after the heat recovery in the supply air duct is used for calculating the efficiency. To use the temperature measurement after the heat recovery in the exhaust air duct, TRUE must be applied at bSnsRcvTExh.

**Limit violation (rContrVar, rLmtEffc, bLmtRchd)**

A limit violation has occurred, if the calculated efficiency is less than the specified limit value rLmtEffc, and at the same time the control value for the heat recovery is at 100%. To this end the control value must be linked to input rContrVar.

The limit violation message can be delayed by an entry at udiLmtVioDly_sec [s]: If the two criteria, violation and override, are met for longer than udiLmtVioDly_sec [s], this is indicated with a TRUE signal at bLmtRchd.

A warning message, which may have occurred, is canceled if a complete measuring cycle provides "good" values, or with a rising edge at bRst or deactivation of the function block.
**This warning message only occurs if the plant is in a stable operating mode (bStblOp=TRUE).**

---

**Taking into account the temperature increase of the outlet air due to the fan motor (rTIncFan)**

It is possible that the outlet air is warmed by a fan motor, resulting in distortion of the measurement. This temperature increase can be specified through \( rTIncFan \). Internally, the measured outlet air temperature is then reduced by this value.

### VAR_INPUT

- **bEn**: Function block enable.
- **bRst**: Reset - all determined values are deleted.
- **rContrVar**: Control value for the heat recovery, i.e. the actual value.
- **rTOts**: Outside temperature.
- **rTExh**: Exhaust air temperature.
- **rTAftRcv**: Temperature after energy recovery.
- **bSnsRcvTExh**: Temperature at the measuring point after energy recovery: FALSE -> in inlet air duct (SupplyAir) - TRUE -> in outlet air duct (ExhaustAir).
- **rTIncFan**: Temperature increase due to fan.
- **rLmtEffc**: Limit value efficiency.
- **udiLmtVioDly_sec**: Limit violation delay [s]. Internally limited to a minimum value of 0.

### VAR_OUTPUT

- **bNewVal**: Output trigger for new value \( rEffc \).
- **rEffc**: Efficiency
- **bLmtRchd**: Limit value reached
- **bStblOp**: Stable operation.

---

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.1.5 FB_BA_SpSupvis

<table>
<thead>
<tr>
<th>Checking</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>rSpLo &gt; rSpHi</td>
<td>last valid values of rSpLo and rSpHi are used</td>
</tr>
<tr>
<td>rSpMin &gt;= rSpMax</td>
<td>last valid values of rSpMin and rSpMax are used</td>
</tr>
<tr>
<td>rSpHi &gt; rSpMax</td>
<td>rPrSpHi = rSpMax</td>
</tr>
<tr>
<td>rSpLo &lt; rSpMin</td>
<td>rPrSpLo = rSpMin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checking</th>
<th>bErr</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>rSpMin &gt;= rSpMax</td>
<td>TRUE</td>
<td>rSpErr = ((rSpMin + rSpMax) / 2)</td>
</tr>
<tr>
<td>rSpHi &lt; rSpMin</td>
<td></td>
<td>rPrSpHi = rPrSpLo = rPrRcv = rSpErr</td>
</tr>
<tr>
<td>rSpLo &gt; rSpMax</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The difference between the setpoints describes an energy-neutral zone. With inlet air control, no heating or cooling would take place within the neutral zone.

The checked and, if necessary, limited setpoints are output at the function block output as rPrSpHi and rPrSpLo (Present Setpoint).

Setpoint for heat recovery

For heat recovery, the setpoint rSpRcv is optionally calculated from the mean value of the upper and lower setpoint, rSpHi and rSpLo, or depending on the control direction of the heat recovery system.

The method is defined through the input variable bSlcnSpRcv:
### b SlnSpRcv

<table>
<thead>
<tr>
<th>TRUE</th>
<th>rSpRcv</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Mean value of rSpLo and rSpHi</td>
</tr>
<tr>
<td>FALSE</td>
<td>Depends on direction of action, defined through input bActRcv</td>
</tr>
</tbody>
</table>

If the setpoint is defined depending on the direction of action, the following applies:

### bActRcv

<table>
<thead>
<tr>
<th>TRUE</th>
<th>Control direction</th>
<th>rSpRcv</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct (cooling)</td>
<td>rSpHi</td>
<td></td>
</tr>
<tr>
<td>indirect (heating)</td>
<td>rSpLo</td>
<td></td>
</tr>
</tbody>
</table>

#### Heat recovery

- **Room temperature > outside temperature**
  - Controller direction reverse (heating)
- **Room temperature < outside temperature**
  - Controller direction direct (cooling)

#### VAR_INPUT

- **bEn**: function block enable. If bEn = FALSE, all output parameters are 0.0.
- **rSpHi**: Upper setpoint input value to be checked.
- **rSpLo**: Lower setpoint input value to be checked.
- **rSpMax**: Maximum setpoint.
- **rSpMin**: Minimum setpoint.
- **bActnRcv**: Direction of action of the downstream heat recovery.
- **bSlnSpRcv**: Setpoint selection of the downstream heat recovery system.

#### VAR_OUTPUT

- **rPrSpHi**: Output value for the upper setpoint.
rPrSpLo: Output value for the lower setpoint.

rSpRcv: Output value for the resulting heat recovery setpoint.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

<table>
<thead>
<tr>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Warning: The setpoints are not in a logical order: Either ( rSpMin \geq rSpMax ) OR ( rSpHi &lt; rSpMin ) OR ( rSpLo &gt; rSpMax )</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.2 Provision of hot water

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_DHW2P [► 27]</td>
<td>Charge control for a hot water tank via an on-off controller.</td>
</tr>
<tr>
<td>FB_BA_LglPrev [► 29]</td>
<td>Function block for disinfecting service water and destroying legionella.</td>
</tr>
</tbody>
</table>

5.1.2.1 FB_BA_DHW2P

This function block controls the heating of a hot water tank via an on-off controller. Tank heating is activated at input \( bEn \). If tank heating is active the output \( bLd \) is TRUE. The variable \( rSp \) is used to transfer the setpoint for the hot water temperature to the function block. At input \( rTMin \) a minimum selection of all temperature sensors for the hot water tank is connected, at input \( rTMax \) a maximum selection of all temperature sensors.

Due to the thermal stratification in the hot water tank, the sensor at the top is generally the one showing the highest temperature, the one at the bottom the lowest.

The tank can be charged in two ways via the variables \( bKepFul \):

bKepFul = FALSE

Charging is requested if \( rTMax \) falls below the value of \( rSp-rSpHys \). The charge request is disabled if \( rTMin \) is above the setpoint of \( rSp \).

Due to the fact that the sensor at the top generally measures the highest temperature, the heating is not switched on until the hot water tank has been discharged.
bKepFul = TRUE

Charging is requested if \( rTMin \) falls below the value of \( rSp-rSpHys \). The charge request is disabled once \( rTMin \) is above the setpoint again. Selecting the minimum of all tank temperatures ensures that the coldest point of the tank is used for control purposes. Recharging takes place when the tank is no longer full.

VAR_INPUT
bEn : BOOL;
rSp : REAL;
rSpHys : REAL;
rTMax : REAL;
rTMin : REAL;
bKepFul : BOOL;

bEn: Enable boiler charging.
rSp: Service water temperature setpoint [°C].
rSpHys: Hysteresis, recommended 1°K to 5°K.
rTMax: Maximum selection of all tank temperatures [°C].
rTMin: Minimum selection of all tank temperatures [°C].
bKepFul: Control temperature selection:
FALSE = \( rTMax \) is used to request \( bLd \), \( rTMin \) to switch off
TRUE = \( rTMin \) alone controls switching on/off of \( bLd \)
VAR_OUTPUT

bLd : BOOL;
rSpOut : REAL;

bLd: Enable charging mode.

rSpOut: Setpoint transfer to charging circuit:
  • \( rSpOut = rSp \) (input) if the function block is enabled
  • \( rSpOut = 0 \) if the function block is not enabled

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.2.2 FB_BA_LglPrev

This function block is used for disinfection of the service water and for killing off Legionella. Disinfection mode is activated at input \( bEnLglPrev \) via a timer program. It is advisable to run the disinfection at least once per week (during the night). The temperature should be at least 70 °C. The activation interval at \( bEnLglPrev \) must be adequately long. The output \( bLd \) activates tank heating.

For hot water tanks with several temperature sensors, a minimum selection feature for all sensors must be connected at \( rTMin \).

If \( rTMin \) exceeds the value of \( rSp \), a monitoring timer is started with a time of \( udiTi\_sec \) [s]. If the minimum tank temperature \( rTMin \) remains above \( rSp - rSpHys \) while the timer is active, the tank was heated adequately. If circulation is active, the output \( bLd \) must be linked to enabling of the circulation pump, to ensure that the water pipe within the service water system is included in the disinfection. If the temperature has fallen below \( rSp - rSpHys \) during the disinfection process, the process must be restarted and run until the time \( udiTi\_sec \) has fully elapsed. If the disinfection was successful, the output \( bLd \) is reset.

If the disinfection process was incomplete during the function block activation (\( bEnLglPrev \)), this is indicated with the output \( bAlm \). The output must be reset with \( bRst \).
**Explanation of the diagram:**

t0 Start of the legionella program and switching of output \( bLd \). Heating of the hot water tank.

t1 The tank has reached the temperature \( rSp \). The timer for the heating time is started.

t2 The minimum tank temperature has fallen below \( rSp - rSpHys \). The timer for the heating time is reset.

t3 The temperature exceeds \( rSp \) again, and the heating timer is started again.

t4 The minimum tank temperature was above the limit \( rSp - rSpHys \) over the period \( udiTi_sec \); the disinfection was successful. \( bLd \) is reset, and the hot water tank switches back to normal operation.

**VAR_INPUT**

- \( bEnLglPrev \): Enabling of disinfection operation via a timer program.
- \( rTMin \): Minimum tank temperature \([\degree C]\). Minimum selection of temperature sensors at the top and bottom.
- \( rSp \): Setpoint for disinfection \([\degree C]\).
- \( rSpHys \): Temperature difference \([K]\) lower limit; always calculated absolute.
- \( udiTi_sec \): Monitoring period \([s]\).
bRst: Resetting of the legionella alarm;

**VAR_OUTPUT**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bLd</strong></td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>rSpOut</strong></td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>udiRTi</strong></td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>udiSta</strong></td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**bLd:** Anti-legionella mode active.

**rSpOut:** Setpoint transfer to charging circuit:
- rSp (input) if the function block is enabled
- 0 if the function block is not enabled

**udiRTi:** Disinfection mode timer countdown.

**udiSta:** Disinfection program status:

1. The disinfection operation was successful.
2. The disinfection was completed successfully. After the disinfection, and to reactivate legionella prevention, **bEnLglPrev** must be FALSE.
3. The disinfection operation is active.
4. Disinfection was not successful. Alarm is pending.
5. Disinfection was not successful, the alarm was acknowledged.
6. Controller restart, or legionella mode has not yet been requested.

**bAlm:** The temperature setpoint was not reached consistently over via the interval **udiTi_sec**, so that adequate disinfection is not guaranteed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3  Room automation

5.1.3.1  Heating, cooling

**Function blocks**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FB_BA_FnctSel</strong></td>
<td>Function selection (heating and/or cooling) in two- or four-pipe network.</td>
</tr>
<tr>
<td><strong>FB_BA_RmTAdj</strong></td>
<td>Adjustment of the room temperature setpoint.</td>
</tr>
<tr>
<td><strong>FB_BA_SpRmT</strong></td>
<td>Adjustment of the room temperature setpoint.</td>
</tr>
</tbody>
</table>
The function block is used for enabling heating or cooling mode in a room. The distribution network type plays a significant role:

In a two-pipe system, all rooms served by the plant can either be heated or cooled at the same time. In a four-pipe system, the room conditioning can be demand-based, i.e. some rooms can be heated, while other rooms can be cooled by the same plant.

The function block used for each room, as already mentioned, selects its controllers, depending on which type of piping system is available:

Two-pipe network

The two-pipe system is selected if the function block has a FALSE entry at input bPipeSys. Since all rooms served by the plant can only either be heated or cooled, the choice is specified centrally for all rooms via the input bMedium. If bMedium is FALSE, the room heating controller is selected. If the input is TRUE the cooling controller is selected. The controller enable states bEnHtg and bEnCol are always issued with a delay of udiChgOvrDly_h [s]. In other words, heating cannot be enabled until the cooling enable state bEnCol for udiChgOvrDly_h is FALSE, and vice versa. In addition to the elapsing of this changeover time, the system checks that the output from controller to be switched off is 0.0. This is based on feedback at the inputs rCtrlValHtg and rCtrlValCol. In this way, a drastic change from heating to cooling and vice versa is avoided.

Four-pipe network

The four-pipe system is selected if the function block has a TRUE entry at input bPipeSys. In this case, the choice of controller can be different for the individual rooms as required, based on the room temperature rRmT and the setpoints rSpHtg for heating and rSpCol for cooling. If the room temperature exceeds the cooling setpoint, the cooling controller is activated (bEnCol), if it falls below the heating setpoint, the heating controller is activated (bEnHtg). If the temperature is between the two setpoints, both controllers are switched off (energy-neutral zone). Here too, the output of the controller enable states bEnHtg and bEnCol is delayed by udiChgOvrDly_h [s] (see two-pipe network). In addition to the elapsing of this changeover time, the system checks that the output from controller to be switched off is 0.0. This is based on feedback at the inputs rCtrlValHtg and rCtrlValCol. In this way, a drastic change from heating to cooling and vice versa is avoided, if the changeover time is inadequate.

Dew-point monitor (bDewPnt)

In both systems (two- and four-pipe) the dew-point monitor has the task of deactivating cooling immediately, if required.

Program sequence

The function block can have 3 possible states:

1. Waiting for heating or cooling enable
2. Heating enable
3. Cooling enable

In the first step, the function block waits for compliance with the conditions required for heating or cooling:
Programming

<table>
<thead>
<tr>
<th>Heating</th>
<th>Cooling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooling controller output = 0 (rCtrlValCol)</td>
<td>Heating controller output = 0 (rCtrlValHtg)</td>
</tr>
<tr>
<td>Room temperature (rRmT) &lt; heating setpoint (rSpHtg)</td>
<td>Room temperature (rRmT) &gt; cooling setpoint (rSpCol)</td>
</tr>
<tr>
<td>Cooling controller enable (bEnCol) is FALSE over at least the changeover time udiChgOvrDly_sec [s]</td>
<td>Heating controller enable (bEnHtg) is FALSE over at least the changeover time udiChgOvrDly_sec [s]</td>
</tr>
<tr>
<td>Four-pipe system is selected (bPipesys=TRUE) OR two-pipe system is selected and heating medium is available (bPipeSys=False AND bMedium=False)</td>
<td>Four-pipe system is selected (bPipesys=TRUE) OR two-pipe system is selected and cooling medium is available (bPipeSys=False AND bMedium=True)</td>
</tr>
</tbody>
</table>

If a chain of conditions is met, the function block switches to the respective state (heating or cooling) and remains in this state until the corresponding controller issues 0 at the function block input (rCtrlValHtg/ rCtrlValCol). This ensures that only one controller is active at any one time, even if a high heating controller output, for example, would call for a brief cooling intervention (overshoot). Heating or cooling continues until there is no longer a demand.

There are 3 exceptions, for which heating or cooling is immediately interrupted:

1. A two-pipe system (bPipeSys=False) is in heating mode (bEnHtg), but a switch to cooling medium occurred bMedium=TRUE
2. A two-pipe system (bPipeSys=False) is in cooling mode (bEnCol), but a switch to heating medium occurred bMedium=FALSE
3. The dew-point monitor was triggered (bDewPnt=True) in cooling mode (two- or four-pipe system)

In these cases the heating or cooling enable states are canceled, and the plant switches to standby.

Demand message (udiReqdMedium)

To notify the plant of the current demand for heating or cooling, a demand ID is issued at the function block output, i.e. for each room, depending on the actual and set temperature. These can be collected and evaluated centrally. The evaluation always takes place, irrespective of the network type (two- or four-pipe).

<table>
<thead>
<tr>
<th>udiReqdMedium</th>
<th>Medium</th>
<th>Room temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No medium is requested</td>
<td>rRmT &gt; rSpHtg AND rRmT &lt; rSpCol</td>
</tr>
<tr>
<td>2</td>
<td>Heating medium is requested</td>
<td>rRmT &lt; rSpHtg</td>
</tr>
<tr>
<td>3</td>
<td>Cooling medium is requested</td>
<td>rRmT &gt; rSpCol</td>
</tr>
</tbody>
</table>

Error handling

The heating setpoint must not be greater than or equal to the cooling setpoint, since this would result in temperature range with simultaneous heating and cooling demand. However, since the function block only issues one enable state at a time (i.e. heating or cooling), the case is harmless from a plant engineering perspective. In this case only a warning message is issued (bErr=True, sErrDescr=warning message); the function block does not interrupt its cycle.

VAR_INPUT

<table>
<thead>
<tr>
<th>bPipeSys</th>
<th>bool;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bMedium</td>
<td>bool;</td>
</tr>
<tr>
<td>bDewPnt</td>
<td>bool;</td>
</tr>
<tr>
<td>rRmT</td>
<td>real;</td>
</tr>
<tr>
<td>rSpHtg</td>
<td>real;</td>
</tr>
<tr>
<td>rSpCol</td>
<td>real;</td>
</tr>
<tr>
<td>rCtrlValHtg</td>
<td>real;</td>
</tr>
<tr>
<td>rCtrlValCol</td>
<td>real;</td>
</tr>
<tr>
<td>udiChgOvrDel_sec</td>
<td>uint;</td>
</tr>
</tbody>
</table>

bPipeSys: In two-pipe system bPipeSys is FALSE, in four-pipe systems it is TRUE.
bMedium: Current supply of the whole two-pipe network with cooling or heating medium. If heating medium is active, bMedium is FALSE.

bDewPnt: Dew-point monitor: If bDewPnt = TRUE, the cooling controller is locked.

rTRm: Room temperature.

rSpHtg: Heating setpoint.

rSpCol: Cooling setpoint.

rCtrlValHtg: Current output value of the heating controller. Used internally as switching criterion from heating to cooling: rCtrlValHtg must be 0.

rCtrlValCol: Current output value of the cooling controller. Used internally as switching criterion from cooling to heating: rCtrlValCol must be 0.

udiChgOvrDel_sec: Switchover delay [s] from heating to cooling or vice versa. Internally limited to a minimum value of 0.

VAR_OUTPUT

bEnHtg : BOOL;
bEnCol : BOOL;
udiReqdMedium : UDINT;
udiRemTiChgOvrDlyHtg_sec : UDINT;
udiRemTiChgOvrDlyCol_sec : UDINT;
bErr : BOOL;
sErrDescr : T_MAXSTRING;

bEnHtg: Heating controller enable.

bEnCol: Cooling controller enable.

udiReqdMedium:

<table>
<thead>
<tr>
<th>udiReqdMedium</th>
<th>Medium</th>
<th>Room temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No medium is requested</td>
<td>rRmT &gt; rSpHtg AND rRmT &lt; rSpCol</td>
</tr>
<tr>
<td>2</td>
<td>Heating medium is requested</td>
<td>rRmT &lt; rSpHtg</td>
</tr>
<tr>
<td>3</td>
<td>Cooling medium is requested</td>
<td>rRmT &gt; rSpCol</td>
</tr>
</tbody>
</table>

udiRemTiChgOvrDlyHtg_sec: Countdown [s] for switchover delay from cooling to heating.

udiRemTiChgOvrDlyCol_sec: Countdown [s] for switchover delay from heating to cooling.

bErr: In case of a fault, e.g. if warning stages are active, this output is set to TRUE.

sErrDescr: Contains the error description.

Error description

01: Warning: The heating setpoint is higher than or equal to the cooling setpoint

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.3.1.2 FB_BA_RmTAdj

The function block FB_BA_RmTAdj is used for user adjustment of the room temperature setpoint. It shifts the setpoints at the input of a function block depending on an offset rRmTAdj, as shown in the following diagram. At the rRmTAdj input, the value of a resistance potentiometer or a bus-capable field device can be used for the setpoint correction.
If the set value $rRmTAdj$ is greater than zero, room heating is requested: The Comfort Heating value is increased by $rRmTAdj$. At the same time, the values for Comfort Cooling and PreComfort Cooling are increased. If the value $rRmTAdj$ is less than zero, a lower room temperature is requested. Analog to the heating case, the values for Comfort Cooling, Comfort Heating and PreComfort Heating are now reduced by the value $rRmTAdj$.

**Auto-correction**

The temperature adjustment is intended for small corrections of the values. Although it is possible to enter any input values, a heating system will only work in a meaningful manner if the setpoints have ascending values in the following order:
• Protection Heating
• Economy Heating
• Precomfort Heating
• Comfort Heating
• Comfort Cooling
• Precomfort Cooling
• Economy Cooling
• Protection Cooling

Auto-correction works according to the following principle: Starting with the value Economy Heating, the system checks whether this value is smaller than the lower value of Protection Heating. If this is the case, the value for Economy Heating is adjusted to match the value for Protection Heating. The system then checks whether the value for Precomfort Heating is less than Economy Heating and so on, until the value for Protection Cooling is compared with the value for Economy Cooling. If one or several values were corrected, this is indicated with a TRUE signal at output $b_{ValCorr}$.

VAR_INPUT

| rRmTAdj : REAL;          | stSp : ST_BA_SpRmT; |

rRmTAdj: Room temperature offset value.

stSp: Input structure for the setpoints (see ST_BA_SpRmT [183]).

VAR_OUTPUT

| bValCorr : BOOL;         | rPrPrtcHtg : REAL;  |
| rPrEcoHtg : REAL;       | rPrPreCmfHtg : REAL; |
| rPrCmfHtg : REAL;       | rPrPrtcCol : REAL;  |
| rPrEcoCol : REAL;       | rPrPreCmfCol : REAL; |
| rPrCmfCol : REAL;       | stPrSp : ST_BA_SpRmT; |

bValCorr: Autocorrection for the values was performed, see above.

rPrPrtcHtg: Resulting Protection Heating setpoint.

rPrEcoHtg: Resulting Economy Heating setpoint.

rPrPreCmfHtg: Resulting PreComfort Heating setpoint.

rPrCmfHtg: Resulting Comfort Heating setpoint.

rPrCmfCol: Resulting Comfort Cooling setpoint.

rPrPreCmfCol: Resulting PreComfort Cooling setpoint.

rPrEcoCol: Resulting Economy Cooling setpoint.

rPrPrtcCol: Resulting Protection Cooling setpoint.

stPrSp: Consolidated output of the resulting values in a structure (see ST_BA_SpRmT [183]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.3.1.3 \textbf{FB\_BA\_SpRmT}

The function block \textit{FB\_BA\_SpRmT} assigns setpoints for cooling and heating operation to each of the energy levels Protection, Economy, PreComfort and Comfort.

The following graphics illustrates the behavior of the function block; the entered values should be regarded as examples:
The parameter $r_{\text{ShiftHtg}}$ is applied to the Comfort and Precomfort values for the heating mode as central setpoint shift. In addition, winter compensation $r_{\text{WinCpsn}}$ is applied. Similarly, the following applies for the cooling mode: The parameter $r_{\text{ShiftCol}}$ is applied to the Comfort and Precomfort values. In addition, the summer compensation value $r_{\text{SumCpsn}}$ is applied.
Auto-correction

The setpoint shift is intended for small corrections of the values. Although it is possible to enter any input values, a heating system will only work in a meaningful manner if the setpoints have ascending values in the following order:

- Protection Heating
- Economy Heating
- Precomfort Heating
- Comfort Heating
- Comfort Cooling
- Precomfort Cooling
- Economy Cooling
- Protection Cooling

Auto-correction works according to the following principle: Starting with the value Economy Heating, the system checks whether this value is smaller than the lower value of Protection Heating. If this is the case, the value for Economy Heating is adjusted to match the value for Protection Heating. The system then checks whether the value for Precomfort Heating is less than Economy Heating and so on, until the value for Protection Cooling is compared with the value for Economy Cooling. If one or several values were corrected, this is indicated with a TRUE signal at output bValCorr.

VAR_INPUT

rSumCpsn : REAL;
rWrWinCpsn : REAL;

rSumCpsn: Summer compensation value

rWinCpsn: Winter compensation value

VAR_OUTPUT

bValCorr : BOOL;
rPrPrtcHtg : REAL;
rPrEcoHtg : REAL;
rPrPreCmfHtg : REAL;
rPrCmfHtg : REAL;
rPrCmfCol : REAL;
rPrPreCmfCol : REAL;
rPrEcoCol : REAL;
rPrPrtcCol : REAL;

bValCorr: Autocorrection: At least one of the resulting setpoints was adjusted such that the values continue to monotonically increase.

rPrPrtcHtg: Resulting Protection Heating setpoint.

rPrEcoHtg: Resulting Economy Heating setpoint.

rPrPreCmfHtg: Resulting PreComfort Heating setpoint.

rPrCmfHtg: Resulting Comfort Heating setpoint.

rPrCmfCol: Resulting Comfort Cooling setpoint.

rPrPreCmfCol: Resulting PreComfort Cooling setpoint.

rPrEcoCol: Resulting Economy Cooling setpoint.

rPrPrtcCol: Resulting Protection Cooling setpoint.

stPrSp: Consolidated output of the resulting values in a structure (see ST_BA_SpRmT [183]).
VAR_INPUT_CONSTANT_PERSISTENT (Parameter)

rShiftCol  :  REAL  :=  0;
rShiftHtg  :  REAL  :=  0;
rPrtcCol   :  REAL  :=  35;
rEcoCol    :  REAL  :=  28;
rPreCmfCol :  REAL  :=  25;
rCmfCol    :  REAL  :=  23;
rCmfHtg    :  REAL  :=  21;
rPreCmfHtg :  REAL  :=  18;
rEcoHtg    :  REAL  :=  14;
rPrtcHtg   :  REAL  :=  6;

rShiftCol: Cooling setpoint value shift.
rShiftHtg: Heating setpoint value shift.
rPrtcCol: Basic Protection Cooling setpoint.
rEcoCol: Basic Economy Cooling setpoint.
rPreCmfCol: Basic PreComfort Cooling setpoint.
rCmfCol: Basic Comfort Cooling setpoint.
rCmfHtg: Basic Comfort Heating setpoint.
rPreCmfHtg: Basic PreComfort Heating setpoint.
rEcoHtg: Basic Economy Heating setpoint.
rPrtcHtg: Basic Protection Heating setpoint.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.2  Lighting

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_Toggle [41]</td>
<td>Switching of lamps.</td>
</tr>
</tbody>
</table>

5.1.3.2.1  FB_BA_Toggle

The function block is used to switch an actuator on or off.

The input bEn is used for enabling the function block.
A positive edge at the input bOn results in setting of output bQ. The output is reset by a rising edge at the bOff input. If a rising edge is presented to bToggle, the output is negated; i.e., if On it goes Off, and if Off it goes On.
VAR_INPUT
bEn : BOOL;
bOn : BOOL;
bOff : BOOL;
bToggle : BOOL;

bEn: Function block enable.
bOn: Switches the output on.
bOff: Switches the output off.
bToggle: Negates the current output state.

VAR_OUTPUT
bQ : BOOL;

bQ: Control output.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3 Shading

Overview of shading correction [44]

Shading correction: Basic principles and definitions [44]

Overview of automatic sun protection [52]

Sun protection: Basic principles and definitions [54]

List of shading elements [59]

List of facade elements [59]
## Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_BldPosEntry [59]</td>
<td>Sun protection function: Input of blind positions.</td>
</tr>
<tr>
<td>FB_BA_CalcSunPos [61]</td>
<td>Calculation of sun position</td>
</tr>
<tr>
<td>FB_BA_FcdElemEntry [63]</td>
<td>Shading correction: Input of facade elements per function block.</td>
</tr>
<tr>
<td>FB_BA_InRngAzm [68]</td>
<td>Verification of valid sun position and sun direction range (azimuth angle)</td>
</tr>
<tr>
<td>FB_BA_InRngElv [70]</td>
<td>Verification of valid sun position and sun elevation range (elevation angle)</td>
</tr>
<tr>
<td>FB_BA_RdFcdElemLst [73]</td>
<td>Shading correction: Input of facade elements via data list (csv).</td>
</tr>
<tr>
<td>FB_BA_RdShdObjLst [77]</td>
<td>Shading correction: Input of shading objects via data list (csv).</td>
</tr>
<tr>
<td>FB_BA_RolBldActr [81]</td>
<td>Roller shutter actuator</td>
</tr>
<tr>
<td>FB_BA_ShdCorr [83]</td>
<td>Shading correction function block</td>
</tr>
<tr>
<td>FB_BA_ShdObjEntry [86]</td>
<td>Shading correction: Input of shading objects per function block.</td>
</tr>
<tr>
<td>FB_BA_SunBldActr [89]</td>
<td>Blind actuator</td>
</tr>
<tr>
<td>FB_BA_SunBldEvt [94]</td>
<td>Output of a specified blind position and angle in percent</td>
</tr>
<tr>
<td>FB_BA_SunBldIcePrtc [95]</td>
<td>Anti-icing</td>
</tr>
<tr>
<td>FB_BA_SunBldPosDlv [96]</td>
<td>Switch-on delay for blinds/groups of blinds</td>
</tr>
<tr>
<td>FB_BA_SunBldPrioSwi4 [97]</td>
<td>Priority control, 4 inputs</td>
</tr>
<tr>
<td>FB_BA_SunBldPrioSwi8 [98]</td>
<td>Priority control, 8 inputs</td>
</tr>
<tr>
<td>FB_BA_SunBldScn [99]</td>
<td>Manual operation with scene selection and programming</td>
</tr>
<tr>
<td>FB_BA_SunBldSwi [102]</td>
<td>Manual operation</td>
</tr>
<tr>
<td>FB_BA_SunBldTwltAuto [104]</td>
<td>Automatic twilight function</td>
</tr>
<tr>
<td>FB_BA_SunBldWndPrtc [105]</td>
<td>Protection against wind damage</td>
</tr>
<tr>
<td>FB_BA_SunPrtc [107]</td>
<td>Sun protection function, see Overview of automatic sun protection (shading correction)</td>
</tr>
</tbody>
</table>
5.1.3.3.1 Overview of 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:

5.1.3.3.2 Shading correction: Basic principles and definitions

The shading correction can be used in conjunction with the automatic sun function or louvre adjustment. The function checks whether a window or a window group that is assigned to a room, for example, is temporarily placed in the shade by surrounding buildings or parts of its own building. Sun shading for windows that stand in the shadow of surrounding buildings or trees is not necessary and may even be disturbing under certain circumstances. On the basis of data entered regarding the facade and its surroundings, the shading correction determines which parts of the front are in the shade. Hence, it is then possible to decide whether the sun protection should be active for individual windows or window groups.
• 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:

<table>
<thead>
<tr>
<th>Line of sight</th>
<th>Facade orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>β=0°</td>
</tr>
<tr>
<td>East</td>
<td>β=90°</td>
</tr>
<tr>
<td>South</td>
<td>β=180°</td>
</tr>
<tr>
<td>West</td>
<td>β=270°</td>
</tr>
</tbody>
</table>

In the southern hemisphere is the sun path is the other way round: Although it also rises in the east, at midday it is in the north. The facade orientation is adjusted to this path:

<table>
<thead>
<tr>
<th>Line of sight</th>
<th>Facade orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>South</td>
<td>β=0°</td>
</tr>
<tr>
<td>East</td>
<td>β=90°</td>
</tr>
<tr>
<td>North</td>
<td>β=180°</td>
</tr>
<tr>
<td>West</td>
<td>β=270°</td>
</tr>
</tbody>
</table>

For convenience, the other explanations refer to the northern hemisphere. The calculations for the southern hemisphere are analogous. When the function block \texttt{FB_BA_ShdCorr}\[83\] (shading correction) is parameterized they are activated through a boolean input, \texttt{bSouth}

The following two illustrations are intended to further clarify the position of the point of origin \(P_0\) as well as the orientation of the coordinate system:
Observation from the side

The angle of elevation (height of the sun) can be represented using this illustration: by definition this is 0° at sunrise (horizontal incidence of light) and can reach maximally 90°, but this applies only to places within the Tropic of Cancer and the Tropic of Capricorn.

Observation from the front

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

Position of the windows

The position of the windows is defined by the specification of their bottom left corner in relation to the facade coordinate system. Since a window lies flat on the facade, the entry is restricted to the X and Y coordinates.
In addition, the window width and the window height have to be specified.

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

**Positioning of the shading objects**

When describing the shading objects, distinction is made between angular objects (building, column) and objects that are approximately spherical (e.g. trees). Angular objects can be subdivided into rectangular shadow-casting facades depending on their shadow projection; you should consider which surfaces cast the main shadows over the day:
Morning/noon

In the morning and around noon, the shadow is mainly cast by the sides $S_1$ and $S_4$. $S_2$ and $S_3$ do not have to be considered, unless they are higher.
In the afternoon and evening, the total shade can be determined solely through $S_1$ and $S_2$. In this case it is therefore sufficient to specify $S_1$, $S_2$, and $S_4$ as shadow casters. The entry is made on the basis of the four corners or their coordinates in relation to the zero point of the facade:
In this sketch only the upper points, P₂ and P₃, are illustrated due to the plan view. The lower point P₁ lies underneath P₂ and P₄ lies underneath P₃.

The input of shadow-casting ball elements is done by entering the center of the ball and its radius:

**Ball elements**

**Meaning of colors**
- Blue: effective for the whole building
- Red: effective for one facade
- Green: effective for a room/group of rooms

- **Ice protection**
  - FB_BA_SunBldIcePrtc

- **Wind protection**
  - FB_BA_SunBldWndPrtc

- **Maintenance position**
  - FB_BA_SunBldEvt

- **Manual-mode**
  - FB_BA_SunBldSwi
  - or
  - Manual-mode with scene selection and scene-programming
    - FB_BA_SunBldScn

- **Twilight-automatic**
  - FB_BA_SunBldTwilLgtAuto

- **Thermo-automatic**
  - FB_BA_SunBldEvt

- **Sun-protection**
  - FB_BA_SunPrtc

- **Parking-position**
  - FB_BA_SunBldEvt

**Priority selection**
- FB_BA_SunBldPrioSwi

**Positioning telegram**
- ST_BA_SunBld

**Input**
- Prio1-input
- Prio2-input
- Prio3-input
- Prio4-input
- Prio5-input
- Prio6-input
- Prio7-input
- Prio8-input

**Output**
- Sunblind-actuator
  - FB_BA_SunBldActr
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.
5.1.3.3.3  Overview of automatic sun protection
5.1.3.3.4 Sun protection: Basic principles and definitions

The direct incidence of daylight is regarded as disturbing by persons in rooms. On the other hand, however, people perceive natural light to be more pleasant in comparison with artificial light. Two options for glare protection are to be presented here:

- Louvre adjustment
- Height adjustment

Louvre adjustment

A louvered blind that can be adjusted offers the option of intelligent sun protection here. The position of the louvres is cyclically adapted to the current position of the sun, so that no direct daylight enters through the blinds, but as much diffuse daylight can be utilized as possible.

The illustration shows that diffuse light can still enter from underneath, whereas no further direct daylight, or theoretically only a single ray, can enter. The following parameters are necessary for the calculation of the louvre angle:

- the current height of the sun (angle of elevation)
- the sun position, i.e. the azimuth angle
- the facade orientation
- the louvre width
- the louvre spacing

effective elevation angle

If the blind is viewed in section as above, the angle of incidence does not depend solely on the solar altitude (elevation), but also on the direction of the sun:

- If the facade orientation and the sun position (azimuth) are the same, i.e. the sunlight falls directly onto the facade, the effective light incidence angle is the same as the current elevation angle.
- However, if the sunlight falls at an angle onto the facade as seen from the sun direction, the effective angle is larger for the same angle of elevation.

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

We therefore refer to the "effective elevation angle", which describes the proportion of light that falls directly onto the blind.
The following three images illustrate the relationship between the effective elevation angle and the blind dimensions, and how the resulting louvre angle \( \lambda \) changes during the day:

**louvre-angle**

louvre at an angle of \( \lambda = 0 \)

Louver-angle in the morning and in the evening

Louver angle at noon
Height adjustment

With a high position of the sun at midday, the direct rays of sunlight do not penetrate into the full depth of the room. If direct rays of sunlight in the area of the window sill are regarded as uncritical, the height of the sun protection can be adapted automatically in such a way that the rays of sunlight only ever penetrate into the room up to an uncritical depth.

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

Required for the calculation of the respective blind height:

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

The following illustration shows where these parameters are to be classified:
Influence of the facade inclination

In both of the methods of sun protection described, it was assumed that the facade and thus the windows are perpendicular to the ground. In the case of an inclined facade, however, the incidence of light changes such that this influence will also be taken into account. The facade inclination is defined as follows:
facade angle: $\varphi = 0^\circ$

facade angle: $\varphi < 0^\circ$

facade angle: $\varphi > 0^\circ$
5.1.3.3.5 List of shading elements

The data of all shading objects (building components, trees, etc.) per facade are stored in a field of structure elements of type ST_BA_ShdObj [182] within the program.

The shading correction FB_BA_ShdCorr [83] reads the information from this list. The management function block FB_BA_ShdObjEntry [86] reads and writes it as input/output variable. It is therefore advisable to declare this list globally:

VAR_GLOBAL
  arrShdObj : ARRAY [1..gBA_cMaxShdObj] OF ST_BA_ShdObj;
END_VAR

The variable gBA_cMaxShdObj represents the upper limit of the available elements and is defined as a global constant within the program library:

VAR_GLOBAL CONSTANT
  gBA_cMaxShdObj : INT := 20;
END_VAR

5.1.3.3.6 List of facade elements

The data of all windows (facade elements) per facade are saved within the program in a field of structure elements of the type ST_BA_FcdElem [181].

The management function block FB_BA_FcdElemEntry [63] and the shading correction FB_BA_ShdCorr [83] read and write to this list (the latter sets the shading information); they access this field as input/output variables. It is therefore advisable to declare this list globally:

VAR_GLOBAL
  arrFcdElem : ARRAY [1..uiMaxRowFcd, 1..uiMaxColumnFcd] OF ST_BA_FcdElem;
END_VAR

The variables uiMaxColumnFcd and uiMaxRowFcd define the upper limit of the available elements and are declared as global constants within the program library:

VAR_GLOBAL CONSTANT
  uiMaxRowFcd := 10;
  uiMaxColumnFcd := 20;
END_VAR

5.1.3.3.7 FB_BA_BldPosEntry

This function block is used for entering interpolation points for the function block FB_BA_SunPrtc [107], if this function block is operated in height positioning mode with the aid of a table, see E_BA_PosMod [179].

In addition to the operating modes "Fixed shutter height" and "Maximum incidence of light", the function block FB_BA_SunPrtc [107] 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 [107], this function block is to be preceded by the function block
FB_BA_BldPosEntry. Four interpolation points can be parameterized on this function block, whereby a missing entry is evaluated as a zero entry. The function block does not sort the values entered independently, but instead ensures that the positions of the sun entered in the respective interpolation points are entered in ascending order. Unintentional erroneous entries are noticed faster as a result. The values chosen for $rSunElv1 .. rSunElv4$ must be unique; for example, the following situation must be avoided:

\[ rSunElv1 = 10 ; rBldPos1 = 50 \] and at the same time \[ rSunElv2 = 10 ; rBldPos2 = 30 \].

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:

- $rSunElv1 < rSunElv2 < rSunElv3 < rSunElv4$ - (values ascending and not equal)
- $0 \leq rSunElv \leq 90$ (° - validity range source values)
- $0 \leq rPos \leq 100$ (in percent - validity range target values)

The function block checks the entered values for these conditions and issues an error message if they are not met. In addition, the value $bValid$ of $ST_BA_BldPosTab[180]$ is set to FALSE. Furthermore the function block independently ensures that the boundary areas are filled out: Internally, a further interpolation point is set at $rSunElv = 0$ with $rBldPos1$ and another one above $rSunElv4$ at $rSunElv = 90$ with $rBldPos4$. This ensures that a meaningful target value is available for all valid input values $0 \leq rSunElv \leq 90$, without the user having to enter $rSunElv = 0$ and $rSunElv = 90$:

This increases the actual number of interpolation points transferred to the function block $FB_BA_SunPrtc [107]$ to 6; see $ST_BA_BldPosTab[180]$.

The interpolation of the values takes place in the glare protection function block.

VAR_INPUT

\[
\begin{align*}
&rSunElv1 : \text{REAL}; \\
&rPos1 : \text{REAL}; \\
&rSunElv2 : \text{REAL}; \\
&rPos2 : \text{REAL}; \\
&rSunElv3 : \text{REAL}; \\
&rPos3 : \text{REAL}; \\
&rSunElv4 : \text{REAL}; \\
&rPos4 : \text{REAL};
\end{align*}
\]

$rSunElv1$: Sun position at the first interpolation point (0°..90°).
rBldPos1: Blind position (degree of closure) at the first interpolation point (0%..100%).
rSunElv2: Sun position at the second interpolation point (0°..90°).
rBldPos2: Blind position (degree of closure) at the second interpolation point (0%..100%).
rSunElv3: Sun position at the third interpolation point (0°..90°).
rBldPos3: Blind position (degree of closure) at the third interpolation point (0%..100%).
rSunElv4: Sun position at the fourth interpolation point (0°..90°).
rBldPos4: Blind position (degree of closure) at the fourth interpolation point (0%..100%).

VAR_OUTPUT

stBldPosTab : ST_BA_BldPosTab;
bErr : BOOL;
sErrDescr : T_MAXSTRING;

stBldPosTab: Transfer structure of the interpolation points, see ST_BA_BldPosTab [180].
bErr: This output is switched to TRUE if the parameters entered are erroneous.
sErrDescr: Contains the error description.

<table>
<thead>
<tr>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Error: The x-values (elevation values) in the table are either not listed in ascending order, or they are duplicated.</td>
</tr>
<tr>
<td>02: Error: An elevation value that was entered is outside the valid range of 0°..90°.</td>
</tr>
<tr>
<td>03: Error: An position value that was entered is outside the valid range of 0°...100%.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.8 FB_BA_CalcSunPos

Calculation of sun position based on the date, time, longitude and latitude.

The position of the sun for a given point in time can be calculated according to common methods with a defined accuracy. For applications with moderate requirements, the present function block is sufficient. As the basis for this, the SUNAE algorithm was used, which represents a favorable compromise between accuracy and computing effort.

The position of the sun at a fixed observation point is normally determined by specifying two angles. One angle indicates the height above the horizon; 0° means that the sun is in the horizontal plane of the location; a value of 90° means that the is perpendicular to the observer. The other angle indicates the direction at which the sun is positioned. The SUNAE algorithm is used to distinguish whether the observer is standing on the northern hemisphere (longitude > 0 degrees) or on the southern hemisphere (longitude < 0 degrees) of the earth. If the observation point is in the northern hemisphere, then a value of 0° is assigned for the northern sun direction and it then runs in the clockwise direction around the compass, i.e. 90° is east, 180° is south, 270° is west etc. If the point of observation is in the southern hemisphere, then 0° corresponds to the southern direction and it then runs in the counter clockwise direction, i.e. 90° is east, 180° is north, 270° is west etc.
The time has to be specified as coordinated world time (UTC, Universal Time Coordinated, previously referred to as GMT, Greenwich Mean Time).

The latitude is the northerly or southerly distance of a location on the Earth’s surface from the equator, in degrees [°]. The latitude can assume a value from 0° (at the equator) to ±90° (at the poles). A positive sign thereby indicates a northern direction and a negative sign a southern direction. The longitude is an angle that can assume values up to ±180° starting from the prime meridian 0° (an artificially determined North-South line). A positive sign indicates a longitude in an eastern direction and a negative sign in a western direction.

Examples:

<table>
<thead>
<tr>
<th>Location</th>
<th>Longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sydney, Australia</td>
<td>151.2°</td>
<td>-33.9°</td>
</tr>
<tr>
<td>New York, USA</td>
<td>-74.0</td>
<td>40.7°</td>
</tr>
<tr>
<td>London, England</td>
<td>-0.1°</td>
<td>51.5°</td>
</tr>
<tr>
<td>Moscow, Russia</td>
<td>37.6°</td>
<td>55.7°</td>
</tr>
<tr>
<td>Peking, China</td>
<td>116.3°</td>
<td>39.9°</td>
</tr>
<tr>
<td>Dubai, United Arab Emirates</td>
<td>55.3°</td>
<td>25.4°</td>
</tr>
<tr>
<td>Rio de Janeiro, Brazil</td>
<td>-43.2°</td>
<td>-22.9°</td>
</tr>
<tr>
<td>Hawaii, USA</td>
<td>-155.8°</td>
<td>20.2°</td>
</tr>
<tr>
<td>Verl, Germany</td>
<td>8.5°</td>
<td>51.9°</td>
</tr>
</tbody>
</table>

If the function block `FB_BA_CalcSunPos` returns a negative value for the solar altitude `rSunElv`, the sun is invisible. This can be used to determine sunrise and sunset.

**VAR_INPUT**

```
VAR_INPUT
rDegLngd : REAL;
rDegLatd : REAL;
stUTC : TIMESTRUCT;
```

- **rDegLngd**: Longitude [°].
- **rDegLatd**: Latitude [°].
- **stUTC**: Input of the current time as coordinated world time (see TIMESTRUCT). The function block `FB_BA_GetTime` can be used to read this time from a target system.

**VAR_OUTPUT**

```
VAR_OUTPUT
rSunAzm : REAL;
rSunElv : REAL;
```

- **rSunAzm**: 
- **rSunElv**: 

---

[112]
**rSunAzm:** Direction of the sun (northern hemisphere: 0° north ... 90° east ... 180° south ... 270° west ... / southern hemisphere: 0° south ... 90° east ... 180° north ... 270° west ...).

**rSunElv:** Height of the sun (0° horizontal - 90° vertical).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.9.9 **FB_BA_FcdElemEntry**

This function block serves the administration of all facade elements (windows) in a facade, which are saved globally in a list of facade elements [p. 59]. It is intended to facilitate inputting element information - not least with regard to using the TC3 PLC HMI. A schematic illustration of the objects with description of the coordinates is given in Shading correction: principles and definitions [p. 44].

The facade elements are declared in the global variables as a two-dimensional field above the window columns and rows:

```plaintext
VAR_GLOBAL
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Each element `arrFcdElem[x,y]` contains the information for an individual facade element (*ST_BA_FcdElem [p.181]*) The information includes the group affiliation, the dimensions (width, height) and the coordinates of the corners. The function block thereby accesses this field directly via the IN-OUT variable `arrFcdElem`.

**Note:** The fact that the coordinates of corners C2 to C4 are output values arises from the fact that they are formed from the input parameters and are to be available for use in a visualization:
All entries in [m]!

\[ \begin{align*}
    r\text{Cnr2}X &= r\text{Cnr1}X \\
    r\text{Cnr2}Y &= r\text{Cnr1}Y + rWdwHght \text{ (window height)} \\
    r\text{Cnr3}X &= r\text{Cnr1}X + rWdwWdth \text{ (window width)} \\
    r\text{Cnr3}Y &= r\text{Cnr2}Y \\
    r\text{Cnr4}X &= r\text{Cnr1}X + rWdwWdth \text{ (window width)} \\
    r\text{Cnr4}Y &= r\text{Cnr1}Y
\end{align*} \]

The function block is used in three steps:

- Read
- Change
- Write

Read

The entries \textit{udiColumn} and \textit{udiRow} are used to select the corresponding element from the list, \textit{arrFcdElem[udiColumn, udiRow]}. A rising edge on \textit{bRd} reads the following data from the list element:

- \textit{usiGrp} group membership,
- \textit{rCnr1X} x-coordinate of corner point 1 [m]
- \textit{rCnr1Y} y-coordinate of corner point 1 [m]
- \textit{rWdwWdth} window width [m]
- \textit{rWdwHght} window height [m]

These are then assigned to the corresponding input variables of the function block, which uses them to calculate the coordinates of corners C2-C4 as output variables in accordance with the correlation described above. It is important here that the input values are not overwritten in the reading step. Hence, all values can initially be displayed in a visualization.
Change

In a next program step the listed input values can then be changed. The values entered are constantly checked for plausibility. The output \( bErr \) indicates whether the values are valid (\( bErr=\text{FALSE} \)). If the values are invalid, a corresponding error message is issued at output \( sErrDescr \). See also "Error (\( bErr=\text{TRUE} \))" below.

Write

The parameterized data are written to the list element with the index \( nId \) upon a rising edge on \( bWrt \), regardless of whether they represent valid values or not. The element structure \( \text{ST_BA_FcdElem} \) therefore also contains a plausibility bit \( bVld \), which forwards precisely this information to the function block \( \text{FB_BA_ShdCorr} \) to prevent miscalculations.

This approach is to be regarded only as a proposal. It is naturally also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on \( bWrt \).

Error (\( bErr=\text{TRUE} \))

The function block \( \text{FB_BA_ShdCorr} \), 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:

- \( \text{usiGrp} \) must be greater than 0
- \( rCnr1X \) must be greater than or equal to 0.0
- \( rCnr1Y \) must be greater than or equal to 0.0
- \( rWdwWdth \) must be greater than 0
- \( rWdwHght \) must be greater than 0

If one of these criteria is not met, it is interpreted as incorrect input, and the error output \( bErr \) is set at the function block output of \( \text{FB_BA_FcdElemEntry} \). Within the window element \( \text{ST_BA_FcdElem} \), the plausibility bit \( bVld \) is set to FALSE.

If on the other hand all entries of a facade element are zero, it is regarded as a valid, deliberately omitted facade element:
In the case of a facade of 6x4 windows, the elements window (2.1), window (3.5) and window (4.4) would be empty elements here.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>var</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>udiColumn</td>
<td>UDINT</td>
</tr>
<tr>
<td>udiRow</td>
<td>UDINT</td>
</tr>
<tr>
<td>bWrt</td>
<td>BOOL</td>
</tr>
<tr>
<td>bRd</td>
<td>BOOL</td>
</tr>
<tr>
<td>udiGrp</td>
<td>UDINT</td>
</tr>
<tr>
<td>rCnr1X</td>
<td>REAL</td>
</tr>
<tr>
<td>rCnr1Y</td>
<td>REAL</td>
</tr>
<tr>
<td>rWdwWdth</td>
<td>REAL</td>
</tr>
<tr>
<td>rWdwHght</td>
<td>REAL</td>
</tr>
</tbody>
</table>

**udiColumn:** Column index of the selected component on the facade. This refers to the selection of a field element of the array stored in the IN-OUT variable `arrFcdElem`.

**udiRow:** ditto. row index. **udiRow and udiColumn must not be zero!** This is due to the field definition, which always starts with 1; see above.
bRd: A positive edge at this input causes the information of the selected element, \( arrFcdElem[udiColumn, udiRow] \), to be read into the function block and assigned to the input variables \( diGrp \) to \( rWdwHght \). The resulting output variables are \( rCnr2X \) to \( rCnr4Y \). If data are already present on the inputs \( diGrp \) to \( rWdwHght \) at time of reading, then the data previously read are immediately overwritten with these data.

bWrt: A positive edge writes the entered and calculated values into the selected field element \( arrFcdElem[udiColumn, udiRow] \).

udiGrp: Group membership. Internally limited to a minimum value of 0.

rCnr1X: X-coordinate of corner point 1 [m].

rCnr1Y: Y-coordinate of corner point 1 [m].

rWdwWdth: Window width [m].

rWdwHght: Window height [m].

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rCnr2X</td>
<td>REAL</td>
</tr>
<tr>
<td>rCnr2Y</td>
<td>REAL</td>
</tr>
<tr>
<td>rCnr3X</td>
<td>REAL</td>
</tr>
<tr>
<td>rCnr3Y</td>
<td>REAL</td>
</tr>
<tr>
<td>rCnr4X</td>
<td>REAL</td>
</tr>
<tr>
<td>rCnr4Y</td>
<td>REAL</td>
</tr>
<tr>
<td>bErr</td>
<td>BOOL</td>
</tr>
<tr>
<td>sErrDesc</td>
<td><em>T_MAXSTRING</em></td>
</tr>
</tbody>
</table>

rCnr2X: Calculated X-coordinate of corner point 2 of the window [m]. See "Note [53]" above.

rCnr2Y: Calculated Y-coordinate of corner point 2 of the window [m]. See "Note [53]" above.

rCnr3X: Calculated X-coordinate of corner point 3 of the window [m]. See "Note [53]" above.

rCnr3Y: Calculated Y-coordinate of corner point 3 of the window [m]. See "Note [53]" above.

rCnr4X: Calculated X-coordinate of corner point 4 of the window [m]. See "Note [53]" above.

rCnr4Y: Calculated Y-coordinate of corner point 4 of the window [m]. See "Note [53]" above.

bErr: Result verification for the entered values.

sErrDesc: Contains the error description.

<table>
<thead>
<tr>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Error: Index error! udiColumn and/or udiRow are outside the permitted limits, 1.. uiMaxColumnFcd and</td>
</tr>
<tr>
<td>1..uiMaxColumnFcd, respectively. See list of facade elements.</td>
</tr>
<tr>
<td>02: Error: The group index is 0, but at the same time another entry of the facade element is not zero. Only if</td>
</tr>
<tr>
<td>all entries of a facade element are zero is it considered to be a valid, deliberately omitted facade</td>
</tr>
<tr>
<td>component, otherwise it is interpreted as an incorrect entry. NOTE: Group entries less than zero are</td>
</tr>
<tr>
<td>internally limited to zero.</td>
</tr>
<tr>
<td>03: Error: The X-component of the first corner point (Corner1) is less than zero.</td>
</tr>
<tr>
<td>04: Error: The Y-component of the first corner point (Corner1) is less than zero.</td>
</tr>
<tr>
<td>05: Error: The window width is less than or equal to zero.</td>
</tr>
<tr>
<td>06: Error: The window height is less than or equal to zero.</td>
</tr>
</tbody>
</table>

**VAR_IN_OUT**

\[ arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem; \]

arrFcdElem: List of facade elements (see List of facade elements [59]).
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.10 FB_BA_InRngAzm

This function block checks whether the current azimuth angle (horizontal position of the sun) lies within the limits entered. As can be seen in the overview [52], the function block provides an additional evaluation as to whether the sun shading of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

The sun incidence azimuth angle on a smooth facade will always be \textit{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 start of the range \( r\text{SttRng} \) may be greater than the end \( r\text{EndRng} \), in which case values beyond 0° are considered:

**Example**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r\text{Azm} )</td>
<td>10.0°</td>
</tr>
<tr>
<td>( r\text{SttRng} )</td>
<td>280.0°</td>
</tr>
<tr>
<td>( r\text{EndRng} )</td>
<td>20.0°</td>
</tr>
<tr>
<td>( \text{bOut} )</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

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 \( \text{bErr} \) – the test output \( \text{bOut} \) is then additionally set to FALSE.

**VAR_INPUT**

\[
\begin{align*}
\text{rAzm} & : \text{REAL}; \\
\text{rSttRng} & : \text{REAL}; \\
\text{rEndRng} & : \text{REAL};
\end{align*}
\]

- \( \text{rAzm} \): Current azimuth angle.
- \( \text{rSttRng} \): Start of range [°].
**rEndRng**: End of range [°].

### VAR_OUTPUT

- **bOut**: The facade element is in the sun if the output is TRUE.
- **bErr**: This output is switched to TRUE if the parameters entered are erroneous.
- **sErrDescr**: Contains the error description.

### Error description

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Error: ( rSttRng ) or ( rEndRng ) less than 0° or greater than 360°.</td>
</tr>
<tr>
<td>02</td>
<td>Error: The difference between ( rSttRng ) and ( rEndRng ) is greater than 180°. This range is too large for analyzing the insolation on a facade.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

#### 5.1.3.3.11 FB_BA_InRngElv

This function block checks whether the current angle of elevation (vertical position of the sun) lies within the limits entered. As can be seen in the overview [52], the function block provides an additionally evaluation as to whether the sun shading of a window group should be activated. Therefore the observations in the remainder of the text always apply to one window group.

A normal vertical facade is irradiated by the sun at an angle of elevation of 0° to maximally 90°.
If the facade has projections, however, this range is limited. This limitation can be checked with the help of this function block. However, the position of the window group on the facade also plays a role. If it lies in the lower range, this gives rise to the following situation (the values are only examples):
The values change for a group below the projection:
The lower observation limit, \( r_{LoLmt} \), may thereby not be greater than or equal to the upper limit, \( r_{HiLmt} \). Such entries result in an error on the output \( bErr \) – the test output \( bOut \) is then additionally set to FALSE.

**VAR_INPUT**

- \( rElv : \text{REAL}; \)
- \( rLoLmt : \text{REAL}; \)
- \( rHiLmt : \text{REAL}; \)

- \( rElv: \text{Current elevation angle \( [\degree] \).} \)
- \( rLoLmt: \text{Lower limit value \( [\degree] \).} \)
- \( rHiLmt: \text{Upper limit value \( [\degree] \).} \)

**VAR_OUTPUT**

- \( bOut : \text{BOOL}; \)
- \( bErr : \text{BOOL}; \)
- \( sErrDescr : \text{T_MAXSTRING}; \)

- \( bOut: \text{The facade element is in the sun} \)
- \( bErr: \text{This output is switched to TRUE if the parameters entered are erroneous.} \)
- \( sErrDescr: \text{Contains the error description.} \)

**Error description**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Error: ( r_{HiLmt} ) less than or equal to ( r_{LoLmt} ).</td>
<td></td>
</tr>
<tr>
<td>02: Error: ( r_{LoLmt} ) is less than 0(^\circ) or ( r_{HiLmt} ) is greater than 90(^\circ).</td>
<td></td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.1.3.3.12 FB_BA_RdFcdElemLst

With the help of this function block, data for facade elements (windows) can be imported from a pre-defined Excel table in csv format into the list of facade elements \([59]\). In addition the imported data are checked for plausibility and errors are written to a log file.

The following example shows the Excel table with the entries of the window elements. All text fields are freely writable. The fields marked in green are important; each line in these fields identifies a data set.

The following rules are to be observed:

- A data set must always start with a ‘\(@\)’.
- The indices \(\text{IndexColumn} \) and \(\text{IndexRow} \) must lie within the defined limits, see List of facade elements \([59]\). These indices directly describe the facade element in the list \(\text{arrFcdElem} \) to which the data from the set are saved.
- Window width and window height must be greater than zero
- The corner coordinates \( P1x \) and \( P1y \) must be greater than or equal to zero.
- Each window element must be assigned to a group 1..255.
- For system-related reasons the total size of the table may not exceed 65534 bytes.
- This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)".

It is not necessary to describe all window elements that would be possible by definition or declaration. Before the new list is read in, the function block deletes the entire old list in the program. All elements that are not described by entries in the Excel table then have pure zero entries and are thus marked as non-existent and also non-evaluable, since the function block for shading correction, `FB_BA_Shading[83]`, does not accept elements with the group entry '0'.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Description</td>
<td>IndexColumn</td>
<td>IndexRow</td>
<td>Window-Width</td>
<td>Window-Height</td>
<td>P1x</td>
<td>P1y</td>
<td>Group</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Description</td>
<td>@</td>
<td>1</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>2.7</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Description</td>
<td>@</td>
<td>3</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>4.4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Description</td>
<td>@</td>
<td>4</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>6.1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Description</td>
<td>@</td>
<td>5</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>7.8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>Description</td>
<td>@</td>
<td>6</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>9.5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>Description</td>
<td>@</td>
<td>7</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>11.2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>Description</td>
<td>@</td>
<td>8</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>12.9</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>Description</td>
<td>@</td>
<td>9</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>14.6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>Description</td>
<td>@</td>
<td>10</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>16.3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>Description</td>
<td>@</td>
<td>11</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>Description</td>
<td>@</td>
<td>12</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>2.7</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>Description</td>
<td>@</td>
<td>13</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>4.4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>Description</td>
<td>@</td>
<td>14</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>6.1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>Description</td>
<td>@</td>
<td>15</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>7.8</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>Description</td>
<td>@</td>
<td>16</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>9.5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>Description</td>
<td>@</td>
<td>17</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>11.2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>Description</td>
<td>@</td>
<td>18</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>12.9</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>Description</td>
<td>@</td>
<td>19</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>14.6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>23</td>
<td>Description</td>
<td>@</td>
<td>20</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>16.3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>Description</td>
<td>@</td>
<td>21</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.5</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>Description</td>
<td>@</td>
<td>22</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>2.7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>26</td>
<td>Description</td>
<td>@</td>
<td>23</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>4.4</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>27</td>
<td>Description</td>
<td>@</td>
<td>24</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>6.1</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>28</td>
<td>Description</td>
<td>@</td>
<td>25</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>7.8</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>29</td>
<td>Description</td>
<td>@</td>
<td>26</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>9.5</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>30</td>
<td>Description</td>
<td>@</td>
<td>27</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>11.2</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>31</td>
<td>Description</td>
<td>@</td>
<td>28</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>12.9</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td>Description</td>
<td>@</td>
<td>29</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>14.6</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>33</td>
<td>Description</td>
<td>@</td>
<td>30</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>16.3</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>34</td>
<td>Description</td>
<td>@</td>
<td>31</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>35</td>
<td>Description</td>
<td>@</td>
<td>32</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>2.7</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>36</td>
<td>Description</td>
<td>@</td>
<td>33</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>4.4</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>37</td>
<td>Description</td>
<td>@</td>
<td>34</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>6.1</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>38</td>
<td>Description</td>
<td>@</td>
<td>35</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>7.8</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>39</td>
<td>Description</td>
<td>@</td>
<td>36</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>9.5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>Description</td>
<td>@</td>
<td>37</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>11.2</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>41</td>
<td>Description</td>
<td>@</td>
<td>38</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>12.9</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>42</td>
<td>Description</td>
<td>@</td>
<td>39</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>14.6</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>43</td>
<td>Description</td>
<td>@</td>
<td>40</td>
<td>1</td>
<td>1.2</td>
<td>1.3</td>
<td>16.3</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>
Log file

Each time the reading function block is restarted, the log file is rewritten and the old contents are deleted. If there is no log file, it will be automatically created first. The log file then contains either an OK message or a list of all errors that have occurred. Errors connected with the opening, writing or closing of the log file itself cannot be written at the same time. Therefore, always note the output sErrDescr of the reading function block that indicates the last error code. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.

Program sample

```plaintext
PROGRAM ReadFacadeElements
VAR
  bInit : BOOL;
  bRead : R_TRIG;
  fbReadFacadeElementList : FB_BA_RdFacadeElemList;
  arrFacadeElement : ARRAY [1..uiMaxColumnFcd, 1..uiMaxRowFcd] OF ST_BA_FcdElem;
  bBusy : BOOL;
  udiAmountOfSetsRead : UDINT;
  bError : BOOL;
  sErrorDescr : T_MaxString;
  bErrDataSet : BOOL;
END_VAR
```

In this sample the variable `bInit` is initially set to TRUE when the PLC starts. Hence, the input `bStt` of the function block `fbReadFacadeElementList` receives a once-only rising edge that triggers the reading process. The file "FacadeElements.csv" is read, which is located in the folder "C:\Projects\". The log file "Logfile.txt" is then saved in the same folder. If this log file does not yet exist it will be created, otherwise the existing contents are overwritten. Reading and writing take place on the same computer on which the PLC is located. This is defined by the input `sNetID = "` (=local). All data are written to the list `arrFcdElem` declared in the program. During reading and writing the output `bBusy` is set to TRUE. The last file handling error that occurred is displayed at `sErrDescr`; `bErr` is TRUE. If an error is detected in the data set, this is displayed at `bErrDataSet` and described in more detail in the log file. The number of found and read data rows is displayed at `udiAmountSetsRd` for verification purposes.

The errors marked were "built into" the following Excel list. This gives rise to the log file shown:
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_ShdoEntry [86] and allocated an error description. The third and the fourth errors likewise occurred after the internal validation.

Important here is that the data set numbers (in this case 22 and 24) do not go by the numbers entered in the list, but by the actual sequential numbers: only 30 data sets were read in here.

VAR_INPUT

- bStt: A TRUE edge on this input starts the reading process.
- sDataFile: Contains the path and file name for the data file to be opened. This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)". If the file is opened with a simple text editor, the values must be separated by semicolons. Example of an entry: sDataFile: = ‘C:\Projekte\FacadeElements.csv’
- sLogFile: ditto. Log file for the accumulating errors. This file is overwritten each time the function block is activated, so that only current errors are contained.
- sNetId: A string can be entered here with the AMS Net ID of the TwinCAT computer on which the files are to be written/read. An empty string can be specified for the local computer (see T_AmsNetId).
The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>udiAmtSetsRd</td>
<td>UDINT</td>
</tr>
<tr>
<td>bErr</td>
<td>BOOL</td>
</tr>
<tr>
<td>sErrDescr</td>
<td>T_MAXSTRING</td>
</tr>
<tr>
<td>bErrDataSet</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bBusy:** This output is TRUE as long as elements are being read from the file.

**udiAmtSetsRd:** Number of data sets read.

**bErr:** This output is switched to TRUE, if a file write or read error has occurred.

**sErrDescr:** Contains the error description.

**Error description**

<table>
<thead>
<tr>
<th>Error number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>File handling error: Opening the log file - the ADS error number is stated.</td>
</tr>
<tr>
<td>02</td>
<td>File handling error: Open the data file - the ADS error number is stated.</td>
</tr>
<tr>
<td>03</td>
<td>File handling error: Reading the data file - the ADS error number is stated.</td>
</tr>
<tr>
<td>04</td>
<td>Error: During reading of the data file it was determined that the file is too large (number of bytes larger than udiMaxDataFileSize)</td>
</tr>
<tr>
<td>05</td>
<td>File handling error: Writing to the log file - the ADS error number is stated.</td>
</tr>
<tr>
<td>06</td>
<td>File handling error: Closing the data file - the ADS error number is stated.</td>
</tr>
<tr>
<td>07</td>
<td>File handling error: Writing to the log file (OK message if no errors were detected) - the ADS error number is stated.</td>
</tr>
<tr>
<td>08</td>
<td>File handling error: Closing the log file - the ADS error number is stated.</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrFcdElem</td>
<td>ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;</td>
</tr>
</tbody>
</table>

**arrFcdElem:** List of facade elements [59].

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.3.3.13 FB_BA_RdShdObjLst**

```
FB_BA_RdShdObjLst
  bStt  bBusy
  sDataFile udiAmtSetsRd
  sLogFile bErr
  sNetId sErrDescr
  arrShdObj bErrDataSet
```

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 [59]. In addition the imported data are checked for plausibility and errors are written to a log file.
The following example shows the Excel table with the entries of the window elements. All text fields are freely writable. The fields marked in green are important; each line in these fields identifies a data set. The columns G to J have a different meaning depending on whether the type rectangle or ball is concerned. The columns K to M are to be left empty in the case of balls. With regard to the rectangle coordinates, only the relevant data are entered and the remainder are internally calculated, see FB_BA_ShodObjEntry [P 86].

The following rules are to be observed:

- A data set must always start with a '@'.
- The monthly entries must not be 0 or greater than 12; all other combinations are possible.
- **Examples:**
  - Start=1, End=1: shading in January.
  - Start=1, End=5: shading from the beginning of January to the end of May.
  - Start=11, End=5: shading from the beginning of November to the end of May (the following year).
- Window width and window height must be greater than zero.
- The z-coordinates P1z and P3z or Mz must be greater than zero.
- The radius must be greater than zero.
- For system-related reasons the total size of the table may not exceed 65534 bytes.
- This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)".

Is not necessary to describe all shading objects that are possible per facade. Only those contained in the list ultimately take effect.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Description</td>
<td>Type</td>
<td>Begin</td>
<td>End</td>
<td>P1x/Mx</td>
<td>P1y/Mx</td>
<td>P1z/Mz</td>
<td>P2y/R</td>
<td>P3x</td>
<td>P3y</td>
<td>P3z</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 - Tetragon (Month)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>36.06</td>
<td>11</td>
<td>-70.71</td>
<td>11</td>
<td>68.59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 - Globe</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>-94.75</td>
<td>0</td>
<td>36.06</td>
<td>11</td>
<td>-70.71</td>
<td>11</td>
<td>68.59</td>
</tr>
<tr>
<td>3</td>
<td>2 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>-23.33</td>
<td>0</td>
<td>9.9</td>
<td>10.5</td>
<td>-3.54</td>
<td>10.5</td>
<td>22.62</td>
</tr>
<tr>
<td>4</td>
<td>3 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>62.23</td>
<td>0</td>
<td>0</td>
<td>14.47</td>
<td>62.23</td>
<td>14.47</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>4 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>48</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>62.23</td>
<td>14.47</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>11 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>12 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>15 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>16 - Description</td>
<td>@</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>46</td>
<td>0</td>
<td>13</td>
<td>14.47</td>
<td>14.47</td>
<td>38.89</td>
<td></td>
</tr>
</tbody>
</table>

Log file

Each time the reading function block is restarted, the log file is rewritten and the old contents are deleted. If there is no log file, it will be automatically created first. The log file then contains either an OK message or a list of all errors that have occurred. Errors connected with the opening, writing or closing of the log file itself cannot be written at the same time. Therefore, always note the output sErrDescr of the reading function block that indicates the last error code. Since the log file is always closed last during the reading process, a corresponding alarm is ensured in the event of an error.
In this sample the variable \textit{bInit} is initially set to TRUE when the PLC starts. Hence, the input \textit{bStt} on the function block \textit{fbReadShadingObjects} receives a once-only rising edge that triggers the reading process. The file "ShadingObjects.csv" is read, which is located in the folder "C:\Projects". The log file "Logfile.txt" is then saved in the same folder. If this log file does not yet exist it will be created, otherwise the existing contents are overwritten. Reading and writing take place on the same computer on which the PLC is located. This is defined by the input \textit{sNetID} = " (=local). All data are written to the list \textit{arrShdObj} declared in the program. During reading and writing the output \textit{bBusy} is set to TRUE. The last file handling error that occurred is displayed at \textit{sErrDescr}; \textit{bErr} is TRUE. If an error is detected in the data set, this is displayed at \textit{bErrDataSet} and described in more detail in the log file. The number of found and read data rows is displayed at \textit{udiAmtSetsRd} for verification purposes.

The errors marked were built into the following Excel list. This gives rise to the log file shown:
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 [86] and allocated an error description. The third error likewise occurred after the internal validation.

---

VAR_INPUT

bStt : BOOL;
sDataFile : STRING;
sLogFile : STRING;
sNetId : T_AmsNetId;;

bStt: A TRUE edge on this input starts the reading process.

sDataFile: Contains the path and file name for the data file to be opened. This must have been saved in Excel as file type "CSV (comma-separated values) (*.csv)". If the file is opened with a simple text editor, the values must be separated by semicolons. Example of an entry: sDataFile := 'C:\Projects\ShadingObjects.csv'

sLogFile: ditto. Log file for the accumulating errors. This file is overwritten each time the function block is activated, so that only current errors are contained.

sNetId: A string can be entered here with the AMS Net ID of the TwinCAT computer on which the files are to be written/read. An empty string can be specified for the local computer (see T_AmsNetId).

The data can be saved only on the control computer itself and on the computers that are connected by ADS to the control computer. Links to local hard disks in this computer are possible, but not to connected network hard drives.

---

VAR_OUTPUT

bBusy : BOOL;
udiAmtSetsRd : UDINT;
bErr : BOOL;
sErrDescr : T_MAXSTRING;
bErrDataSet : BOOL;

bBusy: Indicates that the reading process is active.
bBusy: This output is TRUE as long as elements are being read from the file.

udiAmtSetsRd: Number of data sets read.

bErr: This output is switched to TRUE, if a file write or read error has occurred.

sErrMsg: Contains the error description.

<table>
<thead>
<tr>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: File handling error: Opening the log file - the ADS error number is stated.</td>
</tr>
<tr>
<td>02: File handling error: Open the data file - the ADS error number is stated.</td>
</tr>
<tr>
<td>03: File handling error: Reading the data file - the ADS error number is stated.</td>
</tr>
<tr>
<td>04: Error: During reading of the data file it was determined that the file is too large (number of bytes larger than udiMaxDataFileSize)</td>
</tr>
<tr>
<td>05: File handling error: Writing to the log file - the ADS error number is stated.</td>
</tr>
<tr>
<td>06: File handling error: Closing the data file - the ADS error number is stated.</td>
</tr>
<tr>
<td>07: File handling error: Writing to the log file (OK message if no errors were detected) - the ADS error number is stated.</td>
</tr>
<tr>
<td>08: File handling error: Closing the log file - the ADS error number is stated.</td>
</tr>
</tbody>
</table>

bErrDataSet: This output is set to TRUE, if the read data sets are faulty. Further details are entered in the log file.

VAR_IN_OUT

arrShdObj: ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;

arrShdObj: List of shading objects [59].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.14 FB_BA_RolBldActr

This function block is used to position a roller shutter over two outputs: up and down. The positioning telegram stSunBld [183] can be used to move the roller shutter to any position. In addition, the positioning telegram stSunBld [183] offers manual commands, which can be used to move the roller shutter to particular positions. These manual commands are controlled by the function block FB_BA_SunBldSwi [102].

Structure of the blind positioning telegram stSunBld [183].

```
TYPE ST_BA_SunBld:
  STRUCT
    rPos        : REAL;
    rAngl       : REAL;
    bManUp      : BOOL;
    bManDwn     : BOOL;
```
The current height position is not read in by an additional encoder; it is determined internally by the runtime of the roller shutter. The two runtime parameters \(u\text{di}Ti\text{Up}\) (roller shutter travel-up time [ms]) and \(u\text{di}Ti\text{Dwn}\) (roller shutter travel-down time [ms]) take account of the different movement characteristics.

As a rule, the function block controls the roller shutter based on the information from the positioning telegram \(st\text{SunBld}\) [183]. If automatic mode is active \((b\text{ManMod}=\text{FALSE})\), the roller shutter always moves to the current position; changes are reflected immediately. In manual mode \((b\text{ManMod}=\text{TRUE})\), the roller shutter is controlled by the commands \(b\text{ManUp}\) and \(b\text{ManDwn}\).

**Referencing**

Safe referencing refers to a situation when the roller shutter is upwards-controlled for longer than its complete travel-up time. The position is then always "0". Since roller shutter positioning without encoder is always error-prone, it is important to use automatic referencing whenever possible: Whenever "0" is specified as the target position, the roller shutter initially moves upwards normally, based on continuous position calculation. Once the calculated position value 0% is reached, the output \(b\text{Up}\) continues to be held for the complete travel-up time + 5s.

For reasons of flexibility there are now two possibilities to interrupt the referencing procedure: Until the calculated 0% position is reached, a change in position continues to be assumed and executed. Once this 0% position is reached, the roller shutter can still be moved with the manual "travel-down" command. These two sensible restrictions make it necessary for the user to ensure that the roller shutter is referenced safely whenever possible.

After a system restart, the function block executes a reference run. Completion of the initial referencing is indicated through a TRUE signal at output \(b\text{InitRefCmpl}\). The initial referencing can also be terminated through a manual "travel-down" command.

**VAR_INPUT**

- \(b\text{En}\) : BOOL; Enable input for the function block. As long as this input is TRUE, the actuator function block accepts and executes commands as described above. A FALSE signal on this input resets the control outputs \(b\text{Up}\) and \(b\text{Dwn}\) and the function block remains in a state of rest.
- \(st\text{SunBld}\) : ST_BA_Sunblind; Positioning telegram, see \(ST\_BA\_Sunblind\) [183].
- \(u\text{di}Ti\text{Up\_ms}\) : UDINT; Complete time for driving up [ms].
- \(u\text{di}Ti\text{Dwn\_ms}\) : UDINT; Complete time for driving down in ms.

**VAR_OUTPUT**

- \(b\text{Up}\) : BOOL; Roller shutter control output up.
- \(b\text{Dwn}\) : BOOL; Roller shutter control output down.
- \(r\text{ActlPos}\) : REAL; Current position in percent.
- \(b\text{Ref}\) : BOOL; Reference output.
- \(u\text{di}Ref\text{Ti\_sec}\) : UDINT; Complete time for reference.
- \(b\text{InitRefCmpl}\) : BOOL; Reference completion.
- \(b\text{Busy}\) : BOOL; Busy state.
- \(b\text{Err}\) : BOOL; Error state.
- \(s\text{ErrDescr}\) : STRING; Error description.
**bRef:** The roller shutter is in referencing mode, i.e. the output \(bUp\) is set for the complete travel-up time + 5s. Only a manual "down" command can move the roller shutter in the opposite direction and terminate this mode.

**udiRefTi_sec:** Referencing countdown display [s].

**bInitRefCompl:** Initial referencing process complete.

**bBusy:** A positioning or a referencing procedure is in progress.

**bErr:** This output is switched to TRUE if the parameters entered are erroneous.

**sErrDescr:** Contains the error description.

### Error description

<table>
<thead>
<tr>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Error: The total travel-up or travel-down time (udiTiUp_ms/udiTiDwn_ms) is zero.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

#### 5.1.3.3.15 FB_BA_ShdCorr

The function block is used to assess the shading of a group of windows on a facade.

The function block FB_BA_ShdCorr calculates whether a window group lies in the shadow of surrounding objects. The result, which is output at the output \(bGrpNotShdd\), can be used to judge whether sun shading makes sense for this window group.

The function block thereby accesses two lists, which are to be defined:

- The parameters that describe the shading elements that are relevant to the facade on which the window group is located. This list of shading objects \([59]\) 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 \([59]\) is accessed via the IN/OUT variable \(arrFcdElem\), since not only the window coordinates are read, but the function block FB_BA_ShdCorr also stores the shading information for each window corner in this list. In this way, the information can also be used in other parts of the application program.

On the basis of the facade orientation \((rFcdOrtn)\), the direction of the sun \((rAzm)\) and the height of the sun \((rElv)\), a calculation can be performed for each corner of a window to check whether this lies in a shaded area. A window group is considered to be completely shaded if all corners are shaded.

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):
The function block performs its calculations only if the sun is actually shining on the facade. Considering the drawing presented in the introduction, this is the case if:

Facade orientation < azimuth angle < facade orientation + 180°

In addition, a calculation is also not required, if the sun has not yet risen, i.e. the solar elevation is below 0°. In both cases the output $bFcdSunlit$ is set to FALSE.

The situation is different for the southern hemisphere. The following applies to the facade orientation (looking out the window):

<table>
<thead>
<tr>
<th>Line of sight</th>
<th>Facade orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>South</td>
<td>β=0°</td>
</tr>
<tr>
<td>East</td>
<td>β=90°</td>
</tr>
<tr>
<td>North</td>
<td>β=180°</td>
</tr>
<tr>
<td>West</td>
<td>β=270°</td>
</tr>
</tbody>
</table>

The internal calculation or the relationship between facade and sunbeam also changes:
To distinguish between the situation in the northern and southern hemisphere, set the input parameter \(bSouth\) to FALSE (northern hemisphere) or TRUE (southern hemisphere)

**VAR_INPUT**

- \(stTiActl\) : TIMESTRUCT;
- \(rFcdOrtn\) : REAL;
- \(rAzm\) : REAL;
- \(rElv\) : REAL;
- \(diGrpID\) : DINT;
- \(bSouth\) : BOOL;
- \(arrShdObj\) : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;

\(stTiActl\): Input of the current time - local time in this case, since this time takes into account the shaded months. If the UTC time (or GMT) is used, the month may change in the middle of the day, depending on the location on the earth (see TIMESTRUCT).

\(rFcdOrtn\): Facade orientation, see illustration above.

\(rAzm\): Direction of the sun at the time of observation [°].

\(rElv\): Solar altitude at the time of observation [°].

\(diGrpID\): Window group regarded. The group 0 is reserved here for unused window elements, see FB_BA_FcdElemEntry [p. 63]. A 0-entry would lead to an error output (bErr=TRUE). The function block is then not executed any further and \(bGrpNotShdd\) is set to FALSE.

\(bSouth\): FALSE: Calculations refer to conditions in the northern hemisphere - TRUE: In the southern hemisphere

\(arrShdObj\): List of shading objects [p. 59].

**VAR_OUTPUT**

- \(bGrpNotShdd\) : BOOL;
- \(bFcdSunlit\) : BOOL;
- \(bErr\) : BOOL;
- \(sErrDescr\) : T_MAXSTRING

\(bGrpNotShdd\): Is TRUE as long as the window group is not calculated as shaded.

\(bFcdSunlit\): This output is set to TRUE if the sun is shining on the facade. See description above.
bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

### Error description

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:</td>
<td>Error: The index of the window group usiGrpId under consideration is 0.</td>
</tr>
<tr>
<td>02:</td>
<td>Error: An element of the facade list is invalid. This is specified in the error description sErrDescr as arrFcdElem[nColumn,nRow].</td>
</tr>
</tbody>
</table>

### VAR_IN_OUT

arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;

arrFcdElem: List of facade elements [59].

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

#### 5.1.3.3.16 FB_BA_ShdObjEntry

![FB_BA_ShdObjEntry](image)

This function block serves for the administration of all shading elements in a facade, which is globally saved in a list of shading elements [59]. It is intended to facilitate the input of the element information - also with regard to the use of a visualization. A schematic illustration of the objects with description of the coordinates is given in Shading correction: principles and definitions [44].

The shading elements are declared in the global variables:

VAR_GLOBAL

```
arrShdObj : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;
```

END_VAR

Each individual element arrShdObj[1] to arrShdObj [uiMaxShdObj] carries the information for an individual shading element (ST_BA_ShdObj [182]). This information consists of the selected type of shading (rectangle or ball) and the respectively associated coordinates. For a rectangle, these are the corner points (rP1x, rP1y, rP1z), (rP2x, rP2y, rP2z), (rP3x, rP3y, rP3z) and (rP4x, rP4y, rP4z), for a sphere these are the center point ...
The phase of the shading can be defined via the inputs `udiBegMth` and `udiEndMth`, which is important in the case of objects such as trees that bear no foliage in winter.

The function block directly accesses the array of this information via the IN-OUT variable `arrShdObj`.

**Note:** The fact that the rectangle coordinates `rP2x`, `rP2z`, `rP4x`, `rP4y` and `rP4z` are output values results from the fact that they are formed from the input parameters:

\[ rP2x = rP1x; \quad rP2z = rP1z; \quad rP4x = rP3x; \quad rP4y = rP1y; \quad rP4z = rP3z; \]

That limits the input of a square to the extent that the lateral edges stand vertically on the floor (`rP2x = rP1x` and `rP4x = rP3x`), that the square has no inclination (`rP2z = rP1z` and `rP4z = rP3z`) and can only have a different height "upwards", i.e. in the positive y-direction (`rP4y = rP1y`).

The function block is used in three steps:

- **Read**
  - Selection of the element from the list `arrShdObj[udiId]` is based on the entry at `udiId`. A rising edge on `bRd` reads the data. These values are assigned to the input and output variables of the function block. These are the input values `rP1x`, `rP1y`, `rP1z`, `rP2y`, `rP3x`, `rP3y`, `rMx`, `rMy`, `rMz`, `rRads` and the object enumerator `eType` and the output values `rP2x`, `rP2z`, `rP4x`, `rP4y` and `rP4z`. It is important here that the input values are not overwritten in the reading step. Hence, all values can initially be displayed in a visualization.

- **Change**
  - In a next program step the listed input values can then be changed. If a rectangle is preselected at input `eType` via the value "eObjectTypeTetragon", the output values `rP2x`, `rP2z`, `rP4x`, `rP4y` and `rP4z` result from the rectangle coordinates that were entered (see above). The values entered are constantly checked for plausibility. The output `bErr` indicates whether the values are valid (`bErr=FALSE`). If the value is invalid, a corresponding error message is issued at output `sErrDescr`. If a rectangle is defined, only the inputs `rP1x`, `rP1y`, `rP1z`, `rP2y`, `rP3x`, `rP3y` and `rP3z` have to be described; the inputs `rMx`, `rMy`, `rMz` and `rRads` do not have to be linked. For a sphere definition, only `rMx`, `rMy`, `rMz` and `rRads` have to be described; the rectangle coordinates can remain unlinked.

- **Write**
  - The parameterized data are written to the list element with the index `udiId` upon a rising edge on `bWrt`, regardless of whether they represent valid values or not. The element structure `ST_BA_ShdObj` therefore contains a plausibility bit `bVld`, which forwards precisely this information to the function block `FB_BA_ShdCorr` to prevent miscalculations.

This approach is to be regarded only as a proposal. It is naturally also possible to parameterize the function block quite normally in one step and to write the values entered to the corresponding list element with a rising edge on `bWrt`.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>udiId</td>
<td>UDINT</td>
</tr>
<tr>
<td>bRd</td>
<td>BOOL</td>
</tr>
<tr>
<td>bWrt</td>
<td>BOOL</td>
</tr>
<tr>
<td>rP1x</td>
<td>REAL</td>
</tr>
<tr>
<td>rP1y</td>
<td>REAL</td>
</tr>
<tr>
<td>rP1z</td>
<td>REAL</td>
</tr>
<tr>
<td>rP2x</td>
<td>REAL</td>
</tr>
<tr>
<td>rP2y</td>
<td>REAL</td>
</tr>
<tr>
<td>rP3x</td>
<td>REAL</td>
</tr>
<tr>
<td>rP3y</td>
<td>REAL</td>
</tr>
<tr>
<td>rP3z</td>
<td>REAL</td>
</tr>
<tr>
<td>rMx</td>
<td>REAL</td>
</tr>
<tr>
<td>rMy</td>
<td>REAL</td>
</tr>
<tr>
<td>rMz</td>
<td>REAL</td>
</tr>
</tbody>
</table>
udilid: Index of the selected element. This refers to the selection of a field element of the array saved in the IN-OUT variable arrShdObj. The variable udilid must not be zero! This is due to the field definition, which starts with 1. However, an incorrect input is recognized and displayed as such at bErr/sErrDescr.

bRd: The information of the selected element, arrShdObj[udilid], is read into the function block with a positive edge at this input and assigned to the input variables rP1x to eType and the output variables rP2x to rP4z. If data are already present on the inputs rP1x to eType at this time, then the data previously read are immediately overwritten with these data.

bWrt: A positive edge writes the values applied to the inputs rP1x to eType and the values determined and assigned to the outputs rP2x to rP4z to the selected field element arrShdObj[udilid].

rP1x: X-coordinate of point 1 of the shading element (rectangle) [m].

rP1y: Y-coordinate of point 1 of the shading element (rectangle) [m].

rP1z: Z-coordinate of point 1 of the shading element (rectangle) [m].

rP2y: Y-coordinate of point 2 of the shading element (rectangle) [m].

rP3x: X-coordinate of point 3 of the shading element (rectangle) [m].

rP3y: Y-coordinate of point 3 of the shading element (rectangle) [m].

rP3z: Z-coordinate of point 3 of the shading element (rectangle) [m].

rMx: X-coordinate of the center of the shading element (ball) [m].

rMy: Y-coordinate of the center of the shading element (ball) [m].

rMz: Z-coordinate of the center of the shading element (ball) [m].

rRads: Radius of the shading element (ball) [m].

udiBegMth: Beginning of the shading period (month).

udiEndMth: End of the shading period (month).

eType: Selected type of element: Rectangle or sphere (see E_BA_ShdObjType[179]).

Remark about the shading period:

The entries for the months may not be 0 or greater than 12, otherwise all combinations are possible.

Examples:
Start=1, End=1: shading in January.
Start=1, End=5: shading from the beginning of January to the end of May.
Start=11, End=5: shading from the beginning of November to the end of May (the following year).

VAR_OUTPUT

rP2x : REAL;
rP2z : REAL;
rP4x : REAL;
rP4y : REAL;
rP4z : REAL;
bErr : BOOL;
sErrDescr : T_MAXSTRING;

rP2x: Calculated X-coordinate of point 2 of the shading element (rectangle) [m]. See "Note[86]" above.

rP2z: Calculated Z-coordinate of point 2 of the shading element (rectangle) [m]. See "Note[86]" above.

rP4x: Calculated X-coordinate of point 4 of the shading element (rectangle) [m]. See "Note[86]" above.

rP4y: Calculated Y-coordinate of point 4 of the shading element (rectangle) [m]. See "Note[86]" above.
rP4z: Calculated Z-coordinate of point 4 of the shading element (rectangle) [m]. See "Note [86] above.

bErr: Result of the plausibility check for the values entered. For a rectangle, the internal angle is 360° and the points are in a plane in front of the facade under consideration. In the case of a ball the center must likewise lie in front of the facade and the radius must be greater than zero.

sErrDescr: Contains the error description.

<table>
<thead>
<tr>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Error: The input uIdld is outside the permissible limits 1..uiMaxShdObj.</td>
</tr>
<tr>
<td>02 Error: The sum of the angles of the rectangle is not 360°. This means that the corners are not in the order P1, P2, P3 and P4 but rather P1, P3, P2 and P4. This results in a crossed-over rectangle.</td>
</tr>
<tr>
<td>03: Error: The corners of the square are not on the same level.</td>
</tr>
<tr>
<td>04: Error: The z-component of P1 is less than zero. This corner would thus lie behind the facade.</td>
</tr>
<tr>
<td>05: Error: The z-component of P3 is less than zero. This corner would thus lie behind the facade.</td>
</tr>
<tr>
<td>06: Error: P1 is equal to P2. The object entered is thus not a rectangle.</td>
</tr>
<tr>
<td>07: Error: P1 is equal to P3. The object entered is thus not a rectangle.</td>
</tr>
<tr>
<td>08: Error: P1 is equal to P4. The object entered is thus not a rectangle.</td>
</tr>
<tr>
<td>09: Error: P2 is equal to P3. The object entered is thus not a rectangle.</td>
</tr>
<tr>
<td>10: Error: P2 is equal to P4. The object entered is thus not a rectangle.</td>
</tr>
<tr>
<td>11: Error: P3 is equal to P4. The object entered is thus not a rectangle.</td>
</tr>
<tr>
<td>12: Error: The radius entered is zero.</td>
</tr>
<tr>
<td>13: Error: The z-component of the ball center is less than zero. This point would thus lie behind the facade.</td>
</tr>
<tr>
<td>14: Error: Error object type eType - neither rectangle nor ball.</td>
</tr>
<tr>
<td>15: Error: Month input error.</td>
</tr>
</tbody>
</table>

VAR_IN_OUT

arrShdObj : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;

arrShdObj: List of shading objects [59].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.17 FB_BA_SunBldActr

<table>
<thead>
<tr>
<th>FB_BA_SunBldActr</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn</td>
</tr>
<tr>
<td>bUp</td>
</tr>
<tr>
<td>stSunbld</td>
</tr>
<tr>
<td>bDwn</td>
</tr>
<tr>
<td>udiTUp_ms</td>
</tr>
<tr>
<td>rActIPos</td>
</tr>
<tr>
<td>udiTDown_ms</td>
</tr>
<tr>
<td>rActIAngl</td>
</tr>
<tr>
<td>udiTumTUp_ms</td>
</tr>
<tr>
<td>bRef</td>
</tr>
<tr>
<td>udiTumTDown_ms</td>
</tr>
<tr>
<td>udiRefTi_sec</td>
</tr>
<tr>
<td>udiBckLshTUp_ms</td>
</tr>
<tr>
<td>bInitRefCompl</td>
</tr>
<tr>
<td>udiBckLshTDown_ms</td>
</tr>
<tr>
<td>bEbusy</td>
</tr>
<tr>
<td>rAnglLmtUp</td>
</tr>
<tr>
<td>bErr</td>
</tr>
<tr>
<td>rAnglLmtDwn</td>
</tr>
<tr>
<td>sErrDescr</td>
</tr>
</tbody>
</table>
This function block is used for positioning of a louvered blind via two outputs: drive up and drive down. The blind can be driven to any desired (height) position and louvre angle via the positioning telegram \texttt{stSunBld [\ref{183}]}. On top of that, the positioning telegram \texttt{stSunBld [\ref{183}]} also contains manual commands with which the blind can be moved individually to certain positions. These manual commands are controlled by the function block \texttt{FB_BA_SunBldSwi [\ref{102}]}.

Structure of the blind positioning telegram \texttt{stSunBld [\ref{183}]}.

\begin{verbatim}
TYPE ST_BA_SunBld:
  STRUCT
    rPos     : REAL;
    rAngl    : REAL;
    bManUp   : BOOL;
    bManDwn  : BOOL;
    bManMod  : BOOL;
    bActv    : BOOL;
  END_STRUCT
END_TYPE
\end{verbatim}

The current height position and the louvre angle are not read in by an additional encoder, but determined internally by the travel time of the blind. The calculation is based on the following travel profile (regarded from the highest and lowest position of the blind):

\textbf{Downward travel profile:}

1) The blind is in the uppermost position

2) The backlash was moved out. The blind was driven down a little bit without turning the lamellas.

3) The lamellas are turned to the lowest angle.

4) The blind is completely driven down.

More detailed explanations of the terms "backlash" and "turning" are given here in the downward movement:

The blind normally describes its downward movement with the louvre low point directed outwards, as in fig. 3). If the blind is in an initial position with the low point directed inwards (i.e. after the conclusion of an upward movement), then a certain time elapses after a new downward movement begins before the louvres start to turn from the "inward low point" to the "outward low point". During this time the louvre angle does not change; the blind only drives downward (fig.1 and fig. 2). This time is an important parameter for the movement calculation and is entered in the function block under \texttt{udiBckLshTiDwn_ms [ms]}. Since it is not known at any point after a blind movement of any length whether backlash has already taken effect, the backlash of the downward movement or its travel time can be measured most reliably if the blind was first raised fully. A further important parameter is the time interval of the subsequent turning of the louvres from the "Outward low point" to the "Inward low point". This time should be entered as \texttt{udiTurnTiDwn_ms [ms]} at the function block.
Upward travel profile:

1) The blind is in the lowermost position
2) The backlash was moved out. The blind was driven up a little bit without turning the lamellas.
3) The lamellas are turned to the highest angle.
4) The blind is completely driven up.

More detailed explanations of the terms "backlash" and "turning" are given here in the upward movement:

The circumstances are similar to the downward movement described above: The blind normally describes its upward movement with the louvre low point directed inwards, as in fig. 3). If the blind is in an initial position with the low point directed outwards (i.e. after the conclusion of a downward movement), then a certain time elapses after a new upward movement begins before the louvres start to turn from the "Outward low point" to the "Inward low point". During this time the louvre angle does not change; the blind only drives upward (fig. 1 and fig. 2). Also this time is an important parameter for the movement calculation and is entered in the function block under $udiBckLshTiUp_{ms}$ [ms]. Since it is not known at any point after a blind movement of any length whether backlash has already taken effect, the backlash of the upward movement or its travel time can be measured most reliably if the blind was first driven fully downward. A further important parameter is the time interval of the subsequent turning of the louvres from the "Outward low point" to the "Inward low point". This time should be entered as $udiTurnTiUp_{ms}$ [ms] at the function block.

Parameterization

For the calculation of the (height) position and the louvre angle, the following times now have to be determined for both the upward and downward movement:

- the travel time of the backlash ($udiBckLshTiUp_{ms} / udiBckLshTiDwn_{ms}$ [ms])
- the turning duration ($udiTurnTiUp_{ms} / udiTurnTiDwn_{ms}$ [ms])
- the total travel time ($udiTiUp_{ms} / udiTiDwn_{ms}$ [ms])

Furthermore the following are required for the calculation:

- the highest louvre angle after turning upwards ($rAnglLmtUp$ ['°'])
- the lowest louvre angle after turning downwards ($rAnglLmtDwn$ ['°'])

The louvre angle $\lambda$ 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. If automatic mode is active (bManMod=FALSE), then the current position and louvre angle are always driven to, wherein changes are immediately accounted for. The height positioning takes priority: First the entered height and afterwards the louvre angle are driven to. For reasons of the simplicity the position error due to the angle movement is disregarded. In manual mode (bManMod=TRUE), the blind is controlled by the commands bManUp and bManDwn. An automatic movement command is triggered whenever a change from manual to automatic mode occurs.

Referencing

Secure referencing is ensured if the blind is driven upward for longer than its complete drive-up time. The position is then in any case "0" and the louvre angle is at its maximum. Since blind positioning without an encoder is naturally always susceptible to error, it is important to automatically reference as often as possible: each time the "0" position is to be driven to (the angle is unimportant), the blind initially drives upward quite normally with continuous position calculation. Once the calculated position value 0% is reached, the output bUp continues to be held for the complete travel-up time + 5 s. For reasons of flexibility, there are two ways to interrupt the referencing process: Until the calculated 0% position is reached, a change in position continues to be assumed and executed. Once this 0% position is reached, the blind can still be moved with the manual "travel-down" command. These two sensible limitations make it necessary for the user to ensure that the blind is securely referenced as often as possible.

After a system restart, the function block executes a reference run. Completion of the initial referencing is indicated through a TRUE signal at output bInitRefCmpl. The initial referencing can also be terminated through a manual "travel-down" command.

Target accuracy

Since the function block determines the blind position solely via run times, the cycle time of the PLC task plays a crucial role for positioning accuracy. If the switching time for a louvre angle range of -70° to 10° is 1 second, for example, the accuracy at a cycle time of 50 ms is +/-4°.

VAR_INPUT

<table>
<thead>
<tr>
<th>bEn</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>stSunbld</td>
<td>ST_BA_SunBld;</td>
</tr>
<tr>
<td>udiTiUp_ms</td>
<td>UDINT;</td>
</tr>
<tr>
<td>udiTiDwn_ms</td>
<td>UDINT;</td>
</tr>
<tr>
<td>udiTurnTiUp_ms</td>
<td>UDINT;</td>
</tr>
<tr>
<td>udiTurnTiDwn_ms</td>
<td>UDINT;</td>
</tr>
<tr>
<td>udiBckLshTiUp_ms</td>
<td>UDINT;</td>
</tr>
<tr>
<td>udiBckLshTiDwn_ms</td>
<td>UDINT;</td>
</tr>
<tr>
<td>rAnglLmtUp</td>
<td>REAL;</td>
</tr>
<tr>
<td>rAnglLmtDwn</td>
<td>REAL;</td>
</tr>
</tbody>
</table>

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).
udiTiUp_ms: Complete time for driving up [ms].

udiTiDwn_ms: Complete time for driving down [ms].

udiTurnTiUp_ms: Time for turning the louvres in the upward direction [ms].

udiTurnTiDwn_ms: Time for turning the louvres in the downward direction [ms].

udiBckLshTiUp_ms: Time to traverse the backlash in the upward direction [ms]. This input is internally limited to a minimum value of 0.

udiBckLshTiDwn_ms: Time to traverse the backlash in the downward direction [ms]. This input is internally limited to a minimum value of 0.

rAnglLmtUp: Highest position of the louvres ['°].

This position is reached once the blind has moved to the top position.
The louvre angle \( \lambda \), as defined above, is then typically greater than zero.

\[ \lambda \]

Inside

Outside

rAnglLmtDwn: Lowest position of the louvres ['°].

This position is reached once the blind has moved to the bottom position.
The louvre angle \( \lambda \), as defined above, is then typically less than zero.

\[ \lambda \]

Inside

Outside

**VAR_OUTPUT**

bUp : BOOL;
bDwn : BOOL;
rAct1Pos : REAL;
rAct1Angl : REAL;
bRef : BOOL;
udiRefTi_sec : UDINT;
bInitRefCompl : BOOL;
bBusy : BOOL;
bErr : BOOL;
sErrDesc : T_MAXSTRING;

bUp: Control output for blind up.

bDwn: Control output for blind down.

rActlPos: Current position in percent.

rActlAngl: Current louvre angle [°].

bRef: The blind is referencing, i.e. the output bUp is set for the complete travel-up time + 5s. Only a manual "down" command can move the blind in the opposite direction and terminate this mode.

udiRefTi_sec: Referencing countdown display [s].

bInitRefCompl: Initial referencing process complete.

bBusy: A positioning or a referencing procedure is in progress.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDesc: Contains the error description.

**Error description**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Error: Up/Down timer = 0.</td>
</tr>
<tr>
<td>02</td>
<td>Error: Turning timer = 0.</td>
</tr>
<tr>
<td>03</td>
<td>Error: Louvre angle limits: The upper limit is less than or equal to the lower limit (rAnglLmtUp &lt;= rAnglLmtDwn).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.18 FB_BA_SunBldEvt

This function block serves to preset the position and angle for any desired event. It can be used, for example, in order to drive to a parking position or to drive the blind upward for maintenance.

The function is activated via the input bEn. If this is the case, the active flag in the positioning telegram (bActv in stSunBld) at output stSunBld is set, and the values entered for the In/Out variables rPos for the blind height [%] and rAngl the louvre angle [°] are passed on in this telegram. If the function is no longer active due to the resetting of bEn, then the active flag in the positioning telegram stSunBld is reset and the positions for height and angle are set to "0". The priority function block (e.g. FB_BA_SunBldPrioSwi4) enables a function with lower priority to take over the control by resetting.

**VAR_INPUT**

bEn : BOOL;
rPos : REAL;
rAngl : REAL;

bEn: A TRUE signal on this input activates the function block and transfers the entered setpoint values together with the active flag in the positioning telegram ST_BA_SunBld [183]. A FALSE signal resets the active flag again and sets position and angle to zero.
rPos: Height position of the blind [%] in case of activation.

rAngl: Louvre angle of the blind [°] in case of activation.

**VAR_OUTPUT**

| stSunBld  | ST_BA_SunBld; |
| bActv    | BOOL;        |

bActv: Corresponds to the boolean value bActv in the blind telegram ST_BA_SunBld [183] 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 [183]

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.3.3.19 FB_BA_SunBldIcePrtc**

The function block FB_BA_SunBldIcePrtc deals with direction-independent anti-freezing.

The weather protection has the highest priority in the blind controller (see overview [52]) and is intended to ensure that the blind is not damaged by ice or wind.

Impending icing up is detected by the fact that, during precipitation detection at bRainSns, the measured outside temperature rOtsT is below the frost limit rFrstT. This event is saved internally and remains active until it is ensured that the ice has melted again. In addition, the outside temperature must have exceeded the frost limit value for the entered deicing time udiDeiceTi_sec [s]. For safety reasons the icing event is persistently saved, i.e. also beyond a PLC failure. Thus, if the controller fails during the icing up or deicing period, the blind is considered to be newly iced up when then the controller restarts and the deicing timer starts from the beginning again.

If there is a risk of icing, the blind is moved to the protective position specified by rPosProt (height position in percent) and rAnglProt (louvre angle [°]).

**VAR_INPUT**

| bEn       | BOOL;   |
| rOtsT     | REAL;   |
| bRainSns  | BOOL;   |
| rFrstT    | REAL;   |
| udiDeiceTi_sec | UDINT; |
| rPosProt  | REAL;   |
| rAnglProt | REAL;   |

bEn: The function block has no function if this input is FALSE. In the positioning telegram ST_BA_SunBld [183] 0 is output for the position and the angle, and bActv is FALSE. This means that another function takes over control of the blind via the priority controller.

rOtsT: Outside temperature [°C].

bRainSns: Input for a rain sensor.
**rFrstT:** Icing up temperature limit value \[°\] Celsius. This value may not be greater than 0. Otherwise an error is output.

**udiDeiceTi_sec:** Time until the deicing of the blind after icing up [s]. After that the icing up alarm is reset.

**rPosProt:** Height position of the blind [%] in the case of protection.

**rAnglProt:** Louvre angle of the blind \[°\] in the case of protection.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>stSunBld</th>
<th>: ST_BA_SunBld;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bActv</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>bIceAlm</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>udiRemTiIceAlm_sec</td>
<td>: UDINT;</td>
</tr>
</tbody>
</table>

**stSunBld**: Output structure of the blind positions, see ST_BA_SunBld \[183\].

**bActv**: Corresponds to the boolean value \(bActv\) in the blind telegram ST_BA_SunBld \[183\] and is solely used to indicate whether the function block sends an active telegram.

**bIceAlm**: Indicates the icing up alarm.

**udiRemTiIceAlm_sec**: In the case of impending icing up \((bIceAlm=TRUE)\), this second counter is set to the deicing time. As soon as the temperature lies above the frost point entered \((rFrstT)\), the remaining number of seconds until the 'all-clear' signal is given \((bIceAlm=FALSE)\) are indicated here. This output is 0 as long as no countdown of the time is taking place.

If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.3.3.20 FB_BA_SunBldPosDly**

This function block delays changes in position based on automatic commands.

If an event, e.g. weather protection, results in too many blind drives being started at the same time, fuses may be triggered by motor starting current peaks. It is therefore advisable to start the blind drives slightly staggered, in order to avoid excessive total current values.

This function block relays automatic commands from the input telegram stIn \[183\] to the output telegram stOut \[183\] with a delay. A distinction is made between three cases

1. the blind position \(rPos\) has changed in automatic mode \((bManMode=FALSE\) in telegram stIn\)
2. the louvre angle \(rAngl\) has changed in automatic mode \((bManMode=FALSE\) in telegram stIn\)
3. manual mode has just been exited, i.e. automatic mode has just become active (falling edge \(bManMode\) in telegram stIn)

The output telegram stOut is always a direct copy of the input telegram stIn. In these three cases, however, the output telegram stOut is set for the time udiDly_ms [ms].
This ensures that the blind controlled via the function block FB BA_SunBldActr [89] 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:

![Diagram of FB BA_SunBldPoeDly and FB BA_SunBldActr]

**VAR_INPUT**

```plaintext
VAR_INPUT
stIn : ST_BA_Sunblind;
udiDly_ms : UDINT;
```

- **stIn**: Input positioning telegram, see ST_BA_SunBld [183].
- **udiDly_ms**: Delay time of the active bit in the positioning telegram [ms].

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
stOut : ST_BA_Sunblind;
udiRemTiDly_sec : UDINT;
```

- **stOut**: Output positioning telegram, see ST_BA_SunBld [183].
- **udiRemTiDly_sec**: Display output for elapsed delay time [s].

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.3.3.21 FB BA_SunBldPrioSwi**

The function block is used for priority control for up to 4 positioning telegrams (stSunBld_Prio1 ... stSunBld_Prio4) of type ST_BA_SunBld [183].

Structure of the blind positioning telegram ST_BA_SunBld [183].

```plaintext
TYPE ST_BA_SunBld:
STRUCT
  rPos : REAL;
  rAngl : REAL;
  bManUp : BOOL;
END_STRUCT;
```

TC3 Building Automation  Version: 1.0  97
Up to 4 positioning telegrams from different control function blocks can be applied to this function block. The telegram on \textit{stSunBld\_Prio1} has the highest priority and that on \textit{stSunBld\_Prio4} the lowest. The active telegram with the highest priority is output at the output \textit{stSunBld}. "Active" means that the variable \textit{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. \textit{rPos}=0, \textit{rAngl}=0, \textit{bManUp}=FALSE, \textit{bManDwn}=FALSE, \textit{bManMod}=FALSE, \textit{bActv}=FALSE. Since the blind function block \textit{FB\_BA\_SunBldActr} \[89\] or the roller blind function block \textit{FB\_BA\_RolBldActr} \[81\] does not take account of the flag \textit{bActv}, this telegram would be interpreted as movement command to position "0", i.e. fully open. The absence of an activetelegram therefore does not represent a safety risk for the blind.

**VAR\_INPUT**

\begin{verbatim}
stSunBld\_Prio1 : ST\_BA\_SunBld;
stSunBld\_Prio2 : ST\_BA\_SunBld;
stSunBld\_Prio3 : ST\_BA\_SunBld;
stSunBld\_Prio4 : ST\_BA\_SunBld;
\end{verbatim}

\textit{stSunBld\_Prio1..stSunBld\_Prio4}: Positioning telegrams available for selection. \textit{stSunBld\_Prio1} has the highest priority and \textit{stSunBld\_Prio4} the lowest.

**VAR\_OUTPUT**

\begin{verbatim}
stSunBld : ST\_BA\_SunBld;
udiActvPrio : UDINT;
\end{verbatim}

\textit{stSunBld}: Resulting positioning telegram.
\textit{udiActvPrio}: Active positioning telegram. If none is active, "0" is output.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.22 \textit{FB\_BA\_SunBldPrioSwi8}

The function block is used for priority control for up to 8 positioning telegrams (\textit{stSunBld\_Prio1} ... \textit{stSunBld\_Prio8}) of type \textit{ST\_BA\_SunBld} \[183\].

Structure of the blind positioning telegram \textit{ST\_BA\_Sunbld} \[183\].

\begin{verbatim}
TYPE ST\_BA\_SunBld:
  STRUCT
    rPos        : REAL;
    rAngl       : REAL;
    bManUp      : BOOL;
    bManDwn     : BOOL;
    bManMod     : BOOL;
END_STRUCT
END_TYPE
\end{verbatim}
Up to 8 positioning telegrams from different control function blocks can be applied to this function block. The telegram on \textit{stSunBld\_Prior1} has the highest priority and that on \textit{stSunBld\_Prior8} the lowest. The active telegram with the highest priority is output at the output \textit{stSunBld}. "Active" means that the variable \textit{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. \textit{rPos}=0, \textit{rAngl}=0, \textit{bManUp}=\text{FALSE}, \textit{bManDwn}=\text{FALSE}, \textit{bManMod}=\text{FALSE}, \textit{bActv}=\text{FALSE}. Since the blind function block FB\_BA\_SunBldActr \[89\] or the roller blind function block FB\_BA\_RolBldActr \[81\] does not take account of the flag \textit{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.

\begin{verbatim}
VAR_INPUT
stSunBld\_Prior1 : ST_BA_SunBld;
stSunBld\_Prior2 : ST_BA_SunBld;
stSunBld\_Prior3 : ST_BA_SunBld;
stSunBld\_Prior4 : ST_BA_SunBld;
stSunBld\_Prior5 : ST_BA_SunBld;
stSunBld\_Prior6 : ST_BA_SunBld;
stSunBld\_Prior7 : ST_BA_SunBld;
stSunBld\_Prior8 : ST_BA_SunBld;
\end{verbatim}

\textit{stSunBld\_Prior1..stSunBld\_Prior8}: Positioning telegrams available for selection. \textit{stSunBld\_Prior1} has the highest priority and \textit{stSunBld\_Prior8} the lowest.

\begin{verbatim}
VAR_OUTPUT
stSunBld : ST_BA_SunBld;
udiActvPrio : UDINT;
\end{verbatim}

\textit{stSunBld}: Resulting positioning telegram.
\textit{udiActvPrio}: Active positioning telegram. If none is active, "0" is output.

\textbf{Requirements}

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

\textbf{5.1.3.3.23 FB\_BA\_SunBldScn}

This function block represents an extension of the manual controller FB\_BA\_SunBldSwi \[102\] by a scene memory and a call function. The blind control FB\_BA\_SunBldActr \[89\] or the roller blind control FB\_BA\_RolBldActr \[81\] can be active in manual mode and also directly target previously stored positions (scenes). Up to 21 scenes can be saved.
Structure of the blind positioning telegram ST_BA_SunBld [183].

```plaintext
TYPE ST_BA_SunBld:
  STRUCT
    rPos        : REAL;
    rAngl       : REAL;
    bManUp      : BOOL;
    bManDwn     : BOOL;
    bManMod     : BOOL;
    bActv       : BOOL;
  END_STRUCT
END_TYPE
```

Operation

In manual mode, the function block controls the blind function block FB_BA_SunBldActr [89] or the roller shutter function block FB_BA_RolBldActr [81] via the command inputs bUp and bDwn; bUp has priority. The commands are passed on to the respective commands bManUp and bManDwn of the positioning telegram. If a command input is activated that is longer than the entered time udiSwiOvrTi_ms [ms], then the corresponding control command latches. Activating a command input again releases this latch.

A rising edge on bSavScn saves the current position and louvre angle in the scene selected in udiScdScn. This procedure is possible at any time, even during active positioning. The selected scene is called with bClScn, i.e. the saved position and angle values are driven to.

If the function block is activated by input bEn=TRUE, bit bActv is set immediately in the positioning telegram. The function block uses this to notify a priority switch (FB_BA_SunBldPrioSwi4 [97] or FB_BA_SunBldPrioSwi8 [98]) of its priority over lower priorities. If the command "Call Scene" is not active (bClScn =TRUE), the bit bManMod is also set in the positioning telegram to notify the connected actuator function blocks that they should respond to manual commands.

If the function block is deactivated by bEn=FALSE, both bits, bActv and bManMod, are set to FALSE again.

Linking to the blind function block

Like the "normal" manual mode function block FB_BA_SunBldSwi [102], the scene selection function block can be connected either via an upstream priority control FB_BA_SunBldPrioSwi4 [97] or FB_BA_SunBldPrioSwi8 [98], or directly via the blind function block. The connection is established via the positioning telegram ST_BA_SunBld [183]. Furthermore the scene function block requires the current positions from the blind function block for the reference blind.

Use of a priority controller:
Direct connection:

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bUp</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bDwn</td>
<td>BOOL;</td>
</tr>
<tr>
<td>udiSwiOvrT.im</td>
<td>UDINT;</td>
</tr>
<tr>
<td>udiSlcdScn</td>
<td>UDINT;</td>
</tr>
<tr>
<td>bClScn</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bSavScn</td>
<td>BOOL;</td>
</tr>
<tr>
<td>rSpPos</td>
<td>REAL;</td>
</tr>
<tr>
<td>rSpAngl</td>
<td>REAL;</td>
</tr>
</tbody>
</table>

bEn: The function block has no function if this input is FALSE. In the positioning telegram ST_BA_SunBld [183], 0 is output for the position and the angle - bManMod and bActv are FALSE. For a connection with priority controller this means that another functionality takes over control of the blind. Conversely, a direct connection allows the blind to drive directly to the 0 position, i.e. fully up, since the actuator function block does not evaluate the bit bActv itself.

bUp: Command input for blind up.

bDwn: Command input for blind down.

udiSwiOvrT.im: Time [ms] until the corresponding manual command in the positioning telegram ST_BA_SunBld [183] switches to latching mode, if the command input is activated permanently. Internally limited to a minimum value of 0.

udiSlcdScn: Selected scene which should either be saved (bSavScn) or called (bClScn). Internally limited to a minimum value of 0 to cMaxSunBldScn.

bClScn: Call selected scene.

bSavScn: Save selected scene.

rSpPos: Set position [%] that is to be saved in the selected scene. This must be linked to the actual position of the actuator function block FB_BA_SunBldActr [89] or FB_BA_RolBlcActr [81] of the reference blind/roller shutter, in order to be able to save a position that was previously approached manually. Internally limited to values between 0 and 100.

rSpAngl: ditto. Louvre angle [°].

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>stSunBld</td>
<td>ST_BA_SunBld;</td>
</tr>
<tr>
<td>bActv</td>
<td>BOOL;</td>
</tr>
<tr>
<td>rActlScnPos</td>
<td>REAL;</td>
</tr>
<tr>
<td>rActlScnAngl</td>
<td>REAL;</td>
</tr>
</tbody>
</table>

stSunBld: Positioning telegram, see ST_BA_SunBld [183].

bActv: Corresponds to the boolean value bActv in the blind telegram ST_BA_SunBld [183] and is solely used to indicate whether the function block sends an active telegram.
**rActlScnPos**: Indicates the saved relative blind height position [%] for the currently selected scene.

**rActlScnAngl**: ditto. Louvre angle [°].

If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

**VAR_IN_OUT**

```plaintext
arrSunBldScn : ARRAY[0..Param.usiMaxSunBldScn] OF ST_BA_SunBldScn;
```

**arrSunBldScn**: Table with the scene entries of the type `ST_BA_SunBldScn` [184].

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.24 **FB_BA_SunBldSwi**

This function block can be used to control the blind `FB_BA_SunBldActr` [89] or roller shutter `FB_BA_RolBldActr` [81] in manual mode. The connection takes place via the positioning telegram `ST_BA_Sunbld` [183] either directly or with an additional priority controller.

**Structure of the blind positioning telegram** `ST_BA_Sunbld` [183].

```plaintext
TYPE ST_BA_Sunbld:
  STRUCT
    rPos        : REAL;
    rAngl       : REAL;
    bManUp      : BOOL;
    bManDwn     : BOOL;
    bManMod     : BOOL;
    bActv       : BOOL;
  END_STRUCT
END_TYPE
```

**Operation**

In manual mode, the function block controls the blind function block `FB_BA_SunBldActr` [89] or the roller shutter function block `FB_BA_RolBldActr` [81] via the command inputs `bUp` and `bDwn`; `bUp` has priority. The commands are passed on to the respective commands `bManUp` and `bManDwn` of the positioning telegram. If a command input is activated that is longer than the entered time `udiSwiOvrTi_ms` [ms], then the corresponding control command latches. Activating a command input again releases this latch. If the function block is activated by input `bEn`=TRUE, bit `bActv` is set immediately in the positioning telegram. If the function block is deactivated by `bEn`=FALSE, both bits, `bActv` and `bManMod`, are set to FALSE again.

If the function block uses this to notify a priority switch (`FB_BA_SunBldPrioSwi4` [97] or `FB_BA_SunBldPrioSwi8` [98]) of its priority over lower priorities. At the same time, the bit `bManMod` is set in the positioning telegram to notify the connected actuator function blocks that they should respond to manual commands.
Linking to the blind function block

The manual mode function block can be connected either via an upstream priority control FB_BA_SunBldPrioSwi4 [97] or FB_BA_SunBldPrioSwi8 [98], or directly at the blind function block. The connection is established via the positioning telegram ST_BA_Sunbld [183].

Use of a priority controller:

Direct connection:

VAR_INPUT

<table>
<thead>
<tr>
<th>bEn</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bUp</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>bDwn</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>udiSwiOvrTi_ms</td>
<td>: UDINT;</td>
</tr>
</tbody>
</table>

bEn: The function block has no function if this input is FALSE. In the positioning telegram ST_BA_Sunbld [183], 0 is output for the position and the angle - bManMod and bActv are FALSE. For a connection with priority controller this means that another functionality takes over control of the blind. Conversely, a direct connection allows the blind to drive directly to the 0 position, i.e. fully up, since the actuator function block does not evaluate the bit bActv itself.

bUp: Command input for blind up.

bDwn: Command input for blind down.

udiSwiOvrTi_ms: Time [ms] until the corresponding manual command in the positioning telegram ST_BA_Sunbld [183] switches to latching mode, if the command input is activated permanently. Internally limited to a minimum value of 0.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>stSunBld</th>
<th>: ST_BA_SunBld;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bActv</td>
<td>: BOOL;</td>
</tr>
</tbody>
</table>

stSunBld: Positioning telegram, see ST_BA_SunBld [183].
bActv: Corresponds to the boolean value bActv in the blind telegram ST_BA_SunBld [183] and is solely used to indicate whether the function block sends an active telegram.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.3.3.25 FB_BA_SunBldTwiLgtAuto

This function block controls the blind when the outdoor brightness has fallen below a limit value.

The automatic twilight function operates with both a value hysteresis and a temporal hysteresis: If the outdoor brightness value \( rBrtns \) [lux] falls below the value \( rActvVal \) [lux] for the time \( udiActvDly_sec \) [s], the function block is active and will provide the blind positions \( rPosTwiLgt \) (height [\%]) and \( rAnglTwiLgt \) (louvre angle [°]) specified for the input variables at the output in the positioning telegram ST_BA_Sunbld [183]. If the outdoor brightness exceeds the value \( rActvVal \) [lux] for the time \( udiDctvDly_sec \) [s], automatic mode is no longer active. The active flag in the positioning telegram ST_BA_Sunbld [183] is reset and the positions for height and angle are set to "0". A function with a lower priority can then take over control.

VAR_INPUT

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn</td>
<td>BOOL</td>
</tr>
<tr>
<td>rBrtns</td>
<td>REAL</td>
</tr>
<tr>
<td>rActvVal</td>
<td>REAL</td>
</tr>
<tr>
<td>rDctvVal</td>
<td>REAL</td>
</tr>
<tr>
<td>udiActvDly_sec</td>
<td>UDINT</td>
</tr>
<tr>
<td>udiDctvDly_sec</td>
<td>UDINT</td>
</tr>
<tr>
<td>rPosTwiLgt</td>
<td>REAL</td>
</tr>
<tr>
<td>rAnglTwiLgt</td>
<td>REAL</td>
</tr>
</tbody>
</table>

bEn: The function block has no function if this input is FALSE. In the positioning telegram ST_BA_Sunbld [183] 0 is output for the position and the angle, and bActv is FALSE. This means that another function takes over control of the blind via the priority controller.

rBrtns: Outdoor brightness [lx].

rActvVal: Activation limit value [lx]. The value rActvVal is internally limited to values from 0 to rDctvVal.

rDctvVal: Deactivation limit value [lx]. Internally limited to a minimum value of 0.

udiActvDly_sec: Activation delay [s]. Internally limited to a minimum value of 0.

udiDctvDly_sec: Deactivation delay [s]. Internally limited to a minimum value of 0.

rPosTwiLgt: Vertical position of the blind [%] if the automatic twilight function is active. Internally limited to values between 0 and 100.

rAnglTwiLgt: Louvre angle of the blind [°] if the automatic twilight function is active.
VAR_OUTPUT

stSunBld : ST_BA_SunBld;
bActv : BOOL;
udiRemTiActv_sec : UDINT;
udiRemTiDctv_sec : UDINT;

**stSunBld**: Output structure of the blind positions, see ST_BA_SunBld [183].

**bActv**: Corresponds to the boolean value bActv in the blind telegram ST_BA_SunBld [183] and is solely used to indicate whether the function block sends an active telegram.

**udiRemTiActv_sec**: Shows the time remaining [s] after falling below the switching value rActvVal until automatic mode is activated. This output is 0 as long as no countdown of the time is taking place.

**udiRemTiDctv_sec**: Shows the time remaining [s] after exceeding of the switching value rDctvVal until automatic mode is disabled. This output is 0 as long as no countdown of the time is taking place.

---

If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.1.3.26 FB_BA_SunBldWndPrtc

The function block **FB_BA_SunBldWndPrtc** deals with the direction-dependent wind protection.

The weather protection has the highest priority in the blind controller (see overview [52]) and is intended to ensure that the blind is not damaged by ice or wind.

If the measured wind speed is above the value rWndSpdStrmOn for the time udiDlyStrmOn_sec [s], it is assumed that high winds are imminent. The storm is regarded as having subsided, so that the blind can be moved safely, once the wind speed falls below the value rWndSpdStrmOff for the time udiDlyStrmOff_sec [s]. For safety reasons the storm event is also persistently saved. Thus, if the controller fails during a storm, the sequence timer is started again from the beginning when the controller is restarted.

If there is a risk of high wind, the blind is moved to the protection position specified by rPosProt (height position in percent) and rAnglProt (louvre angle [°]).

VAR_INPUT

bEn : BOOL;
rWndSpd : REAL;
rWndSpdStrmOn : REAL;
rWndSpdStrmOff : REAL;
udiDlyStrmOn_sec : UDINT;
udiDlyStrmOff_sec : UDINT;
rPosProt : REAL;
rAnglProt : REAL;
**bEn:** The function block has no function if this input is FALSE. In the positioning telegram `ST_BA_Sunbld` [183] 0 is output for the position and the angle, and `bActv` is FALSE. This means that another function takes over control of the blind via the priority controller.

**rWndSpd:** Wind speed. The unit of entry is arbitrary, but it is important that no value is smaller than 0 and that the values become larger with increasing speed.

**rWndSpdStrmOn:** Wind speed limit value for the activation of the storm alarm. This value may be not smaller than 0 and must lie above the value for the deactivation. Otherwise an error is output. The unit of entry must be the same as that of the input `rWndSpd`. A value greater than this limit value triggers the alarm after the specified time `udiDlyStrmOn_sec`.

**rWndSpdStrmOff:** Wind speed limit value for the deactivation of the storm alarm. This value may be not smaller than 0 and must lie below the value for the activation. Otherwise an error is output. The unit of entry must be the same as that of the input `rWndSpd`. A value smaller than or equal to this limit value resets the alarm after the specified time `udiDlyStrmOff_sec`.

**udiDlyStrmOn_sec:** Time delay until the storm alarm is triggered [s].

**udiDlyStrmOff_sec:** Time delay until the storm alarm is reset [s].

**rPosProt:** Height position of the blind [%] in the case of protection.

**rAnglProt:** Louvre angle of the blind [°] in the case of protection.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>stSunBld</code></td>
<td>Output structure of the blind positions, see <code>ST_BA_SunBld</code> [183].</td>
</tr>
<tr>
<td><code>bActv</code></td>
<td>Corresponds to the boolean value <code>bActv</code> in the blind telegram <code>ST_BA_SunBld</code> [183] and is solely used to indicate whether the function block sends an active telegram.</td>
</tr>
<tr>
<td><code>bStrmAlm</code></td>
<td>Indicates the storm alarm.</td>
</tr>
<tr>
<td><code>udiRemTiStrmDetc_sec</code></td>
<td>In the non-critical case, this second counter continuously shows the alarm delay time <code>udiDlyStrmOn_sec</code>. If the measured wind speed <code>rWndSpd</code> is above the activation limit value <code>rWndSpdStrmOn</code>, the seconds to the alarm are counted down. This output is 0 as long as no countdown of the time is taking place.</td>
</tr>
<tr>
<td><code>udiRemTiStrmAlm_sec</code></td>
<td>As soon as the storm alarm is initiated, this second counter initially constantly indicates the deactivation time delay of the storm alarm <code>udiDlyStrmOff_sec</code>. If the measured wind speed <code>rWndSpd</code> falls below the deactivation limit value <code>rWndSpdStrmOff</code>, the seconds to the all-clear signal (<code>bStrmAlm</code> = FALSE) are counted down. This output is 0 as long as no countdown of the time is taking place.</td>
</tr>
</tbody>
</table>

If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.3.3.27 FB_BA_SunPrtc

The function block is used for glare protection with the aid of a slatted blind.

Glare protection is realized through variation of the louvre angle and positioning of the blind height.

The louvre angle is set as a function of the sun position such that direct glare is prevented, while letting as much natural light through as possible.

Three different operating modes are available for varying the blind height.

1. When sun protection is active, the blind moves to a fixed height. The height value is specified with the variable **rFixPos**.
2. The blind position is varied as a function of the sun position. The position is specified in the table (ST_BA_BldPosTab [180]). See also description of FB_BA_BldPosEntry [59].
3. The high of the blind is calculated based on the window geometry such that the sun's rays reach a specified depth in the room. The incidence depth of the sun's rays is defined with the variable **rMaxLgtIndc**.

In order to avoid excessive repositioning of the louvre angle, the variable **udiPosIntval_min** can be used to specify a time interval, within which the louvre angle is not adjusted. In order to avoid glare, the angle is always changed sufficiently for the respective time interval.

The following conditions must be met for positioning the blind and setting the louvre angle.

- 1. The input **bEn** must be TRUE.
- 2. The sun must have risen. (elevation > 0)
- 3. The function block is parameterized correctly (bErr=FALSE)

**VAR_INPUT**

```
bEn : BOOL;
stUTC : TIMESTRUCT;
udiPosIntval_min : UDINT;
rDegLngd : REAL;
rDegLatd : REAL;
rFcdOrtn : REAL;
rFcdAngl : REAL;
rLamWdth : REAL;
rLamDsc : REAL;
rFixPos : REAL;
rMaxLgtIndc : REAL;
rWdwHght : REAL;
rDscWdwFlr : REAL;
stBldPosTab : ST_BA_BldPosTab;
ePosMod : E_BA_PosMod;
```
**bEn:** If this input is set to FALSE the positioning is inactive, i.e. the active bit \( (bActv) \) is reset in the positioning structure \( stSunBld \) of the type \( ST\ BA\ SunBld \) and the function block itself remains in a standstill mode. If on the other hand the function block is activated, then the active bit is TRUE and the function block outputs its control values \( (rPos, rAngl) \) in the positioning structure at the appropriate times.

**stUTC:** Input of current time as coordinated world time (UTC - Universal Time Coordinated, previously referred to as GMT, Greenwich Mean Time) (see TIMESTRUCT). The function block \( FB\ BA\ GetTime \) can be used to read this time from a target system.

A jump of more than 300 seconds leads to immediate repositioning, if the blind is in the sun and glare protection is active, based on the above criteria. This functionality was added to ensure a reproducible program execution.

**udiPosIntval\_min:** Positioning interval in minutes - time between two blind position outputs. Valid range: 1 min...720 min.

**rDegLnd:** Longitude [°]. Valid range: -180°...180°.

**rDegLat:** Latitude [°]. Valid range: -90°...90°.

**rFcdOrtn:** Facade orientation [°]:

In the northern hemisphere, the following applies for the facade orientation (looking out of the window):

<table>
<thead>
<tr>
<th>Line of sight</th>
<th>Facade orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>( \beta=0° )</td>
</tr>
<tr>
<td>East</td>
<td>( \beta=90° )</td>
</tr>
<tr>
<td>South</td>
<td>( \beta=180° )</td>
</tr>
<tr>
<td>West</td>
<td>( \beta=270° )</td>
</tr>
</tbody>
</table>

The following applies for the southern hemisphere:

<table>
<thead>
<tr>
<th>Line of sight</th>
<th>Facade orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>South</td>
<td>( \beta=0° )</td>
</tr>
<tr>
<td>East</td>
<td>( \beta=90° )</td>
</tr>
<tr>
<td>North</td>
<td>( \beta=180° )</td>
</tr>
<tr>
<td>West</td>
<td>( \beta=270° )</td>
</tr>
</tbody>
</table>

**rFcdAngl:** Facade inclination [°]. See facade inclination [57].

**rLamWdth:** Width of the louvres in mm, see sketch [54].

**rLamDstc:** Louvre spacing in mm, see sketch [54].

**rFixPos:** Fixed (constant) shutter height \([0..100\%]\). Applies if \( ePosMod = ePosModFix \) (see enumerator \( E\ BA\ PosMod \) [179]).

**rMaxLgtIndc:** Maximum desired light incidence in mm measured from the outside of the wall (see Sun protection: Basic principles and definitions [56]). The parameters \( rWdwHght \) and \( rDstcWdwFlr \) are used to calculate how high the blinds must be, depending on the position of the sun, such that the incidence of light does not exceed the value \( rMaxLgtIndc \). Applies if \( ePosMod = ePosModeMaxIncidence \) (see enumerator \( E\ BA\ PosMod \) [179]).

**rWdwHght:** Window height in mm for the calculation of the shutter height if the mode "maximum desired incidence of light" is selected.

**rDstcWdwFlr:** Distance between the floor and the window sill in mm for the calculation of the shutter height if the mode "maximum desired incidence of light" is selected.
**stBldPosTab**: Table of 6 interpolation points, 4 of which are parameterizable, from which a blind position is then given in relation to the position of the sun by linear interpolation. Applies if $ePosMod = ePosModFix$ (see enumerator E_BA_PosMod [179]). For a more detailed description please refer to FB_BA_BldPosEntry [59].

**ePosMod**: Selection of the positioning mode, see enumerator E_BA_PosMod [179].

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>stSunBld</th>
<th>ST_BA_SunBld;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bActv</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bErr</td>
<td>BOOL;</td>
</tr>
<tr>
<td>sErrorDescr</td>
<td>T_MAXSTRING;</td>
</tr>
</tbody>
</table>

**stSunBld**: Output structure of the blind positions, see ST_BA_SunBld [183]

**bActv**: The function block is in active state, i.e. no error is pending, the function block is enabled, and the sun position is in the specified facade area (the facade is sunlit).

**bErr**: This output is switched to TRUE if the parameters entered are erroneous.

**sErrDescr**: Contains the error description.

**Error description**

| 01: | Error: The duration of the positioning interval is less than or equal to zero, or it exceeds 720 min. |
| 02: | Error: The longitude entered is not within the valid range from -180°..180°. |
| 03: | Error: The latitude entered is not within the valid range from -90°..90°. |
| 04: | Error: The value entered for the facade inclination $rFcdAngl$ is outside the valid range of -90°..90°. |
| 05: | Error: The value for the louvre spacing ($rLamDstc$) is greater than or equal to the value for the louvre width ($rLamWdth$). This does not represent a "valid" blind, since the louvres cannot close fully. Mathematically, this would lead to errors. |
| 06: | Error: The value entered for the louvre width $rLamWdth$ is zero. |
| 07: | Error: The value entered for the louvre spacing $rLamDstc$ is zero. |
| 08: | Error: The value entered for the fixed blind height ($rFixPos$) is greater than 100 or less than 0. At the same time, "fixed blind height" positioning is selected - ePosMod=ePosModFix. |
| 09: | Error: The bit "values valid" (bVld) in the positioning table stBldPosTab is not set - invalid values: see FB_BA_BldPosEntry. At the same time, "Table" positioning is selected - ePosMod=ePosModTab. |
| 10: | Error: The value entered for the maximum required light incidence $rMaxLgtIndc$ is less than or equal to zero. At the same time, "maximum light incidence" is selected - ePosMod=ePosModMaxIndc. |
| 11: | Error: The value entered for the window height $rWdwHght$ is less than or equal to zero. At the same time, "maximum light incidence" is selected - ePosMod=ePosModMaxIndc. |
| 12: | Error: The distance between lower window edge and floor $rDstcWdwFlr$ that was entered is less than zero. At the same time, "maximum light incidence" is selected - ePosMod=ePosModMaxIndc. |
| 13: | Error: An invalid positioning mode is entered at input ePosMod. |

If an error occurs, this automatic control is deactivated, and the position and angle are set to 0. This means that if a priority controller is in use, another function with a lower priority (see Overview) automatically takes over control of the blind. In the case of a direct connection, conversely, the blind will drive to position/angle 0.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>TC3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.4 System

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_CnvtTiSt</td>
<td>Conversion year, month, day, hour, minute and second in time structure</td>
</tr>
<tr>
<td>FB_BA_ExtTiSt</td>
<td>Conversion time structure in year, month, day, hour, minute and second</td>
</tr>
<tr>
<td>FB_BA_GetTime</td>
<td>Internal clock with time information - can be synchronized with system time</td>
</tr>
<tr>
<td>FB_BA_SetTime</td>
<td>Setting the system time</td>
</tr>
</tbody>
</table>

5.1.4.1 FB_BA_CnvtTiSt

![Diagram]

The function block FB_BA_CnvtTiSt can be used to consolidate the different components of a time structure.

**Note:**

The function block does not check for incorrect entries, such as an hour entry of 99. It makes sense to check this in the connected function blocks, which have to check the time structure in any case. The limit values are shown as part of the variable explanations.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Var Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>wYear</td>
<td>WORD</td>
</tr>
<tr>
<td>wMonth</td>
<td>WORD</td>
</tr>
<tr>
<td>wDay</td>
<td>WORD</td>
</tr>
<tr>
<td>wHour</td>
<td>WORD</td>
</tr>
<tr>
<td>wMinute</td>
<td>WORD</td>
</tr>
<tr>
<td>wSecond</td>
<td>WORD</td>
</tr>
<tr>
<td>wMilliseconds</td>
<td>WORD</td>
</tr>
</tbody>
</table>

- **wYear:** The year (1970..2106).
- **wMonth:** The month (1..12).
- **wDay:** The day of the month (1..31).
- **wHour:** The hour (0..23).
- **wMinute:** The minutes (0..59).
- **wSecond:** The seconds (0..59).
- **wMilliseconds:** The milliseconds (0..999).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Var Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>stTi</td>
<td>TIMESTRUCT</td>
</tr>
</tbody>
</table>

- **stTi:** Output time structure (see TIMESTRUCT)
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.4.2 FB_BA_ExtTiSt

The function block *FB_BA_ExtTiSt* resolves a time structure into the different components, so that it can be used for time conditions, for example.

**VAR_INPUT**

```
stTi   : TIMESTRUCT;
```

**stTi**: Input time structure (see TIMESTRUCT)

**VAR_OUTPUT**

```
wYear   : WORD;
wMonth   : WORD;
wDayOfWeek: WORD;
wDay     : WORD;
wHour    : WORD;
wMinute  : WORD;
wSecond  : WORD;
wMilliseconds : WORD;
```

**wYear**: The year (1970..2106).

**wMonth**: The month (1..12).

**wDayOfWeek**: The day of the week (0(Sun)..0(Sat)).

**wDay**: The day of the month (1..31).

**wHour**: The hour (0..23).

**wMinute**: The minutes (0..59).

**wSecond**: The seconds (0..59).

**wMilliseconds**: The milliseconds (0..999).
### 5.1.4.3 FB_BA_GetTime

With this function block an internal clock (Real Time Clock RTC) can be implemented in the TwinCAT PLC. When the function block is enabled via \( b\text{En} \), the RTC clock is initialized with the current NT system time. One system cycle of the CPU is used to calculate the current RTC time. The function block must be called once per PLC cycle in order for the current time to be calculated. Within the function block, an instance of the function blocks NT_GetTime, FB_GetTimeZoneInformation and RTC_EX2 is called. The time is output at the outputs \( \text{stSysTi} \) for the read system time and \( \text{stUtc} \) for the Coordinated Universal Time (UTC). This is determined internally from the system time and the time zone. If the system time and/or the time zone was entered incorrectly, the UTC time will also be wrong.

The system time is read cyclically via the timer to be set \( \text{udiUpdRTC_sec} \); 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 \( \text{udiRemTiUpd_sec} \) indicates the seconds remaining to the next read cycle. The time structures that are output, \( \text{stSysTi} \) and \( \text{stUtc} \), can be resolved with the aid of the function block FB_BA_ExtTiSt [111] into the components day, month, hour, minute etc.

#### Notes regarding read/wait cycle

During the read cycle, the outputs \( \text{bRdySysTi} \) and \( \text{bRdyTiZolInfo} \) change to FALSE, and the enumerator \( \text{eTimeZoneId} \) shows \( 0 = \text{eTimeZoneId_Unknown} \). 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 \( \text{bErr} \) and \( \text{sErrDescr} \). The countdown output \( \text{udiRemTiUpd_sec} \) is not restarted until the wait cycle starts.

#### VAR_INPUT

- \( b\text{En} \): Enables the function block. If \( b\text{En} = \text{TRUE} \), then the RTC clock is initialized with the NT system time.
- \( s\text{NetId} \): This parameter can be used to specify the AmsNetID (see \( T\_\text{AmsNetId} \)) 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.
- \( \text{udiUpdRtc_sec} \): Time specification [s] with which the RTC clock is regularly synchronized with the NT system time. Internally this value is limited to a minimum of 5 seconds, in order to ensure correct processing of the internal function blocks.
- \( b\text{UpdRtc} \): In parallel with the time \( \text{udiUpdRtc_sec} \), the RTC clock can be synchronized via a positive edge at this input.
VAR_OUTPUT

bRdySysTi : BOOL;
bRdyTiZoInfo : BOOL;
bRdyRTC : BOOL;
udiRemTiUpd_sec : UDINT;
stSysTi : TIMESTRUCT;
stUTC : TIMESTRUCT;
dtSysTi : DT;
dtUTC : DT;
udiCurrentTime_ms : UDINT;
eTiZId : E_TimeZoneID;
bErr : BOOL;
sErrDescr : T_MAXSTRING;

bRdySysTi: The system time was read successfully from the target system.

bRdyTiZoInfo: The additional time information (time zone, time shift relative to UTC and summer/winter time) was read successfully.

bRdyRTC: This output is set if the function block has been initialized at least once. If this output is set, then the values for date, time and milliseconds at the outputs are valid.

udiRemTiUpd_sec: Countdown to next synchronization/update of the time information.

stSysTi: System time of the read target system (see TIMESTRUCT). The time structure can be resolved with the aid of the function block FB_BA_ExtTiSt into its components: day, month, hour, minute etc. Note: If the function block is not enabled (bEn=FALSE), the output stSysTi and its subelements (day month, etc.) show 0.

stUTC: Coordinated world time (see TIMESTRUCT). This is determined internally from the system time and the time information read from the target system. The time structure can be resolved with the aid of the function block FB_BA_ExtTiSt into its components: day, month, hour, minute etc. Note: If the function block is not enabled (bEn=FALSE), the output stUTC and its subelements (day month, etc.) show 0.

dtSysTi / dtUTC: As stSysTi / stUTC, but in DATE-AND-TIME format: year-month-day-hours-minutes-seconds. Note: If the function block is not enabled (bEn=FALSE), the outputs dtSysTi and dtUTC show DT#1970-01-01-00:00, since this is the lower limit, which corresponds to the zeros in the structure representation of stSysTi / stUTC.

udiCurrentTime_ms: Current time of day in ms.

eTiZId: Enumerator for summer/winter time information (see E_TimeZoneID).

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description

01: Warning: ADS error when reading the time (FB_NT_GetTime). The ADS error number is stated.

02: Warning: ADS error when reading the time zone information (FB_GetTimeZoneInformation). The ADS error number is stated.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.4.4 FB_BA_SetTime
The function block *FB_BA_SetTime* can be used to set the local NT system time and the date for a TwinCAT system (the local NT system time is shown in the taskbar). The system time is specified via the structure *stSysTi*.

Internally, an instance of the function block NT_SetLocalTime from the TcUtilities library is called in the function block.

The local NT system time can also be synchronized with a reference time with the aid of the SNTP protocol. For further information please refer to the Beckhoff Information System under: Beckhoff Information System > Embedded-PC > Operating systems > CE > SNTP: Simple Network Time Protocol.

**VAR_INPUT**

- **bSet**: Activation of the function block with a rising edge.
- **sNetId**: This parameter can be used to specify the AmsNetID of the TwinCAT computer, whose local NT system time is to be set. An empty string `sNetId := ''`; can also be specified for the local computer (see `T_AmsNetId`).
- **stSysTi**: Structure with the new local NT system time (see `TIMESTRUCT`). If the time is not available as structure, it is advisable to use the function block `FB_BA_CnvrtTiSt` [110], which brings the subvariables of date and time in a structure together.
- **udiTiOut_sec**: Indicates the timeout time [s], which must not be exceeded during execution.

**VAR_OUTPUT**

- **bBusy**: If the function block is activated via a rising edge at `bSet`, this output is set and remains set until feedback occurs.
- **bErr**: This output is set to TRUE, if either the system time to be transferred is incorrect or an ADS error occurs during the transfer.
sErrDescr: Contains the error description.

<table>
<thead>
<tr>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01: Error: Error: range exceeded year</td>
</tr>
<tr>
<td>02: Error: Error: range exceeded month</td>
</tr>
<tr>
<td>03: Error: Error: range exceeded day of the month</td>
</tr>
<tr>
<td>04: Error: Error: range exceeded hour</td>
</tr>
<tr>
<td>05: Error: Error: range exceeded minute</td>
</tr>
<tr>
<td>06: Error: Error: range exceeded second</td>
</tr>
<tr>
<td>07: Error: Error: range exceeded millisecond</td>
</tr>
<tr>
<td>08: Warning: An ADS error occurred while setting the time (FB NT_SetLocalTime). The ADS error number is stated.</td>
</tr>
</tbody>
</table>

Time specification limits

The time structure stUtcTi that was created is internally checked for limits (see TIMESTRUCT)

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.4.5 FB_BA_WrtPersistDat

The function block **FB_BA_WrtPersistDat** first saves the persistent data in the Port_xxx.bootdata file. It is not necessary to explicitly specify at which port or runtime system the PLC is located; this is determined internally. Once the data has been written, the content of the file Port_xxx.bootdata is copied to the backup file Port_xxx.bootdata-old. Thus both files are always synchronized. In case the original file with the persistent data is not readable, the backup copy, which is then read, contains the same data.

In any case, the checkmark for "Clear Invalid Persistent Data" must be removed (see **FB_BA_WrtPersistDat**).

The function block can be started in two ways:

Via a positive edge at input bStt, if the function block is not in the set start-up phase.

Initially once the start-up phase is completed after a reset or TwinCAT restart. The duration is set at udiIniSttDly_sec in seconds. If 0 is entered there, the duration of the start-up phase is 0 and an initial execution of the function block is ignored.

No commands are accepted at bStt during the start-up phase.

If errors occur while reading, writing, opening or closing the files, this is indicated with a corresponding error message at bErr/sErrDescr. After an internally fixed waiting time of two seconds, the function block automatically attempts to execute the command (read, write, open or close) again.

It is therefore advisable to keep an eye on the error outputs or to evaluate them.
It is also important to note whether the backup file for the persistent data was loaded during the TwinCAT restart or after a reset. This indicates that the original file cannot be read and that the memory card of the controller is defective. It can be queried for each runtime system with the Boolean assignment of `TwinCAT_SystemInfoVarList._AppInfo.OldBootData` (see PlcAppSystemInfo).

Sample in ST:

```st
program Example_ST

var
  bOldData : bool;
end_var

bOldData := TwinCAT_SystemInfoVarList._AppInfo.OldBootData;
```

Sample in CFC:

```cfc
program Example_CFC

var
  bOldData : bool;
end_var

TwinCAT_SystemInfoVarList._AppInfo.BootDataLoaded bOldData
```

**NOTE**

Make sure that only this function block and only one instance of it accesses the persistent data. If several function blocks open a file and do not close it again, unforeseen file handle conflicts can occur which cannot be intercepted. The persistent data will then no longer be updated in the xxx.bootdata file.

**Description of persistent data handling under TwinCAT 3**

TwinCAT saves the persistent data for each runtime system in a file during each orderly shutdown, i.e. when switching from Run to Config or Stop mode.

The file name consists of the ADS port name of the runtime system with the file extension `.bootdata`, e.g.: `Port_851.bootdata` and is stored in the TwinCAT directory under `TwinCAT\3.1\Boot\PLC`.

When the system is restarted, i.e. when switching to run mode, this file is read and then saved as `Port_xxx.bootdata-old`. If the file already exists, it is overwritten.

The original file `Port_xxx.bootdata` then no longer exists. It is created again automatically when switching to Stop mode or by the function block `FB_WritePersistentData` from the TC2_Utilities library.

This behavior applies to each runtime system; each system has its own files with persistent data.

If the file is defective when the TwinCAT system is restarted, the system automatically accesses the backup file `Port_xxx.bootdata-old`. However, this behavior only applies if the **Clear Invalid Persistent Data** checkmark is unchecked in the runtime settings. If it is checked and the original file is defective, no data will be read.
The Port_xxx.bootdata-old backup file is also used when the controller is de-energized. In this case, too, the current persistent data is not stored in Port_xxx.bootdata. When the system is restarted, only the old data is available, unless a more up-to-date file was created by the FB_WritePersistentData function block before the system was switched off.
VAR_INPUT

bStt : BOOL;
udiInitSttDly_sec : UDINT;

bStt: A rising edge at this input starts the function block if it is not in the startup phase.
udlnInitSttDly_sec: Startup phase after a reset or TwinCAT restart. The duration is set in seconds. Once the startup phase has elapsed, the function block is automatically started once. No commands are accepted at bStt during the start-up phase. If "0" is set at udlnInitSttDly_sec, the startup phase is skipped. This input is preconfigured with 10 s.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bBusy</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>udiRemTiInitSttDly_sec</td>
<td>UDINT</td>
</tr>
<tr>
<td>bErr</td>
<td>BOOL</td>
</tr>
<tr>
<td>sErrDescr</td>
<td>T_MaxString</td>
</tr>
</tbody>
</table>

bBusy: The function block is being executed.

udiRemTiInitSttDly_sec: Countdown of the set startup phase.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

Error description

01: Error: The number of the ADS port issued by the PLC is "0"

02: Warning: Error when writing the persistent data via the internal function block FB_WritePersistentData. Additionally its error number.

03: Warning: Error when opening the backup file (xxx.bootdata-old) via the internal function block FB_FileOpen. Additionally its error number.

04: Warning: Error when reading the original file (xxx.bootdata) via the internal function block FB_FileRead. Additionally its error number

05: Warning: Error when writing to the backup file (xxx.bootdata-old) via the internal function block FB_FileWrite. Additionally its error number.

06: Warning: Error when closing the original file (xxx.bootdata) via the internal function block FB_FileClose. Additionally its error number.

07: Warning: Error when closing the backup file (xxx.bootdata-old) via the internal function block FB_FileClose. Additionally its error number.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5 Universal

5.1.5.1 Actuators

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_Actuator_3Point [120]</td>
<td>Control of three-point dampers or valves</td>
</tr>
<tr>
<td>FB_BA_Anlg3Pnt [121]</td>
<td>Analog value for three-point converters</td>
</tr>
<tr>
<td>FB_BA_AntBlkg [122]</td>
<td>Blocking protection for pump or actuators</td>
</tr>
<tr>
<td>FB_BA_Motor1St [123]</td>
<td>Control of single-speed drives</td>
</tr>
<tr>
<td>FB_BA_Motor2St [124]</td>
<td>Control of two-speed drives</td>
</tr>
<tr>
<td>FB_BA_PWM [125]</td>
<td>Pulse width modulation function block</td>
</tr>
</tbody>
</table>
5.1.5.1.1 FB_BA_Actuator_3Point

The function block is used to control a 3-point actuator, e.g. a 3-point flap or a 3-point valve.

The command for opening the actuator is connected to output \textit{bOpen}.

The command for closing the actuator is connected to output \textit{bClose}.

In automatic mode (\textit{udiOpMode}=0) the control commands of \textit{bCmdOpen} and \textit{bCmdClose} are forwarded directly to the outputs \textit{bOpen} and \textit{bClose}.

The \textit{udiOpMode} input is used to determine the operating mode of the 3-point actuator:

- 0 = Automatic
- 1 = Stop \((\textit{bOpen} = \textit{bClose} = \text{FALSE})\)
- 2 = Close
- 3 = Open

\textbf{VAR_INPUT}

\begin{verbatim}
bEn : BOOL;
bAutoOpen : BOOL;
bAutoClose : BOOL;
udiOpMode : UDINT
\end{verbatim}

\textbf{bEn}: General enable of the function block.

\textbf{bAutoOpen}: Command to open the actuator.

\textbf{bAutoClose}: Command to close the actuator.

\textbf{udiOpMode}: Select operating mode \((0 = \text{Automatic}, 1 = \text{Stop (bOpen = bClose = FALSE)}, 2 = \text{Close}, 3 = \text{Open})\)

\textbf{VAR_OUTPUT}

\begin{verbatim}
bOpen : BOOL;
bClose : BOOL;
\end{verbatim}

\textbf{bOpen}: Open control output.

\textbf{bClose}: Close control output.

\textbf{Requirements}

\begin{tabular}{|l|l|l|}
\hline
\textbf{Development environment} & \textbf{required library} & \textbf{Necessary function} \\
\hline
TwinCAT3.1 4022.22 & Tc3 Building Automation from V1.1.0.0 & TF8040 | TwinCAT Building Automation from V3.1.11.0 \\
\hline
\end{tabular}
5.1.5.1.2 FB_BA_Anlg3Pnt

The function block is intended for control of three-point actuators for valves or dampers.

A continuous control signal for positioning an actuator is converted into binary commands for opening and closing.

If the deviation between the set position value \( rIn \) and the calculated actual position value \( rPos \) of the actuator exceeds the set threshold value \( rHys/2 \), the function block starts to correct the position by switching the outputs \( bOpn \) or \( bCls \), depending on the magnitude of the control deviation:

<table>
<thead>
<tr>
<th>Condition</th>
<th>( bOpn )</th>
<th>( bCls )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( rIn - rPos &gt; rHys/2 )</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>( rIn - rPos &lt; - rHys/2 )</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

If the function block reaches an end position \( rOut = 0 \) or \( rOut = 100 \) through a corresponding input value \( rIn \), the corresponding switching output remains permanently set in order to safely reach this end position at the valve or damper:

<table>
<thead>
<tr>
<th>Condition</th>
<th>( bOpn )</th>
<th>( bCls )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( rOut = 0 )</td>
<td>FALSE</td>
<td>permanently TRUE</td>
</tr>
<tr>
<td>( rOut = 100 )</td>
<td>permanently TRUE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Any deactivation of the continuous signal must be implemented by the user through external programming.

The input \( rIn \) is automatically limited to the range 0..100% internally.

This also applies to the entries \( rHys \) and \( rRefVal \). The travel times \( udiTiCls_ms \) and \( udiTiOpn_ms \) both have a lower limit value of 10 (milliseconds).

A rising edge at \( bRef \) triggers a referencing command (the calculated actual position is set to \( rRefVal \)).

If the drive has limit switches, they can be sampled directly via the digital input and used for referencing at \( bRef \).

### VAR_INPUT

- \( rIn \): Setpoint for the actuator position [0 - 100%]. Internally limited to values between 0 and 100.
- \( rHys \): Hysteresis for the actuator position [0 - 100%]. Internally limited to values between 0 and 100.
- \( udiTiCls_ms \): Run time of the actuator from open to closed [ms]. Internally limited to values between 0 and 100.
- \( udiTiOpn_ms \): Run time of the actuator from closed to open [ms]. Internally limited to values between 0 and 100.
- \( bRef \): Edge references the internal position memory of the drive to value of \( rRefVal \) [0 - 100%].
rRefVal: Value for referencing the actuator with \( bRef \) [0 - 100\%]. Internally limited to values between 0 and 100.

bCloseInit: If this input is TRUE, output \( bCls \) is TRUE for the time \( udiTiOpn_ms \)

**VAR_OUTPUT**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bCls</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bOpn</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>rPos</td>
<td>REAL</td>
<td></td>
</tr>
</tbody>
</table>

**bCls**: Output for closing the actuator.

**bOpn**: Output for opening the actuator.

**rPos**: Current calculated actuator position [0 - 100\%].

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.1.5.1.3 FB_BA_AntBlkg

This function block prevents blocking of pumps or actuators after prolonged idle periods by issuing a switch-on pulse. The maximum idle period before such a pulse is issued is determined by the value of \( udiTiOffMin_sec \). For logging the idle time, the input \( bFdb \) must be linked to the operating feedback from the unit. The length of the pulse is parameterized with \( udiTiImplLngt_sec \). The input \( bExe \) should be used if the blocking protection pulses are to be issued cyclically based on a switching schedule, rather than depending on the idle times. A rising edge at \( bExe \) immediately triggers output of a pulse to \( bQ \). Generally, a pulse output only occurs if the function block at \( bEn \) is enabled.

**VAR_INPUT**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bFdb</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bExe</td>
<td>BOOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udiTiOffMin_sec</td>
<td>UDINT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udiTiImplLngt_sec</td>
<td>UDINT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**bEn**: Enable function block.

**bFdb**: Input for connecting the feedback signal of a motor or valve.

**bExe**: Rising edge forces a pulse output.

**udiTiOffMin_sec**: Minimum switch-off time [s]: a pulse is issued once the time \( udiTiOffMin_sec \) has elapsed without movement of the unit.

**udiTiImplLngt_sec**: Length of the blocking protection pulse [s] at \( bQ \).

**VAR_OUTPUT**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bQ</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>udiRTiOffMin_sec</td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>udiRTiImplLngt_sec</td>
<td>UDINT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**bQ**: Pulse output.
udiRTiOffMin_sec: Remaining time [s] before the next pulse is issued in the absence of movement.
udiTiImplLngt_sec: Remaining residual time [s] of the pulse at bQ.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5.1.4 FB_BA_Motor1St

Function block for controlling a simple single-stage motor.

The input bEn is used for general enabling of the motor.

The input udiOpMode is used to set the operating mode of the motor:

- 0 = Automatic
- 1 = Manual off
- 2 = Manual on

In automatic mode (udiOpMode = 0) the motor can be operated via the input bAuto (bAuto = bQ = TRUE).

The collection of all possible malfunctions of a motor is connected to bDst.

VAR_INPUT

bEn : BOOL;
bAuto : BOOL;
bDst : BOOL;
udiOpMode : UDINT;

bEn: Enable motor.
bAuto: Request of the actuator in automatic mode (udiOpMode = 0).
bDst: Input for collecting the possible motor malfunctions.
udiOpMode: Select the operating mode (0 = Automatic, 1 = Manual off, 2 = Manual on).

VAR_OUTPUT

bQ : BOOL;
bQ: Control output.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.1.5 FB_BA_Motor2St

Function block for controlling a simple two-stage motor.

The input bEn is used for general enabling of the motor.

The input udiOpMode is used to set the operating mode of the motor:

- 0 = Automatic
- 1 = Manual off
- 2 = Manual stage 1
- 3 = Manual stage 2

In automatic mode (udiOpMode = 0) the desired stage can be set via the inputs bAutoSt1 (stage 1) and bAutoSt2 (stage 2).

The collection of all possible malfunctions of a motor is connected to bDst.

**VAR_INPUT**

- bEn : BOOL;
- bAutoSt1 : BOOL;
- bAutoSt2 : BOOL;
- bDst : BOOL;
- udiOpMode : UDINT;

bEn: Enable motor.

bAutoSt1: Request of the actuator at stage 1 in automatic mode (udiOpMode = 0).

bAutoSt2: Request of the actuator at stage 2 in automatic mode (udiOpMode = 0).

bDst: Input for collecting the possible motor malfunctions.

udiOpMode: Select the operating mode (0 = Automatic, 1 = Manual off, 2 = Manual stage 1, 3 = Manual stage 2).

**VAR_OUTPUT**

- bQ1 : BOOL;
- bQ2 : BOOL;

bQ1: Control output stage 1.

bQ2: Control output stage 2.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.1.6 FB_BA_PWM

The function block calculates switch-on and switch-off times $rActTiOn\_sec$ and $rActTiOff\_sec$ [s] from an analog input signal $rIn$ (0..100%, internally limited) and the period $udiPrd\_sec$ [s].

The following relationships apply:

- 100% at the input of a switch-on time $rActTiOn\_sec$ of the total period $udiPrd\_sec$ and a switch-off time $rActTiOff\_sec$ of 0 s
- 0% at the input of a switch-on time $rActTiOn\_sec$ of 0 s and a switch-off time $rActTiOff\_sec$ of the total period $udiPrd\_sec$.

In addition, $udiMinSwiTi\_sec$ [s] can be used to set a lower limit for the switching time, in order to prevent damage to drives caused by too short actuating pulses. This behavior is only valid for $0>rIn>100$!

If $rIn=0$ or 100, the output $bQ$ remains deleted or set. After the period time has elapsed, the current input signal is evaluated again. If it is still set to 0 or 100, there is no change of state of $bQ$.

Switching characteristics

1. A FALSE signal at input $bEn$ disables the function block and sets $bQ$ to FALSE. Only the switch-on and switch-off times are continuously calculated and displayed at the outputs $rActTiOn\_sec/rActTiOff\_sec$ [s].
2. A rising edge at input $bEn$ enables the function block: It will initially jump to a decision step. Depending on the previous state of the switching output $bQ$, the switching step is now accessed. However, if the input $rIn$ is set to 0, an immediate jump occurs to the Off step ($bQ=False$), or to the On step if $rIn=100$ ($bQ=True$), irrespective of the previous state of $bQ$. The minimum switching time is deactivated for these two cases.
3. A countdown timer with the current calculated starting value runs in the respective active step (ON or OFF), which is based on the pulse/pause ratio. The on- or off-step is completed with the calculated time, irrespective of whether the pulse/pause ratio changes in the meantime. The respective countdown is displayed at the outputs $udiRemTiOn\_sec/udiRemTiOff\_sec$ in full seconds.
4. Completion of the on- or off-step is followed by a jump back to the decision step (point 2).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bEn</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>rIn</td>
<td>REAL</td>
</tr>
<tr>
<td>$udiPrd_sec$</td>
<td>UDINT</td>
</tr>
<tr>
<td>$udiMinSwiTi_sec$</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

- **bEn**: Activation of pulse width modulation.
- **rIn**: Input signal, internally limited to 0..100%.
- **$udiPrd\_sec$**: Period time [s]. Internally limited to a minimum value of 0.
- **$udiMinSwiTi\_sec$**: Minimum switch-on time [s], to avoid too short pulses. Internally limited to values between 0 and $udiPrd\_sec$.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bQ</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>bLmtSwiTi</td>
<td>BOOL</td>
</tr>
<tr>
<td>$rActTiOn_sec$</td>
<td>REAL</td>
</tr>
<tr>
<td>$rActTiOff_sec$</td>
<td>REAL</td>
</tr>
<tr>
<td>$udiRemTiOn_sec$</td>
<td>UDINT</td>
</tr>
<tr>
<td>$udiRemTiOff_sec$</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

TC3 Building Automation  Version: 1.0  125
bQ: PWM output.

bLmtSwiT: Information output to indicate that the input signal is so low that the minimum switch-on time is used as limit.

rActTiOn_sec: Information output: Calculated switch-on time.

rActTiOff_sec: Information output: Calculated switch-off time.

udiRemTiOn_sec: Switch-on timer countdown.

udiRemTiOff_sec: Switch-off timer countdown.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5.2 Analog inputs/outputs

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_AI [126]</td>
<td>Acquisition of analog input signals</td>
</tr>
<tr>
<td>FB_BA_AO [129]</td>
<td>Control of analog actuators with integrated scaling function</td>
</tr>
<tr>
<td>FB_BA_KL32xxConfig</td>
<td>Parameterization of the connected sensor type on an input channel from the PLC</td>
</tr>
</tbody>
</table>

5.1.5.2.1 FB_BA_AI

The function block is used for measured data processing and terminal configuration of all standard K-bus analog input terminals.
Terminal configuration

The first step when using this function block is to select the corresponding terminal type eTerminal. Correct functioning of the function block is only guaranteed if the terminal selected with the eTerminal variable matches the terminal actually inserted and linked.

With the terminals for resistance temperature measurement of type KL3208_0010 and KL320x_0000, the temperature sensor used at the terminal input is additionally selected with the enumeration eSensor. A rising edge at bConfigure writes the terminal settings to the terminal using the variables TO_usiCtrl and TO_iDataOut.

Measured value scaling

At the analog input terminals of types: KL300x, KL306x, KL3132_0000, KL3162_0000, KL3172_0000, KL3172_0500, KL3172_1000, KL3182_0000, KL3404, KL3464, KL3408, KL3468, the raw value TI_iDataIn of the terminal is scaled by the internal function block FB_BA_Chrc02 [163]. (See diagram). The parameterization of the scaling function is carried out using parameters X(01/02) and Y(01/02). A signal offset that may be required for measured value correction can be parameterized with the variable rOffset. rSmoothingFactor attenuates the scaled measurement signal, including the offset.

Signal flow:

[Diagram showing signal flow with symbols for TI_iDataIn, Scaling, Offset, Damping, TO_iDataOut]

Scaling:

[Graph showing rY02 vs rX01 with a curve defined by rY01, rX01, and rX02]
With the temperature measurement terminals of types KL3208_0010 and KL320x_0000, the measuring signal is not scaled within the function block, but directly in the terminal. Attenuation of excessively fluctuating measuring signals takes place with the factor $r_{SmoothingFactor}$. If an error is detected at the terminal, such as $b_{ShortCircuit}$ or $b_{WireBrake}$, the last valid value is output until the error is eliminated at the output of the function block.

Signal flow:

![Signal flow diagram]

**Measuring range monitoring**

The variables $r_{HighLimit}$ and $r_{LowLimit}$ are used to monitor the measured value $r_{Val}$ at the output of the function block for compliance with a valid range. If the measured value $r_{Val}$ is above $r_{HighLimit}$ or below $r_{LowLimit}$, this is displayed at the outputs $b_{HighLimit}$ or $b_{LowLimit}$. Measuring range monitoring is deactivated if the input variable $b_{EnLimitCtrl}$ is FALSE or if an error (e.g., short circuit) is detected by means of the terminal status $TI_{usiState}$.

The response of the measuring range monitoring can be delayed by the variable $udi_{DelayLimitCtrl}_{sec}$.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI_usiState</td>
<td>USINT</td>
</tr>
<tr>
<td>TI_iDataIn</td>
<td>INT</td>
</tr>
<tr>
<td>eTerminal</td>
<td>E_BA_TERMINAL_KL</td>
</tr>
<tr>
<td>eSensor</td>
<td>E_BA_SENSOR</td>
</tr>
<tr>
<td>bConfigure</td>
<td>BOOL</td>
</tr>
<tr>
<td>rX01</td>
<td>REAL</td>
</tr>
<tr>
<td>rX02</td>
<td>REAL</td>
</tr>
<tr>
<td>rY01</td>
<td>REAL</td>
</tr>
<tr>
<td>rY02</td>
<td>REAL</td>
</tr>
<tr>
<td>rOffset</td>
<td>REAL</td>
</tr>
<tr>
<td>rSmoothingFactor</td>
<td>REAL</td>
</tr>
<tr>
<td>udiDecimalPlace</td>
<td>UDINT(1..6)</td>
</tr>
<tr>
<td>bEnLimitCtrl</td>
<td>BOOL</td>
</tr>
<tr>
<td>rHighLimit</td>
<td>REAL</td>
</tr>
<tr>
<td>rLowLimit</td>
<td>REAL</td>
</tr>
<tr>
<td>udiDelayLimitCtrl_sec</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

- **TI_usiState**: Linking with the corresponding status byte of the Bus Terminal in the I/O area of the program.
- **TI_iDataIn**: Linking with the corresponding raw data (Data In) of the Bus Terminal in the I/O area of the program (0..32767).
- **eTerminal**: Selection of the respective Bus Terminal (see E_BA_Terminal_KL).
- **eSensor**: Selection of the sensor type (see E_BA_Sensor).
- **bConfigure**: A rising edge starts the configuration of the Bus Terminal.
- **rX01**: x-value for the interpolation point P1.
- **rX02**: x-value for the interpolation point P2.
- **rY01**: y-value for the interpolation point P1.
- **rY02**: y-value for the interpolation point P2.
- **rOffset**: Offset.
- **rSmoothingFactor**: attenuation factor. Internally limited to between 1 and 10000.
- **udiDecimalPlace**: Specifies the decimal places for the value $r_{Val}$. Preset to 1.
- **bEnLimitCtrl**: Enable limit value monitoring.
**rHighLimit**: Upper limit value.

**rLowLimit**: Lower limit value.

**udiDelayLimitCtrl_sec**: Time delay until limit value monitoring is activated.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO_usiCtrl</td>
<td>Linking with the corresponding control byte of the Bus Terminal in the I/O area of the program.</td>
</tr>
<tr>
<td>TO_iDataOut</td>
<td>Linking with the corresponding raw data (Data Out) of the Bus Terminal in the I/O area of the program.</td>
</tr>
<tr>
<td>rVal</td>
<td>Scaled output value.</td>
</tr>
<tr>
<td>bWireBreak</td>
<td>Broken wire at the sensor.</td>
</tr>
<tr>
<td>bShortCircuit</td>
<td>Short-circuit at the sensor.</td>
</tr>
<tr>
<td>bHighLimit</td>
<td>Upper limit value exceeded.</td>
</tr>
<tr>
<td>bLowLimit</td>
<td>Value below lower limit.</td>
</tr>
<tr>
<td>bErr</td>
<td>This output is switched to TRUE if the parameters entered are erroneous.</td>
</tr>
<tr>
<td>sErrDescr</td>
<td>Contains the error description.</td>
</tr>
</tbody>
</table>

### Error description

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Error: Incorrect scaling parameter rX01/rX02/rY01/rY02</td>
</tr>
<tr>
<td>02</td>
<td>Error: Check the terminal configuration KL32xx eTerminal/eSensor/TO_usiCtrl/IO_iDataOut</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.5.2.2 FB_BA_AO**

The function block **FB_BA_AO** is used for output and scaling of an analog output. The input value at \( rX \) is converted into an output value from 0 to 32767 or 0-10 Volt or 4-20 mA by means of a linear equation.
The operating mode is set via the input `udiOpMod`:

- 0 = Automatic
- 1 = Manual

In automatic mode (`udiOpMod= 0`) the input value `rX` is passed on.

In manual mode (`udiOpMod= 1`) the input value `rXManual` is passed on.

**VAR_INPUT**

```plaintext
rX : REAL;
rX01 : REAL;
rX02 : REAL;
rY01 : REAL;
rY02 : REAL;
udiOpMod : UDINT(0..1);
rXManual : REAL;
```

- `rX`: Input value of the process in automatic mode.
- `rX01`: x-value for the interpolation point P1.
- `rX02`: x-value for the interpolation point P2.
- `rY01`: y-value for the interpolation point P1.
- `rY02`: y-value for the interpolation point P2.
- `udiOpMod`: Selection of the operating mode (0 = Automatic, 1 = Manual).
- `rXManual`: Input value for manual operation.

**VAR_OUTPUT**

```plaintext
rY : REAL;
TO_iY : INT;
```
rY: Output signal as floating point number.

TO_iY: Output signal as integer value for linking to the output value of the bus terminal in the I/O section of the program.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5.3 General control functions

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_DMUX XX</td>
<td>Demultiplexer function blocks</td>
</tr>
<tr>
<td>FB_BA_MMUX XX</td>
<td>The function blocks activate an input value on the output, depending on a selector and the corresponding input selector condition</td>
</tr>
<tr>
<td>FB_BA_MUX XX</td>
<td>Multiplexer function blocks</td>
</tr>
<tr>
<td>FB_BA_PrioSwi XX</td>
<td>Priority switch</td>
</tr>
<tr>
<td>FB_BA_Blink</td>
<td>Simple oscillator function block</td>
</tr>
<tr>
<td>FB_BA_FIFO04</td>
<td>Sequential control of up to four units</td>
</tr>
<tr>
<td>FB_BA_FIFO08</td>
<td>Sequential control of up to eight units</td>
</tr>
<tr>
<td>FB_BA_StepCtrl08</td>
<td>Step sequence function block, 8 steps</td>
</tr>
<tr>
<td>FB_BA_StepCtrl12</td>
<td>Step sequence function block, 12 steps</td>
</tr>
</tbody>
</table>

5.1.5.3.1 FB_BA_DMUX_XX

Demultiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different output parameters (4, 8, 12 and 16), but they all have the same functionality.
The function block FB_BA_DMUX_LR16 is described as an example.
In active state \((b\text{En}=\text{TRUE})\), the function block outputs the value at input \(lrIn\) at the output \((lrQ01..lrQ16)\) whose number is entered at input \(udi\text{Sel}\). All other outputs are set to 0 (for boolean demultiplexers to FALSE).

Example:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b\text{En} = \text{TRUE})</td>
<td>(lrQ01 = 0.0)</td>
</tr>
<tr>
<td>(udi\text{Sel} = 5)</td>
<td>(lrQ02 = 0.0)</td>
</tr>
<tr>
<td>(lrIn = 32.5)</td>
<td>(lrQ03 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ04 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ05 = 32.5)</td>
</tr>
<tr>
<td></td>
<td>(lrQ06 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ07 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ08 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ09 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ10 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ11 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ12 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ13 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ14 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ15 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ16 = 0.0)</td>
</tr>
</tbody>
</table>

If the value entered at \(udi\text{Sel}\) is greater than the number of outputs, the value of \(lrIn\) is output at the "highest" output:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b\text{En} = \text{TRUE})</td>
<td>(lrQ01 = 0.0)</td>
</tr>
<tr>
<td>(udi\text{Sel} = 25)</td>
<td>(lrQ02 = 0.0)</td>
</tr>
<tr>
<td>(lrIn = 32.5)</td>
<td>(lrQ03 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ04 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ05 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ06 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ07 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ08 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ09 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ10 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ11 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ12 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ13 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ14 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ15 = 0.0)</td>
</tr>
<tr>
<td></td>
<td>(lrQ16 = 32.5)</td>
</tr>
</tbody>
</table>

If \(b\text{En} = \text{FALSE}\), 0.0 is output at all outputs, or FALSE for boolean demultiplexers.

**VAR_INPUT**

- \(b\text{En} : \text{BOOL};\)
- \(udi\text{Sel} : \text{UDINT};\)
- \(r\text{In} : \text{LREAL};\)

\(b\text{En}:\) 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**

<table>
<thead>
<tr>
<th>lrQ00</th>
<th>LREAL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>lrQ01</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ02</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ03</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ04</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ05</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ06</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ07</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ08</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ09</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ10</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ11</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ12</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ13</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ14</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ15</td>
<td>LREAL;</td>
</tr>
<tr>
<td>lrQ16</td>
<td>LREAL;</td>
</tr>
</tbody>
</table>

lrQ00...lrQ16: Value outputs.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
The function block activates an input value on the output, depending on a selector and the corresponding input selector condition.

Multiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different input parameters (4, 8, 12, 16 and 24), but they all have the same functionality.

The function block FB_BA_MMUX_R16 is described as an example.

In active state \( (bEn=TRUE) \), the function block activates one of the input values \( rValxx \) at output \( rVal \), depending on a selector \( udiSel \) and the corresponding input selector condition \( udiEnxx \).

If several input selector conditions \( udiEn01 \ldots udiEn16 \) are identical and the selector \( udiSel \) matches a condition, the input value \( rVal01 \ldots rVal16 \) of the lowest active selector condition is activated at output \( rVal \). \( udiEn01 \) is the lowest selector condition, \( udiEn16 \) the highest.

The output variable \( bQ \) indicates that the selector \( udiSel \) matches the input selector condition \( udiEnxx \).

The output variable \( udiActvPrio \) indicates the active selector condition.
If no selector condition is active, \( rReplVal \) is output at \( rVal \). \( bQ \) is then FALSE and \( udiActvPrio \) shows 255.

**Example:**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Value</td>
</tr>
<tr>
<td>bEn</td>
<td>TRUE</td>
</tr>
<tr>
<td>udiSel</td>
<td>5</td>
</tr>
<tr>
<td>udiEn01</td>
<td>4</td>
</tr>
<tr>
<td>rVal01</td>
<td>123</td>
</tr>
<tr>
<td>udiEn02</td>
<td>8</td>
</tr>
<tr>
<td>rVal02</td>
<td>345</td>
</tr>
<tr>
<td>udiEn03</td>
<td>3</td>
</tr>
<tr>
<td>rVal03</td>
<td>321</td>
</tr>
<tr>
<td>udiEn04</td>
<td>3</td>
</tr>
<tr>
<td>rVal04</td>
<td>1.123</td>
</tr>
<tr>
<td>udiEn05</td>
<td>8</td>
</tr>
<tr>
<td>rVal05</td>
<td>5</td>
</tr>
<tr>
<td>udiEn06</td>
<td>5</td>
</tr>
<tr>
<td>rVal06</td>
<td>1.123</td>
</tr>
<tr>
<td>udiEn07</td>
<td>5</td>
</tr>
<tr>
<td>rVal07</td>
<td>345</td>
</tr>
<tr>
<td>udiEn08</td>
<td>3</td>
</tr>
<tr>
<td>rVal08</td>
<td>321</td>
</tr>
<tr>
<td>udiEn09</td>
<td>3</td>
</tr>
<tr>
<td>rVal09</td>
<td>5</td>
</tr>
<tr>
<td>udiEn10</td>
<td>3</td>
</tr>
<tr>
<td>rVal10</td>
<td>5.4321</td>
</tr>
<tr>
<td>udiEn11</td>
<td>3</td>
</tr>
<tr>
<td>rVal11</td>
<td>3.123</td>
</tr>
<tr>
<td>udiEn12</td>
<td>3</td>
</tr>
<tr>
<td>rVal12</td>
<td>3.456</td>
</tr>
<tr>
<td>udiEn13</td>
<td>3</td>
</tr>
<tr>
<td>rVal13</td>
<td>5.4321</td>
</tr>
<tr>
<td>udiEn14</td>
<td>3</td>
</tr>
<tr>
<td>rVal14</td>
<td>5.4321</td>
</tr>
<tr>
<td>udiEn15</td>
<td>3</td>
</tr>
<tr>
<td>rVal15</td>
<td>5.4321</td>
</tr>
<tr>
<td>udiEn16</td>
<td>3</td>
</tr>
<tr>
<td>rVal16</td>
<td>5.4321</td>
</tr>
<tr>
<td>rReplVal</td>
<td>5.4321</td>
</tr>
</tbody>
</table>

If no priority is active, the value of the global constant \( \text{Const.udiNoActvPrio} \) [186] is output at \( udiActvPrio \).
VAR_INPUT
rVal01 : REAL;
udiEn02 : UDINT := Const.udiNoActvPrio;
rVal02 : REAL;
udiEn03 : UDINT := Const.udiNoActvPrio;
rVal03 : REAL;
udiEn04 : UDINT := Const.udiNoActvPrio;
rVal04 : REAL;
udiEn05 : UDINT := Const.udiNoActvPrio;
rVal05 : REAL;
udiEn06 : UDINT := Const.udiNoActvPrio;
rVal06 : REAL;
udiEn07 : UDINT := Const.udiNoActvPrio;
rVal07 : REAL;
udiEn08 : UDINT := Const.udiNoActvPrio;
rVal08 : REAL;
udiEn09 : UDINT := Const.udiNoActvPrio;
rVal09 : REAL;
udiEn10 : UDINT := Const.udiNoActvPrio;
rVal10 : REAL;
udiEn11 : UDINT := Const.udiNoActvPrio;
rVal11 : REAL;
udiEn12 : UDINT := Const.udiNoActvPrio;
rVal12 : REAL;
udiEn13 : UDINT := Const.udiNoActvPrio;
rVal13 : REAL;
udiEn14 : UDINT := Const.udiNoActvPrio;
rVal14 : REAL;
udiEn15 : UDINT := Const.udiNoActvPrio;
rVal15 : REAL;
udiEn16 : UDINT := Const.udiNoActvPrio;
rVal16 : REAL;
rReplVal : REAL;

bEn: Activation of the block function.

udiSel: Selector. Internally limited to values between 0 and 4294967294.

udiEn01..udiEn16: Input selector condition. The input variables are pre-initialized to the value 255.

rVal01...rVal16: Input values to select from.

rReplVal: Substitute value, if no input selector condition is active

VAR_OUTPUT
bQ : BOOL;
rVal : REAL;
udiActvPrio : UDINT;

bQ: TRUE, if the selector udiSel matches an input selector condition udiEnxx.

rVal: Value of the selected input selector condition.

udiActvPrio: Indicates which input selector condition is active. If no priority is active, the value of the global constant Const.udiNoActvPrio [186] is output at udiActvPrio.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
Multiplexer function blocks exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different input parameters (4, 8, 12 and 16), but they all have the same functionality.

The function block FB_BA_MUX_LR16 is described as an example.

In active state (bEn=TRUE), the function block outputs the input value (lrIn01..lrIn16) at output lrQ, whose number is entered at input udiSel.

Example:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn = TRUE</td>
<td>lrQ = 16.5</td>
</tr>
<tr>
<td>udiSel = 5</td>
<td></td>
</tr>
<tr>
<td>lrIn01 = 15.9</td>
<td></td>
</tr>
<tr>
<td>lrIn02 = 32.5</td>
<td></td>
</tr>
<tr>
<td>lrIn03 = 17.4</td>
<td></td>
</tr>
<tr>
<td>lrIn04 = 5.84</td>
<td></td>
</tr>
<tr>
<td>lrIn05 = 9.56</td>
<td></td>
</tr>
<tr>
<td>lrIn06 = 16.5</td>
<td></td>
</tr>
<tr>
<td>lrIn07 = 32.781</td>
<td></td>
</tr>
<tr>
<td>lrIn08 = 25.4</td>
<td></td>
</tr>
<tr>
<td>lrIn09 = 44.5</td>
<td></td>
</tr>
<tr>
<td>lrIn10 = 66.1</td>
<td></td>
</tr>
<tr>
<td>lrIn11 = 45.5</td>
<td></td>
</tr>
<tr>
<td>lrIn12 = 83.3</td>
<td></td>
</tr>
<tr>
<td>lrIn13 = 54.56</td>
<td></td>
</tr>
<tr>
<td>lrIn14 = 33.8</td>
<td></td>
</tr>
<tr>
<td>lrIn15 = 98.5</td>
<td></td>
</tr>
<tr>
<td>lrIn16 = 71.3</td>
<td></td>
</tr>
</tbody>
</table>

If the value entered at udiSel is greater than the number of inputs, the "highest-ranking" input is output at lrQ:
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn = TRUE</td>
<td>lrQ = 2.3</td>
</tr>
<tr>
<td>udiSel = 25</td>
<td></td>
</tr>
<tr>
<td>lrIn01 = 15.9</td>
<td></td>
</tr>
<tr>
<td>lrIn02 = 32.5</td>
<td></td>
</tr>
<tr>
<td>lrIn03 = 17.4</td>
<td></td>
</tr>
<tr>
<td>lrIn04 = 5.84</td>
<td></td>
</tr>
<tr>
<td>lrIn05 = 9.56</td>
<td></td>
</tr>
<tr>
<td>lrIn06 = 16.5</td>
<td></td>
</tr>
<tr>
<td>lrIn07 = 32.781</td>
<td></td>
</tr>
<tr>
<td>lrIn08 = 25.4</td>
<td></td>
</tr>
<tr>
<td>lrIn09 = 44.5</td>
<td></td>
</tr>
<tr>
<td>lrIn10 = 66.1</td>
<td></td>
</tr>
<tr>
<td>lrIn11 = 45.5</td>
<td></td>
</tr>
<tr>
<td>lrIn12 = 83.3</td>
<td></td>
</tr>
<tr>
<td>lrIn13 = 54.56</td>
<td></td>
</tr>
<tr>
<td>lrIn14 = 33.8</td>
<td></td>
</tr>
<tr>
<td>lrIn15 = 98.5</td>
<td></td>
</tr>
<tr>
<td>lrIn16 = 71.3</td>
<td></td>
</tr>
</tbody>
</table>

If \( bEn=FALSE \), 0.0 is output at \( lrQ \), or FALSE for boolean multiplexers.

**VAR_INPUT**

| bEn : BOOL; |
| udiSel : UDINT; |
| lrIn00 : LREAL; |
| lrIn01 : LREAL; |
| lrIn02 : LREAL; |
| lrIn03 : LREAL; |
| lrIn04 : LREAL; |
| lrIn05 : LREAL; |
| lrIn06 : LREAL; |
| lrIn07 : LREAL; |
| lrIn08 : LREAL; |
| lrIn09 : LREAL; |
| lrIn10 : LREAL; |
| lrIn11 : LREAL; |
| lrIn12 : LREAL; |
| lrIn13 : LREAL; |
| lrIn14 : LREAL; |
| lrIn15 : LREAL; |
| lrIn16 : LREAL; |

**bEn**: Activation of the block function.

**udiSel**: Number of the input, whose value is to be output at \( lrQ \).

**lr00...lr16**: Input values to select from.

**VAR_OUTPUT**

| lrQ : LREAL; |

**lrQ**: Value of the selected input.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.3.4 FB_BA_PrioSwi_XX

The priority switches exist for different variable types (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT and DINT) and in different output sizes (4, 8, 12 and 16 or 24), but they all have the same functionality. The function block FB_BA_PrioSwi_LR08 is described as an example.

Priority switches are available for selecting different values. At output lrVal the value with the highest priority is applied whose input bEnxx is TRUE.

Example:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn01</td>
<td>bQ</td>
</tr>
<tr>
<td>lrVal01</td>
<td>lrVal</td>
</tr>
<tr>
<td>lrVal02</td>
<td>udiActvPrio</td>
</tr>
<tr>
<td>lrVal03</td>
<td></td>
</tr>
<tr>
<td>lrVal04</td>
<td></td>
</tr>
<tr>
<td>lrVal05</td>
<td></td>
</tr>
<tr>
<td>lrVal06</td>
<td></td>
</tr>
<tr>
<td>lrVal07</td>
<td></td>
</tr>
<tr>
<td>lrVal08</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bEn01</th>
<th>FALSE</th>
<th>lrVal01</th>
<th>32.5</th>
<th>lrVal</th>
<th>0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>lrVal02</td>
<td>bEn02</td>
<td>FALSE</td>
<td>17.4</td>
<td>lrVal03</td>
<td>0.0</td>
</tr>
<tr>
<td>lrVal04</td>
<td>bEn03</td>
<td>FALSE</td>
<td>5.84</td>
<td>lrVal05</td>
<td>0.0</td>
</tr>
<tr>
<td>lrVal06</td>
<td>bEn04</td>
<td>FALSE</td>
<td>9.56</td>
<td>lrVal07</td>
<td>0.0</td>
</tr>
<tr>
<td>lrVal08</td>
<td>bEn05</td>
<td>FALSE</td>
<td>16.5</td>
<td>lrVal09</td>
<td>0.0</td>
</tr>
<tr>
<td>lrVal10</td>
<td>bEn06</td>
<td>FALSE</td>
<td>32.781</td>
<td>lrVal11</td>
<td>0.0</td>
</tr>
<tr>
<td>lrVal12</td>
<td>bEn07</td>
<td>FALSE</td>
<td>25.4</td>
<td>lrVal13</td>
<td>0.0</td>
</tr>
<tr>
<td>lrVal14</td>
<td>bEn08</td>
<td>FALSE</td>
<td>44.5</td>
<td>lrVal15</td>
<td>0.0</td>
</tr>
</tbody>
</table>

If no priority is active, the value of the global constant ConstudiNoActvPrio is output at udiActvPrio.

### VAR_INPUT

```
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;
```

### VAR_OUTPUT

```plaintext
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

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.3.5 FB_BA_Blink

This function block is an oscillator with adjustable pulse and pause time, \( udiTiOn\_ms \) and \( udiTiOff\_ms \) [ms]. It is enabled with a TRUE signal at \( bEn \) and starts with the pulse phase.

\[ udiTiNextSwi\_sec \] is a countdown [s] to the next change of \( bQ \).

**VAR_INPUT**

- \( bEn \): Function block enable.
- \( udiTiOn\_ms \): pulse time [ms].
- \( udiTiOff\_ms \): pause time [ms].

**VAR_OUTPUT**

- \( bQ \): Oscillator output.
- \( udiTiNextSwi\_sec \): Countdown to next change of \( bQ \) [s].

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
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 [143] 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 that are enabled at inputs *bEn01..bEn04*. *udiNum* indicates the
number of requested units.

The operating hours of the units are entered at inputs *udiActvTi01_h* to *udiActvTi04_h*. If all these inputs are
set to a constant value of zero, the sequence change is controlled cyclically, depending on *bChg*.

The first unit is removed from the FIFO, the other units are advanced, and the first unit is appended at the
end of the FIFO again. As a result is an alternating sequence of units.

If more units are requested at input *udiNum* than are available at inputs *bEn01 to bEn04*, this is indicated
with TRUE at *bErr*.

**Error handling**

If more units are requested at input *udiNum* than are available at inputs *bEn01 to bEn04*, this is indicated
with TRUE at *bErr*.

**VAR_INPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bEn</td>
<td>BOOL</td>
</tr>
<tr>
<td>udiNum</td>
<td>UDINT</td>
</tr>
<tr>
<td>bChg</td>
<td>BOOL</td>
</tr>
<tr>
<td>bEn01</td>
<td>BOOL</td>
</tr>
<tr>
<td>bEn02</td>
<td>BOOL</td>
</tr>
<tr>
<td>bEn03</td>
<td>BOOL</td>
</tr>
<tr>
<td>bEn04</td>
<td>BOOL</td>
</tr>
<tr>
<td>udiActvTi01_h</td>
<td>UDINT</td>
</tr>
<tr>
<td>udiActvTi02_h</td>
<td>UDINT</td>
</tr>
<tr>
<td>udiActvTi03_h</td>
<td>UDINT</td>
</tr>
<tr>
<td>udiActvTi04_h</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

- **bEn**: Enables the function block.
- **udiNum**: Number of units.
- **bChg**: Force sequence change.
- **bEn01..bEn04**: Enable unit 1...enable unit 4.
- **udiActvTi01_h..udiActvTi04_h**: Operating hours unit 1...operating hours unit 4.

**VAR_OUTPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bQ01</td>
<td>BOOL</td>
</tr>
<tr>
<td>bQ02</td>
<td>BOOL</td>
</tr>
<tr>
<td>bQ03</td>
<td>BOOL</td>
</tr>
<tr>
<td>bQ04</td>
<td>BOOL</td>
</tr>
<tr>
<td>udiNextOn</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

- **bQ01**: bQ02...bQ04: Next on unit 1...next on unit 4.
TC3 Building Automation

5.1.5.3.7 FB_BA_FIFO08

The function block FB_BA_FIFO08 enables sequential control of up to eight units, with automatic switching of the switch-on sequence based on operating hours.

The function block is available in two versions: for a sequence of four or eight units.

Units with fewer operating hours take precedence in the sequence over units with more operating hours.
A rising edge at $b\text{Chg}$ forces a sequence change. The units with the fewest operating hours are set to the top of the FIFO and thus given priority for switching on.

In the sequence only units are entered that are enabled at inputs $b\text{En01..bEn08}$. $udi\text{Num}$ indicates the number of requested units.

The operating hours of the units are entered at inputs $udi\text{ActivTi01}_h$ to $udi\text{ActivTi08}_h$. If all these inputs are set to a constant value of zero, the sequence change is controlled cyclically, depending on $b\text{Chg}$.

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 $udi\text{Num}$ than are available at inputs $b\text{En01}$ to $b\text{En08}$, this is indicated with TRUE at $b\text{Err}$.

Error handling

If more units are requested at input $udi\text{Num}$ than are available at inputs $b\text{En01}$ to $b\text{En08}$, this is indicated with TRUE at $b\text{Err}$.

VAR_INPUT

$\text{bEn}$: Enables the function block.

$\text{udiNum}$: Number of units.

$\text{bChg}$: Force sequence change.

$b\text{En01}...b\text{En08}$: Enable unit 1...enable unit 8.

$udi\text{ActivTi01}_h...udi\text{ActivTi08}_h$: Operating hours unit 1...operating hours unit 8.

VAR_OUTPUT

$b\text{Q01}...b\text{Q08}$: Switches unit 1..8.

$\text{udiNextOn}$: Number of the unit that is switched on next.

$\text{udiNextOff}$: Number of the unit that is switched on next.

$\text{arrFIFO}$: FIFO buffer as a field.

```plaintext
VAR_INPUT
bEn : BOOL;
udiNum : UDINT;
bChg : BOOL;
bEn01 : BOOL;
bEn02 : BOOL;
bEn03 : BOOL;
bEn04 : BOOL;
bEn05 : BOOL;
bEn06 : BOOL;
bEn07 : BOOL;
bEn08 : BOOL;
udiActivTi01_h : UDINT;
udiActivTi02_h : UDINT;
udiActivTi03_h : UDINT;
udiActivTi04_h : UDINT;
udiActivTi05_h : UDINT;
udiActivTi06_h : UDINT;
udiActivTi07_h : UDINT;
udiActivTi08_h : UDINT;

VAR_OUTPUT
bQ01 : BOOL;
bQ02 : BOOL;
bQ03 : BOOL;
bQ04 : BOOL;
bQ05 : BOOL;
bQ06 : BOOL;
bQ07 : BOOL;
bQ08 : BOOL;
udiNextOn : UDINT;
udiNextOff : UDINT;
arrFIFO : ARRAY [1..8] OF UDINT;
udiNumOfEn : UDINT;
bErr : BOOL;
sErrDescr : STRING;
```
udiNumOfEn: Number of devices, depending on the individual enable states.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrMsg: Contains the error description.

### Error description
01: Warning: More than 8 devices are entered at input udiNum. The number is limited to the number enabled at inputs bEn01..bEn08.

### Requirements
<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.1.5.3.8 FB_BA_StepCtrl08

The function block is used for issuing sequential control commands. A typical application for this function block is startup of an air conditioning system. \( b\text{En} \) is used for general enable of the function block. If \( b\text{En} = \text{FALSE} \), all outputs of \( bQ01 \) to \( bQ08 \) are set to FALSE. The control sequence starts at input \( b\text{Evt}01 \). Once the timer \( udiDlyOn01\text{sec} \) (see Parameters) has elapsed, the corresponding output \( bQ01 \) is set. Further stages are activated after a rising edge at the inputs \( b\text{Evt}02 \) to \( b\text{Evt}08 \), in each case delayed via the timers \( udiDlyOn02\text{sec} \) to \( udiDlyOn08\text{sec} \). If \( b\text{Evt}01 \) becomes FALSE once the control chain is up and running, the control sequence switches back in reverse order. Switching off of the outputs is delayed by the timers \( udiDlyOff01\text{sec} \) to \( udiDlyOff08\text{sec} \); see Parameters.

The outputs \( b\text{Up} \) and \( b\text{Dwn} \) indicate whether the control chain is in ascending or descending state. The variable \( udi\text{ActvEvt} \) indicates the current step of the control chain. "0" means the step sequence is not active.
The output `udiRemTiDlyOn_sec` indicates the time remaining to the next step during up-switching of the control chain. The output `udiRemTiDlyOff_sec` indicates the time remaining to the next lower step during down-switching of the control chain.

**Example**

- **t0** step sequence switch-on
  - t1 switch on step 1 \( udiDlyOn01\_sec = t1 - t0 \)
- **t2** event enable step 2, switch on step 2, \( udiDlyOn02\_sec = 0 \)
- **t3** event enable step 3, switch on step 3, \( udiDlyOn03\_sec = 0 \)
- **t4** event enable step 4, switch on step 4, \( udiDlyOn04\_sec = 0 \)
- **t5** event enable step 5, switch on step 5, \( udiDlyOn05\_sec = 0 \)
- **t6** disable the step sequence, disable step 5, disable step 4; \( udiDlyOff05\_sec = 0, udiDlyOff04\_sec = 0 \)
- **t7** disable step 3, \( udiDlyOff03\_sec = t7 - t6 \)
- **t8** disable step 2, \( udiDlyOff02\_sec = t8 - t7 \)
- **t9** disable step 1, \( udiDlyOff01\_sec = t9 - t8 \)

**VAR_INPUT**

```
bEn : BOOL;
bEvt01 : BOOL;
udiDlyOn01_sec : UDINT;
udiDlyOff01_sec : UDINT;
bEvt02 : BOOL;
udiDlyOn02_sec : UDINT;
udiDlyOff02_sec : UDINT;
bEvt03 : BOOL;
udiDlyOn03_sec : UDINT;
udiDlyOff03_sec : UDINT;
bEvt04 : BOOL;
udiDlyOn04_sec : UDINT;
udiDlyOff04_sec : UDINT;
bEvt05 : BOOL;
udiDlyOn05_sec : UDINT;
udiDlyOff05_sec : UDINT;
```
bEn: Enable function block.

bEvt01..08: Switch-on command for steps 1 to 8.

udiDlyOn01..08_sec: Switch-on delay for output bQ01 .. 08 [s].
udiDlyOff01..08_sec: Switch-off delay for output bQ01 .. 08 [s].

VAR_OUTPUT

bQ01..08: Step 1 to 8 On.

bUp: Control chain is in ascending state.

bDwn: Control chain is in descending state.

udiActvEvt: Active step, display 0..8; "0" represents an active step sequence.

udiRTiDlyOn: Time remaining to up-switching to the next step [s].

udiRTiDlyOff: Time remaining to down-switching to the previous step [s].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twincat3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
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 \( bEn = \text{FALSE} \), all outputs of \( bQ01 \) to \( bQ12 \) are set to \text{FALSE}. The control sequence starts at input \( bEvt01 \). Once the timer \( udiDlyOn01\_sec \) (see Parameters) has elapsed, the corresponding output \( bQ01 \) is set. Further stages are activated after a rising edge at the inputs \( bEvt02 \) to \( bEvt12 \), in each case delayed via the timers \( udiDlyOn02\_sec \) to \( udiDlyOn12\_sec \). If \( bEvt01 \) becomes \text{FALSE} once the control chain is up and running, the control sequence switches back in reverse order. Switching off of the outputs is delayed by the timers \( udiDlyOff01\_sec \) to \( udiDlyOff12\_sec \); see Parameters.

The outputs \( bUp \) and \( bDwn \) indicate whether the control chain is in ascending or descending state. The variable \( udiActvEvt \) indicates the current step of the control chain. "0" means the step sequence is not active.

The output \( udiRemTiDlyOn\_sec \) indicates the time remaining to the next step during up-switching of the control chain. The output \( udiRemTiDlyOff\_sec \) indicates the time remaining to the next lower step during down-switching of the control chain.
Example

- t0 step sequence switch-on
- t1 switch on step 1 \( udiDlyOn01\_sec = t1 - t0 \)
- t2 event enable step 2, switch on step 2, \( udiDlyOn02\_sec = 0 \)
- t3 event enable step 3, switch on step 3, \( udiDlyOn03\_sec = 0 \)
- t4 event enable step 4, switch on step 4, \( udiDlyOn04\_sec = 0 \)
- t5 event enable step 5, switch on step 5, \( udiDlyOn05\_sec = 0 \)
- t6 disable the step sequence, disable step 5, disable step 4; \( udiDlyOff05\_sec = 0, udiDlyOff04\_sec = 0 \)
- t7 disable step 3, \( udiDlyOff03\_sec = t7 -t6 \)
- t8 disable step 2, \( udiDlyOff02\_sec = t8 -t7 \)
- t9 disable step 1, \( udiDlyOff01\_sec = t9 -t8 \)

VAR_INPUT

- bEn : BOOL;
- bEvt01 : BOOL;
- udiDlyOn01\_sec : UDINT;
- udiDlyOff01\_sec : UDINT;
- bEvt02 : BOOL;
- udiDlyOn02\_sec : UDINT;
- udiDlyOff02\_sec : UDINT;
- bEvt03 : BOOL;
- udiDlyOn03\_sec : UDINT;
- udiDlyOff03\_sec : UDINT;
- bEvt04 : BOOL;
- udiDlyOn04\_sec : UDINT;
- udiDlyOff04\_sec : UDINT;
- bEvt05 : BOOL;
- udiDlyOn05\_sec : UDINT;
- udiDlyOff05\_sec : UDINT;
- bEvt06 : BOOL;
- udiDlyOn06\_sec : UDINT;
- udiDlyOff06\_sec : UDINT;
- bEvt07 : BOOL;
- udiDlyOn07\_sec : UDINT;
bEn: Enable function block.

bEvt01..12: Switch-on command for steps 1 to 12.

udiDlyOn01..12_sec: Switch-on delay for output bQ01 .. 12 [s].

udiDlyOff01..12_sec: Switch-off delay for output bQ01 .. 12 [s].

VAR_OUTPUT

bQ01 : BOOL;
bQ02 : BOOL;
bQ03 : BOOL;
bQ04 : BOOL;
bQ05 : BOOL;
bQ06 : BOOL;
bQ07 : BOOL;
bQ08 : BOOL;
bQ09 : BOOL;
bQ10 : BOOL;
bQ11 : BOOL;
bQ12 : BOOL;
bUp : BOOL;
bDwn : BOOL;
udiActvEvt : UDINT;
udiRemTiDlyOn_sec : UDINT;
udiRemTiDlyOff_sec : UDINT;

bQ01..12: Step 1 to 12 On.

bUp: Control chain is in ascending state.

bDwn: Control chain is in descending state.

udiActvEvt: Active step, display 0..12; "0" represents an active step sequence.

udiRTiDlyOn: Time remaining to up-switching to the next step [s].

udiRTiDlyOff: Time remaining to down-switching to the previous step [s].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>TC3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.3.10 FB_BA_FIFO04_XX

This function block is used to evaluate the FiFo memory from FB_BA_FIFO04[142]. The inputs are linked according to the FIFO table to the corresponding outputs of the function block FB_BA_FIFO04_BOOL or FB_BA_FIFO04_REAL.

Sample:
In the sample the array contains: 4,3,1,2,0,0,0,0. The following result is output in FB_BA_FIFO04_REAL:

- rIn01 at output rVal04
- rIn02 at output rVal03
- rIn03 at output rVal01
- rIn04 at output rVal02

VAR_INPUT

arrFIFO : Array [1..4] OF UDINT;
rIn01 – rIn04 : REAL;

arrFIFO: Contains the assignment table with a maximum of eight values. The first value indicates where the first input will be copied to, the second value indicates where the second input will be copied to, etc. No assignment takes place with "0".

rIn01 - rIn04: Setpoints to be linked.

VAR_OUTPUT

rVal01 – rVal04 : REAL;

rVal01 - rVal04: set actuator value, input value linked according to FIFO table.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5.3.11 FB_BA_FIFO08_XX
This function block is used to evaluate the FiFo memory from FB_BA_FIFO08. The inputs are linked according to the FIFO table to the corresponding outputs of the function block FB_BA_FIFO08_BOOL or FB_BA_FIFO08_REAL.

Sample:
In the sample the array contains: 4,3,1,2,0,0,0,0. The result output in FB_BA_FIFO08_REAL is

- rIn01 at output rVal04
- rIn02 at output rVal03
- rIn03 at output rVal01
- rIn04 at output rVal02

VAR_INPUT

| arrFIFO | Array [1..8] OF UDINT;            |
| rIn01 – rIn08 | REAL;                      |

arrFIFO: Contains the assignment table with a maximum of eight values. The first value indicates where the first input will be copied to, the second value indicates where the second input will be copied to, etc. No assignment takes place with "0".

rIn01 – rIn08: Setpoints to be linked.

VAR_OUTPUT

| rVal01 – rVal08 | REAL;                      |

rVal01 – rVal08: set actuator value, input value linked according to FIFO table.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5.4 Hysteresis, 2-point control

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_Cont4Stp01</td>
<td>Step switch with four stages</td>
</tr>
<tr>
<td>FB_BA_Swi2P</td>
<td>Two-point switch</td>
</tr>
<tr>
<td>FB_BA_SwiHys2P</td>
<td>Two-point switch with one switching point</td>
</tr>
</tbody>
</table>
5.1.5.4.1 FB_BA_Cont4Stp01

The function block determines the resulting switching stages of a multi-level unit, depending on the input signal. Four switch-on thresholds and four hysteresis values can be parameterized.

Diagram 01

Control direction of parameter $bActn = \text{FALSE} = \text{Reverse} = \text{Heating}$
Diagram 02

<table>
<thead>
<tr>
<th>udiStp</th>
<th>udiNumOfStp</th>
<th>rSwiOn</th>
<th>rSwiOff</th>
<th>udi-RemTiD-lyOn_sec</th>
<th>udi-RemTiD-lyOff_sec</th>
<th>bQ01</th>
<th>bQ02</th>
<th>bQ03</th>
<th>bQ04</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>rSwiOn01</td>
<td>rSwiOn01</td>
<td>udiDlyOn01_sec</td>
<td>udiDlyOff01_sec</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>1</td>
<td>&gt;= 1</td>
<td>rSwiOn02</td>
<td>rSwiOn01</td>
<td>udiDlyOn02_sec</td>
<td>udiDlyOff02_sec</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>2</td>
<td>&gt;= 2</td>
<td>rSwiOn03</td>
<td>rSwiOn02</td>
<td>udiDlyOn03_sec</td>
<td>udiDlyOff03_sec</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>&gt;= 3</td>
<td>rSwiOn04</td>
<td>rSwiOn03</td>
<td>udiDlyOn04_sec</td>
<td>udiDlyOff04_sec</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>4</td>
<td>&gt;= 4</td>
<td>rSwiOn04</td>
<td>rSwiOn04</td>
<td>udiDlyOff04_sec</td>
<td>0</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

Control direction parameter bActn = TRUE = Direct = Cooling
Diagram 03

Timing of the switch-on and switch-off delays

At time t1, rIn jumps from rSwiOn01 to rSwiOn04
At time \( t_2 \), \( rIn \) jumps from \( rSwiOn04 \) to \( rSwiOn01 - rHys01 \)

**VAR_INPUT**

```plaintext
bEn : BOOL;
rIn : REAL;
rSwiOn01 : REAL;
rHys01 : REAL;
udiDlyOn01_sec : UDINT;
udiDlyOff01_sec : UDINT;
rSwiOn02 : REAL;
rHys02 : REAL;
udiDlyOn02_sec : UDINT;
udiDlyOff02_sec : UDINT;
rSwiOn03 : REAL;
rHys03 : REAL;
udiDlyOn03_sec : UDINT;
udiDlyOff03_sec : UDINT;
rSwiOn04 : REAL;
rHys04 : REAL;
udiDlyOn04_sec : UDINT;
udiDlyOff04_sec : UDINT;
udiNumOfStp : UDINT;
bActn : BOOL;
```

**bEn:** General enable of the function block. If \( bEn \) is FALSE, all outputs are set to 0.

**rIn:** Input value, from which the switching state is derived.

**rSwiOn01:** Switch-on point stage 01

**rHys01:** Absolute value hysteresis stage 01

**udiDlyOn01_sec:** Switch-on delay stage 01

**udiDlyOff01_sec:** Switch-off delay stage 01

**rSwiOn02:** Switch-on point stage 02

**rHys02:** Absolute value hysteresis stage 02

**udiDlyOn02_sec:** Switch-on delay stage 02

**udiDlyOff02_sec:** Switch-off delay stage 02
rSwiOn03: Switch-on point stage 03
rHys03: Absolute value hysteresis stage 03
udiDlyOn03_sec: Switch-on delay stage 03
udiDlyOff03_sec: Switch-off delay stage 03
rSwiOn04: Switch-on point stage 04
rHys04: Absolute value hysteresis stage 04
udiDlyOn04_sec: Switch-on delay stage 04
udiDlyOff04_sec: Switch-off delay stage 04
udiNumOfStp: Number of stages that are required. The input is limited to a range from 0 to 4
bActn: Input variable used to determine the control direction of the step switch. TRUE = direct = cooling; FALSE = reverse = heating

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bQ01</th>
<th>bQ02</th>
<th>bQ03</th>
<th>bQ04</th>
<th>udiStp</th>
<th>rSwiOn</th>
<th>rSwiOff</th>
<th>udiRemTiDlyOn_sec</th>
<th>udiRemTiDlyOff_sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>BOOL</td>
<td>BOOL</td>
<td>BOOL</td>
<td>UDIINT</td>
<td>REAL</td>
<td>REAL</td>
<td>UDIINT</td>
<td>UDIINT</td>
</tr>
</tbody>
</table>

bQ01: Display of status step 01
TRUE = ON; FALSE = OFF
udiStp >= 1

bQ02: Display of status step 02
TRUE = ON; FALSE = OFF
udiStp >= 2

bQ03: Display of status step 03
TRUE = ON; FALSE = OFF
udiStp >= 3

bQ04: Display of status step 04
TRUE = ON; FALSE = OFF
udiStp >= 4

udiStp: Shows the current step of the step switch
rSwiOn: Shows the next switch-on point
rSwiOff: Shows the next switch-off point
udiRemTiDlyOn_sec: If the switch-on point for switching to the next level is met, the progress of the switch-on delay time is displayed here.
udiRemTiDlyOff_sec: If the switch-off point for switching down to the next level is met, the progress of the switch-off delay time is displayed here.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
The function block FB_BA_Swi2P is a two-point switch with one switch-on point and one switch-off point. A general function block enable can be implemented at input bEn. The output bQ is FALSE as long as bEn is FALSE. The control direction of the function block depends on the relative position of the switch-on/switch-off points.

If the switch-on point is greater than the switch-off point, the control direction is direct/synchronous (cooling mode).

Is the switch-off point is greater than the switch-on point, the control direction is indirect/reversed (heating mode).
### VAR_INPUT

- `bEn : BOOL;`  
  General enable of the function block.
- `rIn : REAL;`  
  Input value.
- `rOn : REAL;`  
  Switch-on point.
- `rOff : REAL;`  
  Switch-off point.
- `udiDlyOn_sec : UDINT;`  
  Switch-on delay.
- `udiDlyOff_sec : UDINT;`  
  Switch-off delay.

### VAR_OUTPUT

- `bQ : BOOL;`  
  Control output.
- `udiRemTiDlyOn_sec : UDINT;`  
  Remaining time of the switch-on delay.
- `udiRemTiDlyOff_sec : UDINT;`  
  Remaining time of the switch-off delay.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.4.3 FB_BA_SwiHys2P

The function block FB_BA_SwiHys2P is a two-point switch with adjustable hysteresis and hysteresis offset.

A general function block enable can be implemented at input bEn. If the function block is locked, the output bQ is FALSE. The setpoint for the two-point switch is connected at input rSp. The control direction of the function block depends on the input variable bActn.

The active switching points result from the setpoint, the hysteresis and the hysteresis offset. They are output at rSwiHi and rSwiLo.

- The upper switching point results from \( rSp + \frac{rHys}{2} + rHysOffs \).
- The lower switching point results from \( rSp + \frac{rHys}{2} + rHysOffs \).

If bActn TRUE, the result is direct/synchronous control direction (cooling mode).

If bActn is FALSE, the result is indirect/reversed control direction (heating mode).

VAR_INPUT

<table>
<thead>
<tr>
<th>bEn</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>rIn</td>
<td>REAL</td>
</tr>
<tr>
<td>rSp</td>
<td>REAL</td>
</tr>
</tbody>
</table>
**VAR_IN**

- **rHys**: REAL;
- **rHysOffs**: REAL;
- **udiDlyOn_sec**: UDINT;
- **udiDlyOff_sec**: UDINT;
- **bActn**: BOOL;

**bEn**: General enable of the function block.

**rIn**: Input value.

**rSp**: Setpoint input.

**rHys**: Hysteresis.

**rHysOffs**: Hysteresis offset.

**udiDlyOn_sec**: Switch-on delay

**udiDlyOff_sec**: Release delay

**bActn**: Control direction.

**VAR_OUTPUT**

- **bQ**: BOOL;
- **rSwiHi**: REAL;
- **rSwiLo**: REAL;
- **udiRemTiDlyOn_sec**: UDINT;
- **udiRemTiDlyOff_sec**: UDINT;

**bQ**: Output.

**rSwiHi**: Upper switching point.

**rSwiLo**: Lower switching point.

**udiRemTiDlyOn_sec**: Time remaining before switching on.

**udiRemTiDlyOff_sec**: Time remaining before switching off.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.1.5.5 Mathematical functions

**Function blocks**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_MultiCalc_XX</td>
<td>Multi-calculation function blocks</td>
</tr>
<tr>
<td>FB_BA_Chrct02</td>
<td>Linear interpolation for 2 interpolation points</td>
</tr>
<tr>
<td>FB_BA_Chrct04</td>
<td>Linear interpolation for 4 interpolation points</td>
</tr>
<tr>
<td>FB_BA_Chrct07</td>
<td>Linear interpolation for 7 interpolation points</td>
</tr>
<tr>
<td>FB_BA_Chrct32</td>
<td>Linear interpolation for 32 interpolation points</td>
</tr>
<tr>
<td>FB_BA_TiAvrg</td>
<td>Arithmetic mean value over time</td>
</tr>
</tbody>
</table>
5.1.5.5.1 FB_BA_MultiCalc_XX

The multi-calculation function blocks exist for the variable types LREAL and REAL, although they all have the same functionality.

The function block FB_BA_R08 is described as an example.

In enabled state \( \text{bEn} = \text{TRUE} \), the function block determines the following from the 8 input values \( r01 \ldots r08 \):

- the maximum value of all inputs \( rMax \)
- the input at which this maximum value occurs \( \text{udiMaxActv} \)
- the minimum value of all inputs \( rMin \)
- the input at which this minimum value occurs \( \text{udiMinActv} \)
- the mean value of all inputs \( rAvrg \)
- the sum of all inputs \( rSum \)
- the difference between the maximum and minimum value \( rDiff \)

If not all inputs are used for the calculation, the number can be limited via an entry at \( \text{udiNum} \): \( \text{udiNum} = 6 \), for example, can be used to limit the calculations to inputs \( r01 \) to \( r06 \).

Any entry greater than 8 is automatically limited to 8, any entry less than 1 is automatically set to 1.

Sample:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( bEn = \text{TRUE} )</td>
<td>( rMax = 32 )</td>
</tr>
<tr>
<td>( r01 = 32 )</td>
<td>( \text{udiMaxActv} = 1 )</td>
</tr>
<tr>
<td>( r02 = 17 )</td>
<td>( rMin = 5 )</td>
</tr>
<tr>
<td>( r03 = 5 )</td>
<td>( \text{udiMinActv} = 3 )</td>
</tr>
<tr>
<td>( r04 = 9 )</td>
<td>( rAvrg = 18.5 )</td>
</tr>
<tr>
<td>( r05 = 16 )</td>
<td>( rSum = 111 )</td>
</tr>
<tr>
<td>( r06 = 32 )</td>
<td>( rDiff = 27 )</td>
</tr>
<tr>
<td>( r07 = 25 )</td>
<td></td>
</tr>
<tr>
<td>( r08 = 44 )</td>
<td></td>
</tr>
<tr>
<td>( \text{udiNum} = 6 )</td>
<td></td>
</tr>
</tbody>
</table>

If \( bEn = \text{FALSE} \), 0 is output at all outputs.

**VAR_INPUT**

\[
\begin{align*}
\text{bEn} & : \text{BOOL}; \\
\text{r01} & : \text{REAL}; \\
\text{r02} & : \text{REAL}; \\
\text{r03} & : \text{REAL}; \\
\text{r04} & : \text{REAL}; \\
\text{r05} & : \text{REAL}; \\
\text{r06} & : \text{REAL}; \\
\text{r07} & : \text{REAL}; \\
\text{r08} & : \text{REAL}; \\
\text{udiNum} & : \text{UDINT};
\end{align*}
\]
bEn: Activation of the block function.

r01...r08: Input values to be used for the calculation.

udiNum: Number of input values to be used for the calculation.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rMax</td>
<td>REAL</td>
</tr>
<tr>
<td>udiMaxActv</td>
<td>UDINT</td>
</tr>
<tr>
<td>rMin</td>
<td>REAL</td>
</tr>
<tr>
<td>udiMinActv</td>
<td>UDINT</td>
</tr>
<tr>
<td>rAvrg</td>
<td>REAL</td>
</tr>
<tr>
<td>rSum</td>
<td>REAL</td>
</tr>
<tr>
<td>rDiff</td>
<td>REAL</td>
</tr>
</tbody>
</table>

rMax: Maximum value of all inputs.

udiMaxActv: Input at which the maximum value is present.

rMin: Minimum value of all inputs.

udiMinActv: Input at which the minimum value is present.

rAvrg: Mean value of all inputs.

rSum: Sum of all inputs.

rDiff: Difference between maximum and minimum value.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5.5.2 FB_BA_Chrcct02

The function block FB_BA_Chrcct02 represents a linear interpolation with 2 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points \( [rX1/rY1] \) and \( [rX2/rY2] \). If the input variable \( bLmt \) is TRUE, \( rY \) is limited by \( rY01 \) and \( rY02 \). If \( bLmt \) is FALSE, \( rY \) is not limited.
Error handling

The input values for \( rX[n+1] \) must always be at least 0.0000001 greater than the values for \( rX[n] \). In the event of an error the variable \( sErrDescr \) indicates that at one point of the characteristic curve the values are not monotonically increasing.

**VAR_INPUT**

- \( rX \): Input value of the characteristic curve.
- \( rX01 \): X-value for interpolation point P1.
- \( rX02 \): X-value for interpolation point P2.
- \( rY01 \): Y-value for interpolation point P1.
- \( rY02 \): Y-value for interpolation point P2.
- \( bLmt \): Limit for the output value \( rY \).

**VAR_OUTPUT**

- \( rY \): Calculated output value of the characteristic curve.
- \( bErr \): This output is switched to TRUE if the parameters entered are erroneous.
- \( sErrDescr \): Contains the error description.

**Error description**

01: Error: \( rX01 \) must not be equal to \( rX02 \).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
The function block FB_BA_Chrc04 represents a linear interpolation with 4 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points \([rX1/rY1]\) to \([rX4/rY4]\). If the input variable \(bLmt\) is TRUE, \(rY\) is limited by \(rY01\) and \(rY04\). If \(bLmt\) is FALSE, \(rY\) is not limited.

Error handling

The input values for \(rX[n+1]\) must always be at least 0.0000001 greater than the values for \(rX[n]\). In the event of an error the variable \(sErrDescr\) indicates that at one point of the characteristic curve the values are not monotonically increasing.

**VAR_INPUT**

- \(rX\) : REAL;
- \(rX01\) : REAL;
- \(rX02\) : REAL;
- \(rX03\) : REAL;
- \(rX04\) : REAL;
- \(rY01\) : REAL;
- \(rY02\) : REAL;
- \(rY03\) : REAL;
- \(rY04\) : REAL;
- \(bLmt\) : BOOL;

\(rX\): Input value of the characteristic curve.

\(rX01\): X-value for interpolation point P1.

\(rX02\): X-value for interpolation point P2.

\(rX03\): X-value for interpolation point P3.

\(rX04\): X-value for interpolation point P4.

\(rY01\): Y-value for interpolation point P4.
rY02: Y-value for interpolation point P2.
rY03: Y-value for interpolation point P3.
rY04: Y-value for interpolation point P4.

bLmt: Limit for the output value \( rY \).

**VAR_OUTPUT**

```
VAR_OUTPUT
rY       : REAL;
bErr     : BOOL;
sErrDescr: T_MAXSTRING;
```

rY: Calculated output value of the characteristic curve.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

**Error description**

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Error: at the specified element. The sequence must always be ( rX01 &gt; rX02 &gt; rXn ) or ( rX01 &lt; rX02 &lt; rXn ).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.5.5.4 FB_BA_Chrc8707**

The function block FB_BA_Chrc8707 represents a linear interpolation with 7 interpolation points and can be used to generate a characteristic curve. The characteristic curve is determined by the interpolation points \([rX1/rY1] to [rX7/rY7]\). If the input variable \( bLmt \) is TRUE, \( rY \) is limited by \( rY01 \) and \( rY07 \). If \( bLmt \) is FALSE, \( rY \) is not limited.
Error handling
The input values for \( rX[n+1] \) must always be at least 0.0000001 greater than the values for \( rX[n] \).
In the event of an error the variable \( sErrDescr \) indicates that at one point of the characteristic curve the values are not monotonically increasing.

**VAR_INPUT**

- \( rX \) : REAL; Input value of the characteristic curve.
- \( rX01 \) : REAL; X-value for interpolation point P1.
- \( rX02 \) : REAL; X-value for interpolation point P2.
- \( rX03 \) : REAL; X-value for interpolation point P3.
- \( rX04 \) : REAL; X-value for interpolation point P4.
- \( rX05 \) : REAL; X-value for interpolation point P5.
- \( rX06 \) : REAL; X-value for interpolation point P6.
- \( rX07 \) : REAL; X-value for interpolation point P7.
- \( rY01 \) : REAL; Y-value for interpolation point P1.
- \( rY02 \) : REAL; Y-value for interpolation point P2.
- \( rY03 \) : REAL; Y-value for interpolation point P3.
- \( rY04 \) : REAL; Y-value for interpolation point P4.
- \( rY05 \) : REAL; Y-value for interpolation point P5.
- \( rY06 \) : REAL; Y-value for interpolation point P6.
- \( rY07 \) : REAL; Y-value for interpolation point P7.
- \( bLmt \) : BOOL; Limit for the output value \( rY \).
**VAR_OUTPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rY</td>
<td>REAL;</td>
</tr>
<tr>
<td>bErr</td>
<td>BOOL;</td>
</tr>
<tr>
<td>sErrDescr</td>
<td>T_MAXSTRING;</td>
</tr>
</tbody>
</table>

rY: Calculated output value of the characteristic curve.

bErr: This output is switched to TRUE if the parameters entered are erroneous.

sErrDescr: Contains the error description.

**Error description**

01: Error: at the specified element. The sequence must always be rX01 > rX02 > rXn or rX01 < rX02 < rXn.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

**5.1.5.5.5 FB_BA_Chrct32**

The function block FB_BA_Chrct32 represents a linear interpolation with up to 32 interpolation points and can be used to generate a characteristic curve. In contrast to the “smaller” interpolation function blocks FB_BA_Chrct02 [163], FB_BA_Chrct04 [165] and FB_BA_Chrct07 [166], and in the interest of clarity, the interpolation points are determined via field variables [arrX[1]/arrY[1] to [arrX[n]/arrY[n]]. If the input variable bLmt is TRUE, rY is limited by arrY[1] and arrY[n]. If bLmt is FALSE, rY is not limited.

**Error handling**

The input values for rX[n+1] must always be at least 0.0000001 greater than the values for rX[n]. In the event of an error the variable sErrDescr indicates that at one point of the characteristic curve the values are not monotonically increasing.

The parameter for the number of interpolation points, diNumOfElem, must be in the range 2..32.

**VAR_INPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rX</td>
<td>REAL;</td>
</tr>
<tr>
<td>arrX</td>
<td>ARRAY [1..cBA_NumOfElem] OF REAL;</td>
</tr>
<tr>
<td>arrY</td>
<td>ARRAY [1..cBA_NumOfElem] OF REAL;</td>
</tr>
<tr>
<td>diNumOfElem</td>
<td>DINT(2..32);</td>
</tr>
<tr>
<td>bLmt</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>
**VAR_OUTPUT**

- `rY` : REAL
- `bErr` : BOOL
- `sErrDescr` : T_MAXSTRING

**rY**: Calculated output value of the characteristic curve.

**bErr**: This output is switched to TRUE if the parameters entered are erroneous.

**sErrDescr**: Contains the error description.

**Error description**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Error: at the specified element. The sequence must always be rX01 &gt; rX02 &gt; rXn or rX01 &lt; rX02 &lt; rXn.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.1.5.5.6 FB_BA_TiAvrg

The function block `FB_BA_TiAvrg` calculates the arithmetic mean value of an analog value that was logged over a certain period. Discrete values are written into a FIFO buffer. `udiIntval_sec` specifies the time interval [s] over which the values are logged and written into the FIFO. Values are written if the input `bEn` is TRUE. The variable `udiNumOfElem` is used to determine the size of the FIFO buffer. It is limited to 1..512. The function block can be used for calculating an hourly mean outside temperature over a day, for example. In this case `udiNumOfElem` would be 24 and `udiIntval_sec` would be 3600 seconds. `bEn` is the general enable of the function block. If `bEn` = FALSE, the FIFO buffer within the function block is deleted completely, and no data are recorded.

**Example:**

`udiNumOfElem = 5`
### First cycle

<table>
<thead>
<tr>
<th>rln</th>
<th>rOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>t0</td>
<td>2/1 = 2</td>
</tr>
</tbody>
</table>

### Second cycle

<table>
<thead>
<tr>
<th>rln</th>
<th>rOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>(2+4)/2 = 3</td>
</tr>
</tbody>
</table>

### Third cycle

<table>
<thead>
<tr>
<th>rln</th>
<th>rOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>t2</td>
<td>(2+4+6)/3 = 4</td>
</tr>
</tbody>
</table>

### Fourth cycle

<table>
<thead>
<tr>
<th>rln</th>
<th>rOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>t3</td>
<td>(2+4+6+7)/4 = 4.75</td>
</tr>
</tbody>
</table>

### Monitoring functions

#### VAR_INPUT

- **bEn**: Enables the function block.
- **rIn**: Input value for averaging.
- **udiIntVal_SEC**: Time interval [s] for writing new values into the FIFO. Internally limited to a value between 1 and 2147483.
- **udiNumOfElem**: Size of the FIFO buffer. A change resets the previous averaging. Internally limited to a value between 1 and 512.

#### VAR_OUTPUT

- **rOut**: Calculated mean value.
- **rMax**: Largest value in the FIFO buffer.
- **rMin**: Smallest value in the FIFO buffer.

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_FixedLimitCtrl [171]</td>
<td>Limit value monitoring of a fixed value.</td>
</tr>
</tbody>
</table>

---

170  Version: 1.0  TC3 Building Automation
5.1.5.6.1 FB_BA_FixedLimitCtrl

Function block for monitoring a fixed limit value.

The input \( bEn \) is used for enabling the function block.

A tolerance range is defined around the value \( rIn \) to be monitored.

The tolerance range results from an upper limit value \( rInHighLimit \) and a lower limit value \( rInLowLimit \).

If the value \( rIn \) exceeds the upper limit value of the tolerance range, then the output \( bHighLimit \) becomes TRUE. A response delay of the output \( bHighLimit \) must be parameterized with the timer \( udiDelay_sec \).

If the value \( rIn \) falls below the lower limit of the tolerance range, output \( bLowLimit \) becomes TRUE. A response delay of the output \( bLowLimit \) must be parameterized with the timer \( udiDelay_sec \).

VAR_INPUT

\( bEn \): Function block enable.

\( rHighLimit \): Default upper limit value, preset to 32.

\( rLowLimit \): Default lower limit value, preset to 16.

\( rIn \): Input value to be monitored.

\( udiDelay_sec \): Output response delay [s]. Internally limited to values between 0 and \texttt{Const.udiTiSec[186]}. 

VAR_OUTPUT

\( bHighLimit \): Upper limit value reached.

\( bLowLimit \): Lower limit value reached.

\( udiRemTiDelay_sec \): Time remaining after a limit value has been exceeded until either the output \( bHighLimit \) or \( bLowLimit \) responds.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.6.2 FB_BA_FdbCtrlBinary

The function block is used for feedback monitoring of an actuator by means of digital feedback. Application examples of the function block are, for example, an operation feedback monitoring, a process feedback monitoring or the run monitoring of a drive by means of limit switches.

The input $bEn$ is used for enabling the function block. If $bEn$ is FALSE, the message output $bQ$ will always be FALSE.

The switching actuator output of the unit to be monitored is connected to the input $bActuator$. The $bSwitch$ input is used to connect the feedback signal (e.g. differential pressure switch, flow monitor or limit switch).

By means of the timer $udiFdbDelay_sec$ [s] a response delay of the feedback control after the start of the unit is set.

The second timer $udiInterruptionDelay_sec$ [s] serves for a response delay of the feedback control after reaching the final state.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bEn</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bActuator</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>bSwitch</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>$udiFdbDelay_sec$</td>
<td>: UDINT;</td>
</tr>
<tr>
<td>$udiInterruptionDelay_sec$</td>
<td>: UDINT;</td>
</tr>
</tbody>
</table>

$bEn$: Function block enable.

$bActuator$: Feedback of the switching output.

$bSwitch$: Feedback signal from the process.

$udiFdbDelay_sec$: Response delay [s] of the monitoring function when the actuator is started. Internally limited to values between 0 and $Const.udiTiSec\ [186]$.

$udiInterruptionDelay_sec$: Response delay [s] of the monitoring function when the actuator has already been started successfully (e.g. pressure fluctuations when monitoring the running of a fan). Internally limited to values between 0 and $Const.udiTiSec\ [186]$.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bQ</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>$udiRemTiFdbDelay$</td>
<td>: UDINT;</td>
</tr>
<tr>
<td>$udiRemTiInterruptionDelay$</td>
<td>: UDINT;</td>
</tr>
</tbody>
</table>

$bQ$: Output an error message if the feedback signal is not present within the parameterized time of $udiFdbDelay_sec$, or the feedback signal has been interrupted longer than after $udiInterruptionDelay_sec$.

$udiRemTiFdbDelay$: Remaining time [s] until output $bErrOpn$ is set.

$udiRemTiInterruptionDelay$: Remaining time [s] until output $bErrSwi$ is set.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.6.3 FB_BA_SlidingLimitCtrl

Function block for monitoring a floating setpoint.

The input \( b\text{En} \) is used for enabling the function block.

To check the function of a control system, the actual value is compared with the setpoint of the controlled system.

If the deviation of setpoint and actual value is within the tolerance range \( r\text{Hys} \), then the control is OK. If the actual value deviates from the setpoint by an amount outside this tolerance range over a longer period of time, the timer \( \text{udiDelay_sec} \) is started. After the timer has expired, if the control deviation is permanent, either the output \( b\text{LowLimit} \) or \( b\text{HighLimit} \) TRUE of the function block outputs a message.

\[
\begin{align*}
\text{VAR_INPUT} & \\
b\text{En} & : \text{BOOL} \\
r\text{W} & : \text{REAL} \\
r\text{X} & : \text{REAL} \\
r\text{Hys} & : \text{REAL} \\
\text{udiDelay_sec} & : \text{UDINT}
\end{align*}
\]

\( b\text{En} \): Function block enable.
**rW:** Setpoint.
**rX:** Actual value.
**rHys:** Hysteresis.

**udiDelay_sec:** Output response delay [s]. Internally limited to values between 0 and \texttt{Const.udiTiSec [186]}. 

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bHighLimit</td>
<td>BOOL</td>
</tr>
<tr>
<td>bLowLimit</td>
<td>BOOL</td>
</tr>
<tr>
<td>rHighLimit</td>
<td>REAL</td>
</tr>
<tr>
<td>rLowLimit</td>
<td>REAL</td>
</tr>
<tr>
<td>udiRemTiDelay_sec</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

**bHighLimit:** Upper limit value reached.

**bLowLimit:** Lower limit value reached.

**rHighLimit:** Output of the upper limit value.

**rLowLimit:** Output of the lower limit value.

**udiRemTiDelay_sec:** Time remaining after a limit value has been exceeded until either the output \texttt{bHighLimit} or \texttt{bLowLimit} responds.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

#### 5.1.5.7 Ramps, filters, controllers

**Function blocks**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_FltrPT1</td>
<td>First order filter</td>
</tr>
<tr>
<td>FB_BA_FltrPT1</td>
<td>First order filter</td>
</tr>
<tr>
<td>FB_BA_RampLmt</td>
<td>Ramp limitation</td>
</tr>
<tr>
<td>FB_BA_RampLmt</td>
<td>Ramp limitation</td>
</tr>
<tr>
<td>FB_BA_SeqCtrl</td>
<td>Sequence controller (function block in Tc3_BA_Common)</td>
</tr>
<tr>
<td>FB_BA_SeqCtrl</td>
<td>Sequence controller (function block in Tc3_BA_Common)</td>
</tr>
<tr>
<td>FB_BA_SeqLink</td>
<td>Sequence linker (function block in Tc3_BA_Common)</td>
</tr>
<tr>
<td>FB_BA_SeqLink</td>
<td>Sequence linker (function block in Tc3_BA_Common)</td>
</tr>
<tr>
<td>FB_BA_PIDCtrl</td>
<td>PID controller (function block in Tc3_BA_Common)</td>
</tr>
<tr>
<td>FB_BA_PIDCtrl</td>
<td>PID controller (function block in Tc3_BA_Common)</td>
</tr>
</tbody>
</table>

#### 5.1.5.7.1 FB_BA_FltrPT1

![FB_BA_FltrPT1](image)

First order filter.

- When the function block is first called (system startup), the output \texttt{rOut} is automatically set (once) to the input \texttt{rIn}.  


VAR_INPUT
rIn : REAL;
udiDampConst_sec : UDINT;
bSetActl : BOOL;

rIn: Input signal
udiDampConst_sec: Filter time constant [s]. Internally limited to values between 0 and 86400.
bSetActl: A rising edge at this input sets the output value rOut to the input value rIn.

VAR_OUTPUT
rOut : REAL;

rOut: Filtered output signal.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.1.5.7.2 FB_BA_RampLmt

The function block limits the increase or decrease speed of an input signal.
An increase of rIn results in the output rOut to be limited to the slope of (rHi-rLo)/udiTiUp.
A decrease of rIn results in the output rOut to be limited to the slope of (rHi-rLo)/udiTiUp.
VAR_INPUT

bEn : BOOL;
bEnRamp : BOOL;
rIn : REAL;
rHi : REAL;
rLo : REAL;
udiTiUp_sec : UDINT;
udiTiDwn_sec : UDINT;

**bEn**: Enable function block if FALSE, in which case \( rOut = 0.0 \).

**bEnRamp**: Enable ramp limitation if FALSE, in which case \( rOut = rIn \).

**rIn**: Input value of the ramp function

**rHi**: Upper interpolation point for calculating the ramps.

**rLo**: Lower interpolation point for calculating the ramps. \( rHi \) must be greater than \( rLo \), otherwise an error is output!

**udiTiUp_sec**: Rise time [s].

**udiTiDwn_sec**: Fall time [s]

VAR_OUTPUT

**rOut**: Output signal, slope-limited through the ramps

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.1.5.8 Calendar

Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_BA_SchedulerWeeklyXXCh</td>
<td>Weekly scheduler</td>
</tr>
<tr>
<td>FB_BA_CalendarXXCh</td>
<td>Yearly scheduler</td>
</tr>
</tbody>
</table>

5.1.5.8.1 FB_BA_SchedulerWeeklyXXCh

Weekly timer with 1, 7 or 28 timer channels.

The function block FB_BA_SchedulerWeekly07Ch is described as an example.

The function block is used to enter a total of up to 7 switch-on periods. Each switch-on period can be assigned a switch-on time [hh:mm:ss] and a switch-off time [hh:mm:ss]. The variables bMonday to bSunday can be used to select on which days of the week the switch-on period should be active.

A switch-on period is only active if the variable bEn of the channel is set to TRUE.

For irregular but recurring events, the variable bResetAfterOn can be set to TRUE. This will automatically reset the enable of channel bEn to FALSE after the event has finished.

To facilitate data entry, a rising edge at bAllActive sets bEn and all days of the week (bMonday to bSunday) to TRUE.

The function block is only active if a TRUE signal is present at bEn.

For demand-dependent switch-on optimization, switching on of the output bQ can be brought forward by the time of the variable udiPredictTime_sec.

The switch-on and switch-off points of a channel must be in the same year. The switch-off point must not be earlier than the switch-on point. Otherwise the switch-off point is automatically corrected and set to the same value as the switch-on point.

If the switch-on point is equal to the switch-off point, the channel remains off.

VAR_INPUT

| bEn : BOOL; |
| stSysTi : TIMESTRUCT; |
| udiPredictTime_sec : UDINT; |

bEn: General function block enable.

stSysTi: Structure with the local NT system time (see TIMESTRUCT).

udiPredictTime_sec: Precalculated switch-on time. Internally limited to values between 0 and 43200.

VAR_OUTPUT

| bQ : BOOL; |

bQ: Switching output

VAR_IN_OUT

arrChannel: ARRAY [1..cBA_NumOfChannels] OF ST_BA_SchedulerWeeklyChannel;

arrChannel: Weekly scheduler; with the single-channel function block, the name of the variable is stChannel (see ST_BA_SchedulerWeeklyChannel [185]). Internally limited to the respective number of possible channels via the variable cBA_NumOfChannels.
5.1.5.8.2 FB_BA_CalendarXXCh

Yearly scheduler with 1, 7 or 28 channels.

The function block FB_BA_Calendar07Ch is described as an example.

This function block is used to enter periods such as school holidays or company holidays. The function block is enabled by the input variable bEnable. The input stSysTi is linked to the current system time. If the time switching condition is fulfilled, output bQ is set. Within the calendar, a time period is described by a switch-on date [day, month, hour, minute] and a switch-off date [day, month, hour, minute].

A switch-on period is only active if the variable bEn of the channel is set to TRUE. For irregular but recurring periods, the variable bResetAfterOn can be set to TRUE. The enable parameter bEn is then automatically reset to FALSE after the time has elapsed.

The switch-on and switch-off points of a channel must be in the same year. The switch-off point must not be earlier than the switch-on point. Otherwise the switch-off point is automatically corrected and set to the same value as the switch-on point.

If the switch-on point is equal to the switch-off point, the channel remains off.

**VAR_INPUT**

bEn : BOOL;
stSysTi : TIMESTRUCT;

bEn: General function block enable.
stSysTi: Structure with the local NT system time (see TIMESTRUCT).

**VAR_OUTPUT**

bQ : BOOL;

bQ: Switching output

**VAR_IN_OUT**

arrChannel: ARRAY [1..7] OF ST_BA_CalendarChannel;

arrChannel: Yearly scheduler; with the single-channel function block, the name of the variable is stChannel (see ST_BA_CalendarChannel [184]). Internally limited to the respective number of possible channels via the variable cBA_NumOfChannels.

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
5.2 DUTs

5.2.1 Enums

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_BA_PosMod [179]</td>
<td>Enumerator for the definition of the positioning mode.</td>
</tr>
<tr>
<td>E_BA_ShdObjType [179]</td>
<td>Enumerator for selecting the shading object type.</td>
</tr>
<tr>
<td>E_BA_Sensor</td>
<td>Enumerator for selecting a sensor type for measuring analog values.</td>
</tr>
<tr>
<td>E_BA_Terminal_KL</td>
<td>Enumerator for selecting the respective Bus Terminal.</td>
</tr>
</tbody>
</table>

5.2.1.1 E_BA_PosMod

Enumerator for the definition of the positioning mode.

```lang-c
TYPE E_BA_PosMod :
  (PosModFix:= 0,
   PosModTab,
   PosModMaxIndc);
END_TYPE
```

`PosModFix`: The blind height is a fixed value, which is set at function block FB_BA_SunPrtc [107] via the value `lrFixPos [%].`

`PosModTab`: The height positioning takes place with the help of a table of 6 interpolation points, 4 of which are parameterizable. A blind position in relation to the position of the sun is then calculated from these points by linear interpolation. For a more detailed description please refer to FB_BA_BldPosEntry [59].

`PosModMaxIndc`: The positioning takes place with specification of the maximum desired incidence of light.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.1.2 E_BA_ShdObjType

Enumerator for selecting the shading object type.

```lang-c
TYPE E_BA_ShdObjType :
  (ObjTypeTetragon := 0,
   ObjTypeGlobe);
END_TYPE
```

`ObjTypeTetragon`: Object type is a rectangle.

`ObjTypeGlobe`: Object type is a ball.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2 Structures

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_BA_BldPosTab [▲180]</td>
<td>Structure of the interpolation point entries for the height adjustment of the blind.</td>
</tr>
<tr>
<td>ST_BA_Cnr [▲180]</td>
<td>Information about window corners.</td>
</tr>
<tr>
<td>ST_BA_Fcd [▲181]</td>
<td>Facade-specific data for activating the automatic functions.</td>
</tr>
<tr>
<td>ST_BA_FcdElem [▲181]</td>
<td>List entry for a facade element (window).</td>
</tr>
<tr>
<td>ST_BA_Shdo [▲182]</td>
<td>List entry for a shading object</td>
</tr>
<tr>
<td>ST_BA_SpRmT [▲183]</td>
<td>Room temperature setpoints.</td>
</tr>
<tr>
<td>ST_BA_Sunbl [▲183]</td>
<td>Structure of the blind positioning telegram.</td>
</tr>
<tr>
<td>ST_BA_SunBldScn [▲184]</td>
<td>Table entry for a blind scene.</td>
</tr>
<tr>
<td>ST_BA_CalenderChannel [▲184]</td>
<td>Input of calendar entries.</td>
</tr>
<tr>
<td>ST_BA_SchedulWecklyChan [▲185]</td>
<td>Input of time switch entries.</td>
</tr>
</tbody>
</table>

5.2.2.1 ST_BA_BldPosTab

Structure of the interpolation point entries for the height adjustment of the blind.

```
TYPE ST_BA_BldPosTab:
  STRUCT
    rSunElv : ARRAY[0..5] OF REAL;
    rPos   : ARRAY[0..5] OF REAL;
    bVld   : BOOL;
  END_STRUCT
END_TYPE
```

rSunElv / rPos: The 6 interpolation points that are transferred, wherein the array elements 0 and 5 represent the automatically generated edge elements mentioned above.

bVld: Validity flag for the function block FB_BA_SunPrtc [▲107]. It is set to TRUE by the function block FB_BA_BldPosEntry [▲59] if the data entered correspond to the validity criteria described.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2.2 ST_BA_Cnr

Information about window corners.

```
TYPE ST_BA_Cnr :
  STRUCT
    rX      : REAL;
    rY      : REAL;
  END_STRUCT
END_TYPE
```
Programming

**rX**: X-coordinate of the window (on the facade).

**rY**: Y-coordinate of the window (on the facade).

**bShdd**: Information as to whether this corner is in the shade: \( bShdd=\text{TRUE} \): Corner is in the shade.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.2.2.3 **ST_BA_Fcd**

Facade-specific data at room level for activating the automatic functions.

**TYPE** ST_BA_Fcd:

```plaintext
STRUCT
  rSunPrtcAngl  : REAL;
  rSunPrtcPos   : REAL;
  rFcdThAutoPos : REAL;
  rFcdThAutoAngl: REAL;
  bFcdThAutoEn  : BOOL;
  bThAutoEn     : BOOL;
  bTwiLgtAutoEn : BOOL;
  bSunPrtcEn    : BOOL;
END_STRUCT
END_TYPE
```

**rSunPrtcAngl**: Sun protection: Current calculated position [%] for the blinds.

**rSunPrtcPos**: Sun protection: Current calculated louvre angle [°] for the blinds.

**rFcdThAutoPos**: Thermo-automatic function for whole facade: Currently valid position [%] for the blinds (heating or cooling position).

**rFcdThAutoAngl**: Thermo-automatic function for whole facade: Currently valid louvre angle [°] for the blinds (heating or cooling position).

**bFcdThAutoEn**: Thermo-automatic function for whole facade enabled.

**bThAutoEn**: Thermo-automatic function enabled.

**bTwiLgtAutoEn**: Automatic twilight function enabled.

**bSunPrtcEn**: Automatic sun protection enabled.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.2.2.4 **ST_BA_FcdElem**

List entry for a facade element (window).

**TYPE** ST_BA_FcdElem:

```plaintext
STRUCT
  rWdwWdth  : REAL;
  rWdwHght  : REAL;
  stCnr     : ARRAY [1..4] OF ST_BA_Cnr;
  diGrp     : DINT;
END_STRUCT
END_TYPE
```
bVld : BOOL;
END_STRUCT
END_TYPE

rWdwWdth: Width of the window.

rWdwHght: Height of the window.

stCnr: Coordinates of the window corners and information as to whether this corner point is in the shade; see ST_BA_Cnr [\textnumero 180].

bVld: Plausibility of the data entered: bVld=TRUE: Data are plausible.

diGrp: Group membership of the element.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2.5 ST_BA_ShdoBj

List entry for a shading object.

```plaintext
TYPE ST_BA_ShdoBj :
STRUCT
  rP1x       : REAL;
  rP1y       : REAL;
  rP1z       : REAL;
  rP2x       : REAL;
  rP2y       : REAL;
  rP2z       : REAL;
  rP3x       : REAL;
  rP3y       : REAL;
  rP3z       : REAL;
  rP4x       : REAL;
  rP4y       : REAL;
  rP4z       : REAL;
  rMx        : REAL;
  rMy        : REAL;
  rMz        : REAL;
  rRads      : REAL;
  diBegMth   : USINT;
  diEndMth   : USINT;
  eType      : E_BA_ShdoBjType;
  bVld       : BOOL;
END_STRUCT
END_TYPE
```

rP1x .. rP4z: Corner coordinates. Of importance only if the element is a square.

rMx .. rMz: Center coordinates. Of importance only if the element is a ball.

rRads: Radius of the ball. Of importance only if the element is a ball.

diBegMth: Beginning of the shading period (month).

diEndMth: End of the shading period (month).

eType: Object type, see E_BA_ShdoBjType [\textnumero 179].

bVld: Plausibility of the data: bVld=TRUE: Data are plausible.

Remark about the shading period:

The entries for the months may not be 0 or greater than 12, otherwise all combinations are possible.
Examples:
Start=1, End=1: shading in January.
Start=1, End=5: shading from the beginning of January to the end of May.
Start=11, End=5: shading from the beginning of November to the end of May (the following year).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2.6 **ST_BA_SpRmT**

Room temperature setpoints.

```plaintext
TYPE ST_BA_SpRmT :
STRUCT
  rPrtcHtg : REAL := 12.0;
  rEcoHtg  : REAL := 15.0;
  rPreCmfHtg : REAL := 19.0;
  rCmfHtg  : REAL := 21.0;
  rPrtcCol : REAL := 40.0;
  rEcoCol  : REAL := 35.0;
  rPreCmfCol : REAL := 28.0;
  rCmfCol  : REAL := 24.0;
END_STRUCT
END_TYPE
```

The values in the structure are defined with the preset values.

The variables have the following meaning:

- **rPrtcHtg**: Protection Heating.
- **rEcoHtg**: Economy Heating.
- **rPreCmfHtg**: Pre-Comfort Heating.
- **rCmfHtg**: Comfort Heating.
- **rPrtcCol**: Protection Cooling.
- **rEcoCol**: Economy Cooling.
- **rPreCmfCol**: Pre-Comfort Cooling.
- **rCmfCol**: Comfort Cooling.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2.7 **ST_BA_Sunbld**

Structure of the blind positioning telegram.

```plaintext
TYPE ST_BA_Sunbld:
STRUCT
  rPos        : REAL;
  rAngl       : REAL;
  bManUp      : BOOL;
  bManDwn     : BOOL;
  bManMod     : BOOL;
END_STRUCT
```

TC3 Building Automation
Version: 1.0
bActv : BOOL;
END_STRUCT
END_TYPE

rPos: Transferred blind height [%].

rAngl: Transferred louvre position [°].

bManUp: Manual command: blind up.

bManDwn: Manual command: blind down.

bManMod: TRUE: Manual mode is active. FALSE: Automatic mode is active.

bActv: The sender of the telegram is active. This bit is only evaluated by the priority control
FB_BA_SunBldPrioSwi4 [97] or FB_BA_SunBldPrioSwi8 [98]. The sun protection actuators
FB_BA_SunBldActr [89] and FB_BA_RolBldActr [81] ignore it.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2.8 ST_BA_SunBldScn

Table entry for a blind scene.

TYPE ST_BA_SunBldScn:
STRUCT
  rPos      : REAL;
  rAngl    : REAL;
END_STRUCT
END_TYPE

rPos: Blind height [%].

rAngl: Louvre position [°].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2.9 ST_BA_CalendarChannel

Structure for entering calendar entries.

TYPE ST_BA_CalendarChannel:
STRUCT
  udiOn_Day      : UDINT(1..31);
  udiOn_Month    : UDINT(1..12);
  udiOn_hh       : UDINT(0..23);
  udiOn_mm       : UDINT(0..59);
  udiOff_Day     : UDINT(1..31);
  udiOff_Month   : UDINT(1..12);
  udiOff_hh      : UDINT(0..23);
  udiOff_mm      : UDINT(0..59);
  bEn            : BOOL;
  bResetAfterOn  : BOOL;
  bQ              : BOOL;
END_STRUCT
END_TYPE

udiOn_Day: Switch-on point for day.

udiOn_Month: Switch-on point for month.
udiOn_hh: Switch-on point for hour.
udiOn_mm: Switch-on point for minute.
udiOff_Day: Switch-off point for day.
udiOff_Month: Switch-off point for month.
udiOff_hh: Switch-off point for hour.
udiOff_mm: Switch-off point for minute.
bEn: TRUE -> Enable channel, FALSE -> bQ = FALSE
bResetAfterOn: One-time and non-recurring switching-on.
bQ: Channel status output.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.2.2.10 ST_BA_SchedulerWeeklyChannel

Structure for entering time switch entries.

```
TYPE ST_BA_SchedulerWeeklyChannel:
  STRUCT
    udiOn_hh   : UDINT(0..23);
    udiOn_mm   : UDINT(0..59);
    udiOn_ss   : UDINT(0..59);
    udiOff_hh  : UDINT(0..23);
    udiOff_mm  : UDINT(0..59);
    udiOff_ss  : UDINT(0..59);
    bAllActive : BOOL;
    bEn        : BOOL;
    bMonday    : BOOL;
    bTuesday   : BOOL;
    bWednesday : BOOL;
    bThursday  : BOOL;
    bFriday    : BOOL;
    bSaturday  : BOOL;
    bSunday    : BOOL;
    bResetAfterOn : BOOL;
    bQ         : BOOL;
  END_STRUCT
END_TYPE
```

udiOn_hh: Switch-on point for hour.
udiOn_mm: Switch-on point for minute.
udiOn_ss: Switch-on point for second.
udiOff_hh: Switch-off point for hour.
udiOff_mm: Switch-off point for minute.
udiOff_ss: Switch-off point for second.
bAllActive: Activation of the timer condition for all weekdays.
bEn: TRUE -> Enable channel, FALSE -> bQ = FALSE.
bMonday: Switch-on point for Monday.
bTuesday: Switch-on point for Tuesday.
bWednesday: Switch-on point for Wednesday.
bThursday: Switch-on point for Thursday.
bFriday: Switch on point for Friday.
bSaturday: Switch-on point for Saturday.
bSunday: Switch-on point for Sunday.
bResetAfterOn: One-time and non-recurring switching-on.
bQ: Channel status output.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

5.3 GVLs

5.3.1 Constants

Global constants

VAR_GLOBAL CONSTANT

rClsZero : REAL := 0.00001;
udiNoActvPrio : UDINT := 4294967295;
udiTiSec : UDINT := 4294967295;
wSUNDAY : WORD := 0;
wMONDAY : WORD := 1;
wTUESDAY : WORD := 2;
wWEDNESDAY : WORD := 3;
wTHURSDAY : WORD := 4;
wFRIDAY : WORD := 5;
wSATURDAY : WORD := 6;
TimeValue24h_ms : UDINT := 86400000;
END_VAR

rClsZero: Reference value to avoid division by zero.
udiNoActvPrio: The value of the constants indicates that no priority is active.
udiTiSec: Constant for specifying a time in seconds.
wSUNDAY: Constant value for Sunday.
wMONDAY: Constant value for Monday.
wTUESDAY: Constant value for Tuesday.
wWEDNESDAY: Constant value for Wednesday.
wTHURSDAY: Constant value for Thursday.
wFRIDAY: Constant value for Friday.
wSATURDAY: Constant value for Saturday.
TimeValue24h_ms: Time value for 24 hours in milliseconds.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>

### 5.3.2 Parameter

Global parameters

```plaintext
VAR_GLOBAL CONSTANT
usiMaxSunBldScn   : USINT := 20;
iMaxRowFcd       : UINT  := 10;
iMaxColumnFcd    : UINT  := 20;
iMaxShdObj       : UINT  := 20;
udiMaxDataFileSize: UDINT := 100000;
END_VAR
```

**usiMaxSunBldScn**: Maximum number of scenes that are processed by the function block `FB_BA_SunBldScn` [99].

**uiMaxRowFcd**: Maximum number of floors for which the shading correction applies (horizontal arrangement of windows).

**uiMaxColumnFcd**: Maximum number of axes for which the shading correction applies (vertical arrangement of windows).

**uiMaxShdObj**: Maximum number of shading objects that cast shadows on the facade.

**udiMaxDataFileSize**: Maximum file size for the Excel list (in bytes), which is read by the function blocks `FB_BA_RdFcdElemLst` [73] and `FB_BA_RdShdObjLst` [77].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required library</th>
<th>Necessary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT3.1 4022.22</td>
<td>Tc3 Building Automation from V1.1.0.0</td>
<td>TF8040</td>
</tr>
</tbody>
</table>
Appendix

6 Appendix

6.1 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.

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 her internet pages:
http://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone: +49(0)5246/963-0
Fax: +49(0)5246/963-198
e-mail: info@beckhoff.com

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(0)5246/963-157
Fax: +49(0)5246/963-9157
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(0)5246/963-460
Fax: +49(0)5246/963-479
e-mail: service@beckhoff.com