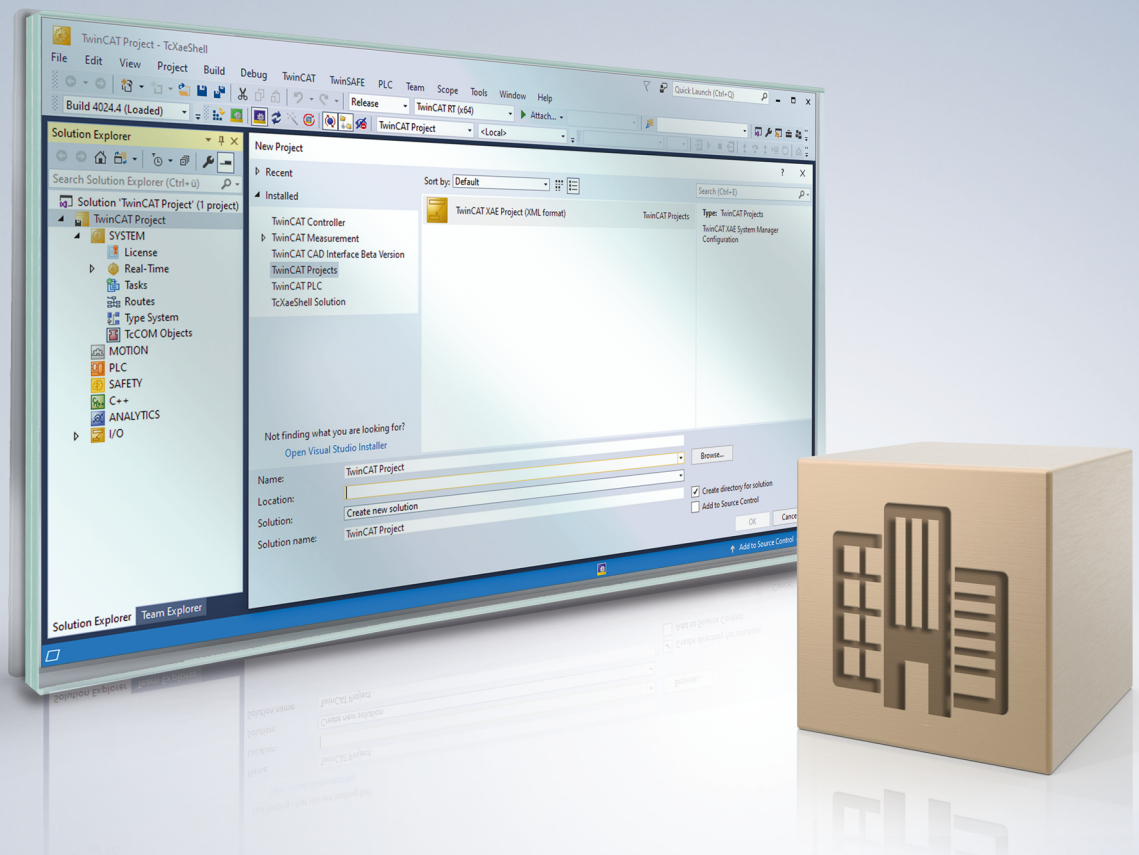
BECKHOFF New Automation Technology

Handbuch | DE

TF8040

TwinCAT 3 | Building Automation



26.05.2025 | Version: 1.14.0

Inhaltsverzeichnis

1 Vorwort ... 5
1.1 Hinweise zur Dokumentation ... 5
1.2 Zu Ihrer Sicherheit ... 6
1.3 Hinweise zur Informationssicherheit ... 7
2 Übersicht ... 8
2.1 Zielgruppen ... 8
2.2 Anforderungsprofil ... 8
2.3 Installation ... 8
2.4 Lizenzierung ... 11
2.5 Systemvoraussetzungen ... 13
2.6 Energieeffizienz ... 14
3 Konzepte ... 15
3.1 Benutzerrollen ... 15
3.2 Kommunikation ... 16
3.3 Systemaufbau ... 17
3.4 ADS-Kommunikation innerhalb der Templates ... 18
3.5 Raumautomation ... 21
3.5.1 Schalenmodell ... 23
3.5.2 Raumfunktionen ... 23
3.6 Grundgerüst ... 31
3.6.1 Objekte ... 33
3.6.2 Projektstruktur ... 39
3.6.3 DPAD ... 43
3.7 SPS ... 53
3.7.1 Steuer- und Regelmechanismen ... 53
3.7.2 Schnelle Task für serielle Kommunikation ... 55
3.8 HMI ... 56
3.8.1 Generik ... 56
3.8.2 Trending ... 56
3.8.3 Feedback ... 62
3.8.4 BaObject ... 62
3.8.5 BaTemplate ... 63
3.8.6 BaInterface ... 64
4 Tutorials ... 66
4.1 SPS ... 66
4.1.1 Start eines Projekts ... 66
4.2 HMI ... 72
4.2.1 Start eines Projekts ... 72
4.2.2 Generische HMI ... 79
5 Beispiele ... 89
5.1 Konzeptbeispiele ... 90
5.1.1 HMI ... 90
5.2 Template Beispiele ... 96

5.2.1 HMI ... 96
5.2.2 PLC ... 98

6 Programmierung ... 100
6.1 SPS ... 100
6.1.1 Allgemein ... 100
6.1.2 Bibliotheken ... 101
6.1.3 SPS-Project-Templates ... 706
6.1.4 Templates ... 709
6.2 HMI ... 1139
6.2.1 TcHmiBa ... 1139

7 Tools ... 1306
7.1 Building Automation Site Explorer ... 1306
7.2 Symbol Explorer ... 1324
7.2.1 Einleitung ... 1324
7.2.2 Begriffserklärung ... 1325
7.2.3 Benutzeroberfläche ... 1325
7.2.4 Getting started ... 1338
7.2.5 Filtern, Suchen und Ersetzen ... 1344
7.2.6 Command Line Interface ... 1348
7.2.7 Workflows ... 1349
7.2.8 Extensions ... 1361
7.2.9 Beispiele für reguläre Ausdrücke ... 1366
7.3 Template Repository ... 1366

8 Anhang ... 1373
8.1 Third-party components ... 1373
8.2 Support und Service ... 1373

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.



EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.
Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.
Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt



Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

Die Grundvoraussetzung, um die hohen Anforderungen an die Gebäudeautomation wie Komfort, Energieeffizienz, geringe Investitions- und Betriebskosten und ein schneller Return on Investment zu erfüllen, ist ein durchgängiges, aufeinander abgestimmtes Steuerungssystem für die Automation aller technischen Ausbaugewerke.

Mit TwinCAT 3 Building Automation hat Beckhoff ein Softwareprodukt entwickelt, das die Engineering- Zeit reduziert und alle wesentlichen Funktionen für alle Gewerke einer modernen Gebäudeautomation integriert. Umfangreiche Softwarebibliotheken und Tools setzen den Gedanken des modularen Beckhoff Automatisierungsbaukastens auch auf der Softwareebene fort. Die neue Softwaresuite umfasst im Wesentlichen drei Grundfunktionen:

TwinCAT 3 BA PLC Bibliotheken [▶ 101]:

Basisfunktionen für alle Gewerke.

TwinCAT 3 BA PLC Templates [▶ 709]:

Funktionsvorlagen für alle Gewerke.

TwinCAT 3 BA Tools [▶ 1306]:

Softwaretools für die optimierte Realisierung von Gebäudeautomationsprojekten bezüglich der Datenbearbeitung, der Inbetriebnahme sowie der Einregulierung und der Wartung.

Durch den Einsatz der Funktion TwinCAT 3 Building Automation sind alle SPS-Programme von der Heizungszentrale über die Vollklimaanlage bis hin zu den Raumautomationsfunktionen mit TwinCAT PLC Control programmierbar und stehen innerhalb der Gebäudeautomationsbibliotheken als Bausteine zur Verfügung.

Die für die TwinCAT 3 Building Automation Bibliothek (Tc3_BA) notwendigen PID-Regler, Sequenzregler und der Sequenzlinker befinden sich in der vorinstallierten Bibliothek Tc3_BA_Common.

Die für die TwinCAT 3 Building Automation Bibliothek (Tc3_BA2) notwendigen Funktionen und Regler befinden sich in der vorinstallierten Bibliothek Tc3_BA2_Common.

2.1 Zielgruppen

Diese Software ist für Gebäudeautomations-Systempartner der Beckhoff Automation GmbH & Co. KG. Die Systempartner sind tätig in dem Bereich Gebäudeautomation und beschäftigen sich mit Errichtung, Inbetriebsetzung, Erweiterung, Wartung und Service von mess-, steuer- und regelungstechnischen Anlagen der technischen Gebäudeausrüstung.

2.2 Anforderungsprofil

Für den Nutzer werden die folgenden Grundkenntnisse vorausgesetzt.

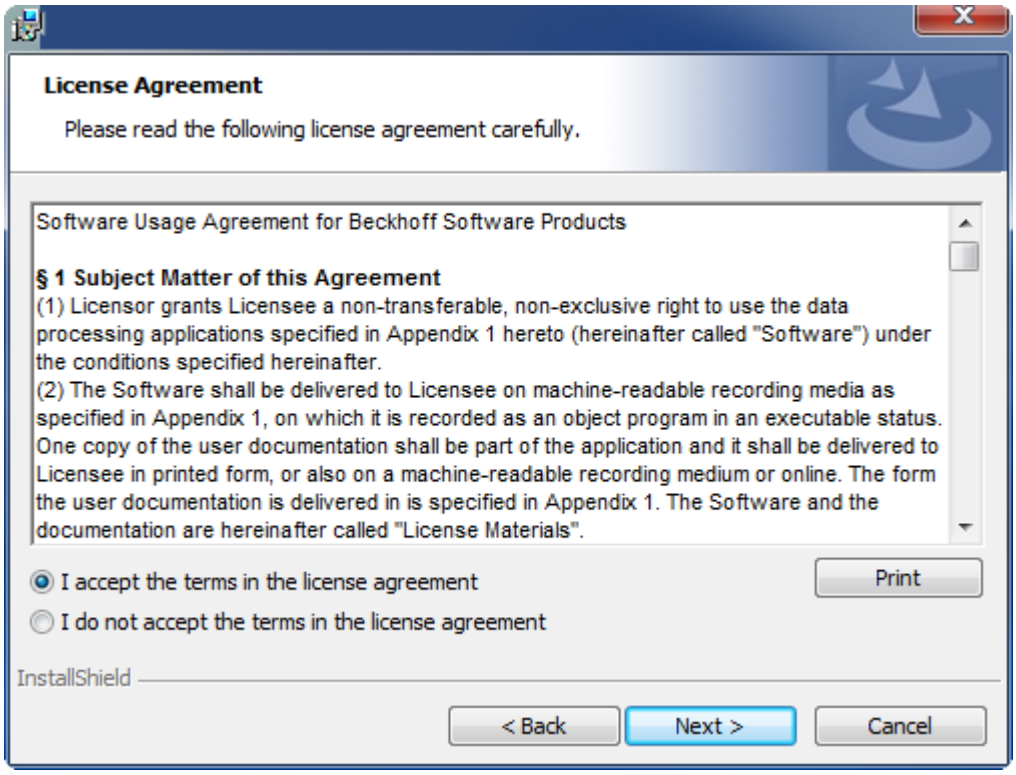
- TwinCAT 3
- Programmierkenntnisse in IEC61131-3
- PC und Netzwerkkenntnisse
- Aufbau und Eigenschaften der Beckhoff Embedded-PC und deren Busklemmensystem
- Kenntnisse in Heizungs-, Lüftungs-, Klima-, Sanitäranlagen- sowie der Raumautomation
- Einschlägige Sicherheitsvorschriften der technischen Gebäudeausrüstung

2.3 Installation

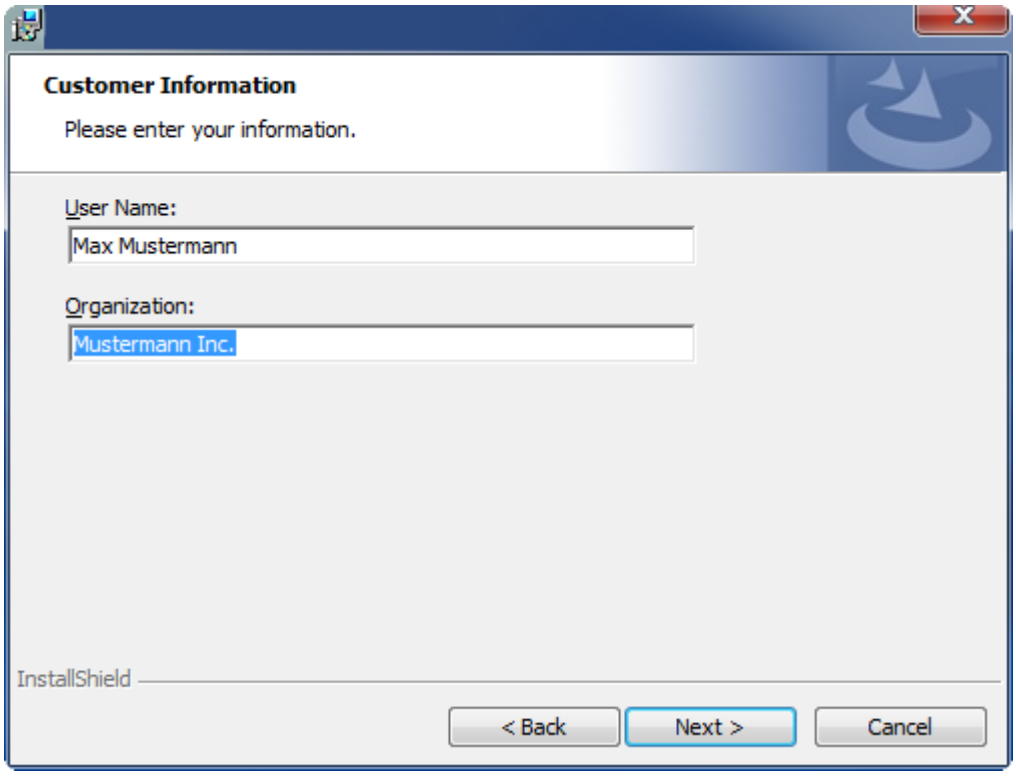
Nachfolgend wird beschrieben, wie die TwinCAT 3 Function für Windows-basierte Betriebssysteme installiert wird.

✓ Die Setup-Datei der TwinCAT 3 Function wurde von der Beckhoff-Homepage heruntergeladen.

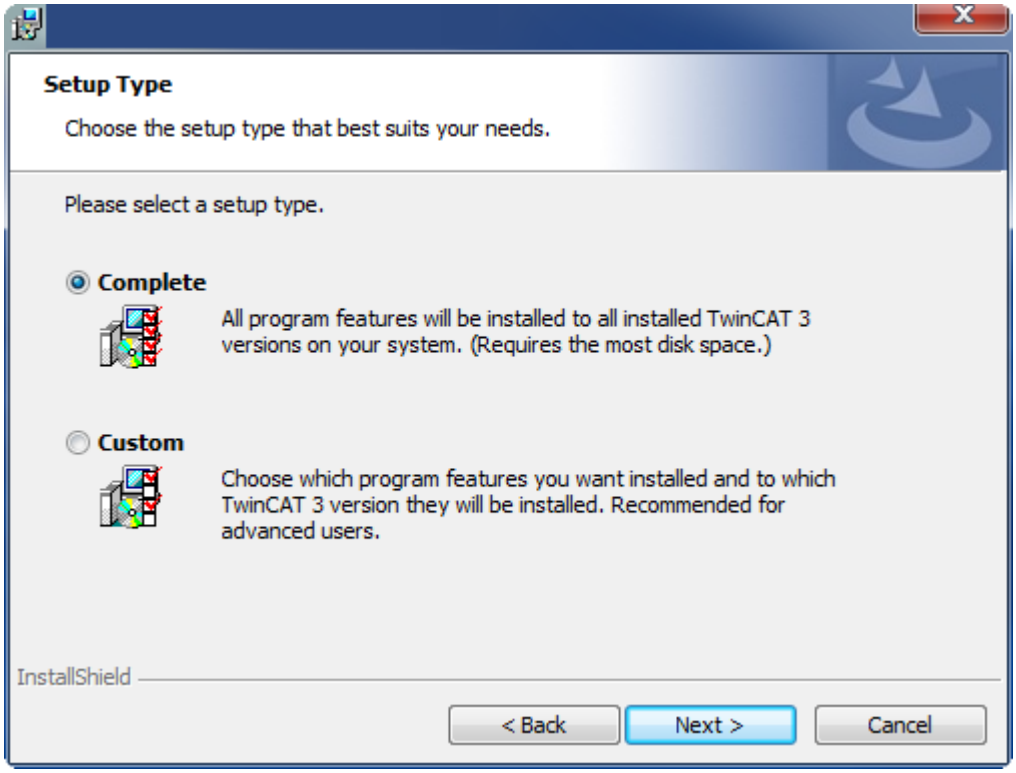
1. Führen Sie die Setup-Datei als Administrator aus. Wählen Sie dazu im Kontextmenü der Datei den Befehl **Als Administrator ausführen**.
   ⇨ Der Installationsdialog öffnet sich.
2. Akzeptieren Sie die Endbenutzerbedingungen und klicken Sie auf **Next**.



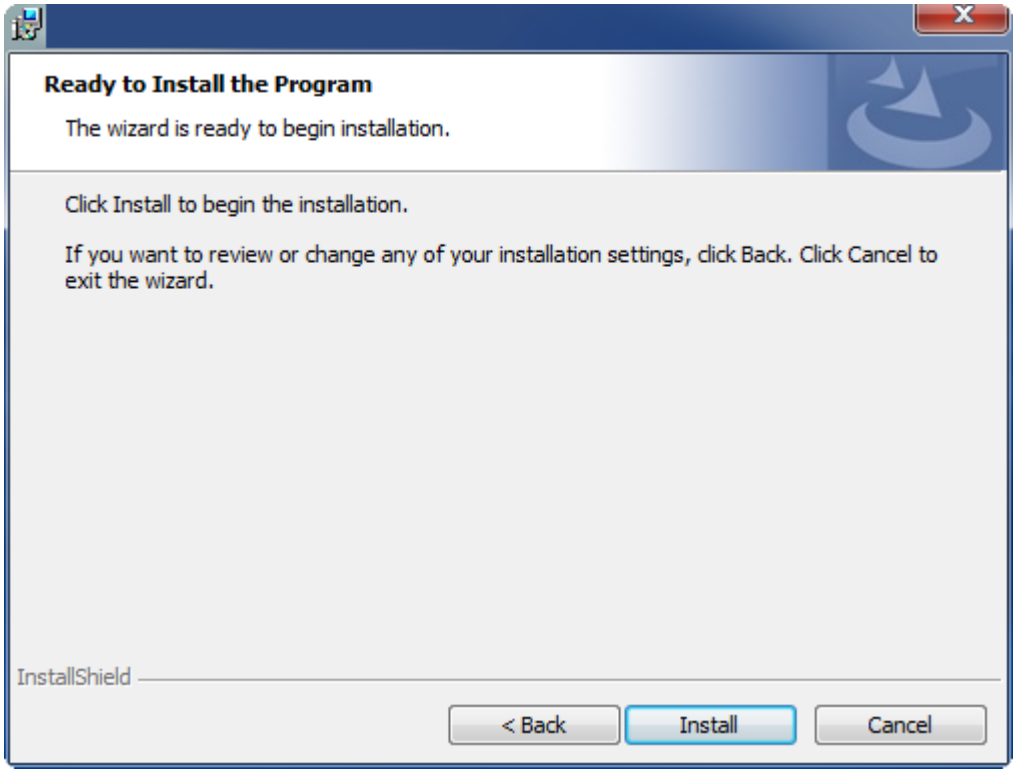
3. Geben Sie Ihre Benutzerdaten ein.



4. Wenn Sie die TwinCAT 3 Function vollständig installieren möchten, wählen Sie **Complete** als Installationstyp. Wenn Sie die Komponenten der TwinCAT 3 Function separat installieren möchten, wählen Sie **Custom**.

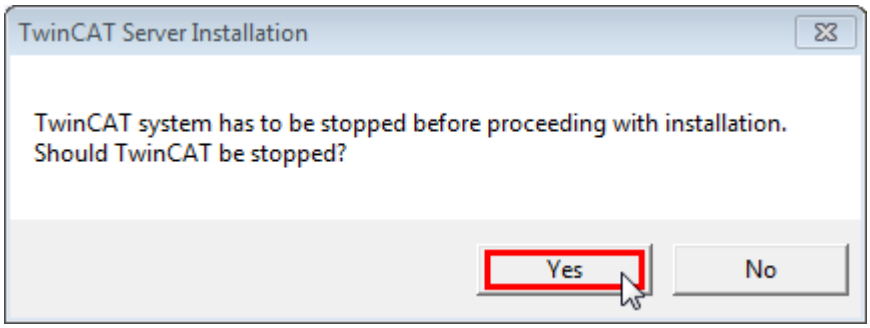


5. Wählen Sie **Next** und anschließend **Install**, um die Installation zu beginnen.

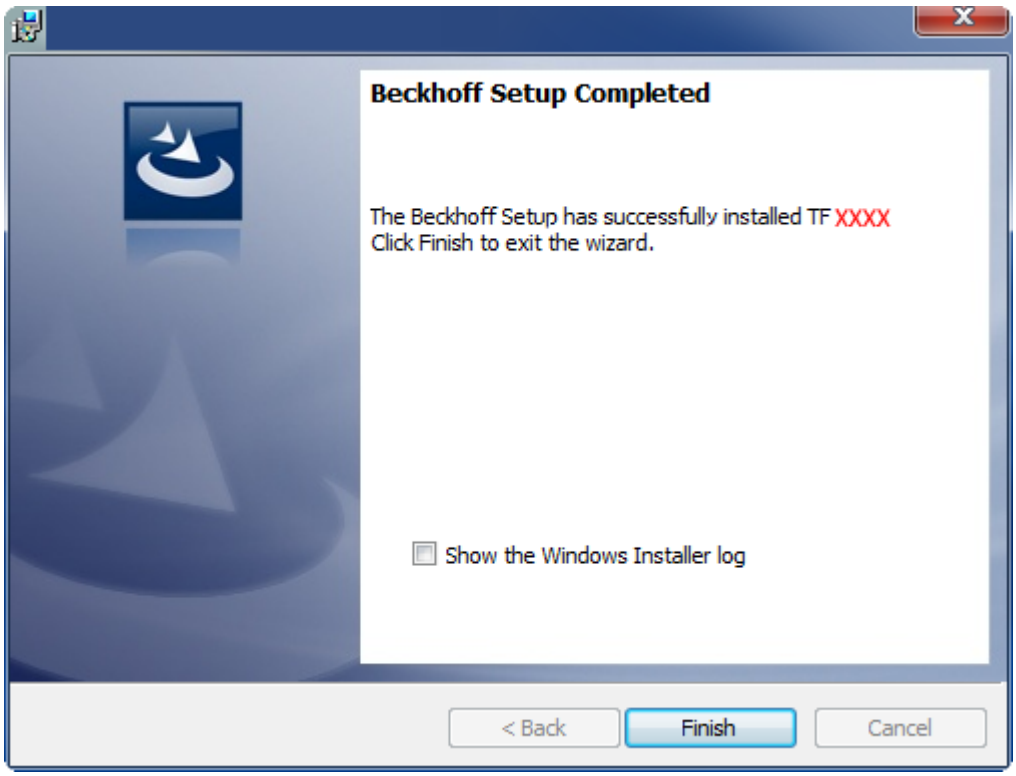


⇨ Ein Dialog weist Sie darauf hin, dass das TwinCAT-System für die weitere Installation gestoppt werden muss.

6. Bestätigen Sie den Dialog mit **Yes.**



7. Wählen Sie **Finish**, um das Setup zu beenden.



⇨ Die TwinCAT 3 Function wurde erfolgreich installiert.

2.4 Lizenzierung

Die TwinCAT 3 Function ist als Vollversion oder als 7-Tage-Testversion freischaltbar. Beide Lizenztypen sind über die TwinCAT-3-Entwicklungsumgebung (XAE) aktivierbar.

Lizenzierung der Vollversion einer TwinCAT 3 Function

Die Beschreibung der Lizenzierung einer Vollversion finden Sie im Beckhoff Information System in der Dokumentation „TwinCAT-3-Lizenzierung“.

Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

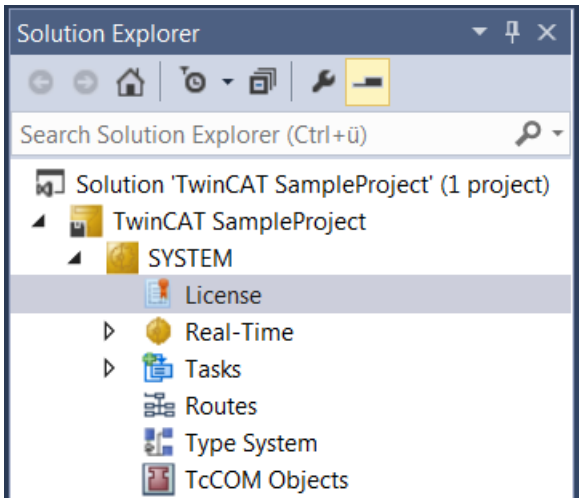
Eine 7-Tage-Testversion kann nicht für einen TwinCAT-3-Lizenz-Dongle freigeschaltet werden.

1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.

3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **Choose Target System** das Zielsystem aus.

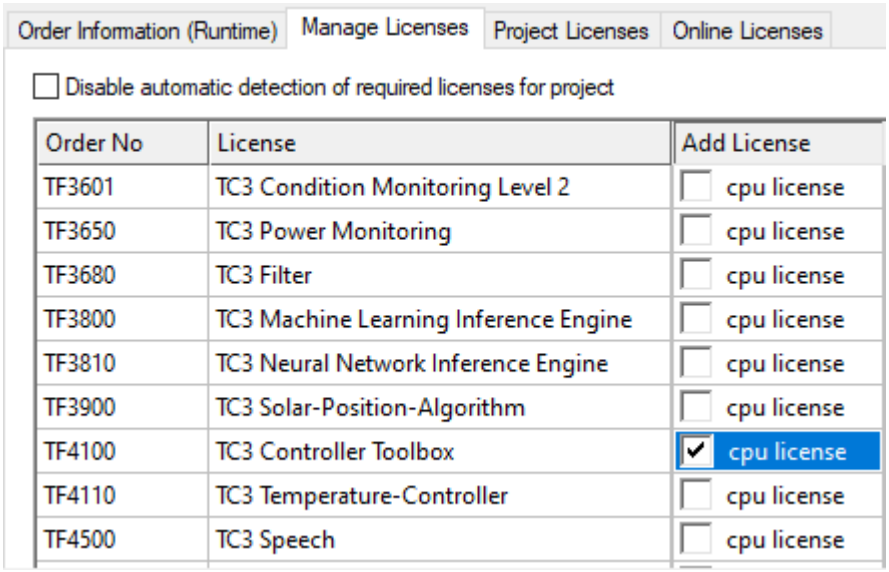
   ⇨ Die Lizenzierungseinstellungen beziehen sich immer auf das eingestellte Zielsystem. Mit der Aktivierung des Projekts auf dem Zielsystem werden automatisch auch die zugehörigen TwinCAT-3-Lizenzen auf dieses System kopiert.

4. Klicken Sie im **Solution Explorer** im Teilbaum **SYSTEM** doppelt auf **License.**



   ⇨ Der TwinCAT-3-Lizenzmanager öffnet sich.

5. Öffnen Sie die Registerkarte **Manage Licenses.** Aktivieren Sie in der Spalte **Add License** das Auswahlkästchen für die Lizenz, die Sie Ihrem Projekt hinzufügen möchten (z. B. „TF4100 TC3 Controller Toolbox“).

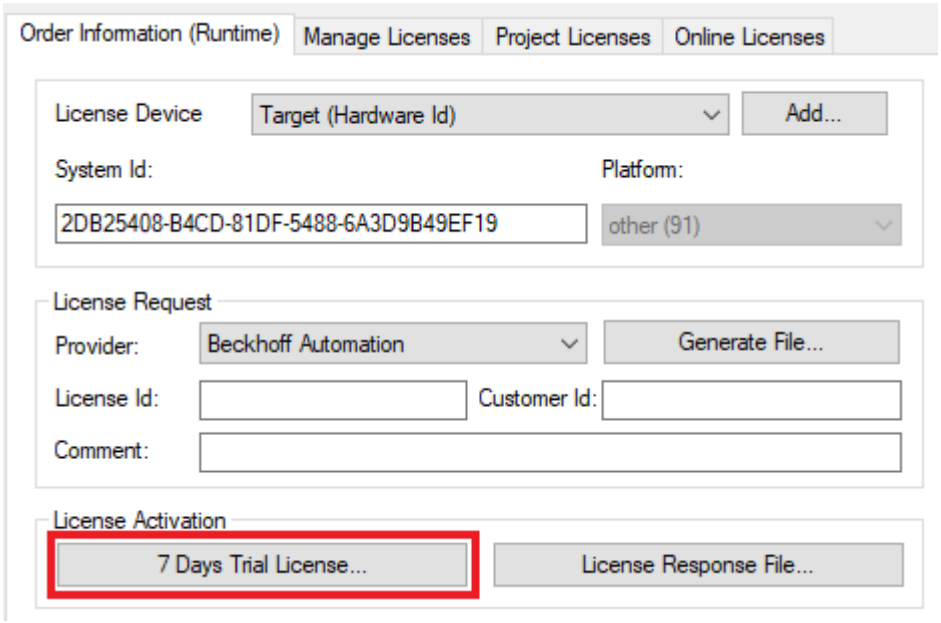


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu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

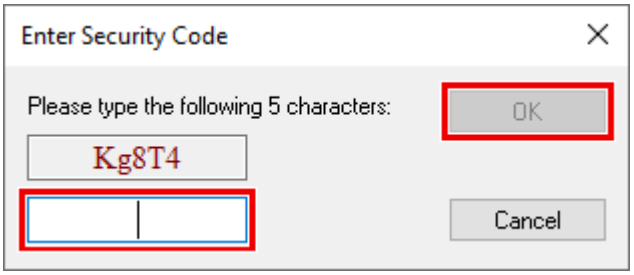
6. Öffnen Sie die Registerkarte **Order Information (Runtime).**

   ⇨ In der tabellarischen Übersicht der Lizenzen wird die zuvor ausgewählte Lizenz mit dem Status „missing“ angezeigt.

7. Klicken Sie auf **7 Days Trial License...**, um die 7-Tage-Testlizenz zu aktivieren.



⇨ Es öffnet sich ein Dialog, der Sie auffordert, den im Dialog angezeigten Sicherheitscode einzugeben.



8. Geben Sie den Code genauso ein, wie er angezeigt wird, und bestätigen Sie ihn.
9. Bestätigen Sie den nachfolgenden Dialog, der Sie auf die erfolgreiche Aktivierung hinweist.
   ⇨ In der tabellarischen Übersicht der Lizenzen gibt der Lizenzstatus nun das Ablaufdatum der Lizenz an.
10. Starten Sie das TwinCAT-System neu.

⇨ Die 7-Tage-Testversion ist freigeschaltet.

2.5 Systemvoraussetzungen

Voraussetzungen

TF8040 besteht aus verschiedenen Software-Komponenten, welche unterschiedliche Systemvoraussetzungen haben.

SPS-Bibliotheken

Die in TF8040 enthaltenen SPS-Bibliotheken benötigen die TwinCAT Version >= 3.1.4024.35 und laufen auf folgenden Betriebssystemen:

- Windows 10
- Windows CE

Für die Verwendung der SPS-Bibliotheken ist eine TF8020 Lizenz notwendig.

Tools

Die in TF8040 enthaltenen Tools laufen auf dem Betriebssystem **Windows 10**.

Für den SiteExplorer ist zusätzlich noch die .NET Desktop Runtime > v6.0 erforderlich.

TcHmiBa

Hier finden Sie die Systemvorrausetzungen für TcHmiBa [▶ 1140].

2.6 Energieeffizienz

Der Einfluss von Gebäudeautomation und Gebäudemanagement ist in der DIN EN ISO 52120-1:2019-12 beschrieben.

Diese DIN EN ISO 52120 beinhaltet eine strukturierte Liste von Funktionen der Gebäudeautomation und des technischen Gebäudemanagements, die Auswirkungen auf die Energieeffizienz von Gebäuden haben.

Die Funktionen der DIN EN ISO 52120 werden nach Gewerken und sogenannten Building Automation and Control (BAC)-Funktionen kategorisiert und strukturiert aufgelistet.

Ein faktorbasiertes Verfahren ermöglicht eine einfache Abschätzung der Auswirkungen dieser GA-Funktionen auf die Gesamtenergieeffizienz eines Gebäudes. Die Anwendung dieses vereinfachten Verfahrens zeigt, dass Energieeinsparungen von 30% durch die darin aufgelisteten Methoden möglich sind.

TwinCAT 3 Building Automation bietet Funktionen wie sie in der DIN EN ISO 52120 beschrieben sind.

Damit ist eine Abschätzung des Energieeinsparpotenzials durch den Einsatz von Anlagen- und Raumautomation mit TwinCAT 3 Building Automation möglich. Templates für Anlagen- und Raumfunktionen in TF8040 erleichtern die Planung, Ausführung und den späteren Betrieb von Gebäuden.

Eine wesentliche Voraussetzung für die Realisierung eines energieeffizienten und nachhaltigen Gebäudeautomationssystems ist die Durchgängigkeit des Gebäudeautomationssystems. Anlagen- und Raumautomation mit allen seinen Ausbaugewerken sind in ein System zu integrieren. Die Templates aus TF8040 TwinCAT 3 Building Automation realisieren die in der Norm DIN EN ISO 52120 beschriebenen Energieeffizienzfunktionen.

3 Konzepte

In diesem Kapitel befinden sich einleitende Informationen, die den Einstieg in TwinCAT 3 Building Automation erleichtern. Konzepte dienen als Entwürfe und Vorschläge zur Projektierung eines Gebäudeautomationssystems mit TwinCAT 3 Building Automation.

3.1 Benutzerrollen

Das Grundgerüst der SPS kennt keine Benutzer, sondern nur Rollen. Daher geht die weitere Beschreibung auf die Bedeutung der Rollen ein.
Eine Anwendung (z. B. TcHmiBa [▶ 1139]) definiert Benutzer und weist diesen dann eine entsprechende Rolle zu.
Entsprechend der Rolle eines angemeldeten Benutzers werden in der Anwendung Funktionen verfügbar gemacht oder nicht.

Rollen

Für unterschiedliche Anwender sind verschiedene Zugriffsrechte vorgesehen:

Rolle	Beschreibung
Guest	Geringste Zugriffsberechtigungen. Nutzer können keine Parameter ändern und nur aktuelle Werte, die in der Tc3_XBA [▶ 101] unter VAR_INPUT CONSTANT PERSISTENT stehen, lesen. Empfohlen für Standard-Zugänge ohne Benutzer-Anmeldung (z. B. allgemein zugängliche Bedien-Panels).
Basic	Eingeschränkte Zugriffsberechtigungen. Nutzer können rudimentäre Parameter einsehen und kaum Werte ändern. Empfohlen für Bediener mit geringen Anlagenkenntnissen.
Advanced	Erweiterte Zugriffsberechtigungen. Nutzer haben Einsicht in verschiedene Parameter und z.B. Berechtigung zum Ändern von Sollwerten oder Zeitschaltprogrammen. Empfohlen für Bediener mit grundlegenden Anlagen-Kenntnissen und Anweisung zur Betreuung dieser Anlagen.
Expert	Vollständige Zugriffsberechtigungen. Empfohlen für Inbetriebnahme- und Service-Personal, da auch tiefergehende Eingriffe (z. B. Anpassen von Regler-Parametern) möglich sind.
Internal	Nur für den Beckhoff-Support.



Der Zugriffsbereich des Benutzers wird an unterschiedlichen Stellen in der Anwendung ausgewertet, um bestimmte Funktionen freizugeben bzw. zu verbergen.

Anpassung der Zugriffsrechte aus der PLC heraus

Es gibt die Möglichkeit, Zugriffsrechte aus der PLC heraus zu definieren.

Dies allerdings geht nur initial, also nicht zur Laufzeit, und bezieht sich dann auf alle Objekte, die dieses Property haben.

Dazu wird in einem geeigneten Programmteil, vorzugsweise im MAIN, eine Instanz des FB_BA_Param [▶ 163] angelegt und mit den entsprechenden Schreib- und Leserechten vorinitialisiert.



Die Initialisierung reicht dabei aus - ein weiterer Aufruf im eigentlichen Programmteil erfolgt nicht.

Beispiel:

Im folgenden Beispiel sollen an zwei Parametern die Lese- und Schreibrechte verändert werden:

- Acknowledge (*bAcknowledgeRm*): Lesen ab Rolle Basic, Schreiben ab Rolle Expert.
- DefaultValue (*bDefaultValue*, *fDefaultValue*, *nDefaultValue*): Lesen ab Rolle Internal, Schreiben ab Rolle Advanced.

Diese Anpassung erfordert die Instanziierung eines FB_BA_Param [▶ 163], hier *Parameters* genannt, die sich wie folgt gestaltet:

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

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

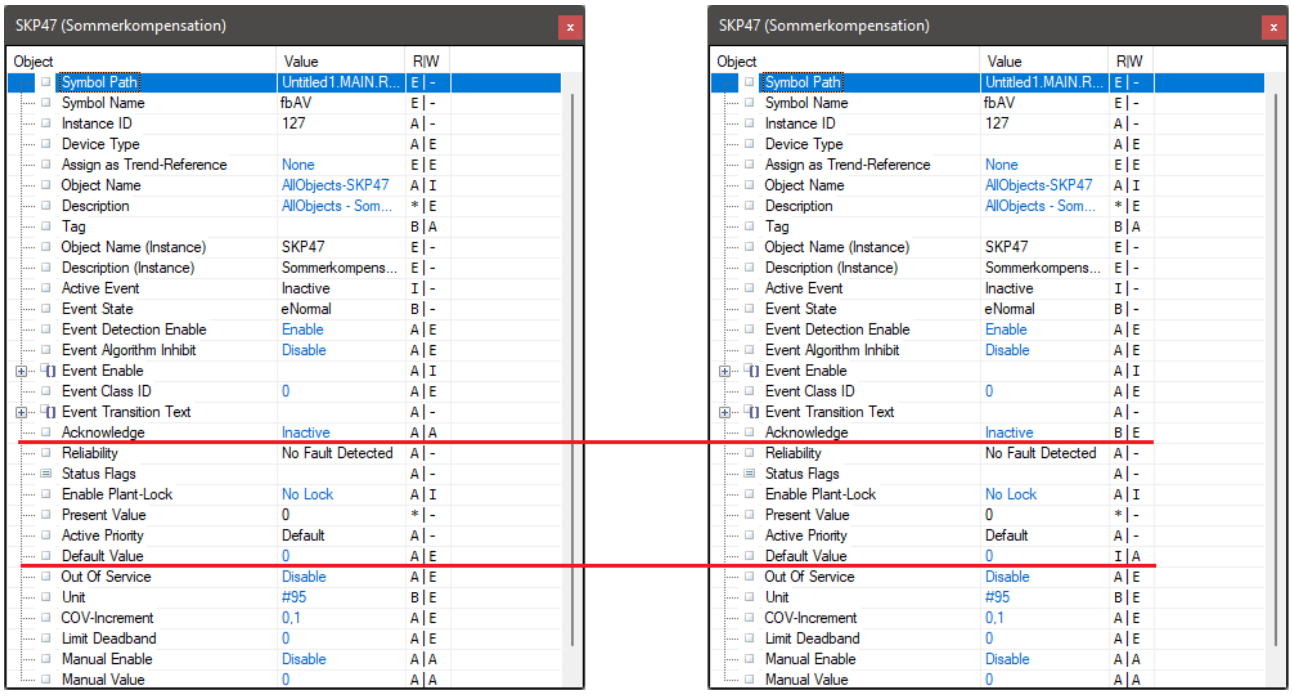
Die Properties *DefReadAccess* und *DefWriteAccess*, stellen jeweils ein Array der verschiedenen Parameter mit den dazugehörigen Rollen als Mindestzugriff dar und dienen nur zur Veränderung der Rollen, welche allerdings schon vordefiniert sind.

Die Set-Methoden verändern für einen bestimmten Parameter (E_BA_Parameter [▶ 110]) die Mindest-Zugriffsrolle (E_BA_Role [▶ 101]). Für jeden Parameter, dessen Zugriffsrecht verändert werden soll, ist jeweils eine Set-Methode hinzuzufügen.

Abgeschlossen wird die Parametrierung der Properties jeweils mit einem Aufruf der Methode *Build*. Diese gilt sowohl für das Property des Lesezugriffs *DefReadAccess*, als auch für das Property des Schreibzugriffs *DefWriteAccess*.

Die oben aufgeführte Implementierung verändert den Lese- und Schreibzugriff **aller** Objekte, die die Properties *bAcknowledgeRm* und DefaultValue (*bDefaultValue*, *fDefaultValue*, *nDefaultValue*) besitzen. Es ist auch möglich nur den Lese- oder nur den Schreibzugriff zu ändern, die Änderung muss keineswegs paarweise geschehen.

Am Beispiel eines AV-Objektes stellt sich die Änderung im Site-Explorer wie folgt dar:



3.2 Kommunikation

Die Kommunikationsprotokolle OPC UA; Modbus TCP; MQTT; und ADS dienen alle gemeinsam der Übertragung von Informationen, welche sich auf die Werte von Symbolen und Strukturen der SPS beziehen.

Sie beschreiben im Gegensatz zu BACnet keine standardisierten Kommunikationsobjekte, sodass die Struktur der Daten in vielen Projekten sehr spezifisch und individuell ist. Unabhängig von BACnet ist TwinCAT TF8040 auch für solche Projekte sehr hilfreich, da die Basisbibliothek Tc3_BA2 [▶ 272] eine Vielzahl von Funktionsbausteinen beinhaltet, welche unabhängig von dem verwendeten Kommunikationsprotokoll für ein Gebäudeautomationsprojekt benötigt werden.

Weltweit hat sich das Kommunikationsprotokoll BACnet als Standard in der Gebäudeautomation etabliert. Mit seinen Objekten und Diensten gibt BACnet im Gegensatz zu den anderen Protokollen einen sehr detaillierten Standard vor.

TwinCAT Building Automation nutzt die objektorientierte Struktur des BACnet.

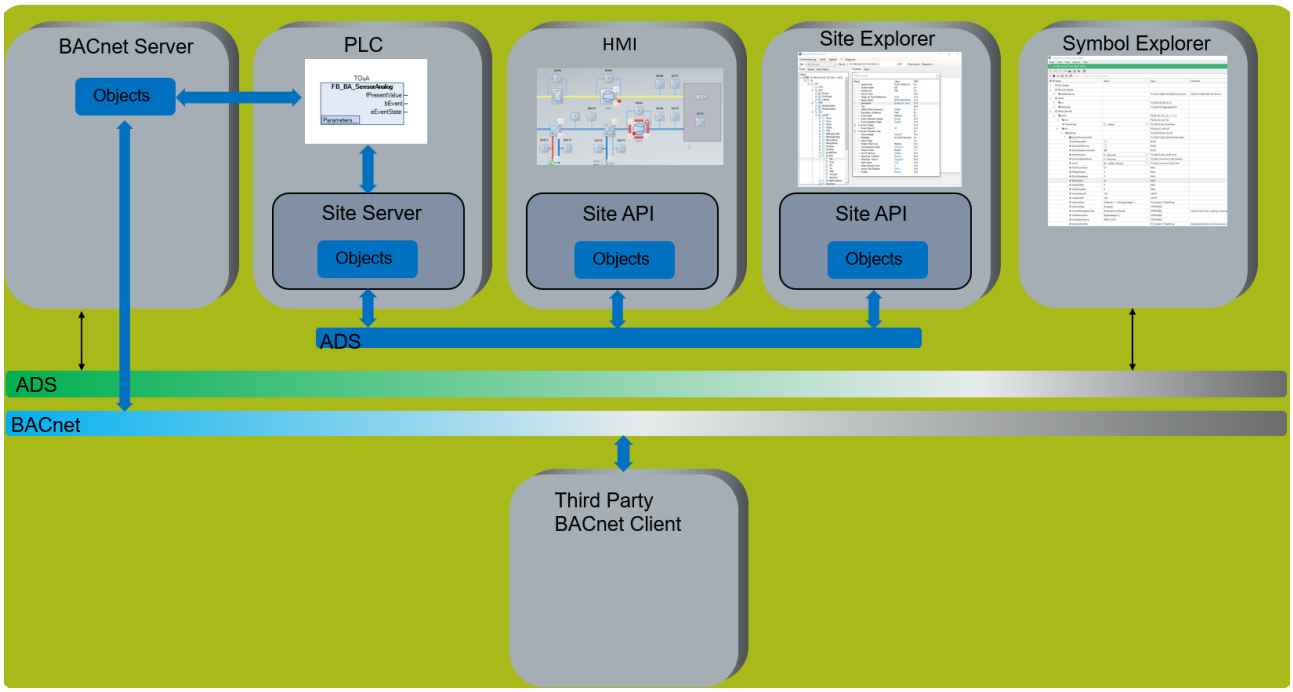
Die Bibliothek Tc3_XBA [▶ 101] beinhaltet ein Framework namens Grundgerüst [▶ 31].

Es dient der Abbildung einer Projektstruktur [▶ 39] und zahlreicher weiterer generischer Funktionen.

Der Site Server innerhalb der SPS organisiert die ADS-Kommunikation zur TwinCAT HMI.

Die Site API ist das Gegenstück des Site Severs und realisiert die generischen Darstellungen in der HMI.

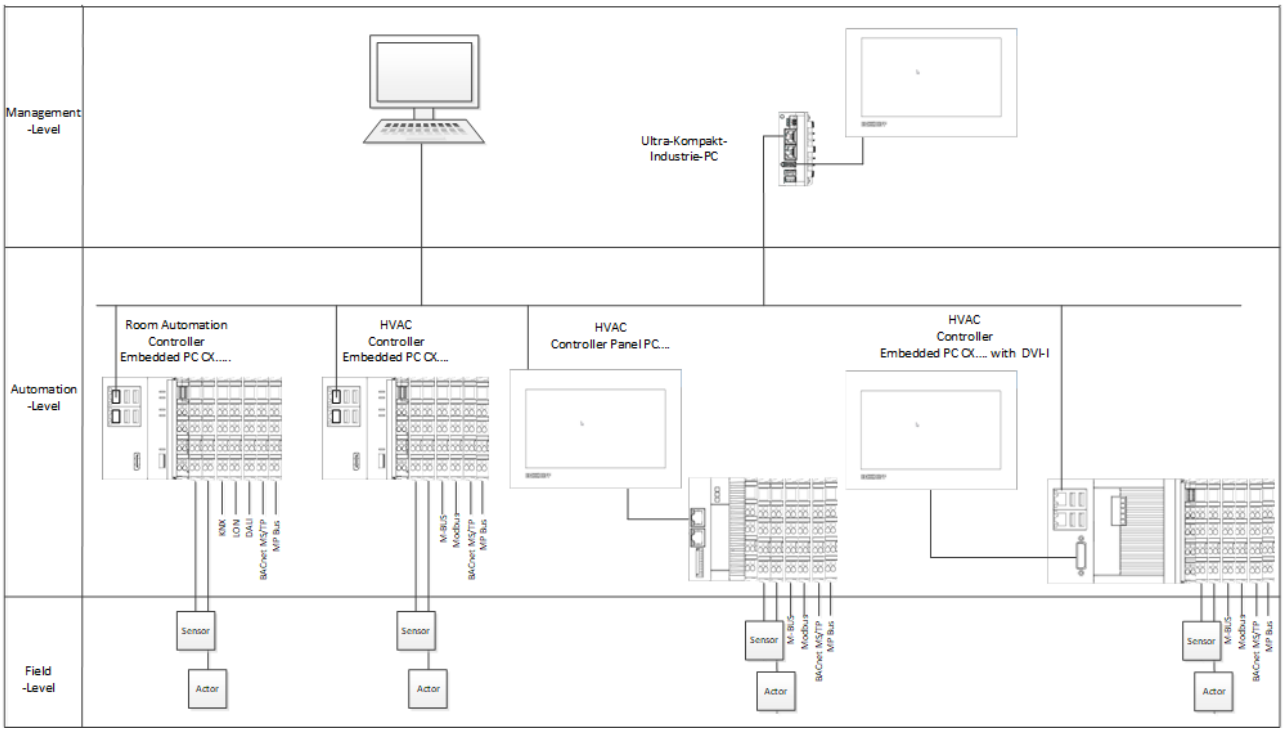
Das Service Tool Site Explorer kommuniziert ebenfalls mit der Site API via ADS mit den Automationsstationen.



3.3 Systemaufbau

Die Beckhoff Industrie PCs erfüllen alle Anforderungen hinsichtlich der Automation von Heizungs- Lüftungs- und Klimatechnik als auch der Raumautomation [▶ 21].

Durch das große Portfolio der Beckhoff Industrie PC ergibt sich eine hervorragende Skalierbarkeit des Gebäudeautomationssystems. Der hohe industrielle Qualitätsstandard gewährt eine hohe Anlagenverfügbarkeit und Investitionssicherheit.



Alle PCs werden unabhängig ihres Einsatzes mit TwinCAT Engineering TE1000 programmiert.

Durch den Einsatz der TwinCAT Funktionen TF8020 BACnet und TF8040 Building Automation werden alle Industrie PCs des Beckhoff Portfolios zu einer leistungsfähigen Automationsstation der Gebäudeautomation. (Für die Kleinsteuerungen der Baureihe CX7xxx ist die TwinCAT Funktion TF8040 derzeit nicht verfügbar).

Durch die Auswahl des Prozessors kann die Automationsstation hinsichtlich ihrer Rechenleistung den Anforderungen der zu projektierenden Automationsstation fein granular angepasst werden.

Die Bauform des Industrie-PC ist ebenfalls frei wählbar.

Hervorragend geeignet sind die Embedded PCs der Baureihe CX.

An dem Embedded PC können auf der rechten Seite Busklemmen (K-Bus) oder EtherCAT-Klemmen (E-Bus) angereiht werden.

Je nach Ausführung des Embedded PC kann dieser mit einer DVI-I Schnittstelle bestückt sein.

An diese kann bei Bedarf ein Control Panel angeschlossen werden. Somit kann der Industrie PC als Automationsstation eines Schaltschrankes mit einer lokalen Bedienung via Touch Panel dienen. Auf dem IPC ist dann die Funktion TF2000 HMI Server zu installieren.

Für übergeordneten Aufgaben innerhalb des Automationssystems wie z.B. der MBE (Management- und Bedieneinrichtung) stehen sehr leistungsfähige Ultra kompakte PCs zur Verfügung.

Die TwinCAT HMI kann so nicht nur als lokale Bedienung eines IPC, sondern als Web-HMI sehr vieler Automationsstationen agieren.

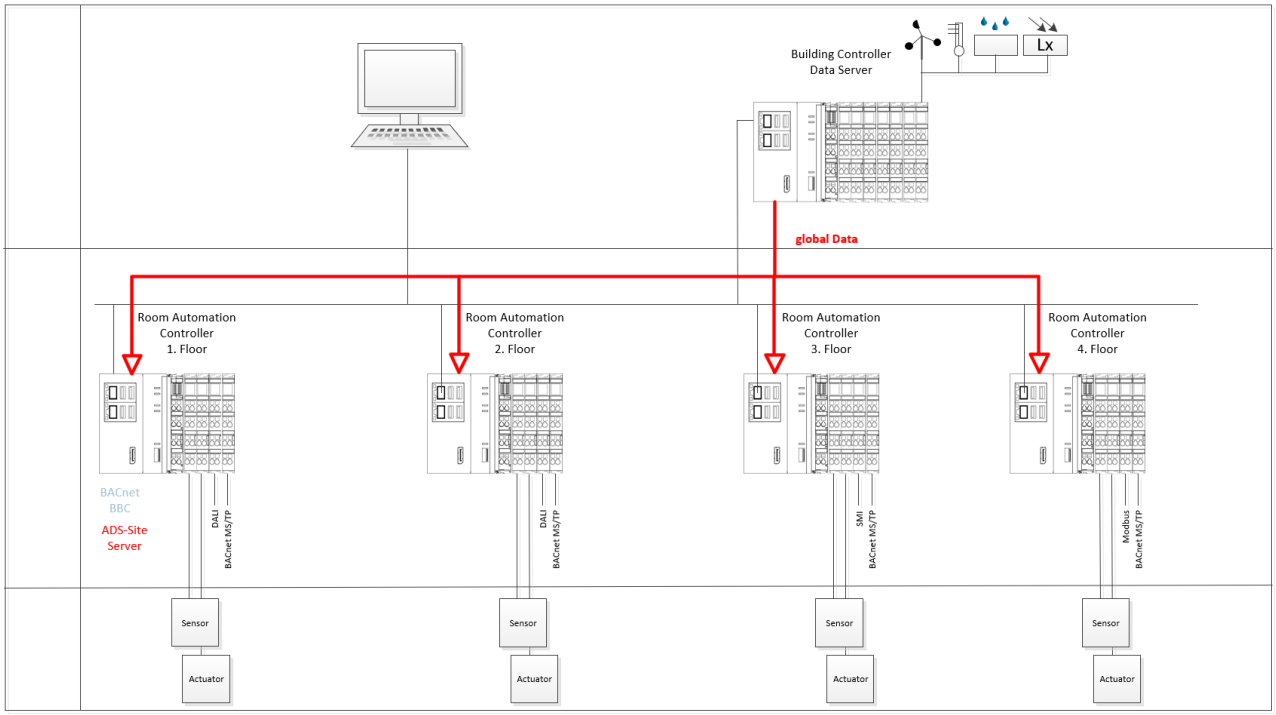
Unabhängig davon, ob es sich um Anlagenautomation oder Raumautomation handelt, wird durch alle Anwendungsbereiche die gleiche Hard- und Software verwendet. Eine Interoperabilität aller Komponenten der Gebäudeautomation ist deshalb systembedingt gegeben.

3.4 ADS-Kommunikation innerhalb der Templates

TwinCAT Building Automation stellt ein einfaches Konzept zur Verteilung von Daten innerhalb eines Gebäudeautomationsnetzwerkes zur Verfügung.

Der Building Controller dient innerhalb diese Konzeptes als zentraler Verteiler von Daten.

Er sollte aus einem Leistungsfähigen IPC bestehen.



Die folgenden Daten werden innerhalb des Gebäudecontrollers erfasst, berechnet und bereitgestellt.

Wetterdaten

- Außentemperatur
- Niederschlagsmeldung
- Taupunkttemperatur
- Luftdruck
- Luftfeuchte
- Helligkeit
- Globalstrahlung
- Windgeschwindigkeit
- Windrichtung

globale Ereignisse

- Einbruch
- Brand

globale Einstellparameter

- Frostsollwert
- Frostschutzgrenzwert
- gedämpfter Wert der Außentemperatur
- Heizfreigabe der ungedämpften Außentemperatur
- Heizfreigabe der gedämpften Außentemperatur

globaler Zeitschaltplan mit den aktiven Energieniveau

- Protection oder Economy oder PreComfort oder Comfort

Sollwerte der Energieniveaus

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

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

Betriebsarten

- Default (standard)
- Nachtwächterrundgang
- Reinigungsbetrieb

Sonnenschutz für das gesamte Gebäude

- Resultierendes Positioniertelegramm aus den Anwenderfunktionen: Feuer, Einbruch, Vereisung,
  - Position
  - Winkel
  - Priorität
- Freigabe der Thermoautomatik (globalstrahlungsabhängig)
- Freigabe der Dämmerungsautomatik (außenlichtabhängig)
- Zentrales Rücksetzen des Manual-Betriebs aller Jalousieaktuatoren

Sonnenschutz selektiv jeweils pro Fassade

- Resultierendes Positioniertelegramm aus den Anwenderfunktionen:
  - Position
  - Sturmschutz (Windrichtungsabhängig)
  - Wartung
  - Thermoautomatik (global)
  - Position bzw. Lamellenwinkel bei aktivem Sonnenschutz
  - Freigabe Sonnenschutz Außenlicht- und Sonnenstandsabhängig

Im Gebäudecontroller schreiben alle datenbereitstellenden Templates ihre Daten in die Site-GVL [▶ 1135].

Aus diesem Datenpool der Site-GVL [▶ 1135] werden die Daten via ADS an die Clients übertragen.

Die Daten sind nach Gewerken aufgeteilt.

Gebäudeautomation allgemein:

Im Funktionsbaustein *FB_BA_BuildingAutomationServer* werden, Wetterdaten, globale Ereignisse, globale Einstellparameter für HLK-Anlagen erzeugt und die die Site-GVL [▶ 1135] hineingeschrieben.

Der Funktionsbaustein *FB_BA_AdsComServer* published diese Daten via ADS.

In einem Client werden dieser Daten wiederum mittels Subscriber empfangen und in die lokale Site-GVL [▶ 1135] der Automationsstation hineingeschrieben.

Türen,Tore,Fenster,Sonnenschutz

Im Funktionsbaustein *FB_BA_DGWSPServer* werden, Fassadendaten und globale Daten des Sonnenschutzes erzeugt und die die Site-GVL [▶ 1135] hineingeschrieben.

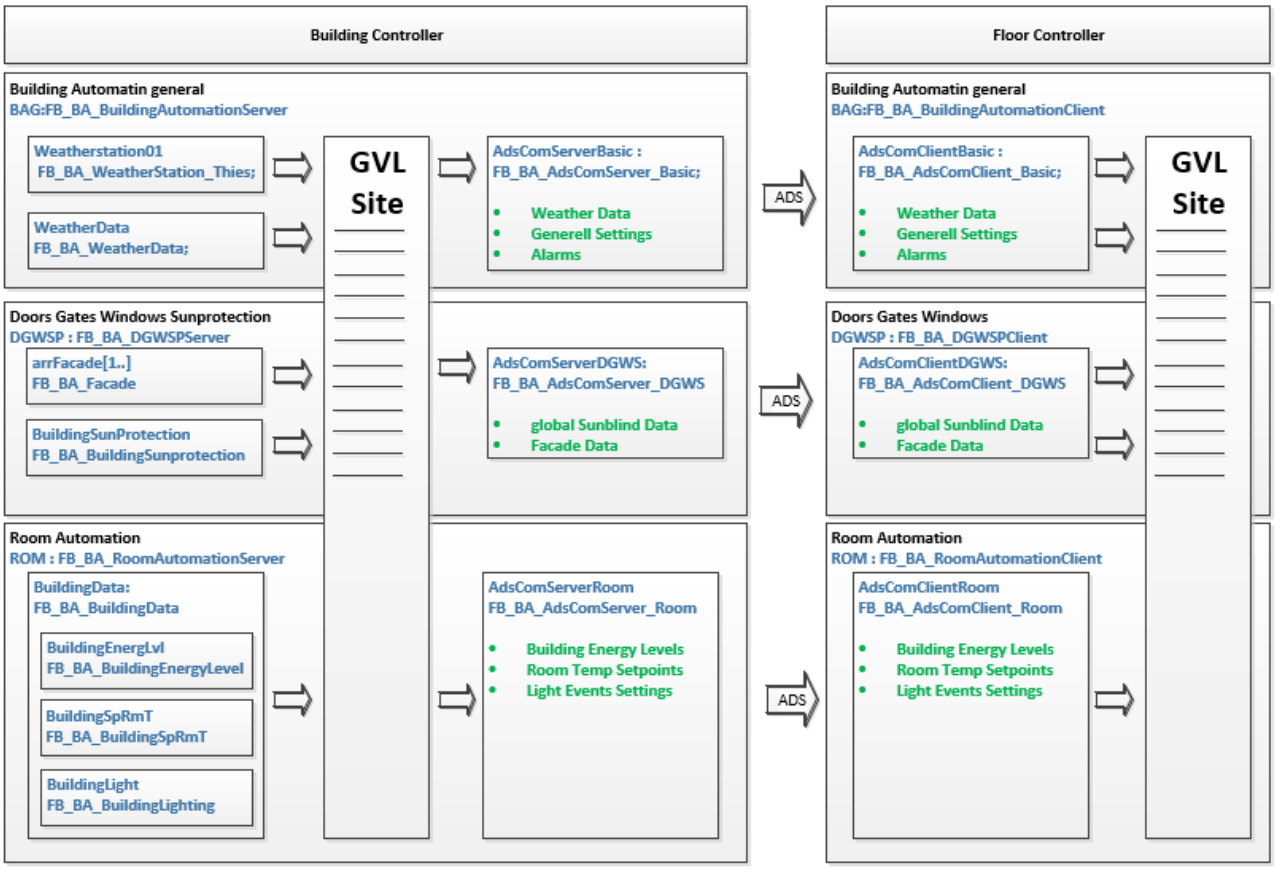
Der Funktionsbaustein *FB_BA_AdsComServer_DGWS* published diese Daten via ADS.

In einem Client werden dieser Daten wiederum mittels Subscriber empfangen und in die lokale Site-GVL [▶ 1135] der Automationsstation hineingeschrieben.

Raumautomation

Im Funktionsbaustein *FB_BA_RoomAutomationServer* werden, Raumautomationsdaten z.B. globale Raumtemperatursollwerte und Energieniveaus erzeugt und die die Site-GVL [▶ 1135] hineingeschrieben.

Der Funktionsbaustein *FB_BA_RoomAutomationClient* empfängt diese Daten und schreibt sie wiederum in die lokale Site-GVL [▶ 1135] der Automationsstation hinein.



3.5 Raumautomation

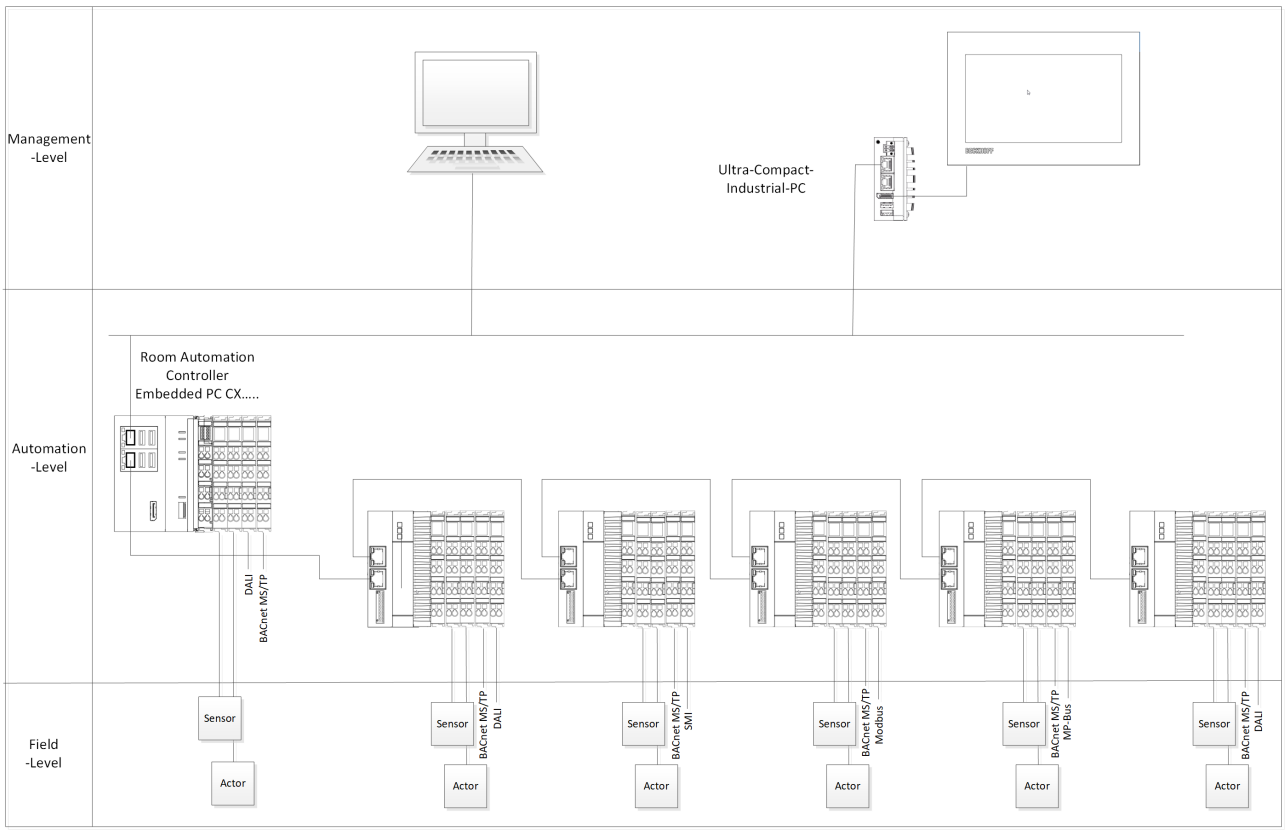
Die Raumautomation integriert sich in die Automationsebene.

Die Automationsstationen der Raumautomation in TwinCAT Building Automation bestehen ebenfalls aus Beckhoff IPCs.

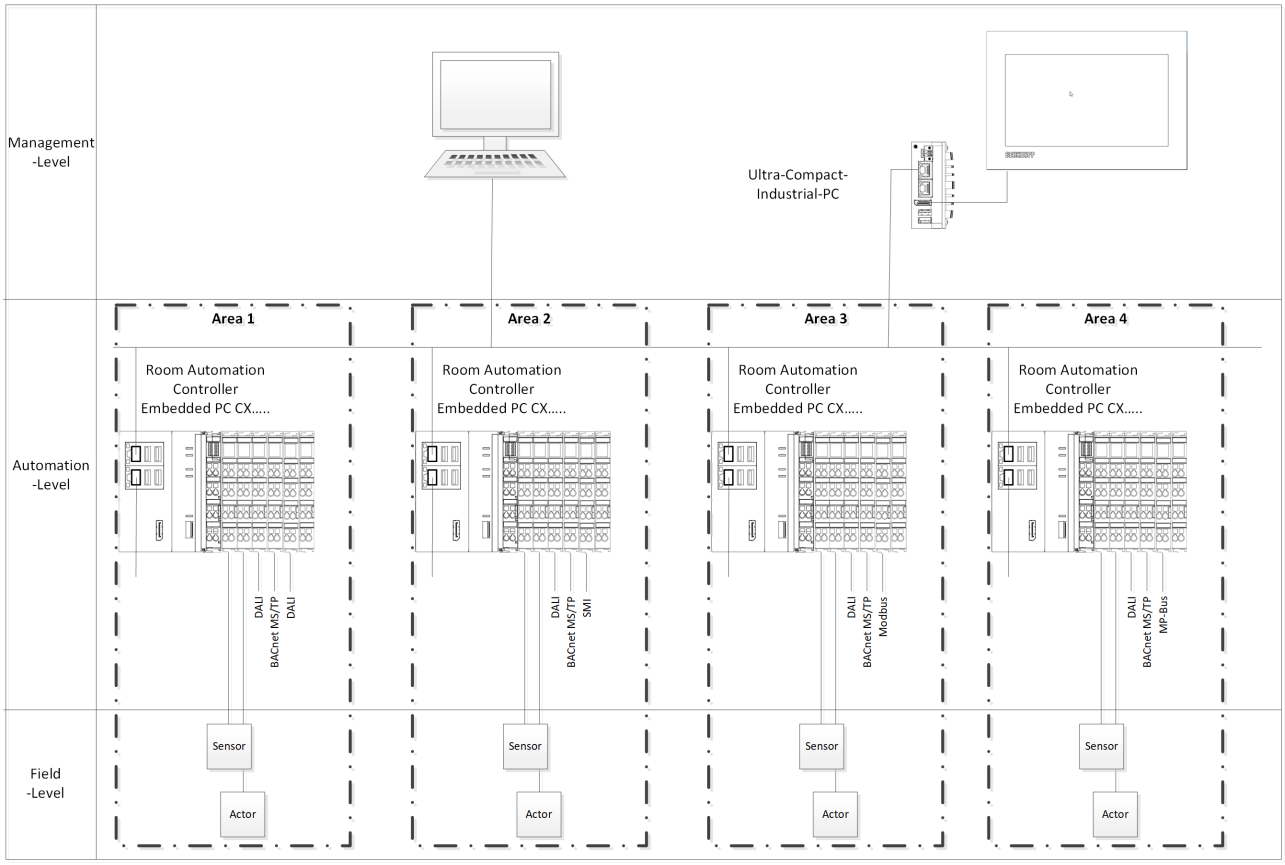
Die Auswahl und die Zahl der Beckhoff IPCs hängt stark von den Anforderungen an die Gebäudeautomation ab.

Aufgrund der außerordentlich hohen Rechenleistung der IPCs ist es möglich, sehr große Bereiche bis hin zu einer ganzen Etage, mit einem einzigen IPC zu automatisieren. Die Erfassung der Sensoren und Aktoren in der Feldebene kann durch räumlich abgesetzte Feldbuskoppler erfolgen. So kann der Verkabelungsaufwand und ebenfalls die Brandlasten reduziert werden.

Auch die Subsysteme wie z. B. DALI für die Beleuchtung, können an den Feldbuskopplern integriert und in der Etage verteilt werden.



Falls erforderlich kann eine Etage in mehrere Bereiche unterteilt werden. Die Bereiche werden dann jeweils von einem kleineren separaten IPC gesteuert.



3.5.1 Schalenmodell

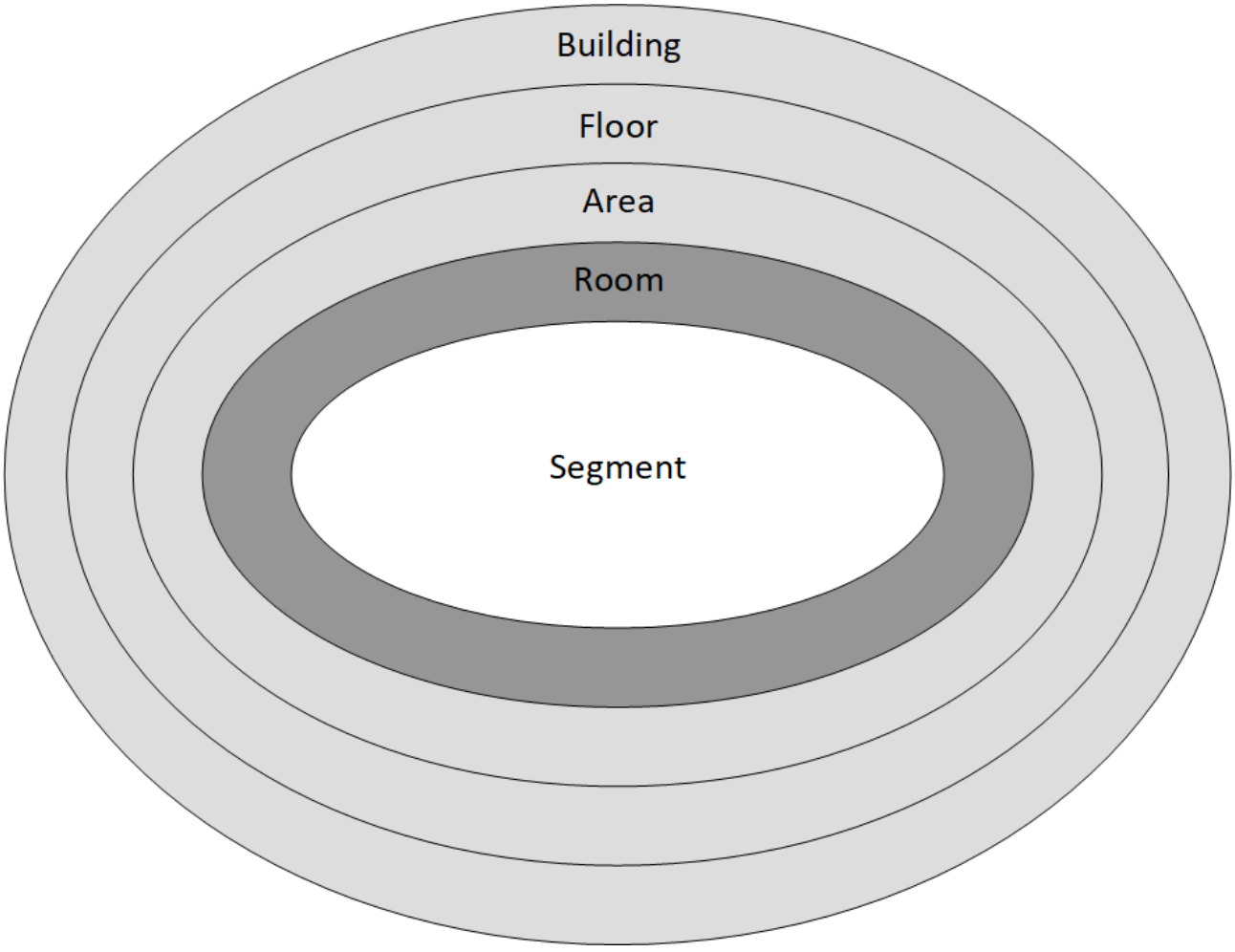
Zur Erläuterung der Raumautomation in TF8040 wird das Schalenmodel der VDI 3813 verwendet. Das Schalenmodel beschreibt Ebenen innerhalb derer die Raumfunktionen [▶ 23], bestehend aus Sensor-, -Anwender oder Aktuator- Funktionen eingesetzt werden.

Die Anwender- und Sensorfunktionen einer Schale wirken auf die Aktoren aller eingeschlossenen Schalen. Die Position einer Anwenderfunktion im Schalenmodell bestimmt also die Teilmenge der Aktoren, auf die diese Anwenderfunktion wirkt.

Beispiel:

Ein Zeitschaltplan zur zentralen Ansteuerung aller Leuchten einer Etage, z. B. für den Reinigungsbetrieb eines Büros, ist in der Schale Etage positioniert.

Sollte der Reinigungsbetrieb in den Unterbereichen der Etage separat geplant werden müssen, zum Beispiel wenn es einzelne Mieterbereiche gibt, in denen der Reinigungsbetrieb separat durchgeführt wird, so muss jeweils pro Bereich ein Zeitschaltplan platziert werden.



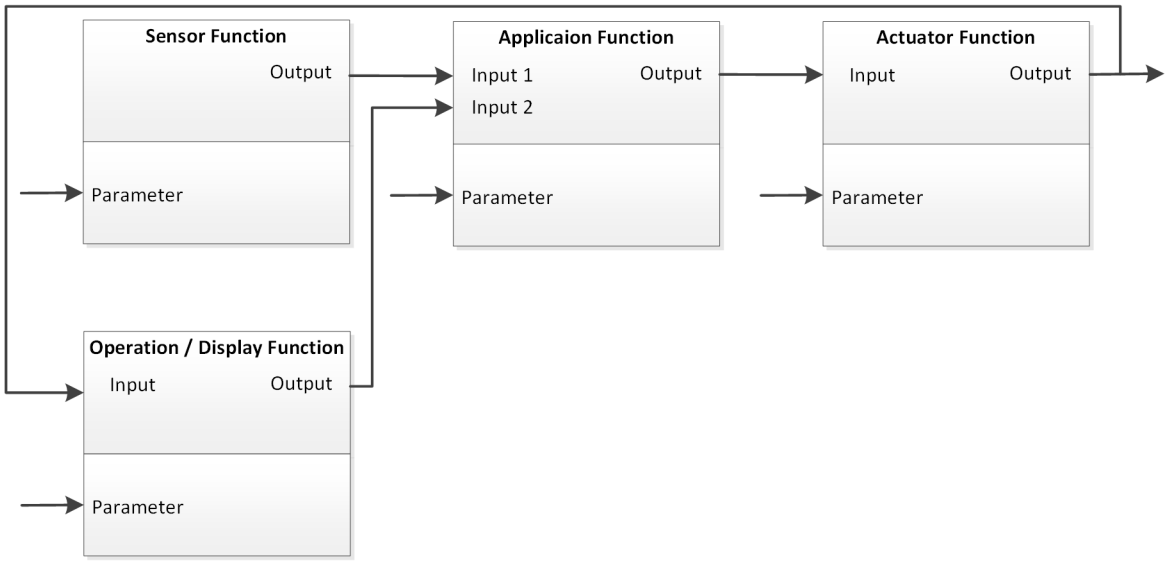
3.5.2 Raumfunktionen

Die Funktionen eines Raums unterteilen sich in Sensor-, Anwendungs- und Aktor-Funktionen.

Die Signale der Sensoren und Aktoren werden entweder unmittelbar durch eine Klemme erfasst bzw. ausgegeben oder mittels eines Feldbussystems integriert. Die Schnittstelle bzw. die Kommunikation der Sensor- und Aktor-Funktionen mit den Anwendungsfunktionen des Raums ist dabei stets identisch. Die Raumanwendungsfunktionen können deshalb unabhängig von dem gewählten Feldbussystem gleichbleiben.

Bei der Verwendung von Feldbussystemen der Raumautomation werden die Messsignale der Sensoren bzw. die Stellbefehle der Aktoren durch die Einbindung der entsprechenden Feldbusklemme integriert.

Raumautomationsfunktionen nach VDI 3813:



Die Sensorfunktionen des Raums dienen nicht nur den Anwendungsfunktionen eines Gewerkes, sondern werden von den Anwendungsfunktionen der Beleuchtung, dem Sonnenschutz und der Raum-Klima-Regelung verarbeitet. Besonders repräsentativ ist hierfür die Präsenzerfassung eines Raums, welche der Automation der drei zuvor erwähnten Gewerke dient.

Die Art der Anwendungsfunktionen für die Gewerke Beleuchtung und Sonnenschutz hängen stark von der Nutzung des Raums ab. Diese sollten bereits im frühen Stadium der Planung durch dem Fachplaner für Gebäudeautomation gemeinsam mit dem Betreiber des Gebäudes spezifiziert werden.

3.5.2.1 Beleuchtung

Die Lichtfunktionen innerhalb von TF8040 orientieren sich an dem Schalenmodell [▶ 23] aus der Norm für Raumautomation VDI 3813.

In jeder Schale des Modells für die Lichtsteuerung kann es eine oder mehrere Raumfunktionen [▶ 23] geben. Eine Anwenderfunktion des Lichts sendet ein Telegramm zur Ansteuerung der Leuchten.

```
TYPE ST_BA_Lighting :
STRUCT
  {attribute 'parameterUnit':= '%'}
  fLgtVal    : REAL
  {attribute 'parameterUnit':= 'K'}
  fLgtT      : REAL;

  bActv      : BOOL;
  ePrio      : E_BA_LightingPrio;

  nEvtInc    : ULINT;
END_STRUCT
END_TYPE
```

Innerhalb des Telegramms wird ein Stellwert für die Lichtstärke in % und die Lichttemperatur in Kelvin übertragen. Ebenfalls sind in dem Telegramm die Variablen *bActv* und *ePrio*.
Um zu entscheiden welche der Funktionen in einer Schale Vorrang, hat bekommt jedes Lichttelegramm eine Priorität *ePrio*.
Ein Telegramm-Selektor vom Typ FB_BA_LightingTgmSel4 [▶ 317] /FB_BA_LightingTgmSel8 [▶ 317] entscheidet bei mehreren Lichtfunktionen in einer Schale welches der Telegramme an die nächstinnere Schale hindurchgeleitet wird.

In den Schalen Gebäude, Etage und Bereich, werden globale Lichtfunktionen [▶ 25] mit hohen Prioritäten positioniert.

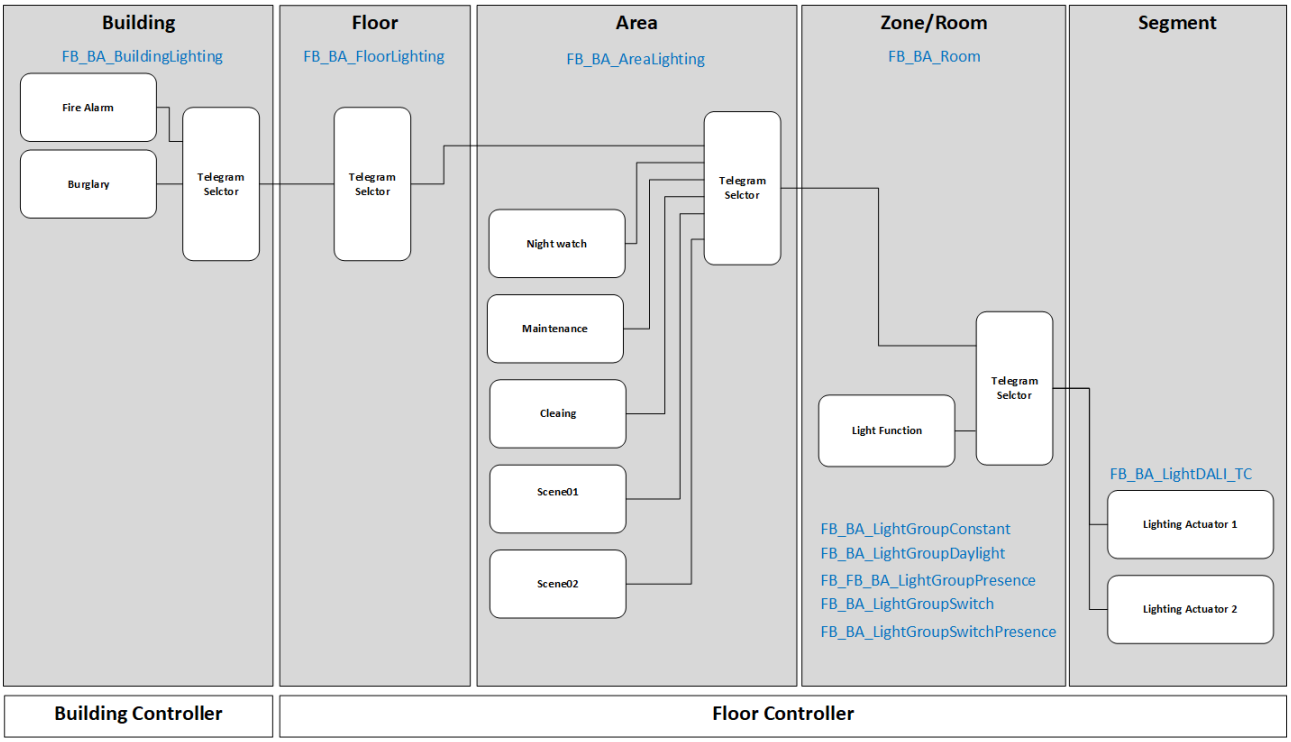
Diese Lichtfunktionen sind innerhalb der Templates FB_BA_BuildingLighting [▶ 933] und FB_BA_FloorLighting [▶ 935] realisiert.

Globale Lichtfunktionen [▶ 25]

In der Schale Zone/Raum (lokale Lichtfunktionen [▶ 26]) wird die jeweilige Anwendungsfunktion ausgewählt, welche den Anforderungen an die Beleuchtung des Raums entspricht. Wenn sich mehrere Lichtgruppen in einem Raum befinden, wird für jede eine separate Anwendungsfunktion platziert.

Die Sensor- und Anwenderfunktionen der Schale Gebäude befinden sich in der Automationsstation Building Controller. Die der Funktionen der Schalen Etage, bis Segment befinden sich in dem Floor Controller. Weitere Details hierzu befinden sich im Kapitel ADS-Kommunikation innerhalb der Templates [▶ 18]

Durch die Auswahl und Position der Lichtfunktionen im Schalenmodell wird die Gesamtfunktionalität der Lichtsteuerung bestimmt. Das Beleuchtungskonzept ist sehr individuell und stark von der Nutzung des Gebäudes abhängig.



3.5.2.1.1 Globale Lichtfunktionen

Zusätzlich zu den Funktionen innerhalb des Raums werden Leuchten durch übergeordnete Anwendungsfunktionen gesteuert. Diese sind den Anforderungen an den Betrieb des Gebäudes anzupassen und in der Planungsphase zu bestimmen.

Mögliche globale Beleuchtungsfunktionen können sein:

- **Einbruch**
  Durch eine Verbindung des GA-Systems mit der Einbruchmeldeanlage wird ein globaler Befehl ausgelöst, welcher im gesamten Gebäude die Beleuchtung einschaltet.
  Die Realisierung dieser Lichtfunktion befindet sich in dem Template FB_BA_BuildingLighting [▶ 933].
- **Feuer**
  Durch eine Verbindung des GA-Systems mit der Brandmeldeanlage wird die Beleuchtung eingeschaltet.
  Die Realisierung dieser Lichtfunktion befindet sich in dem Template FB_BA_BuildingLighting [▶ 933].
- **Wartungsbetrieb**
  Durch das Auslösen des Wartungsbetriebs via Management- und Bedieneinrichtung (MBE) oder TwinCAT 3 HMI wird in einem bestimmten Bereich des Gebäudes die Beleuchtung eingeschaltet.
  Die Realisierung dieser Lichtfunktion befindet sich in dem Template FB_BA_FloorLighting [▶ 935].
- **Nachtwächterrundgang**.
  Bei den Rundgängen des Sicherheitspersonal wird die Beleuchtung eingeschaltet und auf einen bestimmten Wert gedimmt.
  Die Bedienung kann via Zeitschaltplan oder manuell erfolgen.
  Die Realisierung dieser Lichtfunktion befindet sich in dem Template FB_BA_FloorLighting [▶ 935].

3.5.2.1.2 Lokale Lichtfunktionen

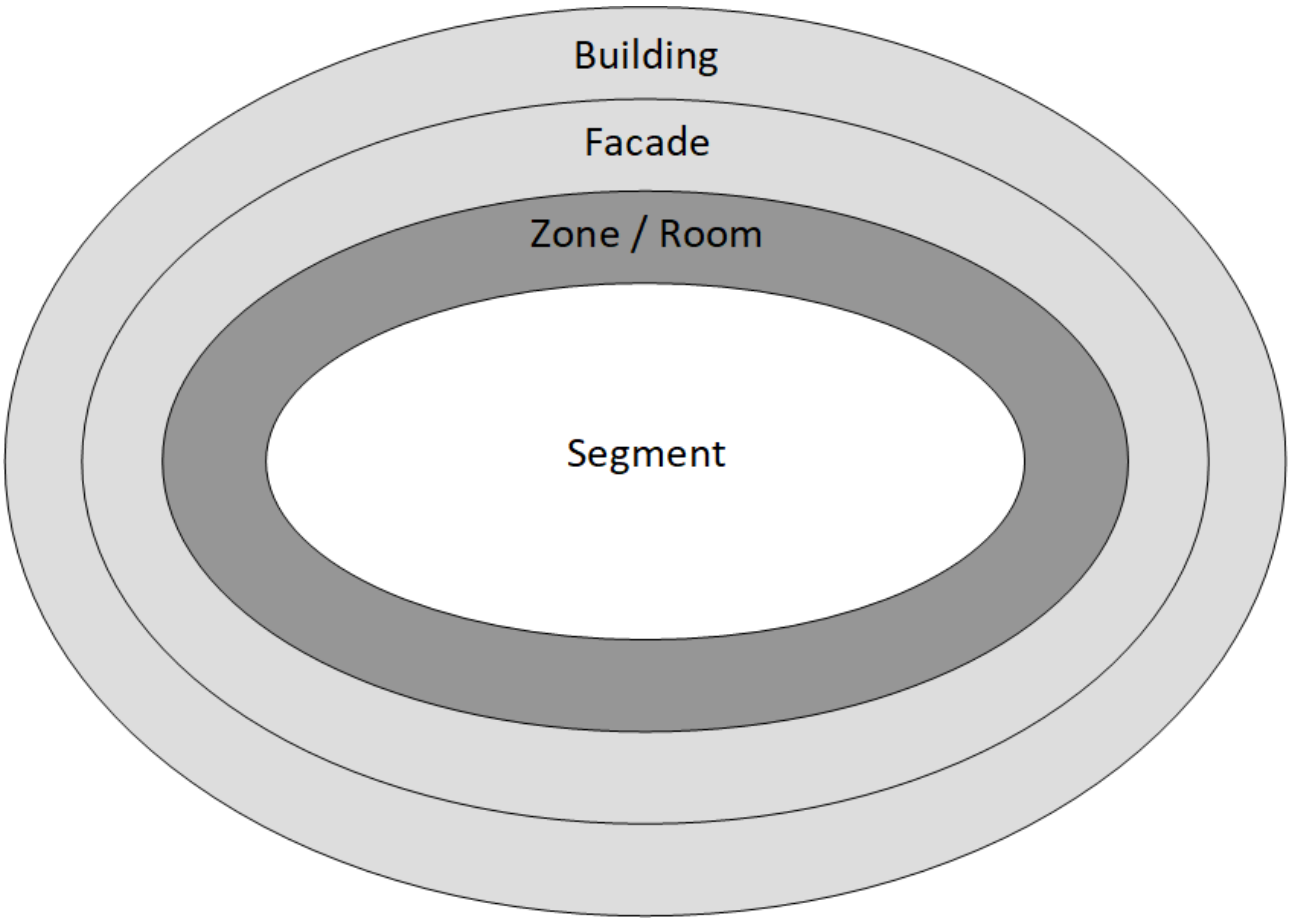
Die Anwendungsfunktionen der Beleuchtung im Raum hängen sehr stark von der Nutzung bzw. Art des Raums ab.

Die folgenden Anwendungsfunktionen für die Beleuchtung sind in TwinCAT 3 Building Automation als Template vorhanden. Alle Anwendungsfunktionen beziehen sich auf eine oder mehrere Leuchten.

- **Lichtgruppe**
  Mit Anwendungsfunktion Lichtgruppe können schalt- oder dimmbare Beleuchtungseinrichtungen ein- und ausgeschaltet oder gedimmt werden. Die Bedienung der Lichtgruppe kann über Lichttaster, Raumbediengeräte oder über eine grafische Bedienung mittels TwinCAT 3 HMI erfolgen.
  Realisiert ist diese Lichtfunktion in dem Template FB_BA_LightGroupSwitch [▶ 919].
- **Automatiklicht**
  Die Anwendungsfunktion Automatiklicht schaltet die Raumbeleuchtung bei Belegung automatisch ein. Eine Berücksichtigung der natürlichen Beleuchtung durch Tageslicht erfolgt nicht. Die Schaltung des Lichts erfolgt lediglich über Präsenzerkennung. Die Funktion dient damit besonders dem energiesparenden Einsatz der Beleuchtung in Räumen, die nicht ausreichend mit Tageslicht versorgt werden, z. B. Korridore oder Sanitärräume.
  Realisiert ist diese Lichtfunktion in dem Template FB_BA_LightGroupPresence [▶ 916].
- **Konstantlichtregelung**
  Die Anwendungsfunktion Konstantlichtregelung regelt die Raumbeleuchtung oder Teile davon bei Belegung automatisch so, dass eine eingestellte Mindestbeleuchtungsstärke nicht unterschritten wird. Dadurch wird ein kontrastreiches Arbeiten bei gleichzeitig minimalem Energieeinsatz gewährleistet. Während das Einschalten bei Unterschreitung der Mindesthelligkeit oder das Abschalten der Beleuchtung nach dem Erreichen des Stellwert-Minimums über die Ein- und Abschaltverzögerungen beeinflusst werden kann, führt eine Änderung des Belegungszustands zu unverzögerten Schaltvorgängen. Eine Übersteuerung durch einen Taster oder die grafische Bedienung via TwinCAT 3 HMI stoppt die Konstantlichtregelung. Stattdessen wird der Eingabewert des Tasters oder der TwinCAT 3 HMI an die Leuchten weitergegeben.
  Realisiert ist diese Lichtfunktion in dem Template FB_BA_LightGroupConstant [▶ 908].

3.5.2.2 Sonnenschutz

Die Sonnenschutzfunktionen orientieren sich an dem Schalenmodell der VDI 3813.



In jeder Schale des Modells für den Sonnenschutz kann es eine oder mehrere Anwendungsfunktionen des Sonnenschutzes geben. Jede Anwendungsfunktion erzeugt ein Positioniertelegramm für die Sonnenschutzaktuatoren.

Alle Positioniertelegramme (ST_BA_SunBld [▶ 279]) einer Schale werden durch einen Telegramm-Selektor ausgewertet. Das Telegramm mit der höchsten Priorität wird an die innere Schale durchgeleitet.

```
TYPE ST_BA_SunBld :
STRUCT
  fPos      : REAL;
  fAngl     : REAL;
  bManUp    : BOOL;
  bManDwn   : BOOL;
  bManMod   : BOOL;
  bActv     : BOOL;
  ePrio     : E_BA_SunBldPrio;
  nEvtInc   : UDINT;
END_STRUCT
END_TYPE
```

Innerhalb des Telegramms wird ein Stellwert für die Position des Behangs in % und ein Lamellenwinkel in Grad übermittelt. Ebenfalls sind in dem Telegramm die Variable *bActv* und *ePrio*.

In den Schalen Gebäude, Etage und Bereich, wirken globale Anwenderfunktionen des Sonnenschutzes.

Globaler Sonnenschutz [▶ 28]

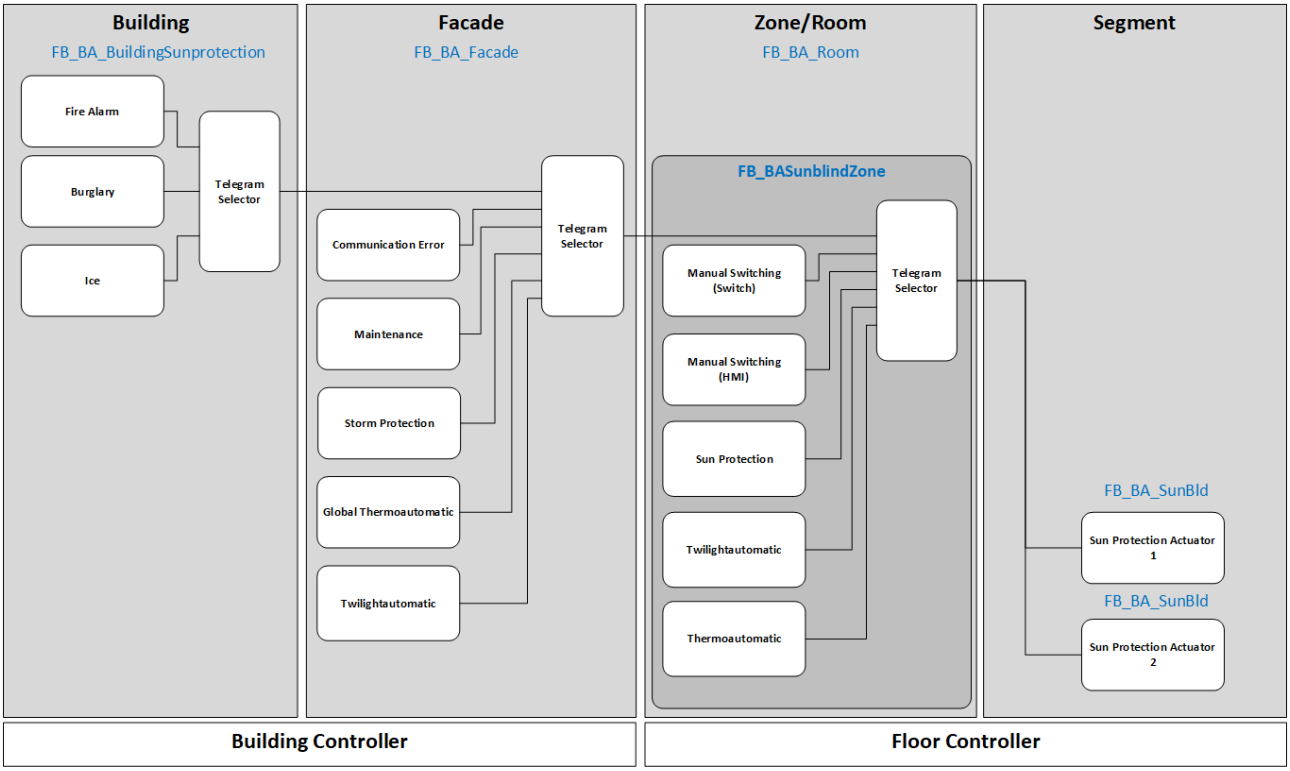
In der Schale Raum wirken die Raumanwenderfunktionen.

Lokaler Sonnenschutz [▶ 29]

Durch die Auswahl und Position der Sonnenschutzfunktionen im Schalenmodell wird die Gesamtfunktionalität des Sonnenschutzsystems bestimmt.

Die Funktionen der äußeren Schale für das Gebäude wirken auf alle Sonnenschutz Aktoren des Gebäudes.

Die Funktionen der inneren Schalen jeweils nur auf die entsprechende Teilmenge der Sonnenschutz Aktoren, welche sich innerhalb der darin eingeschlossenen Schalen befinden. Die Sonnenschutz Aktoren selbst befinden sich in der innersten Schale Segment.



3.5.2.2.1 Globaler Sonnenschutz

Neben den lokalen Funktionen des Sonnenschutzes gibt es noch andere Anwendungsfunktionen im System von denen globale Stellbefehle an die Sonnenschutz Aktoren ausgesendet werden.

In der Ebene Gebäude des Schalenmodells befinden sich die folgenden Funktionen des Sonnenschutzes.

Die Funktionen sind in dem Template FB_BA_BuildingSunprotection [▶ 953] realisiert.

- **Brand**
  Im Brandfall werden sämtliche Behänge eines Gebäudes hochgefahren.
- **Sturmschutz**
  Die Funktion Sturmschutz verhindert die Beschädigung von außenliegenden Sonnenschutzeinrichtungen durch Wind.
- **Vereisungsschutz**
  Durch die Kombination der Messwerte Außentemperatur und Niederschlag wird die Gefahr einer Vereisung der Jalousien prognostiziert. Zum Schutz einer mechanischen Beschädigung der Behänge fahren sie bei Vereisungsgefahr hoch und verharren dort so lange bis eine prognostizierte Vereisungszeit abgelaufen ist.

Für jede Fassade des Gebäudes gibt es eine Instanz des Templates FB_BA_Facade [▶ 956].

Innerhalb des Templates befinden sich auf eine Fassade bezogenen Funktionen und Berechnungen.

- **Wartung**
  Zur Kontrolle ob alle Behänge einer Fassade voll funktionstüchtig sind und richtig positionieren, ist es über die Wartungsfunktion möglich alle Behänge einer Fassade Synchron auf und abzufahren. Im Falle einer Fassadenreinigung können die Behänge einer Fassade alle hochgefahren und in dieser Position blockiert werden.
- **Globale Thermoautomatik**
  Die globale Thermoautomatik hat die gleiche Funktion wie die lokale Thermoautomatik innerhalb der Räume. Sie steuert jedoch alle Behänge einer Fassade gleich, sodass in längeren Abwesenheitszeiten z. B. an Wochenenden ein einheitliches Bild aller Behänge einer Fassade entsteht. Die globale Thermoautomatik nutzt den Raumtemperaturmesswert eines Referenzraumes.

- **Kommunikationsfehler**
  Bei Kommunikationsproblemen innerhalb des GA-Netzwerkes ist nicht sichergestellt, dass die Positioniertelegramme mit einer hohen Priorität bis zu den Sonnenschutzaktuatoren durchgeleitet werden. Aus Sicherheitsgründen werden die Behänge dann hochgefahren.

3.5.2.2.2 Lokaler Sonnenschutz

Je nach Nutzung und technischer Ausstattung eines Raums können die Anwendungsfunktionen variieren.

In den TF8040 Templates des Sonnenschutzes wird von einer oder mehreren Gruppen an außenliegenden Raffstoren bzw. Jalousien ausgegangen.

Die Templates der Sonnenschutzfunktionen beziehen sich immer auf eine Gruppe von Jalousien welche parallel angesteuert werden. Funktionsbausteine für die Einbindung von Rollladen sind ebenfalls verfügbar.

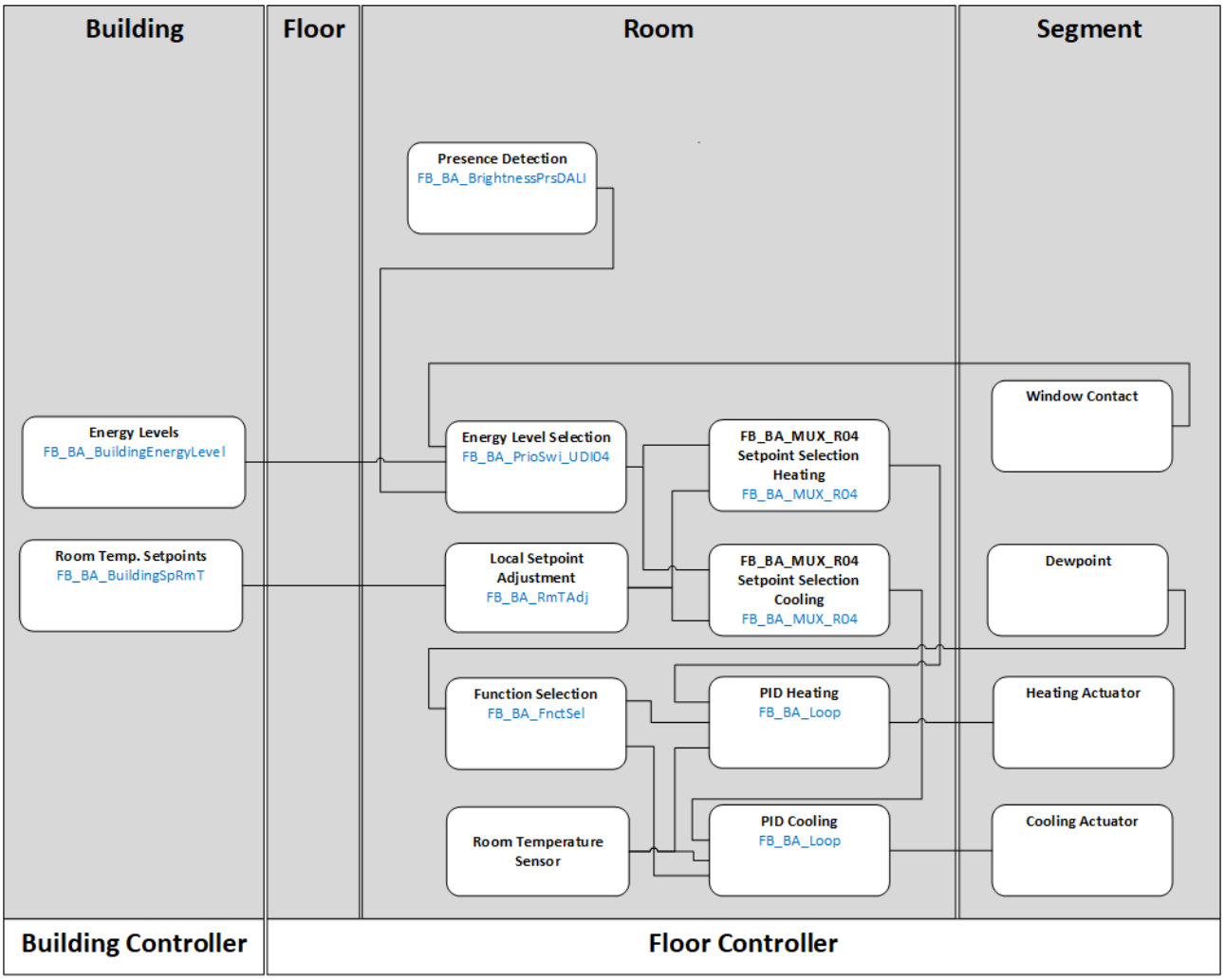
Alle raumbezogenen Funktionen des Sonnenschutzes befinden sich in dem Template FB_BA_SunblindZone [▶ 980].

Die Raumfunktionen für eine Jalousiegruppe sind:

- **Sonnenschutz mit Lamellennachführung**
  Der Sonnenschutz dient in erster Linie als Blendschutz. Die Stellung der Lamellen wird dabei zyklisch dem aktuellen Sonnenstand angepasst. Dadurch wird jeder Raum trotz der Verhinderung des direkten Sonnenlichteinfalls bestmöglich mit Tageslicht versorgt. Der Energieverbrauch für die künstliche Beleuchtung eines Raums wird so gering wie möglich gehalten. Der Sonnenschutz verhindert ebenfalls die Überhitzung des Raums durch Einstrahlung und reduziert so den Energieaufwand für die Kühlung. Die Aktivierung des Sonnenschutzes erfolgt bei Präsenz im Raum.
- **Thermoautomatik**
  Mithilfe der Thermoautomatik wird der Sonnenschutz in unbelegten Räumen zur Unterstützung der Heizung oder Kühlung eingesetzt, indem gezielt solare Wärmeeinträge zugelassen oder vermieden werden. So kann im Sommer eine Überhitzung vermieden und im Winter die Heizung entlastet werden. Die Thermoautomatik verarbeitet die Sensorfunktion Raumtemperatur und Raumtemperatursollwert.
- **Dämmerungsautomatik**
  Mithilfe der Funktion Dämmerungsautomatik können Sonnenschutzeinrichtungen in Abhängigkeit der Außenhelligkeit positioniert werden. Diese Funktion gestattet z. B. das Schließen des Sonnenschutzes während der Nachtstunden, um z. B. ein Auskühlen über die Fenster zu verringern oder die Lichtemissionen des Gebäudes zu reduzieren, und ungewollte Blicke in das Gebäude von außen zu vermeiden.
- **Manuelles Stellen**
  Die manuelle Übersteuerung der Automatikfunktionen des Sonnenschutzes kann mittels eines Tasters, eines Raumbediengerätes welches über Feldbus eingebunden sein kann, oder über ein grafisches Bedienfeld via TwinCAT 3 HMI erfolgen.

3.5.2.3 Klima

Die Raumklimafunktionen orientieren sich an dem Schalenmodell aus der Norm für die Raumautomation VDI 3813.



In der Schale Gebäude befindet sich ein Zeitschaltprogramm. Dieses Zeitschaltprogramm beschreibt die Zeiträume in denen sich die Räume des Gebäudes in Schutzbetrieb (*Protection*) in Absenkbetrieb (*Economy*), im *Pre-Comfort* oder im *Comfort*-Betrieb befinden.

Die Betriebsarten der Gebäude werden auch als Energie-Niveaus bezeichnet.

Je länger das Gebäude unbelegt ist, desto weiter kann das Energieniveau abgesenkt werden.

Bei sehr langer Nichtbenutzung kann das Gebäude in den Schutzbetrieb gesetzt werden.

Im Winter wird die Raumtemperatur auf einen Frostschutzsollwert und im Sommer auf einen Überhitzungsschutzsollwert reduziert bzw. erhöht.

Bei längeren regulären Abwesenheitszeiten, z.B. an den Wochenenden oder während der Nacht, wird das Energieniveau *Economy* geplant.

Mit dem *Pre-Comfort* Niveau werden die Sollwerte so weit angehoben, dass sie für den *Comfort*-Betrieb kurzfristig erreicht werden können.

Die Sollwerte des Niveaus *Comfort* können bei Bedarf mittels eines lokalen Sollwertstellers im Raum geringfügig verändert werden.

Die zentrale Erzeugung und Verteilung der Raumtemperatursollwerte für das gesamte Gebäude befinden sich im Gebäude Controller, siehe Skizze ADS-Kommunikation innerhalb der Templates [▶ 18] enthalten.

3.6 Grundgerüst

Das Grundgerüst in TwinCAT 3 Building Automation ist eine wesentliche Eigenschaft und Grundlage von TF8040. Das Grundgerüst (GG) beinhaltet Funktionen, welche im Hintergrund einer TF8040-Solution in der SPS ablaufen. Die Funktionen des Grundgerüstes sind Bestandteil der Bibliothek Tc3_XBA [▶ 101].

Das Grundgerüst bringt einen sehr großen Nutzen bei der Umsetzung eines GA-Projektes (Gebäudeautomationsprojektes). Dieses Kapitel dient der Erläuterung dieser Funktionen.

Erstellung der Projektstruktur (PS)

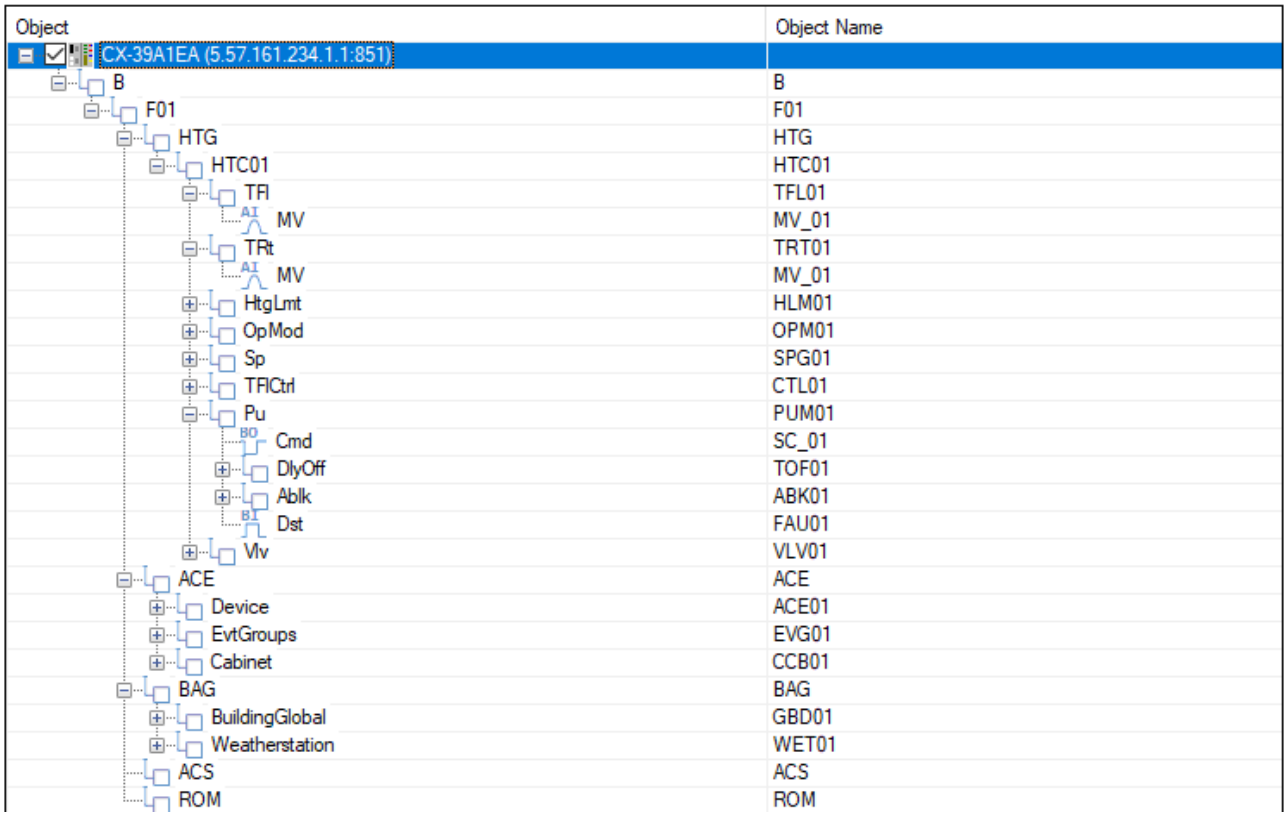
Ein Gebäudeautomationsprojekt organisiert sich in der Regel in Ebenen. Die Ebenen beginnen beispielsweise mit der Liegenschaft oder dem Ort. Weitere Ebenen können das Gebäude, die darin enthaltenden Anlagen, die dazugehörigen Aggregate, und zuletzt die darin enthaltenen Datenpunkte sein.

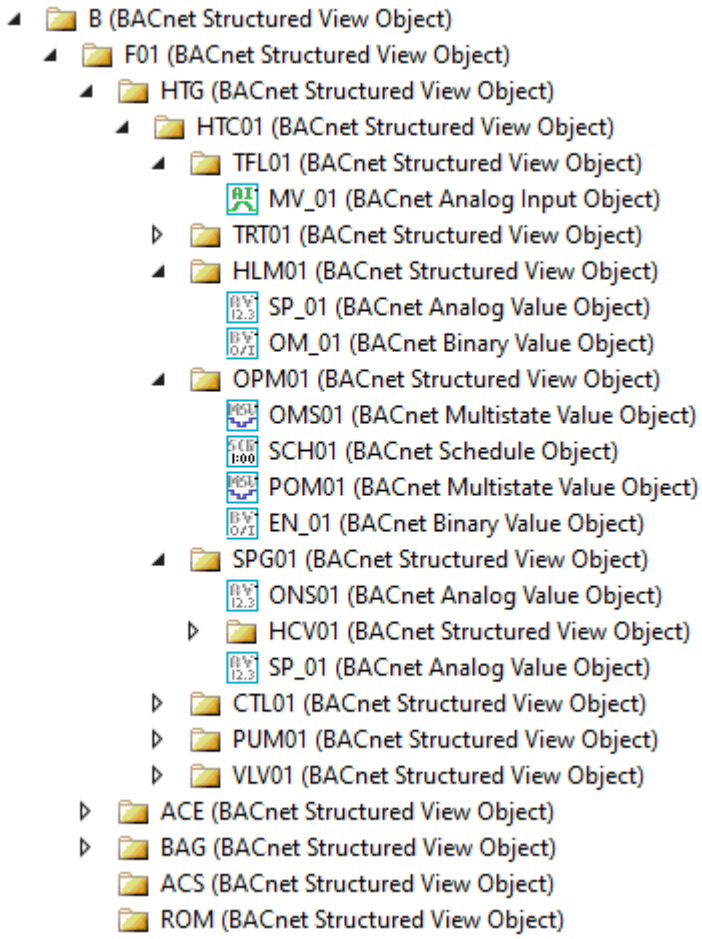
Die wichtigste Aufgabe des Grundgerüst ist es, die Struktur eines Projektes in TwinCAT abzubilden. Die Projektstruktur ist ein System bestehend aus Ordnern und darin enthaltenen Objekten.

Bei großen Projekten mit sehr vielen Objekten ist diese Projektstruktur eine wichtige Basis, um ein Projekt nachhaltig und übersichtlich zu gestalten.

Das Bild zeigt eine mittels des Grundgerüstes erzeugte Projektstruktur im Site Explorer [▶ 1306].

Die Projektstruktur bildet sich in dem Tool Site Explorer, in der BACnet Konfiguration und in einer generischen Navigation der TwinCAT HMI ab.





Der Nutzen des Grundgerüstes welches die Projektstruktur in einer übersichtlichen Form in der HMI, dem BACnet und auch in der BACnet MBE erzeugt ist sehr groß.

Von der Inbetriebnahme, der Parametrierung, der Bedienung via HMI, der späteren Wartung und dem Aufschalten auf eine BACnet MBE wird alles durch die Projektstruktur erleichtert.

Die Erstellung einer Projektstruktur wird in dem Kapitel Projektstruktur [▶ 39] beschrieben.

DPAD(Benutzer-Adress-Schlüssel)

Ein weiterer wesentlicher Vorteil ist die Verarbeitung eines Benutzer-Adress-Schlüssels (BAS).

Alle vorab beschriebenen Ebenen der Projektstruktur erhalten einen Namen und einen Beschreibungstext.

Das Grundgerüst von TF8040 kann die Objektnamen und die Beschreibungstexte der Ebenen verketten und daraus die Namen und Beschreibungen der BACnet-Objekte erzeugen. In TwinCAT 3 Building Automation wird der BAS auch als DPAD Datapoint Adressing Descripton bezeichnet. Ausführliche Informationen befinden sich in dem Kapitel DPAD [▶ 43].

Events

Das Grundgerüst erfasst die Events der Objekte innerhalb der Projektstruktur.

In jeder Ebene der Projektstruktur werden eine Sammlung der Events inklusive aller Unterelemente erzeugt.

Ein Funktionsbaustein namens FB_BA_PlantLock [▶ 138] kann für Anlagenabschaltungen benutzt werden, ohne eine Sammelstörmeldung programmieren zu müssen.

Das Quittieren und Rücksetzen von Events wird ebenfalls durch das Grundgerüst organisiert. Ein Rücksetz- oder Quittier- Befehl wird automatisch vom Top-Level der Projektstruktur bis hinein in die tiefste Ebene der Projektstruktur den Objekten durchgeleitet. Die Verarbeitung von Events wird in der Dokumentation Events [▶ 33] näher beschrieben.

Ereignisse Zählen

Jeder Ordner bzw. jedes View-Objekt ist in der Lage, die Statusinformationen wie z. B. OutOfService, InAlarm, Overridden etc. von allen untergeordneten Objekten zu sammeln und deren Anzahl auszugeben. Die Methode EventConditionCount zählt die anstehenden Ereignisse von Objekten der jeweiligen Ebene der Projektstruktur.

3.6.1 Objekte

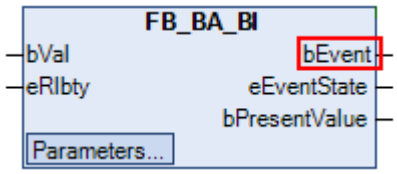
Objekte sind die elementaren Bestandteile des Grundgerüstes. Ein Objekt bietet verschiedene Eigenschaften. Alle Objekte in TwinCAT 3 Building Automation beziehen sich auf die Objekte der TwinCAT Function TF8020 BACnet.

3.6.1.1 Events

Das Grundgerüst von TwinCAT 3 Building Automation und die darin enthaltenen Objekte bieten umfangreiche Funktionen für die Verarbeitung von Ereignissen bzw. Events.

Ein Event bezieht sich in TF8040 grundsätzlich auf ein Objekt. Events entstehen, wenn ein Objekt einen unnormalen bzw. einen fehlerhaften Zustand annimmt. Eventfähige Objekte in TwinCAT 3 Building Automation besitzen zur weiteren Verarbeitung des Events innerhalb des TwinCAT Programms den Ausgang *bEvent*.

Beispiel:FB_BA_AI



Der Zustand eines Objektes wird mit dem *EventState* beschrieben.

Mögliche Event States sind:

Zustand	Beschreibung
eNormal	Der Zustand des Objektes ist normal.
eFault	Der Zustand des Objektes ist fehlerhaft.
eOffnormal	Der Zustand des Objektes ist unnormal.
eLowLimit	Der obere Grenzwert eines analogen Objektes wurde überschritten.
eHighLimit	Der untere Grenzwert eines analogen Objektes wurde unterschritten.

Anzeige von Events

Events werden in der Eventliste des Site Explorer [▶ 1306] und der TcHmiBa [▶ 1139] angezeigt. Ebenfalls werden die Events via BACnet-Server bei Bedarf an BACnet-Clients übertragen.

Die Darstellung eines Events hängt von den folgenden Eigenschaften ab:

- Event Typ
- Alarm Modus
- Quittier- und Rücksetz-Zustand

Somit ergeben sich folgende Möglichkeiten zur Darstellung eines Events (Illustration am Beispiel eines Alarm-Events):

Bezeichnung	Abbildung	Beschreibung
Ausgeblendet		Es steht kein Event an.
Angedeutet*		Das Event steht nicht (mehr) an, wird jedoch zu Informationszwecken angedeutet, bis es quittiert wird.
Vergangen und **quittiert****		Das Event steht nicht (mehr) an. Es wurde aber bereits quittiert aber noch nicht zurückgesetzt.
Vergangen**		Das Event steht nicht (mehr) an. Es wurde aber weder quittiert noch zurückgesetzt.
Anstehend und **quittiert**		Das Event steht an und wurde bereits quittiert.
Anstehend		Das Event steht an.

* Nur möglich im Alarm-Modus *Standard*!
** Nur möglich bei *erweitertem* Alarm-Modus!

Pro Event-Typ ergeben sich folgende Darstellungen:

Zustand	Alarm	Störung	Wartung	Benachrichti-gung	Sonstiges
Ausgeblendet	-	-	-	-	-
Angedeutet					
Vergangen, Quittiert					
Vergangen					
Anstehend, Quittiert					
Anstehend					

Eventsteuerungen

Kritische Ereignisse erfordern oftmals eine steuerungstechnische Reaktion, wie zum Beispiel das Abschalten einer Lüftungsanlage nach dem Fall einer Brandschutzklappe.

Mit den Lock-Funktionalitäten der eventfähigen Objekte wird die gewünschte Steuerungsfunktionalität parametriert.

Hierzu werden die Events innerhalb der Ebenen in der Projektstruktur [▶ 31] mittels des Funktionsbausteins FB_BA_PlantLock [▶ 138] zusammengefast und ausgewertet.

Parametrieren von Events

Ein Event kann unterschiedliche Anforderungen bezüglich seiner Anzeige, seiner Steuerungstechnischen Verarbeitung und des Quittierens und Rücksetzens haben. Die meisten dieser Eigenschaften werden nicht am Objekt selbst sondern, mit der dem Objekt zugewiesenen Eventklasse FB_BA_EC [▶ 223] parametriert.

Ein Event wird als aktiv bezeichnet, sobald es sich nicht mehr im Normalzustand (Ausgeblendet) befindet.

Quittieren und Rücksetzen

Der Benutzer kann mit aktiven Events interagieren. Dabei hat er (je nach konfiguriertem Alarm Modus) folgende Möglichkeiten:

Mit dem Funktionsbaustein FB_BA_EventObserver [▶ 137] ist es möglich eine Sammelquittierung bzw. Rücksetzung aller Events innerhalb der Projektstruktur vorzunehmen. Welche Objekte quittiert bzw. rückgesetzt werden, hängt von der Position des FB_BA_EventObserver [▶ 137] in der Projektstruktur ab. Generell werden alle Objekte quittiert oder zurückgesetzt, welche sich in der Projektstruktur in dem gleichen Ordner oder einem Unterordner befinden.

- **Quittieren**
  Signalisiert (z. B. dem Wartungspersonal) ein wahrgenommenes Event.
  Vom Verständnis her sollte sich ableiten lassen, dass ein entsprechender Handlungsbedarf jetzt vollzogen wird.
  Das Quittieren hat also einen informierenden Charakter.
- **Rücksetzen**
  Im erweiterten Alarm Modus muss ein Event (bzw. ein Objekt) nicht nur quittiert, sondern auch zurückgesetzt werden, um ein bereits vergangenes Event in den Normalzustand zu versetzen.
  Das Rücksetzen verhindert also das Auftreten von undefinierten Zuständen (z. B. unkontrolliertes Wiederanlaufen von Anlagen) und bietet somit eine zusätzliche Sicherheit.

Lock-Prioritäten

Definieren die Priorität für abschaltende Events die so zum Beispiel eine gewünschte Auswirkung auf das FB_BA_PlantLock [▶ 138] bewirken.

- **Lokal Medium**
  Gibt ein lokales Abschalten mittlerer* Priorität frei.
- **Lokal Hoch**
  Gibt ein lokales Abschalten höherer** Priorität frei.
- **Medium**
  Gibt ein übergeordnetes Abschalten* mittlerer Priorität frei.
- **Hoch**
  Gibt ein übergeordnetes Abschalten** höherer Priorität frei.

* Wird für anlagensichere Programmabschnitte verwendet.

** Wird für personensichere Programmabschnitte verwendet.

3.6.1.2 Kommandierung

Mehrere Steuerungs- bzw. Stellbefehle eines kommandierbaren Objektes haben Einfluss auf den Ausgabewert *PresentValue*. Alle Befehle werden in einem Array gespeichert. Der Befehl mit der höchsten Priorität bestimmt das Ergebnis am Ausgang des Objektes.

Eine Priorität kann einen Wert zwischen 1 und 16 erhalten. Die höchste Priorität hat den Wert 1.

Die folgenden Objekte in TwinCAT 3 Building Automation sind kommandierbare Objekte und besitzen somit ein Prioritäten-Array.

FB	Typ	Beschreibung
FB_BA_BO_Raw	BO	Binärer Ausgang mit externer Deklaration der Variablen für das Mapping der Hardware.
FB_BA_BO	BO	Binärer.
FB_BA_BO_IO	BO	Binärer Ausgang mit interner Deklaration der Variablen für das Mapping der Hardware.
FB_BA_AO_Raw	AO	Analoger Ausgang mit externer Deklaration der Variablen für das Mapping der Hardware.
FB_BA_AO	AO	Analoger.
FB_BA_AO_IO	AO	Analoger Ausgang mit interner Deklaration der Variablen für das Mapping der Hardware.
FB_BA_MO_Raw	MO	Multi State Ausgang mit externer Deklaration der Variablen für das Mapping der Hardware.
FB_BA_MO	MO	Multi State Ausgang.
FB_BA_MO_IO	MO	Multi State Ausgang mit interner Deklaration der Variablen für das Mapping der Hardware.
FB_BA_BV	BV	Binary-Value-Objekt.
FB_BA_MV	MV	Multistate-Value-Objekt.
FB_BA_AV	AV	Analog-Value-Objekt.

Damit ein Eintrag in das Priority Array eines Objektes erfolgt, muss der zugehörige Freigabeeingang *bEn*… am dem Funktionsbaustein TRUE sein.

Die Priorität Manual Remote wird nicht mittels eines Eingangs am Funktionsbaustein, sondern durch das Beschreiben einer Parametervariablen aktiviert. Die Kommandierung des Objektes erfolgt entweder über das BACnet-Protokoll z. B. von einer MBE oder von TwinCAT via ADS.

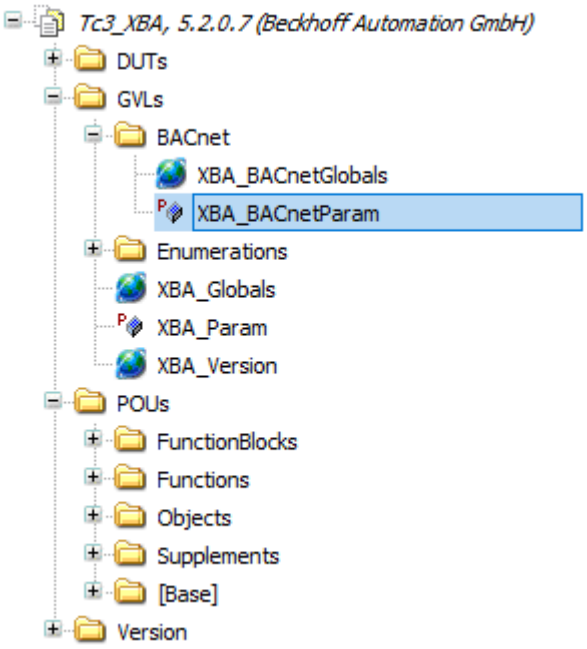
Die folgenden Prioritäten sind an den Objekten von der TwinCAT Building Automation vordefiniert:

Name	Beschreibung	Symbol Freigabe	Symbol Wert	Default Prio
Safety	Personensicherheit	bEnSafety	*ValSfty	1
Critical	Anlagensicherheit	bEnCrit	*ValCritical	3
ManLoc	Lokale manuelle Übersteuerung (LVB)	bEnManLoc	*ValManLoc	7
ManualRm	Manuelle Übersteuerung aus der Ferne (MBE)	bEnManualRm	*ValManualRm	8
Pgm	Programmsteuerung	bEnPgm	*ValPgm	15

Je nach Art des Objektes ob Analog, Binär oder Multi-State, ist der Wert eines Kommandos ein REAL, BOOL oder UDINT.

Ändern der Prioritäten

Die Prioritäten können in der Variablenliste BA_BACnet_Param geändert werden.


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

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

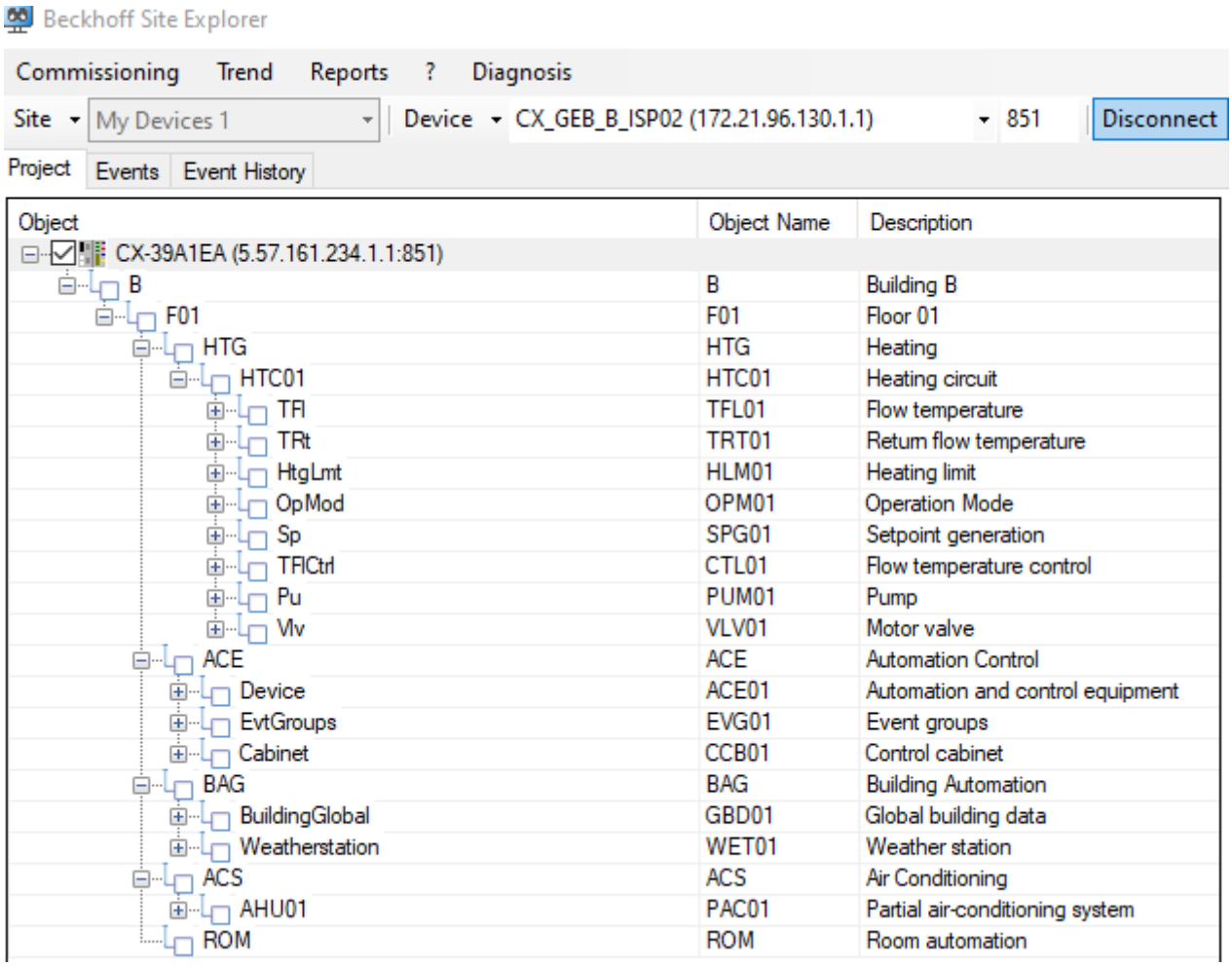
Die aktive Priorität wird am Ausgang aller zu kommandierenden Objekte mittels der Variablen *eActivePrio* angezeigt.

Variablen

Name	Typ	Beschreibung
bEnSafety	BOOL	Freigabe der Priorität „Safety“.
fValSafety	REAL	Analoger Wert für die Priorität „Safety“.
bValSafety	BOOL	Binärer Wert für die Priorität „Safety“.
nValSafety	INT	Integer Wert für die Priorität „Safety“.
bEnCritical	BOOL	Freigabe der Priorität „Critical“.
fValCritical	REAL	Analoger Wert für die Priorität „Critical“.
bValCritical	BOOL	Binärer Wert für die Priorität „Critical“.
nManCritical	INT	Integer Wert für die Priorität „Critical“.
bEnManLocal	BOOL	Freigabe der Priorität „Manual Local“.
fValManLocal	REAL	Analoger Wert für die Priorität „Manual Local“.
bValManLocal	BOOL	Binärer Wert für die Priorität „Manual Local“.
nManLocal	INT	Integer Wert für die Priorität „Manual Local“.
bEnManualRm	BOOL	Freigabe der Priorität „Manual Remote“.
fValManualRm	REAL	Analoger Wert für die Priorität „Manual Remote“.
bValManualRm	BOOL	Binärer Wert für die Priorität „Manual Remote“.
nManualRm	INT	Integer Wert für die Priorität „Manual Remote“.
bEnPgm	BOOL	Freigabe der Priorität „Program“.
fValPgm	REAL	Analoger Wert für die Priorität „Program“.
bValPgm	BOOL	Binärer Wert für die Priorität „Program“.
nPgm	INT	Integer Wert für die Priorität „Program“.

3.6.2 Projektstruktur

In diesem Abschnitt wird das Erstellen einer Projektstruktur beschreiben, wie in der folgenden Abbildung zu sehen.



Die Initialisierung der Projektstruktur beginnt im Hauptprogramm MAIN der Automationsstation.

Zunächst muss festgelegt werden, wie viele Ebenen in dem Projekt abzubilden sind.

Die Projektstruktur dient nicht nur der Gliederung eines Projektes, sondern auch der Verarbeitung eines BAS (Benutzer-Adress-Schlüssels).

Die Ebenen der Projektstruktur werden deshalb mit dem Initialisieren des Funktionsbausteins FB_BA_CarelessDPAD beschrieben.

Der Aufruf des Funktionsbausteins erfolgt in dem Hauptprogramm MAIN der Automationsstation. Eine detailliertere Beschreibung zur Verarbeitung des BAS findet sich im Kapitel DPAD [▶ 43].

Die Grafiken dieser Beschreibung beziehen sich auf die Standard TF8040 PLC.

Innerhalb der Muster-SPS werden die Ebenen Gebäude, Etage, Gewerk, Anlage, Aggregat und Funktion erzeugt.

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

In MAIN wird mit der Initialisierung der Projektstruktur (PS) begonnen. Eine Ebene innerhalb der Projektstruktur wird durch einen Ordner abgebildet. Ein Ordner wird durch eine Instanz des Funktionsbausteins FB_BA_View [▶ 246] erzeugt.

Innerhalb eines Ordners können sich weitere Unterordner oder Objekte befinden. Objekte werden mittels der Funktionsbausteine aus dem Kapitel Objects [▶ 195] der Bibliothek Tc3_XBA [▶ 101] erzeugt.

Das Editieren der Projektstruktur geschieht durch das Anlegen und Verketten von Ordnern. Die Verkettung geschieht, indem jedem Ordner innerhalb des FB_BA_View [▶ 246] ein Parent zugewiesen wird.

Die Zuweisung des Parent geschieht, indem der Variablen *iParent* der Symbolname des in der Projektstruktur nächst höheren FB_BA_View [▶ 246] übergeben wird.

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

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

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

In dem Beispiel ist zu sehen, dass im Programm MAIN die Ebenen Gebäude bis Gewerk angelegt werden.

Es gibt die Gewerke Wärmeversorgungsanlagen, Automation, Gebäudeautomation Allgemein, Lufttechnische Anlagen und Raumautomation.

Der Parent aller Gewerke ist F01. Es werden so die fünf Unterordner für die Gewerke unterhalb des Ordners Etage F01 erstellt.

Eine Methode eine Ebene der Projektstruktur zu erzeugen, ist der unmittelbare Aufruf eines FB_BA_View [▶ 246] im Main Programm.

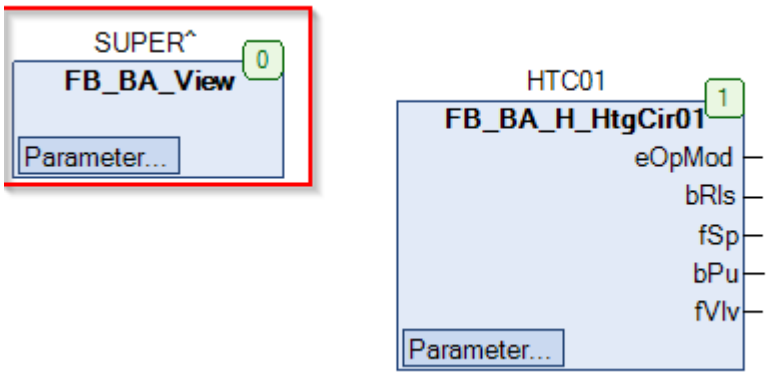
In unserem Beispiel ist dieses so mit den Ebenen Gebäude und Etage realisiert.

Bei den Funktionsbausteinen für die Gewerke Wärmeversorgungsanlagen, Automation, Gebäudeautomation Allgemein, Lufttechnische Anlagen und Raumautomation entstehen die Ordner, mit dem Aufruf eines Funktionsbausteins, welcher ebenfalls einen Ordner repräsentiert.

Allerdings entstehen diese Ordner dadurch, dass die Bausteine ACE, BAG, ACS, HTG und ROM alle von dem Funktionsbaustein FB_BA_View [▶ 246] erben.
Dieses geschieht mit dem Operator EXTENDS.

Hier beispielhaft der Gewerkebaustein Wärmeversorgungsanlagen.

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



So ist der Funktionsbaustein FB_BA_Heating ein Ordner und gleichzeitig ein FB innerhalb dessen alle Anlagen dieses Gewerkes z.B. Heizkreise, aufgerufen werden können.

Dieser Mechanismus der Ordnererstellung mittels des Vererbens vom FB_BA_View [▶ 246], setzt sich bis in die tiefsten Strukturen der Projektstruktur fort.

Im Falle des Heizreises HTC01, welcher innerhalb des FBs HTG aufgerufen wird, ist der Parent des Funktionsbaustein derjenige, in dem HTC01 selbst aufgerufen wird.

In diesem Fall wird der Parent des Heizkreises mit dem THIS^ Operator initialisiert.

```
METHOD FB_init : BOOL
VAR_INPUT
    bInitRetains : BOOL; // if TRUE, the retain variables are initialized (warm start / cold start)
    bInCopyCode : BOOL;  // if TRUE, the instance afterwards gets moved into the copy code (online change)
END_VAR
```

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

Diese Regel setzt sich bis zu den Aggregaten des Heizkreises und seinen darin enthaltenen Objekten fort.

Für die Erstellung der Projektstruktur ist folgendes zu beachten:

- Die Verkettung aller Ordner und Objekte der PS muss geschlossen sein.
- Alle Mechanismen des Grundgerüstes funktionieren sonst nicht!
- Alle Funktionsbausteine, welche einen Ordner innerhalb der Projektstruktur abbilden und wiederum Bausteine von Unterebenen aufrufen müssen vom FB_BA_View [▶ 246] erben.
- Der Aufruf des FB_BA_View [▶ 246] mit dem Super^ Operator hat am Anfang des Funktionsbausteins zu erfolgen.
- Die Ausführungsreihenfolge im CFC ist zu beachten.
- Alle Funktionsbausteine dessen Parent der Funktionsbaustein ist, innerhalb dessen er selbst aufgerufen wird, müssen als Parent den Parameter THIS^ erhalten.
- Sämtliche Funktionsbausteine aus dem Template Repository mit Parametern müssen innerhalb des Deklarationsbereiches von VAR_INPUT_CONSTANT aufgerufen werden.

3.6.3 DPAD

Alle Objekte innerhalb der Projektstruktur von TF8040 erhalten einen Namen und einen Beschreibungstext. Eine gängige Praxis in der Gebäudeautomation ist es, die Objekte nach einen Benutzeradressschlüssel (BAS) zu benennen. Je nach Komplexität des projektspezifischen BAS, bieten sich mit TF8040 verschiedene Möglichkeiten der Bearbeitung des BAS an. Der BAS wird in TwinCAT 3 Building Automation als DPAD (Data Point Addressing Description) bezeichnet.

Generische Erzeugung

Generische Erzeugung [▶ 44] von ObjectName und Description mit dem DPAD (Data Point Addressing Description).

Diese Methode bietet sich an bei einem Projekt, in dem der BAS aus einer Verkettung der Ebenen der Projektstruktur erzeugt werden kann.

Voraussetzung hierfür ist, dass die Reihenfolge der Ebenen der Projektstruktur mit der Reihenfolge der Textverkettungen des BAS übereinstimmt.

Individualtexte für z. B. Schaltplanseite, Gerätekennzeichnung aus dem Schaltplan etc. lassen sich durch den generischen Ansatz des DPAD nur mit höherem Aufwand verarbeiten.

Beim DPAD-Mechanismus ist zu beachten, dass die Initialtexte der Objekte nur im ersten Zyklus der Steuerung in die persistenten Daten der Steuerung übernommen werden. Nur nach einem Urlöschen der Steuerung werden die Initialdaten erneut geladen.

Da der Verkettungsmechanismus auf die Persistenten Daten zugreift, werden Änderungen auch nach einem erneuten Verketten der Texte, nicht wirksam.

Bearbeitung mit dem Symbol Explorer

Der Symbol Explorer [▶ 44] ist ein hervorragendes Tool zur Bearbeitung von Daten.

Mittels spezieller Listen können alle Steuerungsparameter inklusive der Texte komfortabel bearbeitet werden. Es ist möglich die Daten in eine csv-Datei zu exportieren, darin zu bearbeiten und anschließend wieder in die Steuerung hineinzuladen.

3.6.3.1 Bearbeitung mit dem Symbol Explorer

In manchen Projekten gibt es Vorgaben an den Benutzeradressschlüssel, welche mit dem DPAD-Algorithmus nicht optimal realisierbar sind.

Alternativ zu dem DPAD-Algorithmus, können die Namen- und Beschreibungstexte der Objekte, auch mit Hilfe der BaExtension des Symbol Explorers bearbeitet werden.

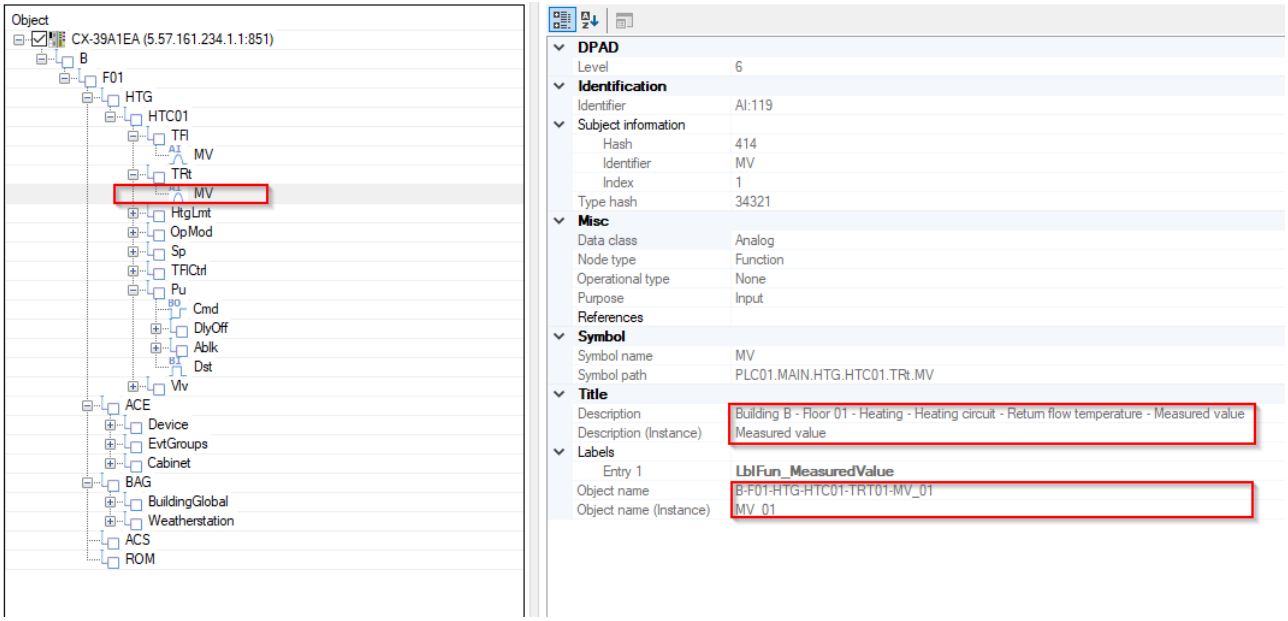
3.6.3.2 Generische Erzeugung

Bei der generischen Erzeugung des DPAD werden die Texte der Ebenen des Grundgerüst automatisch miteinander verkettet.

Aus den Namen- und Beschreibungen der Objekte und Ordner aus der Projektstruktur, entstehen vollständige Objektnamen und Objektbeschreibungen.

Zur Erklärung des Mechanismus hier ein Beispiel:

Die Darstellung zeigt den Messwert (*MV*) des Rücklauftemperaturfühler (*TRt*) aus dem Heizkreis 01 HTC01.



Voraussetzung für die generische Erzeugung des DPAD ist eine Instanz des Funktionsbausteins FB_FB_BA_CarelessDPAD. Es ist zu empfehlen den Funktionsbaustein gleich zu Beginn in MAIN aufzurufen.

In der Initialisierung des DPAD-Bausteins wird die Zahl der Ebenen, die Art der Trennzeichen und die Zeichenbreite der Indexe festgelegt. Wichtig ist, dass die Zahl der Initialisierten Ebenen des DPAD mit denen der Projektstruktur übereinstimmt.

Die Indexe werden meistens für die Benennung der Ebenen Aggregat und Funktion des Datenpunktes verwendet. In dem Initialisierungsbeispiel sind für die Indexe jeweils zwei Stellen (00 bis 99) vorgesehen.

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

Der Verkettungsalgorithmus des DPAD-Bausteins erkennt die Texte, welche verkettet werden sollen, nur wenn sie als Platzhalter {} innerhalb der geschweiften Klammern definiert sind.

Platzhalter sind sehr variabel und vielseitig einsetzbar. Eine ausführlichere Beschreibung der Platzhalter befindet sich im Kapitel: Objekt-Beschreibungstexte

Es gibt zwei Methoden die Initialisierungstexte für den Objektnamen und die Beschreibung vorzugeben:

Direktes Beschreiben von Namen und Beschreibung

Beispiel:

sObjectname := 'Mustername';

sDescription := 'Musterbeschreibung';

Verwendung von Platzhaltern

Eine Variante diese Texte zu initialisieren ist der Einsatz von Platzhaltern.

Diese werden üblicherweise im *FB_init* des Templates (in diesem Beispiel FB_BA_H_HtgCir01 [▶ 993]) definiert.

Nur in Platzhalter {} initialisierte Texte werden vom DPAD-Algorithmus mit den anderen Ebenen der Projektstruktur verkettet.

In den folgenden Beispielen soll ersichtlich werden, wie die Verkettung der Texte durch Platzhalter beeinflusst werden kann.

Alle folgenden DPAD-Initialisierungsbeispiele beziehen sich auf die Benennung der im Heizkreis vorhandenen Pumpe.

Objektnamen und Description werden in diesen Beispielen direkt ohne die Verwendung von Labels beschrieben.

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

Name verkettet und fester Index

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

Beschreibungstext mit Objektindex

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

Name verkettet und automatischer Index

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

Einstiegspunkt für Verkettung durch !{} definiert und ObjektIndex {Idx} aktiviert

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

Name verkettet ohne Index

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

Benutzerdefinierter Name

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

Name unverkettet und fester Index

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

Übernahme des Objekt-Index in den Beschreibungstext

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

Name unverkettet und fester Index

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

Definierter eigener Index im Beschreibungstext

`Pu.sDescription :='Pump{Idx=123}';`	= Pump123

Name unverkettet und InstanzID von Objekt

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

Name unverkettet und automatisch

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

Name unverkettet ohne Index

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

Labels zur Definition von Namen und Beschreibung

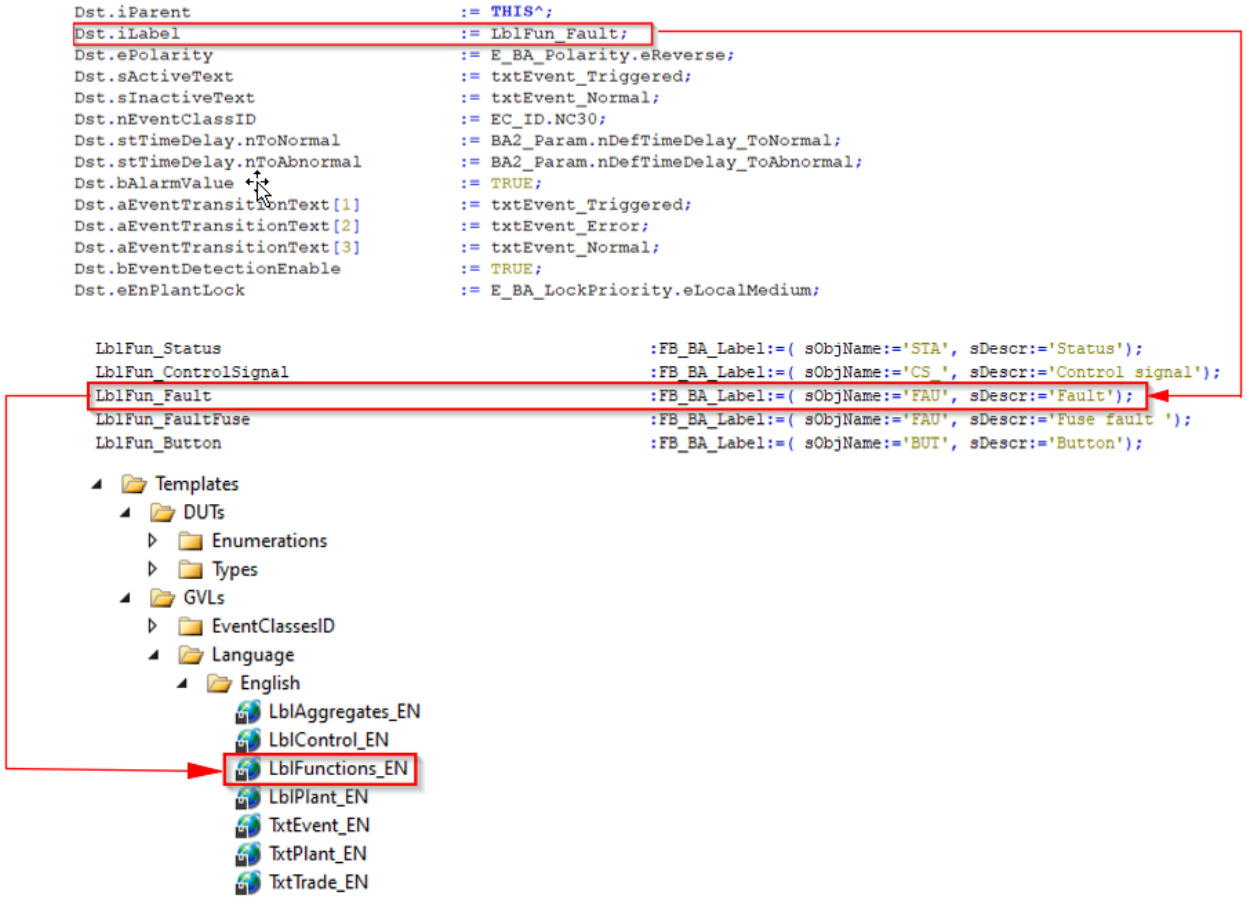
Eine weitere Möglichkeit für die Initialisierung der Texte von Namen und Beschreibung der Objekte sind Labels.

Ein Label ist die paarweise Zusammenfassung der Texte für den Namen und die Beschreibung eines Objekts. Die Initialisierung erfolgt mit dem Funktionsbaustein FB_BA_Label.

Der Funktionsbaustein FB_BA_Label2x ist dann einzusetzen, wenn die Texte der Ebenen Aggregat und Funktion des Datenpunktes gemeinsam zu initialisieren sind.

In TwinCAT 3 Building Automation sind die Labels in einer Liste globaler Variablen gespeichert.

Innerhalb der Templates von TwinCAT Building 3 Automation werden sie zur Initialisierung der Textvariablen *sObjectname* und *sDescription* verwendet.



Die Label-Listen stehen für die Benennung von Aggregaten (*LblAggregates_xx*), allgemeinen Regel- und Steuerungsfunktionen (*LblControll_xx*) und Funktionsbezeichnungen (*LblFunction_xx*) zur Verfügung.

Die Listen mit der Endung _EN enthalten Labels in englischer und die Listen mit der Endung _DE Labels in deutscher Sprache. Für die Auswahl der Sprache muss, die nicht gewünschte Sprache vor dem ersten Kompilieren der Steuerung auskommentiert werden.

DPAD-Mode

Für jede Ebene der Projektstruktur kann mittels des DPAD-Mode bestimmt werden, ob und welche Texte bei der Verkettung berücksichtigt werden sollen.

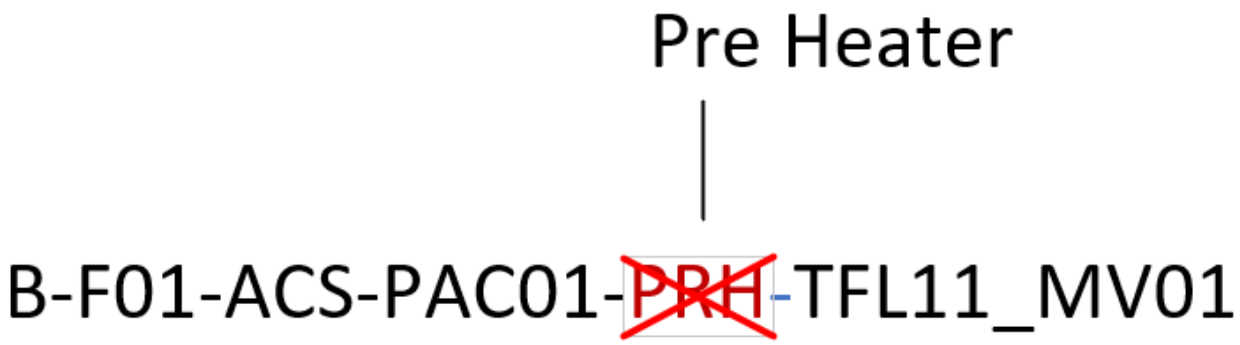
Im folgenden Beispiel erfolgt die Erklärung anhand einer Lüftungsanlage aus dem Template Repository von TwinCAT 3 Building Automation.

Die Hauptaggregate der Lüftungsanlage der Erhitzer (*PreHtr*), die Energierückgewinnung (*ERC*) und der Kühler (*Col*), haben jeweils einen eigenen Funktionsbaustein, innerhalb dessen alle Bestandteile bzw. Unteraggregate aufgerufen werden. Die Hauptaggregate sind nicht Bestandteil der Ebenen im DPAD. In der Definition des DPAD werden direkt hinter der Ebene Anlage die Ebenen für die Teilaggregate und Funktionen definiert.

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

In der SPS befindet sich dadurch eine Ebene mehr als im DPAD vorgesehen ist.

Die Ebene des Erhitzers ist deshalb bei der Erzeugung der Objektnamen auszuschließen.



Das geschieht mit dem DPAD-Mode und der folgenden Initialisierung.

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

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


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

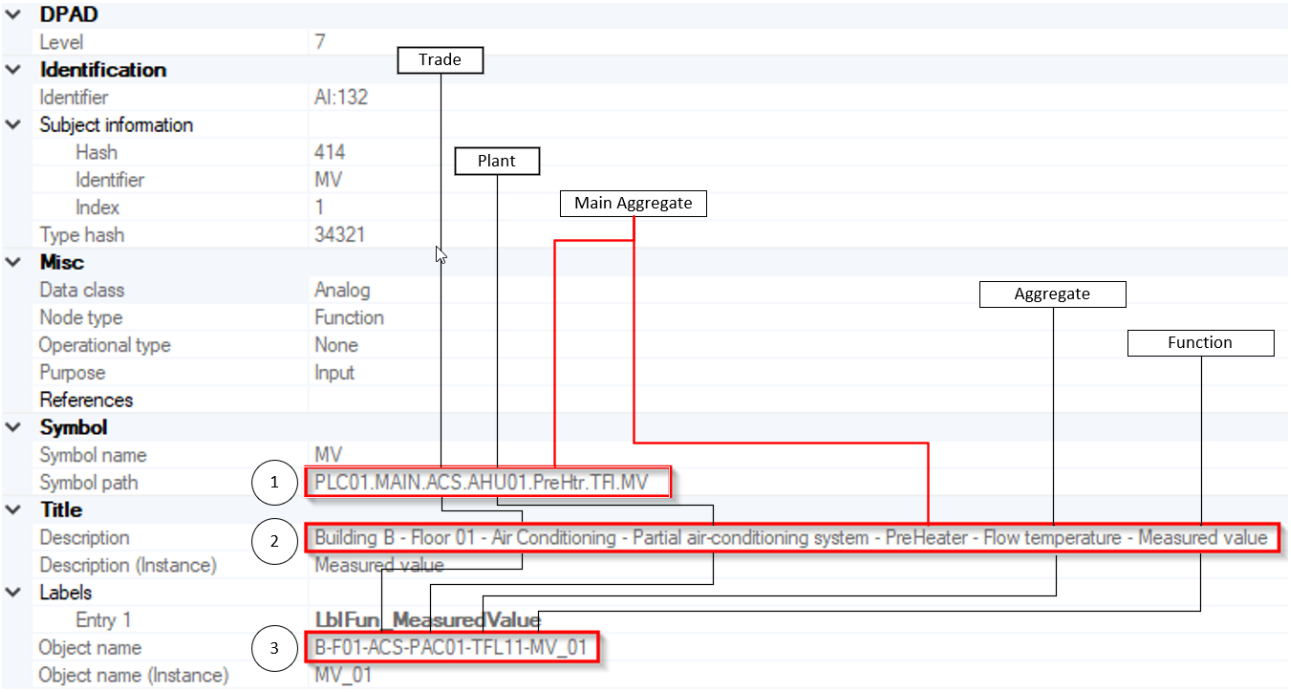
```
PreHtr.iParent                                          := THIS^;
PreHtr.iLabel                                           := LblCtl_PreHeater;
PreHtr.sObjectName                                      := '{Idx=01}';
PreHtr.eDPADMode                                        := E_BA_DPADMode.eIncludeDescription;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

Für den Erhitzer ist die Einstellung *eIncludeDescription* gewählt. Der Beschreibungstext des Erhitzers wird damit verkettet und der Name des Erhitzers bei der Erzeugung der Objektnamen ausgeschlossen.

Das Ergebnis dieser Parametrierung ist im Site Explorer zu sehen.

Der Symbolpfad (1) hat 7 Ebenen, was zur Folge hat das der Beschreibungstext (2) und der Objektname (3) ebenfalls 7 Ebenen haben, wobei beim Objektnamen eine Ebene nicht angezeigt wird.

In dem Beschreibungstext (2) des Vorlauftemperaturfühlers vom Lufterhitzer ist der Beschreibungstext *'PreHeater'* eingebunden und im Objektnamen (3) ausgeschlossen.



3.6.3.3 Objekt-Beschreibungstexte

Objekt-Beschreibungstexte werden genutzt, um eine eindeutige Benennung aller Objekte in der Projektstruktur [▶ 39] zu erzeugen. (Diese wird DPAD [▶ 43] oder auch BAS (Benutzer-Adress-Schlüssel) bezeichnet)

Jede Ebene innerhalb der Projektstruktur erhält einen Namen und einen Beschreibungstext.

Die Verkettung der Teilnamen aus der Projektstruktur ergibt den Namen und Beschreibungstext eines Objektes. Verkettete Texte entstehen unter Verwendung von Platzhaltern oder Labels.

Als Texte werden folgende Objekt-Parameter zusammengefasst:

- ObjectName
- Description

Die Texte eines Objektes [▶ 33] werden nur automatisch generiert, wenn ein Platzhalter enthalten ist. Das ist typischerweise der Fall, wenn die SPS das erste Mal startet.



Da von nun an sämtliche Texte einen definierten Wert haben (welcher auch persistent gespeichert wird) findet ein erneutes Generieren nicht mehr statt!

Texte werden ab diesem Zeitpunkt nur noch durch den Nutzer geändert!

Labels

Ein Label-FB (z.B. in einer GVL) kann bei der Deklaration von ein oder mehreren Objekten [▶ 33] verwendet werden, um dessen Texte automatisch zu generieren.

Es wird zwischen folgenden Label-Typen unterschieden:

- Einfache Labels (FB_BA_Label)
  Bieten die Möglichkeit zur Definition von Texten zur allgemeinen Verwendung.
  Das bedeutet, dass einfache Labels in allen Objekten, aber auch in weiteren Labels wiederverwendet werden können.
- Zweifache Labels (FB_BA_Label2x)
  Bieten die Möglichkeit zur Definition des Equipment-Identifiers eines Objektes.
  Somit können dann sowohl Aggregate- als auch Funktions-Informationen beschrieben werden.

Jedem Objekt [▶ 33] kann ein beliebiges (einfaches oder zweifaches) Label zugewiesen werden.

Jedem View [▶ 33] sollte ein einfaches Label zugewiesen werden welches als Aggregate-Informationen interpretiert wird. Funktions-Informationen werden nicht von Views unterstützt.

Programmierung

Platzhalter

Platzhalter stellen einen später generierten Wert dar. Beispielsweise werden die Parameter ObjectName und Description wie folgt mit Standardplatzhaltern initialisiert:

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

Bei der Objektinitialisierung analysiert der Textgenerierungsmechanismus einen Text auf definierte Platzhalter.

Nur wenn ein gültiger Platzhalter ermittelt wird, kann der Mechanismus ihn durch einen automatisch generierten Text ersetzen. Andererseits hätte ein manuell definierter Text einen Platzhalter ersetzt und würde nicht überschrieben.

Sobald die Texte generiert sind, enthalten Variablen keine Platzhalter mehr.

Auch wenn das Gerät neu gestartet wird und Parameter (persistente Variablen) gespeichert wurden, bleiben die aktuellen Texte bestehen (automatisch generiert, manuell definiert oder sogar zur Laufzeit von einem Bediener geändert) und werden nicht vom Textgenerierungsmechanismus überschrieben.

Einsatzzweck

- **Stellvertretend**
  Ein Platzhalter (Hier ein Index) wird stellvertretend für einen Wert eingesetzt:
  `sDescription := '{Idx}'`
- **Definierend**
  Ein Platzhalter (Hier ein Index) wird eingesetzt um einen bestimmten Wert zu definieren:
  `sDescription := '{Idx=10}'`

Indizierung

- **Automatische Indizierung**
  Für Objekte der letzten beiden Ebenen der Projektstruktur ist eine automatische Indizierung in der Projektvorlage eingestellt. D.h. dass der Parameter ObjectName zusätzlich zu einem definierten Namen auch eine fortlaufende Index Nummer erhält. Indices (z.B. für Aggregate) werden automatisch hinter den Text angehangen.
  Während Indices für den Parameter ObjectName im Normalfall immer erzeugt werden, erscheinen Description-Indices in Abhängigkeit der aktuellen Einstellung).
- **Manuelle Indizierung**
  Die Indizierung kann durch Platzhalter und deren entsprechende Index-Attribute vorgegeben werden.



Manuell definierte Indices (Unabhängig vom Einsatzzweck) werden immer im erzeugten Text angezeigt.

Ein vorgegebener Index wird automatisch auch für Parameter nachfolgender Objekte [▶ 33] verwendet, falls diese nicht auch explizit einen Index definieren.

Beispiel:

Da für Pump1 ein Index von 10 definiert wird, wird für Pump2 automatisch der Index 11 verwendet.

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

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

Manuell vergebene Indices müssen sorgfältig vergeben werden da diese nicht vom Mechanismus validiert werden!

Verkettung

Die Texte eines Objektes [▶ 33] werden unter bestimmten Bedingungen mit Texten von Parent-Objekten [▶ 31] verkettet:

- Parent-Einstellung [▶ 246]
- Einstellungen im verwendeten DPAD-FB [▶ 43]
- Manuelles Überschreiben mittels Index-Attributen
- Globale Einstellungen [▶ 289]
- Platzhalter mit Verkettungs-Verbot

Use-cases

- **Anwendungsfall 1**
  Ein Platzhalter bezieht sich auf einen bestimmten Parameter.
  Beispiel: Initialisierung des ObjectName
  `sObjectName := '{}'`
- **Anwendungsfall 2**
  Ein Platzhalter repräsentiert einen anderen Parameter.
  In diesem Fall muss der Platzhalter mit seinem entsprechenden Kürzel angesprochen werden.
  Beispiel:
  `sDescription := '{} - ID {InstID}'.`

Parameter

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

Attribute

Attribute werden typischerweise in der Applikation (Zum Beispiel in einem Template [▶ 709]) verwendet, da dort die Applikations-spezifischen Details bekannt sind, nicht aber die Betreiber-spezifischen Details.

Attribute	Shortcut	Type
Aggregate Index	`Idx`	Unsigned number
Function Index	`Idx`	Unsigned number

- **Index-Attribute**
  Ob ein Index sich auf ein Aggregat oder eine Funktion bezieht ist abhängig von der Reihenfolge.
  - Ein alleinstehender Index bezieht sich auf die Funktion.
  - Zwei Indices beziehen sich auf das Aggregat (Index 1) und die Funktion (Index 2)

Syntax

- Platzhalter:
  `{}`
- Platzhalter, der einen Parameterwert überschreibt (z.B. `sObjectName := '{MyValue}'`):
  `{Value}`
- Platzhalter, als repräsentierendes Attribut (z.B. `sDescription:= '{Idx}'`):
  `{Attribute}`
  `{Attribute1,Attribute(n),...}`
- Platzhalter, als definierendes Attribut (z.B. `sDescription:= '{Idx=10}'`):
  `{Attribute=Value}`
  `{Attribute1=Value,Attribute(n)=Value,...}`
- Platzhalter, als definierendes Attribut welcher einen Parameter Wert überschreibt (z.B. `sDescription:= '{Pump{Idx=10}}'`):
  `{Value{Attribute=Value}}`
- Platzhalter mit mehreren Attributen:
  `{Placeholder{Attribute1=Value,Attribute2=Value}}`
- Platzhalter, als repräsentierender Parameter (e.g. `sDescription:= '{} {InstID}'`):
  `{Parameter}`
- Platzhalter, der die Verkettung mit Elternelementen verweigert. Es wird der Einstiegspunkt der Verkettung definiert. Ab diesem Punkt werden nur Platzhalter der Kinderelemente verkettet:
  `!{}`
- Platzhalter, der bestimmte Attribute verweigert (ausschalten von Attributen):
  `{Value{!Attribute}}`

Beispiele

Parameter ObjectName mit definiertem Platzhalter (vertettet automatisch generierte Werte):
```
sObjectName := '{}'
```

Verkettet Parameter ObjectNamemit überschriebenen Wert:
```
sObjectName := '{AAA01_BB01}'
```

Verkettet Parameter Description mit Platzhalter und fest definierten Zusatzinformationen, die nicht ersetzt werden:
```
sDescription := '{} North'
```

Verkettet Parameter Description mit einem Platzhalter und erweitert ein stellvertretendes index-attribut (Es wird der Indexwert vom ObjectName übergeben):
```
sDescription := '{Idx}'
```

Verketteter Parameter ObjectName mit Platzhalter inklusive definierendem Index-Attribut:
```
sObjectName := '{Idx=10}'
```

Verketteter Parameter ObjectName mit Platzhalter inklusive stellvertretendem (Aggregat) und definiertem (Funktion) Index-Attribut:
```
sObjectName := '{Idx,Idx=10}'
```

Verketteter Parameter Description mit überschriebenem Wert und einem definiertem (Funktion) Index-Attribut:
```
sDescription := '{Pump{Idx=5}}'
```

Verketteter Parameter Description mit Platzhalter und einem stellvertretenden Parameter:
```
sDescription := '{} - ID {InstID}'
```

ObjectName mit Standardplatzhalter aber ohne Verkettung (Verkettung verweigert):
`sObjectName := '!{}'`

Verketteter Parameter ObjectName mit überschriebenem Wert, aber ohne Index-attribut (Attribut verweigert):
`sDescription := '{PU{!Idx}}'`

Fehlermeldungen

Bezeichnung	Text	Beschreibung
[T23]	Internal error on parsing placeholder "...": Defined value and attributes at once!	Das gleichzeitige Definieren von Platzhalter-Werten und -Attributen ist unzulässig da ein Definierter Wert das Anwenden von Attributen verhindert.

Generieren von Texten

Die folgende Reihenfolge wird verwendet, um zu entscheiden, wie ein Platzhalter ersetzt wird:

1. Texte überschreiben
   Falls definiert, wird ein manuell initialisierter Titel verwendet:
   `fbSensor : FB_BA_AO_IO := (sObjectName:='Tmp03', sDescription:='Temperature sensor 3');`
2. Labels
   Wenn Titel nicht überschrieben werden, kann ein Label angewendet werden:
   `SomeLabel : FB_BA_Label := (sName:='PU', sDescription:='Pumpe');`
   `Pump : FB_BA_View := (iLabel:=SomeLabel);`
3. SymbolName
   Wenn nichts definiert ist, werden die folgenden Standardwerte verwendet:
   - ObjectName erhält die Information vom symbolpath.
   - Description erhält die Information vom symbolname.

Samples

Initialisierung mittels Labels

Übertragung der, im Label definierten Texte, auf Objekte [▶ 33].

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

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

3.7 SPS

3.7.1 Steuer- und Regelmechanismen

In diesem Kapitel werden allgemeine Zusammenhänge bezüglich der Steuerung und Regelung erläutert.

3.7.1.1 Zeitschaltplanung

Die zeitliche Steuerung von TGA- und Raumautomationsanlagen ist ein wichtiger Bestandteil der Gebäudeautomatisierung. Durch den gezielten Einsatz von Zeitschaltplänen lassen sich die Effizienz des Gebäudes und der Komfort für die Nutzer optimieren. Wird z. B. eine Heizungsanlage über einen Zeitschaltplan gesteuert, kann die Raumtemperatur an die Nutzungszeiten angepasst werden, wodurch Heizenergie eingespart werden kann, wenn das Gebäude nicht genutzt wird.

Die Umsetzung von Zeitschaltplänen mit TF8040 ist in diesem Kapitel beschrieben.

Planung

Generell gibt es verschiedene Möglichkeiten, einen Zeitschaltplan einzustellen.

Wöchentliche Planung

Zunächst gibt es den Wochenzeitschaltplan. Dies ist der Zeitschaltplan, der jede Woche gilt, er beschreibt also eine allgemeine Woche.

Eingestellt wird er durch den Parameter aWeek [▶ 261] der Bausteine, FB_BA_SchedA [▶ 227], FB_BA_SchedB [▶ 228] und FB_BA_SchedM [▶ 230].

Ausnahmen

Der wöchentliche Zeitschaltplan, der in Standardwochen, bzw. im normalen Betrieb, des Gebäudes gilt, muss immer wieder durch Ausnahmen abgeändert werden. Beispiele für solche Ausnahmen wären:

- Ferien
- Feiertage
- Buchung von Besprechungsräumen
- etc.
  Die Ausnahmen, die für einen Zeitschaltplan definiert werden können, werden in zwei Kategorien unterteilt.

Lokale Ausnahmen

Die lokalen Ausnahmen eines Zeitschaltplans sind Ausnahmen, die nur für diesen expliziten Zeitschaltplan gelten. D.h. sie haben keine Auswirkungen auf andere Zeitschaltpläne und werden direkt am Zeitschaltplan-Objekt eingestellt. Sie werden über den Parameter aException [▶ 261] der Bausteine, FB_BA_SchedA [▶ 227], FB_BA_SchedB [▶ 228] und FB_BA_SchedM [▶ 230] parametriert.

Die Ausnahmen können wie folgt konfiguriert werden:

- **Datum:**
  Es kann eine Ausnahme an einem bestimmten Datum definiert werden. An diesem Datum kann dann eingestellt werden, was zu welcher Uhrzeit passieren soll. Z. B. könnte die Heizung an diesem Tag nicht um 5 Uhr abends, wie im wöchentlichen Plan, sondern erst um 8 Uhr abends, aufgrund einer längeren Besprechung, heruntergestellt werden.
- **Datumsbereich:**
  Ausnahmen können über einen kompletten Datumsbereich festgelegt werden. Ist z. B. geplant das Gebäude in einer bestimmten Woche, aufgrund von Betriebsferien, nicht zu nutzen, so kann eingestellt werden, dass der wöchentlichen Zeitschaltplan so überschrieben wird, dass das Gebäude in der definierten Zeit nicht in den Komfort-Modus zu den eingestellten Betriebszeiten wechselt und somit Energie eingespart wird.
- **Woche und Tag:**
  Neben einem Datumsbereich können auch wiederkehrende Ausnahmen an bestimmten Wochen zu bestimmten Tagen definiert werden. Damit können Ausnahmen wie z.B. "jeden zweiten Mittwoch im Monat" realisiert werden.

Globale Ausnahmen

Globale Ausnahmen werden genauso durch Datum, Datumsbereich und Woche und Tag definiert. Wie der Name jedoch vermuten lässt, gelten diese Ausnahmen nicht für einen spezifischen Zeitschaltplan, sondern global. Globale Ausnahmen werden über Kalender-Objekte definiert, die dann bestimmten Zeitschaltplänen über den Parameter aCalendar [▶ 261] übergeben werden.

So ein Kalender könnte z. B. die Schulferien in dem jeweiligen Bundesland definieren. Somit könnten alle Zeitschaltpläne in einer Schule, während der Ferien überschrieben werden.

Programmierung

Die Funktionsweise von Zeitschaltplänen und deren Ausnahmen sollten an dieser Stelle verstanden sein und es wird nun auf die Programmierung eingegangen.

Wöchentlicher Zeitschaltplan

Der Parameter aWeek [▶ 261] ist ein Array und somit ist es ziemlich kompliziert ihn manuell zu parametrieren. Um es für den Anwender einfacher zu machen, wird dafür der Parameter-Builder F_BA_WeeklyScheduleBuilder [▶ 193] angeboten. Die Verwendung dieses Parameter-Builders wird in dessen Dokumentation beschrieben.

Lokale Ausnahmen

Der Parameter aException [▶ 261] ist ein Array und somit ist es ziemlich kompliziert ihn händisch zu parametrieren. Um es für den Anwender einfacher zu machen, wird dafür der Parameter-Builder F_BA_ExceptionScheduleBuilder [▶ 185] angeboten. Die Verwendung dieses Parameter-Builders wird in dessen Dokumentation beschrieben.

Globale Ausnahmen

Der Parameter aCalendar [▶ 261] ist ein Array und somit ist es ziemlich kompliziert ihn händisch zu parametrieren. Um es für den Anwender einfacher zu machen, wird dafür der Parameter-Builder F_BA_ScheduleCalendarBuilder [▶ 192] angeboten. Die Verwendung dieses Parameter-Builders wird in dessen Dokumentation beschrieben.

3.7.2 Schnelle Task für serielle Kommunikation

In der Gebäudeautomation finden verschiedene Bussysteme, wie zum Beispiel DALI, MBus oder EnOcean Anwendung. Viele dieser Systeme werden durch die Hardware von Beckhoff eingebunden und basieren auf einer seriellen Kommunikation. Die Kommunikation mit der seriellen Hardware benötigt oft eine Task, welche in deutlich kürzerer Zeit abgearbeitet wird, als die PLC-Task, in der die normale Applikation läuft.

Zusätzlich muss die schnelle Task eine höhere Priorität als die PLC-Task haben.

Dieser Zusammenhang ist in der Einführung zur seriellen Kommunikation gut beschrieben:

Innerhalb von TF8040 benötigen derzeit folgende Komponenten eine schnelle Task zur Abarbeitung serieller Informationen:

- die Thies Wetterstation, die auf einer seriellen Standard-Kommunikation mit 9600 bps basiert, (siehe Einbindung Thies Wetterstation [▶ 1116]).
- die DALI-Funktionsbausteine, welche auf der Bibliothek TC3_DALI basieren. Die Einbindung der Kommunikation ist im folgenden Kapitel beschrieben, (siehe DALI-Kommunikation [▶ 881]).

3.8 HMI

3.8.1 Generik

Die Erstellung eines HMIs für eine Gebäudeautomation kann sehr aufwendig sein und es können nur vereinzelnd Teile eines HMIs in verschiedenen Projekten wiederverwendet werden. Oft ist auch das Budget, welches für ein HMI verfügbar ist, begrenzt. Aus diesem Grund lohnt es sich ein HMI zu nutzen, das gewisse Teile bereits generisch erzeugt.

Diese Generik ist mit der Lösung *TF8040 Building Automation* möglich. Das bedeutet die Tc3_XBA [▶ 101] Bibliothek für die SPS und die BaSite-Extension [▶ 1283] für das HMI.

Das Ziel von TF8040 ist es, dass der Integrator seine Anlage nur an einer zentralen Stelle entwickelt - der SPS.

Aufgrund der Art und Weise, wie die Implementierung von Anlagen mit TF8040 geschieht und welche Strukturen sich daraus ergeben, ist es möglich generische Funktionen für das HMI abzuleiten.



Weitere Informationen sind in der Dokumentation zu Generische HMI [▶ 79] zu finden.

Projektstruktur

Grundstein für die generischen Funktionen innerhalb von *TcHmiBa* ist die Projektstruktur [▶ 39], die in der SPS durch eine Child/Parent-Beziehung hergestellt wird. Durch diese Struktur ist es möglich eine generische Navigation [▶ 79] durch alle Objekte auf einer Steuerung abzuleiten.



Weiterführende Informationen zur Projektstruktur [▶ 39] sind in der Dokumentation zu finden.

Events

Es können die Events aller verknüpften Steuerungen gesammelt und zentralisiert in einer Eventliste [▶ 1185] angezeigt werden. Diese Funktion ergibt sich ebenfalls bereits aus dem Engineering in der SPS und bedarf keiner weiteren Konfiguration im HMI.

Trending

Die Generik ermöglicht es zudem verschiedene Trendfunktionen zu realisieren, ohne dass diese extra für das HMI konfiguriert werden müssen.



Weitere Informationen sind in der Dokumentation Trending [▶ 56] zu finden.

3.8.2 Trending

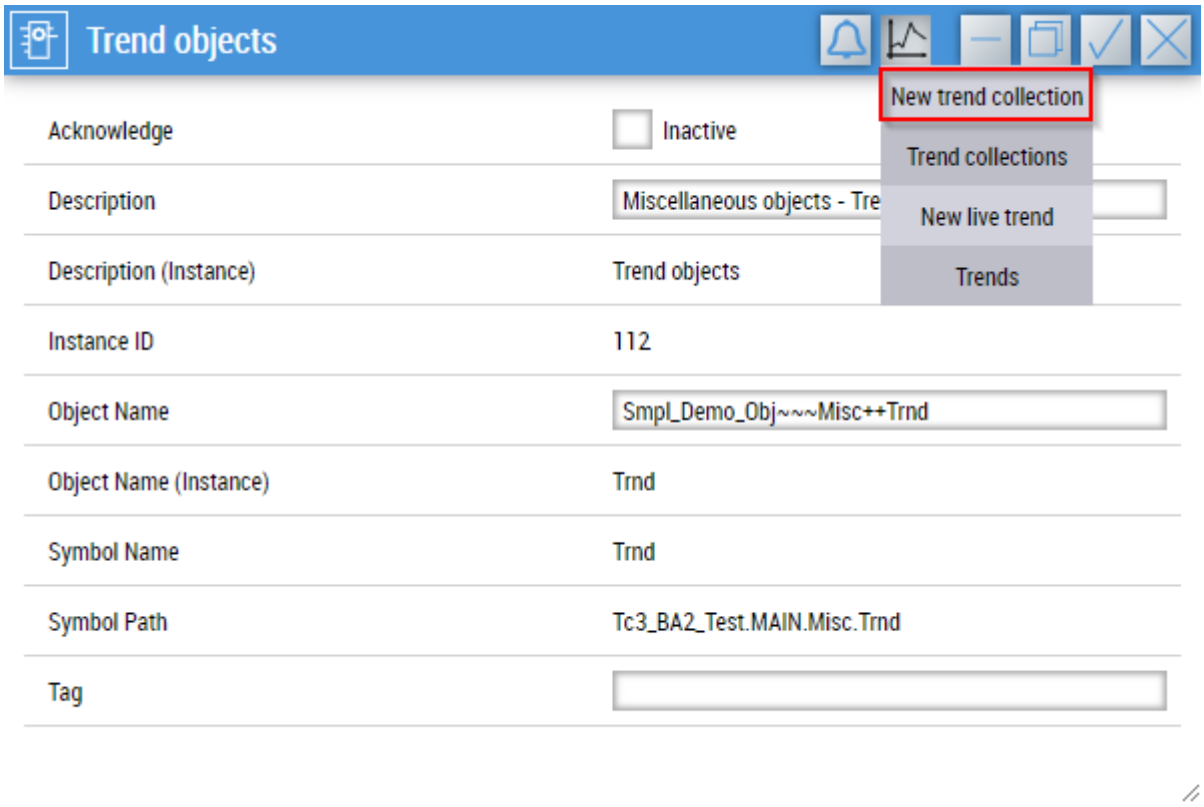
Im HMI können Trends verwendet und verwaltet werden. Die hier beschriebenen Funktionen lassen sich nur mit den generischen Funktionen [▶ 79] von TcHmiBa nutzen.

Trendsammlungen

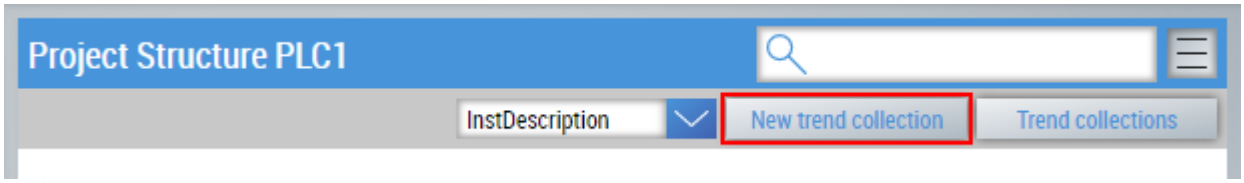
Das Anzeigen von einzelnen Trendkurven ist oft nicht ausreichend, da sie so nur schwer vergleichbar sind. Aus diesem Grund gibt es die Möglichkeit Trendsammlungen anzulegen, um die Trendkurven in einem Chart vergleichen zu können.

Konfigurator

Der Konfigurator lässt sich über das Menü im Parameterfenster [▶ 1189] eines Views öffnen und zeigt dann nur Objekte ab dieser Ebene an.

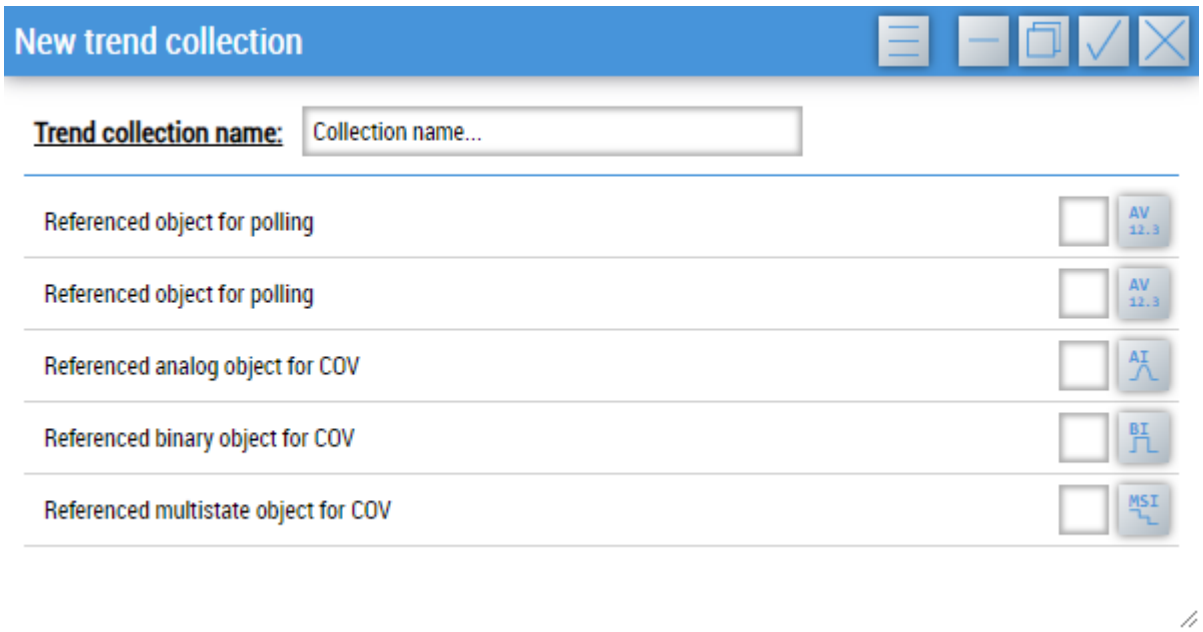


Zusätzlich kann er auch über das Menü der ProjectNavigationTextual [▶ 1186] geöffnet werden.

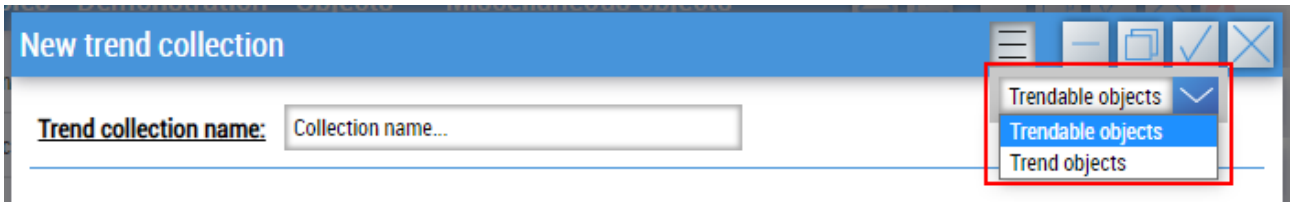


Anlegen von Trendsammlungen

Trendsammlungen werden mit dem Konfigurator angelegt. Nach Vergabe eines Namens lassen sich über die Checkboxen die trendbaren Objekte der Trendsammlung hinzufügen.

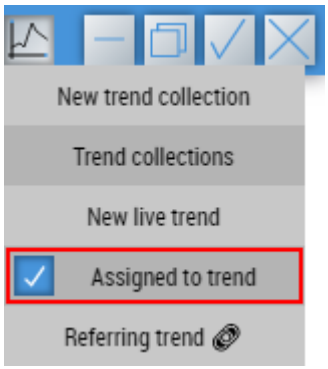


Das Menü bietet dabei die Möglichkeit die anzuzeigenden Objekte auszuwählen.

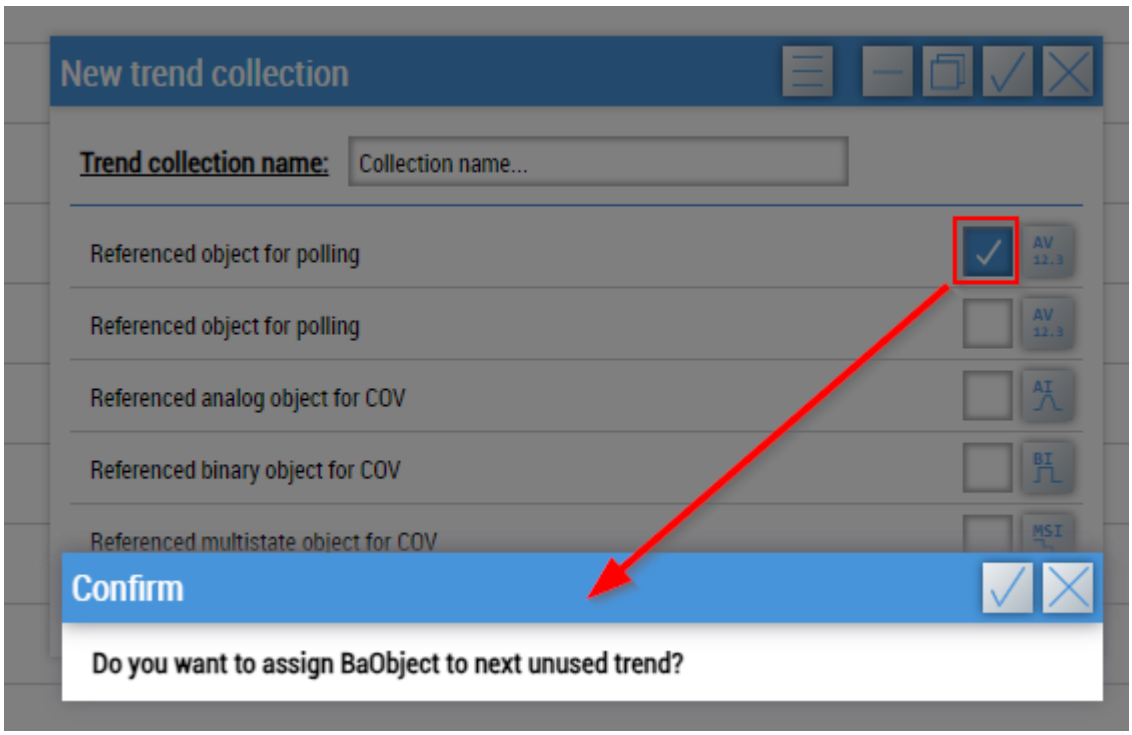


- **Trendable object:** Zeigt alle trendbaren Objekte (z.B. *FB_BA_AV*) an.
- **Trend object**: Zeigt alle Trendobjekte (*FB_BA_Trend*) an.

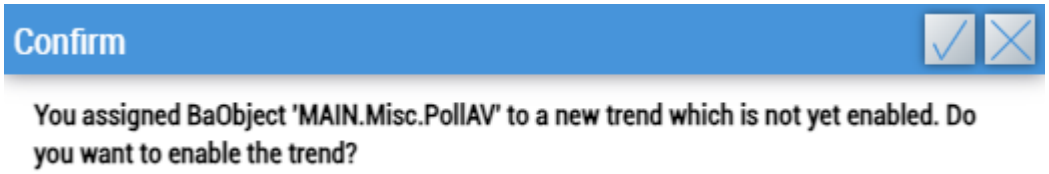
Ein **Trendable object** muss zur Verwendung im Trend einem Trendobjekt zugeordnet sein. Der Zustand dazu lässt sich im Menü des Parameterfensters [▶ 1189] einsehen und auch ändern.



Sollte eine Zuordnung zum Zeitpunkt der Auswahl noch nicht erfolgt sein, so geschieht dies automatisch durch das Bestätigen der Abfrage mit **OK**.



Im Anschluss kann der Trend mit **OK** direkt aktiviert werden.



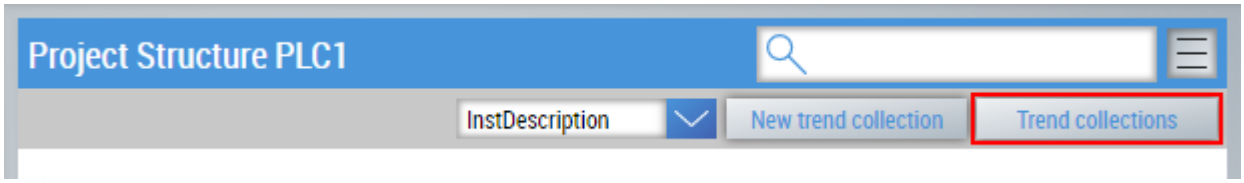
Wenn die Trendsammlung vollständig ist, steht diese nach dem Bestätigen des Dialogs in der Trendsammlungsansicht zur Verfügung.

Beobachten von Trendsammlungen

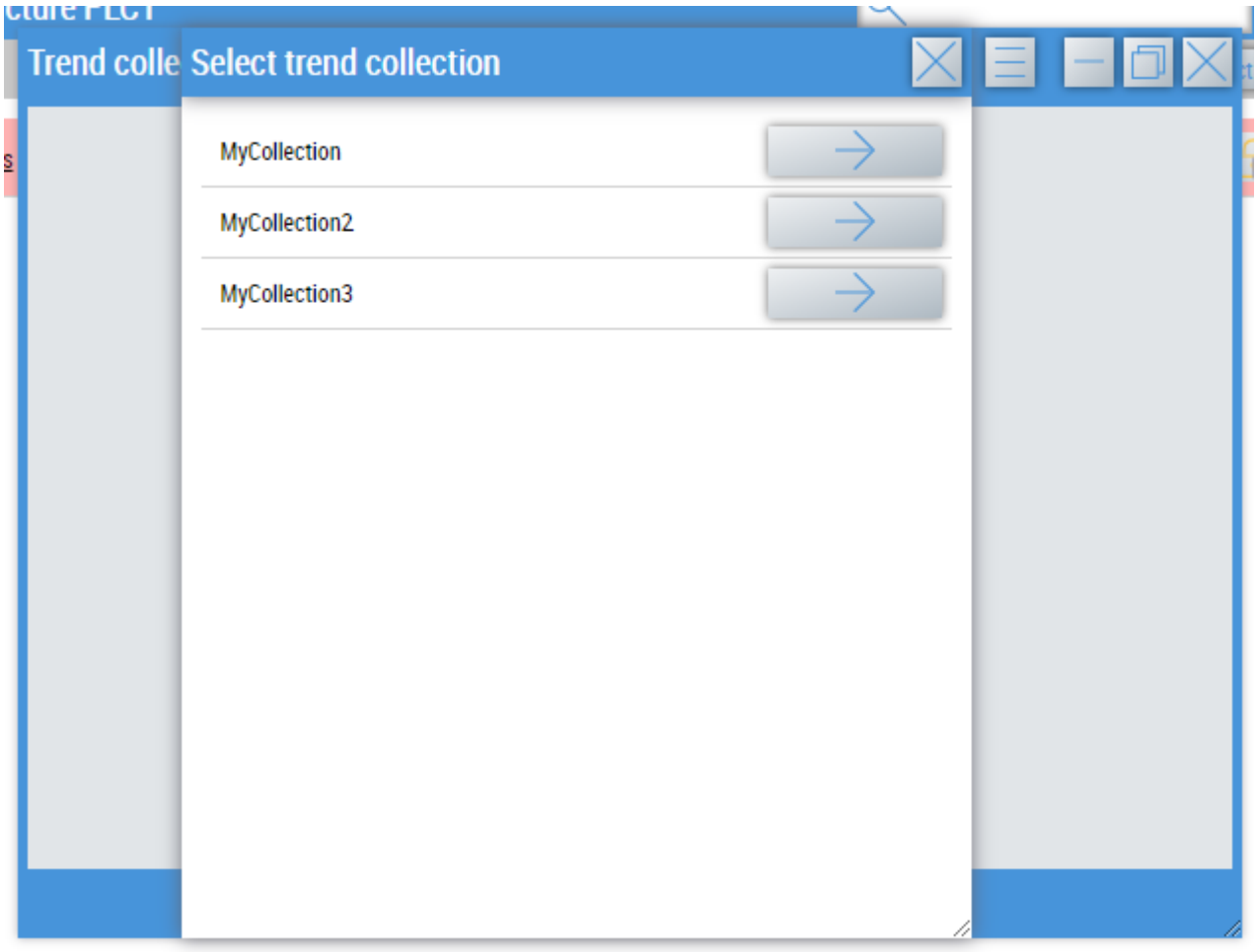
Die erstellten Trendsammlungen lassen sich in der Trendsammlungsansicht anzeigen und hinzufügen.

Diese lässt sich über zwei Wege aufrufen.

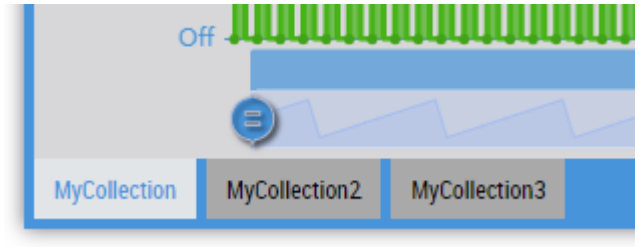
1. Durch Aufrufen der Funktion OpenTrendCollectionView [▶ 1279].
2. Aus der ProjectNavigationTextual [▶ 1186].



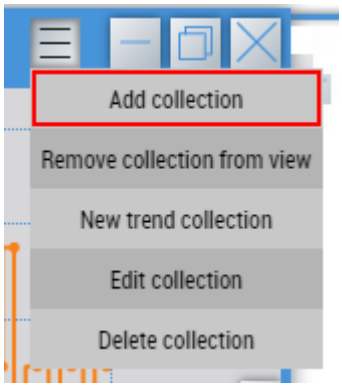
Wird der Dialog das erste Mal geöffnet, erscheint eine Übersicht. Aus ihr kann eine Trendsammlung ausgewählt und anschließend in einem Trend Control [▶ 1194] angezeigt werden.



Die Organisation erfolgt in Tabs und ermöglicht einen schnellen Wechsel zwischen den verschiedenen Trendsammlungen.



Weitere Trendsammlungen lassen sich über das Menü hinzufügen.



Zum Verwalten der Trendsammlung stehen im Menü weitere Aktionen zur Verfügung.

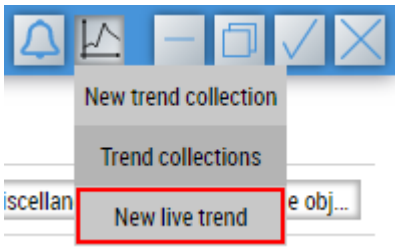
- **Remove collection from view**: Entfernt die Trendsammlung im aktiven Tab aus der Ansicht (wird nicht gelöscht).
- **Edit collection**: Öffnet den Konfigurator zum Bearbeiten der Trendsammlung im aktiven Tab.

- **New trend collection**: Erstellt eine neue Trendsammlung. An dieser Stelle werden alle Objekte der Projektstruktur angezeigt. Der erstmalige Aufruf kann daher etwas Zeit in Anspruch nehmen.
- **Delete collection**: Öffnet die Trendsammlungsansicht. Die ausgewählte Trendsammlung wird permanent gelöscht.

Live-Trends

Soll das Verhalten eines Wertes nur kurzweilig beobachtet und dabei keinem Trendobjekt zugeordnet werden, gibt es die Möglichkeit über das Menü des Objekts im Parameterfenster [▶ 1189] einen Live-Trend anzulegen.

Diese Aktion ist nur verfügbar, wenn das Objekt selbst trendbar (z.B. FB_BA_MV [▶ 242]) oder ein View mit trendbaren Objekten ist.

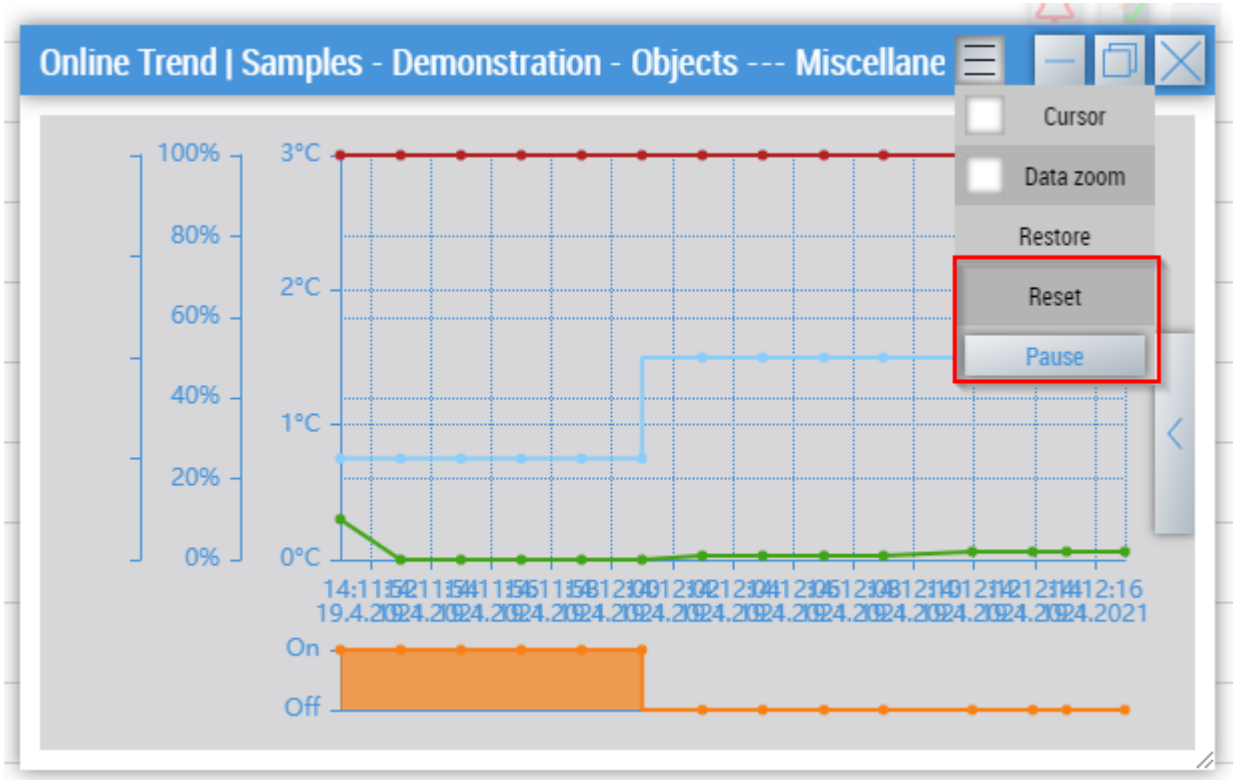


Nach Betätigen der Schaltfläche öffnet sich der Live-Trend, der entweder nur das eine trendbare Objekt oder alle trendbaren Objekte des Views in einem FIFO-Speicher aufzeichnet. Die Funktionen im Menü ähneln dabei denen aus dem Trend Control [▶ 1194].

HINWEIS

Datenverlust

Wenn das Fenster geschlossen wird, gehen die aufgezeichneten Daten verloren.



Das Menü enthält zwei zusätzliche Aktionen:

- *Reset*: Löscht alle aufgezeichnet Werte.
- *Pause*: Pausiert die Aufzeichnung neuer Werte.

3.8.3 Feedback

In einigen TcHmiBa-Controls [▶ 1141] wird mit einem sogenannten *Feedback* gearbeitet.

Das Feedback soll helfen, dem Anwender eine Rückmeldung geben zu können, wenn z. B. ein BV geschrieben wird, aber die Bestätigung (Feedback) der SPS noch aussteht.

Die TcHmiBa-Controls zeigen entsprechend immer nur den Feedback-Wert (z. B. *StateFeedback* der *Checkbox*) an, wodurch sichergestellt wird, dass im HMI immer der aktuelle Wert zur Anzeige kommt.



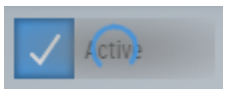
Eine Wertänderung von *State* hat keine Auswirkung auf die Anzeige im HMI, da nur der Wert von *StateFeedback* angezeigt wird.

Ablauf

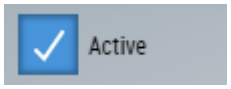
Der Ablauf am Beispiel der Checkbox [▶ 1154] sieht folgendermaßen aus:



1. Wert schreiben durch Markieren der Checkbox
2. Wert wird auf das Symbol *State* geschrieben
3. Anzeige der Ladeanimation



4. Antwort der SPS:
   Ausblenden der Ladeanimation



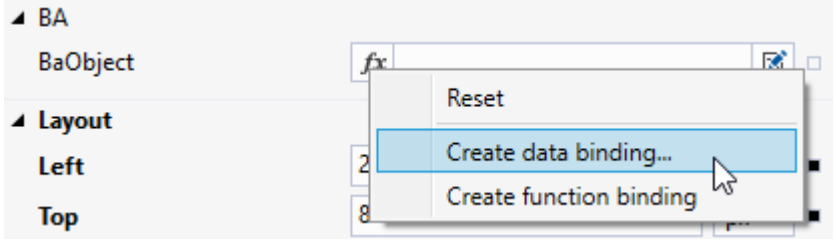
Es gibt nur zwei mögliche Ausgänge.
a. Wert des Symbols, das mit *StateFeedback* verknüpft ist, ändert sich auf denselben Wert, der auf *State* geschrieben wurde.
b. Wert von *StateFeedback* ändert sich nicht.

3.8.4 BaObject

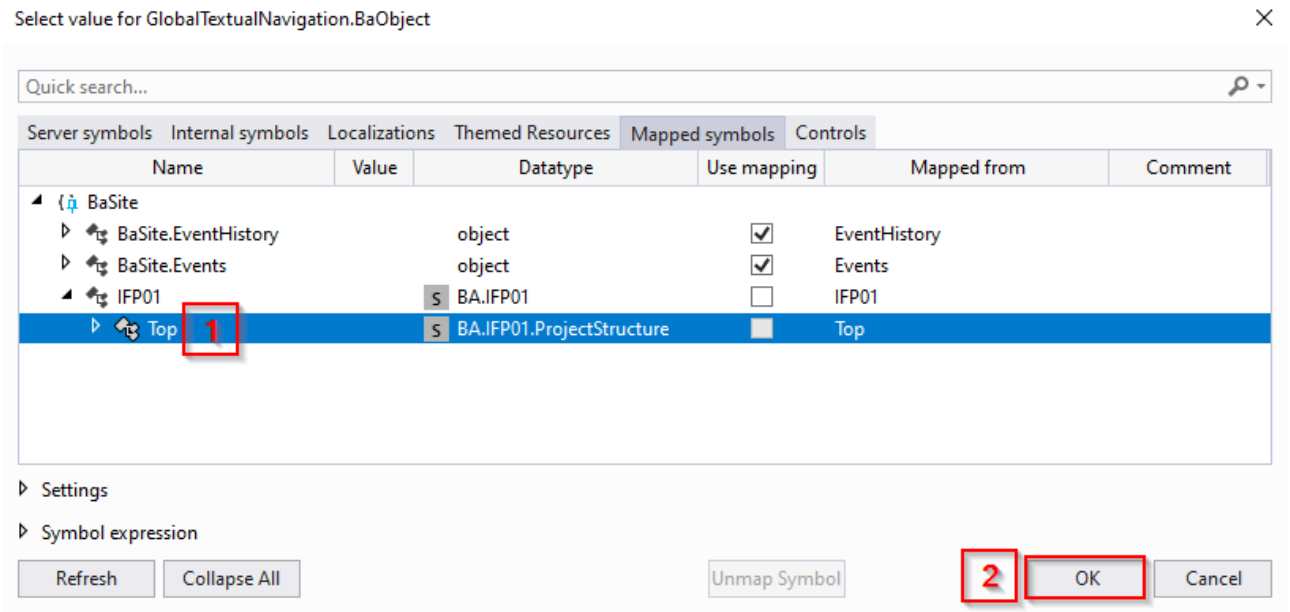
Die BaSite-Extension [▶ 1283] stellt der TcHmi die Struktur der SPS in einer speziellen Form zur Verfügung. Aus dieser werden BaObjects gebildet, die sich anschließend mit dem Attribut BaObject aus der Kategorie BA eines TcHmiBa-Controls [▶ 1141] verknüpfen lassen.

Binding erstellen

Aktion zum Binding erstellen aus dem Menü aufrufen.



Symbol auswählen.



Symbol ist zugewiesen.



BaObject

Ein BaObject ist die Repräsentation eines Objektes aus der Projektstruktur [▶ 39] einer SPS, die auf TF8040 basiert. Die TcHmiBa-Controls nutzen aus den BaObjects generisch die Daten und Funktionen, die sie für ihre Arbeit benötigen.

Weitere Informationen sind in der Dokumentation zur BaObject-Handhabung [▶ 1248] zu finden.

3.8.5 BaTemplate

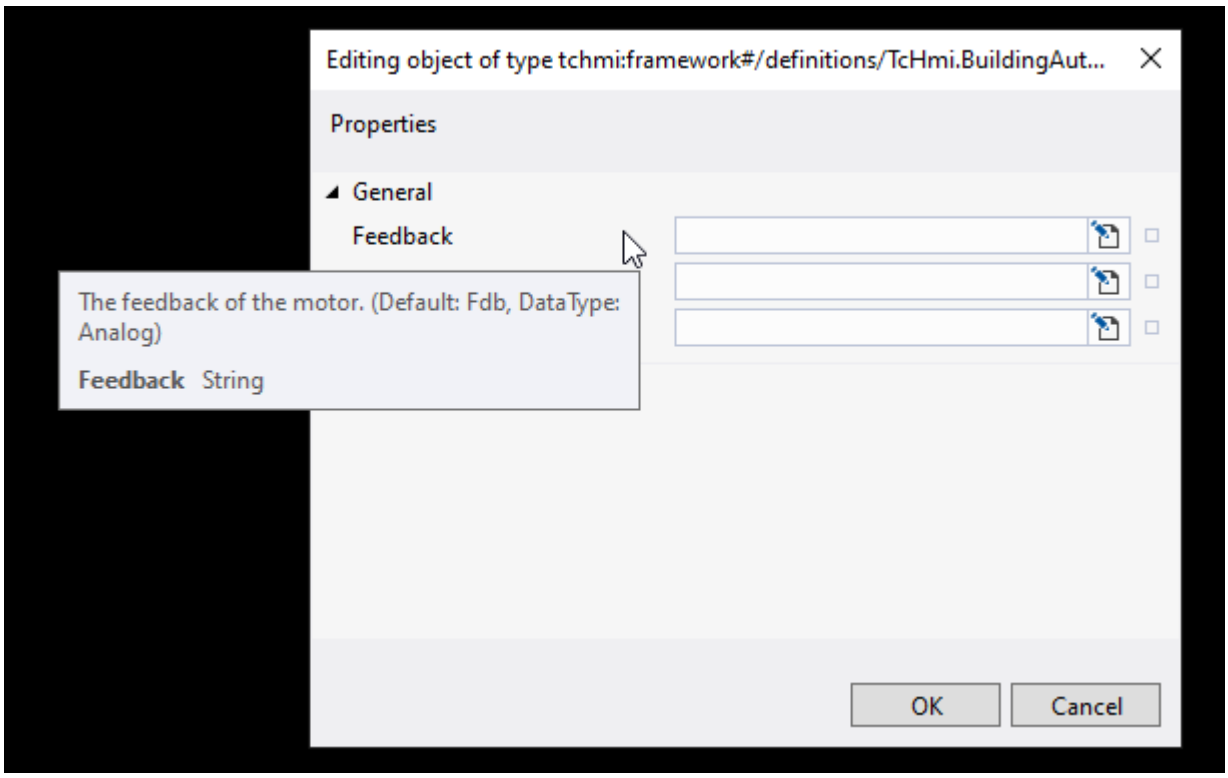
Als TcHmiBa-Templates werden TcHmiBa-Controls [▶ 1141] bezeichnet, die unmittelbar ein TF8040 Standard SPS-Template in der TcHmi repräsentieren.

BaTemplateDescription

Für nahezu alle TF8040 Standard SPS-Templates sind TcHmiBa-Templates vorhanden. Zum Funktionsumfang gehört eine Validierung des verknüpften BaObjects [▶ 62] anhand der BaTemplateDescription. Sie definiert die Symbolnamen samt Hierarchie im Auslieferungszustand, die ein TcHmiBa-Template im BaObject benötigt.

Im Fall von geänderten Symbolnamen [▶ 1250] aufgrund projektspezifischer Eigenheiten kann der neue Name dem TcHmiBa-Template über das Attribut *BaTemplateDescription* aus der Kategorie BA mitgeteilt werden.

Die Standardwerte der BaTemplateDescription, sowie die erwarteten Datentypen sind im Tooltipp des Dialoges zum Einstellen der BaTemplateDescription zu finden.



Weitere Informationen sind in der Dokumentation zur BaTemplate-Handhabung [▶ 1248] zu finden.

Verschachtelte BaObjekte verwenden

Befinden sich die BaObjekte z. B. für den Command von einem Motor nicht auf der standardmäßig vorgesehenen Ebene, sondern in einer tieferen Ebene, so kann auf diese verschachtelten Objekte mit der folgenden Syntax zugegriffen werden.

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

3.8.6 BaInterface

TcHmiBa-Controls [▶ 1141], die das BaInterface unterstützen, lassen sich auch ohne die BaSite und TF8040 SPS-Templates verwenden. Funktionsumfang und Verhalten kann dabei Abweichungen unterliegen.

Weitere Informationen sind in der Dokumentation zum BaInterfaceHandler [▶ 1266] zu finden.

Verwendung

Das Attribut BaInterface aus der Kategorie BaData erfordert ein TcHmi-Symbol.

Ohne Vorgabe von Struktur und Inhalt durch ein TF8040 SPS-Template, muss die Zuweisung der Variablen für das Control über das Attribut *BaInterfaceSymbolNames* aus der Kategorie *BaData* erfolgen.

Beispiel

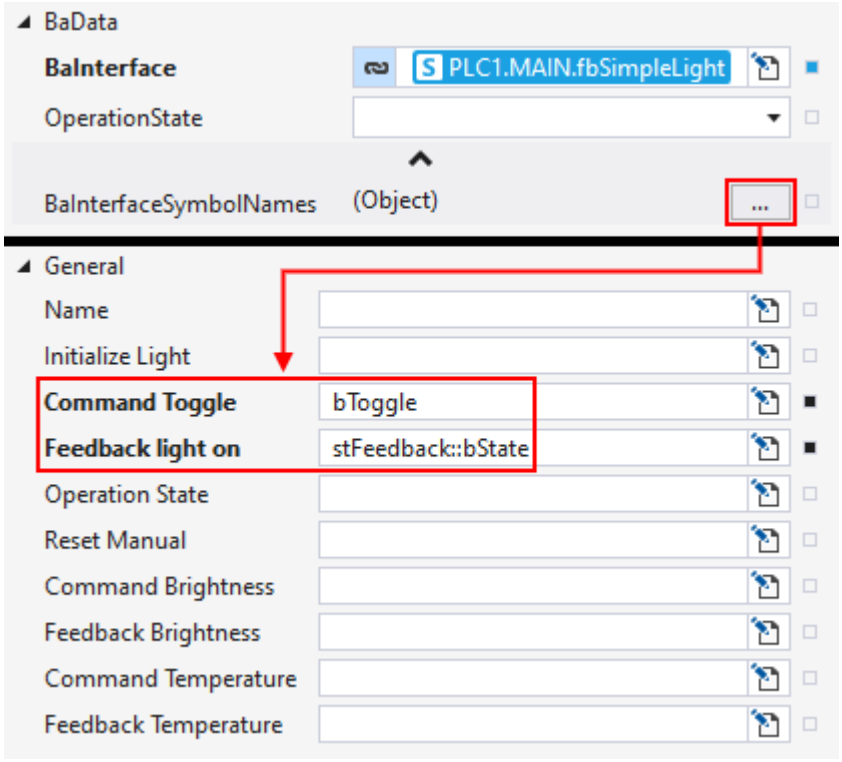
Erläuterung am Beispiel eines selbsterstellten SPS-Funktionsbausteins *FB_SimpleLight* zum Toggeln eines Lichts. Die Zustandsvariable *bState* ist in einer Struktur hinterlegt.

```
FUNCTION_BLOCK FB_SimpleLight
VAR_INPUT
  bToggle       : BOOL;
END_VAR
```

```
VAR_OUTPUT
  stFeedback    : LightFeedback;
END_VAR
```

Dieser soll vom TcHmiBa-Control Light [▶ 1222] verwendet werden.

Hierzu wird das Symbol des Funktionsbausteins dem BaInterface-Attribut übergeben und anschließend die Variablen über das BaInterfaceSymbolNames-Attribut zugewiesen.



Verschachtelte Symbole verwenden

Auf verschachtelte Symbole kann mit folgender Syntax zugegriffen werden.

```
stFeedback::bState
```

4 Tutorials

Im Folgenden werden einige Tutorials vorgestellt, die den Einstieg in TF8040 vereinfachen sollen.

Getting started

SPS

Anlegen eines TwinCAT SPS-Projektes [▶ 66].

HMI

Anlegen eines TcHmiBa-Projektes [▶ 72].

Video-Tutorials

Der Einstieg in TF8040 wird durch eine Video-Tutorialserie weiter vereinfacht. Die Videos sind über die Beckhoff-Website abrufbar.

Filter: Branche -> Gebäudeautomation

4.1 SPS

Kapitel mit Anleitungen zur Programmierung von SPS-Anwendungen.

4.1.1 Start eines Projekts

Dieses Kapitel beschreibt, wie Sie ein Projekt starten.

Vorgehen

Die hier beschriebene Vorgehensweise beinhaltet sämtliche Einstellungen, die für ein funktionierendes Projekt relevant sind.

> Bei der Verwendung der TF8040 SPS-Projektvorlagen [▶ 706] sind einzelne Schritte bereits entsprechend vorbereitet.

Aktualisieren der Runtime

Falls die Runtime auf dem Zielgerät nicht aktuell ist, sollte diese entsprechend aktualisiert werden:

XAR

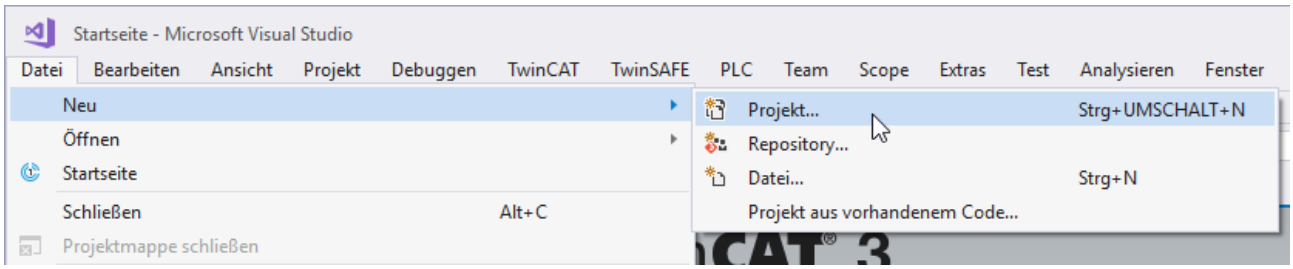
Aktuelle XAR auf vollen Windows-Systemen installieren.

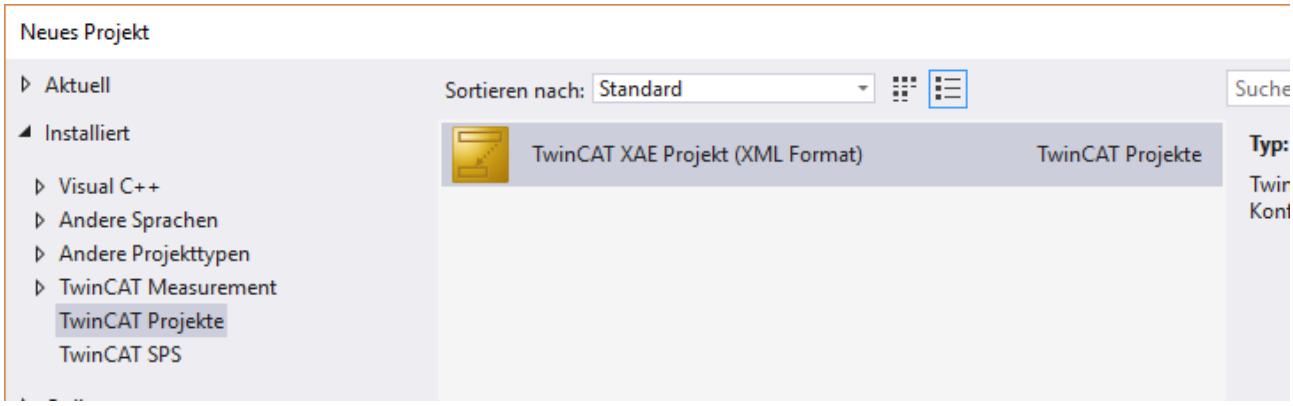
Image

Aktuelles Image auf den sonstigen Systemen (z. B. Windows Compact 7) installieren.

TwinCAT-Projekt erstellen

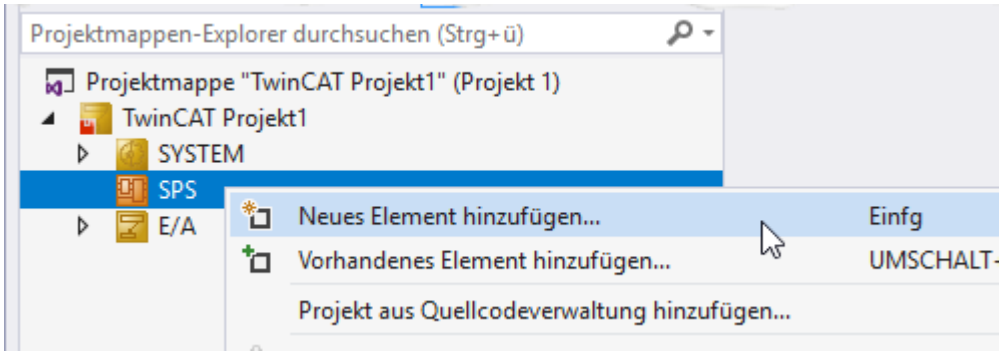
- Neue Solution erzeugen.

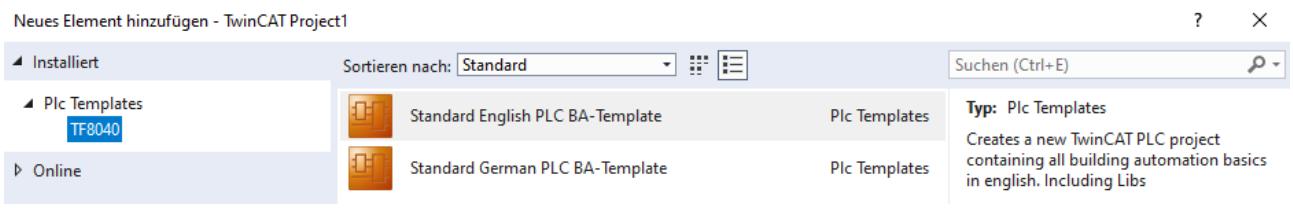




SPS-Projekt hinzufügen

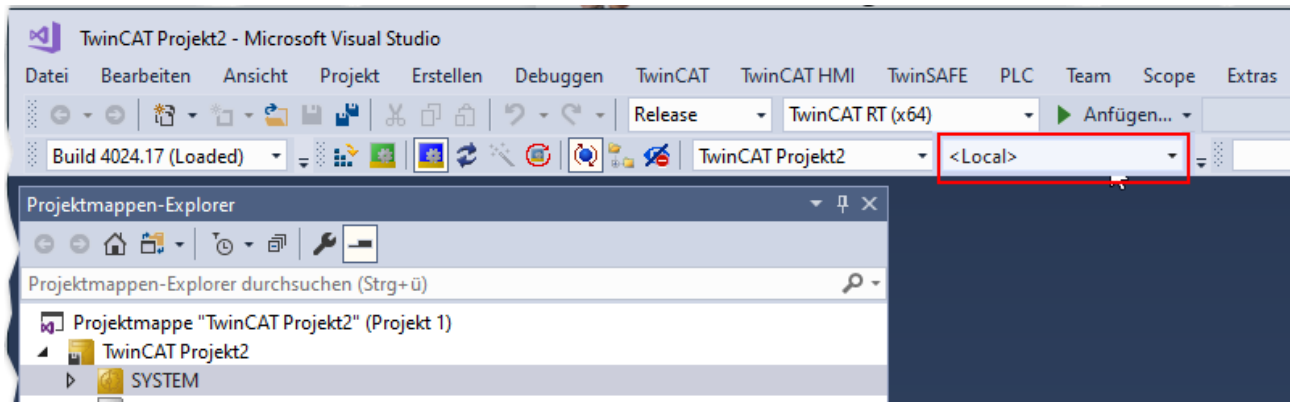
- Passende SPS-Projektvorlage [▶ 706] hinzufügen:

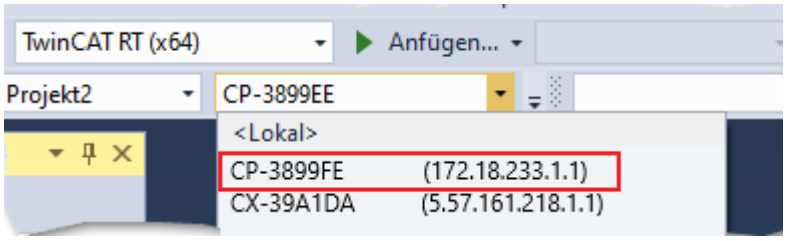




Zielsystem auswählen

- Um mit der Projekteinstellungen fortzufahren, müssen Sie zunächst eine Steuerung als Zielsystem auswählen.





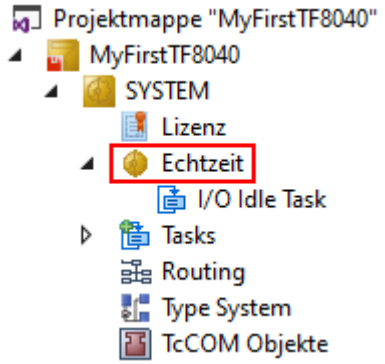
Weitere Informationen befinden sich im Kapitel Auswahl des Zielsystems.

Projekteinstellungen

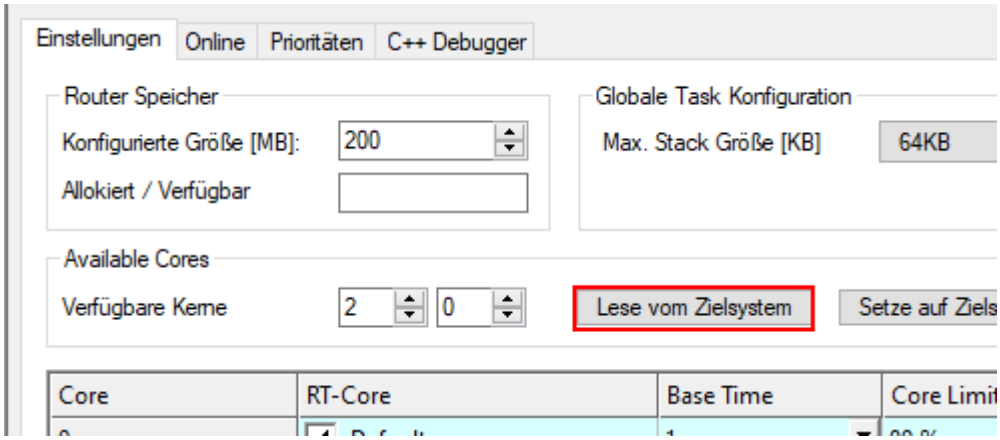
Nachdem das Zielsystem gewählt ist, kann mit den Projekteinstellungen fortgefahren werden.

System

Realtime

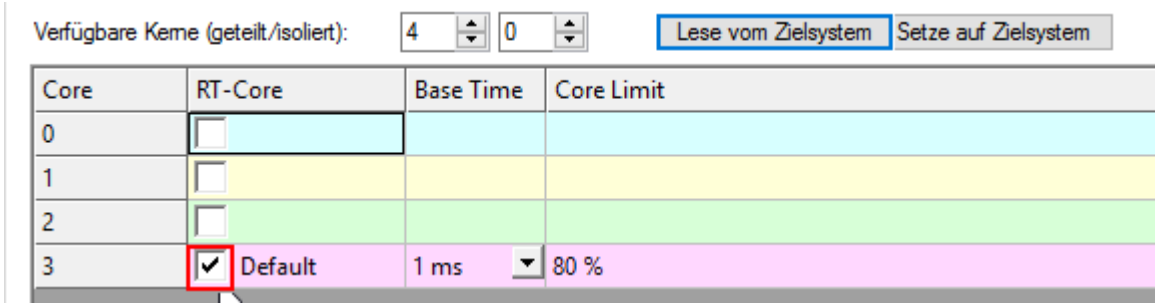


- Auslesen der vorhandenen Hardware-Konfiguration:



- Auswahl eines Kerns:

Falls mehrere Kerne zur Auswahl stehen, wird der letzte Kern empfohlen, da dort tendenziell wenig Last durch das Betriebssystem erzeugt wird.

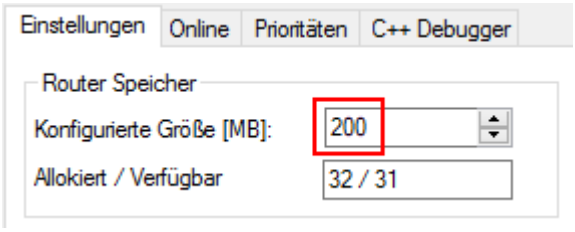


HINWEIS

Keine isolierten Kerne verwenden.

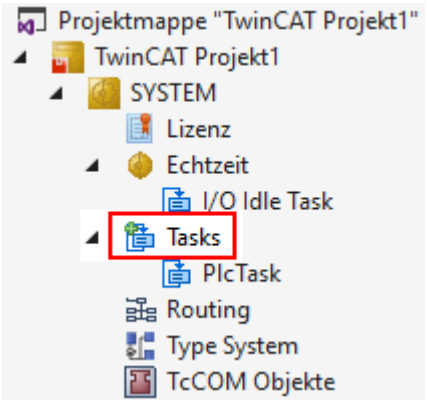
- Router-Speicher konfigurieren

Der Speicher sollte pauschal auf 200 MB eingestellt werden:



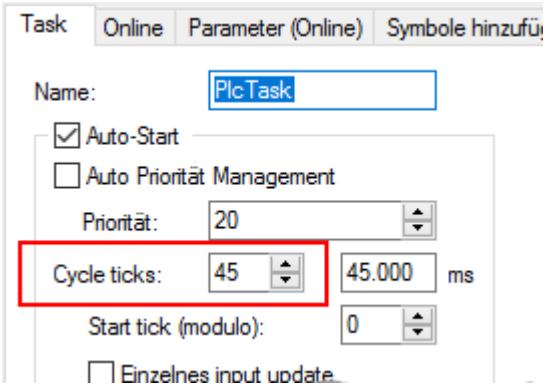
Ein Neustart des Betriebssystems (auf dem Zielgerät) ist zur Übernahme der Einstellung erforderlich.

Tasks



Einstellungen für die SPS-Task:

- Empfohlene Zykluszeit: 45 ms



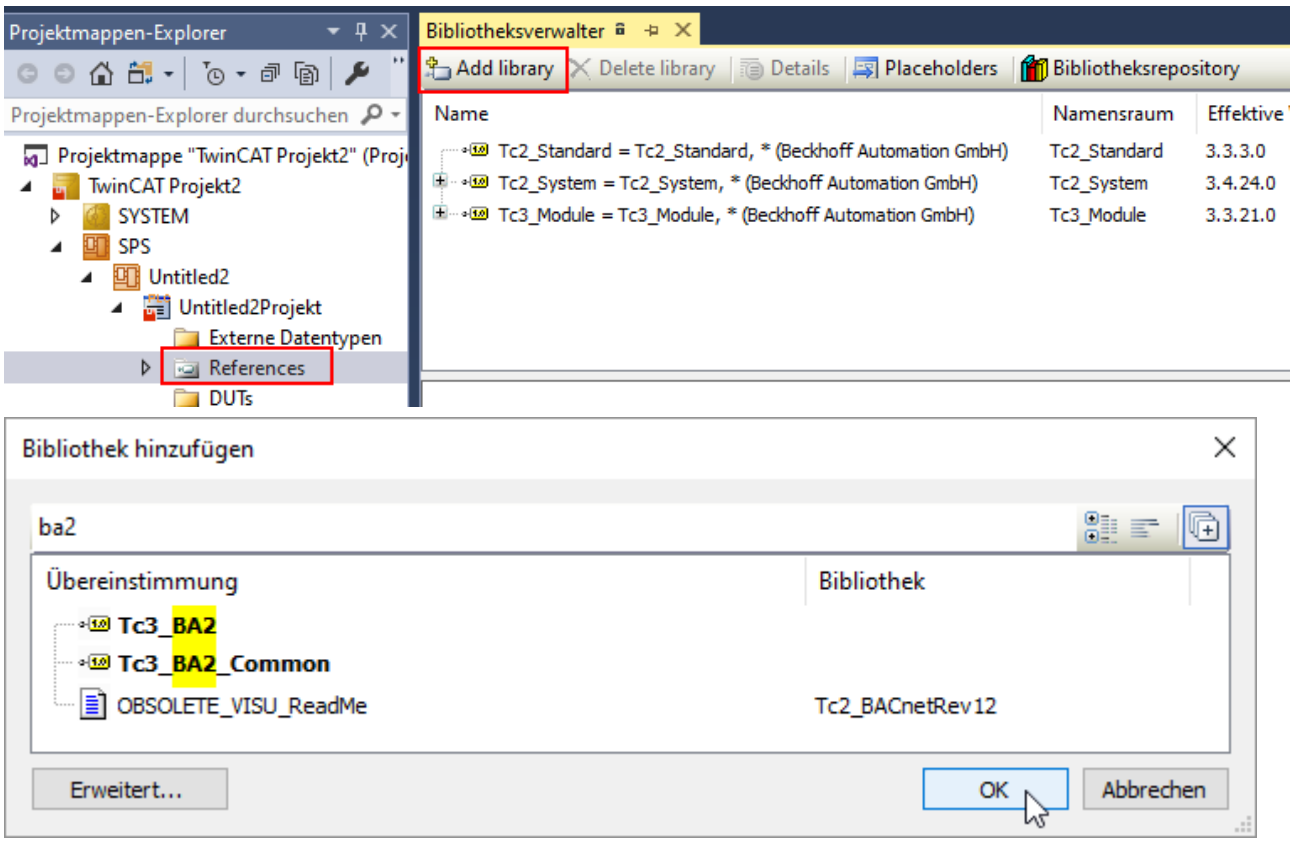
SPS

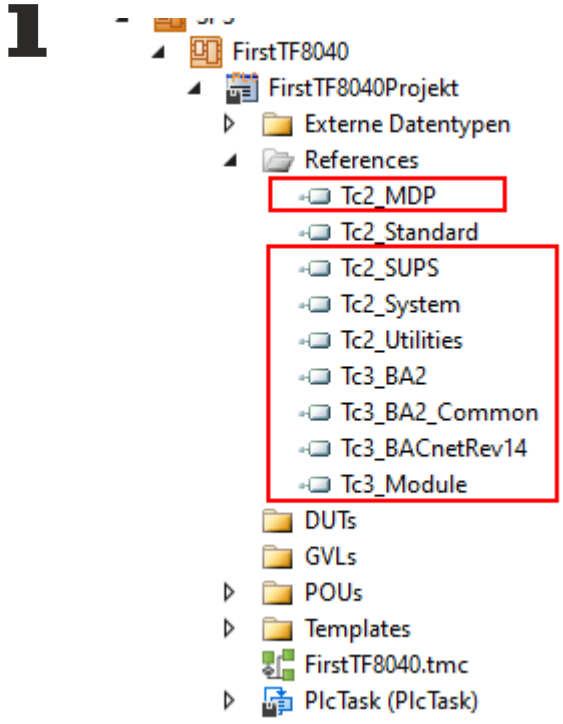
Die im Folgenden beschriebenen Einstellungen sind nicht notwendig, wenn eine SPS-Projekt-Vorlage [▶ 706] verwendet wird.

Bibliotheken

Wenn kein Template verwendet wird, müssen die folgenden Bibliotheken zum SPS-Projekt hinzugefügt werden.

- Tc3_BA2_Common
- Tc3_BACnetRev14
- Tc3_BA2





In der Standard SPS-BA-Vorlage [▶ 706] werden bereits alle notwendigen Bibliotheken automatisch geladen.

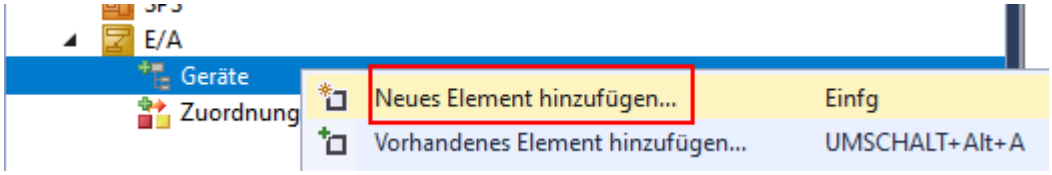
E/A

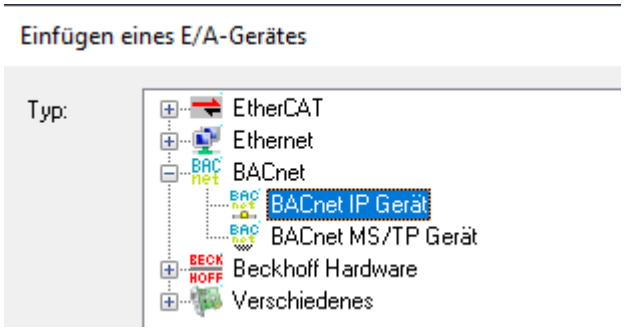
Das hier vorgeschlagene Vorgehen bezieht sich auf die notwendigen Arbeitsschritte, die erforderlich sind, um die Funktion TF8040 auf der gewünschten Hardware in Kombination mit BACnet einsatzbereit zu machen.

Auf weitere Schritte zum Einstellen der Hardware wird an dieser Stelle nicht näher eingegangen.

BACnet

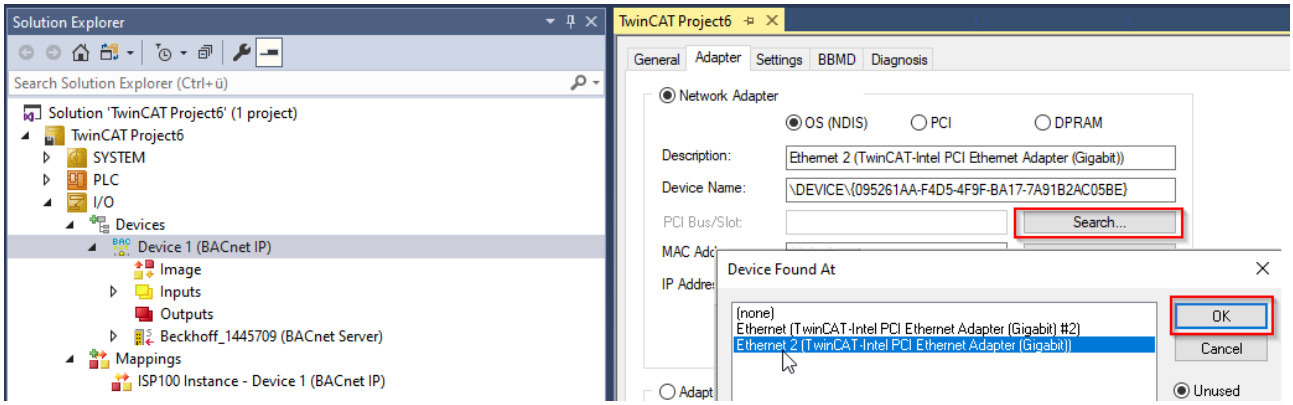
- BACnet-Gerät hinzufügen:





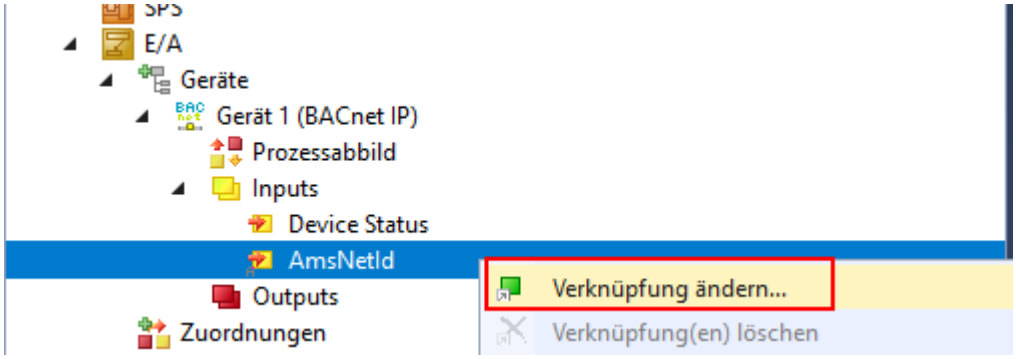
- Den passenden Netzwerk-Adapter auswählen:

Der Adapter wird im BACnet-Gerät unter dem Reiter Adapter eingestellt:

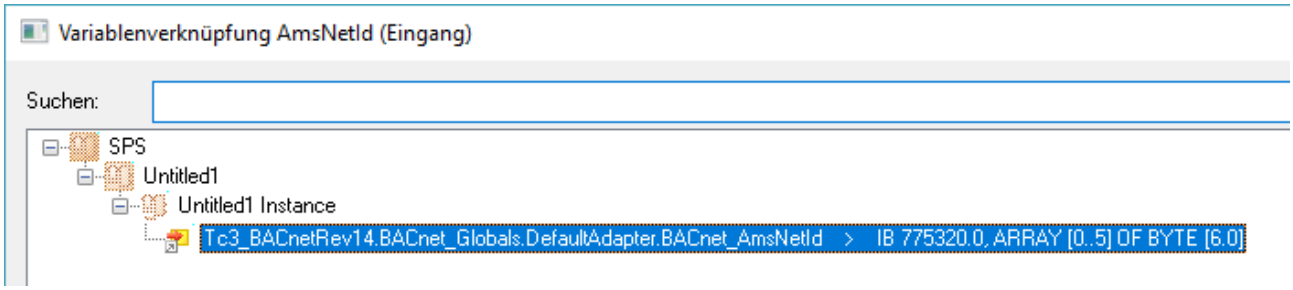


- Verknüpfen des BACnet Adapters mit der AMS-NetID:

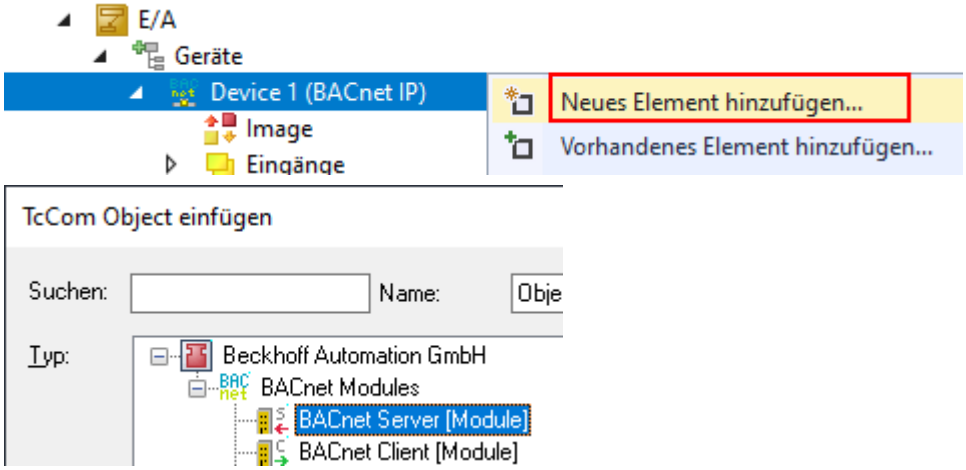
Dieser Arbeitsschritt ist erst möglich, wenn das Projekt zuvor einmal fehlerfrei kompiliert wurde.



Auswahl des zu verwendenden BACnet Adapters:



- Hinzufügen des BACnet Servers:



Lizenz

TwinCAT 3 Standardlizenzen sind an eine eindeutige System-ID eines TwinCAT-3-Lizenz-Dongles (oder IPCs) gebunden.

Standardlizenzen sind kostenpflichtig: Der Lizenzpreis hängt vom Hardware-Plattform-Level ab.

Nähere Informationen sind ausführlich im Informationssystem (siehe Lizenzierung) beschrieben.

- Ermitteln des Lizenzstatus.
- Zum Starten ist es möglich eine Testlizenz zu aktivieren. Diese schaltet sämtliche Funktionen für 7 Tage frei.

SPS-Template anpassen

Passen Sie die verwendete SPS-Projektvorlage [▶ 706] nach der empfohlenen Vorgehensweise [▶ 706] an.

Fortfahren

Das SPS-Projekt ist nun eingerichtet und mit der Projektierung kann begonnen werden.

4.2 HMI

Kapitel mit Anleitungen zur Erstellung eines HMI-Projekts.

4.2.1 Start eines Projekts

Beschreibung

Im Folgenden wird beschrieben, wie ein TcHmiBa-Projekt erstellt und gestartet wird.

Durch die Verwendung von TcHmiBa-Projektvorlagen [▶ 1297] können einzelne Schritte bereits entsprechend vorbereitet sein.

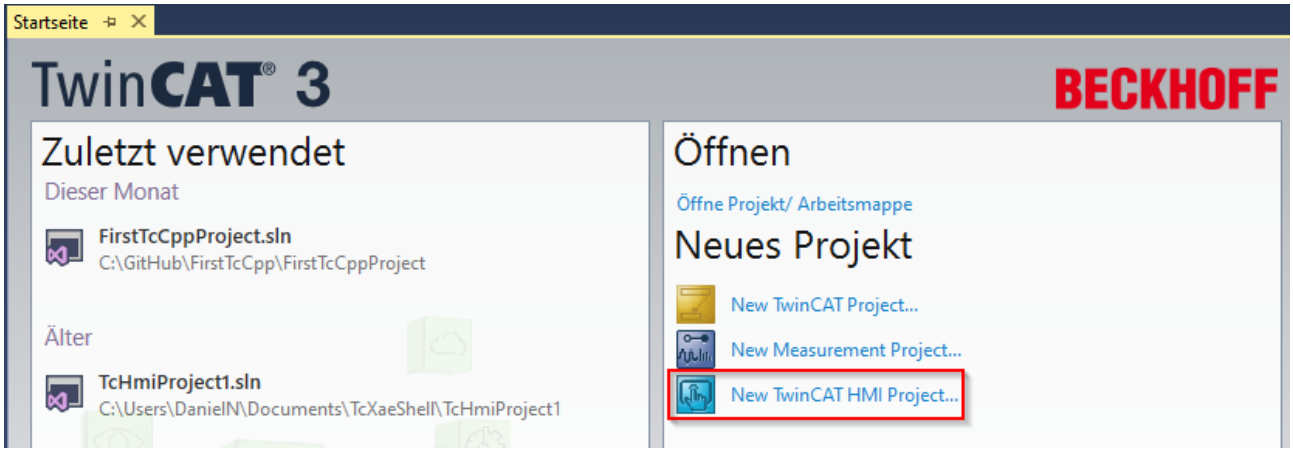
Vorgehen

Installation TwinCAT 3 HMI

Für das Erstellen von TcHmiBa-Projekten wird das TwinCAT 3 HMI Engineering benötigt. Beachten Sie dazu die Systemvoraussetzungen.

Weitere Informationen befinden sich in der Dokumentation zur TwinCAT 3 HMI (TE2000).

Nach der Installation sollte in allen beim Setup ausgewählten Entwicklungsumgebungen (TcXaeShell, Visual Studio 2019 etc.) die Projektvorlage für ein Standard-HMI verfügbar sein.



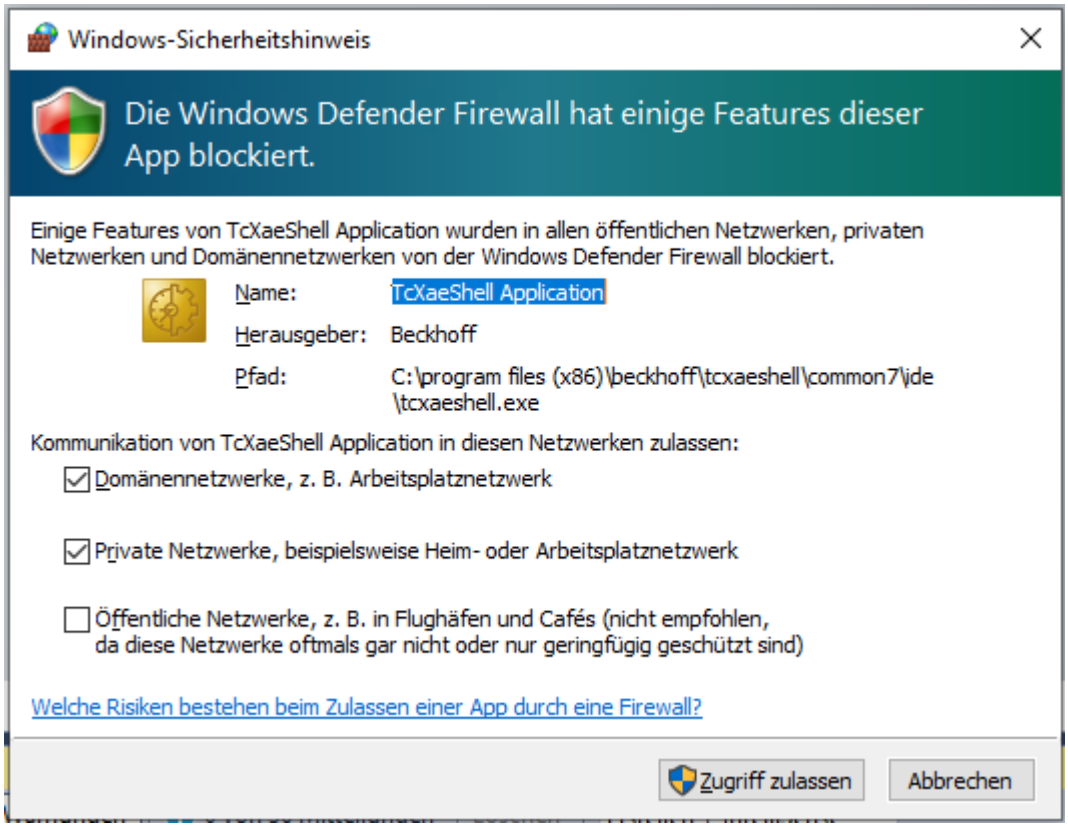
Installation TF8040

Zur Nutzung von TcHmiBa-Projekten muss die TcHmi und TF8040 installiert sein. Das Setup von TF8040 enthält unter anderem die NuGet-Pakete, auf die später näher eingegangen wird.

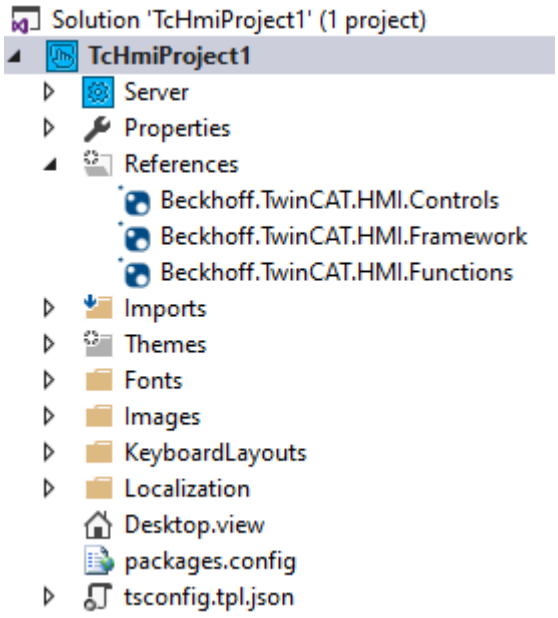
Neues TcHmi-Project erstellen

Auf der Startseite oder über Datei > Neu > Projekt kann unter der Kategorie **TwinCAT HMI** ein neues **TwinCAT HMI Projekt** erstellt werden.

Während der Erstellung des Projektes kann es einen Windows-Sicherheitshinweis geben, da der Engineering-Server auf das Netzwerk zugreift. Die Zugriffsanfrage bitte entsprechend zulassen.



Das erstellte Projekt sollte nun folgenden Aufbau haben:

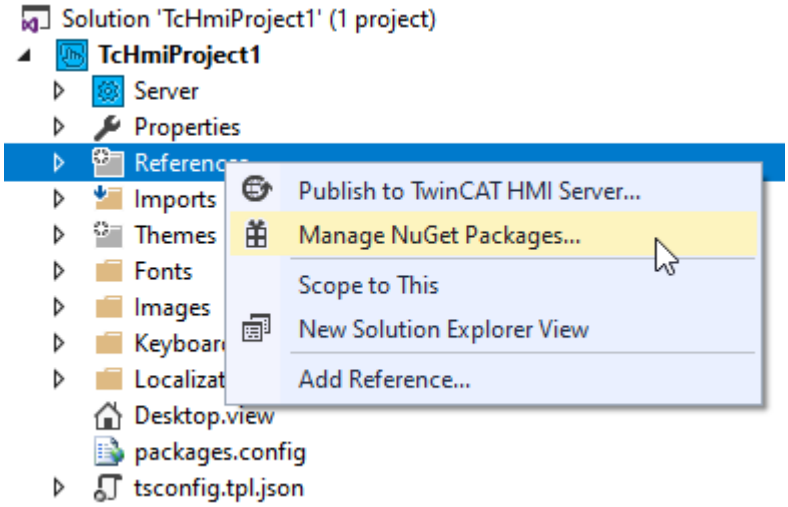


Installation NuGet-Pakete

Spezifische Controls für die Gebäudeautomation sind nicht im Standardumfang der TcHmi-Toolbox enthalten.

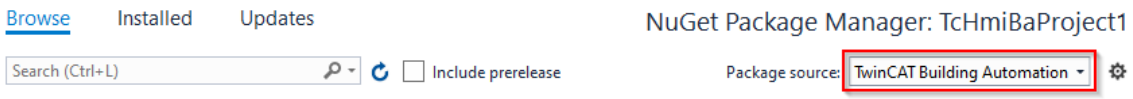
Um diese im erstellten Projekt nutzen zu können, muss das NuGet-Paket **Beckhoff.TwinCAT.HMI.Controls** installiert werden. Das Vorgehen dazu ist wie folgt:

- NuGet-Paketverwaltung öffnen

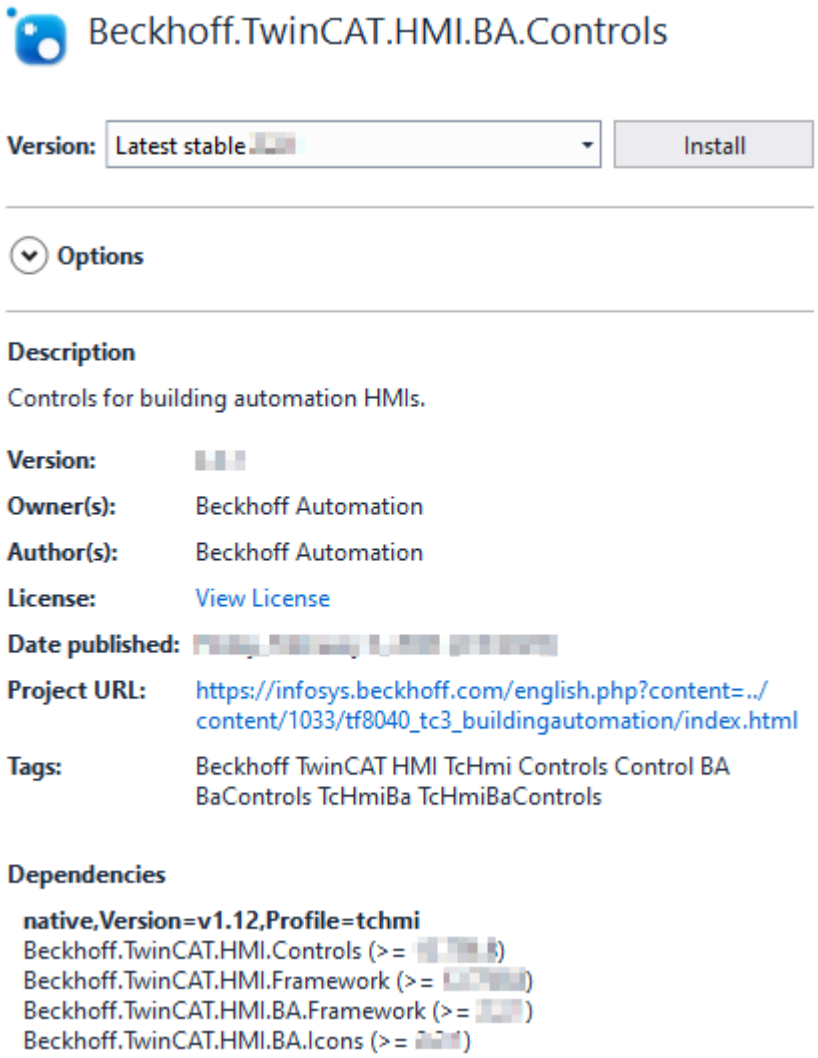


Oben rechts in der Paketverwaltung muss die richtige Paketquelle **TwinCAT Building Automation** ausgewählt werden.

Die Auswahl ist nur verfügbar, wenn TF8040 inkl. der TcHmi-Option (standard) installiert wurde.

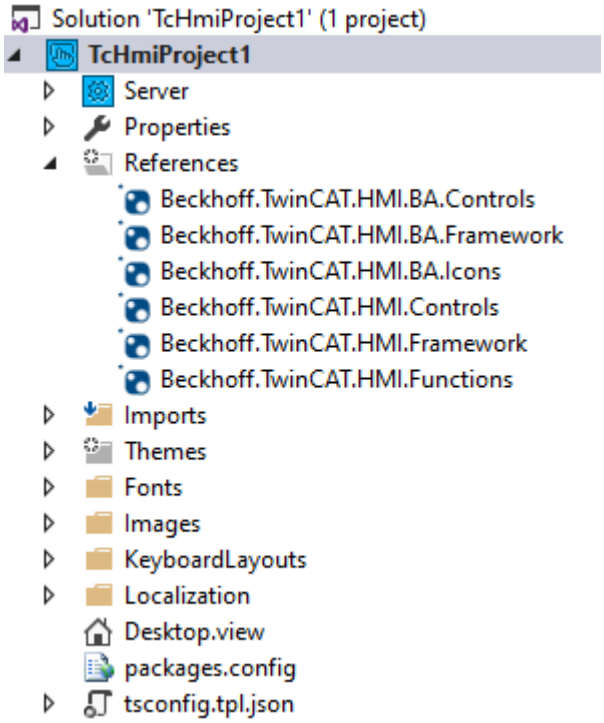


- **Beckhoff.TwinCAT.HMI.BA.Controls** installieren

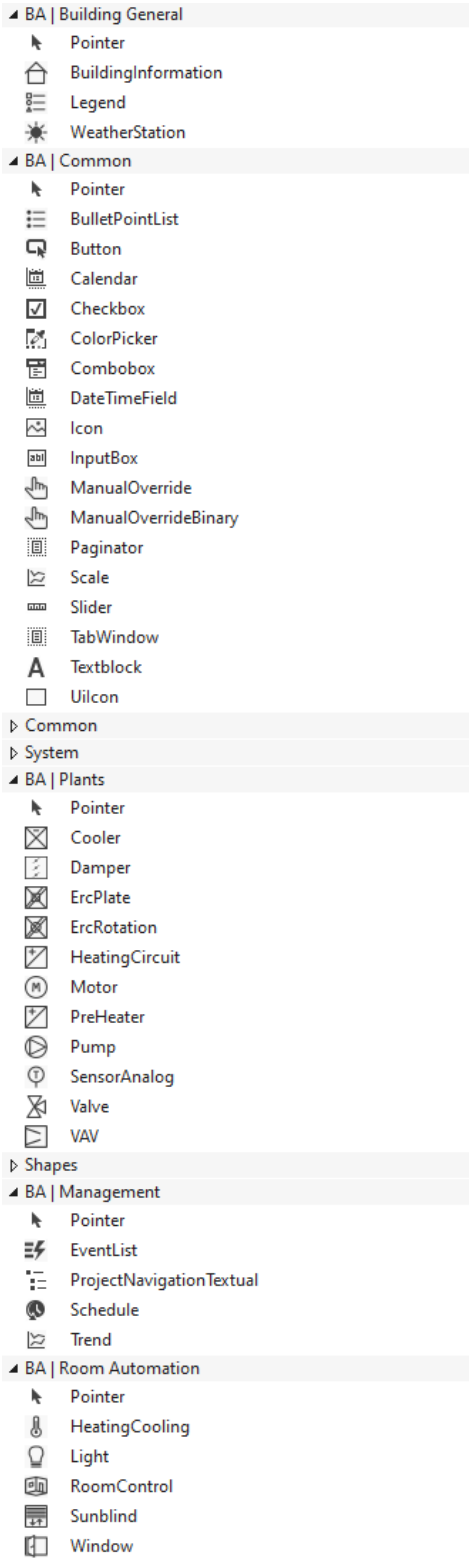


Im Zuge der Installation werden ebenfalls die Pakete Beckhoff.TwinCAT.HMI.BA.Framework [▶ 1263] und Beckhoff.TwinCAT.HMI.BA.Icons [▶ 1252] installiert.

Nach der Installation der NuGet-Pakete sollte der Projektbaum folgendermaßen aussehen.



Wenn der *Desktop.view* und die Toolbox geöffnet sind, sollten nun folgende Controls verfügbar sein:



Weitere Informationen befinden sich in den jeweiligen Dokumentationen [▶ 1141] der einzelnen Controls.

Verwendung

Mit den verfügbaren Controls können sowohl Visualisierungen als auch UserControls erstellt werden. Die Verwendung entspricht damit den Standard-Controls der TcHmi.

Generische Funktionen

Wenn Sie die TF8040 als **komplette Lösung** mit allen Vorteilen während des Engineerings verwenden möchten, fahren Sie bitte mit dem Kapitel Generische Funktionen [▶ 79] fort.

4.2.2 Generische HMI

Beschreibung

Im Folgenden wird beschrieben, wie die **generischen Funktionalitäten** von TcHmiBa verwendet werden, um die Entwicklungsarbeit für das HMI zu minimieren. Als Voraussetzung gilt das Kapitel Start eines Projekts [▶ 72].

Vorgehen

Beachten Sie dazu die Systemvoraussetzungen [▶ 1140] der *BaSite-Extension*.

Weiterhin ist eine SPS erforderlich, die mit den Bibliotheken aus TF8040 erstellt wurde und bereits aktiv ist.

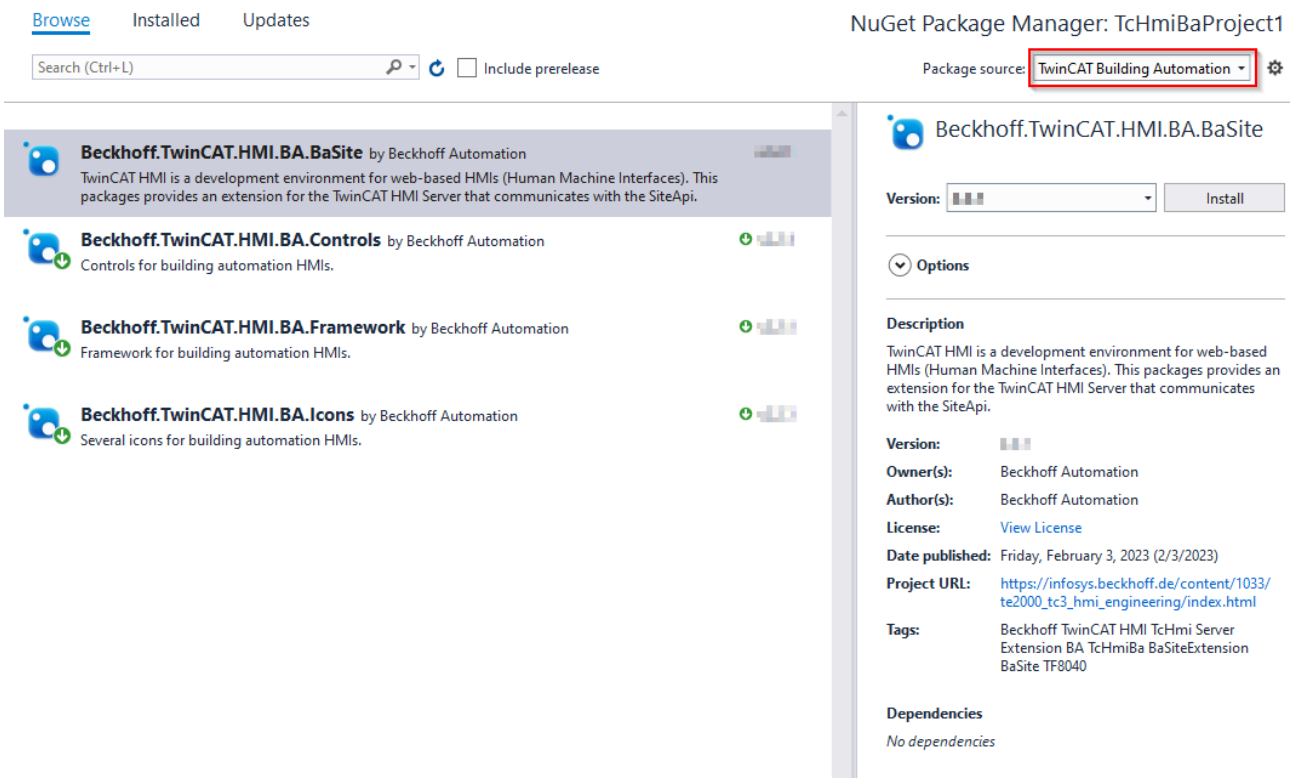
Beschreibung von notwendigen Begriffen [▶ 33].

Server vorbereiten

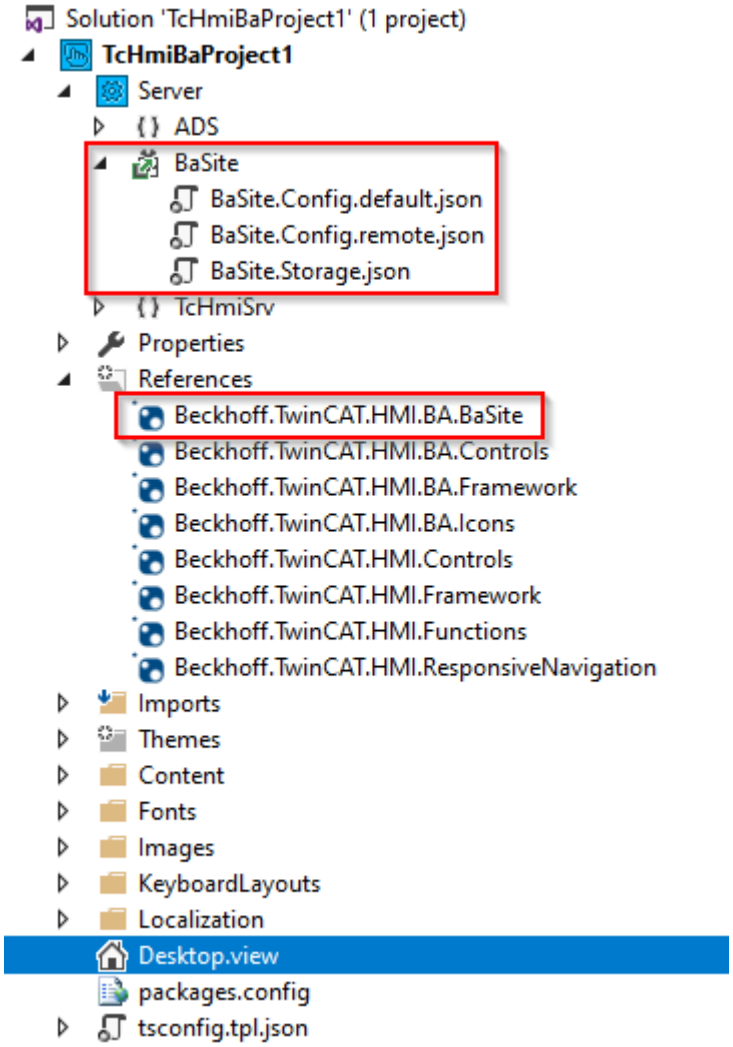
Bevor mit dem Erstellen der Visualisierung mit generischen Funktionen gestartet werden kann, muss der Server vorbereitet werden.

Installation BaSite-Extension

Für die Unterstützung der generischen Funktionalitäten ist es notwendig das NuGet-Paket Beckhoff.TwinCAT.HMI.BA.BaSite [▶ 1283] zu installieren.

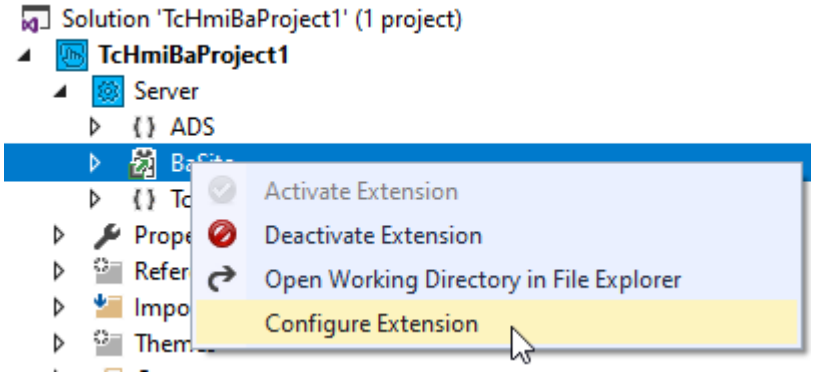


Nach der Installation sollte der Projektbaum wie folgt aussehen:

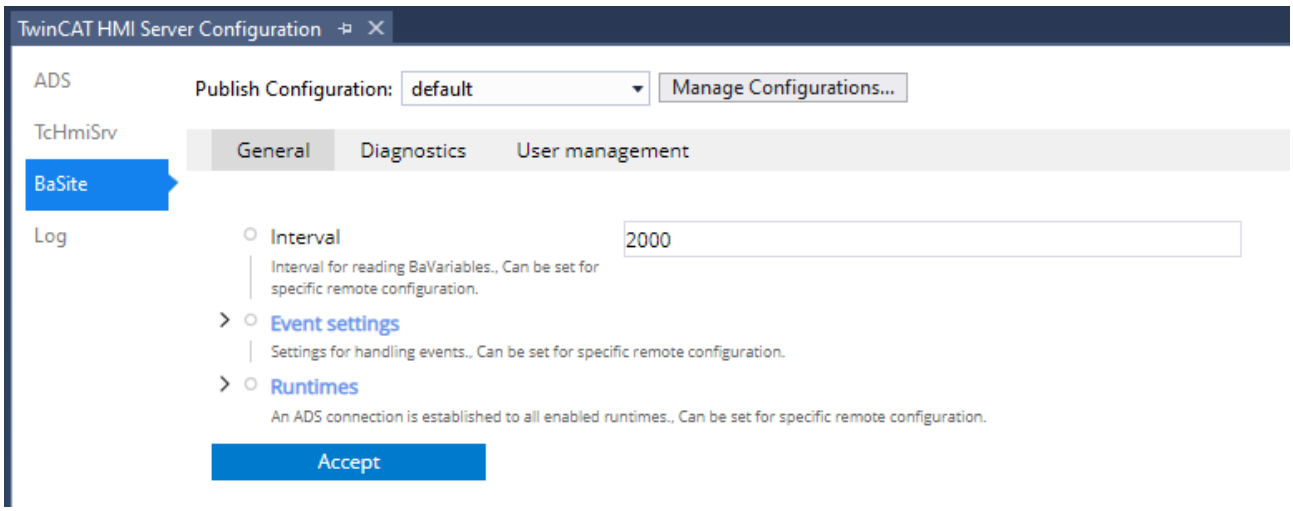


Konfiguration BaSite-Extension

Konfigurationsseite der **BaSite-Extension** öffnen.



In dem geöffneten Fenster können alle Einstellungen für die **BaSite-Extension** vorgenommen werden.



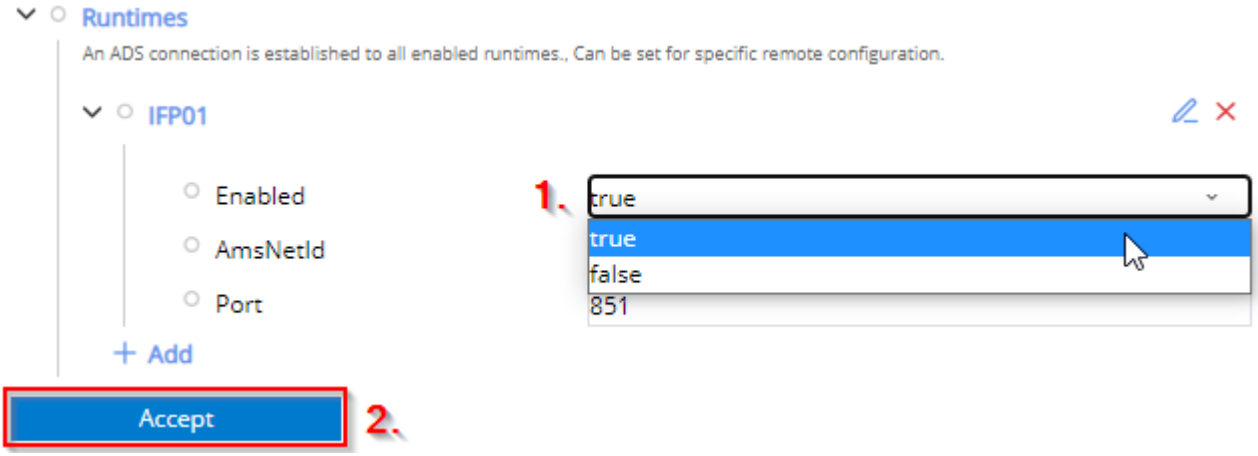
Weitere Informationen über die Konfiguration und Funktionen befinden sich in der Dokumentation [▶ 1283] der Servererweiterung.

Im Eintrag **Laufzeiten** ist standardmäßig der IFP01 angelegt.

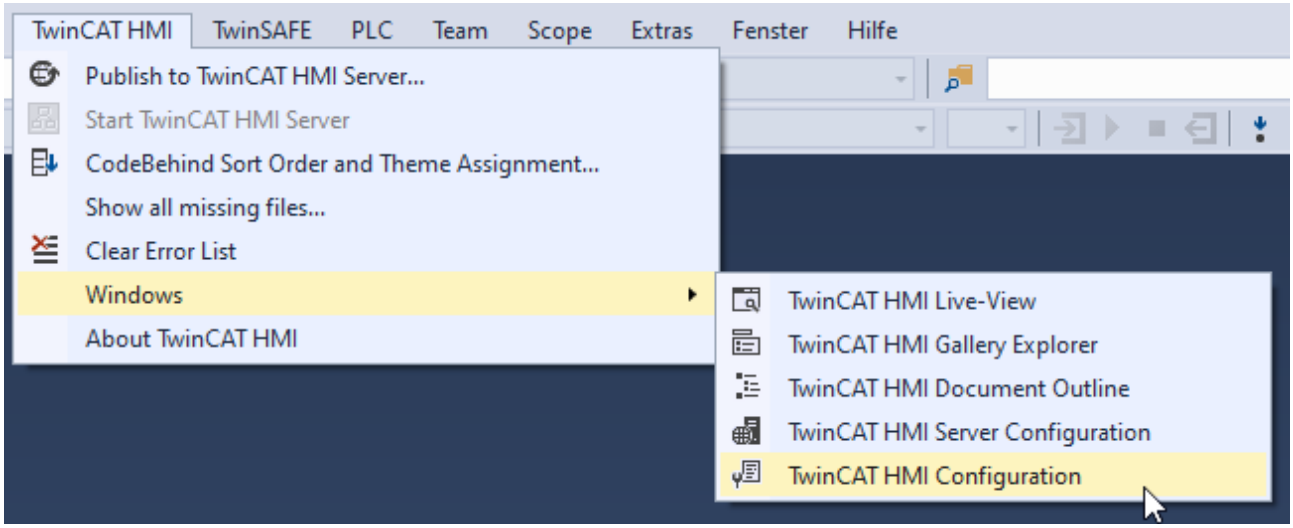
Unter AmsNetId und Port sind die jeweiligen Angaben für die aktive SPS einzutragen.
Tragen Sie unter **AmsNetId** und **Port** die jeweiligen Angaben für die aktive SPS ein.

Wenn die SPS lokal auf dem Entwicklungsrechner läuft, ist keine Änderung der Einstellungen erforderlich.

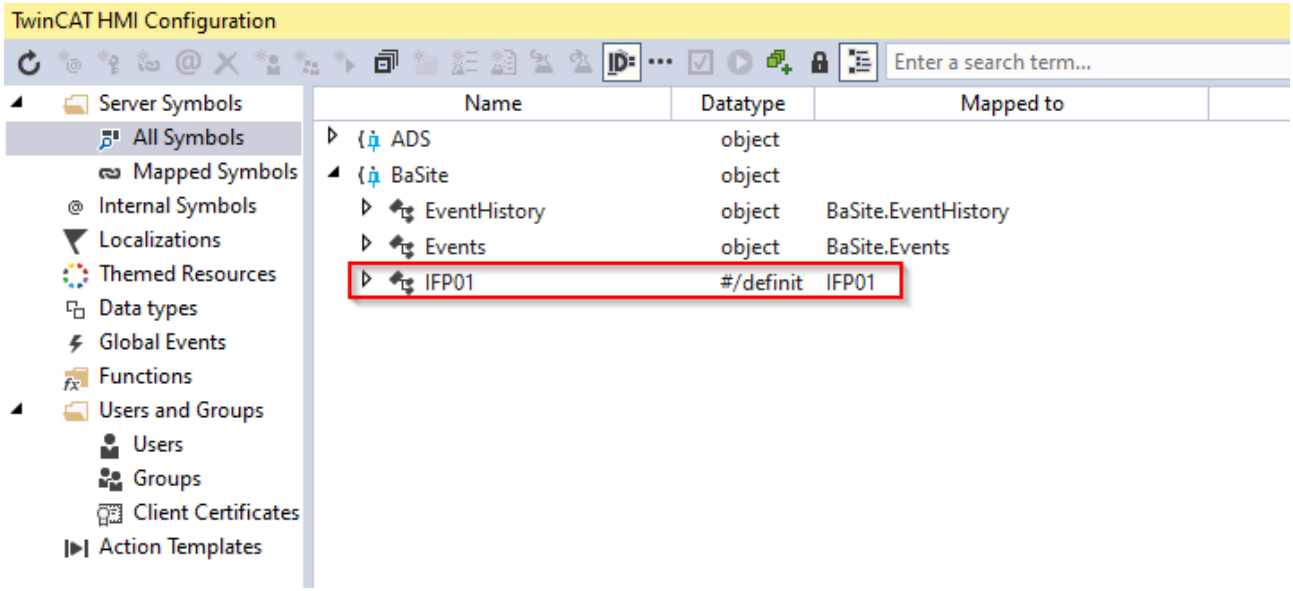
Anschließend muss noch die Laufzeit aktiviert und der Dialog bestätigt werden.



Nun sollte in der **TwinCAT HMI Konfiguration** die SPS erfolgreich in der Servererweiterung angelegt worden sein.

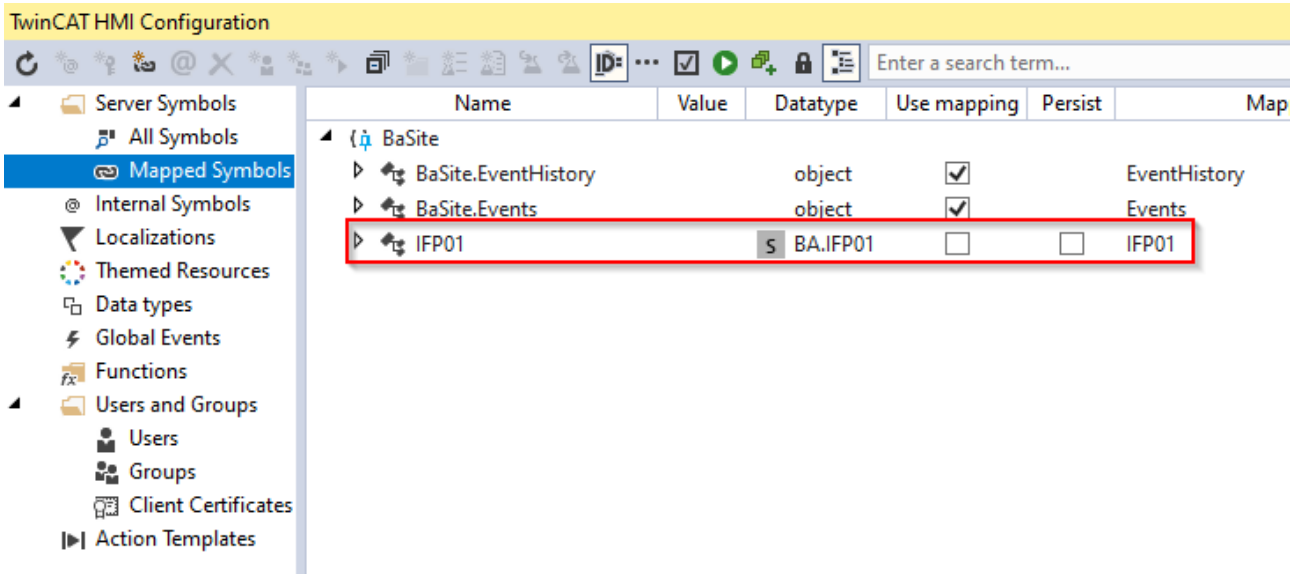


In dem geöffneten Fenster ist dann unter **All Symbols** die Laufzeit aufgelistet.



Ist im Projektbaum kein HMI Projekt ausgewählt, so bleibt die Anzeige in der **TwinCAT HMI Konfiguration** leer.

Ebenfalls wurde bereits ein Mapping für die Laufzeit angelegt. Das Mapping wird unter **Mapped Symbols** aufgelistet.



Es sind die Voraussetzungen zu notwendigen Mappings [▶ 1285] für die generischen Funktionalitäten zu beachten.

Die Konfiguration des Servers ist damit abgeschlossen.

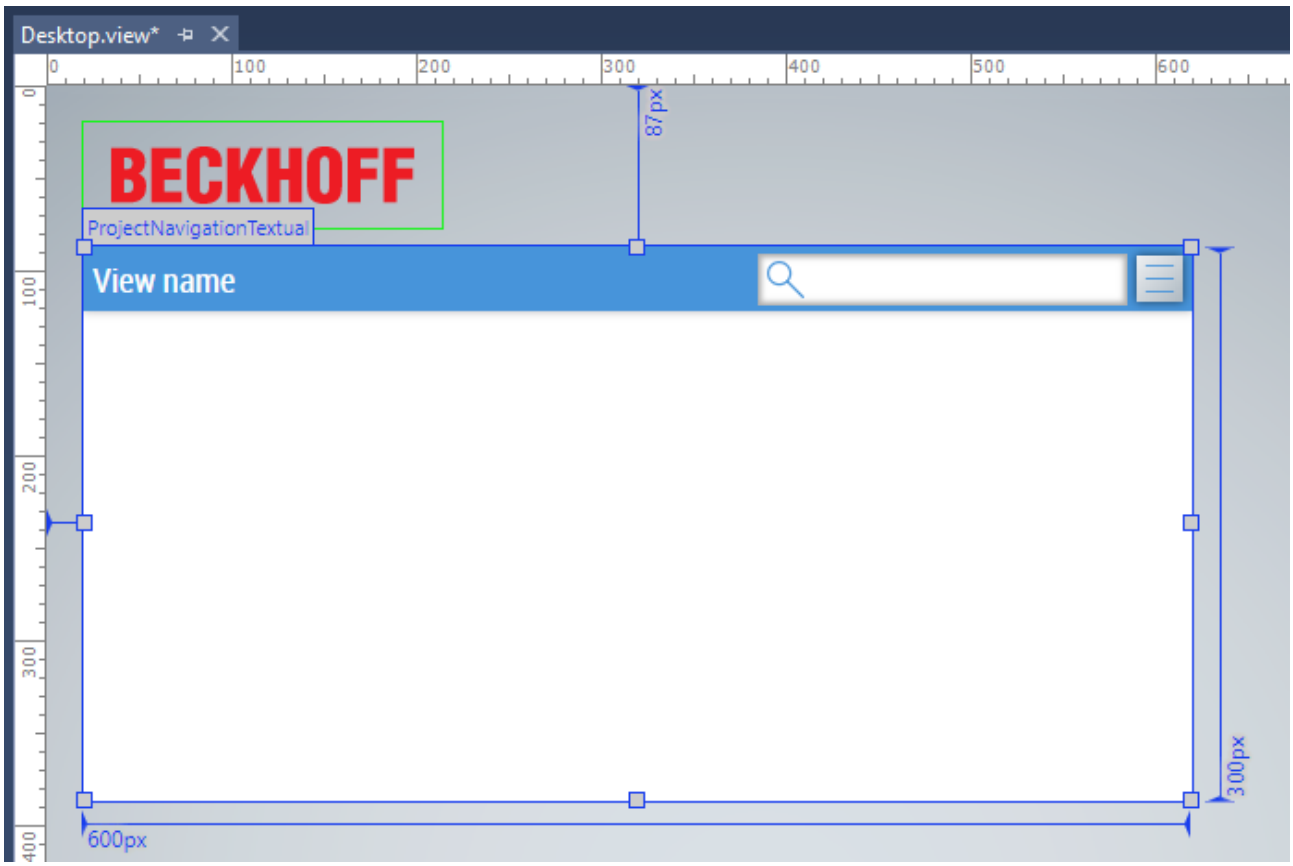
Generische Controls verwenden

Die generischen Controls können nur mit der **BaSite-Extension** verwendet werden. Eine kleine Auswahl wird nachfolgend kurz beschrieben.

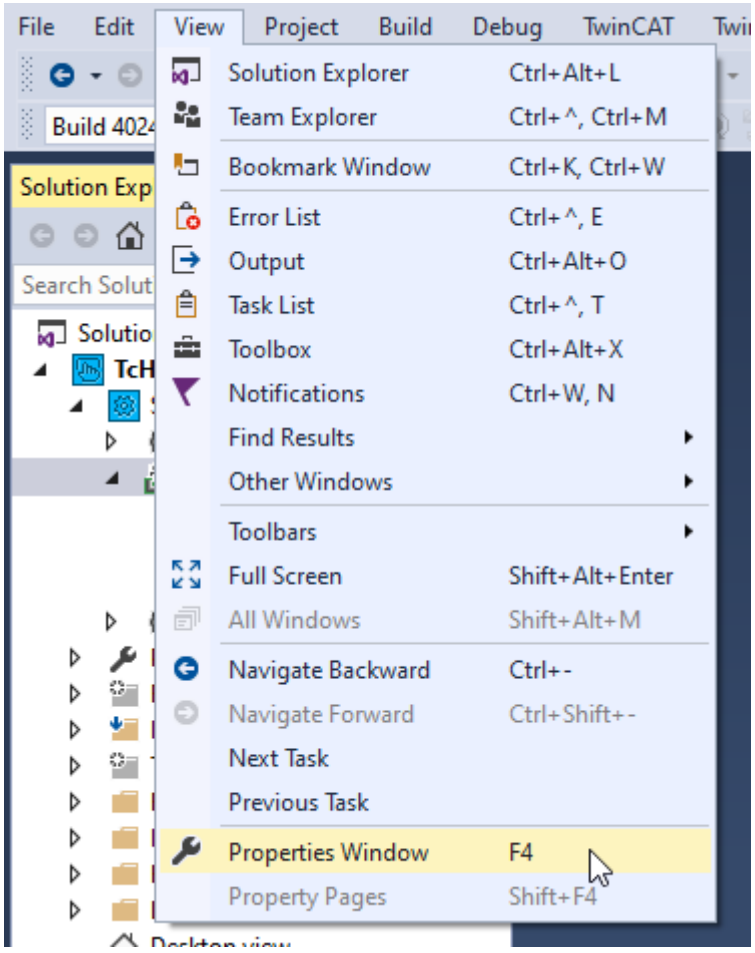
Weitere Informationen befinden sich in den jeweiligen Dokumentationen der einzelnen Controls.

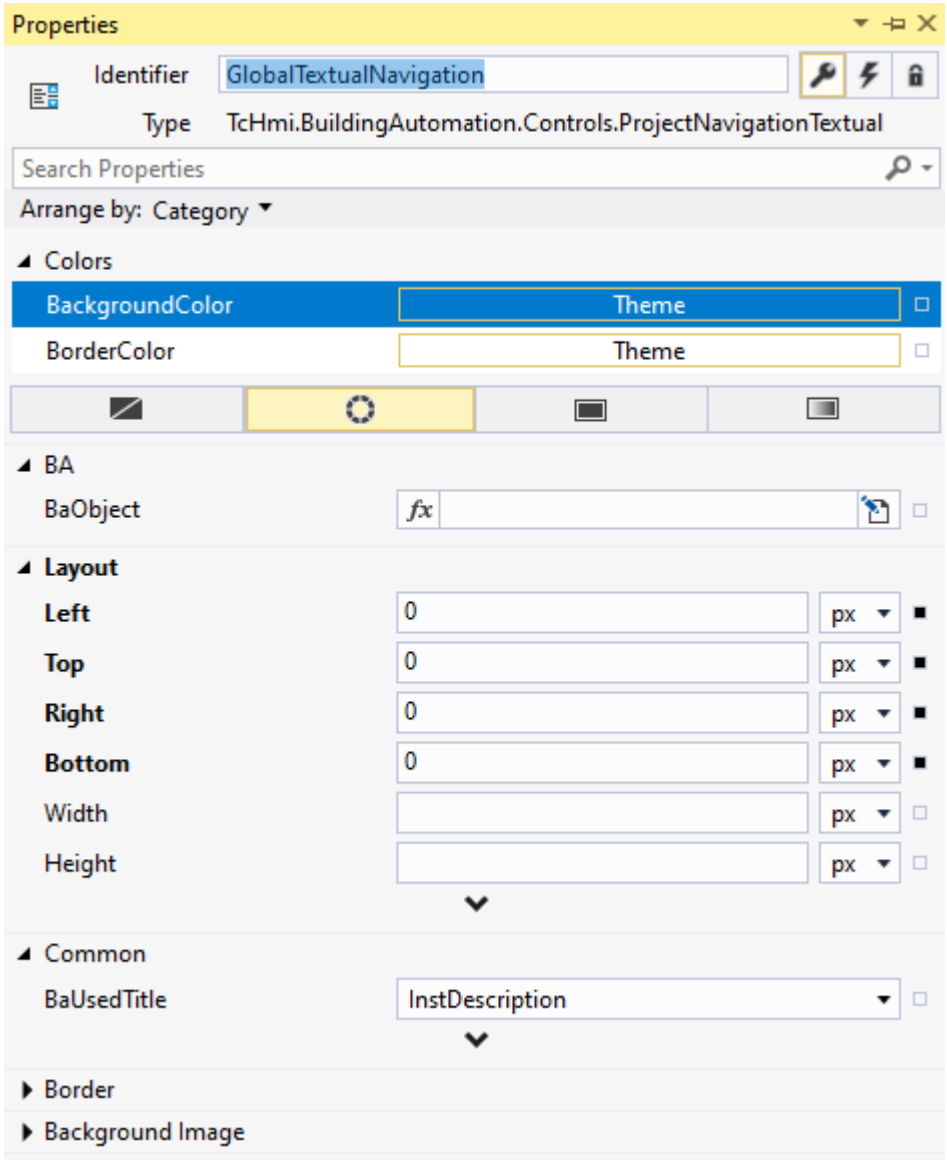
Projektnavigation

Der schnellste Einstieg in das HMI ist die Verwendung des Controls ProjectNavigationTextual. Es befindet sich in der Kategorie **BA | General** der Toolbox. Wenn das Control auf den **Desktop.view** abgelegt wurde, sollte dieser so aussehen:



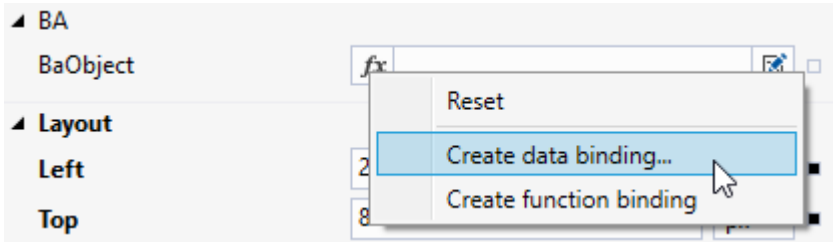
Öffnen des Eigenschaftenfensters.

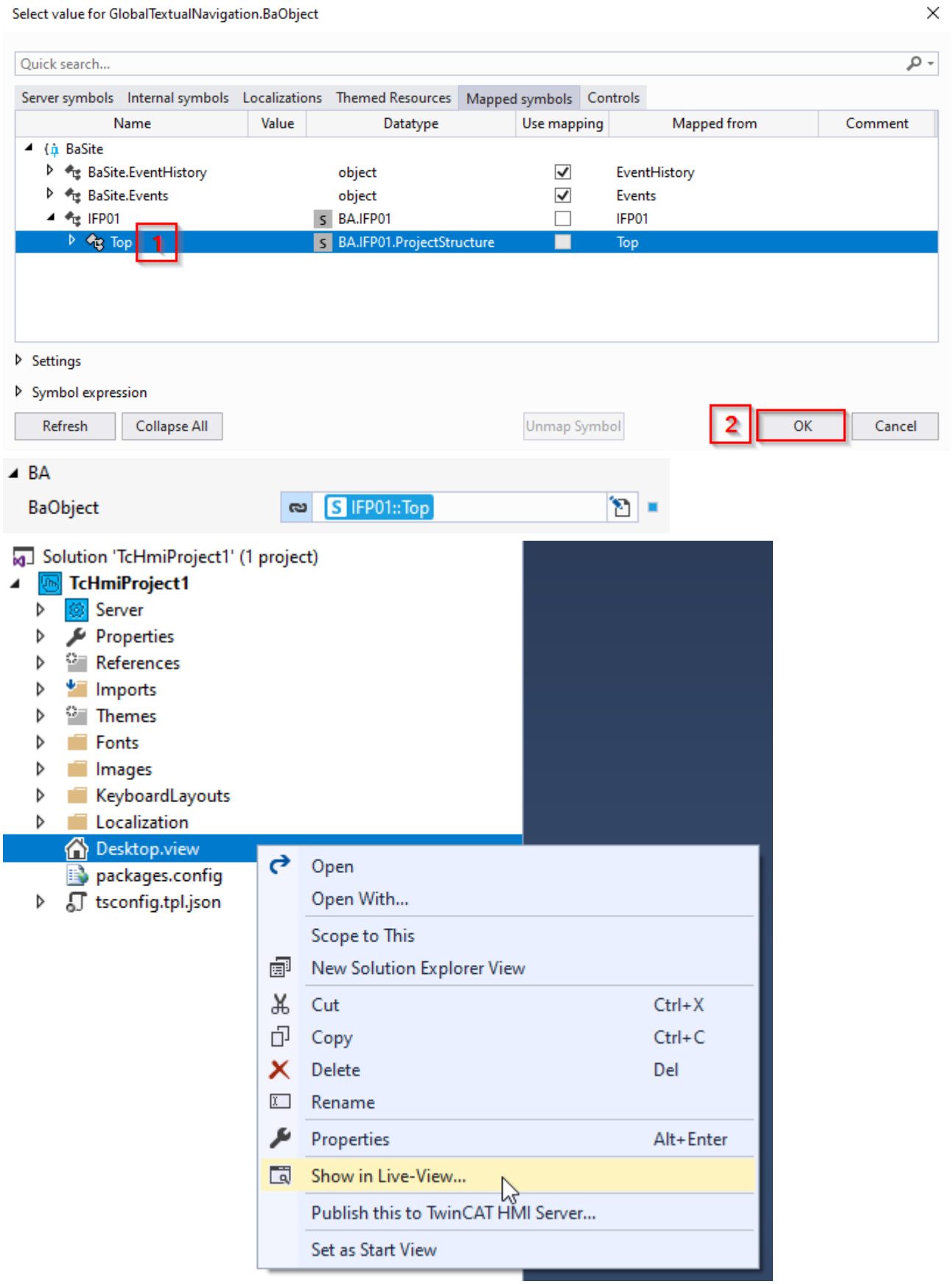




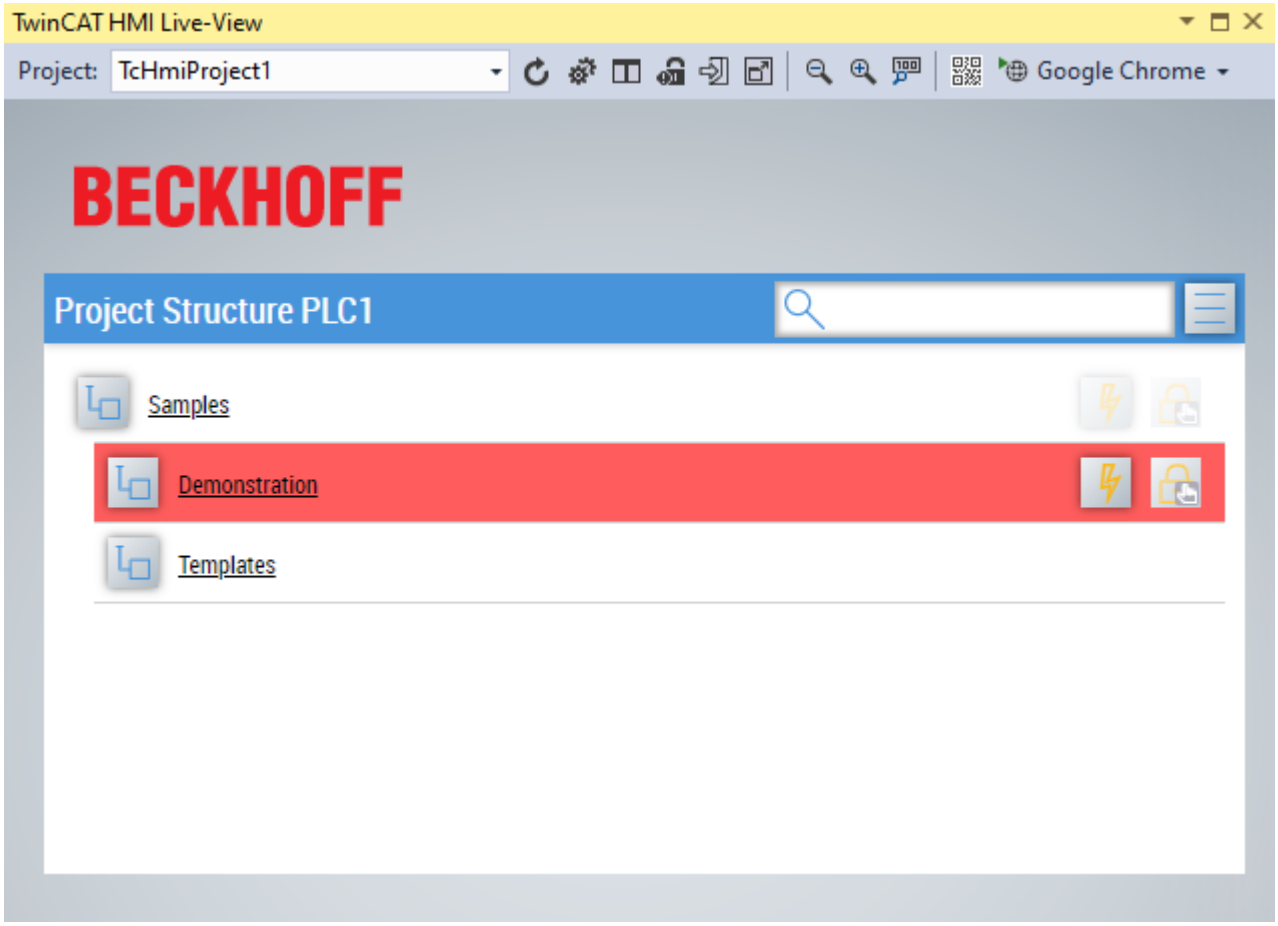
Der Inhalt des Eigenschaftenfensters richtet sich immer nach der aktuellen Auswahl. Um die Eigenschaften der Projektnavigation zu sehen, muss das Control auf dem *Desktop.view* ausgewählt sein.

Wenn das Attribut **BaObject** mit der Projektstruktur (**Top**-Knoten) der Laufzeit verknüpft ist, kann mit diesem Control durch die gesamte Projektstruktur navigiert werden.



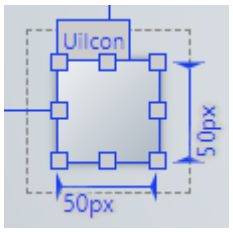


Der Live-View sollte nun wie folgt aussehen:



UiIcon

Das UiIcon [▶ 1178] kann für verschiedenste Anwendungen genutzt werden. Es befindet sich ebenfalls in der Kategorie **BA | General** der Toolbox.

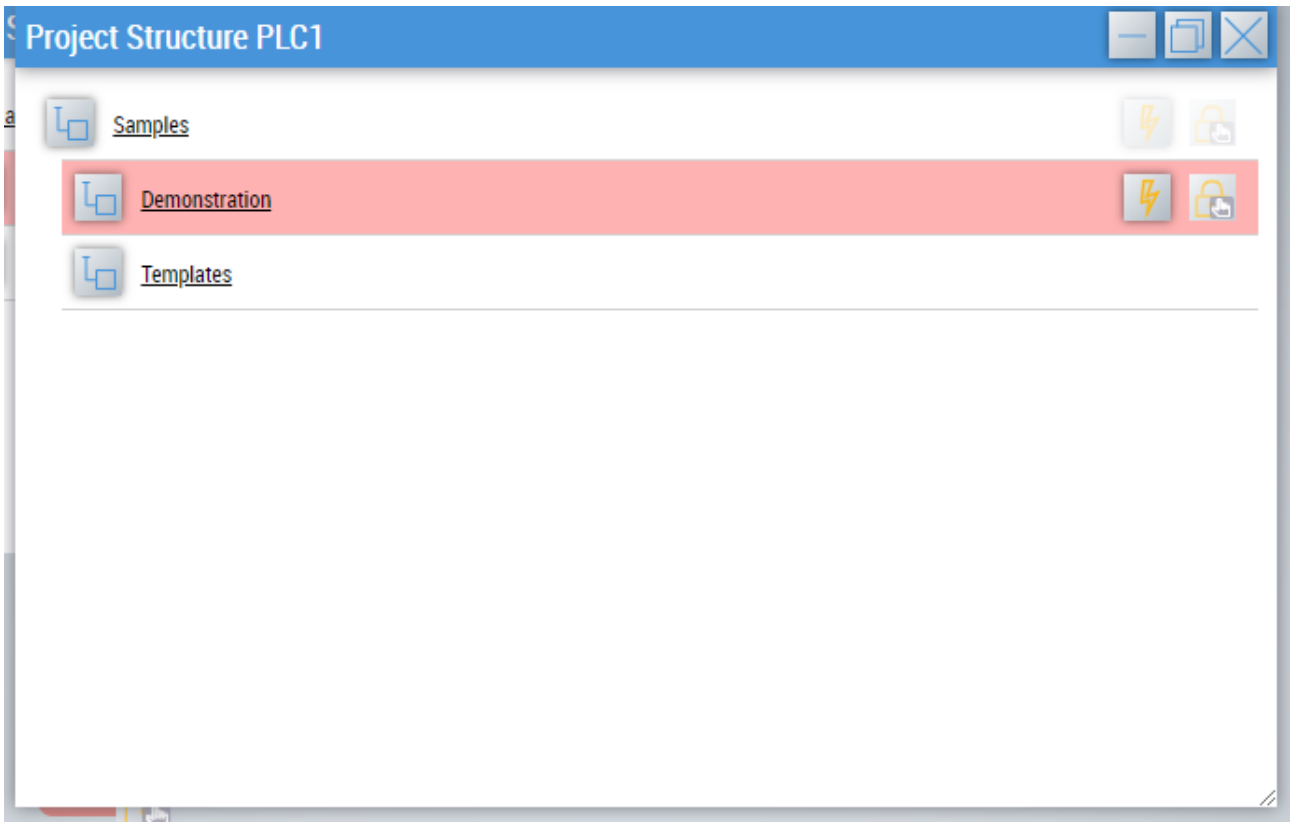


Wie alle Controls aus TcHmiBa, hat das *UiIcon* das Attribut *BaObject*. Auf dieses kann ein beliebiger *BaView* oder ein beliebiges *BaObject* verknüpft werden. Als Beispiel dient erneut die *ProjectStructure* der Laufzeit (s.o.).

Im Live-View sieht das **UiIcon** dann wie folgt aus:



Es zeigt die aktiven Events des verknüpften Views / Objektes an und ermöglicht das Öffnen eines Fensters, welches die generische Navigation von oben enthält.



Aus diesem Fenster stehen damit dieselben Funktionalitäten zur Verfügung.

5 Beispiele

Beispiele zu allen Features aus TF8040.

Downloads

- TF8040 https://infosys.beckhoff.com/content/1031/TF8040_TC3_BuildingAutomation/Resources/18733784075.zip
  Enthält
  - Beispiele zu PLC (Templates [▶ 98])
  - Beispiele zu HMI (Konzept [▶ 90], Templates [▶ 96])

Konfiguration

SPS

Einzelne Einstellungen innerhalb der *TF8040-Concept-Samples-PLC-Solution* müssen an die eingesetzte Hardware angepasst werden.

Die Einstellungen betreffen das Projekt und die Hardware.

Alle erforderlichen Schritte sind unter Start eines Projekts [▶ 66] beschrieben.

Vereinzelte, bereits vorgenommene Einstellungen müssen nicht weiter berücksichtigt werden.

HMI

SPS vorbereiten

Um die HMI auszuführen, muss die SPS muss konfiguriert und gestartet sein.

Installation TwinCAT HMI

Zum Laden des Beispielprojekts, muss vorher das TwinCAT 3 HMI Engineering heruntergeladen und installiert werden.

Weitere Informationen zu den erforderlichen Schritten finden Sie in der Dokumentation zum TwinCAT 3 HMI Engineering im Abschnitt Installation.

Download

Wenn die SPS aktiviert ist und läuft, kann die Sample-Solution geöffnet werden.

Entwicklungssystem lizenzieren

Damit die HMI-Beispiele lauffähig sind, muss die TwinCAT 3 HMI Server Lizenz der TF2000 auf dem Zielsystem lizenziert werden.

Serverkonfiguration

Wenn die Laufzeit auf dem PC aktiviert ist, auf dem auch das HMI Projekt gestartet wird, sind keine weiteren Konfigurationen notwendig.

Falls nicht, muss die Konfiguration angepasst werden. Die Vorgehensweise hierfür ist im Tutorial Generische HMI [▶ 79] beschrieben.

Bitte nehmen Sie die allgemeinen Anmerkungen zur Kenntnis.

Weiterführende Informationen

- SPS-Programmierung [▶ 100]

5.1 Konzeptbeispiele

Die Beispiele in diesem Paket zeigen das Konzept von TF8040 seitens der SPS und HMI.

Das Kapitel stellt die Ansätze zur Erstellung eines TF8040 Projektes dar. Dabei werden auch verschiedene BaObjekt-Typen und das Vorgehen für die Implementierung einer HMI gezeigt.

Die Beispiele referenzieren auf die Downloads [▶ 89].

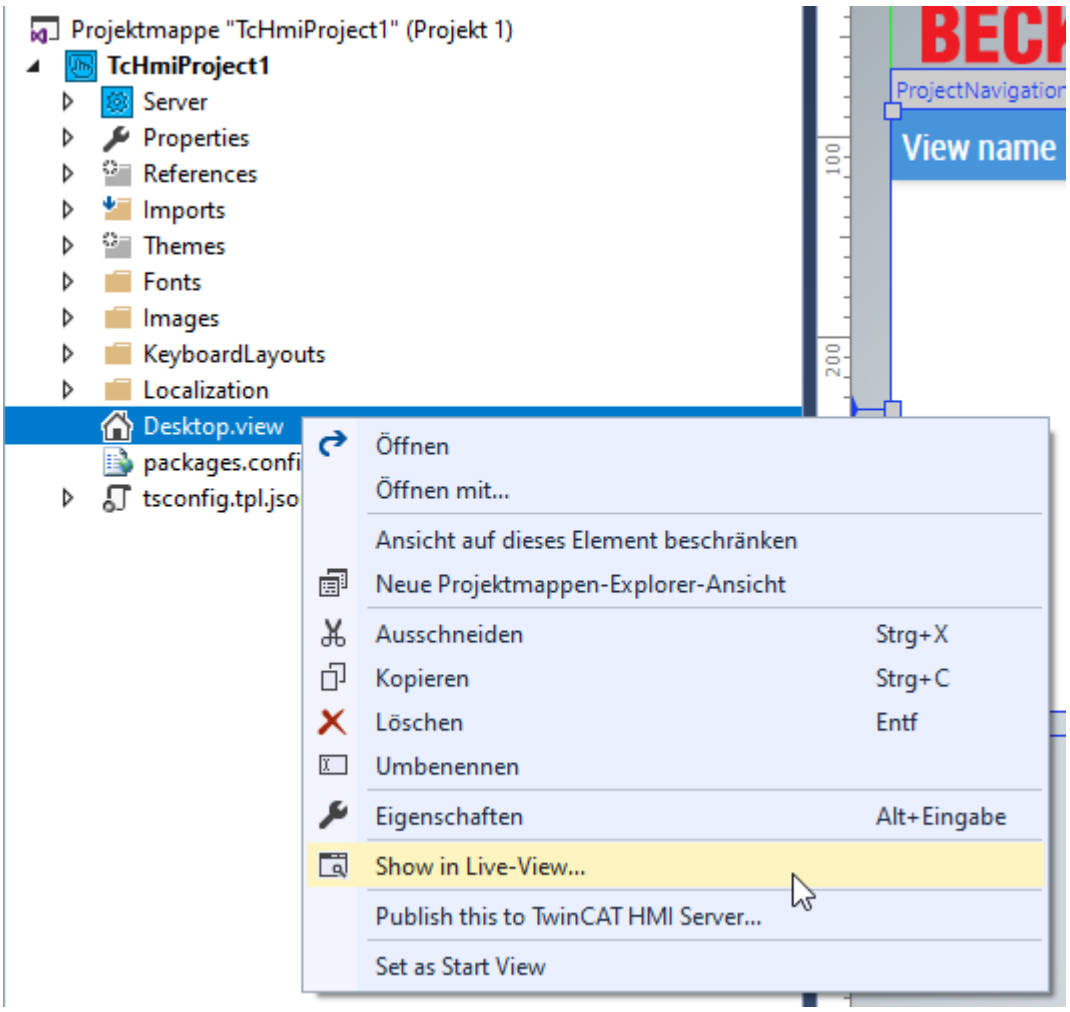
5.1.1 HMI

Erklärung des Beispielprojekts *TF8040-Concept-Samples-HMI*.

Weitere Informationen zu den erforderlichen Schritten sind in der Dokumentation zu den Beispielen im Abschnitt HMI [▶ 89] zu finden.

Inhalt

Im Folgenden werden die einzelnen Beispielseiten des Projekts beschrieben. Dabei empfiehlt es sich, den Live-View zu öffnen, um die Ausführungen leichter nachvollziehen zu können.



Kopfzeile

Die Kopfzeile ist mit verschiedenen Funktionen versehen (v.l.n.r).

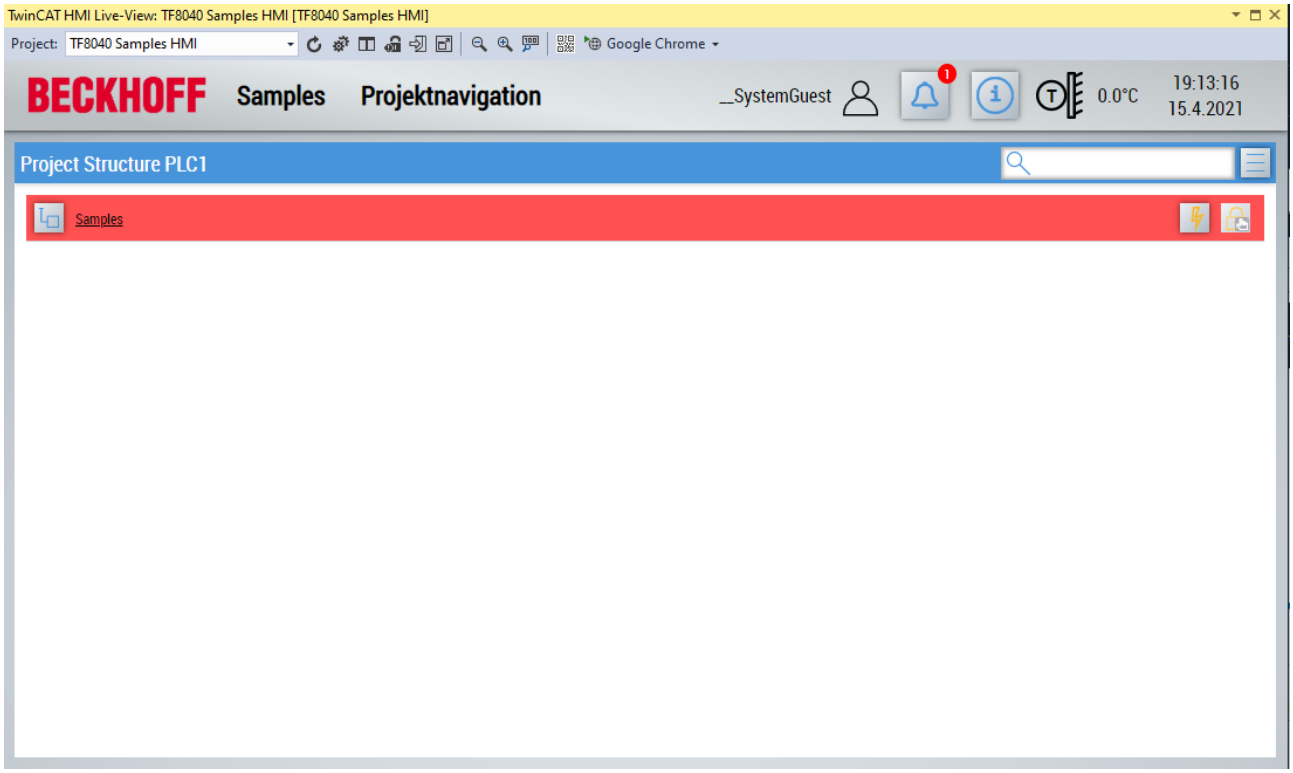


- Logo
- Responsive Navigation
- Benutzereinstellungen und weitere Informationen
- Eventliste
- Gebäudeinformationen
- Außentemperatur
- Datum und Uhrzeit

Weitere Informationen zu den Funktionen sind in der Dokumentation zur Kopfzeile [▶ 1302] zu finden.

Projektnavigation

Als Startseite der Visualisierung wurde die generische Projektnavigation festgelegt. Der Inhalt vom Live-View sollte nach dem Start in etwa so aussehen:



In der Projektnavigation kann durch die Projektstruktur navigiert und die Parameter einzelner Views oder Objekte angezeigt werden.

Weitere Informationen zur Projektnavigation sind in der Dokumentation zur ProjectNavigationTextual [▶ 1186] zu finden.

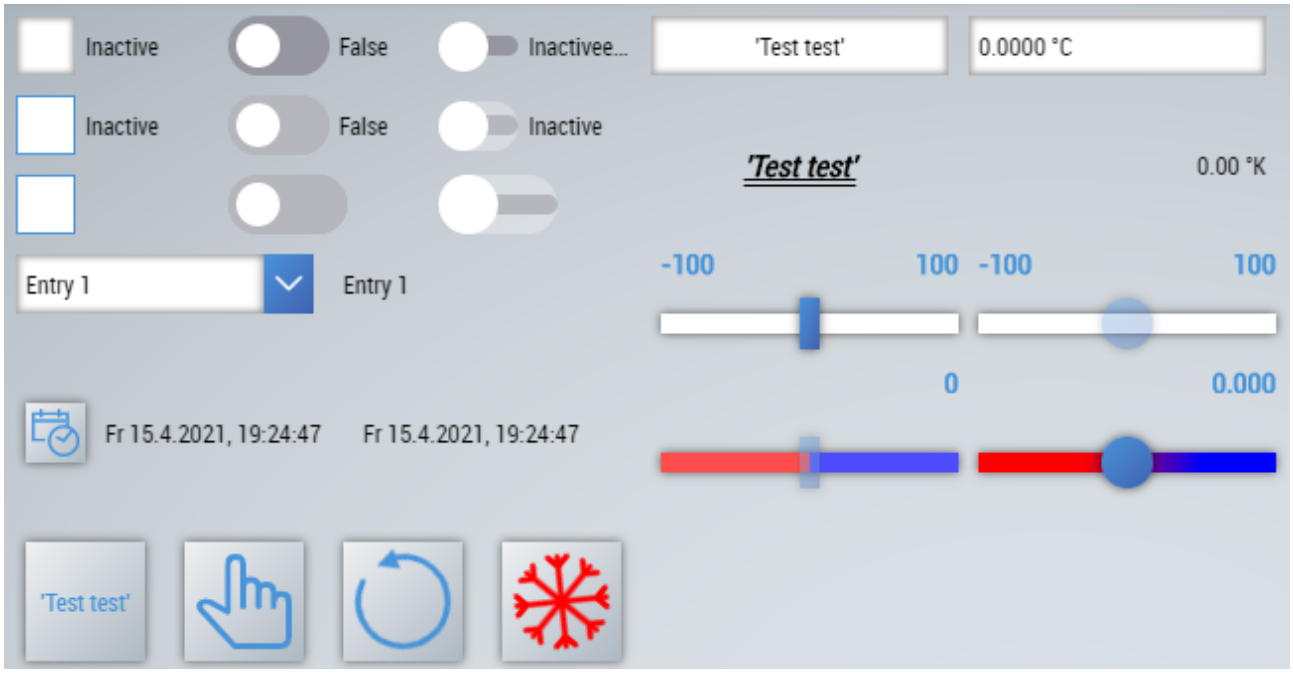
Die folgenden Seiten befinden sich unter dem Eintrag *Content\Samples*.

BasicComponents

Diese Seite stellt alle Controls dar, die in der Toolbox-Kategorie *BA* | *Common* abgelegt sind.

Die Controls sind nicht mit Variablen aus der SPS verbunden.

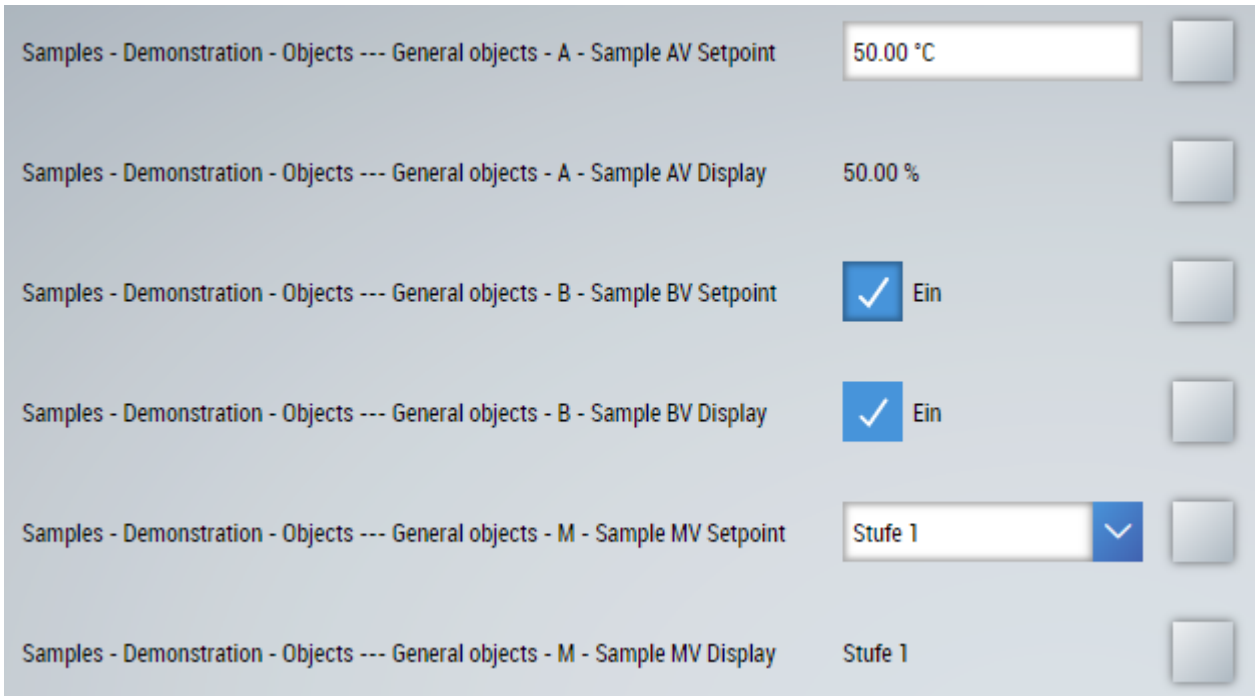
Weitere Informationen sind in den Dokumentationen der Controls [▶ 1141] zu finden.



BaObjects

Auf dieser Seite ist für jeden primitiven Datentypen (*Analog*, *Binary* und *Multistate*) jeweils ein entsprechendes Control abgelegt.

Die Controls sind mit Objekten aus der SPS verbunden und die Werte werden in die SPS geschrieben und wieder gelesen.



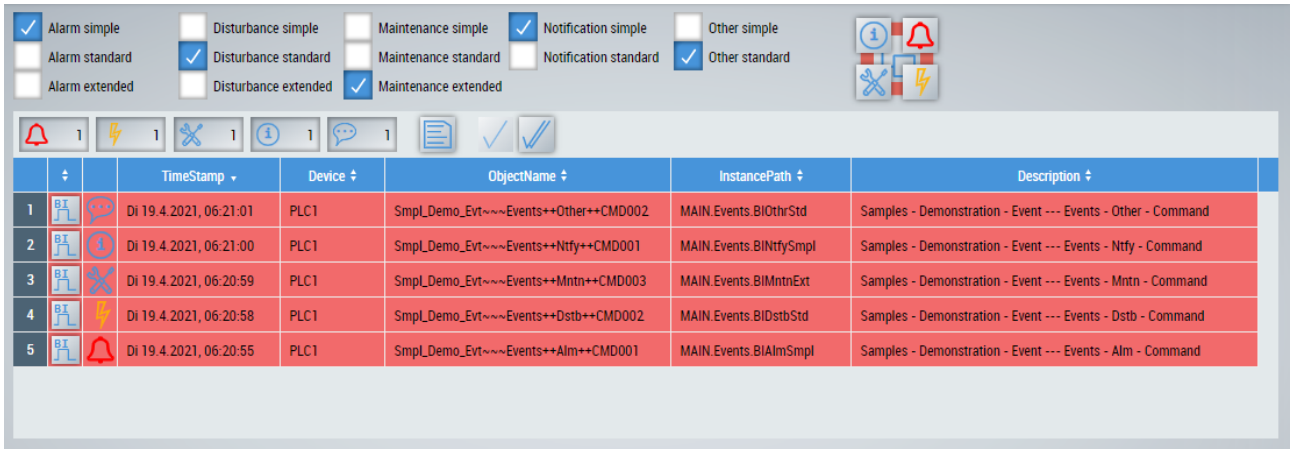
Pro Zeile werden folgende Informationen angezeigt:

- Die *Description* des Objektes.
- Der Wert des Objektes (beschreibbar oder nur lesbar).
- Button zum Öffnen der Projektnavigation des Objektes (nur ein Eintrag sichtbar, da es sich hier um einzelne Objekte handelt).

Event

Die Simulation der verschiedenen Eventtypen ist auf dieser Seite möglich.

Über die Checkboxen kann das jeweilige Event aktiviert und das Verhalten in der darunter liegenden Eventliste beobachtet werden. Ein *UiIcon* zur Anzeige der Events ist ebenfalls auf der Seite positioniert. Der View, in dem sich die Beispielevents befinden, ist sowohl mit der Eventliste als auch mit dem *UiIcon* verknüpft. Deshalb werden auf dieser Seite keine Events außerhalb des Views angezeigt.

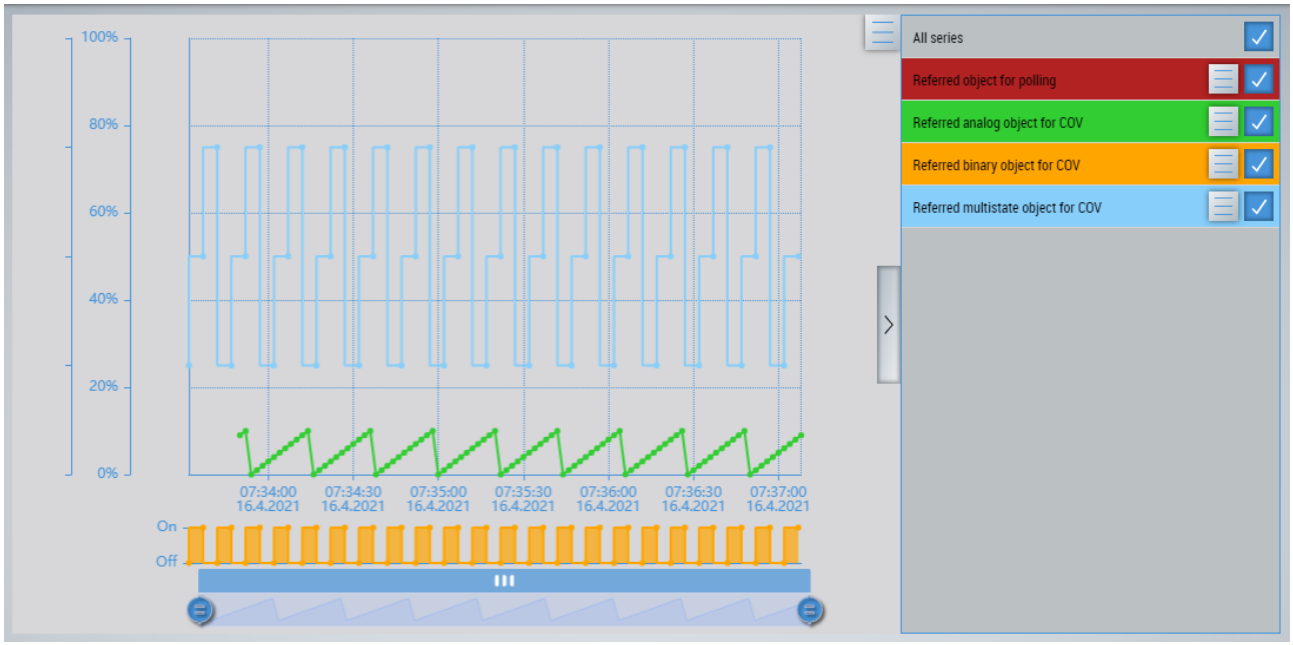


Trend

Auf dieser Seite ist das Trend-Control zur Anzeige verschiedener Trendkurven zu sehen.

In diesem Fall wurde die komplette Projektstruktur mit dem Control verknüpft. Dieses filtert die Projektstruktur nach allen verfügbaren Trends und zeigt sie an. Auf der rechten Seite gibt es die Möglichkeit Trends an- und abzuwählen.

Weitere Informationen sind in der Dokumentation zum Trend [▶ 1194] zu finden.



Schedule

Der Zeitschaltplan zeigt den *aktuellen Zeitschaltplan* und den *wöchentlichen Zeitscheitplan* an. Im Reiter *Kalender* befinden sich die Einträge aus den verknüpften Kalenderreferenzen und den lokalen Ausnahmen.

Die Buttons im unteren Bereich der Seite ermöglichen einen Wechsel zu anderen Schedule-Typen. Daneben kann die Ausrichtung und Genauigkeit des Zeitschaltplans eingestellt werden.

Weitere Informationen sind in der Dokumentation zum Schedule [▶ 1190] zu finden.

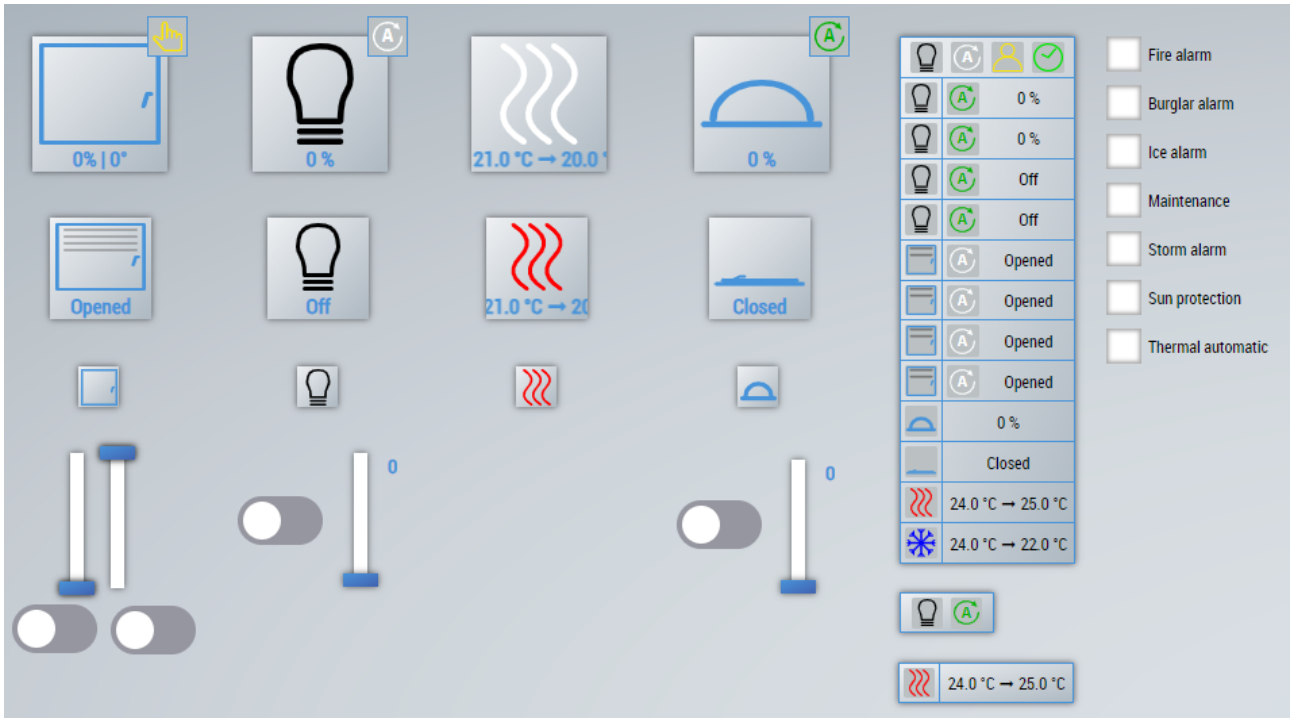


RoomAutomation

Die folgende Auflistung zeigt die verfügbaren Controls für die Raumautomation:

Control	Beschreibung
Sunblind [▶ 1230]	Zeigt und steuert die Position und den Winkel von einem Sonnenschutz.
Light [▶ 1222]	Zeigt und steuert den Helligkeitswert einer Lampe (dimmbar oder ein / aus).
HeatingCooling [▶ 1219]	Zeigt und steuert die Klimatisierung eines Raumes.
Window [▶ 1235]	Zeigt und steuert die Position eines Fensters (prozentual oder auf / zu).
RoomControl [▶ 1226]	Diese Control kann alle oben genannten Controls zusammenfassen.

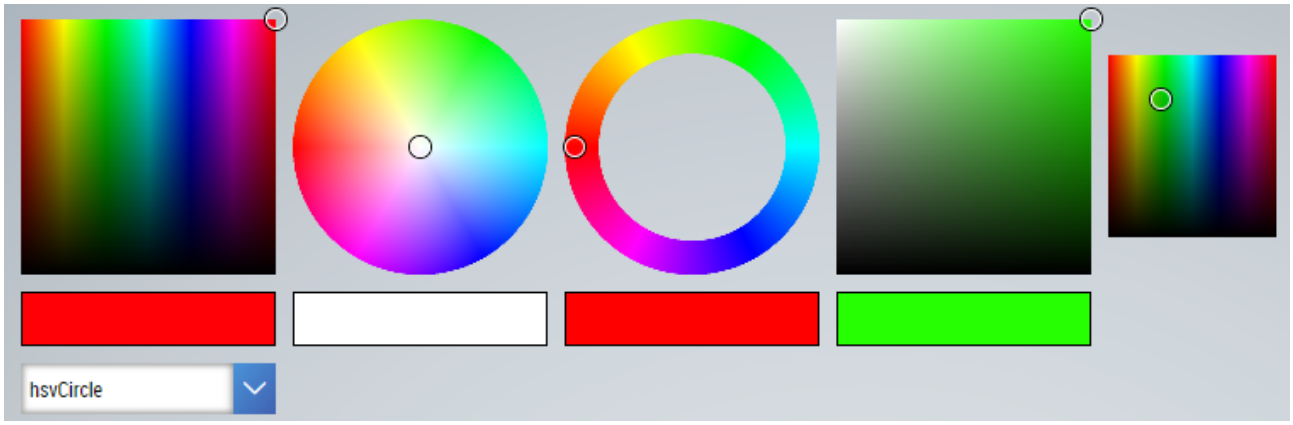
Die Controls dienen nur der Demonstration und sind daher nicht mit Variablen aus der SPS verbunden.



ColorPicker

Auf dieser Seite wird ColorPicker in den verschiedenen Versionen gezeigt.

Weitere Informationen sind in der Dokumentation zum ColorPicker [▶ 1156] zu finden



5.2 Template Beispiele

Die Beispiele in diesem Paket sollen verschiedene Templates in der SPS, HMI und deren Verwendung zeigen.

5.2.1 HMI

Erklärung des Beispielprojekts TF8040 *Template Samples HMI*.

Weitere Informationen zu den erforderlichen Schritten sind in der Dokumentation zu den Beispielen im Abschnitt HMI [▶ 89] zu finden.

Bestandteile

Im Folgenden werden die einzelnen Beispielseiten des Projekts beschrieben.



Der geöffnete Live-View zeigt viele Fehler an. Das liegt daran, dass Objekte in der SPS einen Fehler signalisieren, wenn keine physikalischen Eingänge angeschlossen sind.

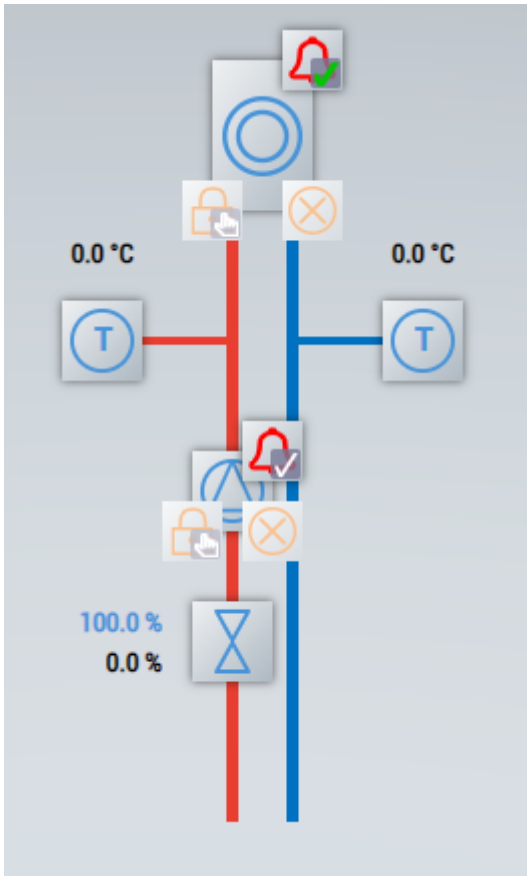
Projektnavigation

Auf dem Inhalt dieser Seite wird die generische Navigation [▶ 1186] angezeigt.

Anlagen

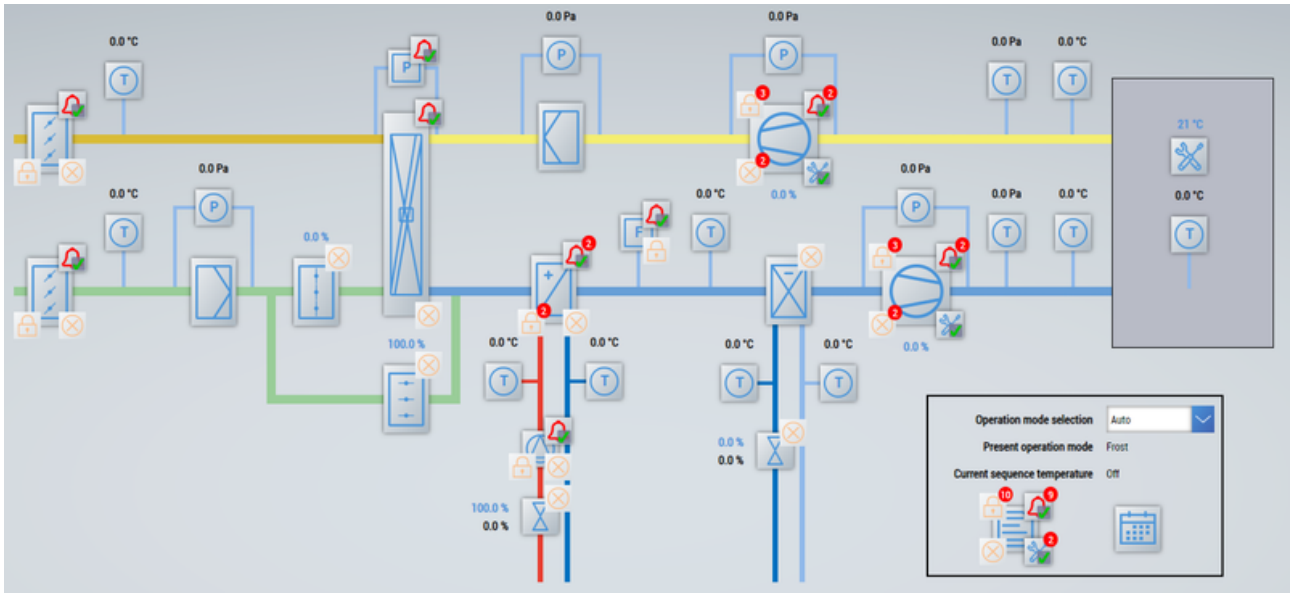
Heizkreis

Zeigt einen Heizkreis an.



RLT

Zeigt eine Lüftungsanlage an.



Raum

Zeigt einen Beispielraum an.



5.2.2 PLC

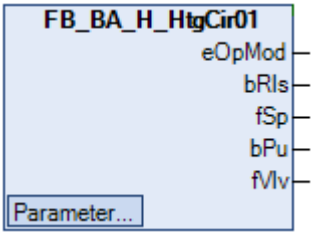
Erklärung des Ablaufs zum Starten des Beispielprojekts TF8040 Template Samples PLC und des Inhalts.

Inhalt

Im Folgenden werden die einzelnen Templates beschrieben, die in dem Projekt angewendet werden.

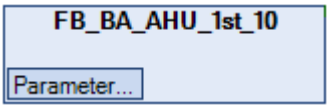
Heizkreis

Verwendung des Templates FB_BA_HtgCir01 [▶ 993].



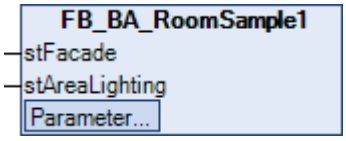
Lüftungsanlage

Verwendung des Templates FB_BA_AHU_1st_10 [▶ 823].



Raumbeispiel

Verwendung des Templates FB_BA_RoomSample1 [▶ 947].



6 Programmierung

Bei neuen Projekten empfehlen wir dringend die Verwendung der Version 5 von TF8040!

6.1 SPS

6.1.1 Allgemein

Allgemeine Informationen zur SPS-Programmierung mit TwinCAT.

Offline

Regionen

Verschiedene Parameter sind für unterschiedliche Anwendungsfälle gedacht.

Zwecks einfacherer Zuordnung werden Parameter innerhalb der folgenden Regionen deklariert:

Region	Beschreibung
Fixed Parameters	Parameter-Variablen zur Konfiguration, die für eine Initialisierung [▶ 270] angepasst werden können. Nachdem ein Objekt initialisiert wurde, ist die Variable nicht mehr für Änderungen zur Laufzeit ausgelegt!
Variable Parameters	Parameter-Variablen zur Konfiguration, die beliebig (d.h. auch zur Laufzeit) angepasst werden können.
Operational Parameters	Parameter-Variablen zur Bedienung, die durch den Anwender beliebig (d.h. auch zur Laufzeit) angepasst werden können.
Fixed Variables	Von der Anwendung wie die Region *Fixed Parameters* zu betrachten. Vorgesehen sind Variablen aber nur für den Integrator **innerhalb** der TwinCAT-Umgebung! Sie sind nicht von Clients (z.B. Site Explorer [▶ 1306] oder TcHmiBa [▶ 1139]) erreichbar.
Output-Properties	Kennzeichnet Variablen, die nicht mittels VAR_OUTPUT sondern durch ein entsprechendes FB-Property nach außen zugänglich gemacht sind.

Um einen Funktionsbaustein, bezogen auf sein Interface, nicht zu überfrachten werden selten verwendete Variablen in Form von Properties implementiert.

Online

Überprüfung

Nach dem Laden der SPS sollte das Log-Fenster überprüft werden.

Wenn keine Fehlermeldungen auftreten, ist das Programm funktionsbereit. So wird ein definierter Ausgangszustand hergestellt.

6.1.2 Bibliotheken

TwinCAT Building Automation besteht aus mehreren SPS-Bibliotheken.

Diese sind schon in der Systeminstallation vorhanden und werden außerdem mit TwinCAT Building Automation ausgeliefert:

- Tc3_BACnetRev14
  - BACnet Funktionsbausteine
- Tc3_BA2_Common
  - Grundfunktionen welche von TwinCAT Building Automation als auch TF8020 BACnet benötigt werden.

Weitere Bibliotheken kommen ausschließlich mit der Installation von TwinCAT Building Automation:

- Tc3_BA2 [▶ 272]
  - Applikationsbausteine der Gebäudeautomation für die Anlagen u. Raumautomation
  - mathematische Grundfunktionen
  - logische Grundfunktionen
- Tc3_XBA [▶ 101]
  - Grundgerüst
  - Site Server für die Kommunikation mit dem Site Explorer und der TwinCAT HMI
  - Objekte

6.1.2.1 Tc3_XBA

Die SPS-Bibliothek Tc3_XBA ist ein wesentliches Element der TwinCAT Function TF8040.
Die wichtigste Komponente dieser Bibliothek ist das Grundgerüst inklusive dem Site-Server.

6.1.2.1.1 DUTs

6.1.2.1.1.1 Enumerationen

6.1.2.1.1.1.1 E_BA_Role

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

Name	Beschreibung
eGuest	Lesezugriff des Present Value für alle Nutzer.
eBasic	Lesezugriff auf wenige Propertys für eine begrenzte Nutzeranzahl
eAdvanced	Zugriff für fortgeschrittene Nutzer
eExpert	Zugriff für Experten
eInternal	Zugriff für Service-Mitarbeiter
eLocked	Kein Zugriff

6.1.2.1.1.1.2 Communication

6.1.2.1.1.1.2.1 E_BA_ComState

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Error	Ein Problem ist aufgetreten und das Objekt wurde von der Ausführung ausgeschlossen.
eUnused	Das Kommunikationsobjekt wird von keinem anderen Objekt genutzt.
eInitialization	Das Objekt wird initialisiert.
eOperation	Das Objekt ist in Betrieb.

6.1.2.1.1.1.3 Control

6.1.2.1.1.1.3.1 E_BA_StateLoopSync

E_BA_StateLoopSync zeigt den Status der Synchronisation des FB_BA_LoopSync [▶ 169] an.

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

Name	Beschreibung
eOk	Die Synchronisation ist fehlerfrei.
eErrAction	Synchronisation von *eActionRm* fehlerhaft.
eErrDampConstant	Synchronisation von *nDampConstant* fehlerhaft.
eErrDerivativeConstant	Synchronisation von *fDerivativeConstant* fehlerhaft.
eErrIntegralConstant	Synchronisation von *fIntegralConstant* fehlerhaft.
eErrMaxOutputRm	Synchronisation von *fMaxOutputRm* fehlerhaft.
eErrMinOutputRm	Synchronisation von *fMinOutputRm* fehlerhaft.
eErrNeutralZone	Synchronisation von *fErrNeutralZone* fehlerhaft.
eErrOpMode	Synchronisation von *eOpMode* fehlerhaft.
eErrProportionalConstant	Synchronisation von *fProportionalConstant* fehlerhaft.

6.1.2.1.1.1.4 DPAD

6.1.2.1.1.1.4.1 E_BA_ConcatDPADMode

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

```
  eNone          := 1,
  eEntryPoint    := 2,
  eParent        := 3
) BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eNone	Das Event steht nicht an.
eEntryPoint	Verknüpfung der Instanz mit einem definierten Einstiegspunkt, z.B. Anlage.
eParent	Verknüpfung der Instanz mit dem Parent.

6.1.2.1.1.1.4.2 E_BA_DPADMode

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eExclude	Keine Anwendung des Objektnamens und der Description.
eInclude	Anwendung des Objektnamens und der Description.
eIncludeObjectName	Anwendung des Objektnamens.
eIncludeDescription	Anwendung der Description.

6.1.2.1.1.1.5 Events

6.1.2.1.1.1.5.1 E_BA_AcknowledgeMode

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

Name	Beschreibung
eSingle	Quittiert ausschließlich die nächste anstehende Ereignis-Transition.
eEntire	Quittieren aller unquittierten Ereignis-Transitionen durch eine einzige Quittierung.

6.1.2.1.1.1.5.2 E_BA_AlarmMode

Beschreibt den Alarm-Modus von eventfähigen Objekten.

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

Name	Beschreibung
Undefined	Keine Funktion
UserDefined	Benutzerdefiniertes Muster Bitkombination [TO-OFFNORMAL | TO-FAULT | TO-NORMAL] ACK_REQUIRED x | x | x EVENT_ENABLED x | x | x Gemäß BACnet kann eine beliebige Kombination der AcknowledgeRequired Bits (0 oder 1) genutzt werden. Die EventNotification kann wahlweise als Event oder als Alarm erfolgen Quittierung: für jeden Zustandsübergang (TO-OFFNORMAL, TO-NORMAL und TO-FAULT) kann definiert werden, ob eine Quittierung notwendig ist oder nicht.
eSimple	Weder kommende noch gehende Alarme müssen bestätigt werden.
eStandard	Es müssen nur kommende, nicht aber ausgehende Alarme bestätigt werden. Eine Quittierung, aber kein Rücksetzen des Alarms ist erforderlich.
eExtended	Es wird sowohl die Quittierung (Acknowledgment) als auch die Rücksetzung (Reset) des Alarms gefordert. Je nach Quittierungs-Modus reicht möglicherweise ein einfaches Quittieren aus, um das Rücksetzen auszulösen.

6.1.2.1.1.1.5.3 E_BA_EventCondition

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

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

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

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

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

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

6.1.2.1.1.1.5.4 E_BA_EventIconState

```
TYPE E_BA_EventIconState :
(
  Invalid          := 0,
  eNone            := 1,
  eIndicated       := 2,
  eGoneAcked       := 3,
  eGone            := 4,
  ePresentAcked    := 5,
```

```
  ePresent          := 6,
) BYTE;
END_TYPE
```

Name	Beschreibung
eNone	 Das Event steht nicht an.
eIndicated	 Das Event steht nicht (mehr) an, wird jedoch zu Informationszwecken angedeutet bis es quittiert wird*.
eGoneAcked	 Das Event steht nicht (mehr) an. Es wurde aber bereits quittiert aber noch nicht zurückgesetzt**.
eGone	 Ein Event steht nicht (mehr) an. Es wurde aber weder quittiert noch zurückgesetzt*.
ePesentAcked	 Ein Ereignis steht an und ist quittiert.
ePresent	 Ein Event steht an.



* Nur möglich bei Alarm-Modus *standard*!
** Nur möglich bei *erweitertem* Alarm-Modus!

6.1.2.1.1.1.5.5 E_BA_EventType

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

Typ	Symbol in der TwinCAT HMI	E_BA_EventType
Alarm		eAlarm
Störung		eDisturb
Wartung		eMaintenance
Benachrichtigung		eNotification
Sonstiges		eOther

6.1.2.1.1.1.5.6 E_BA_ObjectStateFlags

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

6.1.2.1.1.1.6 Groups

6.1.2.1.1.1.6.1 E_BA_AValCalcMode

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

6.1.2.1.1.1.6.2 E_BA_BValCalcMode

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

6.1.2.1.1.1.6.3 E_BA_MValCalcMode

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

6.1.2.1.1.1.7 Objects

6.1.2.1.1.1.7.1 E_BA_ObjectPurpose

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

Name	Beschreibung
eInternal	Interne Managementfunktionen (z. B. EventClass-Objekte)
eStructurize	Organisation der Projektstruktur (siehe Structured-View-Objekte in TF8020)
eDescriptive	Beschreibende Informationen (z.B. Projekt-Objekt)
eOperation	Operativer Wert (Sollwert, Anzeigewert)
eValue	Wert
eInput	Physikalischer Eingangswert
eOutput	Physikalischer Ausgangswert
eManagement	Verwaltet referenzierte Objekte (Anlage, Steuerung, …)

6.1.2.1.1.1.7.2 Loop

6.1.2.1.1.1.7.2.1 E_BA_LoopSeqState

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eNoSequence	Keine Sequenz aktiv
eInactive	Regler inaktiv
eWaiting	Warten auf Token.
eActive	Operational (Die aktuelle Instanz hat das Token.)
ePassed	Die Instanz ist am Maximum und hat das Token weitergegeben.

6.1.2.1.1.1.7.2.2 E_BA_SeqDirection

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

Name	Beschreibung
eNoSequence	Keine Sequenz aktiv
eLeft_to_Right	Die Sequenz startet am Loop auf der linken Seite, der Token wandert zu den nächsten Loops.
eMiddle_to_LeftOrRight	Sequenz startet in der Mitte, der Token wird an den nächsten ODER vorherigen Loop übergeben (abhängig vom Wirksinn der Sequenz).
eRight_to_Left	Die Sequenz startet am Loop auf der rechten Seite, der Token wandert zu den vorherigen Loops.

6.1.2.1.1.1.7.2.3 E_BA_SeqState

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eInactive	Keine Freigabe einer sequenzierten Regelung.
eDetermineToken	Suche des Tokens zur Bestimmung des aktiven Loop-Objektes.
eOperation	Das Objekt ist in Betrieb.

6.1.2.1.1.1.7.3 PlantCtrl

6.1.2.1.1.1.7.3.1 E_BA_AggregateIgnoreFlags

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

Name	Beschreibung
eProcesses	Ignoriert Prozesse eines Aggregats.
eEvents	Ignorieren der Ereignisse eines Aggregats.
eDelayStepDown	Ignoriert eine Herunterschaltverzögerung vom Aggregat.
eDelayStepUp	Ignoriert eine Heraufschaltverzögerung vom Aggregat.

6.1.2.1.1.1.8 Parameters

6.1.2.1.1.1.8.1 E_BA_Attribute

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

6.1.2.1.1.1.8.2 E_BA_CommissioningState

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eUnknown	Status unbekannt
eChecked	Geprüft
eDefectIO	Eingang bzw. Ausgang defekt
eDefectWiring	Leitung defekt
eDefectDevice	Feldgerät defekt
eDefectOther	Anderer Defekt

6.1.2.1.1.1.8.3 E_BA_ParamCOVMode

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eNone	COV-Anzeige des Parameters ist inaktiv.
eStandard	COV-Anzeige des Parameters wird periodisch ausgewertet (abhängig von der Auslastung).
eManual	COV-Anzeige wird ausgeführt, aber die veränderten Werte müssen manuell gelesen, bzw. geschrieben werden.
ePrioritized	COV-Anzeige des Parameters wird in jedem SPS-Zyklus ausgewertet.

6.1.2.1.1.1.8.4 E_BA_Parameter

Mögliche Properties eines BACnet-Objekts.

```
TYPE E_BA_Parameter :
(
  Invalid                    := 0,
  eConfigurate               := 1,
  eToggleMode                := 2,
  eStepDelay                 := 3,
  eCOVIncrement              := 4,
  eAction                    := 5,
  eMinOffTime                := 6,
  eMinOnTime                 := 7,
  eStateChangeCount          := 8,
  eStateChangeTime           := 9,
  eStateChangeResetPoint     := 10,
  eActiveTimeElapsed         := 11,
  eActiveTimeResetPoint      := 12,
  eInstructionText           := 13,
  eAckedTransitions          := 14,
  eAcknowledgeRm             := 15,
  eEnPlantLock               := 16,
  eAlarmValue                := 17,
  eAlarmValues               := 18,
  eFaultValues               := 19,
  eTimeDelay                 := 20,
  eLowLimit                  := 21,
  eHighLimit                 := 22,
  eLimitDeadband             := 23,
  eEventDetectionEnable      := 24,
  eEventEnable               := 25,
  eEventClassID              := 26,
  eEventTransitionText       := 29,
  eEventState                := 30,
  eStatusFlags               := 31,
  eReliability               := 32,
  eEnable                    := 33,
  eOutOfService              := 34,
  eInactiveText              := 35,
  eActiveText                := 36,
  eStateText                 := 37,
  eStateCount                := 38,
  eUnit                      := 39,
  ePresentValue              := 40,
  eDefaultValue              := 41,
  eTag                       := 42,
  eAssignAsTrendReference    := 43,
  eDeviceType                := 44,
  ePolarity                  := 45,
  eMappingMode               := 46,
  eFeedbackMappingMode       := 47,
  eFeedbackPolarity          := 48,
  eOverriddenPolarity        := 49,
  eScaleOffset               := 50,
  eResolution                := 51,
  eFeedbackValue             := 52,
  eRawValue                  := 53,
  eRawFeedback               := 54,
  eRawOverride               := 55,
  eRawState                  := 56,
  eTerminal                  := 57,
  eSensor                    := 58,
  eAddress                   := 59,
  eCommissioningState        := 60,
  eSymbolPath                := 61,
  eSymbolName                := 62,
  eInstanceID                := 63,
  eObjectName                := 64,
  eDescription               := 65,
  eObjectType                := 66,
  ePurpose                   := 67,
  eNodeType                  := 68,
  ePriorityArray             := 69,
  eActivePriority            := 70,
  eProjectInfo               := 71,
  eOperatorInfo              := 72,
  eTechnicalStaffInfo        := 73,
  eEngineerInfo              := 74,
  eDateList                  := 75,
```

```
  eEventType                := 76,
  eAlarmMode                := 77,
  ePriority                 := 78,
  eAcknowledgeRequired      := 79,
  eSetpoint                 := 80,
  eControlledValue          := 81,
  eCtrlDeviation            := 82,
  eProportionalConstant     := 83,
  eIntegralConstant         := 84,
  eDerivativeConstant       := 85,
  eDampConstant             := 86,
  eOutputUnit               := 87,
  eOpMode                   := 88,
  eNeutralZone              := 89,
  eSynchronizedLoop         := 90,
  eMinOutput                := 91,
  eMaxOutput                := 92,
  eWeek                     := 93,
  eExceptionList            := 94,
  eReferencedParam          := 95,
  eBufferSize               := 96,
  eLogBuffer                := 97,
  eLoggingType              := 98,
  eLogInterval              := 99,
  eStartTime                := 100,
  eStopTime                 := 101,
  eStopOnFull               := 102,
  eNotificationThreshold    := 103,
  eRecordCount              := 104,
  eTotalRecordCount         := 105
) BYTE;
END_TYPE
```

6.1.2.1.1.1.8.5 E_BA_ParamPurpose

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eUnique	Einzigartiger Parameterwert, der nicht durch mehrere Objekte genutzt werden soll.
eIndividual	Individueller Parameter pro Objekt-Instanz.
eIndividualSetting	Individuelle Einstellung pro Objekt-Instanz.
eSetting	Generelle Einstellung

6.1.2.1.1.1.8.6 E_BA_ParamSyncMode

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

Name	Beschreibung
eInitialWrite	Der Parameter wird durch einen einzigen Schreibbefehl initialisiert.

6.1.2.1.1.1.9 Priority

6.1.2.1.1.1.9.1 E_BA_LockPriority

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eNoLock	Es wird keine Schalthandlung ausgelöst.
eLocalMedium	Das Ereignis eines Objektes löst eine Schalthandlung mittels des FB_BA_PlantLock [▶ 138] aus, der sich innerhalb des Grundgerüstes in der gleichen Ebene befindet. Die Höhe der Priorität ist „Mittel“ und bezieht sich im allgemeinen auf eine technische Störung bzw. auf die Sicherheit einer Anlage.
eLocalHigh	Das Ereignis eines Objektes löst eine Schalthandlung mittels des FB_BA_PlantLock [▶ 138] aus, der sich innerhalb des Grundgerüstes in der gleichen Ebene befindet. Die Höhe der Priorität ist „Hoch“ und bezieht sich im allgemeinen auf einen Alarm bzw. die Sicherheit von Leben.
eMedium	Das Ereignis eines Objektes löst eine Schalthandlung aus. Es wird an alle FB_BA_PlantLock [▶ 138] gemeldet, die sich in den Ebenen des Grundgerüstes darüber befinden. Die Priorität ist „Mittel“ und bezieht sich im allgemeinen auf eine technische Störung bzw. auf die Sicherheit einer Anlage.
eHigh	Das Ereignis eines Objektes löst eine Schalthandlung aus. Es wird an alle FB_BA_PlantLock [▶ 138] gemeldet, die sich in den Ebenen des Grundgerüstes darüber befinden. Die Priorität ist „Hoch“ und bezieht sich im allgemeinen auf einen Alarm bzw. die Sicherheit von Leben.

6.1.2.1.1.1.9.2 E_BA_Priority

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eDefault	Grundeinstellung, wenn keine Priorität gewählt ist.
eProgram	Steuerung über das Programm.
eManualRemote	Manuelle Übersteuerung durch Parameter.
eManualLocal	Manuelle Übersteuerung, z. B. über Schalter.
eCritical	Einstellung zum Betrieb kritischer Anlagenteile
eLifeSafety	

6.1.2.1.1.1.10 References

6.1.2.1.1.1.10.1 E_BA_AssignRefMode

Bestimmung eines Objektes als Referenz in einem zugehörigen (z. B. Trend) Objekt.

```
TYPE E_BA_AssignRefMode :
(
  eNone             := 1,
  eAssignNow        := 2,
  eInitByProfile    := 3
) BYTE;
END_TYPE
```

Name	Beschreibung
eAssignNow	Referenz unmittelbar zuweisen
eInitByProfile	Referenz bei der Initialisierung des Objektes zuweisen, sofern es im zugehörigen Objektprofil freigegeben ist.

6.1.2.1.1.1.11 Supplements

6.1.2.1.1.1.11.1 E_BA_SupplementType

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

6.1.2.1.1.1.12 Types

6.1.2.1.1.1.12.1 E_BA_NodeType

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Automatic	Das System wählt einen aussagekräftigen Knotentypen.

6.1.2.1.1.1.12.2 E_BA_ObjectType

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

6.1.2.1.1.1.13 Functional

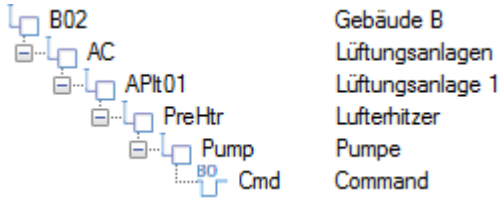
6.1.2.1.1.1.13.1 E_BA_NodeTypeTarget

Die Enumeration beschreibt den relativen Bezug zu einem Objekt.

Beispiel

In der folgenden Projektstruktur ist:

- die Lüftung *Lüftungsanlage 1* eine **Anlage**
- der Lufterhitzer *PreHtr* ein **Aggregat**
- die Pumpe *Pump* ein **Aggregat**
- der Befehl *Cmd* eine **Funktion**



Bezogen auf die Anlage *Lüftungsanlage 1* ist:

- der Lufterhitzer *PreHtr* das **erste Aggregat**
- die Pumpe *Pump* das **letzte Aggregat**
- der Befehl *Cmd* die **Funktion**

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

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eFirstAggregate	Bezieht sich auf das erste Aggregat in Bezug zu einem Objekt.
eLastAggregate	Bezieht sich auf das letzte Aggregat in Bezug zu einem Objekt.
eFunction	Bezieht sich auf die Funktion in Bezug zu einem Objekt.

6.1.2.1.1.1.13.2 E_BA_ProjectState

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

6.1.2.1.1.1.13.3 E_BA_ObjectState

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

6.1.2.1.1.1.13.4 E_BA_ProcessSignalSource

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

6.1.2.1.1.1.13.5 E_BA_SubscriberState

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

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

```
) BYTE;
END_TYPE
```

6.1.2.1.1.13.6 E_BA_InitState

Die Enumeration beschreibt den Status der Initialisierung.

Syntax

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

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

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

  ePreFirstExecute           := 9,
  ePostFirstExecute          := 10

) BYTE;
END_TYPE
```

Name	Beschreibung
ePreInitInstance	Vorinitialisierung der Instanz.
eInitInstanceParam	Initialisierung der Instanzparameter.
eInitInstanceDependency	Initialisierung der Instanzabhängigkeiten.
ePostInitInstance	Nachinitialisierung der Instanz.
eInitComDevice	Initialisierung des ComDevice.
ePreInitSupplement	Vorinitialisierung von Supplements.
ePostInitSupplementInst	Abschluss der Initialisierung eines Instanzteils eines Supplements.
ePostInitSupplement	Initialisierung von Supplements ist abgeschlossen.
ePreFirstExecute	Vor der Ersten Ausführung.
ePostFirstExecute	Nach der Ersten Ausführung.

6.1.2.1.1.2 Types

6.1.2.1.1.2.1 ST_BA_ObjectAttributes

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

6.1.2.1.1.2.2 ST_BA_SubjectAttributes

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

6.1.2.1.1.2.3 DPAD

6.1.2.1.1.2.3.1 ST_BA_DPAD_Identifier

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

6.1.2.1.1.2.4 Events

6.1.2.1.1.2.4.1 ST_BA_EventIcon

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

6.1.2.1.1.2.4.2 ST_BA_EventsPerIconImage

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

Name	Typ	Beschreibung
eMostPriorisedState	E_BA_EventIconState [▶ 104]	Am höchsten priorisierter Icon-Status mit einem oder mehreren aktiven Ereignissen.
nCount	UDINT	Anzahl der aktiven Ereignisse in Abhängigkeit zum am höchsten priorisierten Icon-Status.

6.1.2.1.1.2.4.3 T_BA_EventConditionFlags

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

6.1.2.1.1.2.4.4 T_BA_EventTransitions

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

6.1.2.1.1.2.4.5 T_BA_EventTransitionText

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

6.1.2.1.1.2.4.6 History

6.1.2.1.1.2.4.6.1 ST_BA_EventHistoryEntry

```
TYPE ST_BA_EventHistoryEntry :
STRUCT
  iObject            : I_BA_EventObject;
  nEventIncrement    : UINT;
  dtTimeStamp        : DT;
```

```
  stEvent            : ST_BA_EventIcon;
  eReliability       : E_BA_Reliability;
  stStateFlags       : ST_BA_StatusFlags;
  eLockPriority      : E_BA_LockPriority;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5 Info

6.1.2.1.1.2.5.1 ST_BA_ComStat

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

Name	Typ	Beschreibung
nReq	UDINT	Erhaltene Anfragen
nBadRsp	UDINT	Anfragen, die einen ADS-Fehler als Antwort erzeugt haben.
nGoodRsp	UDINT	Anfragen, die keinen ADS-Fehler als Antwort erzeugt haben.

6.1.2.1.1.2.5.2 ST_BA_ContactInfo

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

6.1.2.1.1.2.5.3 ST_BA_CumObjState

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

Name	Typ	Beschreibung
nPendingInit	UDINT	Anzahl der Objekte, die initialisiert werden sollen.
nOperational	UDINT	Anzahl der betriebsbereiten Objekte (Initialisierung abgeschlossen).
nError	UDINT	Anzahl an Objekten im Fehler-Status (Initialisierung fehlgeschlagen).

6.1.2.1.1.2.5.4 ST_BA_DeviceInfo

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

6.1.2.1.1.2.5.5 ST_BA_ProjectInfo

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

6.1.2.1.1.2.5.6 ST_BA_RuntimeInfo

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

6.1.2.1.1.2.5.7 ST_BA_SupplementInfo

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

6.1.2.1.1.2.5.8 ST_BA_TaskInfo

```
TYPE ST_BA_TaskInfo :
STRUCT
  {attribute 'parameterUnit':= 'ms'}
  nCycleTime     : UDINT;
  nCycleCount    : UDINT;
  nPriority      : UINT;
  nADSPort       : UINT;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.5.9 ST_BA_VersionInfo

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

6.1.2.1.1.2.5.10 Supplements

6.1.2.1.1.2.5.10.1 ST_BA_SplInfo_BACnet

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

6.1.2.1.1.2.5.10.2 ST_BA_SplInfo_PLC

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

6.1.2.1.1.2.5.10.3 [Base]

6.1.2.1.1.2.5.10.3.1 ST_BA_BaseSupplementInfo

```
TYPE ST_BA_BaseSupplementInfo :
STRUCT
  bIsEnabled    : BOOL;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.6 Objects

6.1.2.1.1.2.6.1 ST_BA_ObjectIdentifier

Die Struktur ST_BA_ObjectIdentifier beinhaltet die Instanz-ID der Objekttypen.

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

6.1.2.1.1.2.6.2 Loop

6.1.2.1.1.2.6.2.1 ST_BA_SeqStandByInfo

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

Name	Typ	Beschreibung
nEnCount	UINT	Zählung freigegebener Instanzen.
iFirstEn	I_BA_LoopSeq	Referenz des ersten freigegebenen Loop-Objektes.
iLastEn	I_BA_LoopSeq	Referenz des letzten freigegebenen Loop-Objektes.
iFirstDmd	I_BA_LoopSeq	Referenz des ersten angeforderten Loop-Objektes.
iLastDmd	I_BA_LoopSeq	Referenz des letzten angeforderten Loop-Objektes.
iLastReverse	I_BA_LoopSeq	Referenz des letzten Loop-Objektes mit indirektem Wirksinn.
iFirstDirect	I_BA_LoopSeq	Referenz des ersten Loop-Objektes mit direktem Wirksinn.
nActionInvertCount	UINT	Zähler der Instanzen, die den Wirksinn umkehren.
nDirectCount	UINT	Zähler der Instanzen mit direktem Wirksinn
nReverseCount	UINT	Zähler der Instanzen mit indirektem Wirksinn.
iActive	I_BA_LoopSeq	Referenz mit dem aktuellen Token.

6.1.2.1.1.2.6.3 Trend

6.1.2.1.1.2.6.3.1 ST_BA_TrendSettings

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

6.1.2.1.1.2.6.4 Value

6.1.2.1.1.2.6.4.1 ST_BA_EventObjectValue

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

6.1.2.1.1.2.6.4.2 ST_BA_ObjectValue

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

6.1.2.1.1.2.6.4.3 ST_BA_SchedValue

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

6.1.2.1.1.2.6.4.4 ST_BA_TrendValue

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

6.1.2.1.1.2.7 Parameter

6.1.2.1.1.2.7.1 ST_BA_LimitParam

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

6.1.2.1.1.2.7.2 ST_BA_ObjectParameter

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

6.1.2.1.1.2.7.3 ST_BA_StepDelayParam

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

6.1.2.1.1.2.7.4 ST_BA_TimeDelayParam

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

6.1.2.1.1.2.7.5 T_BA_MedString

```
TYPE T_BA_MedString : STRING(64); END_TYPE
```

6.1.2.1.1.2.7.6 T_BA_ShortString

```
TYPE T_BA_ShortString : STRING(12); END_TYPE
```

6.1.2.1.1.2.7.7 T_BA_SmallString

```
TYPE T_BA_SmallString : STRING(32); END_TYPE
```

6.1.2.1.1.2.7.8 Calendar

6.1.2.1.1.2.7.8.1 T_BA_CalendarDateList

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

6.1.2.1.1.2.7.9 Multistate

6.1.2.1.1.2.7.9.1 T_BA_StateText

```
TYPE T_BA_StateText : STRING(XBA_Param.nStateText_Length); END_TYPE
```

6.1.2.1.1.2.7.9.2 T_BA_StateTextArray

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

6.1.2.1.1.2.7.10 References

6.1.2.1.1.2.7.10.1 ST_BA_AttributeInfo

Beschreibt Details für definierte Attribute.

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

6.1.2.1.1.2.7.10.2 ST_BA_SyncInfo

```
TYPE ST_BA_SyncInfo :
STRUCT
  bChanged    : BOOL;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.7.10.3 U_BA_ParamRef

```
TYPE U_BA_ParamRef :
UNION
  fb1P        : POINTER TO FB_BA_1ParamRef;
  fb2P        : POINTER TO FB_BA_2ParamRef;
  fb3P        : POINTER TO FB_BA_3ParamRef;
  fb4P        : POINTER TO FB_BA_4ParamRef;
  fb5P        : POINTER TO FB_BA_5ParamRef;
  fbArrP      : POINTER TO FB_BA_ArrayParamRef;
  fbEnP       : POINTER TO FB_BA_EnParamRef;
  fbPrioP     : POINTER TO FB_BA_PrioParamRef;
END_UNION
END_TYPE
```

6.1.2.1.1.2.7.11 Scheduler

6.1.2.1.1.2.7.11.1 ST_BA_SchedCalendar

Die Struktur ST_BA_SchedCalendar beschreibt eine Kalenderreferenz aus einer Ausnahme (ExceptionSchedule).

```
TYPE ST_BA_SchedCalendar :
STRUCT
  iRefCalendar          : I_BA_Object;

  aEntry                : ARRAY[1 .. XBA_Param.nSched_EntryCount] OF ST_BA_SchedEntry;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.7.11.2 ST_BA_SchedException

Die Struktur ST_BA_SchedException beschreibt einen Kalendersatz von Ausnahmen (ExceptionSchedule).

```
TYPE ST_BA_SchedException :
STRUCT
  eType     : E_BA_DateValChoice := E_BA_DateValChoice.Invalid;
```

```
  uDate     : U_BA_DateVal;
  aEntry    : T_BA_SchedExceptionEntryList;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.7.11.3 T_BA_SchedCalendar

Die Typdeklaration fasst einzelne Ausnahmen zu einem Array für den Exception Scheduler zusammen.

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

6.1.2.1.1.2.7.11.4 T_BA_SchedExceptionEntryList

Die Typdeklaration fasst einzelne Ausnahmen zu einem Array für den Exception Scheduler zusammen.

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

6.1.2.1.1.2.7.11.5 T_BA_SchedExceptionList

Die Typdeklaration fasst einzelne Ausnahmen zu einem Array für den Exception Scheduler zusammen.

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

6.1.2.1.1.2.7.11.6 T_BA_SchedWeek

Die Typdeklaration fasst einzelne Scheduler-Einträge zu einem Array für ein Wochenzeitschaltprogramm zusammen.

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

6.1.2.1.1.2.7.12 Trend

6.1.2.1.1.2.7.12.1 T_BA_TrendLogBuffer

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

6.1.2.1.1.2.8 Stepped

6.1.2.1.1.2.8.1 ST_BA_ActiveInfo

```
TYPE ST_BA_ActiveInfo :
STRUCT
  iReference          : I_BA_Object;
  {attribute 'TcEncoding':='UTF-8'}
  sSymbolName         : REFERENCE TO STRING;
  {attribute 'TcEncoding':='UTF-8'}
  sDescription        : REFERENCE TO T_MaxString;
  iBlockingProcess    : I_BA_Process;
END_STRUCT
END_TYPE
```

6.1.2.1.1.2.8.2 ST_BA_PlantAggregateReference

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

```
{endregion}
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bWaitForProcesses	BOOL	Aktive Prozesse, die berücksichtigt werden, beim Anfahren von Stufen.
bWaitForEvents	BOOL	Ereignisse dieser (und einer höheren) Lock-Priorität, die beim Wechseln der Stufen berücksichtigt werden.

6.1.2.1.1.2.8.3 ST_BA_PlantOperation

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

6.1.2.1.1.2.9 Supplements

6.1.2.1.1.2.9.1 ST_BA_SupplementRef

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

6.1.2.1.1.2.9.2 BACnet

6.1.2.1.1.2.9.2.1 ST_BA_BACnet_PropSyncInfo

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

6.1.2.1.1.2.10 Universal

6.1.2.1.1.2.10.1 Sequence

6.1.2.1.1.2.10.1.1 ST_BA_SeqLink

Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem Steuerbaustein FB_BA_SequenceLinkBase [▶ 175].

```
TYPE ST_BA_SeqLink :
STRUCT
  arrSeqLinkData    : ARRAY[1..BA_Param.nMaxSeqCtrl] OF ST_BA_SeqLinkData;
  fE                : REAL;
  nActvSeqCtrl      : UDINT;
  bSync             : BOOL;
  bEnSeqLink        : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
arrSeqLinkData	ST_BA_SeqLinkData [▶ 126]	Daten und Befehle der einzelnen Sequenzregler
fE	REAL	Regelabweichung Direkt: fE = fX – fW Indirekt: fE = fW – fX
nActvSeqCtrl	UDINT	Nummer des aktiven Sequenzreglers.
bSync	BOOL	Impuls - Synchronisation
bEnSeqLink	BOOL	Freigabe *bEn* des FB_BA_SequenceLinkBase

6.1.2.1.1.2.10.1.2 ST_BA_SeqLinkData

Diese Struktur beinhaltet die Daten und Befehle der einzelnen Sequenzregler FB_BA_SeqCtrl [▶ 172].

```
TYPE ST_BA_SeqLinkData :
STRUCT
  fY                  : REAL;
  fYMin               : REAL;
  fYMax               : REAL;
  fW                  : REAL;
  fX                  : REAL;
  fE                  : REAL;
  nActvSeqCtrl        : UDINT;
  nMyNum              : UDINT;
  eActn               : E_BA_Action;
  bSeqCtrlOperable    : BOOL;
  bWatchdog           : BOOL;
  bSeqNumMultiple     : BOOL;
  bEn                 : BOOL;
  bIsActvSeqCtrl      : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fY	REAL	Stellgröße
fYMin	REAL	Minimale Stellgröße
fYMax	REAL	Maximale Stellgröße
fW	REAL	Sollwert
fX	REAL	Istwert
fE	REAL	Regelabweichung Direkt: fE = fX – fW Indirekt: fE = fW – fX
nActvSeqCtrl	UDINT	Nummer des aktiven Sequenzreglers.
nMyNum	UDINT	Meine Nummer in der Sequenz
eActn	E_BA_Action	Regler Wirksinn
bSeqCtrlOperable	BOOL	Sequenzregler ist betriebsbereit.
bWatchdog	BOOL	Watchdog wird bei jedem SPS-Zyklus in jedem Sequenzregler gesetzt und nach der Auswertung im Sequenz-Link zurückgesetzt.
bSeqNumMultiple	BOOL	Sequenz Nummer mehrfach in den Sequenzreglern vergeben.
bEn	BOOL	Freigabe der Sequenz Regler.
bIsActvSeqCtrl	BOOL	Aktiver Sequenzregler

6.1.2.1.2 GVLs

6.1.2.1.2.1 XBA_Globals

```
VAR_GLOBAL
  {region 'General'}
    Diag                            : FB_BA_Diagnosis;

    Top                             : FB_BA_TopView;
  {endregion}

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

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

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

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

Name	Typ	Beschreibung
Diag	FB_BA_Diagnosis	Liefert Information zum Projekt und bietet Informationen und Diagnosemöglichkeiten an: • Aktuelle Systemzeit • Eingestellte Zykluszeiten • Verwendete Bibliotheken und BACnet-Treiber • Anzahl der BACnet-Objekte • Status der Publisher und Subscriber • Ausgabe der Projektstruktur • Ausgabe der aktuell anstehenden und Historischen Ereignisse.
Top	FB_BA_TopView	Root-Objekt oder Top-Objekt der Projektstruktur. Sammelt Information aller Kinder-Objekte.
nIncObjInitial	UINT	Zeigt an, dass sich Eigenschaften eines Objektes [▶ 33] geändert haben, welche sich im Regelfall nur selten ändern.
nIncObjActivePriority	UINT	Zeigt an, dass sich die aktive Priorität eines Objektes [▶ 33] geändert hat.
nIncObjStatus	UINT	Zeigt an, dass sich der Zustand eines Objektes [▶ 33] geändert hat.
nIncEvent	UINT	Zeigt die Änderung eines Events [▶ 33] an.
nIncEventConfig	UINT	Zeigt an, dass sich die Event-Konfiguration [▶ 33] eines Objektes [▶ 33] geändert an.
nInstId_Auto	UDINT	Zeigt an, dass eine gültige Instanz ID automatisch erzeugt werden muss.
guidUndefined	GUID	Undefinierte System-ID
nNoActivePrio	UDINT	Wert der Konstante zeigt an, dass keine Priorität aktiv ist.
sPlaceholderSign_Open	STRING(1)	Anfangszeichen für einen Platzhalter.
sPlaceholderSign_Close	STRING(1)	Endzeichen für einen Platzhalter.
sPlaceholderSign_Delimiter	STRING(1)	Trennzeichen für Platzhalter-Attribute.
sPlaceholderSign_DenyConcat	STRING(1)	Zeichen für nicht zu verkettende Platzhalter.
sPlaceholder_Empty	STRING(2)	Leerer Platzhalter.
stUndefinedIdentifier	ST_BA_ObjectIdentifier [▶ 120]	Wert, der eine undefinierte Objekt [▶ 33]-Referenz darstellt.
aAckFlags_None	T_BA_EventTransitions [▶ 117]	Wert, der keine aktiven AcknowledgeRequired Flags darstellt.
aIndFilter_None	T_BA_EventConditionFlags [▶ 117]	Wert, nach dem nach keinen Event-Bedingungen [▶ 104] gefiltert wird.
aIndFilter_EvtAll	T_BA_EventConditionFlags [▶ 117]	Wert, zum Filtern von allen Event-Bedingungen [▶ 104].
aIndFilter_Evt	T_BA_EventConditionFlags [▶ 117]	Wert, zum Filtern von ausgewählten Event-Typen (Gruppiert nach Event-Typen).

6.1.2.1.2.2 XBA_Param

```
VAR_GLOBAL CONSTANT
  {region 'Supplement-Management'}
      bEnableBACnet                       : BOOL := TRUE;
```

```
      nEnabledSupplements                : INT(0..2) := 1;
  {endregion}
  {region 'Communication-Management'}
    {IF defined (BaDebug)}
      nCom_BACnetRM_IOCount              : DINT := 100;
    {ELSE}
      nCom_BACnetRM_IOCount              : DINT := 0;
    {END_IF}
  {endregion}
  {region 'Site service'}
      bSiteServer_Enable                 : BOOL := TRUE;
      nSiteServer_BufferSize             : UINT := 8192;
      nSiteServer_SessionTimeout         : TIME := T#15S;
      nSiteClient_BufferSize             : UINT := 1024;
      tSiteClient_ReadTimeout            : TIME := T#5S;
  {endregion}
  {region 'Project settings'}
    {region 'General'}
      {IF defined (BaDebug)}
        eLanguage                        : E_BA_Language := E_BA_Language.eGerman;
      {ELSE}
        eLanguage                        : E_BA_Language := E_BA_Language.eEnglish;
      {END_IF}

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

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

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

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

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

    nEventHistory_EntryCount             : INT := 2048;

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

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

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

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

```
      aDefReadAccess                        : ARRAY [E_BA_Parameter.First .. E_BA_Parameter.Last] OF E
_BA_Role :=  [E_BA_Parameter.Count(0)];
      aDefWriteAccess                       : ARRAY [E_BA_Parameter.First .. E_BA_Parameter.Last] OF E
_BA_Role := [E_BA_Parameter.Count(0)];
    {endregion}
    {region 'Multistate'}
      nMultistate_StateCount                : DINT := 20;
    {endregion}

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

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

      {region 'Plant Control'}
        nPlantCtrl_OpModeCount              : DINT := 16;
        nPlantCtrl_AggregateCount           : DINT := 16;
      {endregion}
      {region 'Sequence Link'}
      nSeqLink_RefCount                     : DINT := 16;
      {endregion}
      {region 'Sequence'}
      nMaxSeqCtrl                           : USINT := 8
      {endregion}
      {region 'Collector'}
        nCollect_RefCount                   : DINT := 16;
      {endregion}

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

```
    nGroupVal_RefCount                    : DINT := 5;
  {endregion}
END_VAR
```

Name	Typ	Beschreibung
bSiteServer_Enable	BOOL	De- / aktiviert den *Site Server* auf der Automationsstation.
nSiteServer_BufferSize	UINT	Definiert die Größe des Kommunikations-Puffers des Site Servers.
nSiteServer_SessionTimeout	TIME	Definiert die maximale Dauer [s] einer Session, bis diese nach Inaktivität vom Server beendet wird.
nSiteClient_BufferSize	UINT	Definiert die Größe des Kommunikations-Puffers vom Site Client.
tSiteClient_ReadTimeout	TIME	Definiert die maximale Dauer [s] von Lese-Anfragen des Site Clients bis diese mit einem Timeout abgebrochen werden.
eLanguage	E_BA_Language	Sprache, die innerhalb der PLC verwendet wird, um zum Beispiel Aktuelle Werte zu formatieren.
bUtf8AutoConvert	BOOL	Wendet automatisch die UTF-8 Kodierung bei Strings an.
nDPADLevels	UINT	Legt die maximale Anzahl an Ebenen im DPAD [▶ 43] fest.
nDPAD_DefIndexDigits	UINT	Legt die Anzahl an Ziffern zur Indizierung von Ebenen im DPAD [▶ 43] fest.
sDPAD_ObjectName_DefSeparator	T_BA_ShortString [▶ 122]	Legt das Trennzeichen zum Trennen von Objektnamen fest.
eDPAD_ObjectName_ManOvrConcatMode	E_BA_ConcatDPADMode [▶ 102]	Vorgabe zur Verbindung von Strings, wenn ein Objektname manuell überschrieben wurde.
sDPAD_Description_DefSeparator	T_BA_ShortString [▶ 122]	Legt das Trennzeichen zum Trennen von Beschreibungen fest.
eDPAD_Description_ManOvrConcatMode	E_BA_ConcatDPADMode [▶ 102]	Vorgabe zur Verbindung von Strings, wenn eine Description manuell überschrieben wurde.
bDPAD_Description_ExplicitIndex	BOOL	Bedingungen unter denen ein Index im Parameter *Description* angezeigt werden soll. • FALSE zeigt den Index automatisch an, wenn > 1. • TRUE zeigt den Index nur an, wenn es explizit definiert ist (Platzhalter).
eEvtMgmt_AckMode	E_BA_AcknowledgeMode [▶ 103]	Verhalten beim Quittieren von Ereignissen [▶ 33].
sEvtMgmt_AckMsgInternal	STRING	Text zum Anzeigen von integrierten Quittierungsfunktionen (für interne Funktionen).
sEvtMgmt_AckMsgRemote	STRING	Text zum Anzeigen von Quittierungsfunktionen durch externen Zugriff.
sEvtMgmt_AckMsgPLC	STRING	Text zum Anzeigen von Quittierungsfunktionen durch SPS-Logik.
aAckFlags_Simple	T_BA_EventTransitions [▶ 117]	Definiert die zu bestätigenden Übergänge [▶ 33] für den Alarm-Modus „Simpel“.
aAckFlags_Standard	T_BA_EventTransitions [▶ 117]	Definiert die zu bestätigenden Übergänge [▶ 33] für den Alarm-Modus „Standard“.
aAckFlags_Extended	T_BA_EventTransitions [▶ 117]	Definiert die zu bestätigenden Übergänge [▶ 33] für den Alarm-Modus „Erweitert“.
aEventEn_Simple	T_BA_EventTransitions [▶ 117]	Definiert zu berücksichtigende Übergangszustände [▶ 33] für den Alarm-Modus „Simpel“.
aEventEn_Standard	T_BA_EventTransitions [▶ 117]	Definiert zu berücksichtigende Übergangszustände [▶ 33] für den Alarm-Modus „Standard“.
aEventEn_Extended	T_BA_EventTransitions [▶ 117]	Definiert zu berücksichtigende Übergangszustände [▶ 33] für den Alarm-Modus „Erweitert“.

Name	Typ	Beschreibung
nEventHistory_EntryCount	INT	Maximale Anzahl an Einträgen in der Ereignis-Historie.
nEventTransitionText_Length	DINT	Maximale Anzahl an Buchstaben im Event-Transition-Parameter.
aEventTransitionText	T_BA_EventTransitionText [▶ 117]	Standardwert für den Event Transition-Text.
nEventList_EntryCount	DINT	Maximal angezeigte Anzahl von Einträgen in Ereignislisten.
nInstID_AutoGenerateOffset	DINT	Initialwert zur Autogeneration von Instanz-IDs.
nStateText_Length	DINT	Maximale Anzahl an Buchstaben im Status-Text-Parameter.
nTag_Length	DINT	Maximale Anzahl an Buchstaben im Tag-Parameter.
aDefReadAccess	E_BA_Parameter [▶ 110]	Möglichkeit zur Anpassung von Standard-Zugriffsrechten für Lesezugriffe auf Parameter.
aDefWriteAccess	E_BA_Parameter [▶ 110]	Möglichkeit zur Anpassung von Standard-Zugriffsrechten für Schreibzugriffe auf Parameter.
nMultistate_StateCount	DINT	Legt die Anzahl an Zuständen für Multistate-Werte fest.
fInput_DefResolution	REAL	Standardwert für den Parameter *Resolution* von Eingängen.
fInput_DefScaleOffset	REAL	Standardwert für den Parameter *Offset* von Eingängen.
fOutput_DefResolution	REAL	Standardwert für den Parameter *Resolution* von Ausgängen.
fOutput_DefScaleOffset	REAL	Standardwert für den Parameter *Offset* von Ausgängen.
eInput_DefSensor	E_BA_MeasuringElement	Auswahl des Sensors mit dem speziellen Eingangstypen FB_BA_AI_IOEx [▶ 198].
fDefLimitDeadband	REAL	Standardwert für den Parameter Tot-Zone Grenzwert.
nDefTimeDelay_ToAbnormal	UDINT	Standardwert für den Parameter Zeitverzögerung von Übergängen in nicht-normale Zustände.
nDefTimeDelayAO_ToAbnormal	UDINT	Standardwert für den Parameter *Time Delay* bei Übergängen in nicht-normale Zustände analoger Ausgänge.
nDefTimeDelayBO_ToAbnormal	UDINT	Standardwert für den Parameter *Time Delay* bei Übergängen in nicht-normale Zustände binärer Ausgänge.
nDefTimeDelayMO_ToAbnormal	UDINT	Standardwert für den Parameter *Time Delay* bei Übergängen in nicht-normale Zustände mehrstufiger Ausgänge.
fDefCOVIncrement	REAL	Standardwert für den Parameter COV-Inkrement.
nPlantCtrl_OpModeCount	DINT	Legt die maximale Anzahl von Anlagenbetriebsarten von FB_BA_PlantCtrl-Bausteinen fest.
nPlantCtrl_AggregateCount	DINT	Legt die maximale Anzahl an Aggregate [▶ 43]-Referenzen von FB_BA_PlantCtrl-Bausteinen fest.
nSeqLink_RefCount	DINT	Maximale Anzahl an Regler-Referenzen in einem Sequenz-Linker.
nMaxSeqCtrl	USINT	Der globale Parameter gibt die Anzahl der Sequenzregler an. Er limitiert innerhalb der Struktur ST_BA_SeqLink [▶ 125] die Daten- und Befehlsstruktur *arrSeqLinkData*. Der Wert darf nicht kleiner 1 sein.
nCollect_RefCount	DINT	Legt die maximale Anzahl an Referenzen von FB_BA_Collector-Bausteinen fest.

Name	Typ	Beschreibung
eLoop_DefOpMode	E_BA_PIDMode	Standardwert für den Parameter *Betriebsmodus*.
nTrend_BufferSize	UDINT	Anzahl der Einträge in einem Trend-Buffer.
stTrend_DefStartTime	ST_BA_DateTime	Standardwert für den Parameter Startzeit.
stTrend_DefStopTime	ST_BA_DateTime	Standardwert für den Parameter Stoppzeit.
bTrend_DefStopOnFull	BOOL	FALSE, Ringspeicher; TRUE, Fester Speicher, der nichts mehr speichert, wenn der Buffer voll ist.
nTrend_DefLogInterval	UDINT	Standardwert für den Parameter Aufzeichnungsintervall.
nTrend_DefNotificationThreshold	UDINT	Standardwert für den Parameter Benachrichtigungsschwelle.
nCal_EntryCount	DINT	Anzahl der Einträge in einem Kalenderobjekt.
nSched_EntryCount	DINT	Anzahl der Einträge an einem Wochentag ('T_BA_SchedWeek' [▶ 124]) oder einer Exception ('ST_BA_SchedCalendar' [▶ 123] / 'ST_BA_SchedException' [▶ 123]).
nSched_CalendarCount	DINT	Anzahl der Kalenderreferenzen ('T_BA_SchedCalendar [▶ 124]').
nSched_ExceptionCount	DINT	Anzahl der Exceptions ('T_BA_SchedExceptionList' [▶ 124]).
nSim_AISen_DefDampConstant	UDINT	Standardwert für den Parameter *Dämpfungskonstante*.
tSub_ReadTolerance	TIME	Standardwert für den Parameter *Lese-Toleranz.*
bSub_ClearOnReadError	BOOL	Standardwert für den Parameter *Zurücksetzen bei Lesefehlern*.
tSub_DefReadInterval	TIME	Standardwert für den Parameter *Lese-Intervall*.
nGroupCmd_RefCount	DINT	Anzahl der Referenzierungen.
nGroupDsp_RefCount	DINT	Anzahl der Referenzierungen.
nGroupVal_RefCount	DINT	Anzahl der Referenzierungen.

6.1.2.1.2.3 BACnet

6.1.2.1.2.3.1 XBA_BACnetParam

In der Liste befinden sich global gültige Parameter mit denen Grundeinstellungen bezüglich des BACnet in TF8040 vorzunehmen sind:

```
VAR_GLOBAL CONSTANT
  {region 'Objects'}
    {region 'Local'}
      {region 'Event Config'}
        sEventMessageTextFormat           : STRING := '{Descr} - {EvtTrans}';
      {endregion}
    {endregion}
    {region 'Remote'}
      {region 'Analog Output'}
        fRM_AO_WriteIncrement             : REAL := 0.0;
      {endregion}
      {region 'Structured View'}
        eView_SubordinateAnnotationMode   : E_BACnet_AnnotationTitle := E_BACnet_AnnotationTitle.eSy
mbolName;
      {endregion}
    {endregion}
```

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

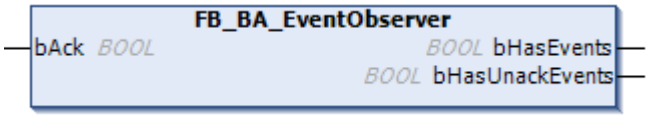
Name	Typ	Beschreibung
sEventMessageTextFormat	STRING	Ein Meldetext für die TwinCAT HMI BA als auch für einen BACnet-Client setzt sich aus verschiedenen Strings zusammen. Der hintere Teil des Strings ist der *EventTrasitionText*. Die Texte für die verschiedensten Eventtransitionen sind in einer Liste Namens TxtEvent_EN bzw. TxtEvent_DE. Diese Listen werden beim Erzeugen einer TF8040 Sample PLC mit angelegt. Der vordere Teil der Meldung kann aus verschiedenen Texten bestehen. Zum Beispiel aus der *ObjectDescription*. Mit dem String sEventMessageTextFormat kann das Format bzw. die Zusammensetzung des EventMessageText bestimmt werden. Er setzt sich zusammen aus Texten und Platzhaltern. Platzhalter und freie Texte wie beispielsweise ein Trennzeichen können hintereinander angeordnet werden. **Platzhalter**: {ObjName} = Objectname {Descr} = Description {InstID} = InstanceID {EvtTrans} = EventTaransition Im folgenden Beispiel wird die *Objectdescription* dem *EventTransitionText* vorangestellt. Mit dem Bindestrich wird zwischen den Beiden Platzhaltern ein Trennzeichen platziert. sEventMessageTextFormat :='{Descr} - {EvtTrans};
fRM_AO_WriteIncrement	REAL	Schwellwert, der überschritten sein muss, um eine Wertänderung am Analogausgang auszugeben.
eView_SubordinateAnnotationMode	E_BACnet_AnnotationTitle	Property welches zur Darstellung, eines Kind-Objektes in der SubordinateList des dazugehörigen Parent-View-Objektes verwendet wird.
aPriority	ARRAY[E_BA_Priority [▶ 112].First .. E_BA_Priority.Last] OF E_BACnet_Priority	Definiert die Zuordnung der BACnet-Prioritäten aus der Tc3_BACnetRev14 Bibliothek im Prioritäten-Array.
aNodeType	ARRAY[E_BA_NodeType [▶ 113].First .. E_BA_NodeType.Last] OF E_BACnet_NodeType	Übersetzungstabelle für das Property *NodeType* zwischen der Tc2_XBA Bibliothek und der Tc3_BACnetRev14 Bibliothek.
aNotifyType	ARRAY[E_BA_EventType [▶ 105].First .. E_BA_EventType.Last] OF E_BACnet_NotifyType	Übersetzungstabelle für das Property *NotifyType* zwischen der Tc2_XBA Bibliothek und der Tc3_BACnetRev14.
aEventState	ARRAY[E_BA_EventState.First .. E_BA_EventState.Last] OF E_BACnet_EventState	Übersetzungstabelle für das Property *EventState* zwischen der Tc2_XBA Bibliothek und der Tc3_BACnetRev14.

6.1.2.1.3 POUs

6.1.2.1.3.1 FunctionBlocks

6.1.2.1.3.1.1 Events

6.1.2.1.3.1.1.1 FB_BA_EventObserver



Der Funktionsbaustein dient der Auswertung von Alarmen / Ereignissen der Projektstruktur und deren Quittierung.

Der Zugriff auf die Projektstruktur mit einer Parent / Child Beziehung geschieht über eine Zuweisung auf das Property *Parent*.

Syntax

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

Eingänge

Name	Typ	Beschreibung
bAck	BOOL	An den Eingang wird das Signal zum Quittieren von Alarmen / Ereignissen angelegt.

Ausgänge

Name	Typ	Beschreibung
bHasEvents	BOOL	Anzeige das Ereignisse anstehen.
bHasUnackEvents	BOOL	Anzeige das unquittierte Ereignisse anstehen.

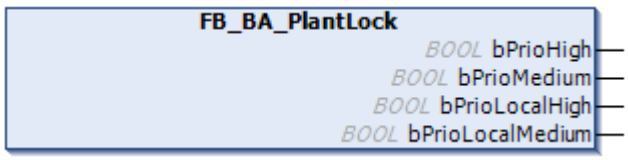
Eigenschaften

Name	Typ	Zugriff	Beschreibung
Parent	I_BA_View	Get, Set	Über eine Zuweisung auf das Property wird der Zugriff auf die Projektstruktur (Grundgerüst) realisiert. Findet keine Zuweisung statt, so wird auf die Basis des Grundgerüst zugegriffen. Dieses bedeutet, dass sämtliche Ereignisse / Alarme ausgewertet werden und dementsprechend deren Quittierung ausgelöst werden kann. Zusätzlich wird eine Warnung in die Fehlerliste der TC3 Entwicklungsumgebung ausgegeben.
Valid	BOOL	Get	Zeigt an, dass eine gültige Zuweisung an der Schnittstelle *iParent* (Parent) anliegt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.2.1.3.1.1.2 FB_BA_PlantLock



Die Events der Objekte signalisieren oftmals eine relevante Störung, diese erfordert gezielte Schalthandlungen an Aggregaten oder Anlagen vorzunehmen. In großen Anlagen sind es oft sehr viele Ereignisse, welche dann häufig zu einer Sammelmeldung zusammengefasst werden müssen.
Um den Programmieraufwand zur Erzeugung von Sammelmeldungen so gering wie möglich zu halten, kann an jedem eventfähigen Objekt eine *LockPriorität* bestimmt werden.
Alle vier Lockprioritäten werden in jeder Ebene der Projektstruktur [▶ 39] gesammelt und von der tiefsten Ebene der Projektstruktur bis zum Top-Level hochgereicht.
Bei Bedarf werden sie durch die Instanz eines Funktionsbausteins *FB_BA_PlantLock* angezeigt, und für die Steuerung von Anlagen- und Aggregaten mit anderen Funktionsbausteinen verschaltet.

Vier Lockprioritäten dienen der Differenzierung der Objekt-Events, um unterschiedliche Reaktionen auslösen zu können.

- **Lokal Medium**
  Gibt ein lokales Abschalten mittlerer Priorität frei.
- **Lokal Hoch**
  Gibt ein lokales Abschalten höherer Priorität frei.
- **Medium**
  Gibt ein übergeordnetes Abschalten mittlerer Priorität frei.
- **Hoch**
  Gibt ein übergeordnetes Abschalten höherer Priorität frei.

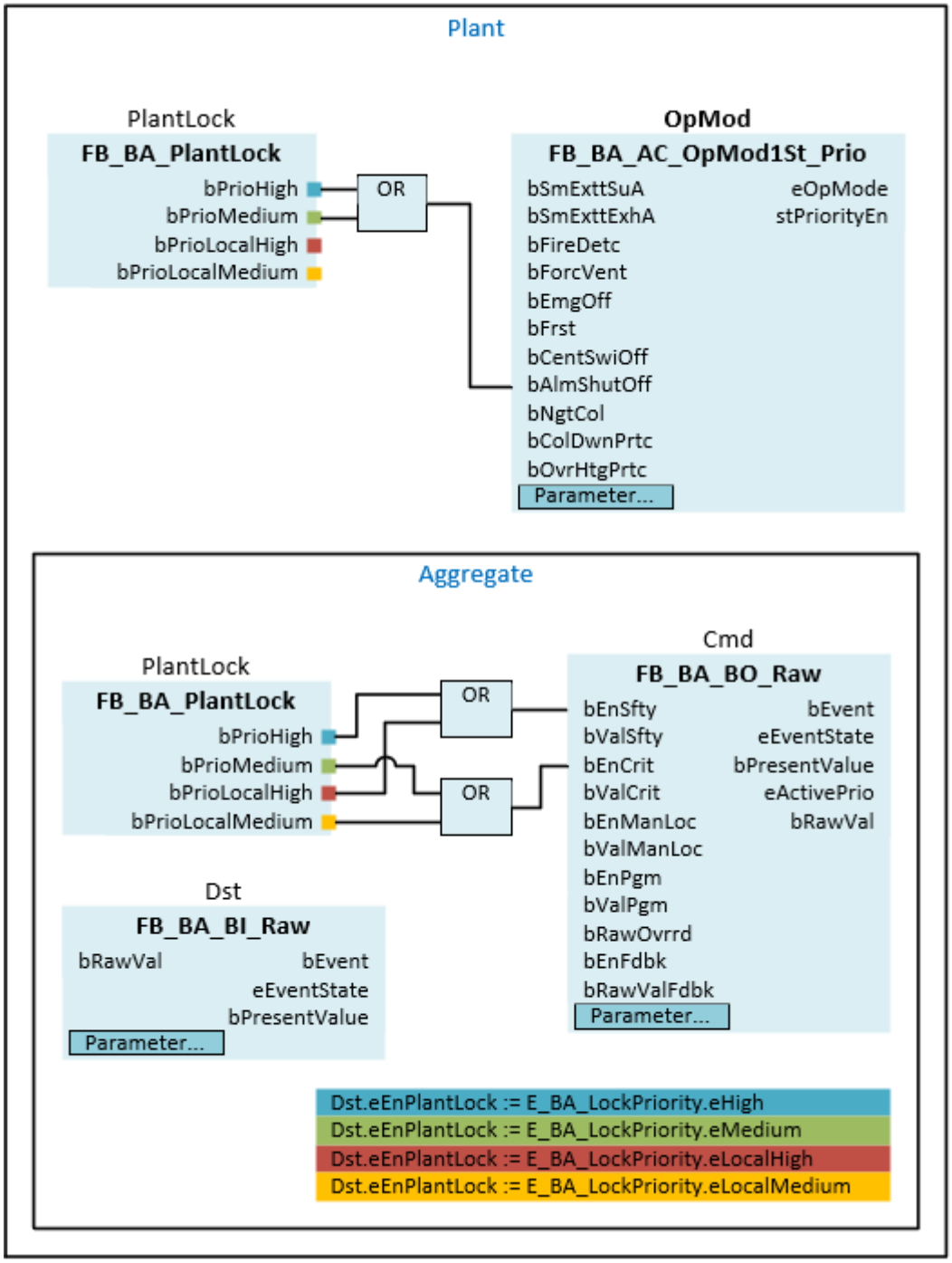
Die mittlere Priorität wird bei technisch relevanten Störungen verwendet, welche die Sicherheit von Anlagen betreffen. Die hohe Priorität kann für Störungen verwendet werden, bei denen die Sicherheit von Personen gefährdet ist.

Die lokalen Störungen sollten verwendet werden, um z. B. Aggregate abzuschalten. Die nicht lokalen sollten verwendet werden, um Anlagen abzuschalten.

Das untenstehende Bild soll diesen Zusammenhang erläutern:

Die Auswirkungen verschiedener Lockprioritäten des Störmeldeeingangs DST sind wie folgt:

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



Illustration

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

Ausgänge

Name	Typ	Beschreibung
bPrioLocalHigh	BOOL	Der Ausgang signalisiert ein globales Event mit einer hohen Priorität.
bPrioMedium	BOOL	Der Ausgang signalisiert ein globales Event mit einer mittleren Priorität.
bPrioLocalHigh	BOOL	Der Ausgang signalisiert ein lokales Event mit einer hohen Priorität.
bPrioLocalMedium	BOOL	Der Ausgang signalisiert ein lokales Event mit einer mittleren Priorität.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

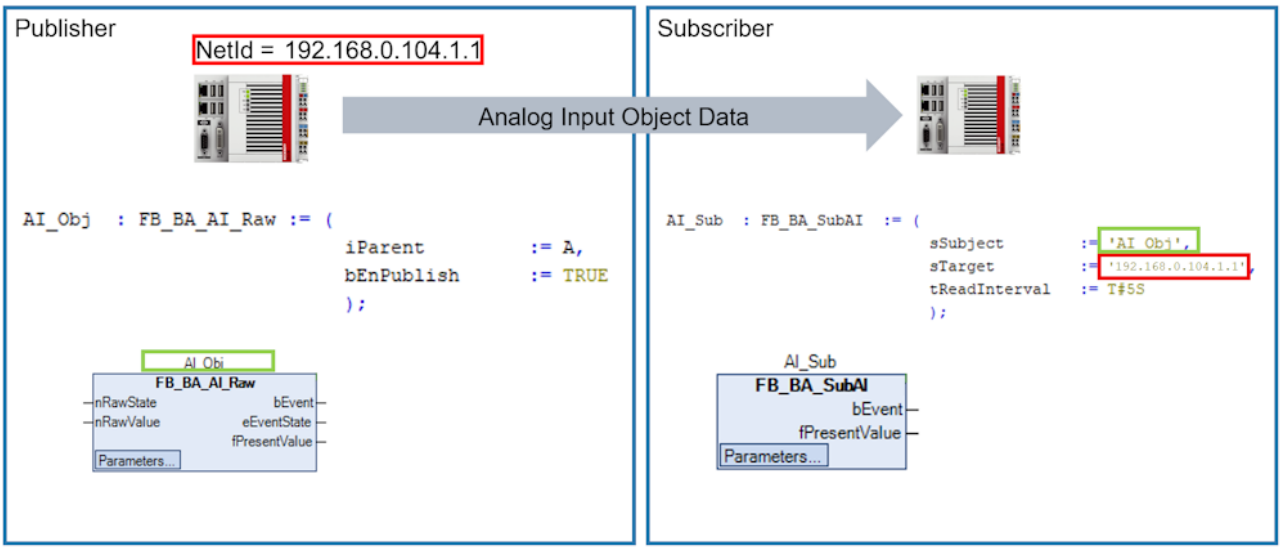
6.1.2.1.3.1.2 Reference

6.1.2.1.3.1.2.1 Publish and Subscribe

Publisher und Subscriber bieten eine Möglichkeit zum einfachen Austausch von Informationen.

- Publisher stellen Informationen zur Verfügung.
- Subscriber abonnieren Informationen eines referenzierten Publishers um diese dann in einem bestimmten Programmabschnitt (z. B. Template) verfügbar zu machen.
  Unabhängig davon, wo Publisher und Subscriber implementiert werden (z. B. auf unterschiedlichen SPS), werden Informationen auf einheitliche Art und Weise ausgetauscht.

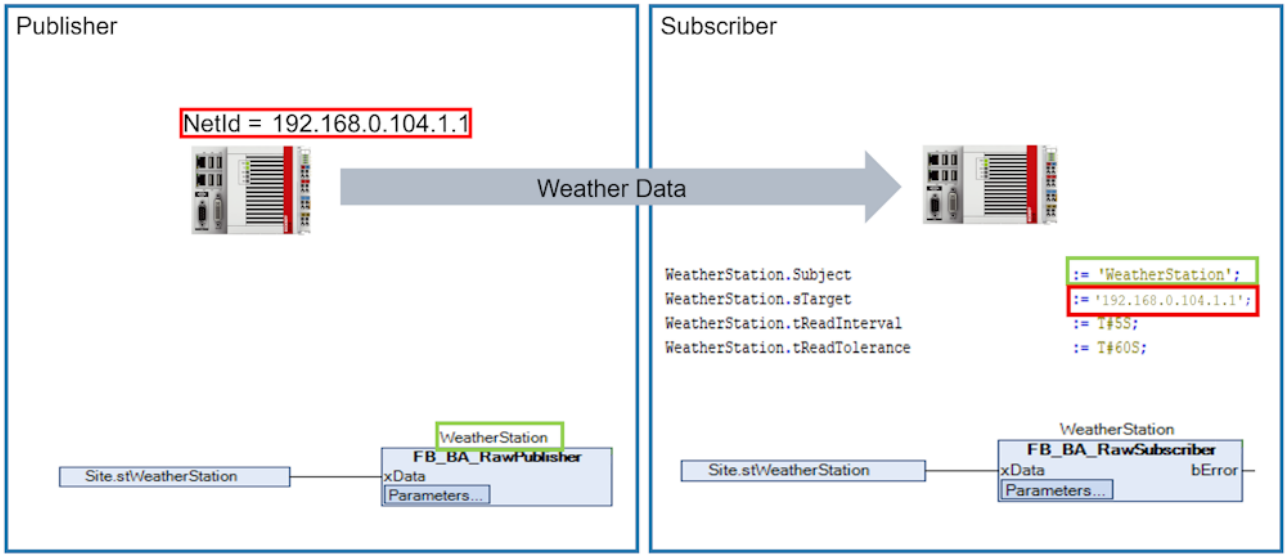
Beispiel 1



Anwendung von Publisher und Subscriber auf konkrete Funktionen und Objekte.

Jeder Funktionsbaustein kann als Publisher eingesetzt sein. Als Subscriber dienen Funktionsbausteine, die den gewünschten Datentypen abonnieren können. Hierzu muss der eindeutige Objektname des Publishers und seine AMS-NetID am Subscriber angegeben sein.

Beispiel 2



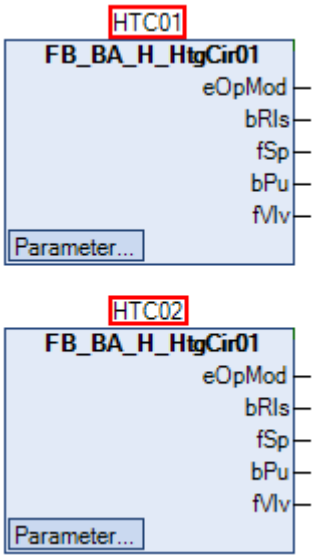
Anwendung von Publisher und Subscriber für beliebige Datentypen.

Über den FB_BA_RawPublisher [▶ 143] können Informationen unterschiedlicher Größe dem FB_BA_RawSubscriber [▶ 162] zur Verfügung gestellt werden. Wie im vorigen Beispiel müssen der Objektname und die AMS-NetID am Subscriber angegeben sein. Diese Konfiguration findet, wie im Beispiel angedeutet, einmalig im FB_init des FB_BA_RawSubscriber [▶ 162] statt.

Wenn von einem Objekt mehrere Instanzen mit dem gleichen Symbolnamen existieren, dann ist es notwendig den kompletten Symbolnamen des Objektes am Parameter *sSubject* anzugeben.

Beispiel 3

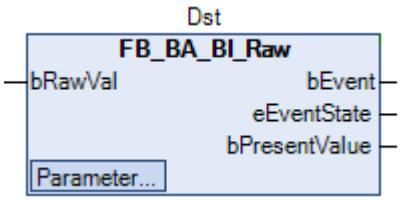
Im Beispiel sollen die Störungen zweier Heizkreispumpen separat veröffentlicht und abonniert werden.



Innerhalb der Heizkreise haben die Pumpen *Pu* und die veröffentlichen Binärobjekte der Störung *Dst* den gleichen Namen.

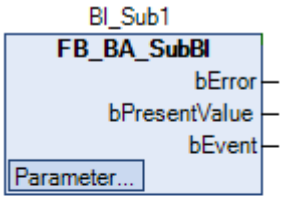
Daher erfolgt das Publishen des Parameters aus beiden Templates analog.

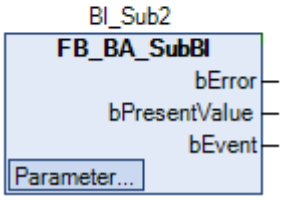
```
Dst                    : FB_BA_BI_Raw  := (
                                          bEnPublish     := TRUE
                                         );
```



Das Abonnieren (subscribe) erfolgt separat, es ist lediglich darauf zu achten, dass der Symbolpfad für das zu abonnierende Ereignis korrekt angegeben ist.

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





6.1.2.1.3.1.2.1.1 Publisher

Bausteine zum lokalen oder steuerungsübergreifenden Veröffentlichen von Informationen.

6.1.2.1.3.1.2.1.1.1 FB_BA_RawPublisher



Baustein zum Bereitstellen von beliebigen Informationen.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Publisher

Syntax

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

Eingänge

Name	Typ	Beschreibung
bEnPublish	BOOL	Variable zur Freigabe der von Variablen oder Datensätzen per ADS.
xData	ANY	Zu veröffentlichende Daten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.2.1.2 Subscriber

Bausteine zum Abonnieren von lokalen oder steuerungsübergreifenden Informationen.

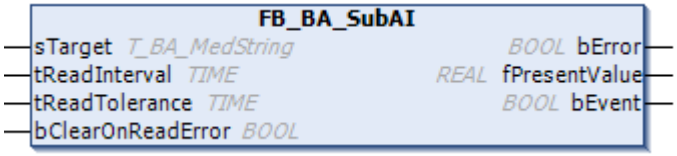
6.1.2.1.3.1.2.1.2.1 Objects

Dieser Ordner bietet Objekte an, die es ermöglichen entweder innerhalb der Steuerung, oder steuerungsübergreifend veröffentlichte (publish) Objektinformationen zu abonnieren (subscribe).

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

Bausteine zur Subskription von analogen Objekten.

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines analogen Eingangsobjektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtA

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

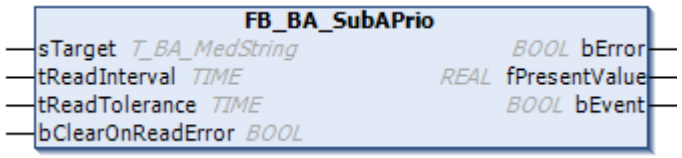
Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.
bEvent	BOOL	Der Ausgang ist TRUE, wenn ein Ereignis oder ein Fehler aufgetreten ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines analogen Eingangsobjektes mit Prioritäten-Array.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtA

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

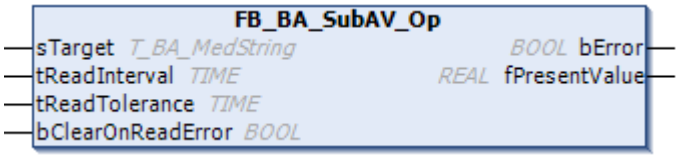
Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.
bEvent	BOOL	Der Ausgang ist TRUE, wenn ein Ereignis oder ein Fehler aufgetreten ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines analogen Value-Objektes [▶ 207] zur Anzeige eines analogen Wertes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubA

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.

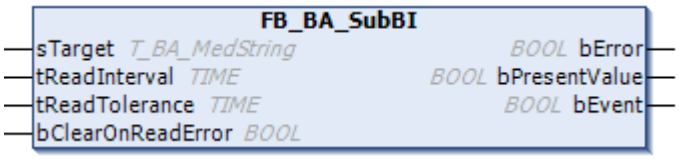
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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

Bausteine zur Subskription von binären Objekten.

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines binären Eingangsobjektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtB

Syntax

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

```
  bClearOnReadError         : BOOL;
END_VAR

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

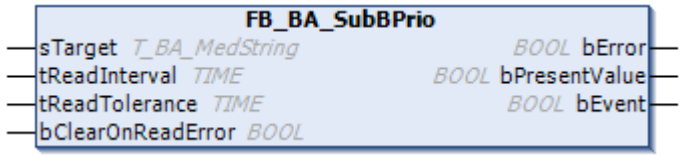
Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.
bEvent	BOOL	Der Ausgang ist TRUE, wenn ein Ereignis oder ein Fehler aufgetreten ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines binären Eingangsobjektes mit Prioritäten-Array.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtB

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

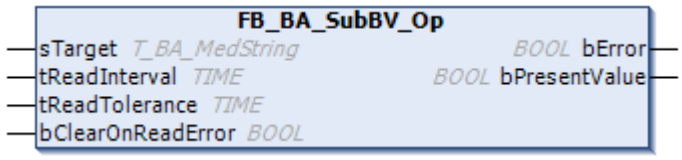
Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.
bEvent	BOOL	Der Ausgang ist TRUE, wenn ein Ereignis oder ein Fehler aufgetreten ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines binären Value-Objektes [▶ 221] zur Anzeige eines binären Wertes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubB

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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

Bausteine zur Subskription von komplexen Objekten.

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines Kalenderobjektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubB

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

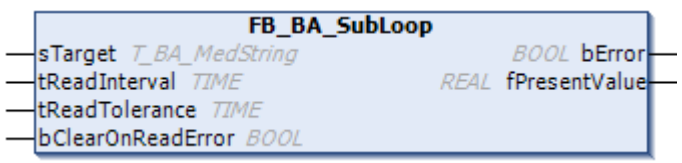
Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines Loop-Objektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubA

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines analogen Scheduler-Objektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

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

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.
fPredictedValue	REAL	Wert, der nach dem nächsten Schalten angenommen wird.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
Unit	E_BA_Unit	Get	Einheit.

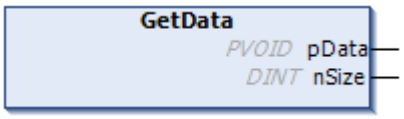
Methoden

Name	Beschreibung
GetData [▶ 154]	Enthält die abonnierten Daten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Die Methode speichert abonnierte Daten eines Publishers.

Syntax

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

Ausgänge

Name	Typ	Beschreibung
pData	pData	Pointer auf die abonnierten Daten.
nSize	DINT	Größe der abonnierten Daten.

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines binären Scheduler-Objektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.
bPredictedValue	BOOL	Wert, der nach dem nächsten Schalten angenommen wird.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
ActiveText	STRING	Get	Aktiv-Text
InactiveText	STRING	Get	Inaktiv-Text

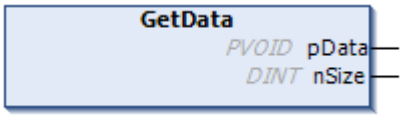
Methoden

Name	Beschreibung
GetData [▶ 155]	Enthält die abonnierten Daten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Die Methode speichert abonnierte Daten eines Publishers.

Syntax

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

Ausgänge

Name	Typ	Beschreibung
pData	pData	Pointer auf die abonnierten Daten.
nSize	DINT	Größe der abonnierten Daten.

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines Multi-State-Scheduler-Objektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
nPresentValue	UDINT	Analoger Ausgangswert.
nPredictedValue	UDINT	Wert, der nach dem nächsten Schalten angenommen wird.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
StateCount	UDINT	Get	Anzahl der möglichen States.
StateText	T_BA_StateTextArray [▶ 123]	Get	Ausgabe des State-Textes.

Methoden

Name	Beschreibung
GetData [▶ 157]	Enthält die abonnierten Daten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Die Methode speichert abonnierte Daten eines Publishers.

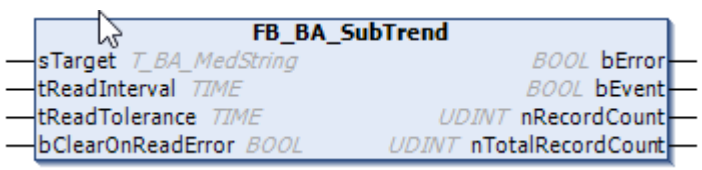
Syntax

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

Ausgänge

Name	Typ	Beschreibung
pData	pData	Pointer auf die abonnierten Daten.
nSize	DINT	Größe der abonnierten Daten.

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines Trend-Objektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
bEvent	BOOL	Die Variable signalisiert ein unnormales Ereignis eines Objektes. Die Art des Ereignisses, z. B. ein Alarm, eine Wartungsmeldung etc., wird innerhalb der dem Objekt zugehörigen Ereignisklasse beschrieben.
nRecordCount	UDINT	Anzahl der Aufzeichnungen.
nTotalRecordCount	UDINT	Absolute Anzahl der Aufzeichnungen.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
Enable	BOOL	Get, Set	Freigabe der Subscription.

Methoden

Name	Beschreibung
GetData [▶ 159]	Enthält die abonnierten Daten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Die Methode speichert abonnierte Daten eines Publishers.

Syntax

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

Ausgänge

Name	Typ	Beschreibung
pData	pData	Pointer auf die abonnierten Daten.
nSize	DINT	Größe der abonnierten Daten.

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

Bausteine zur Subskription von Multi-State-Objekten.

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines Multi-State- Eingangsobjektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubEvtM

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
nPresentValue	UDINT	Analoger Ausgangswert.
bEvent	BOOL	Der Ausgang ist TRUE, wenn ein Ereignis oder ein Fehler aufgetreten ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines Multi-State-Value-Objektes mit Prioritäten-Array.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubM

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

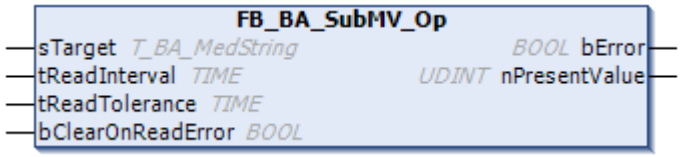
Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
nPresentValue	UDINT	Analoger Ausgangswert.
bEvent	BOOL	Der Ausgang ist TRUE, wenn ein Ereignis oder ein Fehler aufgetreten ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

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



Der Funktionsbaustein ermöglicht das Abonnieren (subscriben) eines Multi-State-Value-Objektes [▶ 244] zur Anzeige eines Multi-State-Wertes.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

FB_BA_SubM

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.
nPresentValue	UDINT	Analoger Ausgangswert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.2.1.2.2 FB_BA_RawSubscriber



Baustein zum Abonnieren von beliebigen Informationen.

Vererbungshierarchie

FB_BA_Base

FB_BA_Subscriber

Syntax

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

VAR_OUTPUT
  bError                : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sTarget	T_BA_MedString [▶ 122]	AMS NetId des Publishers.
tReadInterval	TIME	Leseintervall [s].
tReadTolerance	TIME	Wartezeit, bis ein anstehender Fehler ausgegeben wird [ms].
bClearOnReadError	BOOL	Bei TRUE werden die Daten gelöscht, wenn ein Fehler auftritt.
xData	ANY	Gelesene Daten.

Ausgänge

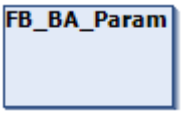
Name	Typ	Beschreibung
bError	BOOL	Zeigt den aktuellen Fehlerzustand der Subskription an. Details können im Fehlerfall der entsprechenden Fehlermeldung entnommen werden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3 Accessories

6.1.2.1.3.1.3.1 FB_BA_Param



Der Funktionsbaustein ermöglicht die Konfiguration der Properties, die in der globalen Variablenliste XBA_Param [▶ 128] enthalten sind.

> Bestehende Limitierungen, die für das Beschreiben von Parametern gelten, werden eingehalten (siehe auch Regionen [▶ 100]).
> So wird zum Beispiel das Beschreiben von fixen Parametern zur Laufzeit abgelehnt.
> Parameter zur Konfiguration von Compiler-Zeit-Werten können **nicht** zur Laufzeit verändert werden!

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3.2 EventSubscriber

Die Event Subscriber ermöglichen das generische Abonnieren von Ereignissen.

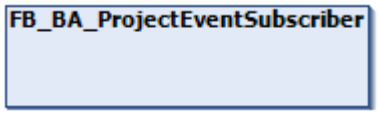
So kann an einer zentralen Stelle auf verschiedene Ereignisse (wie zum Beispiel im Projekt oder auf lokalen Objekten) reagiert werden, indem die jeweiligen Ereignisse innerhalb der bereitgestellten Methoden überschrieben werden.

Die möglichen Ereignisse werden als Methode angeboten und der Aufruf erfolgt je nach Bedarf.

Hinweise zur Anwendung der Event Subscriber

Es ist zu beachten, dass der Einsatz dieser Bausteine die Systemperformance beeinträchtigen können.
In Abhängigkeit der programmierten Logik kann es sein, dass z. B. die Methode *OnObjectStateChange* zu langsam durchlaufen wird. Dieses kann zu Zyklusüberschreitungen führen.

6.1.2.1.3.1.3.2.1 FB_BA_ProjectEventSubscriber



Der Subscriber-Baustein bietet die Möglichkeit auf Projektweite Ereignisse zu reagieren.

Innerhalb der Methode können vom Anwender Informationen über den Projektzustand beschrieben und ausgegeben werden.

Methode

Name	Beschreibung
OnProjectStateChanged [▶ 164]	Änderung des Projektzustandes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3.2.1.1 OnProjectStateChanged



Die Methode wird aufgerufen, wenn sich der Projektzustand geändert hat.

Syntax

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

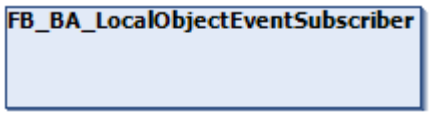
Eingänge

Name	Typ	Beschreibung
eState	E_BA_ProjectState [▶ 115]	Zustand, in den das Projekt gewechselt hat.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3.2.2 FB_BA_LocalObjectEventSubscriber



Der Subscriber-Baustein bietet die Möglichkeit auf Ereignisse in lokalen Objekten zu reagieren.

Methoden

Name	Beschreibung
OnBACnetObjectStateChanged [▶ 165]	Änderung des Zustandes eines BACnet-Objekts.
OnInitStateChange [▶ 166]	Änderung des Initialisierungszustandes eines Objekts.
OnInitInstanceId [▶ 166]	Änderung der Instanz-ID eines Objekts.
OnEventChange [▶ 167]	Änderung eines Ereignisses innerhalb eines Objekts.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3.2.2.1 OnBACnetObjectStateChanged



Die Methode wird aufgerufen, wenn sich der Initialisierungszustand eines BACnet-Objektes ändert.

Syntax

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

Eingänge

Name	Typ	Beschreibung
iObject	I_BA_Object	Schnittstelle zum Kontext-Objekt.
fbBACnetObject	POINTER TO FB_BACnet_BaseObject	Supplement spezifisches BACnet-Objekt des Kontext-Objektes.
tState	TCOM_STATE	Neuer Zustand des BACnet-Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3.2.2.2 OnInitStateChange



Die Methode wird aufgerufen, wenn sich der Initialisierungszustand eines Objektes ändert.

Syntax

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

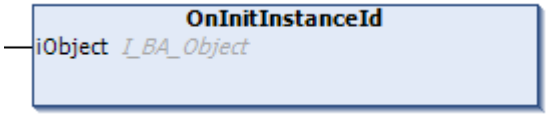
Eingänge

Name	Typ	Beschreibung
iObject	I_BA_Object	Schnittstelle zum Kontext-Objekt. [▶ 269]
eState	E_BA_InitState [▶ 116]	Neuer Zustand nach Initialisierung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3.2.2.3 OnInitInstanceId



Mit der Methode kann der Instanz-ID eines Objektes bestimmter Wert zugeordnet werden.

Andernfalls werden die Instanz-IDs fortlaufend generiert.

Jede Instanz ID darf nur einmal vergeben werden.

Syntax

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

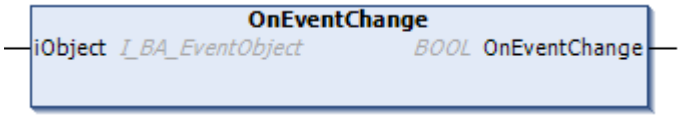
Eingänge

Name	Typ	Beschreibung
iObject	I_BA_Object	Schnittstelle zum Kontext-Objekt. [▶ 269]

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.3.2.2.4 OnEventChange



Die Methode wird aufgerufen, wenn ein Objekt ein Event generiert, bzw. sich ein generiertes Objekt ändert, oder gelöscht wird.

Syntax

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

Eingänge

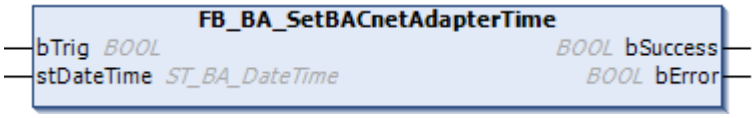
Name	Typ	Beschreibung
iObject	I_BA_Object	Schnittstelle zum Kontext-Objekt. [▶ 269]

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.1.4 System

6.1.2.1.3.1.4.1 FB_BA_SetBACnetAdapterTime



Mit dem Funktionsbaustein FB_BA_SetBACnetAdapterTime kann die lokale BACnet-Systemzeit eines TwinCAT-Systems gesetzt werden.

Syntax

```
FUNCTION_BLOCK FB_BA_SetBACnetAdapterTime
VAR_INPUT
  bTrig          : BOOL;
  stDateTime     : ST_BA_DateTime;
END_VAR
VAR_OUTPUT
  bSuccess       : BOOL;
  bError         : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bTrig	BOOL	Bei einer steigenden Flanke wird der Inhalt der Struktur stDateTime auf die lokale BACnet-Systemzeit des TwinCAT-Systems geschrieben.
stDateTime	ST_BA_DateTime	Zeitangabe, welche auf die lokale BACnet-Systemzeit geschrieben wird.

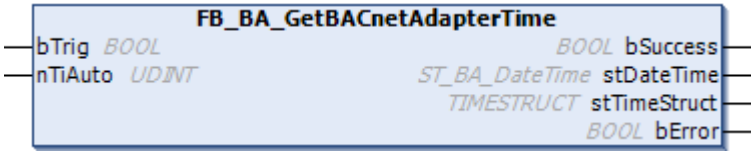
Ausgänge

Name	Typ	Beschreibung
bSuccess	BOOL	Die Zeitangabe *stDateTime* wurde erfolgreich auf das Zielsystem geschrieben. Die Variable ist für einen Zyklus TRUE.
bError	BOOL	Fehler beim Schreiben der BACnet-Systemzeit.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_XBA ab v5.6.1.0

6.1.2.1.3.1.4.2 FB_BA_GetBACnetAdapterTime



Mit dem Funktionsbaustein FB_BA_GetBACnetAdapterTime kann die lokale BACnet-Systemzeit eines TwinCAT-Systems ermittelt werden.

Intern wird anhand des Funktionsbaustein "RTC_EX2" (Extended Real Time Clock) eine interne Software-Uhr realisiert. Diese Software-Uhr wird mit der BACnet-Systemzeit initialisiert und durch die beiden Strukturen *stDateTime* und *stTimeStruct* ausgegeben.

Damit die aktuelle Zeit berechnet werden kann, sollte der Funktionsbaustein in jedem Zyklus der SPS einmal aufgerufen werden.

Syntax

```
FUNCTION_BLOCK FB_BA_GetBACnetAdapterTime
VAR_INPUT
     bTrig                      : BOOL;
     {attribute 'parameterUnit':= 'ms'}
     nTiAuto                    : UDINT;
END_VAR
VAR_OUTPUT
     bSuccess                   : BOOL;
     stDateTime                 : ST_BA_DateTime;
     stTimeStruct               : TIMESTRUCT;
     bError                     : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bTrig	BOOL	Bei einer steigenden Flanke an diesem Eingang wird die lokale BACnet-Systemzeit eines TwinCAT-Systems ermittelt.
nTiAuto	UDINT	Anhand dieser Zeitangabe wird die lokale BACnet-Systemzeit eines TwinCAT-Systems regelmäßig automatisch ermittelt. Die Zeitangabe muss größer gleich 500ms sein.

Ausgänge

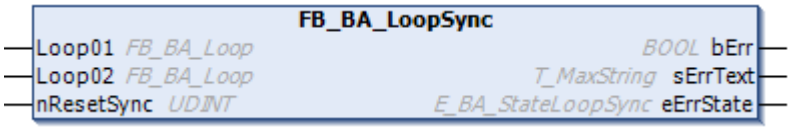
Name	Typ	Beschreibung
bSuccess	BOOL	Die Systemzeit wurde erfolgreich aus dem Zielsystem gelesen. Die Variable ist für einen Zyklus TRUE.
stDateTime	ST_BA_DateTime	Aktuelle BACnet-Systemzeit eines TwinCAT-Systems.
stTimeStruct	TIMESTRUCT	Aktuelle BACnet-Systemzeit eines TwinCAT-Systems.
bError	BOOL	Fehler beim Auslesen der BACnet-Systemzeit.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_XBA ab v5.6.1.0

6.1.2.1.3.1.5 Control

6.1.2.1.3.1.5.1 FB_BA_LoopSync



Der Funktionsbaustein FB_BA_LoopSync dient der Parameter-Synchronisation für zwei Regler vom Typ FB_BA_Loop [▶ 225]. Dieser Baustein kann beispielsweise zum Abgleich von Führungsreglern dienen, welche zur Erzeugung der Zuluft Sollwerte für eine raumlufttechnische Anlage eingesetzt werden.

Die folgenden Parameter des FB_BA_Loop [▶ 225] werden synchronisiert:

- eOpMode
- eActionRm
- fProportionalConstant
- fIntegralConstant
- fDerivativeConstant
- fMaxOutputRm
- fMinOutputRm
- nDampConstant
- fNeutralZone

Fehlererkennung

Die unten aufgeführten Fehlermeldungen werden durch den *FB_BA_LoopSync* erkannt.

Die Fehlermeldungen werden in der Entwicklungsumgebung von TwinCAT 3 in dem Fenster "Fehlerliste" ausgegeben. Dieses kann unter dem Menüpunkt Ansicht aktiviert werden.

Die Fehlertexte werden über das Property *ErrText* und den Ausgang *sErrText* ausgegeben.

Zusätzlich werden die Meldungen durch das Enum *eErrState* angezeigt.

Fehlermeldungen

Meldetext Deutsch	Meldetext Englisch	Erläuterung
'Synchronisation *eActionRm* fehlerhaft'	'Synchronization *eActionRm* faulty'	Möglicherweise kann die falsche Synchronisation durch den Eingang *eActionPgm* ausgelöst werden. Dieser Eingang überschreibt bei Belegung die zu überwachende VAR_INPUT CONSTANT Variable *eActionRm*.
'Synchronisation *nDampConstant* fehlerhaft'	'Synchronization *nDampConstant* faulty'	Es ist möglich, dass die falsche Synchronistation durch den Eingang *eActionPgm* ausgelöst wurde. Dieser Eingang überschreibt bei Belegung die zu überwachende VAR_INPUT CONSTANT Variable *eActionRm*.
'Synchronisation *fDerivativeConstant* fehlerhaft'	'Synchronization *fDerivativeConstant* faulty'	
'Synchronisation *fIntegralConstant* fehlerhaft'	'Synchronization *fIntegralConstant* faulty'	
'Synchronisation *fMaxOutputRm* fehlerhaft'	'Synchronization *fMaxOutputRm* faulty'	Möglicherweise kann die falsche Synchronisation durch den Eingang *fMaxOutputPgm* ausgelöst werden. Dieser Eingang überschreibt bei Belegung die zu überwachende VAR_INPUT CONSTANT Variable *fMaxOutputRm*.
'Synchronisation *fMinOutputRm* fehlerhaft'	'Synchronization *fMinOutputRm* faulty'	Möglicherweise kann die falsche Synchronisation durch den Eingang *fMinOutputPgm* ausgelöst werden. Dieser Eingang überschreibt bei Belegung die zu überwachende VAR_INPUT CONSTANT Variable *fMinOutputRm*.
'Synchronisation *fNeutralZone* fehlerhaft'	'Synchronization *fNeutralZone* faulty'	
'Synchronisation *eOpMode* fehlerhaft'	'Synchronization *eOpMode* faulty'	
'Synchronisation *fProportionalConstant* fehlerhaft'	'Synchronization *fProportionalConstant* faulty'	

Syntax

```
FUNCTION_BLOCK FB_BA_LoopSync
VAR_OUTPUT
  bErr          : BOOL;
  sErrText      : T_MaxString;
  eErrState     : E_BA_StateLoopSync;
END_VAR
VAR_IN_OUT
  Loop01        : FB_BA_Loop;
  Loop02        : FB_BA_Loop;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bErr	BOOL	Der Ausgang zeigt an, wenn ein Fehler bei der Synchronisation aufgetreten ist.
sErrText	T_MaxString	Die Variable zeigt den Status der Synchronisation in Textform [▶ 170] an.
eErrState	E_BA_StateLoopSync [▶ 102]	Die Enumeration zeigt den Status der Synchronisation an.

Ein- Ausgänge

Name	Typ	Beschreibung
Loop1	FB_BA_Loop [▶ 225]	Referenz auf den Regler Nr. 1 des Parameterabgleichs.
Loop2	FB_BA_Loop [▶ 225]	Referenz auf den Regler Nr. 2 des Parameterabgleichs.

Eigenschaften

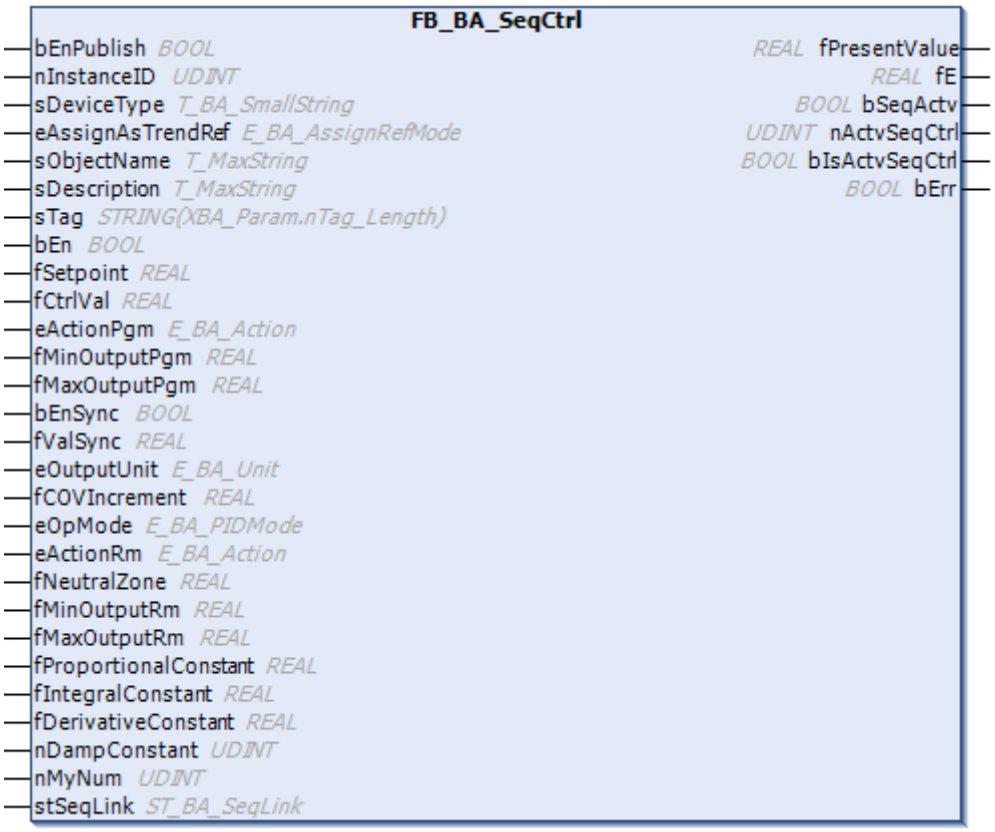
Name	Typ	Zugriff	Beschreibung
ErrText	T_MaxString	Get	Das Property *ErrText* zeigt die Fehlertexte von *sErrText* an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_XBA ab v5.6.1.0

6.1.2.1.3.1.6 Sequence

6.1.2.1.3.1.6.1 FB_BA_SeqCtrl



PID-Sequenzregler als Bestandteil einer Sequenz.

Mit Erhalt der Freigabe *bEn* = TRUE wird dem übergeordneten Kontrollbaustein FB_BA_SequenceLinkBase [▶ 175] mitgeteilt, dass der Sequenzregler betriebsbereit für die Sequenzsteuerung ist.

Der Datenaustausch zwischen den Sequenzreglern *FB_BA_SeqCtrl* und dem Kontrollbaustein FB_BA_SequenceLinkBase [▶ 175] geschieht über die Struktur stSeqLink [▶ 125].

Ausgabewert fPresentValue

Der Funktionsbaustein ermittelt abhängig von den Freigaben *bEn* und *stSequenceLink.bEnSeqLink*, dem Wirksinn *eActionPgm* / *eActionRm* und der Sequenznummer *nActvSeqCtrl* / *nMyNum* den daraus resultierenden Wert am Ausgang *fPresentValue*.

bEn	stSequence-Link.bEnSeqLink	eActionPgm/eAc-tionRm	nActvSeqCtrl/nMy-Num	fPresentValue
TRUE	FALSE	E_BA_Action.eReverse		fMaxOutputPgm / fMaxOutputRm
TRUE	FALSE	E_BA_Action.eDirect		fMinOutputPgm / fMinOutputRm
FALSE	FALSE			0
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl > nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl < nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl < nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl > nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE		nActvSeqCtrl = nMyNum	FB_BA_Loop.fPresentValue

Fehlererkennung

Die unten aufgeführten Fehlermeldungen werden durch den *FB_BA_SeqCtrl* erkannt.

Das x in den Textmeldungen steht für eine nummerische Angabe eines Sequenzreglers.

Die Fehlermeldungen werden in der Entwicklungsumgebung von Tc3 in dem Fenster "Fehlerliste" ausgegeben. Dieses kann unter dem Menüpunkt Ansicht aktiviert werden.

Die Fehlertexte werden über die Properties *ErrorParamMaxSeqCtrl* und *ErrorSeqNumMultiple* ausgegeben.

Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1

Diese Meldung ist die einzige, welche die Sequenzsteuerung deaktiviert.

Deutsch	Englisch
Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1	Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1
Sequenznummer nMyNum = x mehrfach vergeben	Sequence number nMyNum = x assigned multiple times

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_Loop [▶ 225]

Syntax

```
FUNCTION_BLOCK FB_BA_SeqCtrl EXTENDS FB_BA_Loop
VAR_INPUT
  nMyNum            : UDINT;
END_VAR
VAR_OUTPUT
  fE                : REAL;
  bSeqActv          : BOOL;
  nActvSeqCtrl      : UDINT;
  bIsActvSeqCtrl    : BOOL;
  bErr              : BOOL;
END_VAR
```

```
VAR_IN_OUT
  stSeqLink          : ST_BA_SeqLink;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nMyNum	UDINT	Meine Nummer in der Sequenz. Diese Zahl darf nur einmal in einer Sequenzsteuerung vorhanden sein. Eine interne Limitierung lässt nur Werte von 1 - XBA_Param.nMaxSeqCtrl [▶ 128] zu.

Ausgänge

Name	Typ	Beschreibung
fE	REAL	Zeigt die Regelabweichung des Sequenzreglers an. Diese ist abhängig vom Wirksinn des Sequenzreglers. E_BA_Action.eDirect -> fE = fX-fW E_BA_Action.eReverse -> fE = fW-fX
bSeqActv	BOOL	Der Sequenzregler ist in der Regelsequenz eingebunden.
nActvSeqCtrl	UDINT	Nummer des aktiven Sequenzreglers.
bIsActvSeqCtrl	BOOL	Zeigt an, dass der Sequenzregler der aktive in der Sequenzregelung ist. Das Property *IsActvSeqCtrl* zeigt diese Meldung ebenfalls an.
bErr	BOOL	Zeigt an, dass ein Fehler erkannt wurde. Detailliertere Informationen zeigen die Properties *ErrorParamMaxSeqCtrl und SeqNumMultiple*. Unter Fehlererkennung [▶ 173] sind weitere Erläuterungen zur Fehleranalyse zu finden.

/ Ein- Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem Steuerbaustein FB_BA_SequenceLinkBase [▶ 175].

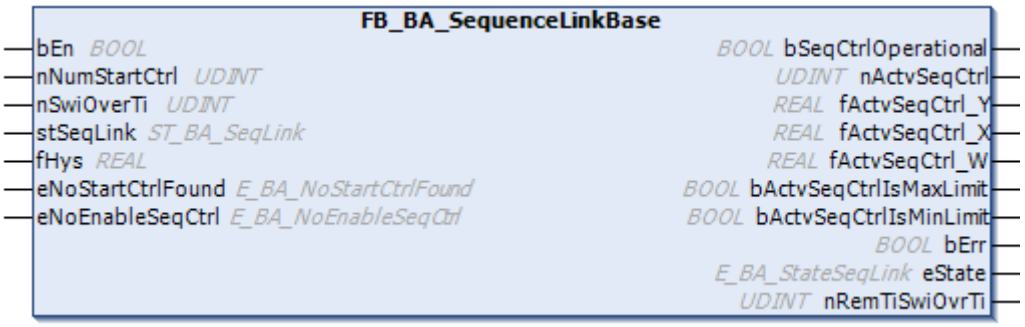
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des aktiven Sequenzcontrollers an.
ErrorParamMaxSeqCtrl	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1", siehe BA_Param.nMaxSeqCtrl [▶ 289]. Der globale Parameter muss auf jeden Fall angepasst werden, weil aufgrund der falschen Parametrierung die Sequenz stoppt.
SeqNumMultiple	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Sequence number nMyNum = x assigned multiple times". Eine Prüfung der angezeigten Sequenznummer ist erforderlich, weil diese mehrfach in der Sequenz vergeben wurde.
IsActvSeqCtrl	BOOL	Get	Der Sequenzregler ist der aktive in der Sequenzsteuerung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_XBA ab v5.6.1.0

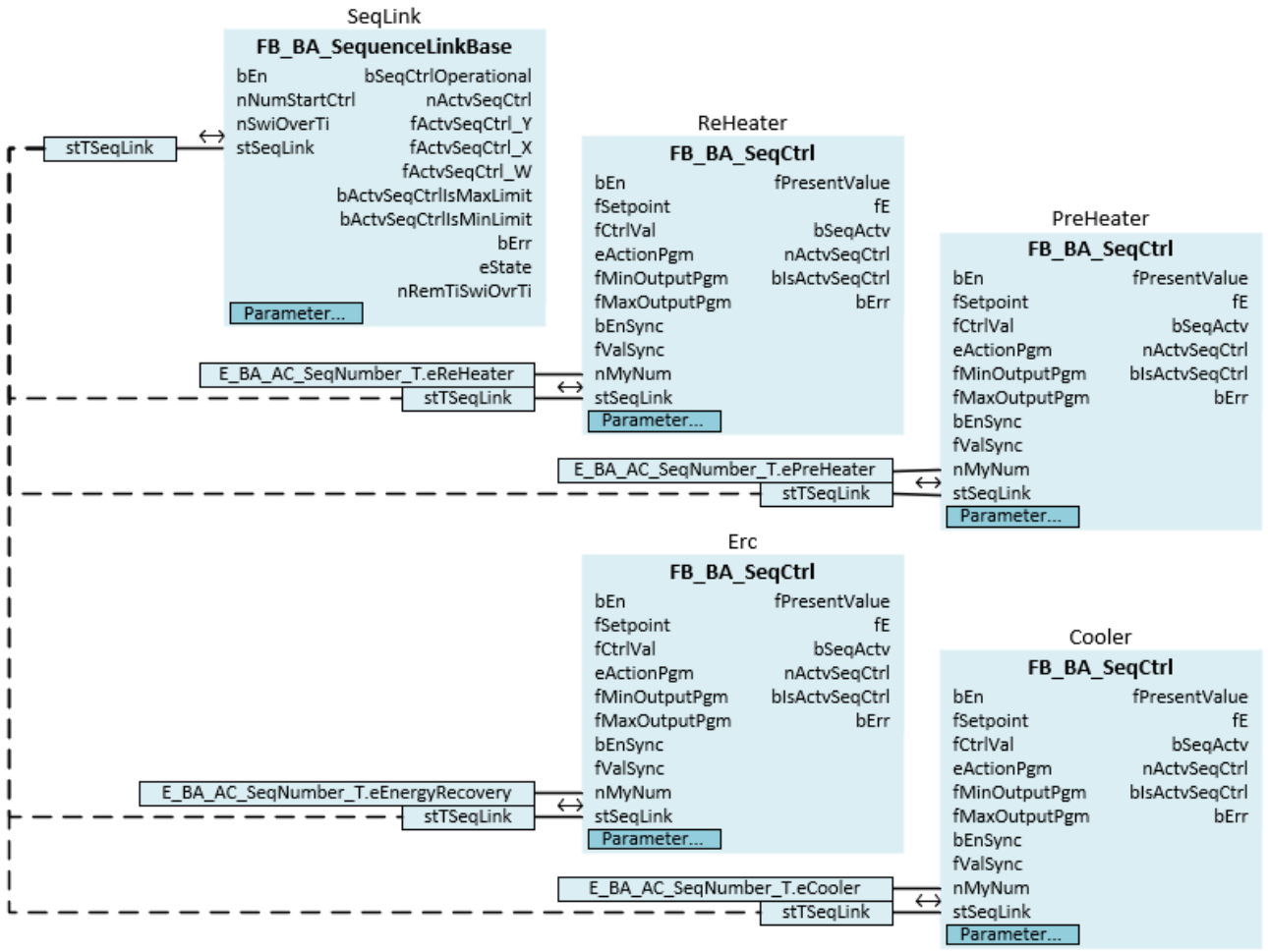
6.1.2.1.3.1.6.2 FB_BA_SequenceLinkBase



Dieser Baustein stellt die übergeordnete Verbindung einer Sequenzregelung dar.

Datenaustausch Sequenzregelung

Der Datenaustausch zwischen den Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem *FB_BA_SequenceLinkBase* geschieht über die Struktur stSeqLink [▶ 125].



Startverhalten der Sequenz

Eine TRUE-Signal am Eingang *bEn* aktiviert die komplette Sequenzregelung. Der Baustein wird zunächst denjenigen Sequenzregler aktiv schalten, welcher an *nNumStartCtrl* benannt ist. Alle anderen Sequenzregler richten Ihren Ausgabewert [▶ 172] nach der Rangordnung des aktiven Reglers.

Ist der an *nNumStartCtrl* angegebene Sequenzregler nicht betriebsbereit, so wird mit dem Enum eNoStartCtrlFound die Vorgehensweise definiert, um einen neuen Startregler zu finden. Somit ist gewährleistet, dass die Sequenz immer startet.

Umschalten in der Sequenz

Erreicht ein Sequenzregler seinen Maximal- oder Minimalwert, so wird je nach Regler-Wirksinn auf den nächsten Regler in der Rangfolge umgeschaltet, sofern der Istwert den Sollwert des nächsten Reglers unterschreitet bzw. überschreitet.

Dabei werden 4 Fälle unterschieden:

- Der noch aktive Regler hat einen direkten Wirksinn (Kühlen, E_BA_Action.eDirect) und befindet sich auf seinem Maximalwert: Es wird in der Rangfolge der nächsthöhere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers überschreitet.
- Der noch aktive Regler hat einen direkten Wirksinn (Kühlen, E_BA_Action.eDirect) und befindet sich auf seinem Minimalwert: Es wird in der Rangfolge der nächstniedrigere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers unterschreitet.
- Der noch aktive Regler hat einen indirekten Wirksinn (Heizen, E_BA_Action.eReverse) und befindet sich auf seinem Maximalwert: Es wird in der Rangfolge der nächstniedrigere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers unterschreitet.
- Der noch aktive Regler hat einen indirekten Wirksinn (Heizen, E_BA_Action.eReverse) und befindet sich auf seinem Minimalwert: Es wird in der Rangfolge der nächsthöhere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers überschreitet.

Das Umschalten auf einen anderen Sequenzregler kann durch die Zeitangabe *nSwiOverTi* verzögert werden.

Betriebsbereitschaft der Sequenzregler

Verliert ein Regler in der Sequenz seine Betriebsbereitschaft (fehlende Freigabe *bEn*), so ist er für die gesamte Sequenz nicht mehr verfügbar.

Handelt es sich dabei nicht um den vorher aktiven Regler, so kann es, je nachdem welche Stellgröße dieser Regler ausgegeben hat, zu einer Temperaturänderung kommen, die durch die Reglersequenz nach Möglichkeit wieder ausgeglichen wird.

Handelt es sich jedoch um den aktiven Sequenzregler, so wird mit der Enumeration eNoEnableSeqCtrl die Vorgehensweise definiert, um einen neuen, aktiven Sequenzregler zu finden.

Erlangt ein Sequenzregler seine Betriebsbereitschaft im laufenden Prozess, so wird der Regler der Sequenz hinzugefügt. Je nach Wirksinn E_BA_Action und eigener Sequenznummer *nMyNum* wird er in die Reglersequenz eingereiht und seinen Minimal- bzw. Maximalwert ausgeben. Die daraus resultierende Temperaturänderung wird durch die Reglersequenz nach Möglichkeit wieder ausgeglichen.

Fehlererkennung

Die unten aufgeführten Fehlermeldungen werden durch den *FB_BA_SequenceLinkBase* erkannt.

Das x in den Textmeldungen steht für eine numerische Angabe eines Sequenzreglers.

Die Fehlermeldungen werden in der Entwicklungsumgebung von Tc3 in dem Fenster "Fehlerliste" ausgegeben. Dieses kann unter dem Menüpunkt Ansicht aktiviert werden.

Ausgabe der Fehlertexte über das Property *StateText* ausgegeben.

Die Enumeration eState zeigt zusätzlich die Meldungen an.



Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1

Diese Meldung ist die einzige, welche die Sequenzsteuerung deaktiviert.

Deutsch	Englisch
Der nächste Sequenzsollwert ist kleiner als sein Vorgänger: x	Next sequence setpoint is smaller than its predecessor: x
Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1	Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1
Mehrfache Wirkrichtung "eActn" in der Sequenzsteuerung	Multiple direction of action "eActn" of the sequence controller
Startsequenzregler ist nicht freigegeben: nNumStartCtrl = x	Start sequence controller is not enabled: nNumStartCtrl = x
Kein Sequenzregler ist betriebsbereit	No sequence controller is operational
Sequenznummer mehrfach vergeben: x	Sequence number assigned multiple times: x

Syntax

```
FUNCTION_BLOCK FB_BA_SequenceLinkBase
VAR_INPUT
  bEn                                 : BOOL;
  nNumStartCtrl                       : UDINT;
  {attribute 'parameterUnit':= 's'}
  nSwiOverTi                          : UDINT(0..1800);
END_VAR
VAR_OUTPUT
  bSeqCtrlOperational                 : BOOL;
  nActvSeqCtrl                        : UDINT;
  fActvSeqCtrl_Y                      : REAL;
  fActvSeqCtrl_X                      : REAL;
  fActvSeqCtrl_W                      : REAL;
  bActvSeqCtrlIsMaxLimit              : BOOL;
  bActvSeqCtrlIsMinLimit              : BOOL;
```

```
  bErr                                    : BOOL;
  eState                                  : E_BA_StateSeqLink;
  {attribute 'parameterUnit':= 's'}
  nRemTiSwiOvrTi                          : UDINT;
END_VAR
VAR_IN_OUT
  stSeqLink                               : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  fHys                                    : REAL := 0.2;
  eNoStartCtrlFound                       : E_BA_NoStartCtrlFound := E_BA_NoStartCtrlFound.eLastActi
onReverse;
  eNoEnableSeqCtrl                        : E_BA_NoEnableSeqCtrl := E_BA_NoEnableSeqCtrl.eNextAction
;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Mit Erhalt der Freigabe des Funktionsbausteins wird die Sequenzsteuerung aktiviert. Zuvor müssen aber die Sequenzregler ihre Freigabe erhalten haben.
nNumStartCtrl	UDINT	Ordnungszahl des Sequenzreglers, welcher bei der Aktivierung der Sequenz der Startregler sein soll. Eine interne Limitierung lässt nur Werte von 1 - BA_Param.nMaxSeqCtrl [▶ 289] zu.
nSwiOverTi	UDINT	Das Umschalten auf einen anderen Sequenzregler kann durch die Zeitangabe verzögert werden. Eine interne Limitierung lässt nur Werte von 0 - 1800 Sekunden zu.

Ausgänge

Name	Typ	Beschreibung
bSeqCtrlOperational	BOOL	Die Sequenzsteuerung ist betriebsbereit bzw. läuft.
nActvSeqCtrl	UDINT	Die Ausgangsvariable zeigt die Nummer des aktiven Sequenzcontrollers an.
fActvSeqCtrl_Y	REAL	Stellgröße des aktiven Sequenzreglers.
fActvSeqCtrl_X	REAL	Istwert des aktiven Sequenzreglers.
fActvSeqCtrl_W	REAL	Sollwert des aktiven Sequenzreglers.
bActvSeqCtrlIsMaxLimit	BOOL	Die obere Ausgabebegrenzung des aktiven Sequenzreglers ist erreicht. Wurde noch nicht auf den nächsten Sequenzregler weitergeschaltet, so wird *bActvSeqCtrlIsMaxLimit* erst auf FALSE gesetzt, wenn *fActvSeqCtrl_Y* unterhalb von (*fActvSeqCtrl_Y* - *fHys*) gefallen ist.
bActvSeqCtrlIsMinLimit	BOOL	Die untere Ausgabebegrenzung des aktiven Sequenzreglers ist erreicht. Wurde noch nicht auf den nächsten Sequenzregler weiter geschaltet, so wird *bActvSeqCtrlIsMinLimit* erst auf FALSE gesetzt, wenn *fActvSeqCtrl_Y* oberhalb von (*fActvSeqCtrl_Y* + *fHys*) gestiegen ist.
bErr	BOOL	Zeigt an, dass ein Fehler erkannt wurde. Detailliertere Informationen zeigen das Property *StateText* an. Zusätzlich zeigt das Enum eState weitere Informationen an. Unter Fehlererkennung [▶ 177] sind weitere Erläuterungen zur Fehleranalyse zu finden.
eState	E_BA_StateSeqLink	Die Enumeration zeigt den Status des *FB_BA_SequenceLinkBase* und der Sequenzregelung an.
nRemTiSwiOvrTi	UDINT	Countdown der Umschaltung auf den nächsten Sequenzregler [s].

/ Ein- Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem Steuerbaustein FB_BA_SequenceLinkBase [▶ 175].

Eingänge CONSTANT

Name	Typ	Beschreibung
fHys	REAL	Dieser Hysteresewert legt eine Hysterese um die obere und untere Ausgabebegrenzung *bActvSeqCtrlIsMaxLimit* / *bActvSeqCtrlIsMinLimit* des aktiven Sequenzreglers. Eine interne Limitierung lässt nur Werte von 0 - 2 zu.
eNoStartCtrlFound	E_BA_NoStartCtrlFound	Wenn während des laufenden Prozesses der aktive Sequenzregler seine Betriebsbereitschaft verliert, dann wird mit der Enumeration die Vorgehensweise definiert, um einen neuen, aktiven Sequenzregler zu finden.
eNoEnableSeqCtrl	E_BA_NoEnableSeqCtrl	Ist der Startregler an dem Eingang *nNumStartCtrl* nicht vorhanden oder freigegeben, so wird mit der Enumeration die Vorgehensweise definiert, um einen neuen Startregler zu finden.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
FirstActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des ersten, betriebsbereiten Sequenzreglers an.
FirstActvSeqCtrlWithActionDirect	UDINT	Get	Das Property zeigt die Nummer des ersten, betriebsbereiten Sequenzreglers mit dem Wirksinn *E_BA_Action.eDirect* an.
LastActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des letzten, betriebsbereiten Sequenzreglers an.
LastActvSeqCtrlWithActionReverse	UDINT	Get	Das Property zeigt die Nummer des letzten, betriebsbereiten Sequenzreglers mit dem Wirksinn *E_BA_Action.eReverse* an.
NextActionActvSeqCtrl	UDINT	Get	Je nach Wirksinn des aktiven Sequenzreglers *nActvSeqCtrl* wird hier der nächste, betriebsbereite Sequenzregler angezeigt. Wirksinn Kühlen/Direct: nächster betriebsbereiter Sequenzregler mit dem Wirksinn Kühlen/Direct (nach 5 würde 6 oder höher kommen). Wenn keiner gefunden wird, dann wird *E_BA_NoEnableSeqCtrl.eLastSeqNum* umgesetzt. Wirksinn Heizen/Reverse: vorheriger betriebsbereiter Sequenzregler mit dem Wirksinn Heizen/Reverse (nach 5 würde 4 oder niedriger kommen). Wenn keiner gefunden wird, dann wird *E_BA_NoEnableSeqCtrl.eFirstSeqNum* umgesetzt.
NextActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des nächsten betriebsbereiten Sequenzreglers nach dem letzten aktiven an (nach 5 würde 6 oder höher kommen).
PreviousActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des vorherigen, betriebsbereiten Sequenzreglers nach dem letzten aktiven an (nach 5 würde 4 oder niedriger kommen).
ActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des aktiven Sequenzcontrollers an, siehe *nActvSeqCtrl*.
ActvSeqCtrl_ControlValue	REAL	Get	Das Property zeigt die Stellgröße des aktiven Sequenzreglers an, siehe *fActvSeqCtrl_Y*.
ActvSeqCtrl_IsMaxLimit	BOOL	Get	Das Property zeigt an, dass die obere Ausgabebegrenzung des aktiven Sequenzreglers erreicht ist, siehe *bActvSeqCtrlIsMaxLimit*.
ActvSeqCtrl_IsMinLimit	BOOL	Get	Das Property zeigt an, dass die untere Ausgabebegrenzung des aktiven Sequenzreglers erreicht ist, siehe *bActvSeqCtrlIsMinLimit*.
ErrorParamMaxSeqCtrl	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1", siehe BA_Param.nMaxSeqCtrl [▶ 289]. Der globale Parameter muss auf jeden Fall angepasst werden, weil aufgrund der falschen Parametrierung die Sequenz stoppt.
ErrorStartSeqCtrlNotOperable	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Start sequence controller is not enabled: nNumStartCtrl = x". Eine Prüfung des Startsequenzreglers, welcher mit der Eingangsvariablen *nNumStartCtrl* angegeben wurde, ist erforderlich.

Name	Typ	Zugriff	Beschreibung
MultiDirectionActionSeqCtrl	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: Multiple direction of action "eActn" of the sequence controller: x. Mehrfache Wirkrichtung der Sequenz Regler. Es wird immer vom Sequenzregler mit der kleinsten Nummer auf den nächstgrößeren geschaut. Eine Änderung der Wirkrichtung ist möglich.
NextSeqSpIsSmaller	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Next sequence setpoint is smaller than its predecessor: x". Eine Prüfung der Sollwerte der Sequenzregler nach aufsteigender Reihenfolge ist erforderlich.
NoSeqCtrlIsOperational	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "No sequence controller is operational". Eine Prüfung der Betriebsbereitschaft der Sequenzregler ist erforderlich.
SeqNumMulti	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Sequence number assigned multiple times: x". Eine Prüfung der angezeigten Sequenznummer ist erforderlich, weil diese mehrfach in der Sequenz vergeben wurde.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_XBA ab v5.6.1.0

6.1.2.1.3.2 Functions

6.1.2.1.3.2.1 Parameter

6.1.2.1.3.2.1.1 Builder

Die Anwendung von *Buildern* vereinfacht das Initialisieren komplexer Parameter.

Funktionsweise

Die Funktionen bieten die Möglichkeit der Initialisierung einzelner Werte. Dazu werden parameterspezifische Methoden bereitgestellt wie z. B. Methoden zum Erstellen eines Kalenders [▶ 122]:

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

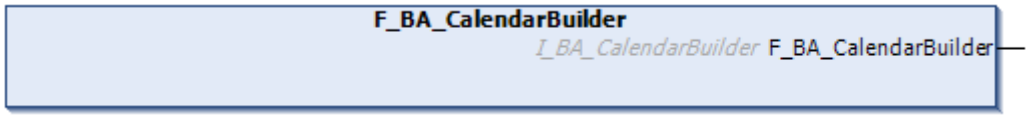
Die Vorteile sind:

- Selbsterklärende Schreibweise
- Hochgradig flexibel
- Gleichermaßen im Deklarations- und Implementierungscode einsetzbar

Siehe auch:

Beispiele

6.1.2.1.3.2.1.1.1 F_BA_CalendarBuilder



Die Funktion stellt Methoden bereit, welche die Initialisierung von Kalendern [▶ 122] vereinfacht:

```
Cal1    : FB_BA_Cal := (
            iParent         := Cals,
            sDescription    := '{Calendar 1}',
            aDateList       := F_BA_CalendarBuilder()
                               .AppendDate(2021, E_BA_Month.eMarch, 10)
                               .AppendDateRange(nFromYear := 2021,E_BA_Month.eMarch,10, nToYear := 2021,E_BA_Month.eMarch,15)
                               .AppendWeekNDay(E_BA_Weekday.eMonday, E_BA_Week.Unspecified, E_BA_Month.Unspecified)
                               .Build()
        );
```

Methoden

Name	Beschreibung
AppendDate [▶ 184]	Fügt ein Datum hinzu.
AppendDateRange [▶ 183]	Fügt einen Datums-Bereich hinzu.
AppendWeekNDay [▶ 185]	Fügt einen bestimmten Wochentag innerhalb eines Monats hinzu.

Siehe auch:

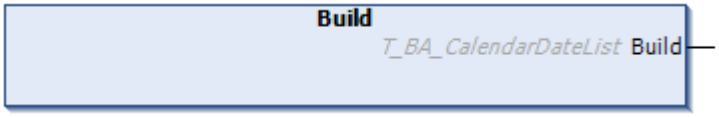
Beispiele

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.2.1.1.1.1 Management

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



Die Methode gibt den Inhalt einer Datumsliste [▶ 122] aus.

6.1.2.1.3.2.1.1.1.2 Value

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



Die Methode fügt dem initialisierten Kalender einen definierten Zeitraum hinzu.

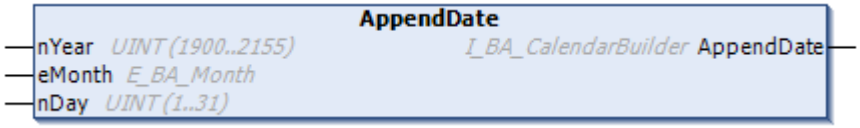
Illustration

```
METHOD AppendDateRange : I_BA_CalendarBuilder
VAR_INPUT
  nFromYear     : UINT(1900 .. 2155);
  eFromMonth    : E_BA_Month := E_BA_Month.Unspecified;
  nFromDay      : UINT(1 .. 31);
  nToYear       : UINT(1900 .. 2155);
  eToMonth      : E_BA_Month := E_BA_Month.Unspecified;
  nToDay        : UINT(1 .. 31);
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nFromYear	UINT	Startjahr von 1900 bis 2155.
eFromMonth	E_BA_Month	Startmonat
nFromDay	UINT	Starttag eines Monats von 1 bis 31.
nToYear	UINT	Endjahr von 1900 bis 2155.
eToMonth	E_BA_Month	Endmonat
nToDay	UINT	Endtag eines Monats von 1 bis 31.

6.1.AppendDate
2.1.
3.2.
1.1.
1.2.
2



Die Methode fügt dem initialisierten Kalender einen Datumswert hinzu.

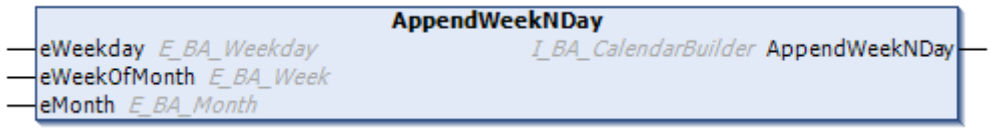
Illustration

```
METHOD AppendDate : I_BA_CalendarBuilder
VAR_INPUT
  nYear     : UINT(1900 .. 2155);
  eMonth    : E_BA_Month := E_BA_Month.Unspecified;
  nDay      : UINT(1 .. 31);
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nYear	UINT	Jahr von 1900 bis 2155.
eMonth	E_BA_Month	Monat
nDay	UINT	Tag eines Monats von 1 bis 31.

6.1.AppendWeekNDay
2.1.
3.2.
1.1.
1.2.
3



Die Methode fügt dem initialisierten Kalender einen definierten Tag, in einer bestimmten Woche eines Monats, in einem bestimmten Monat hinzu.

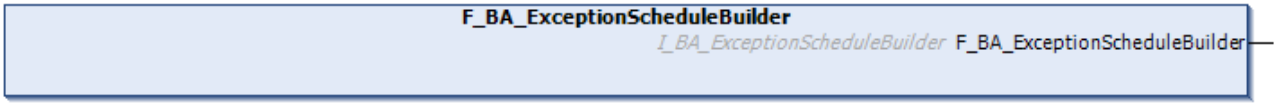
Illustration

```
METHOD AppendWeekNDay : I_BA_CalendarBuilder
VAR_INPUT
  eWeekday        : E_BA_Weekday  := E_BA_Weekday.Invalid;
  eWeekOfMonth    : E_BA_Week     := E_BA_Week.Invalid;
  eMonth          : E_BA_Month    := E_BA_Month.Invalid;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eWeekday	E_BA_Weekday	Wochentag
eWeekOfMonth	E_BA_Week	Woche innerhalb eines Monats.
eMonth	E_BA_Month	Monat

6.1.2.1.3.2.1.1.2 F_BA_ExceptionScheduleBuilder



Die Funktion stellt Methoden bereit, welche die Initialisierung von Ausnahmekalendern vereinfacht:

```
Sched      : FB_BA_SchedA := (
               sDescription        := '{Scheduler 1}',
               aException          := F_BA_ExceptionScheduleBuilder()
                                        .AppendDate(2021, E_BA_Month.eMarch, 10)
                                            .AppendA(T#0H, 1.0).AppendA(T#6H, 2.0).AppendNull(T#12H)
                                            .Close()
                                        .Build()
             );
```

Methoden

Name	Beschreibung
AppendDate [▶ 187]	Fügt ein Datum hinzu.

Siehe auch:

Beispiele

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.2.1.1.2.1 Management

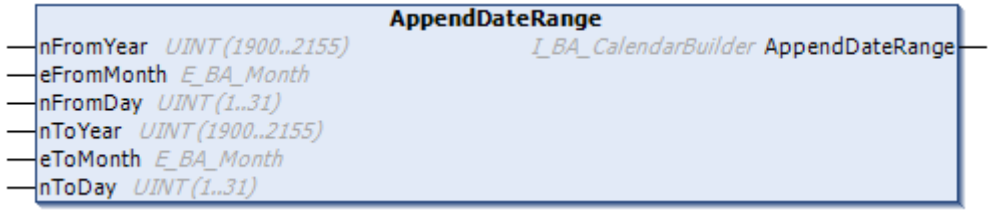
6.1.Build
2.1.
3.2.
1.1.
2.1.
1



Die Methode gibt den Inhalt Ausnahmekalenders [▶ 124] aus.

6.1.2.1.3.2.1.1.2.2 Value

6.1.AppendDateRange
2.1.
3.2.
1.1.
2.2.
1



Die Methode fügt dem initialisierten Kalender einen definierten Zeitraum hinzu.

Illustration

```
METHOD AppendDateRange : I_BA_CalendarBuilder
VAR_INPUT
  nFromYear     : UINT(1900 .. 2155);
  eFromMonth    : E_BA_Month := E_BA_Month.Unspecified;
  nFromDay      : UINT(1 .. 31);
  nToYear       : UINT(1900 .. 2155);
  eToMonth      : E_BA_Month := E_BA_Month.Unspecified;
  nToDay        : UINT(1 .. 31);
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nFromYear	UINT	Startjahr von 1900 bis 2155.
eFromMonth	E_BA_Month	Startmonat
nFromDay	UINT	Starttag eines Monats von 1 bis 31.
nToYear	UINT	Endjahr von 1900 bis 2155.
eToMonth	E_BA_Month	Endmonat
nToDay	UINT	Endtag eines Monats von 1 bis 31.

6.1.AppendDate
2.1.
3.2.
1.1.
2.2.
2



Die Methode fügt dem initialisierten Kalender einen Datumswert hinzu.

Illustration

```
METHOD AppendDate : I_BA_CalendarBuilder
VAR_INPUT
  nYear     : UINT(1900 .. 2155);
  eMonth    : E_BA_Month := E_BA_Month.Unspecified;
  nDay      : UINT(1 .. 31);
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nYear	UINT	Jahr von 1900 bis 2155.
eMonth	E_BA_Month	Monat
nDay	UINT	Tag eines Monats von 1 bis 31.

6.1.AppendWeekNDay
2.1.
3.2.
1.1.
2.2.
3



Die Methode fügt dem initialisierten Kalender einen definierten Tag, in einer bestimmten Woche eines Monats, in einem bestimmten Monat hinzu.

Illustration

```
METHOD AppendWeekNDay : I_BA_CalendarBuilder
VAR_INPUT
  eWeekday        : E_BA_Weekday  := E_BA_Weekday.Invalid;
  eWeekOfMonth    : E_BA_Week     := E_BA_Week.Invalid;
  eMonth          : E_BA_Month    := E_BA_Month.Invalid;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eWeekday	E_BA_Weekday	Wochentag
eWeekOfMonth	E_BA_Week	Woche innerhalb eines Monats.
eMonth	E_BA_Month	Monat

6.1.2.1.3.2.1.1.3 F_BA_ParameterRolesBuilder



Die Funktion stellt Methoden bereit, welche die Initialisierung von Parameterrollen vereinfacht:

```
Parameters      : FB_BA_Param := (
                    DefWriteAccess  := F_BA_ParameterRolesBuilder()
                                       .Set(E_BA_Parameter.eEventEnable, E_BA_Role.eInternal)
                                       .Build()
                );
```

Siehe auch:

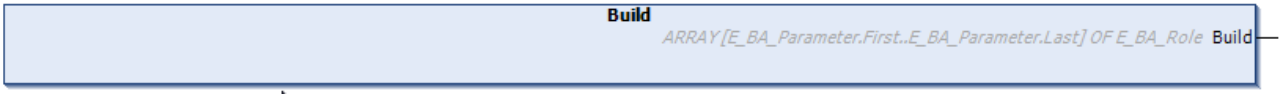
FB_BA_Param [▶ 163]

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.2.1.1.3.1 Management

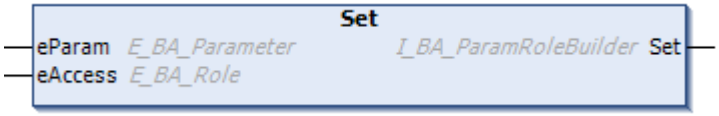
6.1.Build
2.1.
3.2.
1.1.
3.1.
1



Die Methode gibt den Inhalt der möglichen BACnet Properties [▶ 110] entsprechend der Nutzerrolle [▶ 101] aus.

6.1.2.1.3.2.1.1.3.2 Value

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



Die Methode stellte ein BACnet-Property in Abhängigkeit der Nutzerrolle zur Verfügung.

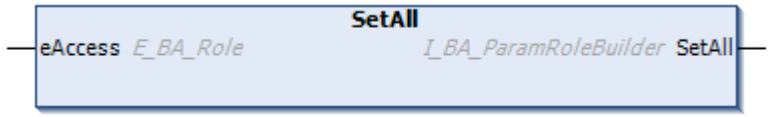
Illustration

```
METHOD Set : I_BA_ParamRoleBuilder
VAR_INPUT
  eParam     : E_BA_Parameter;
  eAccess    : E_BA_Role;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eParam	E_BA_Parameter [▶ 110]	BACnet-Property.
eAccess	E_BA_Role [▶ 101]	Definition der Zugriffsrechte.

6.1.SetAll
2.1.
3.2.
1.1.
3.2.
2



Die Methode stellte alle BACnet-Properties in Abhängigkeit der Nutzerrolle zur Verfügung.

Illustration

```
METHOD SetAll : I_BA_ParamRoleBuilder
VAR_INPUT
  eAccess    : E_BA_Role;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eAccess	E_BA_Role [▶ 101]	Definition der Zugriffsrechte.

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



Die Methode stellte mehrere BACnet-Properties in Abhängigkeit der Nutzerrolle zur Verfügung.

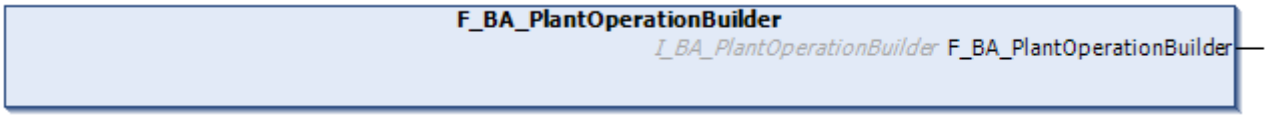
Illustration

```
METHOD SetRange : I_BA_ParamRoleBuilder
VAR_INPUT
  eParamFrom    : E_BA_Parameter;
  eParamTo      : E_BA_Parameter;
  eAccess       : E_BA_Role;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eParamFrom	E_BA_Parameter [▶ 110]	Erstes BACnet Property.
eParamTo	E_BA_Parameter [▶ 110]	Letztes BACnet Property.
eAccess	E_BA_Role [▶ 101]	Definition der Zugriffsrechte.

6.1.2.1.3.2.1.1.4 F_BA_PlantOperationBuilder



Die Funktion stellt Methoden bereit, welche die Initialisierung des Anlagenbetriebs vereinfacht:

```
PlantCtrl   : FB_BA_PlantCtrl := (
              stOperation         := F_BA_PlantOperationBuilder()
                                     .AddAggregate(Dmpr,             FALSE,  FALSE,   T#0S,   T#0S)
                                     .AddAggregate(PreHtr,           TRUE,   FALSE,   T#0S,   T#0S)
                                     .AddAggregate(Fan,              FALSE,  FALSE,   T#0S,   T#10S)
                                     .AddOperationMode(E_OpMod_AC.eDisturb, E_BA_Priority.eCritical, E_BA_AggregateIgnoreFlags.Delay)
                                         .Close()
                                     .AddOperationMode(E_OpMod_AC.eSmokeExtract, E_BA_Priority.eLifeSafety, E_BA_AggregateIgnoreFlags.All)
                                         .SetAggregate(Dmpr,         100)
                                         .SetAggregate(PreHtr,       E_OpMod_Binary.eOn)
                                         .SetAggregate(Fan,          E_OpMod_Step2.eStep2)
                                         .Close()
                                     .AddOperationMode(E_OpMod_AC.eOff, E_BA_Priority.eProgram, 0)
                                         .Close()
                                     .AddOperationMode(E_OpMod_AC.eOn, E_BA_Priority.eProgram, 0)
                                         .SetAggregate(Dmpr,         100)
                                         .SetAggregate(PreHtr,       E_OpMod_Binary.eOn)
                                         .SetAggregate(Fan,          E_OpMod_Step2.eStep1)
                                         .Close()
                                     .AddOperationMode(E_OpMod_AC.eNightCool, E_BA_Priority.eProgram, 0)
                                         .SetAggregate(Dmpr,         100)
                                         .SetAggregate(Fan,          E_OpMod_Step2.eStep2)
                                         .Close()
                                     .Build()
          );
```

Methoden

Name	Beschreibung
AddAggregate [▶ 191]	Fügt ein Aggregat hinzu.
AddOperationMode [▶ 191]	Fügt einen Operation Mode hinzu.

Siehe auch:

FB_BA_PlantCtrl [▶ 491]

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.2.1.1.4.1 Management

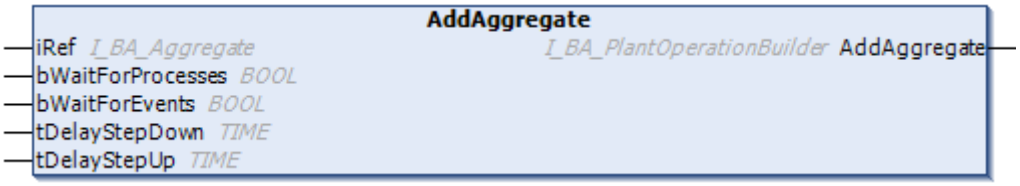
6.1.Build
2.1.
3.2.
1.1.
4.1.
1



Die Methode gibt den Betriebszustand [▶ 125] der Anlage aus.

6.1.2.1.3.2.1.1.4.2 Value

6.1.AddAggregate
2.1.
3.2.
1.1.
4.2.
1



Die Methode definiert das Verhalten von Aggregaten.

Illustration

```
METHOD AddAggregate : I_BA_PlantOperationBuilder
VAR_INPUT
  iRef                : I_BA_Aggregate;
  bWaitForProcesses   : BOOL    := FALSE;
  bWaitForEvents      : BOOL    := FALSE;
  tDelayStepDown      : TIME;
  tDelayStepUp        : TIME;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
iRef	I_BA_Aggregate	Verweis auf das entsprechende Aggregat.
bWaitForProcesses	BOOL	Aktive Prozesse werden beim Schalten berücksichtigt.
bWaitForEvents	BOOL	Ereignisse dieser (und höherer) Lock-Priorität werden beim Schalten berücksichtigt.
tDelayStepDown	TIME	Verzögerung beim Herunterschalten.
tDelayStepUp	TIME	Verzögerung beim Hochschalten.

6.1.AddOperationMode
2.1.
3.2.
1.1.
4.2.
2



Die Methode definiert eine Betriebsart für den Anlagenbetrieb.

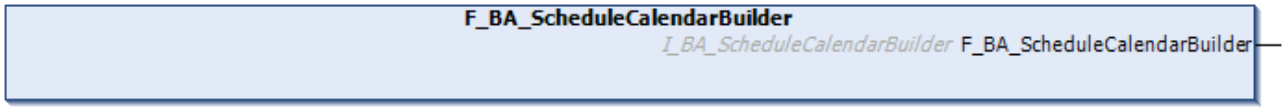
Illustration

```
METHOD AddOperationMode : I_BA_PlantOperationModeBuilder
VAR_INPUT
  nOpMode     : UDINT;
  ePrio       : E_BA_Priority;
  eIgnore     : E_BA_AggregateIgnoreFlags    := E_BA_AggregateIgnoreFlags.None;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nOpMode	UDINT	Betriebsart.
ePrio	E_BA_Priority [▶ 112]	Priorität
eIgnore	E_BA_AggregateIgnoreFlags [▶ 108]	Definiert die Schaltbedingungen.

6.1.2.1.3.2.1.1.5 F_BA_ScheduleCalendarBuilder



Die Funktion stellt Methoden bereit, welche die Initialisierung von Zeitschaltplänen vereinfacht:

```
Sched   : FB_BA_SchedA := (
            sDescription        := 'Scheduler 1',
            aCalendar           := F_BA_ScheduleCalendarBuilder()
                                    .AppendReference(Cal1)
                                        .AppendA(T#0H, 1.0).AppendA(T#6H, 2.0).AppendNull(T#12H)
                                        .Close()
                                    .Build(),
        );
```

Methoden

Name	Beschreibung
AppendReference [▶ 193]	Fügt eine Referenz auf einen Kalender hinzu.

Siehe auch:

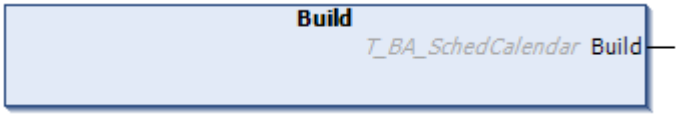
Beispiele

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.2.1.1.5.1 Management

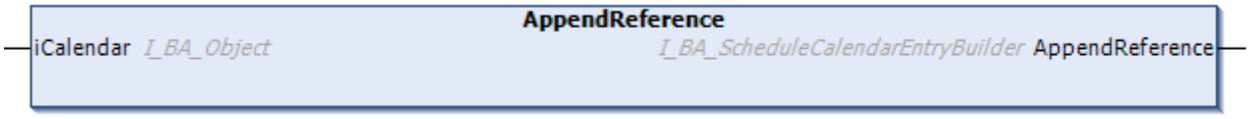
6.1.Build
2.1.
3.2.
1.1.
5.1.
1



Die Methode gibt den Inhalt eines Zeitschaltplans [▶ 124] aus.

6.1.2.1.3.2.1.1.5.2 Value

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



Die Methode übergibt eine Kalenderreferenz.

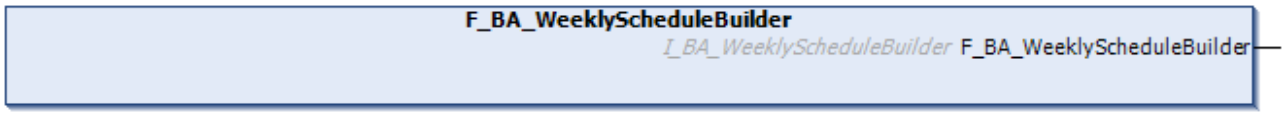
Illustration

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

Eingänge

Name	Typ	Beschreibung
iCalendar	I_BA_Object	Kalenderreferenz.

6.1.2.1.3.2.1.1.6 F_BA_WeeklyScheduleBuilder



Die Funktion stellt Methoden bereit, welche die Initialisierung von Wochenzeitschaltplänen vereinfacht:

```
Sched   : FB_BA_SchedB := (
            sDescription        := 'Scheduler 1',
            aWeek               := F_BA_WeeklyScheduleBuilder()
                                    .SetDayRange(E_BA_Weekday.eSaturday, E_BA_Weekday.eSunday)
                                        .AppendB(T#0H, FALSE).AppendB(T#10H, TRUE).AppendB(T#12H, FALSE)
                                        .Close()
                                    .Build()
        );
```

Methoden

Name	Beschreibung
SetDayRange [▶ 194]	Legt die Tage fest, an denen der Kalender aktiv ist.

Siehe auch:

Beispiele

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.2.1.1.6.1 Management

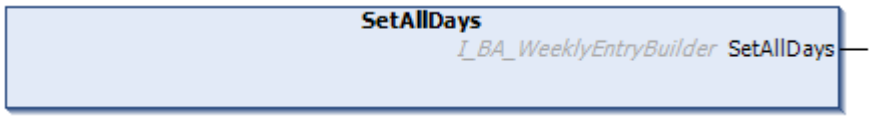
6.1.Build
2.1.
3.2.
1.1.
6.1.
1



Die Methode gibt den Inhalt eines Wochenzeitschaltplans [▶ 124] aus.

6.1.2.1.3.2.1.1.6.2 Value

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



Die Methode definiert alle Wochentage für den Wochenzeitschaltplan.

6.1.SetDay
2.1.
3.2.
1.1.
6.2.
2



Die Methode definiert einen bestimmten Wochentag für den Wochenzeitschaltplan.

Illustration

```
METHOD SetDay : I_BA_WeeklyEntryBuilder
VAR_INPUT
  eDay    : E_BA_Weekday;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eDay	E_BA_Weekday	Ausgewählter Wochentag

6.1.SetDayRange
2.1.
3.2.
1.1.
6.2.
3



Die Methode definiert einen Zeitraum über mehrere Tage für den Wochenzeitschaltplan.

Illustration

```
METHOD SetDayRange : I_BA_WeeklyEntryBuilder
VAR_INPUT
  eDayFrom    : E_BA_Weekday;
  eDayTo      : E_BA_Weekday;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eDayFrom	E_BA_Weekday	Erster Tag des Zeitraums.
eDayTo	E_BA_Weekday	Letzter Tag des Zeitraums.

6.1.2.1.3.3 Objects

6.1.2.1.3.3.1 Local

Das folgende Kapitel bietet Informationen zu den lokalen Serverobjekten.

6.1.2.1.3.3.1.1 Analog

Beschreibung der analogen Bausteine.

6.1.2.1.3.3.1.1.1 FB_BA_AI



Der Funktionsbaustein repräsentiert einen analogen Eingang.

Er dient der Abbildung eines Messwertes innerhalb der Projektstruktur mittels eines analogen Eingangsobjektes.
Der Messwert muss bereits als skalierter Wert innerhalb der Steuerung vorliegen. Das Messsignal kommt bei Verwendung dieses Funktionsbausteins beispielsweise von einem Feldbus und nicht von einer Busklemme.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseAI [▶ 248]

Syntax

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

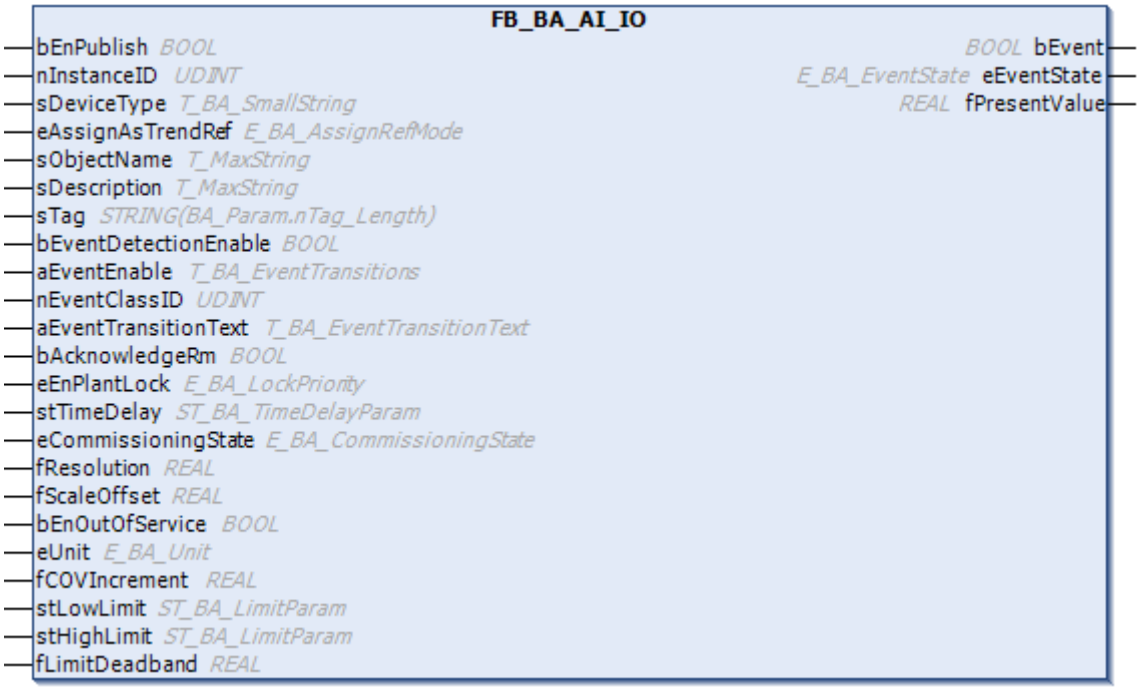
Eingänge

Name	Typ	Beschreibung
fVal	REAL	Skalierter, analoger Eingangswert.
eRlbty	E_BA_Reliability	Verfügbarkeit bzw. Zuverlässigkeit eines Wertes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.2 FB_BA_AI_IO



Der Funktionsbaustein repräsentiert das Objekt für einen Analogeingang.

Er dient der Abbildung eines Messwertes innerhalb der Projektstruktur.
Die Prozessvariablen zur Verknüpfung des Funktionsbausteins mit der Busklemme sind innerhalb des Funktionsbausteins vorhanden.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseAI [▶ 248]

Syntax

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

VAR

Name	Typ	Beschreibung
nRawState	USINT	Variable zur Verknüpfung des Statusinformation einer Klemme.
nRawDataIn	INT	Variable zur Verknüpfung der Rohdaten einer analogen Eingangsklemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.3 FB_BA_AI_IOEx



Der Funktionsbaustein repräsentiert das Objekt eines Analogeingangs.

Die Prozessvariablen zur Verknüpfung des Funktionsbausteins mit der Busklemme sind innerhalb des Funktionsbausteins angelegt. Der Funktionsbaustein ermöglicht ebenfalls die Konfiguration der Analogeingangsklemme. Hierfür wird der Funktionsbaustein FB_BA_KL32xx verwendet.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseAI [▶ 248]

Syntax

```
FUNCTION_BLOCK FB_BA_AI_IOEx EXTENDS FB_BA_BaseAI
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    eSensor               : E_BA_MeasuringElement := BA_Param.fInput_DefSensor;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Operational Parameters'}
```

```
    bConfigurate          : BOOL := TRUE;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    sTerminalType         : STRING;
  {endregion}
END_VAR
VAR
  {region 'I/O'}
    nRawState      AT %I* : USINT;
    nRawDataIn     AT %I* : INT;
    nRawCtrl       AT %Q* : USINT;
    nRawDataOut    AT %Q* : INT;
    iSrcIO                : I_BA_RawAI;
  {endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eSensor	E_BA_MeasuringElement	An diesem Eingang ist der verwendete Sensor einzustellen. Die Einstellung erfolgt durch die Auswahl des Typs in der Enumeration.

Eingänge CONSTANT

Name	Typ	Beschreibung
bConfigurate	BOOL	Start der Klemmenkonfiguration.

VAR

Name	Typ	Beschreibung
sTerminalType	STRING	Gibt den Typen der verwendeten Busklemme an.
nRawState	USINT	Variable zur Verknüpfung des Statusinformation einer Klemme.
nRawDataIn	INT	Variable zur Verknüpfung der Rohdaten einer analogen Eingangsklemme.
nRawCtrl	USINT	Variable zur Verknüpfung mit der Control-Information einer Klemme.
nRawDataOut	INT	Variable zur Verknüpfung des Ausgabewertes der SPS mit dem Prozessabbild einer Ausgangsklemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.4 FB_BA_AI_Raw



Der Funktionsbaustein repräsentiert das Objekt eines Analogeingangs.

Er besitzt Eingangsvariablen zum Anschluss der Prozesswerte der Klemme.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseAI [▶ 248]

Syntax

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

Eingänge

Name	Typ	Beschreibung
nRawState	USINT	Variable zur Verknüpfung der Statusinformation einer Klemme.
nRawVal	INT	Rohwert

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.5 FB_BA_AO



Der Funktionsbaustein dient der Abbildung eines Stellausgangs innerhalb der Projektstruktur.

Das Stellsignal wird bei der Verwendung dieses Funktionsbausteins nicht an eine Klemme, sondern zum Beispiel an einen Feldbus ausgegeben.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseAO [▶ 250]

Syntax

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

Eingänge

Name	Typ	Beschreibung
eRlbty	E_BA_Reliability	Verfügbarkeit bzw. Zuverlässigkeit eines Wertes.
bRawOvrrd	BOOL	An diese Variable kann die Rückmeldung eines in einer Klemme enthaltenen Schalters, für die mechanische Übersteuerung eines Ausgangs, angeschlossen werden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.6 FB_BA_AO_IO



Der Funktionsbaustein repräsentiert ein analoges Ausgangsobjekt innerhalb des Grundgerüstes. Die Variablen zur Verknüpfung des Stellausgangs mit der Klemme sind innerhalb des Funktionsbausteins deklariert.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseAO [▶ 250]

Syntax

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

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eOverridenPolarity	E_BA_Polarity	Ausgangsklemmen mit einer mechanischen Vorrangbedienung melden den Status ihrer Schalter an die Steuerung zurück. Mit dieser Enumeration kann die Polarität der Schalterrückmeldung parametriert werden.

VAR

Name	Typ	Beschreibung
bRawOverridden	BOOL	Variable zur Erfassung einer Übersteuerung von außen.
nRawState	USINT	Variable zur Verknüpfung des Statusinformation einer Klemme.
nRawDataOut	INT	Variable zur Verknüpfung des Ausgabewertes der SPS mit dem Prozessabbild einer Ausgangsklemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.7 FB_BA_AO_Raw



Der Funktionsbaustein repräsentiert ein analoges Ausgangsobjekt innerhalb des Grundgerüstes. Zur Verknüpfung des Stellwertes mit dem Prozessabbild der Klemme dient die Variable *nRawVal*.

Er besitzt ein Prioritäten-Array und ist somit zu kommandieren [▶ 35].

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseAO [▶ 250]

Syntax

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

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

Eingänge

Name	Typ	Beschreibung
bRawOverridden	BOOL	An diese Variable kann die Rückmeldung eines in einer Klemme enthaltenen Schalters, für die mechanischen Übersteuerung eines Ausgangs angeschlossen werden.

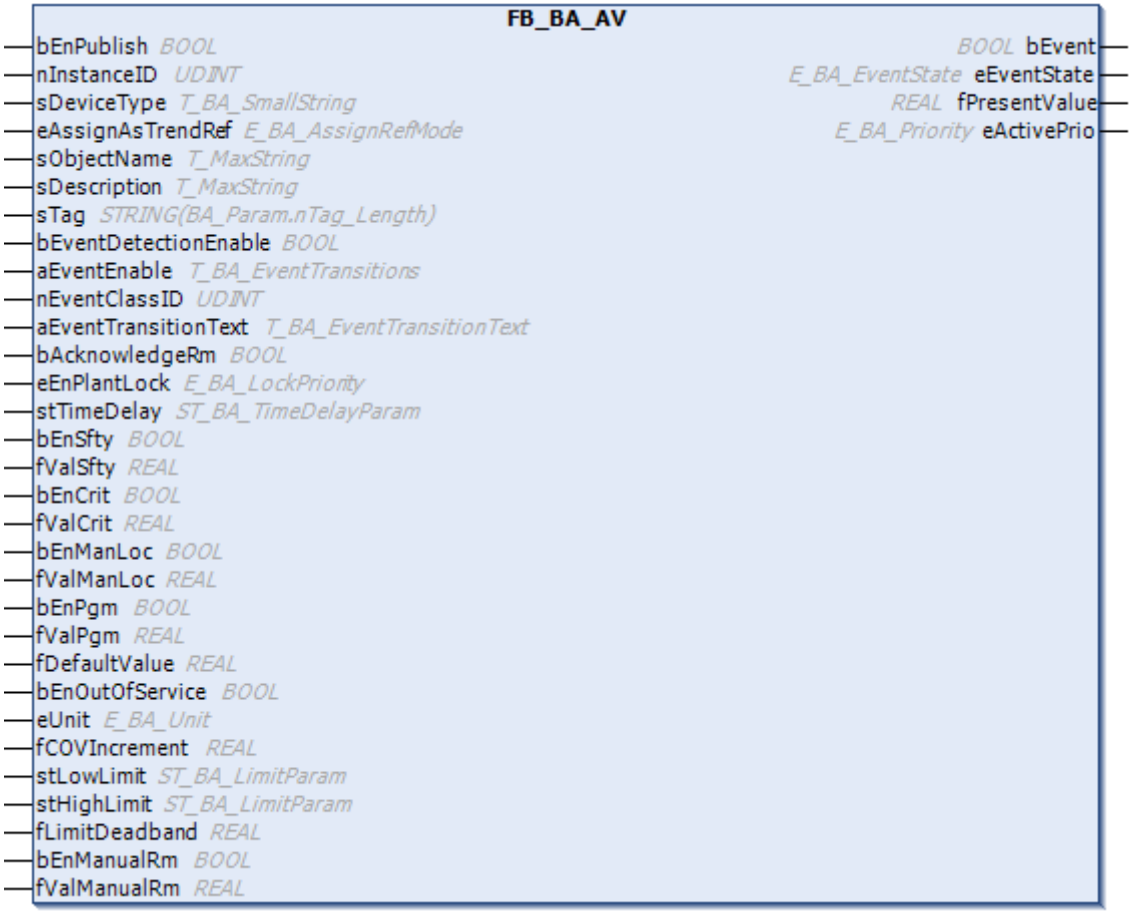
Ausgänge

Name	Typ	Beschreibung
nRawVal	INT	Die Variable dient der Verknüpfung des Rohwertes eines Objektes mit dem Prozessabbild der Ein- bzw. Ausgangsklemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.8 FB_BA_AV



Der Funktionsbaustein repräsentiert ein analoges Value-Objekt.

Er besitzt ein Prioritäten-Array und ist zu kommandieren [▶ 35].

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

Syntax

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

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe der Priorität „Safety“.
fValSfty	REAL	Analoger Wert für die Priorität „Safety“.
bEnCrit	BOOL	Freigabe der Priorität „Critical“.
fValCrit	REAL	Analoger Wert für die Priorität „Critical“.
bEnManLoc	BOOL	Freigabe der Priorität „Manual Local“.
fValManLoc	REAL	Analoger Wert für die Priorität „Manual Local“.
bEnPgm	BOOL	Freigabe der Priorität „Program“.
fValPgm	REAL	Analoger Wert für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.
eActivePrio	E_BA_Priority [▶ 112]	Aktive Priorität

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fDefaultValue	REAL	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
eUnit	E_BA_Unit	Einheit des Ein- bzw. Ausgabewerts eines Analogobjektes.
fCOVIncrement	REAL	Die Variable gibt die Mindeständerung des Present-Value an, die dazu führt, dass eine COV-Notification an die abonnierten COV-Clients ausgegeben wird. Diese Eigenschaft ist erforderlich, wenn die COV-Berichterstattung von diesem Objekt unterstützt wird.
stLowLimit	ST_BA_LimitParam [▶ 122]	Parametrierung der unteren Grenzwertüberwachung eines analogen Objektes. Die Variable *bEnable* muss TRUE sein, damit die Grenzwertüberwachung freigegeben ist. Mit der Variablen *fValue* wird der untere Grenzwert parametriert.
stHighLimit	ST_BA_LimitParam [▶ 122]	Parametrierung der oberen Grenzwertüberwachung eines analogen Objektes. Die Variable *bEnable* muss TRUE sein, damit die Grenzwertüberwachung freigegeben ist. Mit der Variablen *fValue* wird der obere Grenzwert parametriert.
fLimitDeadband	REAL	Totband bzw. Hysterese für die Werte *HighLimit* und *LowLimit* der Grenzwertüberwachung eines analogen Objektes.
bEnManualRm	BOOL	Freigabe der Priorität „Manual Remote“.
fValManualRm	REAL	Analoger Wert für die Priorität „Manual Remote“.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.1.9 FB_BA_AV_Op



Der Funktionsbaustein repräsentiert ein analoges Value-Objekt. Er dient der Anzeige oder Eingabe eines analogen Wertes.

Ist die Eingangsvariable *fValuePgm* verknüpft, dann erkennt der Baustein automatisch, dass er der Anzeige des angeschlossenen Wertes dient. Im anderen Fall dient er der Eingabe z. B. eines Sollwertes.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Syntax

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

Eingänge

Name	Typ	Beschreibung
fValuePgm	Real	Ausgabewerte des Analogobjektes für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eUnit	E_BA_Unit	Einheit des Ein- bzw. Ausgabewerts eines Analogobjektes.
fCOVIncrement	REAL	Die Variable gibt die Mindeständerung des Present-Value an, die dazu führt, dass eine COV-Notification an die abonnierten COV-Clients ausgegeben wird. Diese Eigenschaft ist erforderlich, wenn die COV-Berichterstattung von diesem Objekt unterstützt wird.
fValueRm	REAL	Variable zum Überschreiben eines Analogobjektes von der HMI.

VAR

Name	Typ	Beschreibung
eValueSource	E_BA_ProcessSignalSource [▶ 115]	Die Variable zeigt an, ob ein Objekt des Typs FB_BA_..._OP als Anzeige- oder Eingabeobjekt dient. *eVarInput* = 1 Das Objekt dient der Anzeige eines Wertes. Der Wert wird dem Objekt an einem Eingang innerhalb der SPS übergeben. *eParameter* = 2 Das Objekt dient der Eingabe eines Parameters, welcher von einem BACnet-Client bzw. der TwinCAT HMI verändert werden kann.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2 Binary

Nachfolgend finden Sie die Beschreibung der binären Bausteine.

6.1.2.1.3.3.1.2.1 FB_BA_BI



Der Funktionsbaustein FB_BA_BI repräsentiert ein binäres Eingangsobjekt innerhalb des Grundgerüstes. Die Variablen zur Verknüpfung des Eingangs mit der Klemme sind als Eingangsvariablen am Funktionsbaustein vorhanden.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseBI [▶ 253]

Illustration

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

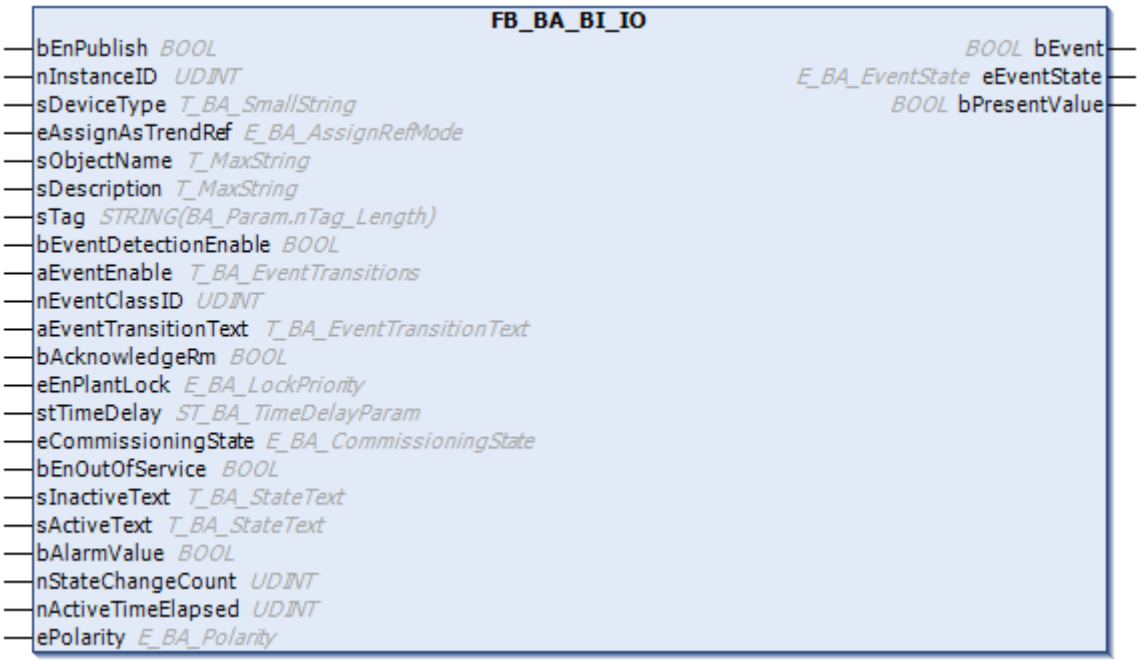
Eingänge

Name	Typ	Beschreibung
bVal	BOOL	Binärwert
eRlbty	E_BA_Reliability	Verfügbarkeit bzw. Zuverlässigkeit eines Wertes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2.2 FB_BA_BI_IO



Der Funktionsbaustein FB_BA_BI_IO repräsentiert ein binäres Eingangsobjekt innerhalb des Grundgerüstes. Die Variablen zur Verknüpfung des Eingangs mit der Klemme sind innerhalb des Funktionsbausteins deklariert.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseBI [▶ 253]

FB_BA_BI [▶ 209]

Illustration

VAR

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

Name	Typ	Beschreibung
bRawVal	BOOL	Variable zur Verknüpfung des Eingangswertes mit der Klemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2.3 FB_BA_BI_Raw



Der Funktionsbaustein FB_BA_BI_Raw generiert ein binäres Input-Objekt innerhalb des Grundgerüsts. Zur Verknüpfung des Binäreingangs mit der Klemme dient die Variable *bRawVal*.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseBI [▶ 253]

Illustration

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

Eingänge

Name	Typ	Beschreibung
bRawVal	BOOL	Rohwert

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2.4 FB_BA_BO



Der Funktionsbaustein FB_BA_BO repräsentiert ein binäres Ausgangsobjekt innerhalb des Grundgerüstes. Die Variablen zur Verknüpfung des Stellausgangs mit der Klemme sind als Ein- bzw. Ausgangsvariablen am Funktionsbaustein vorhanden.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseBO [▶ 255]

Illustration

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

Eingänge

Name	Typ	Beschreibung
eRlbty	E_BA_Reliability	Verfügbarkeit bzw. Zuverlässigkeit eines Wertes.
bRawOvrrd	BOOL	An diese Variable kann die Rückmeldung eines in einer Klemme enthaltenen Schalters, für die mechanische Übersteuerung eines Ausgangs, angeschlossen werden.
bEnFdbck	BOOL	Variable zum Start der Überwachung der Rückmeldung am binären Ausgang.
bRawValFdbck	BOOL	Die Variable entspricht dem Wert des aktuellen Feedbacks.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2.5 FB_BA_BO_IO



Der Funktionsbaustein FB_BA_BO_IO repräsentiert ein binäres Ausgangsobjekt innerhalb des Grundgerüstes. Die Variablen zur Verknüpfung des Schaltausgangs mit der Klemme sind innerhalb des Funktionsbausteins deklariert. Die Feedback-Kontrolle des Binären Ausgangs wird automatisch aktiviert, wenn die Variable *bRawValFeedback* mit dem Prozessabbild einer Klemme verknüpft ist.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseBO [▶ 255]

Illustration

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

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

Eingänge Constant Persistent

Name	Typ	Beschreibung
eFeedbackPolarity	E_BA_Polarity	Variable zur Parametrierung der Polarität der Binären Betriebsrückmeldung eines Ausganges.
eOverridenPolarity	E_BA_Polarity	Ausgangsklemmen mit einer mechanischen Vorrangbedienung melden den Status ihrer Schalter an die Steuerung zurück. Mit dieser Enumeration kann die Polarität der Schalterrückmeldung parametriert werden.

VAR

Name	Typ	Beschreibung
bRawOverridden	BOOL	Variable zur Erfassung einer Übersteuerung von außen.
bRawValFeedback	BOOL	Aktivierung der Feedback Kontrolle nach Verknüpfung mit der Busklemme.
bRawVal	BOOL	Variable zur Verknüpfung des Ausgangswertes mit der Klemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2.6 FB_BA_BO_Raw



Der Funktionsbaustein FB_BA_BO_Raw repräsentiert ein binäres Ausgangsobjekt innerhalb des Grundgerüstes. Zur Verknüpfung des Schaltbefehls mit dem Prozessabbild der Klemme dient die Variable *bRawVal*.

Das Objekt besitzt ein Prioritäten-Array und ist somit zu kommandieren [▶ 35].

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseBO [▶ 255]

Illustration

```
FUNCTION_BLOCK FB_BA_BO_Raw EXTENDS FB_BA_BaseBO
VAR_INPUT
  bRawOvrrd      : BOOL;
  bEnFdbk        : BOOL;
  bRawValFdbk    : BOOL;
END_VAR
VAR_OUTPUT
  bRawVal        : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bRawOvrrd	BOOL	An diese Variable kann die Rückmeldung eines in einer Klemme enthaltenen Schalters, für die mechanische Übersteuerung eines Ausgangs, angeschlossen werden.
bEnFdbck	BOOL	Variable zum Start der Überwachung der Rückmeldung am binären Ausgang.
bRawValFdbck	BOOL	Die Variable entspricht dem Wert des aktuellen Feedbacks.

Ausgänge

Name	Typ	Beschreibung
bRawVal	BOOL	Wert des Binärausgangs zur Verknüpfung mit dem Prozessabbild der Klemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2.7 FB_BA_BV



Der Funktionsbaustein FB_BA_BV repräsentiert ein binäres Value-Objekt.

Er besitzt ein Prioritäten-Array und ist somit zu kommandieren [▶ 35].

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

Illustration

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

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

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe der Priorität „Safety“.
bValSfty	BOOL	Binärer Wert für die Priorität „Safety“.
bEnCrit	BOOL	Freigabe der Priorität „Critical“.
bValCrit	BOOL	Binärer Wert für die Priorität „Critical“.
bEnManLoc	BOOL	Freigabe der Priorität „Manual Local“.
bValManLoc	BOOL	BinärerWert für die Priorität „Manual Local“.
bEnPgm	BOOL	Freigabe der Priorität „Program“.
bValPgm	BOOL	Binärer Wert für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.
eActivePrio	E_BA_Priority [▶ 112]	Aktive Priorität

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nMinimumOffTime	UDINT	Minimale Zeit [s] in der der Present_Value im Zustand INACTIVE verbleiben soll, nachdem ein Schreibvorgang auf Present_Value, den Zustand INACTIVE angenommen hat. Damit kann ein Schutz vor zu schnellem erneutem Anlaufen realisiert werden.
nMinimumOnTime	UDINT	Minimale Zeit [s] in der der Present_Value im Zustand ACTIVE verbleiben soll, nachdem ein Schreibvorgang auf Present_Value, den Zustand ACTIVE angenommen hat. Damit kann ein Schutz vor verfrühtem, erneutem Ausschalten realisiert werden
bDefaultValue	BOOL	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
sInactiveText	T_BA_StateText [▶ 123]	Textausgabe bei inaktivem Objekt.
sActiveText	T_BA_StateText [▶ 123]	Textausgabe bei aktivem Objekt.
bAlarmValue	BOOL	Wert im Falle eines Alarms.
nStateChangeCount	UDINT	Die Variable gibt an, wie oft sich der Zustand des *Present_Value* seit dem Datum und des letzten Reset geändert hat.
nActiveTimeElapsed	UDINT	Zeitangabe [s], in der der *Present_Value* des Objektes den Wert ACTIVE hatte. Die Zeit gilt ab dem letzten Rücksetzen durch das Property *Time_Of_Active_Time_Reset*.
bEnManualRm	BOOL	Freigabe der Priorität „Manual Remote“.
bValManualRm	BOOL	Binärer Wert für die Priorität „Manual Remote“.

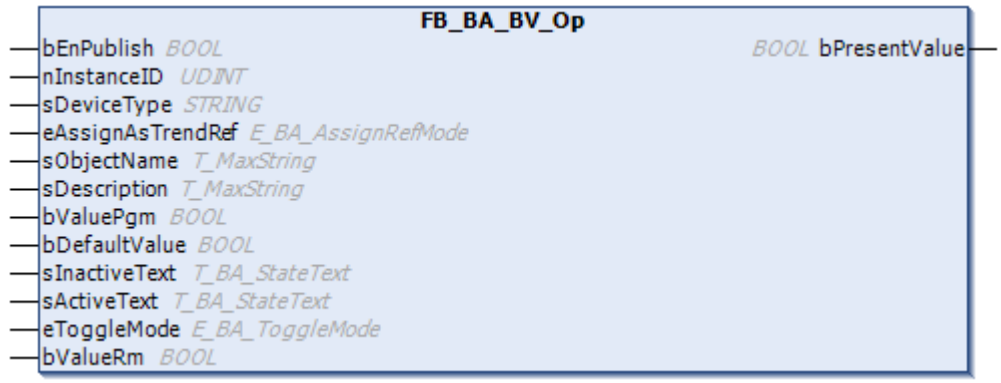
VAR

Name	Typ	Beschreibung
stStateChangeTime	ST_BA_DateTime	Dieses Property zeigt die Uhrzeit der letzten Zustandsänderung an. Die Zustandsänderung bezieht sich auf den *Present_Value* des Objektes.
stStateChangeResetPoint	ST_BA_DateTime	Zeigt Datum und Uhrzeit ab der das Zählen der Statusänderungen begonnen hat.
stActiveTimeResetPoint	ST_BA_DateTime	Gibt den Zeitpunkt an, an dem die Aufzeichnung der Einschaltzeitzeiten des Objektes begonnen hat.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.2.8 FB_BA_BV_Op



Der Funktionsbaustein FB_BA_BV_Op repräsentiert ein binäres Value - Objekt.
Er dient der Anzeige oder Eingaben eines binären Wertes.

Ist die Eingangsvariable *bValuePgm* verknüpft, dann erkennt der Baustein automatisch, dass er der Anzeige des angeschlossenen Wertes dient. Im anderen Fall dient es der Eingabe eines binären Wertes.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

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

Eingänge

Name	Typ	Beschreibung
bValuePgm	BOOL	Wert eines Binärobjektes für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eUnit	E_BA_Unit	Einheit des Ein- bzw. Ausgabewerts eines Analogobjektes.
fCOVIncrement	REAL	Die Variable gibt die Mindeständerung des Present-Value an, die dazu führt, dass eine COV-Notification an die abonnierten COV-Clients ausgegeben wird. Diese Eigenschaft ist erforderlich, wenn die COV-Berichterstattung von diesem Objekt unterstützt wird.
fValueRm	REAL	Variable zum Überschreiben eines Analogobjektes von der HMI.

VAR

Name	Typ	Beschreibung
eValueSource	E_BA_ProcessSignalSource [▶ 115]	Die Variable zeigt an, ob ein Objekt des Typs FB_BA_..._OP als Anzeige- oder Eingabeobjekt dient. *eVarInput* = 1 Das Objekt dient der Anzeige eines Wertes. Der Wert wird dem Objekt an einem Eingang innerhalb der SPS übergeben. *eParameter* = 2 Das Objekt dient der Eingabe eines Parameters, welcher von einem BACnet-Client bzw. der TwinCAT HMI verändert werden kann.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.3 Misc

Beschreibung der Objekte Kalender, Event-Class, Regler, Scheduler und Trend.

6.1.2.1.3.3.1.3.1 FB_BA_Cal



Der Funktionsbaustein FB_BA_Cal repräsentiert einen Kalender innerhalb der Projektstruktur.

Vererbungsgierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK FB_BA_Cal EXTENDS FB_BA_Object IMPLEMENTS I_BA_Cal
VAR_OUTPUT
  bPresentValue       : BOOL;
```

```
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    aDateList         : T_BA_CalendarDateList;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Variable Parameters'}
    bWriteDateList    : BOOL;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
aDateList	T_BA_CalendarDateList [▶ 122]	Datumsliste.

Eingänge CONSTANT

Name	Typ	Beschreibung
bWriteDateList	BOOL	Freigabe zum Beschreiben einer Datumsliste.

Voraussetzungen

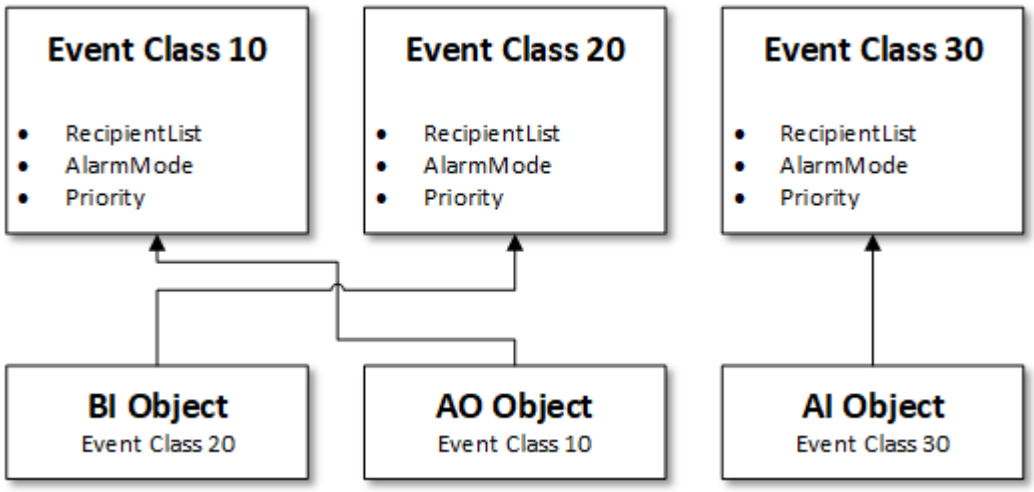
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.17	Tc3_BA2 ab v4.8.9.0

6.1.2.1.3.3.1.3.2 FB_BA_EC



Der Funktionsbaustein FB_BA_EC repräsentiert eine Event–Class (vgl. Notification-Class) innerhalb der Projektstruktur von TF8040.
Die Erkennung eines Events und das objektinterne Melden (Intrinsic Reporting) befindet sich in den eventfähigen Objekten. Die anschließende Verteilung der Events an die Event-Clients wird hingegen nicht in den Objekten, sondern in der Event-Class ausgeführt.

Jedem Eventfähigen Objekt wird eine Event-Class zugewiesen. Einer Eventklasse können eine oder mehrere Objekte zugewiesen werden.



Die Event-Class beschreibt Eigenschaften eines Events. Alle Objekte, welche dieser Event-Class zugewiesen sind, erhalten diese Eigenschaften.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK FB_BA_EC EXTENDS FB_BA_Object IMPLEMENTS I_BA_EventClass
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    aPriority               : ARRAY[E_BA_EventTransition.First .. E_BA_EventTransition.Last] OF UDIN
T;
  {endregion}
  {region 'Fixed Parameters'}
    eEventType              : E_BA_EventType := E_BA_EventType.eOther;
    eAlarmMode              : E_BA_AlarmMode := E_BA_AlarmMode.Invalid;
    aAcknowledgeRequired    : T_BA_EventTransitions;
  {endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
aPriority	ARRAY [E_BA_EventTransition.First..E_BA_EventTransition.Last] OF UDINT	Die Variable gibt die Priorität an, mit der die Ereignisbenachrichtigungen der Event Class übertragen werden. Die Prioritäten reichen von 0 bis einschließlich 255. Eine niedrigere Zahl bedeutet eine höhere Priorität.
eEventType	E_BA_EventType [▶ 105]	Dieser Parameter dient der näheren Beschreibung eines Ereignisses. Der Typ des Ereignisses beschreibt ebenfalls die Darstellung in der TwinCAT HMI [▶ 1139].
eAlarmMode	E_BA_AlarmMode [▶ 103]	Die Anforderungen bezüglich des Quittierens und des Rücksetzens von Events wird nicht an den Objekten, sondern an den Eventklassen parametriert. Es stehen drei Standard Alarm Modi zur Verfügung: Das Quittieren bezieht sich dabei auf kommende und das Rücksetzen auf gehende Alarme. Durch den Alarm-Modus in TwinCAT Building Automation wird festgelegt, ob ein Event (bzw. ein Objekt) quittiert und / oder zurückgesetzt werden muss.

Das Standard-Verhalten einzelner Alarm-Modi kann mittels globaler Parameter [▶ 128] angepasst werden.
Dieses kann notwendig sein, wenn z. B. Betreiber das Einhalten bestimmter Vorgaben einfordern.
Alle Einstellungen sind sorgfältig voreingestellt.
Änderungen wirken sich **maßgeblich** auf das Event-Verhalten eines Objektes aus und sollten mit Bedacht umgesetzt werden!

Name	Typ	Beschreibung
aAcknowledgeRequired	T_BA_EventTransitions [▶ 117]	Quittierung erforderlich.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.3.3 FB_BA_Loop



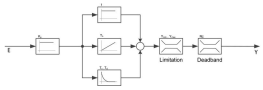
Der Funktionsbaustein FB_BA_Loop repräsentiert einen PID-Regler innerhalb der Projektstruktur von TF8040.

Wirkungsplan

Der Regler kann wahlweise in Parallelstruktur oder mit vorgelagertem P-Glied betrieben werden. Dieses wird durch den Eingang *eOpMode* vorgegeben.

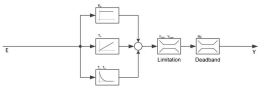
P-Anteil vorgelagert:

eOpMode := E_BA_PIDMode. ePI1D



Parallelstruktur:

eOpMode := E_BA_PIDMode. ePID



Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK FB_BA_Loop EXTENDS FB_BA_Object IMPLEMENTS I_BA_Loop
VAR_INPUT
  bEn                       : BOOL;
  fSetpoint                 : REAL;
  fCtrlVal                  : REAL;
  eActionPgm                : E_BA_Action;
  fMinOutputPgm             : REAL;
  fMaxOutputPgm             : REAL;
  bEnSync                   : BOOL;
  fValSync                  : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue             : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    eOutputUnit             : E_BA_Unit := E_BA_Unit.eOther_Percent;
    fCOVIncrement           : REAL := BA_Param.fDefCOVIncrement;
    eOpMode                 : E_BA_PIDMode := BA_Param.nLoop_DefOpMode;
    eActionRm               : E_BA_Action := E_BA_Action.eDirect;
    fNeutralZone            : REAL:= 0;
    fMinOutputRm            : REAL:= 0;
    fMaxOutputRm            : REAL:= 100;
    fProportionalConstant   : REAL;
    fIntegralConstant       : REAL;
    fDerivativeConstant     : REAL := 0;
    nDampConstant           : UDINT;
    stStepDelay             : ST_BA_StepDelayParam;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    fCtrlDeviation          : REAL;
{endregion}
  END_VAR
VAR
  {region 'Hardware'}
    eActionSource           : E_BA_ProcessSignalSource;
    eMinMaxOutputSource     : E_BA_ProcessSignalSource;
  {endregion}
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung des Bausteins.
fSetpoint	REAL	Sollwert
fCtrlVal	REAL	Rückmeldung des Kontrollwerts zur Errechnung der Regelabweichung vom Sollwert.
eActionPgm	E_BA_Action	Einstellung des Wirksinns.
fMinOutputPgm	REAL	Untere Ausgabebegrenzung des Reglers.
fMaxOutputPgm	REAL	Obere Ausgabebegrenzung des Reglers.
bEnSync	BOOL	Freigabe der Synchronisierung.
fValSync	REAL	Synchronisationswert. Nach einer positiven Flanke an *bEnSync* wird dieser Wert auf *fPresentValue* geschrieben.

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eOutputUnit	E_BA_Unit	Einheit des Ausgangs
fCOVIncrement	REAL	Schrittweite des Present Value, die eine COV Notification auslöst.
eOpMode	E_BA_PIDMode	Vor- oder parallel- gestellter P-Anteil.
eActionRm	E_BA_Action	Wirksinn
fNeutralZone	REAL	Neutrale Zone
fMinOutputRm	REAL	Minimaler Ausgangswert durch externe Übersteuerung.
fMaxOutputRm	REAL	Maximaler Ausgangswert durch externe Übersteuerung.
fProportionalConstant	REAL	Proportionalkonstante.
fIntegralConstant	REAL	Nachhaltezeit.
fDerivativeConstant	REAL	Vorhaltezeit.
nDampConstant	UDINT	Dämpfungskonstante.

VAR

Name	Typ	Beschreibung
fCtrlDeviation	REAL	Regelabweichung
eActionSource	E_BA_ProcessSignalSource [▶ 115]	Definition, ob der Wirksinn als Variable oder als Parameter zu behandeln ist.
eMinMaxOutputSource	E_BA_ProcessSignalSource [▶ 115]	Definition, ob die Ausgangsquelle als Variable oder als Parameter zu behandeln ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.3.4 FB_BA_SchedA



Der Funktionsbaustein FB_BA_SchedA repräsentiert einen analogen Scheduler innerhalb der Projektstruktur von TF8040.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_BaseSched [▶ 261]

Illustration

```
FUNCTION_BLOCK FB_BA_SchedA EXTENDS FB_BA_BaseSched IMPLEMENTS I_BA_SchedA
VAR_OUTPUT
  fPresentValue      : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    fDefaultValue    : REAL;
  {endregion}
  {region 'Variable Parameters'}
    eUnit            : E_BA_Unit:= E_BA_Unit.Invalid;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fDefaultValue	REAL	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
eUnit	E_BA_Unit	Einheit des Ein- bzw. Ausgabewerts eines Analogobjektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.3.5 FB_BA_SchedB



Der Funktionsbaustein FB_BA_SchedB repräsentiert einen binären Scheduler innerhalb der Projektstruktur von TF8040.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_BaseSched [▶ 261]

Illustration

```
FUNCTION_BLOCK FB_BA_SchedB EXTENDS FB_BA_BaseSched IMPLEMENTS I_BA_SchedB
VAR_OUTPUT
  bPresentValue      : BOOL;
  bPredictedValue    : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    bDefaultValue    : BOOL;
  {endregion}
  {region 'Variable Parameters'}
    sInactiveText    : T_BA_StateText;
    sActiveText      : T_BA_StateText;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.
bPredictedValue	BOOL	Wert, der nach dem nächsten Schalten angenommen wird.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bDefaultValue	BOOL	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
sInactiveText	T_BA_StateText [▶ 123]	Textausgabe bei inaktivem Objekt.
sActiveText	T_BA_StateText [▶ 123]	Textausgabe bei aktivem Objekt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.3.6 FB_BA_SchedM



Der Funktionsbaustein FB_BA_SchedM repräsentiert ein Multi-State-Scheduler innerhalb der Projektstruktur von TF8040.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_BaseSched [▶ 261]

Illustration

```
FUNCTION_BLOCK FB_BA_SchedM EXTENDS FB_BA_BaseSched IMPLEMENTS I_BA_SchedM
VAR_OUTPUT
  nPresentValue      : UDINT;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    aStateText       : T_BA_StateTextArray;
    nDefaultValue    : UDINT := 1;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    nStateCount      : UDINT;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
nPresentValue	UDINT	Aktueller Wert für mehrstufige Ausgaben.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
aStateText	T_BA_StateTextArray [▶ 123]	Das Array dient der Deklaration der Statustexte eines Multi-State-Objektes.
nDefaultValue	UDINT	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.

VAR

Name	Typ	Beschreibung
nStateCount	UDINT	Zahl der Status eines Multi-State-Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.3.7 FB_BA_Trend



Der Funktionsbaustein FB_BA_Trend repräsentiert einen Trend innerhalb der Projektstruktur von TF8040.

Informationen zu vererbten Elementen

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

Illustration

```
FUNCTION_BLOCK FB_BA_Trend EXTENDS FB_BA_EventObject IMPLEMENTS I_BA_Trend
VAR_OUTPUT
  nRecordCount                : UDINT;
  nTotalRecordCount           : UDINT;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    nBufferSize               : UDINT := BA_Param.nTrend_BufferSize;
  {endregion}
  {region 'Variable Parameters'}
    aLogBuffer                : T_BA_TrendLogBuffer;
    bEnable                   : BOOL;
    stStartTime               : ST_BA_DateTime := BA_Param.stTrend_DefStartTime;
    stStopTime                : ST_BA_DateTime := BA_Param.stTrend_DefStopTime;
    bStopOnFull               : BOOL := BA_Param.bTrend_DefStopOnFull;
    nLogInterval              : UDINT := BA_Param.nTrend_DefLogInterval;
    nNotificationThreshold    : UDINT := BA_Param.nTrend_DefNotificationThreshold;
```

```
    eLoggingType              : E_BA_LoggingType := BA_Param.eTrend_DefLoggingType;
    stReferencedParam         : ST_BA_ObjectParameter;
  {endregion}
END_VAR
```

VAR_OUTPUT

Name	Typ	Beschreibung
nRecordCount	UDINT	Anzahl der Aufzeichnungen.
nTotalRecordCount	UDINT	Absolute Anzahl der Aufzeichnungen.

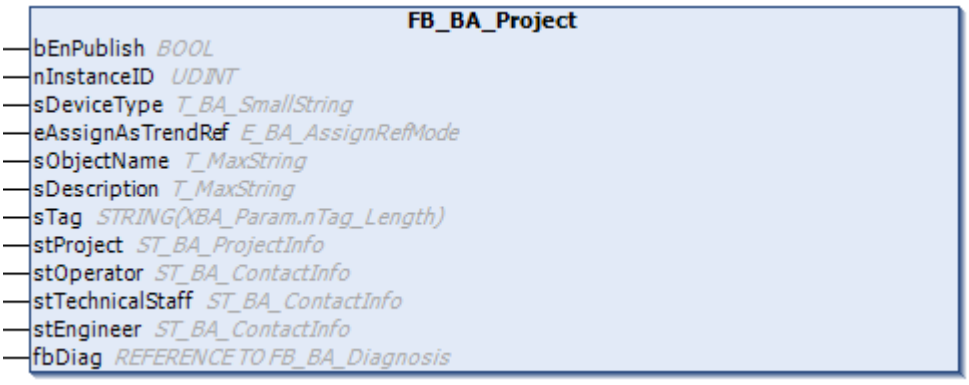
VAR_INPUT CONSTANT PERSISTENT

Name	Typ	Beschreibung
nBufferSize	UDINT	Größe des Puffers.
aLogBuffer	ST_BA_TrendEntry	Ringspeicher für Werte mit Zeitstempel.
bEnable	BOOL	Freigabe zur Aufzeichnung.
stStartTime	ST_BA_DateTime	Startzeit.
stStopTime	ST_BA_DateTime	Anhaltezeit.
bStopOnFull	BOOL	Ein TRUE stoppt die Aufzeichnung wenn der Puffer voll ist.
nLogInterval	UDINT	Intervall zum Speichern der Parameter.
nNotificationThreshold	UDINT	Grenzwert an dem Notifications ausgelöst werden.
eLoggingType	E_BA_LoggingType	Einstellung der Art des Speicherns.
stReferencedParam	ST_BA_ObjectParameter [▶ 122]	Objekt Parameter.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.3.8 FB_BA_Project



Der Funktionsbaustein bietet die Möglichkeit das Projekt mittels optionaler Informationen näher zu beschreiben.

Die angegebenen Informationen können von Clients (z.B. HMI oder Site Explorer) abgerufen und entsprechend aufbereitet (z.B. als Projektbezogene Informations-Übersicht zu Ansprechpartnern, etc.) werden.

Des Weiteren werden die allgemeinen Diagnose-Funktionen (siehe XBA_Globals.Diag [▶ 127]) referenziert.

Im Projekt darf dieser Baustein nur einmal instanziiert werden.

Die Instance ID entspricht immer der des BACnet Devices, auch wenn eine andere ID vorinitialisiert wurde.

Vererbungsgierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Syntax

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

Eingänge

Name	Typ	Beschreibung
stProject	ST_BA_ProjectInfo [▶ 119]	Weiterführende Informationen zum Projekt (Name, Beschreibung, Ort).
stOperator	ST_BA_ContactInfo [▶ 118]	Kontaktinformationen des Betreibers (Name, Telefonnummer, E-Mail-Adresse, Webseite).
stTechnicalStaff	ST_BA_ContactInfo [▶ 118]	Kontaktinformationen des technischen Personals (Name, Telefonnummer, E-Mail-Adresse, Webseite).
stEngineer	ST_BA_ContactInfo [▶ 118]	Kontaktinformationen des Integrators (Name, Telefonnummer, E-Mail-Adresse, Webseite).

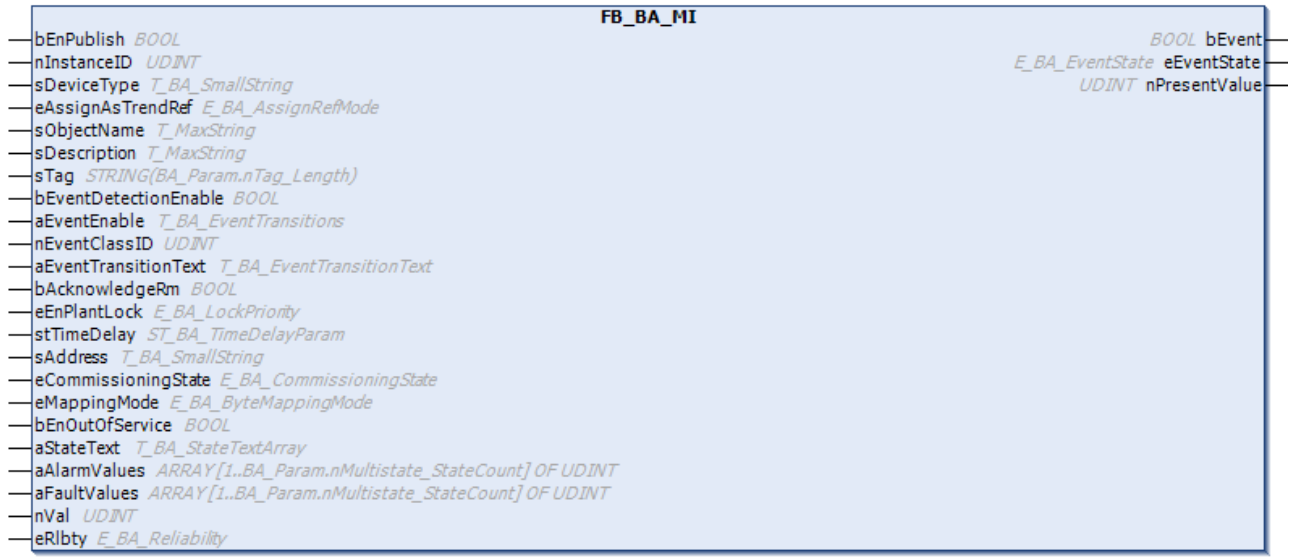
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.2.1.3.3.1.4 Multistate

Melden, Schalten und Verarbeiten von mehrstufigen Werten.

6.1.2.1.3.3.1.4.1 FB_BA_MI



Der Funktionsbaustein FB_BA_MI repräsentiert das Multi-State_Input-Objekt.

Das Multi-State_Input-Objekt spezifiziert einen Objekttyp, dessen Objekt-Properties extern sichtbare Charakteristiken eines Datenpunkts für mehrere binäre Eingänge repräsentieren. Das Objekt wird von physikalischen oder virtuellen Datenpunkten abgeleitet, z. B. den Grenzwertmeldungen mehrerer Analog-Eingaben oder Analogwerte oder von einer mathematischen Berechnung.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseMI [▶ 258]

Illustration

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

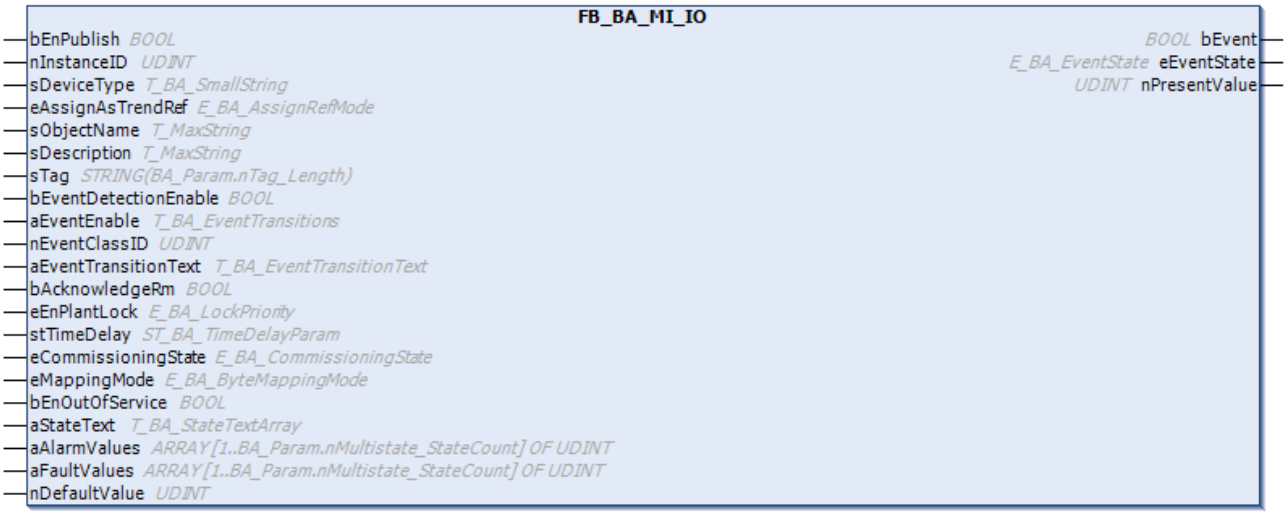
Eingänge

Name	Typ	Beschreibung
nVal	UDINT	Skalierter, analoger Eingangswert.
eRlbty	E_BA_Reliability	Verfügbarkeit bzw. Zuverlässigkeit eines Wertes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.4.2 FB_BA_MI_IO



Der Funktionsbaustein FB_BA_MI_IO repräsentiert das Multi-State_Input-Objekt.

Das Multi-State_Input-Objekt spezifiziert einen Objekttyp, dessen Objekt-Properties extern sichtbare Charakteristiken eines Datenpunkts für mehrere binäre Eingänge repräsentieren. Das Objekt wird von physikalischen oder virtuellen Datenpunkten abgeleitet, z. B. den Grenzwertmeldungen mehrerer Analog-Eingaben oder Analogwerte oder von einer mathematischen Berechnung.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseMI [▶ 258]

Illustration

```
FUNCTION_BLOCK FB_BA_MI_IO EXTENDS FB_BA_BaseMI IMPLEMENTS I_BA_RawMI
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    nDefaultValue          : UDINT := 1;
  {endregion}
END_VAR
VAR
  {region 'Raw I/O'}
    stRawVal    AT %I*     : ST_BA_Byte;
  {endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nDefaultValue	UDINT	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.

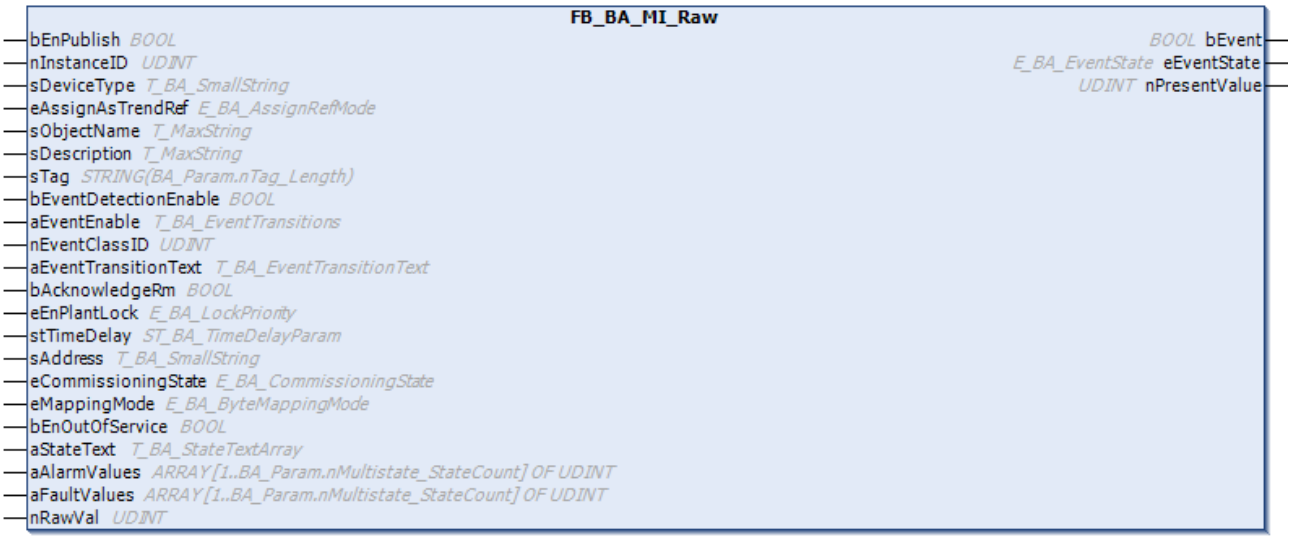
VAR

Name	Typ	Beschreibung
stRawVal	ST_BA_Byte	Strukturvariable zur Verknüpfung mit dem Prozessabbild.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.4.3 FB_BA_MI_Raw



Der Funktionsbaustein FB_BA_MI_Raw repräsentiert das Multi-State_Input-Objekt.

Das Multi-State_Input-Objekt spezifiziert einen Objekttyp, dessen Objekt-Properties extern sichtbare Charakteristiken eines Datenpunkts für mehrere binäre Eingänge repräsentieren. Das Objekt wird von physikalischen oder virtuellen Datenpunkten abgeleitet, z. B. den Grenzwertmeldungen mehrerer Analog-Eingaben oder Analogwerte oder von einer mathematischen Berechnung.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseMI [▶ 258]

Illustration

```
FUNCTION_BLOCK FB_BA_MI_Raw EXTENDS FB_BA_BaseMI
VAR_INPUT
  nRawVal    : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nRawVal	UDINT	Rohwert

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.4.4 FB_BA_MO



Der Funktionsbaustein FB_BA_MO repräsentiert das Multi-State_Output-Objekt.
Er besitzt ein Prioritäten-Array und ist somit kommandierbar (siehe Kommandierung von Objekten).

Das Multi-State_Output-Objekt spezifiziert einen Objekttyp, dessen Objekt-Properties extern sichtbare Charakteristiken eines Datenpunkts für mehrere binäre Ausgänge repräsentieren, verbunden durch eine beliebige Binärlogik. Das Objekt wird von physikalischen oder virtuellen Datenpunkten abgeleitet, z. B. für „active-/inactive“-Zustände mehrerer Binärausgänge oder den Wert eines Analogausgangs.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseMO [▶ 259]

Illustration

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

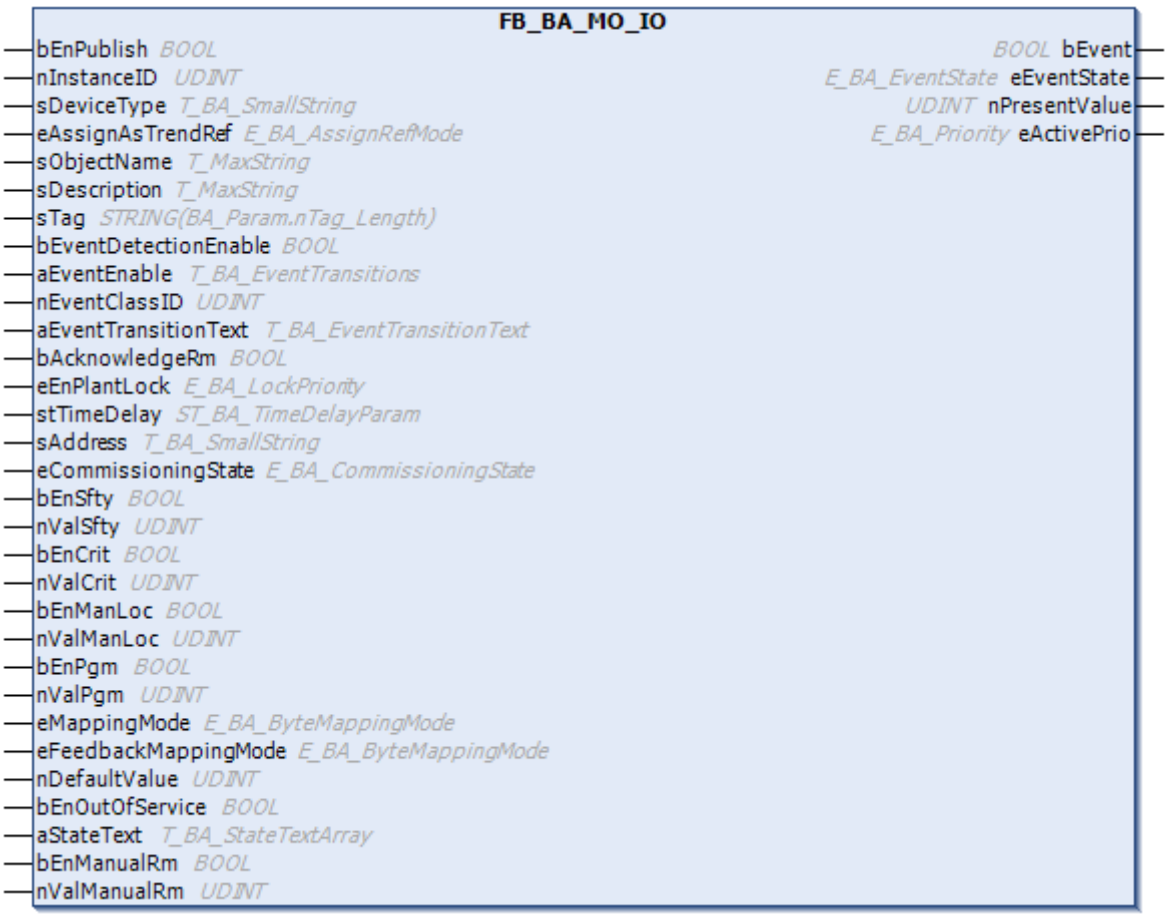
Eingänge

Name	Typ	Beschreibung
eRlbty	E_BA_Reliability	Verfügbarkeit bzw. Zuverlässigkeit eines Wertes.
bRawOvrrd	BOOL	An diese Variable kann die Rückmeldung eines in einer Klemme enthaltenen Schalters, für die mechanische Übersteuerung eines Ausgangs, angeschlossen werden.
bEnFdbck	BOOL	Variable zum Start der Überwachung der Rückmeldung am binären Ausgang.
nRawValFdbck	UDINT	Rohwert Rückmeldung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.4.5 FB_BA_MO_IO



Der Funktionsbaustein FB_BA_MO_IO repräsentiert das Multi-State_Output-Objekt.
Er besitzt ein Prioritäten-Array und ist somit kommandierbar (siehe Kommandierung von Objekten).

Das Multi-State_Output-Objekt spezifiziert einen Objekttyp, dessen Objekt-Properties extern sichtbare Charakteristiken eines Datenpunkts für mehrere binäre Ausgänge repräsentieren, verbunden durch eine beliebige Binärlogik. Das Objekt wird von physikalischen oder virtuellen Datenpunkten abgeleitet, z. B. für „active-/inactive“-Zustände mehrerer Binärausgänge oder den Wert eines Analogausgangs.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseMO [▶ 259]

Illustration

```
FUNCTION_BLOCK FB_BA_MO_IO EXTENDS FB_BA_BaseMO IMPLEMENTS I_BA_RawMO
VAR
  {region 'Raw I/O'}
    stRawVal            AT %Q*    : ST_BA_Byte;
    stRawValFeedback    AT %I*    : ST_BA_Byte;
  {endregion}
END_VAR
```

VAR

Name	Typ	Beschreibung
stRawVal	ST_BA_Byte	Rohwert zur Ausgabe an einer Busklemme.
stRawValFeedback	ST_BA_Byte	Rückmeldeeingang des Rohwertes (optional).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.4.6 FB_BA_MO_Raw



Der Funktionsbaustein FB_BA_MO_Raw repräsentiert das Multi-State_Output-Objekt.
Er besitzt ein Prioritäten-Array und ist somit kommandierbar (siehe Kommandierung von Objekten).

Das Multi-State_Output-Objekt spezifiziert einen Objekttyp, dessen Objekt-Properties extern sichtbare Charakteristiken eines Datenpunkts für mehrere binäre Ausgänge repräsentieren, verbunden durch eine beliebige Binärlogik. Das Objekt wird von physikalischen oder virtuellen Datenpunkten abgeleitet, z. B. für „active-/inactive“-Zustände mehrerer Binärausgänge oder den Wert eines Analogausgangs.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

FB_BA_BaseMO [▶ 259]

Illustration

```
FUNCTION_BLOCK FB_BA_MO_Raw EXTENDS FB_BA_BaseMO
VAR_INPUT
  bEnFeedback        : BOOL;
  nRawValFeedback    : UDINT;
END_VAR
VAR_OUTPUT
  nRawVal            : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnFdbck	BOOL	Variable zum Start der Überwachung der Rückmeldung am binären Ausgang.
nRawValFdbck	UDINT	Rohwert Rückmeldung.

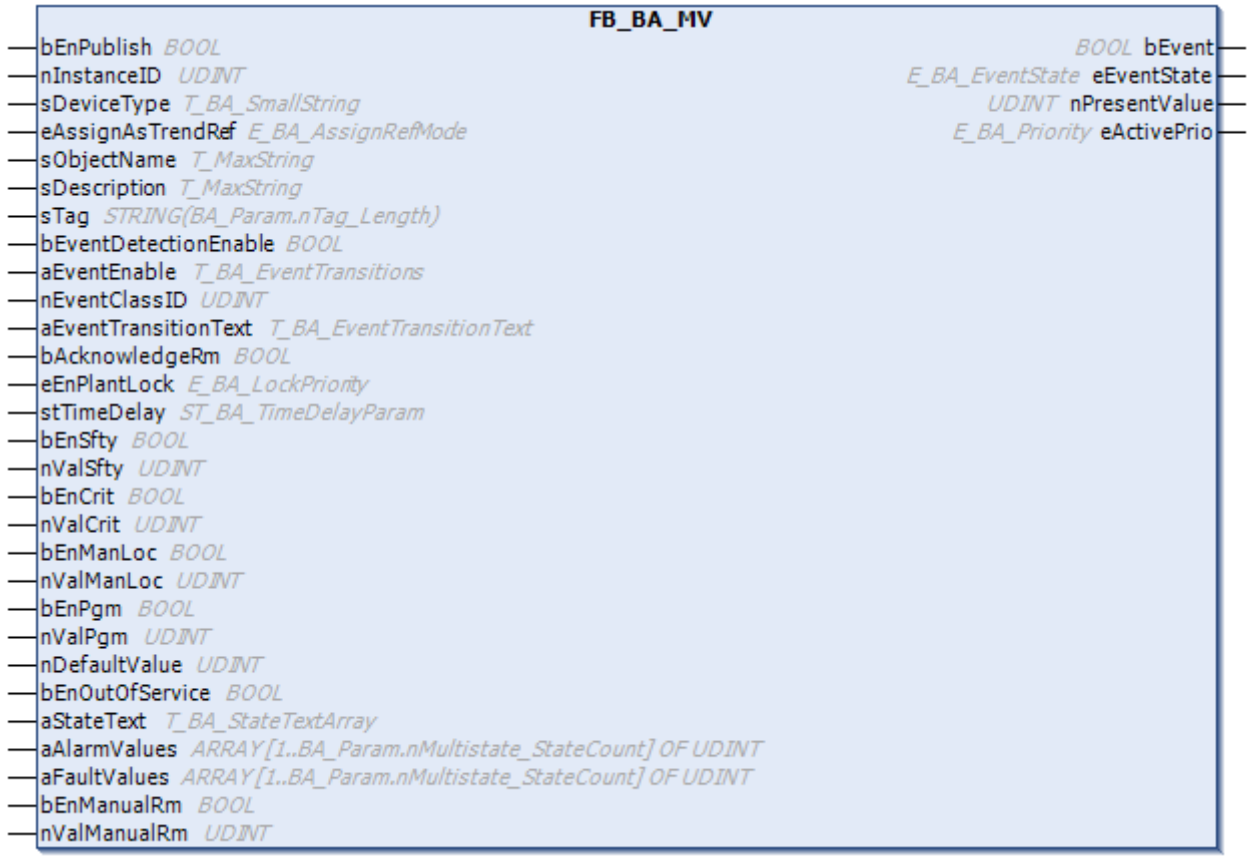
Ausgänge

Name	Typ	Beschreibung
nRawVal	UDINT	Die Variable dient der Verknüpfung des Rohwertes eines Objektes mit dem Prozessabbild der Ein- bzw. Ausgangsklemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.4.7 FB_BA_MV



Der Funktionsbaustein FB_BA_MV repräsentiert das Multi-State_Value-Objekt.
Er besitzt ein Prioritäten-Array und ist somit kommandierbar (siehe Kommandierung von Objekten).

Das Multi-State_Value-Objekt spezifiziert einen Objekttyp, dessen Properties extern sichtbare Charakteristitken eines virtuellen Datenpunkts für einen „Mehrstufigen Wert“ repräsentieren. Der Multi-State_Value hat im dazugehörigen Device-Objekt keine E-/A-Hardware, er ist im Speicher abgelegt.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

Illustration

```
FUNCTION_BLOCK FB_BA_MV EXTENDS FB_BA_EventObjectEx IMPLEMENTS I_BA_MultistatePrioObject, I_BA_AnyVa
lue
VAR_INPUT
  bEnSfty              : BOOL;
  nValSfty             : UDINT := 1;
  bEnCrit              : BOOL;
  nValCrit             : UDINT := 1;
  bEnManLoc            : BOOL;
  nValManLoc           : UDINT := 1;
  bEnPgm               : BOOL;
  nValPgm              : UDINT := 1;
END_VAR
VAR_OUTPUT
  nPresentValue        : UDINT := 1;
  eActivePrio          : E_BA_Priority;
END_VAR
```

```
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    nDefaultValue      : UDINT:= 1;
    bEnOutOfService    : BOOL;
  {endregion}
  {region 'Fixed Parameters'}
    aStateText         : T_BA_StateTextArray;
    aAlarmValues       : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
    aFaultValues       : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
  {endregion}
  {region 'Operational Parameters'}
    bEnManualRm        : BOOL;
    nValManualRm       : UDINT:= 1;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    nStateCount        : UDINT;
  {endregion}
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe der Priorität „Safety“.
nValSfty	UDINT	Analoger Wert für die Priorität „Safety“.
bEnCrit	BOOL	Freigabe der Priorität „Critical“.
nValCrit	UDINT	Analoger Wert für die Priorität „Critical“.
bEnManLoc	BOOL	Freigabe der Priorität „Manual Local“.
nValManLoc	UDINT	Analoger Wert für die Priorität „Manual Local“.
bEnPgm	BOOL	Freigabe der Priorität „Program“.
nValPgm	UDINT	Analoger Wert für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
nPresentValue	UDINT	Aktueller Wert für mehrstufige Ausgaben.
eActivePrio	E_BA_Priority [▶ 112]	Aktive Priorität

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nDefaultValue	UDINT	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
aStateText	T_BA_StateTextArray [▶ 123]	Das Array dient der Deklaration der Statustexte eines Multi-State-Objektes.
aAlarmValues	ARRAY [1..BA_Param.nMultistate_StateCount] OF UDINT	Innerhalb des Arrays werden die Zustände des Multi-State-Objektes beschrieben bei denen ein Alarm vorliegt.
aFaultValues	ARRAY [1..BA_Param.nMultistate_StateCount] OF UDINT	Innerhalb des Arrays werden die Zustände des Multi-State-Objektes beschrieben, bei denen ein Fehler vorliegt.
bEnManualRm	BOOL	Freigabe der Priorität „Manual Remote“.
nValManualRm	UDINT	Variable zum Schreiben eines Wertes auf die Priorität „Manual Remote“.

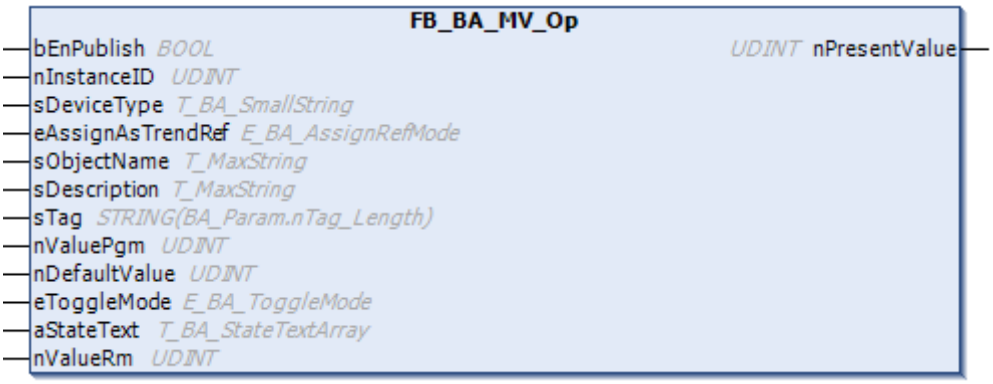
VAR

Name	Typ	Beschreibung
nStateCount	UDINT	Zahl der Status eines Multi-State-Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.4.8 FB_BA_MV_Op



Der Funktionsbaustein FB_BA_MV_Op repräsentiert ein Multi-State-Value-Objekt.
Es dient der Anzeige oder Eingaben eines Multi-State-Wertes.

Wenn an der Eingangsvariable *nValuePgm* etwas angeschlossen ist, dann erkennt das Objekt automatisch, dass es der Anzeige eines Wertes dient. Andernfalls dient es der Eingabe eines Multi-State-Wertes.

Das Multi-State_Value-Objekt spezifiziert einen Objekttyp, dessen Properties extern sichtbare Charakteristiken eines virtuellen Datenpunkts für einen „Mehrstufigen Wert“ repräsentieren. Der Multi-State_Value hat im dazugehörigen Device-Objekt keine E-/A-Hardware, er ist im Speicher abgelegt.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK FB_BA_MV_Op EXTENDS FB_BA_Object IMPLEMENTS I_BA_MultistateOpObject, I_BA_AnyValue
VAR_INPUT
  nValuePgm          : UDINT;
END_VAR
VAR_OUTPUT
  nPresentValue      : UDINT := 1;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    nDefaultValue    : UDINT := 1;
    eToggleMode      : E_BA_ToggleMode := E_BA_ToggleMode.eSwitch;
  {endregion}
  {region 'Fixed Parameters'}
    aStateText       : T_BA_StateTextArray;
  {endregion}
  {region 'Operational Parameters'}
    nValueRm         : UDINT := 1;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    nStateCount      : UDINT;
  {endregion}
```

```
END_VAR
VAR
  {region 'Interface'}
    eValueSource     : E_BA_ProcessSignalSource;
  {endregion}
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nValuePgm	UDINT	Wert eines Multi-State-Objektes für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
nPresentValue	UDINT	Aktueller Wert für mehrstufige Ausgaben.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nDefaultValue	UDINT	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
eToggleMode	E_BA_ToggleMode	Enumeration zur Festlegung, wie der Ausgabewert *bPresentValue* eines Objektes in Abhängigkeit des Eingangs *bValuePgm* erzeugt wird.
aStateText	T_BA_StateTextArray [▶ 123]	Das Array dient der Deklaration der Statustexte eines Multi-State-Objektes.
nValueRm	UDINT	Variable zum Beschreiben eines Objektes von der HMI.

VAR

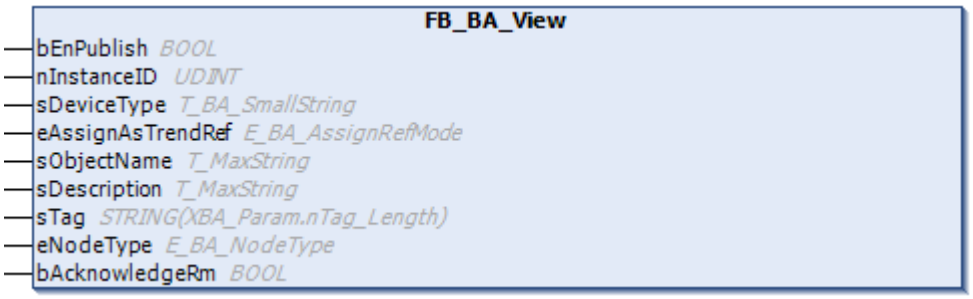
Name	Typ	Beschreibung
nStateCount	UDINT	Zahl der Status eines Multi-State-Objektes.
eValueSource	E_BA_ProcessSignalSource [▶ 115]	Die Variable zeigt an, ob ein Objekt des Typs FB_BA_..._OP als Anzeige- oder Eingabeobjekt dient. *eVarInput* = 1 Das Objekt dient der Anzeige eines Wertes. Der Wert wird dem Objekt an einem Eingang innerhalb der SPS übergeben. *eParameter* = 2 Das Objekt dient der Eingabe eines Parameters, welcher von einem BACnet-Client bzw. der TwinCAT HMI verändert werden kann.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.5 Structurize

6.1.2.1.3.3.1.5.1 FB_BA_View



Der Funktionsbaustein FB_BA_View dient der Erzeugung eines Ordners innerhalb der Projektstruktur.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK FB_BA_View EXTENDS FB_BA_Object IMPLEMENTS I_BA_View
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    eNodeType           : E_BA_NodeType := E_BA_NodeType.Automatic;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Operational Parameters'}
    bAcknowledgeRm      : BOOL;
  {endregion}
END_VAR
VAR
  {region 'SubInit'}
    eDPADMode           : E_BA_DPADMode := E_BA_DPADMode.Undefined;
  {endregion}
  {region 'Events'}
    fbActiveEvents      : FB_BA_EventIndicator;
  {endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eNodeType	E_BA_NodeType [▶ 113]	Mit der Variablen *NodeType* wird die Ordnerebene innerhalb der Projektstruktur beschrieben.

Eingänge CONSTANT

Name	Typ	Beschreibung
bAcknowledgeRm	BOOL	Eingang zur lokalen Quittierung der Ereignisse eines Objektes.

VAR

Name	Typ	Beschreibung
eDPADMode	E_BA_DPADMode [▶ 103]	Mit der Variable wird die Projektstruktur integrierte Erzeugung des Benutzer-Adress-Schlüssels beeinflusst. Zur Erzeugung der Projektstruktur ist es zwingend erforderlich, eine durchgängige Parent / Child Beziehung innerhalb des TwinCAT-Projektes herzustellen. Das heißt, dass allen Instanzen des Funktionsbausteins *FB_BA_View* mitgeteilt werden muss, von wem sie abstammen. Die Zahl der Ebenen innerhalb des TwinCAT Programms bzw. des Symbolpfades weicht jedoch häufig von der Zahl der Ebenen im Benutzer-Adress-Schlüssel ab. Häufig gibt es im Symbolpfad es TwinCAT-Projektes mehr Ebenen als im Benutzer-Adress-Schlüssel. Es kann deshalb erforderlich sein für die generische Erzeugung der Objektnamen und der Beschreibungstexten einige Ebenen aus der Verkettung der Texte auszuschließen. Mit der Enumeration *eDPADMode* ist dieses möglich.
fbActiveEvents	FB_BA_EventIndicator	Gibt Informationen über Events aller Objekte eines View Objekts.

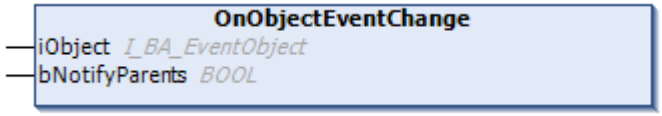
Methoden

Name	Beschreibung
OnObjectEventChange [▶ 247]	Änderung eines Ereignisses in einem Unter-Objekt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.5.1.1 OnObjectEventChange



Die Methode wird aufgerufen, wenn sich das Event eines untergeordneten Objektes ändert.

Der Anwender kann in dieser Methode auf die Änderungen reagieren.

Syntax

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

Eingänge

Name	Typ	Beschreibung
iObject	I_BA_EventObject	Schnittstelle zum Kontext-Objekt. [▶ 269]
bNotifyParents	BOOL	Freigabe für das weiterleiten des Events an das nächste übergeordnete Objekt. Diese Freigabe muss an den Basis Funktionsbaustein übergeben werden. Beispiel: `SUPER^.OnObjectEventChange(iObject, TRUE);`

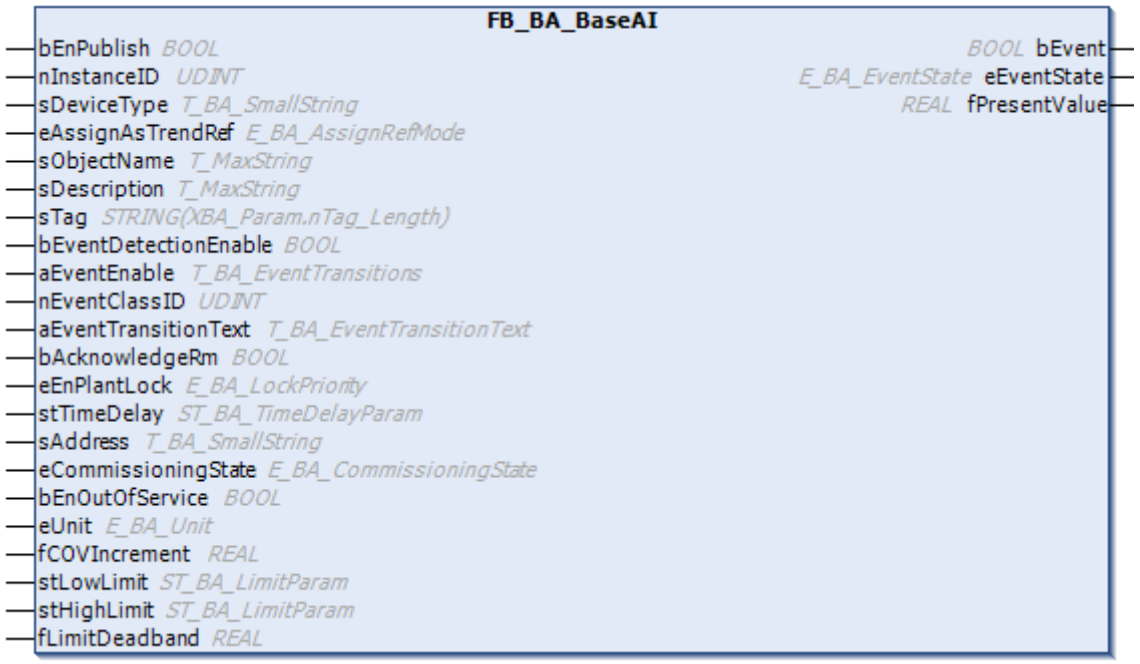
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6 Base

Diese Objekte bilden die Basis der vorherig beschriebenen Bausteine. Sie beerben die lokalen Funktionsbausteine.

6.1.2.1.3.3.1.6.1 FB_BA_BaseAI



Der Funktionsbaustein FB_BA_BaseAI ist die Basis aller analogen Eingangsobjekte.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseAI EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_AnalogInObject, I_
BA_AnyValue
VAR_OUTPUT
  fPresentValue        : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    fResolution        : REAL := BA_Param.fInput_DefResolution;
    fScaleOffset       : REAL := BA_Param.fInput_DefScaleOffset;
    bEnOutOfService    : BOOL;
    eUnit              : E_BA_Unit := E_BA_Unit.Invalid;
    fCOVIncrement      : REAL := BA_Param.fDefCOVIncrement;
    stLowLimit         : ST_BA_LimitParam;
    stHighLimit        : ST_BA_LimitParam;
    fLimitDeadband     : REAL := BA_Param.fDefLimitDeadband;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fResolution [▶ 128]	REAL	Auflösung eines Analogsignals für die Skalierung eines Messwertes.
fScaleOffset [▶ 128]	REAL	Skalierungs-Offset
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
eUnit	E_BA_Unit	Einheit des Ein- bzw. Ausgabewerts eines Analogobjektes.
fCOVIncrement	REAL	Die Variable gibt die Mindeständerung des Present-Value an, die dazu führt, dass eine COV-Notification an die abonnierten COV-Clients ausgegeben wird. Diese Eigenschaft ist erforderlich, wenn die COV-Berichterstattung von diesem Objekt unterstützt wird.
stLowLimit	ST_BA_LimitParam [▶ 122]	Parametrierung der unteren Grenzwertüberwachung eines analogen Objektes. Die Variable *bEnable* muss TRUE sein, damit die Grenzwertüberwachung freigegeben ist. Mit der Variablen *fValue* wird der untere Grenzwert parametriert.
stHighLimit	ST_BA_LimitParam [▶ 122]	Parametrierung der oberen Grenzwertüberwachung eines analogen Objektes. Die Variable *bEnable* muss TRUE sein, damit die Grenzwertüberwachung freigegeben ist. Mit der Variablen *fValue* wird der obere Grenzwert parametriert.
fLimitDeadband	REAL	Totband bzw. Hysterese für die Werte *HighLimit* und *LowLimit* der Grenzwertüberwachung eines analogen Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.2 FB_BA_BaseAO



Der Funktionsbaustein FB_BA_BaseAO repräsentiert das Objekt eines analogen Ausgangs. Er ist die Basis aller Analogausgänge.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseAO EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_AnalogOutObject, I
_BA_AnyValue
VAR_INPUT
  bEnSfty              : BOOL;
  fValSfty             : REAL;
  bEnCrit              : BOOL;
  fValCrit             : REAL;
  bEnManLoc            : BOOL;
  fValManLoc           : REAL;
  bEnPgm               : BOOL;
  fValPgm              : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue        : REAL;
```

```
  eActivePrio          : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    fResolution        : REAL := BA_Param.fOutput_DefResolution;
    fScaleOffset       : REAL := BA_Param.fOutput_DefScaleOffset;
    fDefaultValue      : REAL;
    bEnOutOfService    : BOOL;
    eUnit              : E_BA_Unit := E_BA_Unit.Invalid;
    fCOVIncrement      : REAL := BA_Param.fDefCOVIncrement;
    stLowLimit         : ST_BA_LimitParam;
    stHighLimit        : ST_BA_LimitParam;
    fLimitDeadband     : REAL := BA_Param.fDefLimitDeadband;
  {endregion}
  {region 'Operational Parameters'}
    bEnManualRm        : BOOL;
    fValManualRm       : REAL;
  {endregion}
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe der Priorität „Safety“.
fValSfty	REAL	Analoger Wert für die Priorität „Safety“.
bEnCrit	BOOL	Freigabe der Priorität „Critical“.
fValCrit	REAL	Analoger Wert für die Priorität „Critical“.
bEnManLoc	BOOL	Freigabe der Priorität „Manual Local“.
fValManLoc	REAL	Analoger Wert für die Priorität „Manual Local“.
bEnPgm	BOOL	Freigabe der Priorität „Program“.
fValPgm	REAL	Analoger Wert für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller analoger Ausgabewert des Objektes.
eActivePrio	E_BA_Priority [▶ 112]	Aktive Priorität

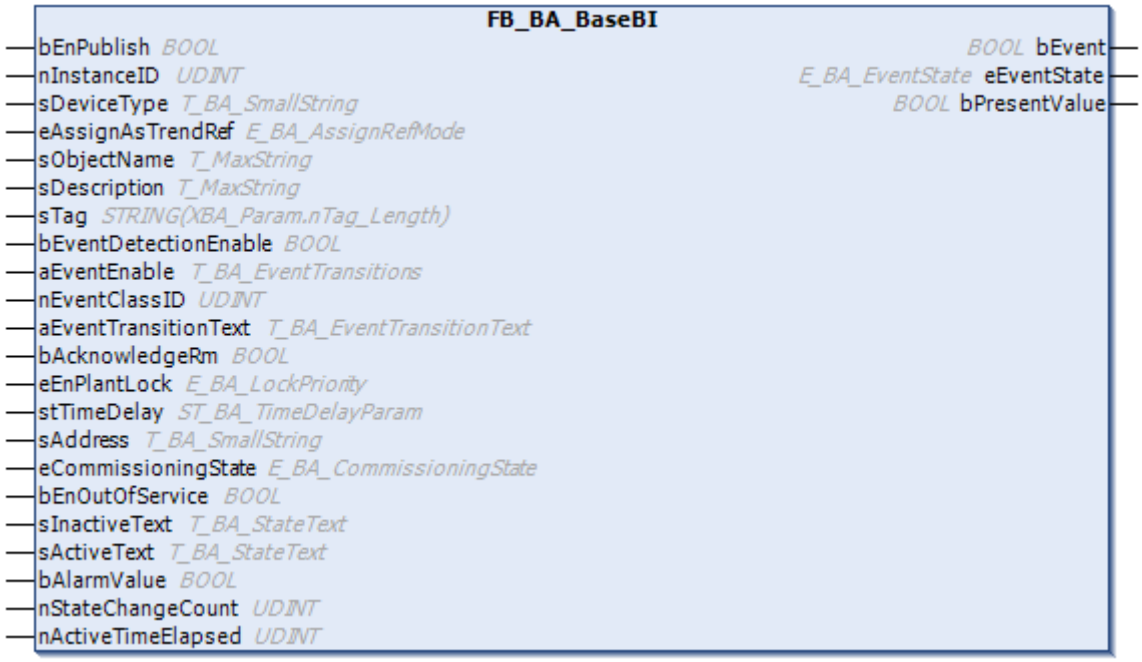
Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fResolution [▶ 128]	REAL	Auflösung eines Analogsignals für die Skalierung eines Messwertes.
fScaleOffset [▶ 128]	REAL	Skalierungs-Offset
fDefaultValue	REAL	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
eUnit	E_BA_Unit	Einheit des Ein- bzw. Ausgabewerts eines Analogobjektes.
fCOVIncrement	REAL	Die Variable gibt die Mindeständerung des Present-Value an, die dazu führt, dass eine COV-Notification an die abonnierten COV-Clients ausgegeben wird. Diese Eigenschaft ist erforderlich, wenn die COV-Berichterstattung von diesem Objekt unterstützt wird.
stLowLimit	ST_BA_LimitParam [▶ 122]	Parametrierung der unteren Grenzwertüberwachung eines analogen Objektes. Die Variable *bEnable* muss TRUE sein, damit die Grenzwertüberwachung freigegeben ist. Mit der Variablen *fValue* wird der untere Grenzwert parametriert.
stHighLimit	ST_BA_LimitParam [▶ 122]	Parametrierung der oberen Grenzwertüberwachung eines analogen Objektes. Die Variable *bEnable* muss TRUE sein, damit die Grenzwertüberwachung freigegeben ist. Mit der Variablen *fValue* wird der obere Grenzwert parametriert.
fLimitDeadband	REAL	Totband bzw. Hysterese für die Werte *HighLimit* und *LowLimit* der Grenzwertüberwachung eines analogen Objektes.
bEnManualRm	BOOL	Freigabe der Priorität „Manual Remote“.
fValManualRm	REAL	Analoger Wert für die Priorität „Manual Remote“.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.3 FB_BA_BaseBI



Der Funktionsbaustein FB_BA_BaseBI generiert ein binäres Input-Objekt.
Er ist die Basis aller binären Eingangsobjekte.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseBI EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_BinaryInObject, I_
BA_AnyValue
VAR_OUTPUT
  bPresentValue                : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    bEnOutOfService            : BOOL;
    sInactiveText              : T_BA_StateText;
    sActiveText                : T_BA_StateText;
    bAlarmValue                : BOOL := TRUE;
    ePolarity                  : E_BA_Polarity := E_BA_Polarity.eNormal;
    nStateChangeCount          : UDINT;
    nActiveTimeElapsed         : UDINT;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    stStateChangeTime          : ST_BA_DateTime;
    stStateChangeResetPoint    : ST_BA_DateTime;
    stActiveTimeResetPoint     : ST_BA_DateTime;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
sInactiveText	T_BA_StateText [▶ 123]	Textausgabe bei inaktivem Objekt.
sActiveText	T_BA_StateText [▶ 123]	Textausgabe bei aktivem Objekt.
bAlarmValue	BOOL	Wert im Falle eines Alarms.
ePolarity	E_BA_Polarity	Die Polarität beschreibt die Abhängigkeit zwischen dem Wert, der aus der Auswertung des Priority_Arrays entsteht und dem Wert, der am Ausgang der Steuerung ausgegeben wird. Ist die Polarität normal dann wird das Ergebnis des Priority_Array unmittelbar an den Ausgang der Steuerung weitergeleitet. Bei der Polarität reverse wird der Ausgang negiert.
nStateChangeCount	UDINT	Die Variable gibt an, wie oft sich der Zustand des *Present_Value* seit dem Datum und des letzten Reset geändert hat.
nActiveTimeElapsed	UDINT	Zeitangabe [s], in der der *Present_Value* des Objektes den Wert ACTIVE hatte. Die Zeit gilt ab dem letzten Rücksetzen durch das Property *Time_Of_Active_Time_Reset*.

VAR

Name	Typ	Beschreibung
stStateChangeTime	ST_BA_DateTime	Dieses Property zeigt die Uhrzeit der letzten Zustandsänderung an. Die Zustandsänderung bezieht sich auf den *Present_Value* des Objektes.
stStateChangeResetPoint	ST_BA_DateTime	Zeigt Datum und Uhrzeit ab der das Zählen der Statusänderungen begonnen hat.
stActiveTimeResetPoint	ST_BA_DateTime	Gibt den Zeitpunkt an, an dem die Aufzeichnung der Einschaltzeitzeiten des Objektes begonnen hat.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.4 FB_BA_BaseBO



Der Funktionsbaustein FB_BA_BaseBO repräsentiert ein binäres Ausgangsobjekt.
Er ist die Basis für alle anderen Binärausgange.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseBO EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_BinaryOutObject, I
_BA_AnyValue
VAR_INPUT
  bEnSfty                       : BOOL;
  bValSfty                      : BOOL;
  bEnCrit                       : BOOL;
  bValCrit                      : BOOL;
  bEnManLoc                     : BOOL;
  bValManLoc                    : BOOL;
  bEnPgm                        : BOOL;
  bValPgm                       : BOOL;
END_VAR
```

```
VAR_OUTPUT
  bPresentValue                : BOOL;
  eActivePrio                  : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    nMinimumOffTime            : UDINT;
    nMinimumOnTime             : UDINT;
    bDefaultValue              : BOOL;
    bEnOutOfService            : BOOL;
    sInactiveText              : T_BA_StateText;
    sActiveText                : T_BA_StateText;
    ePolarity                  : E_BA_Polarity := E_BA_Polarity.eNormal;
    nStateChangeCount          : UDINT;
    nActiveTimeElapsed         : UDINT;
  {endregion}
  {region 'Operational Parameters'}
    bEnManualRm                : BOOL;
    bValManualRm               : BOOL;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    stStateChangeTime          : ST_BA_DateTime;
    stStateChangeResetPoint    : ST_BA_DateTime;
    stActiveTimeResetPoint     : ST_BA_DateTime;
  {endregion}
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe der Priorität „Safety“.
bValSfty	BOOL	Binärer Wert für die Priorität „Safety“.
bEnCrit	BOOL	Freigabe der Priorität „Critical“.
bValCrit	BOOL	Binärer Wert für die Priorität „Critical“.
bEnManLoc	BOOL	Freigabe der Priorität „Manual Local“.
bValManLoc	BOOL	BinärerWert für die Priorität „Manual Local“.
bEnPgm	BOOL	Freigabe der Priorität „Program“.
bValPgm	BOOL	Binärer Wert für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller binärer Ausgabewert des Objektes.
eActivePrio	E_BA_Priority [▶ 112]	Aktive Priorität

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nMinimumOffTime	UDINT	Minimale Zeit [s] in der der Present_Value im Zustand INACTIVE verbleiben soll, nachdem ein Schreibvorgang auf Present_Value, den Zustand INACTIVE angenommen hat. Damit kann ein Schutz vor zu schnellem erneutem Anlaufen realisiert werden.
nMinimumOnTime	UDINT	Minimale Zeit [s] in der der Present_Value im Zustand ACTIVE verbleiben soll, nachdem ein Schreibvorgang auf Present_Value, den Zustand ACTIVE angenommen hat. Damit kann ein Schutz vor verfrühtem, erneutem Ausschalten realisiert werden
bDefaultValue	BOOL	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
sInactiveText	T_BA_StateText [▶ 123]	Textausgabe bei inaktivem Objekt.
sActiveText	T_BA_StateText [▶ 123]	Textausgabe bei aktivem Objekt.
ePolarity	E_BA_Polarity	Die Polarität beschreibt die Abhängigkeit zwischen dem Wert, der aus der Auswertung des Priority_Arrays entsteht und dem Wert, der am Ausgang der Steuerung ausgegeben wird. Ist die Polarität normal dann wird das Ergebnis des Priority_Array unmittelbar an den Ausgang der Steuerung weitergeleitet. Bei der Polarität reverse wird der Ausgang negiert.
nStateChangeCount	UDINT	Die Variable gibt an, wie oft sich der Zustand des *Present_Value* seit dem Datum und des letzten Reset geändert hat.
nActiveTimeElapsed	UDINT	Zeitangabe [s], in der der *Present_Value* des Objektes den Wert ACTIVE hatte. Die Zeit gilt ab dem letzten Rücksetzen durch das Property *Time_Of_Active_Time_Reset*.
bValManualRm	BOOL	Binärer Wert für die Priorität „Manual Remote“.
bEnManualRm	BOOL	Freigabe der Priorität „Manual Remote“.

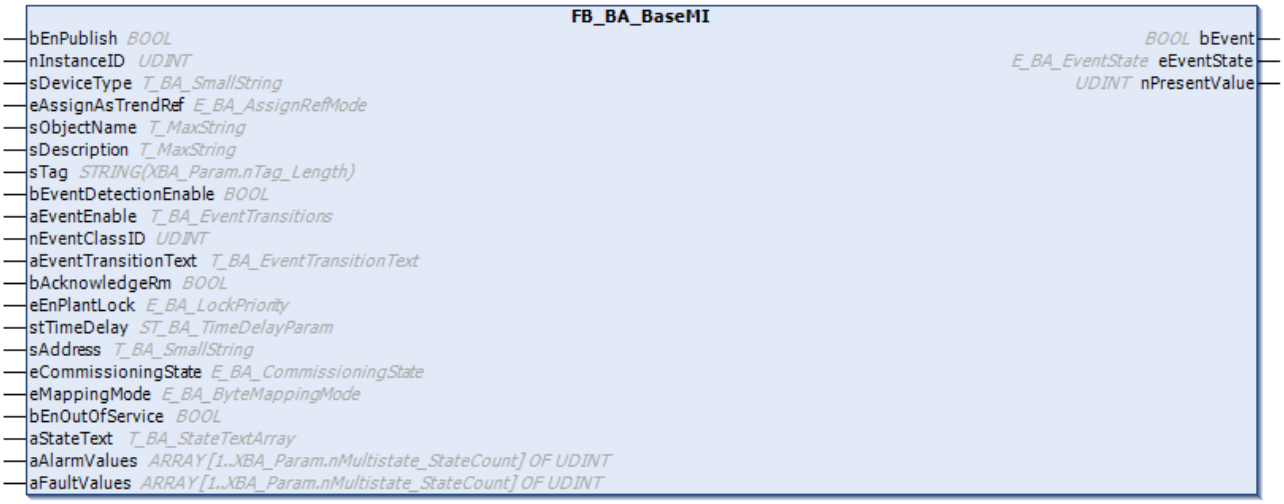
VAR

Name	Typ	Beschreibung
stStateChangeTime	ST_BA_DateTime	Dieses Property zeigt die Uhrzeit der letzten Zustandsänderung an. Die Zustandsänderung bezieht sich auf den *Present_Value* des Objektes.
stStateChangeResetPoint	ST_BA_DateTime	Zeigt Datum und Uhrzeit ab der das Zählen der Statusänderungen begonnen hat.
stActiveTimeResetPoint	ST_BA_DateTime	Gibt den Zeitpunkt an, an dem die Aufzeichnung der Einschaltzeitzeiten des Objektes begonnen hat.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.5 FB_BA_BaseMI



Der Funktionsbaustein FB_BA_BaseMI generiert ein Multistate-Input-Objekt.
Er ist die Basis aller Multistate-Input-Objekte.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseMI EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_MultistateInObject
, I_BA_AnyValue
VAR_OUTPUT
  nPresentValue        : UDINT := 1;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    eMappingMode       : E_BA_ByteMappingMode := E_BA_ByteMappingMode.eIndex1N;
    bEnOutOfService    : BOOL;
  {endregion}
  {region 'Fixed Parameters'}
    aStateText         : T_BA_StateTextArray;
    aAlarmValues       : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
    aFaultValues       : ARRAY[1 .. BA_Param.nMultistate_StateCount] OF UDINT;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    nStateCount        : UDINT;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
nPresentValue	UDINT	Analoger Ausgangswert.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eMappingMode	E_BA_ByteMappingMode	Modus zur Konfiguration der Klemmenverknüpfung.
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
aStateText	T_BA_StateTextArray [▶ 123]	Das Array dient der Deklaration der Statustexte eines Multi-State-Objektes.
aAlarmValues	ARRAY [1..BA_Param.nMultistate_StateCount] OF UDINT	Innerhalb des Arrays werden die Zustände des Multi-State-Objektes beschrieben bei denen ein Alarm vorliegt.
aFaultValues	ARRAY [1..BA_Param.nMultistate_StateCount] OF UDINT	Innerhalb des Arrays werden die Zustände des Multi-State-Objektes beschrieben, bei denen ein Fehler vorliegt.

VAR

Name	Typ	Beschreibung
nStateCount	UDINT	Zahl der Status eines Multi-State-Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.6 FB_BA_BaseMO



Der Funktionsbaustein FB_BA_BaseMO repräsentiert einen mehrstufigen Ausgang.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

FB_BA_ComEventObject [▶ 265]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseMO EXTENDS FB_BA_ComEventObject IMPLEMENTS I_BA_MultistateOutObjec
t, I_BA_AnyValue
VAR_INPUT
  bEnSfty                   : BOOL;
  nValSfty                  : UDINT := 1;
  bEnCrit                   : BOOL;
  nValCrit                  : UDINT := 1;
  bEnManLoc                 : BOOL;
  nValManLoc                : UDINT := 1;
  bEnPgm                    : BOOL;
  nValPgm                   : UDINT := 1;
END_VAR
VAR_OUTPUT
  nPresentValue             : UDINT := 1;
  eActivePrio               : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    eMappingMode            : E_BA_ByteMappingMode := E_BA_ByteMappingMode.eIndex1N;
    eFeedbackMappingMode    : E_BA_ByteMappingMode := E_BA_ByteMappingMode.eIndex1N;
    nDefaultValue           : UDINT := 1;
    bEnOutOfService         : BOOL;
  {endregion}
  {region 'Fixed Parameters'}
    aStateText              : T_BA_StateTextArray;
  {endregion}
  {region 'Operational Parameters'}
    bEnManualRm             : BOOL;
    nValManualRm            : UDINT := 1;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    nStateCount             : UDINT;
  {endregion}
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe der Priorität „Safety“.
nValSfty	UDINT	Analoger Wert für die Priorität „Safety“.
nValCrit	UDINT	Analoger Wert für die Priorität „Critical“.
bEnCrit	BOOL	Freigabe der Priorität „Critical“.
bEnManLoc	BOOL	Freigabe der Priorität „Manual Local“.
nValManLoc	UDINT	Analoger Wert für die Priorität „Manual Local“.
bEnPgm	BOOL	Freigabe der Priorität „Program“.
nValPgm	UDINT	Analoger Wert für die Priorität „Program“.

Ausgänge

Name	Typ	Beschreibung
nPresentValue	UDINT	Analoger Ausgangswert.
eActivePrio	E_BA_Priority [▶ 112]	Aktive Priorität

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eMappingMode	E_BA_ByteMappingMode	Modus zur Konfiguration der Klemmenverknüpfung.
eFeedbackMappingMode	E_BA_ByteMappingMode	Struktur zum Mapping der Rückmeldeeingänge.
nDefaultValue	UDINT	Wert der angenommen wird, wenn alle 16 Prioritäten des Priority_Array eines kommandierbaren Objektes keinen Eintrag haben, bzw. NULL sind.
bEnOutOfService	BOOL	Diese Variable setzt ein Objekt außer Betrieb. Es ist damit OutOfService.
aStateText	T_BA_StateTextArray [▶ 123]	Das Array dient der Deklaration der Statustexte eines Multi-State-Objektes.
bEnManualRm	BOOL	Freigabe der Priorität „Manual Remote“.
nValManualRm	UDINT	Variable zum Schreiben eines Wertes auf die Priorität „Manual Remote“.

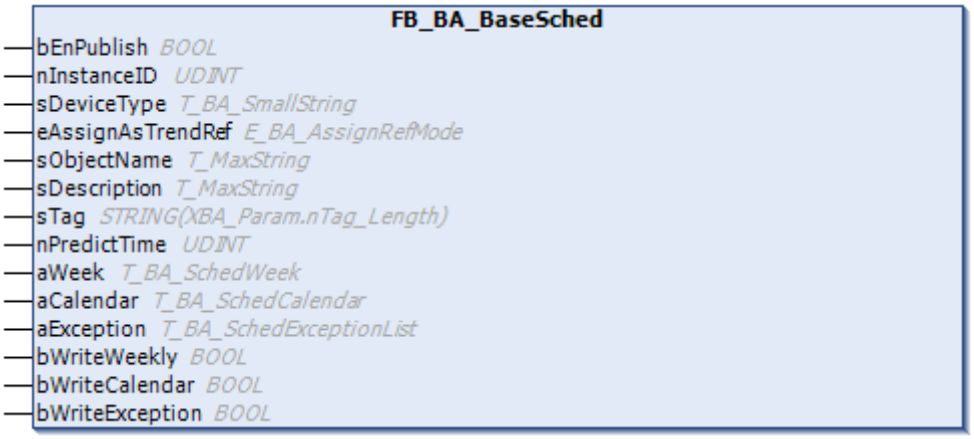
VAR

Name	Typ	Beschreibung
nStateCount	UDINT	Zahl der Status eines Multi-State-Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.7 FB_BA_BaseSched



Der Funktionsbaustein FB_BA_BaseSched bildet die Basis für Scheduler-Bausteine.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseSched EXTENDS FB_BA_Object IMPLEMENTS I_BA_BaseSched
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    nPredictTime       : UDINT;
    aWeek              : T_BA_SchedWeek;
    aCalendar          : T_BA_SchedCalendar;
    aException         : T_BA_SchedExceptionList;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Variable Parameters'}
    bWriteWeekly       : BOOL;
    bWriteException    : BOOL;
  {endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nPredictTime	UDINT	Berechneter Zeitwert.
aWeek	T_BA_SchedWeek [▶ 124]	Wochenzeitschaltplan.
aCalendar	T_BA_SchedCalendar [▶ 124]	Kalender.
aException	T_BA_SchedExceptionList [▶ 124]	Liste von Ausnahmebedingungen.

Eingänge CONSTANT

Name	Typ	Beschreibung
bWriteWeekly	BOOL	Schreibt wöchentlich.
bWriteException	BOOL	Schreibt zeitliche Ausnahmen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.8 FB_BA_BaseStateMV



Der Funktionsbaustein FB_BAStateMV ermöglicht das Anzeigen von Zuständen. Jedes referenzierte Objekt wird dabei als *Zustand* gesehen.
Er wird als *Multistate Objekt* dargestellt.

Die Zustandstexte dieses *Multistate-Objekts* sind automatisch die Namen aller kommandierten Referenzen.

Der PresentValue ist automatisch der höchste aktive Zustand, bzw. Referenz.

Abstract

Der FB dient als Basis (*ABSTRACT*) zum Bereitstellen von beschriebenen Funktionalitäten für erbende FB's.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseStateMV EXTENDS FB_BA_Object IMPLEMENTS I_BA_MultistateObject
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    eStateTextDeterminationMode    : E_BA_NodeTypeTarget := E_BA_NodeTypeTarget.eFunction;
    aStateText                     : T_BA_StateTextArray;
  {endregion}
END_VAR
VAR
  {region 'Informational'}
    stActiveInfo                   : ST_BA_ActiveInfo;
  {endregion}
  {region 'Output-Properties'}
    nPresentValue                  : UDINT := 1;
    nStateCount                    : UDINT;
{endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eStateTextDeterminationMode	E_BA_NodeTypeTarget [▶ 114]	Definiert den Bezug zu dem Objekt, dessen Beschreibung zur Darstellung des Status angezeigt wird.
aStateText	T_BA_StateTextArray [▶ 123]	Das Array dient der Deklaration der Statustexte eines Multi-State-Objektes.

VAR

Name	Typ	Beschreibung
stActiveInfo	ST_BA_ActiveInfo [▶ 124]	Beschreibung des Objekts.
nPresentValue	UDINT	Aktueller Wert für mehrstufige Ausgaben.
nStateCount	UDINT	Zahl der Status eines Multi-State-Objektes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.9 FB_BA_BaseStepMV



Der Funktionsbaustein FB_BA_BaseStepMV ermöglicht das Verarbeiten von Stufen. Jedes referenzierte Objekt wird dabei als Stufe gesehen und kann entsprechend befehligt werden.

Der *PresentValue* ist automatisch der aktuelle Schritt, bzw. die aktuelle Referenz, die von diesem Objekt kommandiert wird.

Abstract

Der FB dient als Basis (*ABSTRACT*) zum Bereitstellen von beschriebenen Funktionalitäten für erbende FB's.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_BaseStateMV [▶ 262]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_BaseStepMV EXTENDS FB_BA_BaseStateMV
VAR_OUTPUT
  sStep     : STRING;
  bOn       : BOOL;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
sStep	STRING	Aktuell befohlene Stufe.
bOn	BOOL	Zeigt Betrieb an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.10 FB_BA_ComEventObject



Der Funktionsbaustein FB_BA_ComEventObject beinhaltet die Basiseigenschaften bzw. Funktionen für das Erfassen, die Anzeige, das Weiterleiten, das Quittieren und Rücksetzen der Ereignisse eines Objektes.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

FB_BA_EventObjectEx [▶ 268]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_ComEventObject EXTENDS FB_BA_EventObjectEx IMPLEMENTS I_BA_ComObject
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    eCommissioningState    : E_BA_CommissioningState := E_BA_CommissioningState.eUnknown;
  {endregion}
END_VAR
```

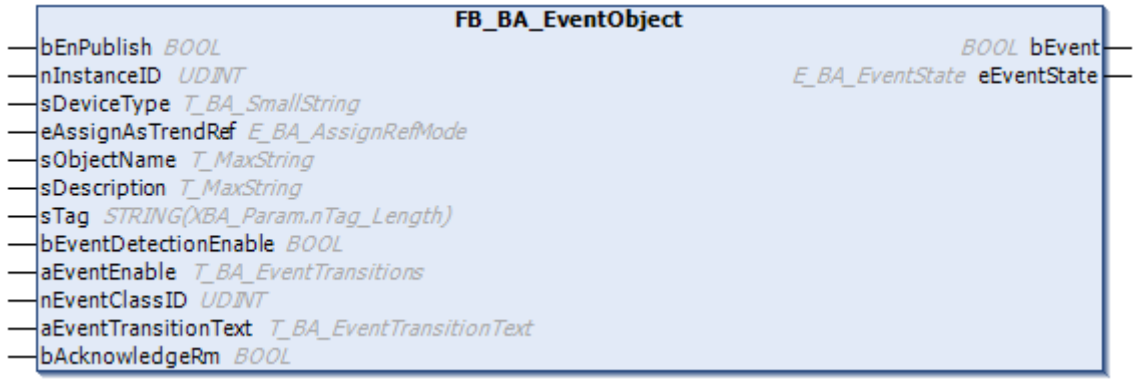
Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eCommissioningState	E_BA_CommissioningState [▶ 109]	Parametrierung des Inbetriebnahme - Status eines Objektes. Die Einstellungen des Inbetriebnahme - Status für die Objekte erfolgt mit dem Tool Site Explorer.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.11 FB_BA_EventObject



Der Funktionsbaustein FB_BA_EventObject erweitert die Grundfunktionalitäten eines BACnet Objektes um die Eigenschaften der objektinternen Benachrichtigung (Intrinsic Reporting). Hierbei werden Benachrichtigungen und Alarmmeldungen vom BACnet Objekt selbst generiert und an eine Meldeklasse weitergeleitet.

In diesem Objekt müssen die entsprechenden Ereignisse aktiviert werden. Dies geschieht über das Property *Event_Enable*. Jedes Ereignis (TO_OFFNORMAL, TO_FAULT und TO_NORMAL) kann einzeln aktiviert werden.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_EventObject EXTENDS FB_BA_Object IMPLEMENTS I_BA_EventObject, I_BA_Eve
ntValue
VAR_OUTPUT
  bEvent                    : BOOL;
  eEventState               : E_BA_EventState;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    bEventDetectionEnable   : BOOL := TRUE;
    nEventClassID           : UDINT;
    aEventTransitionText    : T_BA_EventTransitionText := BA_Param.aEventTransitionText;
  {endregion}
END_VAR
VAR_INPUT CONSTANT
  {region 'Operational Parameters'}
    bAcknowledgeRm          : BOOL;
  {endregion}
END_VAR
VAR
  {region 'Output-Properties'}
    eEventType              : E_BA_EventType := E_BA_EventType.Invalid;
    eAlarmMode              : E_BA_AlarmMode := E_BA_AlarmMode.Invalid;
    eReliability            : E_BA_Reliability;
    eEventTransition        : E_BA_EventTransition := E_BA_EventTransition.Invalid;
    {endregion}
END_VAR
VAR
  {region 'Event'}
    stStateFlags            : ST_BA_StatusFlags;
    stAckedTransitions      : ST_BA_EventTransitions;
    aEventEnable            : T_BA_EventTransitions;
  {endregion}
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bEvent	BOOL	Die Variable signalisiert ein unnormales Ereignis eines Objektes. Die Art des Ereignisses, z. B. ein Alarm, eine Wartungsmeldung etc., wird innerhalb der dem Objekt zugehörigen Ereignisklasse beschrieben.
eEventState	E_BA_EventState	Die Variable beschreibt den Status eines Ereignisses.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bEventDetectionEnable	BOOL	Die Variable gibt an, ob das intrinsische Reporting in dem Objekt aktiviert ist. Zur Aktivierung muss sie TRUE sein. Sie steuert, ob die Ereignisse innerhalb einer weiteren Verarbeitung z.B. in Eventlisten Berücksichtigung finden.
nEventClassID	UDINT	Diese Eigenschaft gibt die Instanz der Ereignis Klasse an, die für die Verteilung von Ereignisbenachrichtigungen verwendet werden soll (siehe Events [▶ 33])
aEventTransitionText	T_BA_EventTransitionText [▶ 117]	Das Array enthält drei Zeichenketten, welche die Basis für die Meldetexte der Ereignisse TO_OFFNORMAL, TO_FAULT und TO_NORMAL darstellen. Die Meldetexte sind in der Liste *TxtEvent_EN* bzw. *TxtEvent_DE* abgelegt. Je nach Sprachauswahl wird bei der Installation von TF8040 die englische oder deutsche Textliste installiert. In TF8040 werden die Meldetexte mit anderen Strings zu einer Meldung verkettet.

Eingänge CONSTANT

Name	Typ	Beschreibung
bAcknowledgeRm	BOOL	Eingang zur lokalen Quittierung der Ereignisse eines Objektes.

VAR

Name	Typ	Beschreibung
eEventType	E_BA_EventType [▶ 105]	Art des Ereignisses.
eAlarmMode	E_BA_AlarmMode [▶ 103]	Definition des Umgangs mit einem Alarm.
eReliability	E_BA_Reliability	Diese Variable gibt an, ob der Wert des Properties *Present_Value* zuverlässig ist (NO_FAULT_DETECTED), andernfalls stellt sie den Grund dar (z. B. Kurzschluss, fehlender Sensor usw.).
eEventTransition	E_BA_EventTransition	Abbildung des BACnet Datentyps *BACnetEventTransitionBits*.
stStateFlags	ST_BA_StatusFlags	Abbildung des BACnet Properties *Status_Flags*.
stAckedTransitions	ST_BA_EventTransitions	Abbildung des BACnet Properties *Acked_Transitions*.
aEventEnable	T_BA_EventTransitions [▶ 117]	Innerhalb des Arrays befinden sich drei Event-Enable-Bits. Damit kann das Erfassen der Statusänderungen TO_OFFNORMAL, TO_FAULT und TO_NORMAL am Objekt selektiv unterdrückt werden. aEventEnable[1] = TO_OFFNORMAL aEventEnable[2] = TO_FAULT aEventEnable[3] = TO_NORMAL

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.12 FB_BA_EventObjectEx



Der Funktionsbaustein FB_BA_EventObjectEx erweitert die Grundfunktionalitäten eines BACnet Objektes um die Eigenschaftenobjektinternen Benachrichtigung (Intrinsic Reporting). Hierbei werden Benachrichtigungen und Alarmmeldungen vom BACnet Objekt selbst generiert und an eine Meldeklasse weitergeleitet.

Zusätzlich wird das Objekt um die Eigenschaft einer Zeitverzögerung (*stTimeDelay*) der Ereignisverarbeitung (*eEnPlantLock*) erweitert.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_EventObject [▶ 266]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_EventObjectEx EXTENDS FB_BA_EventObject IMPLEMENTS I_BA_EventObjectEx
VAR_INPUT CONSTANT PERSISTENT
  {region 'Variable Parameters'}
    eEnPlantLock    : E_BA_LockPriority := E_BA_LockPriority.eNoLock;
    stTimeDelay     : ST_BA_TimeDelayParam := F_BA_TimeDelay(BA_Param.nDefTimeDelay_ToAbnormal, 0);
  {endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eEnPlantLock	E_BA_LockPriority [▶ 112]	Mit der Enumeration können verschiedene Schalthandlungen bezogen auf das Ereignis eines Objektes parametriert werden. Zum Beispiel kann es bei der Störung einer Brandschutzklappe erforderlich sein, die dazugehörige Lüftungsanlage abzuschalten (siehe PlantLock).
stTimeDelay	ST_BA_TimeDelayParam [▶ 122]	Mit der Struktur kann der Ereignisalgorithmus des Objekts für die Statusänderung TO_OFFNORMAL und TO_NORMAL verzögert werden. In der Standard SPS *PLC BA-Template* befindet sich die Liste BA2_Param. Darin sind die Verzögerungszeiten für TO_OFFNORMAL und TO_NORMAL mittels der zwei Variablen *nDefTimeDelay_ToAbnormal* und *nDefTimeDelay_ToNormal* jeweils mit einer Sekunde vorinitialisiert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.1.3.3.1.6.13 FB_BA_Object



Der Funktionsbaustein FB_BA_Object stellt die Grundlage zur Entwicklung weiterer Objekte dar.

Diese Objekte müssen im Projekt zyklisch aufgerufen werden, hierbei verhindert das Aussetzen von Aufrufen (z. B. innerhalb von IF-Bedingungen) eine korrekte Initialisierung.

Objekte sollten in einem VAR oder VAR_INPUT CONSTANT Bereich deklariert werden. Persistente Deklarationsbereiche müssen vermieden werden!

Innerhalb von geerbten FBs muss der Basis-FB mit "SUPER^()" aufgerufen werden!

Initialisierung

Während ein Objekt initialisiert wird, durchläuft es nacheinander mehrere Zustände:

- Alle Objekte initialisieren sich bis "E_BA_ObjectState.eWaitForInit"
- Danach gibt Top das Zeichen zum Fortsetzen der Objekt-Initialisierung
- Alle Objekte machen weiter bis "E_BA_ObjectState.eWaitForProject"
- Wenn alle Objekte auf "E_BA_ObjectState.eWaitForProject" stehen startet Top das Projekt.
- Alle Objekte gehen jetzt auf "E_BA_ObjectState.eOperation"
- Top signalisiert noch einen Zyklus lang "E_BA_ProjectState.eFirstOpCycle"
- Dann geht auch das Projekt in "E_BA_ProjectState.eOperation"

Vererbungshierarchie

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

Illustration

```
FUNCTION_BLOCK ABSTRACT FB_BA_Object EXTENDS FB_BA_BasePublisher IMPLEMENTS I_BA_Object, I_BACnet_Ob
jectOwner
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    nInstanceID          : UDINT;
    sDeviceType          : T_BA_SmallString := XBA_Globals.sPlaceholder_Empty;

    eAssignAsTrendRef    : E_BA_AssignRefMode := E_BA_AssignRefMode.eInitByProfile;
  {endregion}
  {region 'Variable Parameters'}
    sObjectName          : T_MaxString := XBA_Globals.sPlaceholder_Empty;
    sDescription         : T_MaxString := XBA_Globals.sPlaceholder_Empty;
    sTag                 : STRING(XBA_Param.nTag_Length);
  {endregion}
END_VAR
VAR PERSISTENT
  {region 'General'}
    sInstObjectName      : STRING;
    sInstDescription     : STRING;
  {endregion}
END_VAR
VAR
  {region 'Fixed Variables'}
    iParent              : I_BA_View;
    iLabel               : I_BA_Label;
    iProfile             : I_BA_Profile;
  {endregion}
  {region 'SubInit'}
    eObjectType          : E_BA_ObjectType := E_BA_ObjectType.eObject;
    stAttributes         : ST_BA_ObjectAttributes;
  {endregion}
END_VAR
VAR
  {region 'General'}
    nLevel               : UINT := 0; // Current hierarchy level according to root object (which has
t level 0)
    eState               : E_BA_ObjectState := E_BA_ObjectState.First;
    sFmtPresentValue     : STRING;// Present value in a formated string
  {endregion}
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nInstanceID	UDINT	Diese Eigenschaft ist ein numerischer Code, zur Identifizierung des Objekts. Er muss eindeutig innerhalb des Gerätes sein.
sDeviceType	T_BA_SmallString [▶ 122]	Diese Eigenschaft ist eine Textbeschreibung des physischen Geräts, das an den Analogausgang angeschlossen ist.
eAssignAsTrendRef	E_BA_AssignRefMode [▶ 113]	Referenzmodus.
sDescription	T_MaxString	Diese Eigenschaft stellt einen Beschreibungstext für das Objekt dar. Im Grundgerüst von TF8040 kann jede Ebene einen Teil-String erhalten. Die Teil-Strings der Ebenen werden von dem DPAD-Mechanismus verkettet, sodass mittels dieses Algorithmus der komplette Name des Objektes entsteht (siehe DPAD [▶ 43]).
sTag	STRING	Diese Eigenschaft stellt einen zusätzlichen Beschreibungstext für das Objekt dar. Der String *sTag* kann nutzer- bzw. projektspezifisch angewendet werden. Zum Beispiel zum Eintrag der Schaltschrankgerätekennzeichnung.

Eingänge PERSISTENT

Name	Typ	Beschreibung
sInstObjectName	STRING	Objektname der Instanz.
sInstDescription	STRING	Beschreibung der Instanz.

VAR

Name	Typ	Beschreibung
iParent	I_BA_Parent	Schnittstelle Elternobjekt.
iLabel	I_BA_Label	Schnittstelle Beschreibung.
iProfile	I_BA_Profile	Schnittstelle Profil.
eObjectType	E_BA_ObjectType [▶ 114]	Auswahl der Objekttyps.
stAttributes	ST_BA_ObjectAttributes [▶ 116]	Definition der Attribute.
nLevel	UINT	Aktuelle Hierarchieebene gemäß dem Grundobjekt (Ebene 0).
eState	E_BA_SubscriberState [▶ 115]	Auswertung des Subscriberstatus.
sFmtPresentValue	STRING	Aktueller Wert in einem formatierten String.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.35	Tc3_XBA ab v5.3.0.0

6.1.2.2 Tc3_BA2

6.1.2.2.1 DUTs

6.1.2.2.1.1 Enumerations

6.1.2.2.1.1.1 Room Automation

6.1.2.2.1.1.1.1 Heating Cooling

6.1.2.2.1.1.1.1.1 E_BA_Medium

```
TYPE E_BA_Medium :
(
  Invalid                 := 0,
  eNoMedium               := 1,
  eHeatMedium             := 2,
  eCoolMedium             := 3
);
END_TYPE
```

Name	Beschreibung
eNoMedium	Kein Medium.
eHeatMedium	Heizmedium.
eCoolMedium	Kühlmedium.

6.1.2.2.1.1.1.1.1.2 E_BA_PipeSys

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

Name	Beschreibung
e2Pipe	Zweileitersystem bedeutet, dass Heiz- und Kühlmittel in einem Leitersystem wahlweise verwendet werden.
e4Pipe	Im Vierleitersystem besitzen Heiz- und Kühlmittel ein getrenntes Leitersystem.

6.1.2.2.1.1.1.2 Lighting

6.1.2.2.1.1.1.2.1 E_BA_LightActivationMode

Die Aktivierung und Deaktivierung der Lichtsteuerfunktionen kann auf zwei unterschiedliche Weisen erfolgen: vollautomatisch oder halbautomatisch.

```
TYPE E_BA_LightActivationMode :
(
     eFullAutomatic,
     eSemiAutomatic
) BYTE;
END_TYPE
```

Name	Beschreibung
eFullAutomatic	Vollautomatik: Die Lichtsteuerfunktion wird über erkannte Präsenz aktiviert und über wegfallende Präsenz deaktiviert. Tastendrücke können die Funktion auch aktivieren.
eSemiAutomatic	Halbautomatik: Aktivierung der Lichtsteuerfunktion nur über Tastendruck, über wegfallende Präsenz ist sie automatisch deaktiviert.

6.1.2.2.1.1.1.2.2 E_BA_LightingPrio

Die Enumeration beinhaltet die unterschiedlichen Prioritäten der Lichtsteuerfunktionen.

In einem Projekt müssen nicht zwingend alle Prioritäten verwendet werden.

```
TYPE E_BA_LightingPrio:
(
      eNone                     := 0,
      eFire                     := 1,
      eCommError                := 2,
      eBurglary                 := 3,
      eMaintenance              := 4,
      eManualActuator           := 5,
      eManualGroup              := 6,

      eCleaning                 := 8,
      eNightWatch               := 9,

      eAllOff                   := 10,
      eAllOn                    := 11,
      eScene1                   := 12,
      eConstantLightControl     := 13,
      eAutomaticLight           := 14,
      eScene2                   := 16,
      eScene3                   := 17
) BYTE;
END_TYPE
```

Name	Beschreibung
eNone	Die Lichtsteuerung übernimmt keine Funktion.
eFire	Feueralarm
eCommError	Verhalten der Beleuchtung im Fehlerzustand, beispielsweise Kommunikationsunterbrechung.
eBurglary	Einbruch
eMaintenance	Wartung
eManualActuator	Handfunktion
eManualGroup	Hand-Gruppenfunktion
eCleaning	Gebäudereinigung
eNightWatch	Nachtwächterrundgang
eAllOff	Zentral Aus
eAllOn	Zentral Ein
eScene1	Szene 1
eConstantLightControl	Konstantlichtregelung
eAutomaticLight	Automatiklicht
eScene2	Szene 2
eScene3	Szene 3

6.1.2.2.1.1.1.3 Sun Protection

6.1.2.2.1.1.1.3.1 E_BA_PosMod

Enumerator zur Definition des Positioniermodus.

```
TYPE E_BA_PosMod :
(
  Invalid  := 0,
  eFix     := 1,
  eTab     := 2,
  eMaxIndc := 3
) BYTE;
END_TYPE
```

Name	Beschreibung
eFix	Die Jalousiehöhe nimmt einen festen Wert ein, welcher am Baustein FB_BA_SunPrtc [▶ 406] über den Wert *fFixPos* eingestellt wird [%].
eTab	Die Höhenpositionierung erfolgt mit Hilfe einer Tabelle von 6 Stützpunkten, davon 4 parametrierbar. Aus diesen Punkten wird dann durch lineare Interpolation eine Jalousieposition in Abhängigkeit des Sonnenstandes errechnet (siehe FB_BA_BldPosEntry [▶ 347]).
eMaxIndc	Die Positionierung erfolgt unter Angabe des maximal gewünschten Lichteinfalles.

6.1.2.2.1.1.1.3.2 E_BA_ShdObjType

Enumerator zur Auswahl des Verschattungsobjekt-Typs.

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

Name	Beschreibung
eTetragon	Objekttyp ist ein Viereck.
eGlobe	Objekttyp ist eine Kugel.

6.1.2.2.1.1.1.3.3 E_BA_SunBldPrio

```
TYPE E_BA_SunBldPrio :
(
  Invalid                    := 0,
  eFire                      := 1,
  eStorm                     := 2,
  eIce                       := 3,
  eCommError                 := 4,
  eBurglary                  := 5,
  eMaintenance               := 6,
  eReferencing               := 7,
  eManualActuator            := 8,
  eManualGroup               := 9,
  eAllDown                   := 10,
  eAllUp                     := 11,
  eScene1                    := 12,
  eFacadeThermoAutomatic     := 13,
  eFacadeTwilightAutomatic   := 14,
  eParkPosition              := 15,
  eScene2                    := 16,
  eScene3                    := 17,
  eSunProtection             := 18,
  eGroupThermoAuto           := 19,
  eGroupTwiLightAuto         := 20
) BYTE;
END_TYPE
```

Name	Beschreibung
eFire	Feueralarm
eStorm	Sturmfall
eIce	Vereisung
eCommError	Verhalten der Beleuchtung im Fehlerzustand, beispielsweise Kommunikationsunterbrechung.
eBurglary	Einbruch
eMaintenance	Wartung
eReferencing	Zentrale Referenzfahrt
eManualActuator	Handfunktion
eManualGroup	Hand-Gruppenfunktion
eAllDown	Zentral Ab
eAllUp	Zentral Auf
eScene1	Szene 1
eFacadeThermoAutomatic	Fassadenweite Thermoautomatik
eFacadeTwilightAutomatic	Fassadenweite Dämmerungsautomatik
eParkPosition	Parkposition
eScene2	Szene 2
eScene3	Szene 3
eSunProtection	Sonnenschutz bei Präsenz.
eGroupThermoAuto	Gruppen- bzw. zonenweite Thermoautomatik.
eGroupTwiLightAuto	Gruppen- bzw. zonenweite Dämmerungsautomatik.

6.1.2.2.1.1.2 Universal

6.1.2.2.1.1.2.1 E_BA_AntBlkgMode

```
TYPE E_BA_AntBlkgMedium :
(
  eOff                := 1,
  eExternalRequest    := 2,
  eOffTime            := 3
) UDINT;
END_TYPE
```

Name	Beschreibung
eOff	Deaktivierung des Funktionsbausteins
eExternalRequest	Betriebsart externe Anforderung ist aktiv.
eOffTime	Betriebsart Mindestausschaltzeit ist aktiv.

6.1.2.2.1.1.2.2 E_BA_Mdlt

Enum für die Regulierung eines Antriebs oder Aggregates.

```
{attribute 'qualified_only'}
TYPE E_BA_Mdlt :
(
  eOff    := 1,
  eOn     := 2,
  eMin    := 3,
  eMax    := 4
) UDINT;
END_TYPE
```

Name	Beschreibung
eOff	Ausschalten des Antriebs.
eOn	Einschalten des Antriebs.
eMin	Minimale Leistungsstufe des Antriebs.
eMax	Maximale Leistungsstufe des Antriebs.

6.1.2.2.1.1.2.3 Control

6.1.2.2.1.1.2.3.1 E_BA_StatePIDControlSync

E_BA_StatePIDControlSync zeigt den Status der Synchronisation des FB_BA_PIDControlSync [▶ 434] an.

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

Name	Beschreibung
eOK	Die Synchronisation ist fehlerfrei.
eErrAction	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *eAction* am FB_BA_PIDControl [▶ 430].
eErrDampConstant	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *nDampConstant* am FB_BA_PIDControl [▶ 430].
eErrDerivativeConstant	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *fDerivativeConstant* am FB_BA_PIDControl [▶ 430].
eErrIntegralConstant	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *fIntegralConstant* am FB_BA_PIDControl [▶ 430].
eErrMaxOutput	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *fMaxOutput* am FB_BA_PIDControl [▶ 430].
eErrMinOutput	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *fMinOutput* am FB_BA_PIDControl [▶ 430].
eErrNeutralZone	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *fNeutralZone* am FB_BA_PIDControl [▶ 430].
eErrOpMode	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *eOpMode* am FB_BA_PIDControl [▶ 430].
eErrProportionalConstant	Fehlerhafte Synchronisation der VAR_INPUT CONSTANT Variablen *fProportionalConstant* am FB_BA_PIDControl [▶ 430].

6.1.2.2.1.1.2.4 Schedule

6.1.2.2.1.1.2.4.1 E_BA_CtrlFct

Die Optmierungsfunktionen FB_BA_OptimizedOn und FB_BA_OptimizedOff führen intern unterschiedliche Berechnungen aus, je nachdem, ob die anzusteuernde Anlage ein Heiz- oder Kühlsystem ist.

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

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

Name	Beschreibung
eCtrlFct_Off	Keine Anwahl getätigt. Die Optimierungsfunktion wird nicht aufstarten.
eCtrlFct_Heating	Die Optimierungsfunktion betrachtet den Heizbetrieb.
eCtrlFct_Cooling	Die Optimierungsfunktion betrachtet den Kühlbetrieb.

6.1.2.2.1.2 Types

6.1.2.2.1.2.1 Room Automation

6.1.2.2.1.2.1.1 Heating Cooling Functions

6.1.2.2.1.2.1.1.1 ST_BA_SpRmT

Raum-Temperatursollwerte.

```
TYPE ST_BA_SpRmT :
STRUCT
   fPrtcHtg    : REAL := 12.0;
   fEcoHtg     : REAL := 15.0;
   fPreCmfHtg  : REAL := 19.0;
   fCmfHtg     : REAL := 21.0;
   fPrtcCol    : REAL := 40.0;
   fEcoCol     : REAL := 35.0;
   fPreCmfCol  : REAL := 28.0;
   fCmfCol     : REAL := 24.0;
END_STRUCT
END_TYPE
```

Die Werte in der Struktur sind mit den Voreinstellwerten deklariert.

Name	Typ	Beschreibung
fPrtcHtg	REAL	Protection Heating
fEcoHtg	REAL	Economy Heating
fPreCmfHtg	REAL	Pre-Comfort Heating
fCmfHtg	REAL	Comfort Heating
fPrtcCol	REAL	Protection Cooling
fEcoCol	REAL	Economy Cooling
fPreCmfCol	REAL	Pre-Comfort Cooling
fCmfCol	REAL	Comfort Cooling

6.1.2.2.1.2.1.2 Lighting

6.1.2.2.1.2.1.2.1 ST_BA_Lighting

```
TYPE ST_BA_Lighting :
STRUCT
  {attribute 'parameterUnit':= '%'}
  fLgtVal    : REAL
  {attribute 'parameterUnit':= 'K'}
  fLgtT      : REAL;

  bActv      : BOOL;
  ePrio      : E_BA_LightingPrio;

  nEvtInc    : ULINT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fLgtVal	REAL	Übergebener Lichtwert [%] im Absolutwert-Steuermodus.
fLgtT	REAL	Übergebene Lichttemperatur [K].
bActv	BOOL	Der Absender des Telegramms ist aktiv. Dieses Bit wird nur von der Prioritätssteuerung ausgewertet.
ePrio	E_BA_LightingPrio [▶ 273]	Priorität des aktiven Telegramms. Diese Enumeration wird nur von der Prioritätssteuerung ausgewertet.
nEvtInc	ULINT	Telegrammzähler. Mit jedem neuen Telegramm, egal welcher Baustein auf derselben Steuerung es auslöst, wird dieser Zähler um Eins erhöht. Bei Telegrammen gleicher Priorität "gewinnt" das mit dem höheren Zähler.

6.1.2.2.1.2.1.3 Sun Protection

6.1.2.2.1.2.1.3.1 ST_BA_BldPosTab

Struktur der Stützpunkteinträge für die Höhenverstellung der Jalousie.

```
TYPE ST_BA_BldPosTab:
STRUCT
  aSunElv   : ARRAY[0..5] OF REAL;
  aPos      : ARRAY[0..5] OF REAL;
  bVld      : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
aSunElv / aPos	REAL	Die 6 Stützstellen welche übergeben werden, wobei die Array-Elemente 0 und 5 die oben erwähnten automatisch generierten Rand-Elemente darstellen.
bVld	BOOL	Gültigkeitsflag für den Baustein FB_BA_SunPrtc. Wird vom Baustein FB_BA_BldPosEntry auf TRUE gesetzt, wenn die eingegebenen Daten den beschriebenen Gültigkeitskriterien entsprechen.

6.1.2.2.1.2.1.3.2 ST_BA_FcdElem

Listeneintrag eines Fassadenelementes (Fenster).

```
TYPE ST_BA_FcdElem:
STRUCT
  fWdwWdth  : REAL;
  fWdwHght  : REAL;
  aCnr      : ARRAY [1..4] OF ST_BA_Cnr;
  nGrp      : DINT;
  bVld      : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fWdwWdth	REAL	Breite des Fensters [m].
fWdwHght	REAL	Höhe des Fensters [m].
aCnr	ST_BA_Cnr [▶ 279]	Koordinaten der Fenster-Eckpunkte und Information, ob dieser Eckpunkt verschattet ist.
nGrp	DINT	Angabe der Gruppe zu der das Fenster gehört.
bVld	BOOL	Plausibilität der eingetragenen Daten: *bVld* = TRUE: Daten sind plausibel.

6.1.2.2.1.2.1.3.3 ST_BA_Cnr

Information über Fenster-Eckpunkte.

```
TYPE ST_BA_Cnr :
STRUCT
  fX     : REAL;
  fY     : REAL;
  bShdd  : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fX	REAL	X-Koordinate des Fensters (auf der Fassade).
fY	REAL	Y-Koordinate des Fensters (auf der Fassade).
bShdd	BOOL	Information, ob dieser Eckpunkt verschattet ist: *bShdd* = TRUE: Eckpunkt ist verschattet.

6.1.2.2.1.2.1.3.4 ST_BA_SunBld

Struktur des Jalousie-Positioniertelegramms.

```
TYPE ST_BA_SunBld :
STRUCT
  fPos      : REAL;
  fAngl     : REAL;
  bManUp    : BOOL;
  bManDwn   : BOOL;
  bManMod   : BOOL;
  bActv     : BOOL;
  ePrio     : E_BA_SunBldPrio;
  nEvtInc   : UDINT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fPos	REAL	Übergebene Jalousiehöhe [%].
fAngl	REAL	Übergebene Lamellenstellung [°].
bManUp	BOOL	Handbefehl: Jalousie hoch.
bManDwn	BOOL	Handbefehl: Jalousie herunter.
bManMod	BOOL	TRUE: Der Handbedienmodus ist aktiv. FALSE: Der Automatikmodus ist aktiv.
bActv	BOOL	Prioritätssteuerung FB_BA_SunBldPrioSwi4 [▶ 393], FB_BA_SunBldPrioSwi8 [▶ 394], FB_BA_SunBldTgmSel4 [▶ 400], FB_BA_SunBldTgmSel8 [▶ 401] ausgewertet. Die Sonnenschutzaktoren FB_BA_SunBldActr [▶ 383] und FB_BA_RolBldActr [▶ 373] beachten es nicht.
ePrio	E_BA_SunBldPrio [▶ 274]	Die Priorität des Telegramms. Diese Enumeration wird nur von der Prioritätssteuerung ausgewertet.
nEvtInc	EDINT	Telegrammzähler. Mit jedem neuen Telegramm, egal welcher Baustein auf derselben Steuerung es auslöst, wird dieser Zähler um Eins erhöht. Bei Telegrammen gleicher Priorität "gewinnt" das mit dem höheren Zähler.

6.1.2.2.1.2.1.3.5 ST_BA_ShdObj

Listeneintrag eines Verschattungsobjektes.

```
TYPE ST_BA_ShdObj :
STRUCT
  fP1x       : REAL;
  fP1y       : REAL;
  fP1z       : REAL;
  fP2x       : REAL;
  fP2y       : REAL;
```

```
  fP2z       : REAL;
  fP3x       : REAL;
  fP3y       : REAL;
  fP3z       : REAL;
  fP4x       : REAL;
  fP4y       : REAL;
  fP4z       : REAL;
  fMx        : REAL;
  fMy        : REAL;
  fMz        : REAL;
  fRads      : REAL;
  nBegMth    : USINT;
  nEndMth    : USINT;
  eType      : E_BA_ShdObjType;
  bVld       : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fP1x…fP4z	REAL	Eckkoordinaten. Nur von Bedeutung, wenn das Element ein Viereck ist.
fMx…fMz	REAL	Mittelpunktkoordinaten. Nur von Bedeutung, wenn das Element eine Kugel ist.
fRads	REAL	Radius der Kugel. Nur von Bedeutung, wenn das Element eine Kugel ist.
nBegMth	USINT	Anfang der Verschattungsperiode (Monatszahl).
nEndMth	USINT	Ende der Verschattungsperiode (Monatszahl).
eType	E_BA_ShdObjType [▶ 274]	Objekttyp
bVld	BOOL	Plausibilität der Daten: *bVld* = TRUE: Daten sind plausibel.

Bemerkung zur Verschattungsperiode:

Die Monatseinträge dürfen nicht 0 und größer 12 sein, andernfalls sind alle Kombinationen möglich.

Beispiele:

Beginn=1, Ende=1: Verschattung im Januar.

Beginn=1, Ende=5: Verschattung von Anfang Januar bis Ende Mai.

Beginn=11, Ende=5: Verschattung von Anfang November bis Ende Mai (des folgenden Jahres).

6.1.2.2.1.2.1.3.6 ST_BA_SunBldScn

Tabelleneintrag einer Jalousie-Szene.

```
TYPE ST_BA_SunBldScn:
STRUCT
  fPos     : REAL;
  fAngl    : REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fPos	REAL	Jalousiehöhe [%]
fAngl	REAL	Lamellenwinkel [°]

6.1.2.2.1.2.2 Universal

6.1.2.2.1.2.2.1 Aggregate

6.1.2.2.1.2.2.1.1 ST_BA_Multistate

Die Kommandostruktur dient zur Ansteuerung mehrstufiger Aggregate und beinhaltet die Prioritäten Sicherheit, Kritisch und Programm.

```
TYPE ST_BA_Multistate:
STRUCT
     bEnSfty     : BOOL;
     nValSfty    : UDINT;
     bEnCrit     : BOOL;
     nValCrit    : UDINT;
     bEnPgm      : BOOL;
     nValPgm     : UDINT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“.
nValSfty	UDINT	Zu beschreibender Wert der Priorität „Safety“.
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“.
nValCrit	UDINT	Zu beschreibender Wert der Priorität „Critical“.
bEnPgm	BOOL	Freigabe zum Beschreiben der Priorität „Program“.
nValPgm	UDINT	Zu beschreibender Wert der Priorität „Program“.

6.1.2.2.1.2.2.1.2 ST_BA_Analog

```
TYPE ST_BA_Analog :
STRUCT
  bEnSfty     : BOOL;
  fValSfty    : REAL;
  bEnCrit     : BOOL;
  fValCrit    : REAL;
  bEnPgm      : BOOL;
  fValPgm     : REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“.
fValSfty	REAL	Zu beschreibender Wert der Priorität „Safety“.
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“.
fValCrit	REAL	Zu beschreibender Wert der Priorität „Critical“.
bEnPgm	BOOL	Freigabe zum Beschreiben der Priorität „Program“.
fValPgm	REAL	Zu beschreibender Wert der Priorität „Program“.

6.1.2.2.1.2.2.1.3 ST_BA_Mdlt

Die Kommandostruktur dient zur Ansteuerung modulierender Aggregate und beinhaltet die Prioritäten „Safety“, „Critical“ und „Program“.

```
TYPE ST_BA_Mdlt :
STRUCT
  bEnSfty     : BOOL;
  eValSfty    : E_BA_Mdlt;
  bEnCrit     : BOOL;
  eValCrit    : E_BA_Mdlt;
  bEnPgm      : BOOL;
  eValPgm     : E_BA_Mdlt;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“.
eValSfty	E_BA_Mdlt [▶ 275]	Zu beschreibendes Enum der Priorität „Safety“.
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“.
eValCrit	E_BA_Mdlt [▶ 275]	Zu beschreibendes Enum der Priorität „Critical“.
bEnPgm	BOOL	Freigabe zum Beschreiben der Priorität „Program“.
eValPgm	E_BA_Mdlt [▶ 275]	Zu beschreibendes Enum der Priorität „Program“.

6.1.2.2.1.2.2.1.4 ST_BA_Binary

Die Kommandostruktur dient zur Ansteuerung binärer Aggregate und beinhaltet die Prioritäten Sicherheit, Kritisch und Programm.

```
TYPE ST_BA_Binary:
STRUCT
  bEnSfty     : BOOL;
  bValSfty    : BOOL;
  bEnCrit     : BOOL;
  bValCrit    : BOOL;
  bEnPgm      : BOOL;
  bValPgm     : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“.
bValSfty	BOOL	Zu beschreibender Wert der Priorität „Safety“.
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“.
bValCrit	BOOL	Zu beschreibender Wert der Priorität „Critical“.
bEnPgm	BOOL	Freigabe zum Beschreiben der Priorität „Program“.
bValPgm	BOOL	Zu beschreibender Wert der Priorität „Program“.

6.1.2.2.1.2.2.1.5 ST_BA_Step

Daten- und Kommandostruktur zwischen den einzelnen Schrittkettenbausteinen FB_BA_StepBinary / FB_BA_StepMdlt und dem Steuerbaustein der Schrittkette FB_BA_StepCtrlAgg16.

Illustration

```
TYPE ST_BA_Step:
STRUCT
  nStep          : UDINT;
  nRemSecOn      : UDINT;
  nRemSecOff     : UDINT;
  bEnUp          : BOOL;
  bEnDown        : BOOL;
  bRdyUp         : BOOL;
  bRdyDown       : BOOL;
  bUp            : BOOL;
  bDown          : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
nStep	UDINT	Anzeige in welchem Schritt sich die Schrittkettensteuerung befindet.
nRemSecOn	UDINT	Countdown Einschaltung des nächsten Schritts [s]. Das dazu gehörige Zeitglied ist in den Schrittkettenbausteinen FB_BA_StepBinary [▶ 484] / FB_BA_StepMdlt [▶ 486] integriert.
nRemSecOff	UDINT	Countdown Ausschaltung des nächsten Schritts [s]. Das dazu gehörige Zeitglied ist in den Schrittkettenbausteinen FB_BA_StepBinary [▶ 484] / FB_BA_StepMdlt [▶ 486] integriert.
bEnUp	BOOL	Zeigt an, dass der Schritt seine Freigabe hat.
bEnDown	BOOL	Die Variable zeigt an, dass sich der aktive Schritt im Ausschaltzustand befindet. Nach Ablauf der Ausschaltverzögerung *nRemSecOff* wird der Schritt wie auch *bEnDown* ausgeschaltet und der nächste Schritt in der fallenden Ausschaltreihenfolge wird der aktive.
bRdyUp	BOOL	Nach Ablauf der Einschaltverzögerung *nRemSecOn* und einem TRUE an der Rückmeldung *bFdb* des Schrittkettenbausteinen wird *bRdyUp* gesetzt und in den nächsten Schritt weitergeschaltet.
bRdyDown	BOOL	Nach Ablauf der Ausschaltverzögerung *nRemSecOff* wird *bRdyDown* für einen Zyklus gesetzt und der Schritt ausgeschaltet.
bUp	BOOL	Ein TRUE bedeutet eine steigende Einschaltreihenfolge der Aggregate von 1 nach 16.
bDown	BOOL	Ein TRUE bedeutet eine fallende Ausschaltreihenfolge vom höchsten, aktiven Aggregat Richtung 0.

6.1.2.2.1.2.2.1.6 ST_BA_PriorityEn

Die Kommandostruktur beinhaltet die Freigaben der Anlage und der Prioritäten „Safety“, „Critical“ und „Program“.

Illustration

```
TYPE ST_BA_PriorityEn:
STRUCT
  bPlt       : BOOL;
  bEnSfty    : BOOL;
  bEnCrit    : BOOL;
  bEnPgm     : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bPlt	BOOL	Freigabe Anlage.
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“.
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“.
bEnPgm	BOOL	Freigabe zum Beschreiben der Priorität „Program“.

6.1.2.2.1.2.2.2 PlantControl

6.1.2.2.1.2.2.2.1 ST_BA_Aggregate

Die Aggregatestruktur dient der bidirektionalen Kommunikation zwischen dem angeschlossen Aggregat FB_BA_Aggregate [▶ 489] und der Anlagensteuerung FB_BA_PlantControl [▶ 491].

Die Struktur beinhaltet die Schaltwerte, das Anlagenprofil und den aktuellen Schritt der Schrittkettensteuerung.

Die Anlagensteuerung erhält für die Schrittkettensteuerung Rückmeldungen aus dem Aggregat.

Syntax

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

Name	Typ	Beschreibung
nStep	UDINT	Zeigt den aktuellen Schritt der Schrittkettensteuerung an.
nPrio	UDINT	Zeigt die aktive Priorität des Profils an.
nProfile	UDINT	Zeigt an, welches Profil aktiv ist.
fValue	REAL	Überträgt den analogen Schaltwert in das Aggregat.
nValue	UDINT	Überträgt den numerischen Schaltwert in das Aggregat.
bValue	BOOL	Überträgt den booleschen Schaltwert in das Aggregat.
bUp	BOOL	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung in steigendem Zustand befindet.
bDown	BOOL	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung in fallendem Zustand befindet.
bWatchdog	BOOL	Bidirektionale Überwachung der Kommunikation zwischen dem Aggregat und der Anlagensteuerung. Ist ein Aggregat innerhalb der Schrittkettensteuerung nicht vorhanden, so kann die Schrittkettensteuerung nicht in diesen Schritt hineinlaufen.
bFdbUp	BOOL	Rückmeldung „Hoch“ des jeweiligen Aggregats an die Anlagensteuerung.
bFdbDown	BOOL	Rückmeldung „Runter“ des jeweiligen Aggregats an die Anlagensteuerung.

6.1.2.2.1.2.2.2.2 ST_BA_ProfileDescription

Die Parametrierungsstruktur dient innerhalb der Anlagensteuerung FB_BA_PlantControl [▶ 491] der Beschreibung eines Profils.

Syntax

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

Name	Typ	Beschreibung
sDescription	T_MaxString	Klartext um welches Anlagenprofil es sich handelt. Dient rein der Anzeige.
nPrio	UDINT	Die Priorität dient der Kommandierung aller Aggregate einer Anlage. Die Priorität wird vom Anlagensteuerprogramm FB_BA_PlantControl [▶ 491] innerhalb der ST_BA_Aggregate [▶ 283] an die Aggregate übergeben, und am Funktionsbaustein FB_BA_Aggregate [▶ 489] ausgegeben. Innerhalb der Aggregate-Templates wird diese Priorität zur Kommandierung der Objekte verwendet.

6.1.2.2.1.2.2.2.3 ST_BA_ProfileParameter

Die Parametrierungsstruktur dient innerhalb der Anlagensteuerung FB_BA_PlantControl [▶ 491] der Parametrierung der Parameter eines Schrittes respektive eines Aggregates. Die Struktur beinhaltet den binären, numerischen und analogen Schaltwert für das Aggregat, die Verzögerungszeiten und die Freigabe für die Verwendung der Rückmeldungen der Aggregate.

Syntax

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

  fDelayUp       : REAL;
  fDelayDown     : REAL;

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

  bEnFdbUp       : BOOL;
  bEnFdbDown     : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
sStep	T_MaxString	Klartext um welchen Anlagenschritt es sich handelt. Der Inhalt des Textes wird über die Parameterstruktur *arrStepDescription* im Funktionsbaustein FB_BA_PlantControl [▶ 491] vorgegeben.
sProfile	T_MaxString	Klartext um welches Profil es sich handelt. Der Inhalt des Textes wird über die Struktur *arrProfileDescription* im Funktionsbaustein FB_BA_PlantControl [▶ 491] vorgegeben.
fDelayUp	REAL	Zeitangabe der Einschaltverzögerung [s,ms]. Die Zeitangabe wird innerhalb der Anlagensteuerung zur Weiterschaltung nur berücksichtigt, wenn diese > 0 ist.
fDelayDown	REAL	Zeitangabe der Ausschaltverzögerung [s,ms]. Die Zeitangabe wird innerhalb der Anlagensteuerung zur Runterschaltung nur berücksichtigt, wenn diese > 0 ist.
fValue	REAL	Analoger Schaltwert für das Aggregat.
nValue	UDINT	Numerischer Schaltwert für das Aggregat.
bValue	BOOL	Binärer Schaltwert für das Aggregat.
bEnFdbUp	BOOL	Ein TRUE bedeutet, dass die Rückmeldung des Aggregates in dem Schritt „Hoch“ verwendet wird.
bEnFdbDown	BOOL	Ein TRUE bedeutet, dass die Rückmeldung des Aggregates in dem Schritt „Runter“ verwendet wird.

6.1.2.2.1.2.2.3 Sequence

6.1.2.2.1.2.2.3.1 ST_BA_PIDControlSequence

Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_PIDControlSequence und dem Steuerbaustein FB_BA_PIDControlSequenceLink [▶ 500].

Syntax

```
TYPE ST_BA_PIDControlSequence :
STRUCT
  arrSeqLinkData    : ARRAY[1..MAX(1,BA_Param.nMaxSeqCtrl)] OF ST_BA_PIDControlSequenceData;
  fE                : REAL;
  nActvSeqCtrl      : UDINT;
  bSync             : BOOL;
  bEnSeqLink        : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
arrSeqLinkData	ARRAY[1..MAX(1,BA_Param.nMaxSeqCtrl [▶ 289])] OF ST_BA_PIDControlSequenceData [▶ 286]	Dieses Feld der Struktur *ST_BA_PIDControlSequenceData* zeigt die Daten und Befehle der einzelnen Sequenzregler an. Limitiert ist dieses Feld durch den globalen Parameter *BA_Param.nMaxSeqCtrl.*
fE	REAL	Zeigt die Regelabweichung des aktiven Sequenzreglers an. Diese ist abhängig vom Wirksinn des jeweiligen Sequenzreglers. *E_BA_Action.eDirect -> fE = fX-fW* *E_BA_Action.eReverse -> fE = fW-fX*
nActvSeqCtrl	UDINT	Nummer des aktiven Sequenzreglers.
bSync	BOOL	Ein Synchronisationsimpuls ist an einem der Sequenzregler ausgelöst worden. Anhand dieses Impulses wird der Sequenzregler der Aktive in der Sequenz.
bEnSeqLink	BOOL	Diese Variable zeigt die Freigabe *bEn* des Funktionsbausteins FB_BA_PIDControlSequenceLink [▶ 500] an.

6.1.2.2.1.2.2.3.2 ST_BA_PIDControlSequenceData

Diese Struktur beinhaltet die Daten und Befehle der einzelnen Sequenzregler FB_BA_PIDControlSequence [▶ 497].

Syntax

```
TYPE ST_BA_PIDControlSequenceData :
STRUCT
  fY                : REAL;
  fYMin             : REAL;
  fYMax             : REAL;
  fW                : REAL;
  fX                : REAL;
  fE                : REAL;
  nActvSeqCtrl      : UDINT;
  nMyNum            : UDINT;
  eActn             : E_BA_Action;
  bSeqCtrlOperable  : BOOL;
  bWatchdog         : BOOL;
  bSeqNumMultiple   : BOOL;
  bEn               : BOOL;
  bIsActvSeqCtrl    : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fY	REAL	Stellgröße
fYMin	REAL	Minimale Stellgröße
fYMax	REAL	Maximale Stellgröße
fW	REAL	Sollwert
fX	REAL	Istwert
fE	REAL	Zeigt die Regelabweichung des Sequenzreglers an. Diese ist abhängig vom Wirksinn des jeweiligen Sequenzreglers. *E_BA_Action.eDirect -> fE = fX-fW* E_BA_Action.eReverse -> *fE = fW-fX*
nActvSeqCtrl	UDINT	Nummer des aktiven Sequenzcontrollers.
nMyNum	UDINT	Meine Nummer in der Sequenz.
eActn	E_BA_Action	Regler Wirksinn
bSeqCtrlOperable	BOOL	Der Sequenzregler ist betriebsbereit.
bWatchdog	BOOL	Der Watchdog wird bei jedem SPS-Zyklus in jedem Sequenzregler gesetzt und nach der Auswertung im Sequenz-Link zurückgesetzt.
bSeqNumMultiple	BOOL	Zeigt an, dass die eigene Sequenznummer *nMyNum* mehrfach an Sequenzreglern vergeben wurde.
bEn	BOOL	Eingang Freigabe der Sequenz Regler
bIsActvSeqCtrl	BOOL	Zeigt an, dass der Sequenzregler der Aktive in der Sequenz ist.

6.1.2.2.1.2.2.4 Schedule

6.1.2.2.1.2.2.4.1 ST_BA_TempChangeFunction

Diese Struktur enthält zu einer Außentemperatur *fOutsideTemp* die zu erwartende Temperaturänderung *fRoomTempChange* beim Einschalten (FB_BA_OptimizedOn) bzw. beim Ausschalten (FB_BA_OptimizedOff) einer Heiz- oder Kühlanlage.

```
TYPE ST_BA_TempChangeFunction :
STRUCT
  fOutsideTemp     : ARRAY[1..FB_BA_OptimizationBase.nOptOnOff_MaxNodes] OF REAL;
  fRoomTempChange  : ARRAY[1..FB_BA_OptimizationBase.nOptOnOff_MaxNodes] OF REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fOutsideTemp	REAL	Die betrachteten Außentemperaturstützstellen der Optimierungsfunktion in °C.
fRoomTempChange	REAL	Die zu den Außentemperaturen zugehörigen erwarteten Raumtemperatur-Änderungswerte in K/min.

6.1.2.2.1.2.2.4.2 T_BA_ScheduleWeek

Die Typdeklaration beinhaltet einen Wochenzeitplan. Der erste Teil des zweidimensionalen Arrays gibt die Wochentage an, der zweite Teil die Zeitschaltplan-Einträge, siehe ST_BA_SchedEntry.

Die Anzahl der Zeitschaltplan-Einträge ist limitiert durch den globalen Parameter *BA_Param.nScheduleEntryCount*, siehe BA_Param [▶ 289].

```
TYPE
  T_BA_ScheduleWeek : ARRAY[E_BA_Weekday.First..E_BA_Weekday.Last, 1..MAX(1,BA_Param.nScheduleEntryC
ount)] OF ST_BA_SchedEntry;
END_TYPE
```

6.1.2.2.1.2.2.4.3 T_BA_ScheduleExceptionList

Die Typdeklaration beinhaltet ein Feld von 24 Zeitplan Ausnahme-Profilen, siehe ST_BA_ScheduleException [▶ 288].

Limitiert ist dieses Feld durch den globalen Parameter *BA_Param.nScheduleExceptionCount*, siehe BA_Param [▶ 289].

```
TYPE
  T_BA_ScheduleExceptionList : ARRAY[1 .. BA_Param.nScheduleExceptionCount] OF ST_BA_ScheduleExcepti
on;
END_TYPE
```

6.1.2.2.1.2.2.4.4 T_BA_ScheduleExceptionEntryList

Die Typdeklaration beinhaltet 6 Zeitschaltplan-Einträge für ein Ausnahme-Profil, siehe ST_BA_SchedEntry.

Limitiert ist dieses Feld durch den globalen Parameter *BA_Param.nScheduleEntryCount*, siehe BA_Param [▶ 289].

```
TYPE
  T_BA_ScheduleExceptionEntryList    : ARRAY[1 .. BA_Param.nScheduleEntryCount] OF ST_BA_SchedEntry;
END_TYPE
```

6.1.2.2.1.2.2.4.5 ST_BA_ScheduleException

Die Struktur repräsentiert ein Zeitplan-Ausnahme-Profil.

```
TYPE ST_BA_ScheduleException :
STRUCT
  eType     : E_BA_DateValChoice := E_BA_DateValChoice.Invalid;
  uDate     : U_BA_DateVal;
  aEntry    : T_BA_ScheduleExceptionEntryList;
END_STRUCT
END_YPE
```

Name	Typ	Beschreibung
eType	E_BA_DateValChoice	Angabe des Zeitraumtypen für den Ausnahme-Profileintrag. Es kann unter 3 Möglichkeiten (Einfaches Datum, Datumsbereich, Woche Und Tag) ausgewählt werden.
uDate	U_BA_DateVal	Angabe des Datumswertes für den angegebenen Zeitraumtypen *eType*.
aEntry	T_BA_ScheduleExceptionEntryList [▶ 288]	Liste von Zeitschaltplan-Einträgen für ein Ausnahme-Profileintrag.

6.1.2.2.2 GVLs

6.1.2.2.2.1 BA_Globals

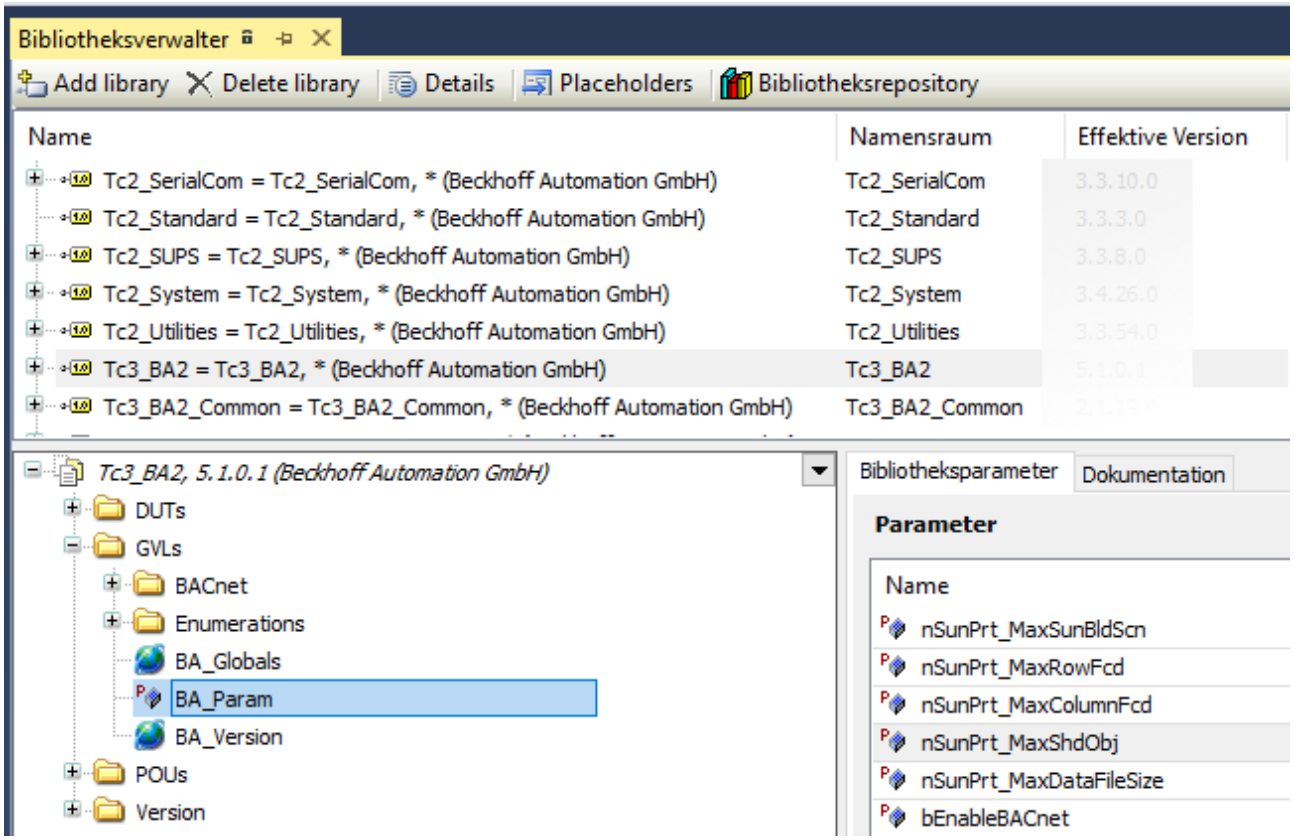
Die globale Variablenliste beinhaltet Wertemerker.

```
VAR_GLOBAL
{region 'Room-Automation'}
  nEvtIncLight     : ULINT;
  nEvtIncSunBld    : UDINT;
{endregion}
{region 'Constants'}
 {region 'General'}
  nNoActivePrio  : UDINT := 16#FFFFFFFF; // Value of the constant indicates that no priority is acti
ve
 {endregion}
{endregion}
END_VAR
```

Name	Typ	Beschreibung
nEvtIncLight	ULINT	Zähler Lichttelegramme. Telegramme bestimmter Funktionsbausteine, z.B. FB_BA_LightingEvt [▶ 316] sind in einem Gebäude öfter instanziiert. Senden sie Telegramme gleicher Priorität, ist es wünschenswert, dass das zuletzt gesendete gültig ist. Dazu inkrementieren diese Bausteine jeweils den globalen Zähler *nEvtIncLight*. Telegrammselektoren des Typs FB_BA_LightingTgmSel4/FB_BA_LightingTgmSel8 [▶ 317] werten diesen Zähler aus und lassen das zuletzt gesendete Telegramm gelten.
nEvtIncSunBld	UDINT	Dieselbe Funktionalität wie *nEvtIncLight*, nur für Sonnenschutz. Die Telegrammselektoren sind hier: FB_BA_SunBldTgmSel4/FB_BA_SunBldTgmSel8 [▶ 317].
nNoActivePrio	UDINT	Der Wert der Konstante zeigt an, dass keine Priorität aktiv ist.

6.1.2.2.2.2 BA_Param

Die globale Parameterliste beinhaltet allgemeine Parameter um Objekte zu initialisieren.




```
VAR_GLOBAL CONSTANT
{region 'Sun Protection'}
  nSunPrt_MaxSunBldScn          : USINT := 20;
  nSunPrt_MaxRowFcd             : UINT := 10;
  nSunPrt_MaxColumnFcd          : UINT := 20;
  nSunPrt_MaxShdObj             : UINT := 20;
{attribute 'parameterUnit' := 'Byte'}
  nSunPrt_MaxDataFileSize       : UDINT := 100000;
{endregion}
  {region 'Universal'}
    {region 'Sequence'}
      nMaxSeqCtrl               : USINT := 8;
    {endregion}
    {region 'Plant Control'}
      nMaxPriority              : UDINT := 16;
      nMaxAggregate             : UDINT := 16;
```

```
      nNumberOfOpModes          : UDINT := 24;
    {endregion}
  {endregion}
  {region 'Objects'}
    {region 'Schedule'}
      nScheduleEntryCount       : UDINT := 6;
      nScheduleCalendarCount    : UDINT := 3;
      nScheduleExceptionCount   : UDINT := 24;
    {endregion}
  {endregion}
END_VAR
```

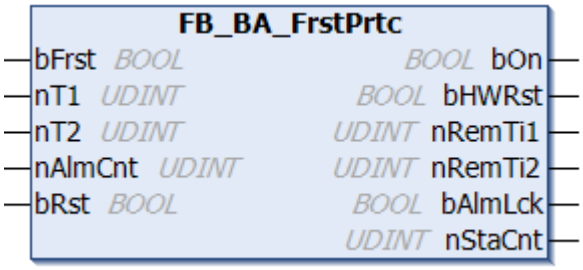
Name	Typ	Beschreibung
nSunPrt_MaxSunBldScn	USINT	Maximale Menge an Szenen, die von einem FB_BA_SunBldScn administriert werden können.
nSunPrt_MaxRowFcd	UINT	Maximale Anzahl an Etagen, die von der Verschattungskorrektur überwacht werden können (horizontale Ausrichtung der Fenster).
nSunPrt_MaxColumnFcd	UINT	Maximale Anzahl an Achsen, die von der Verschattungskorrektur überwacht werden können (vertikale Ausrichtung der Fenster).
nSunPrt_MaxShdObj	UINT	Maximale Anzahl an Verschattungsobjekten, die Schatten auf eine Fassade werfen.
nSunPrt_MaxDataFileSize	UDINT	Maximale Dateigröße [Byte] für zu lesende Excel-Listendurch die Bausteine FB_BA_RdFcdElemLst und FB_BA_RdShdObjLst.
nMaxSeqCtrl	USINT	Der globale Parameter gibt die Anzahl der Sequenzregler an. Er limitiert innerhalb der Struktur ST_BA_SeqLink [▶ 125] die Daten- und Befehlsstruktur *arrSeqLinkData*. Der Wert darf nicht kleiner 1 sein.
nMaxPriority	UDINT	Der globale Parameter gibt die maximale Anzahl an Prioritäten innerhalb der Anlagensteuerung FB_BA_PlantControl [▶ 491] an. Der Wert darf nicht kleiner 1 sein.
nMaxAggregate	UDINT	Der globale Parameter gibt die maximale Anzahl an Aggregaten innerhalb der Anlagensteuerung FB_BA_PlantControl [▶ 491] an. Der Wert darf nicht kleiner 1 sein.
nNumberOfOpModes	UDINT	Der globale Parameter gibt die maximale Anzahl an der Betriebsarten innerhalb der Anlagensteuerung FB_BA_PlantControl [▶ 491] an. Der Wert darf nicht kleiner 1 sein.
nScheduleEntryCount	UDINT	Der globale Parameter gibt die maximale Anzahl der Zeitplan-Einträge an, z. B. für einen Wochentag T_BA_ScheduleWeek [▶ 287] oder Ausnahmeeintrag T_BA_ScheduleExceptionEntryList [▶ 288]. Der Wert darf nicht kleiner 1 sein.
nScheduleExceptionCount	UDINT	Der globale Parameter gibt die maximale Anzahl der Ausnahmeeinträge an, z. B. T_BA_ScheduleExceptionList [▶ 288]. Der Wert darf nicht kleiner 1 sein.

6.1.2.2.3 POUs

6.1.2.2.3.1 FunctionBlocks

6.1.2.2.3.1.1 Air Conditioning

6.1.2.2.3.1.1.1 FB_BA_FrstPrtc



Der Funktionsbaustein FB_BA_FrstPrtc dient zur Frostüberwachung eines Heizregisters in einer raumlufttechnischen Anlage.

Frostgefahr steht an, wenn der Eingang *bFrst* TRUE ist. Der Frostalarm muss im Anlagenprogramm so verknüpft werden, dass die Anlage sofort abschaltet, das Erhitzerventil öffnet und die Erhitzerpumpe eingeschaltet wird.

Bei Frostgefahr wird der Ausgang *bOn* gesetzt und *nT1* (Sekunden) gestartet. Besteht nach Ablauf von *nT1* weiterhin Frostgefahr (*bFrst* = TRUE) dann bleibt *bOn* gesetzt. Ein Rücksetzen kann nur an dem Eingang *bRst* erfolgen.

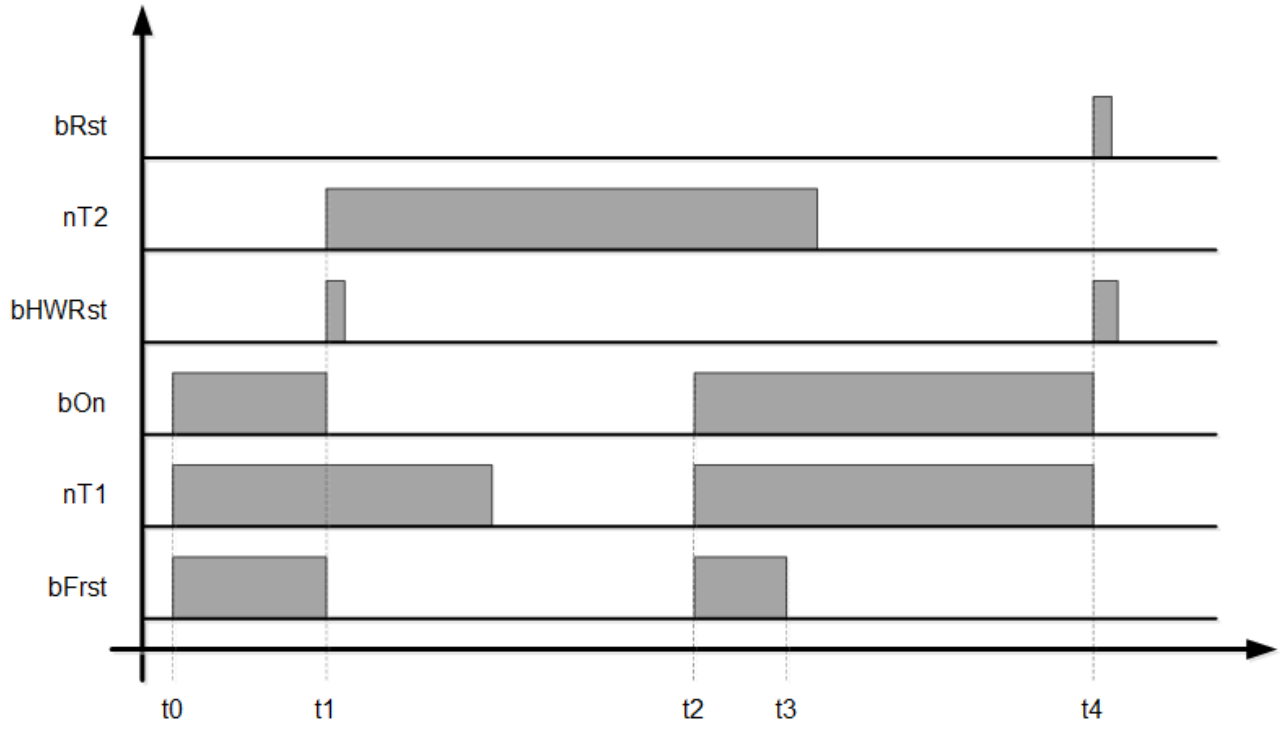
Erlischt der Frostalarm durch das Aufheizen des Heizregisters innerhalb des Zeitraums von *nT1* (*bFrst* = FALSE), so erfolgt ein automatischer Neustart der Anlage. Für den Anlagenneustart wird *bOn* FALSE und am Ausgang *bHWRst* wird ein Impuls zur Quittierung einer Selbsthaltungsschaltung im Schaltschrank ausgegeben. Mit dem Neustart wird eine zweite Überwachungszeit *nT2* (Sekunden) gestartet. Erfolgt innerhalb dieses Zeitraums erneut ein Frostalarm, so wird die Anlage endgültig verriegelt. *bOn* bleibt so lange gesetzt bis der Frostalarm beseitigt und an *bRst* quittiert ist.

Falls die Frostalarme aufeinander folgend, aber immer mit einem Zeitversatz größer als die Zeit von *nT2* erfolgen, würde theoretisch beliebig oft automatisch neu gestartet werden. Um dieses zu vermeiden, werden die Neustarts innerhalb des Bausteins gezählt. Mit *nAlmCnt* ist die Zahl der möglichen automatischen Neustarts zwischen 0 und 4 einstellbar.

Eine Quittierung am Eingang *bRst* setzt den Alarmspeicher innerhalb des Bausteins auf Null zurück.

Beispiel:

t0 = Frostalarm an Eingang *bFrst*, Alarmmeldung an Ausgang *bOn*, Starten des Timers T1 (*nT1* [s])
t1 = Frostalarm erloschen, Rücksetzen von *bOn*, Ausgabe Hardwareimpuls, Starten von Timer T2 (*nT2* [s]), Anlagenneustart
t2 = Erneuter Frostalarm innerhalb von T2, Alarmmeldung an *bOn*, Starten des Timers T1, verriegeln des Frostalarms
t3 = Frostalarm erloschen.
t4 = Quittierung des Alarms an *bRst*, Rücksetzen von *bOn*.



Eingänge

```
VAR_INPUT
  bFrst      : BOOL;
  nT1        : UDINT;
  nT2        : UDINT;
  nAlmCnt    : UDINT;
  bRst       : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bFrst	BOOL	Anschluss für die Frostereignisse luft- und wasserseitig.
nT1	UDINT	Timer zur Verzögerung von Neustarts [s]. Intern begrenzt auf einen Minimalwert von 0.
nT2	UDINT	Timer Überwachungszeit [s]. Intern begrenzt auf einen Minimalwert von 0.
nAlmCnt	UDINT	Maximale Zahl der automatischen Anlagenneustarts ohne Rücksetzen. Intern begrenzt auf Werte von 0 bis 4.
bRst	BOOL	Rücksetzen bzw. quittieren des Frostalarms.

Ausgänge

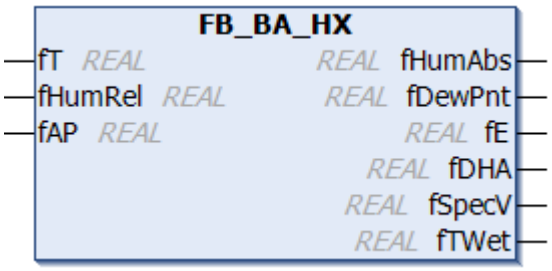
```
VAR_OUTPUT
  bOn        : BOOL;
  bHWRst     : BOOL;
  nRemTi1    : UDINT;
  nRemTi2    : UDINT;
  bAlmLck    : BOOL;
  nStaCnt    : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bOn	BOOL	Frostalarm aktiv
bHWRst	BOOL	Ausgabe eines Impulses zur Quittierung der Frostschutzhardware
nRemTi1	UDINT	Verbleibende Zeit bis Anlagenneustart nach Frostalarm.
nRemTi2	UDINT	Verbleibende Überwachungszeit.
bAlmLck	BOOL	Alarm Lock - gespeicherter Alarm.
nStaCnt	UDINT	Status Counter – aktuelle Anzahl nicht quittierter Fehlstarts.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.1.2 FB_BA_HX



Mit dem Funktionsbaustein FB_BA_HX werden die Taupunkttemperatur, die spezifische Enthalpie und die absolute Feuchte berechnet. Für die Berechnung der Größen werden die Temperatur, die relative Feuchte und der barometrische Luftdruck benötigt.
Die Enthalpie ist ein Maß für die Energie eines thermodynamischen Systems.

Eingänge

```
VAR_INPUT
  fT       : REAL;
  fHumRel  : REAL;
  fAP      : REAL;
END_VAR
```

Name	Typ	Beschreibung
fT	REAL	Temperatur [°C].
fHumRel	REAL	Relative Feuchte [%]
fAP	REAL	Hydrostatischer Luftdruck 1013.25 hPa.

Ausgänge

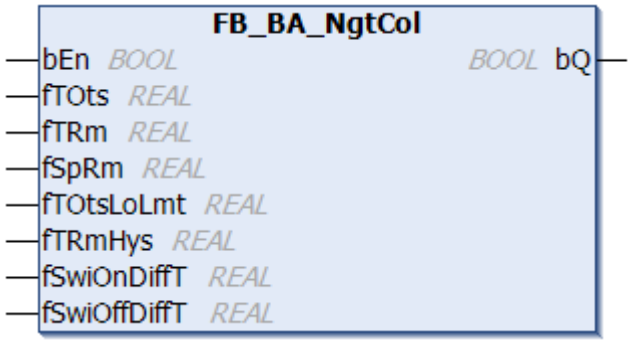
```
VAR_OUTPUT
  fHumAbs  : REAL;
  fDewPnt  : REAL;
  fE       : REAL;
  fDHA     : REAL;
  fSpecV   : REAL;
  fTWet    : REAL;
END_VAR
```

Name	Typ	Beschreibung
fHumAbs	REAL	Absolute Feuchte in g Wasser je kg trockener Luft [g/Kg].
fDewPnt	REAL	Taupunkttemperatur [°C]
fE	REAL	Enthalpie [kJ/kg]
fDHA	REAL	Dichte feuchter Luft ρ [kg Gemisch/m³]
fSpecV	REAL	Spezifisches Volumen [m³/kg]
fTWet	REAL	Feuchtkugeltemperatur [°C]

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.1.3 FB_BA_NgtCol



Der Funktionsbaustein FB_BA_NgtCol wird verwendet, um nachts mit kühler Außenluft die tags zuvor aufgeheizten Räume herunterzukühlen. Die Funktion der Sommernachtkühlung dient zur Verbesserung der Luftqualität und zur Einsparung elektrischer Energie. In den ersten Stunden des nächsten Sommertages wird elektrische Energie zur Kälteerzeugung gespart.

Durch Parametrierung des Funktionsbausteins FB_BA_NgtCol werden die Startbedingungen für die Sommernachtkühlung definiert. Der Baustein kann verwendet werden, um motorisch betätigte Fenster zu öffnen oder Klimaanlagen außerhalb ihrer normalen Betriebszeiten in den Sommernachtkühlbetrieb zu schalten.

Schaltbedingungen

Folgende Bedingungen müssen für die Aktivierung der Sommernachtkühlung erfüllt sein:

- Der Baustein selbst ist aktiviert (*bEn* = TRUE).
- Die Außentemperatur ist nicht zu niedrig (f*TOts* > f*TOtsLoLmt*).
- Die Außentemperatur ist im Vergleich zur Raumtemperatur ausreichend niedrig (f*TRm* - *fTOts*) > *fSwiOnDiffT.*
- Die Raumtemperatur ist so hoch, so dass es sich lohnt die Sommernachtkühlung einzuschalten. *fTRm* > *fSpRm* + *fTRmHys.*

Bei den folgenden Bedingungen wird die Sommernachtkühlung deaktiviert:

- Der Baustein selbst ist deaktiviert (*bEn* = FALSE).
- Die Außentemperatur ist zu niedrig *fTOts* < (*fTOtsLoLmt* - 0,5).
- Die Außentemperatur ist im Vergleich zur Raumtemperatur zu hoch (*fTRm* - *fTOts*) < *fSwiOffDiffT.*
- Die Raumtemperatur ist niedriger als der Sollwert. *fTRm* ≤ *fSpRm.*

Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  fTOts        : REAL;
```

```
  fTRm          : REAL;
  fSpRm         : REAL;
  fTOtsLoLmt    : REAL;
  fTOtsHys      : REAL;
  fTRmHys       : REAL;
  fSwiOnDiffT   : REAL;
  fSwiOffDiffT : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Funktionsbausteins.
fTOts	REAL	Außentemperatur [°C]
fTRm	REAL	Raumtemperatur [°C]
fSpRm	REAL	Raumtemperatur-Sollwert
fTOtsLoLmt	REAL	Untere Grenze der Außentemperatur [°C], verhindert ein zu starkes Auskühlen.
fTOtsHys	REAL	Hysterese für die minimale Außentemperatur [K]. Diese Hysterese, welche nach unten hin intern auf 0.5K begrenzt ist, soll ein Umschalten von bQ verhindern, falls die Außentemperatur genau um den Wert von rTOtsLoLmt pendelt.
fTRmHys	REAL	Hysterese für die Raumtemperatur [K]. Diese Hysterese, welche nach unten hin intern auf 0.5K begrenzt ist, soll ein unnötiges Umschalten von bQ verhindern, falls die Raumtemperatur genau um den Sollwert rSpRm pendelt.
fSwiOnDiffT	REAL	Differenz zwischen der Raumtemperatur und der Außentemperatur ab der die Sommernachtkühlung freigegeben wird [K].
fSwiOffDiffT	REAL	Differenz zwischen der Raumtemperatur und der Außentemperatur ab der die Sommernachtkühlung gesperrt wird [K]. Intern wird überprüft, dass die Differenz zwischen (*fSwiOnDiffT* - *fSwiOffDiffT*) > 0.5 ist.

Ausgänge

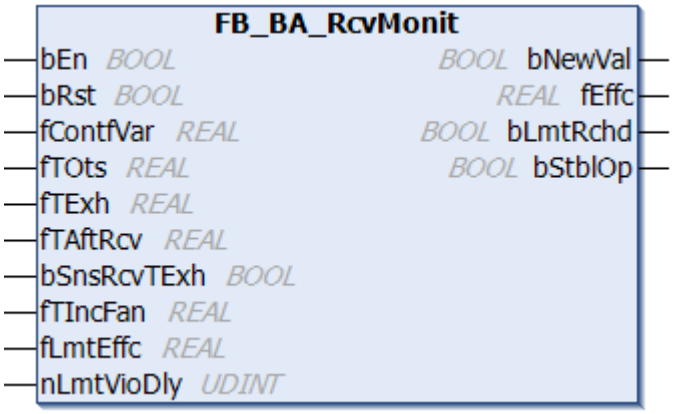
```
VAR_OUTPUT
  bQ   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bQ	BOOL	Sommernachtkühlung Ein

Voraussetzungen

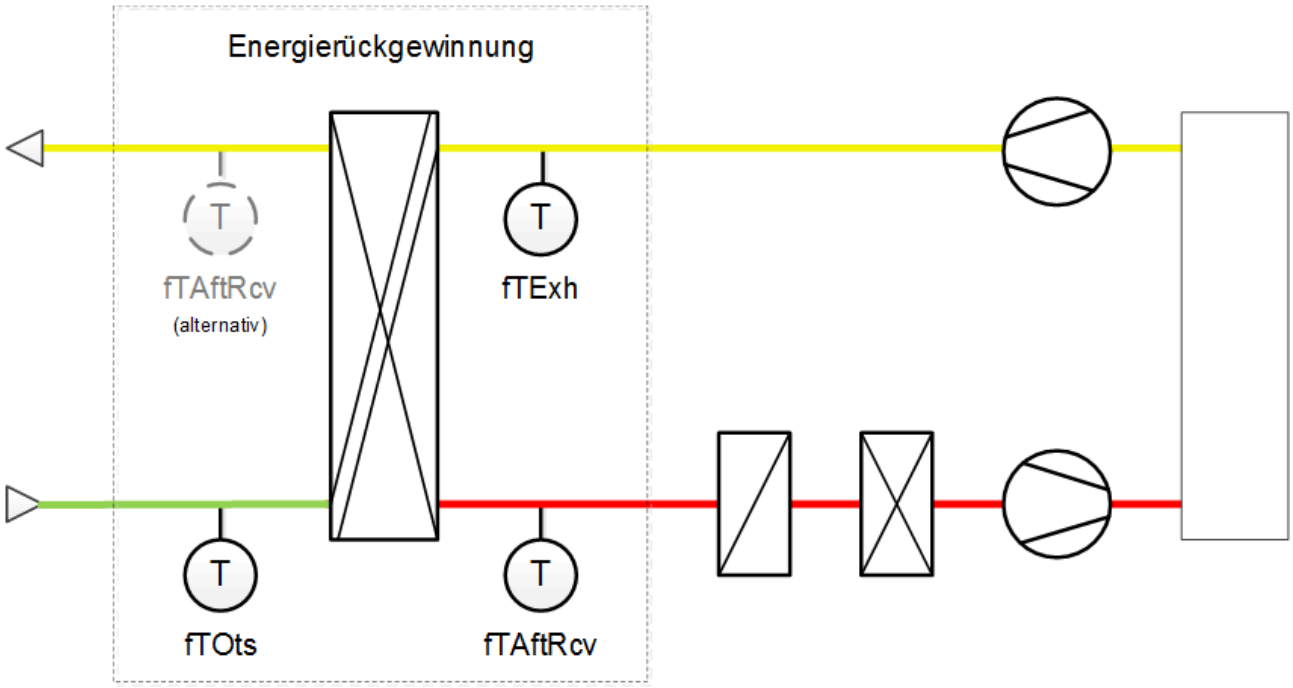
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.1.4 FB_BA_RcvMonit



Der Funktionsbaustein FB_BA_RcvMonit dient zur Berechnung des Wirkungsgrades einer Energie-Rückgewinnung.
Der Baustein benötigt zur Berechnung des Wirkungsgrades (der sogenannten Rückwärmzahl) folgende Temperaturmesswerte:

- Außenlufttemperatur *fTOts*
- Fortlufttemperatur *fTExh*
- Lufttemperatur der Energie-Rückgewinnung im Zuluftkanal (alternativ: im Abluftkanal) *fTAftRcv*



Der Baustein nimmt nun alle 10 s die Temperatur-Messwerte auf und bildet aus jeweils 6 Werten den minütlichen Mittelwert. Danach wird geprüft, ob die Anlage sich in einem eingeregelten, "stabilen" Zustand befindet:

- Die aufgenommenen Temperaturen Außenluft, Fortluft und Luft nach Energierückgewinnung sind annähernd konstant, d.h. keiner der 6 Einzelwerte weicht mehr als 0,5 K vom jeweiligen Mittelwert ab.
- Die Temperaturdifferenz Außenluft zu Fortluft beträgt mindestens 5 K.

Ist dies der Fall, so wird dieser Messzyklus mit einem TRUE-Signal am Ausgang *bStblOp* quittiert und der ermittelte Wirkungsgrad an *fEffc* ausgegeben. Ist der Zustand hingegen nicht "stabil", so erscheint am Ausgang *bStblOp* ein FALSE-Signal und *fEffc* wird auf 0 gesetzt.

In jedem Fall jedoch wird jeder Mess- und Auswerte-Zyklus mit einem Trigger (ein Ein-PLC-Zyklus langes TRUE-Signal) an *bNewVal* als beendet gekennzeichnet.

Freigabe (*bEn*) und Reset (*bRst*)

Der Baustein arbeitet nur dann, wenn ein TRUE-Signal an *bEn* anliegt. Andernfalls stoppt er seine Abarbeitung und alle Ausgänge werden auf FALSE bzw. 0.0 gesetzt.

Während der Abarbeitung kann ein Mess-und Auswertezyklus jederzeit durch ein TRUE-Signal an *bRst* abgebrochen werden. Alle Ausgänge werden auf FALSE bzw. 0.0 gesetzt und der Messzyklus startet automatisch neu.

Auswahl des Temperaturwertes "nach Rückgewinnung" (*bSnsRcvTExh*)

Ein FALSE-Eintrag an *bSnsRcvTExh* bedeutet, dass zur Berechnung des Wirkungsgrades die Temperaturmessung nach der Wärmerückgewinnung im **Zuluftkanal** verwendet wird.
Soll hingegen die Temperaturmessung nach der Wärmerückgewinnung im **Abluftkanal** angewendet werden, so muss an *bSnsRcvTExh* ein TRUE angelegt werden.

Grenzwertüberschreitung (*fContrVar, fLmtEffc, bLmtRchd*)

Eine Grenzwertüberschreitung liegt dann vor, wenn der ermittelte Wirkungsgrad kleiner ist als der eingetragene Grenzwert *fLmtEffc* und gleichzeitig die Stellgröße für die Wärmerückgewinnung bei 100 % liegt. Dazu muss die Stellgröße mit dem Eingang *fContrVar* verbunden werden.
Die Meldung der Grenzwertüberschreitung lässt sich durch einen Eintrag an *nLmtVioDly_sec* [s] verzögern: Liegen die beiden Kriterien, Überschreitung und Übersteuerung länger als *nLmtVioDly _sec* [s] vor, so wird dieses mit einem TRUE-Signal an *bLmtRchd* angezeigt.
Eine aufgetretene Warnmeldung erlischt, wenn ein kompletter Messzyklus "gute" Werte erbringt oder bei einer steigenden Flanke an *bRst* bzw. einer Deaktivierung des Bausteines.



Diese Warnmeldung tritt nur auf, wenn die Anlage in einem stabilen Betrieb ist (*bStblOp*=TRUE).

Berücksichtigung Temperaturanstieg der Abluft durch Ventilatormotor (*fTIncFan*)

Es ist möglich, dass die Abluft durch einen Ventilatormotor erwärmt und die Messung damit verfälscht wird. Diese Temperaturerhöhung kann durch *fTIncFan* angegeben werden. Intern wird dann die gemessene Ablufttemperatur um diesen Wert gemindert.

Eingänge

```
VAR_INPUT
  bEn               : BOOL;
  bRst              : BOOL;
  fContrVar         : REAL;
  fTOts             : REAL;
  fTExh             : REAL;
  fTAftRcv          : REAL;
  bSnsRcvTExh       : BOOL;
  fTIncFan          : REAL;
  fLmtEffc          : REAL;
  nLmtVioDly        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Funktionsbausteins.
bRst	BOOL	Reset - alle ermittelten Werte werden gelöscht.
fContrVar	REAL	Stellgröße für die Wärmerückgewinnung, d.h. ihr Istwert.
fTOts	REAL	Außentemperatur [°C]
fTExh	REAL	Fortlufttemperatur [°C]
fTAftRcv	REAL	Temperatur nach Energierückgewinnung
bSnsRcvTExh	BOOL	Temperatur am Messort nach Energierückgewinnung: FALSE -> im Zuluftkanal (SupplyAir) - TRUE -> im Abluftkanal (ExhaustAir).
fTIncFan	REAL	Temperaturanstieg durch Ventilator.
fLmtEffc	REAL	Wirkungsgrad Grenzwert
nLmtVioDly	UDINT	Zeitverzögerung Grenzwertüberschreitung [s]. Intern begrenzt auf einen Minimalwert von 0.

Ausgänge

```
VAR_OUTPUT
  bNewVal      : BOOL;
  fEffc        : REAL;
  bLmtRchd     : BOOL;
  bStblOp      : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bNewVal	BOOL	Ausgabetrigger neuer Wert fEffc
fEffc	REAL	Effektivität
bLmtRchd	BOOL	Grenzwert erreicht
bStblOp	BOOL	Stabiler Betrieb

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.1.5 FB_BA_SpSupVis



Der Funktionsbaustein FB_BA_SpSupVisdient zur Verarbeitung und Kontrolle des unteren und oberen Sollwertes einer Zuluftfeuchte- oder Zulufttemperaturregelung.

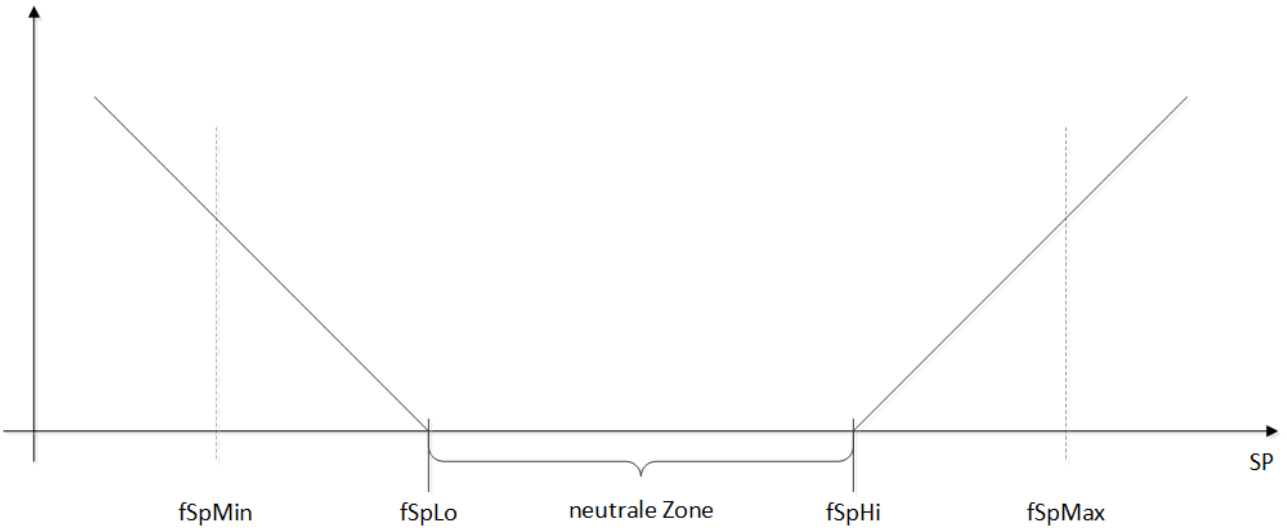
Überprüfung und Limitierung der Sollwerte

Der Funktionsbaustein limitiert und begrenzt die Sollwerte. Die beiden folgenden Tabellen stellen dar, was überprüft wird und wie im Fehlerfall reagiert wird.

Überprüfung	Aktion
fSpLo > fSpHi	Letzten gültigen Werte von fSpLo und fSpHi werden verwendet.
fSpMin >= fSpMax	Letzten gültigen Werte von fSpMin und fSpMax werden verwendet.
fSpHi > fSpMax	fPrSpHi = fSpMax
fSpLo < fSpMin	fPrSpLo = fSpMin

Überprüfung	bErr	Aktion
fSpMin >= fSpMax	TRUE	fSpErr = ((fSpMin + fSpMax) / 2) fPrSpHi = fPrSpLo = fPrRcv = fSpErr
fSpHi < fSpMin		
fSpLo > fSpMax		

Der Abstand zwischen den Sollwerten beschreibt eine Energieneutrale Zone. Bei der Verwendung in einer Zuluftregelung würde innerhalb der Neutralen Zone nicht geheizt und nicht gekühlt werden.



Die überprüften und gegebenenfalls limitierten Sollwerte werden am Bausteinausgang als f*PrSpHi* und *fPrSpLo* (Present Setpoint) ausgegeben.

Sollwert für Wärmerückgewinnung

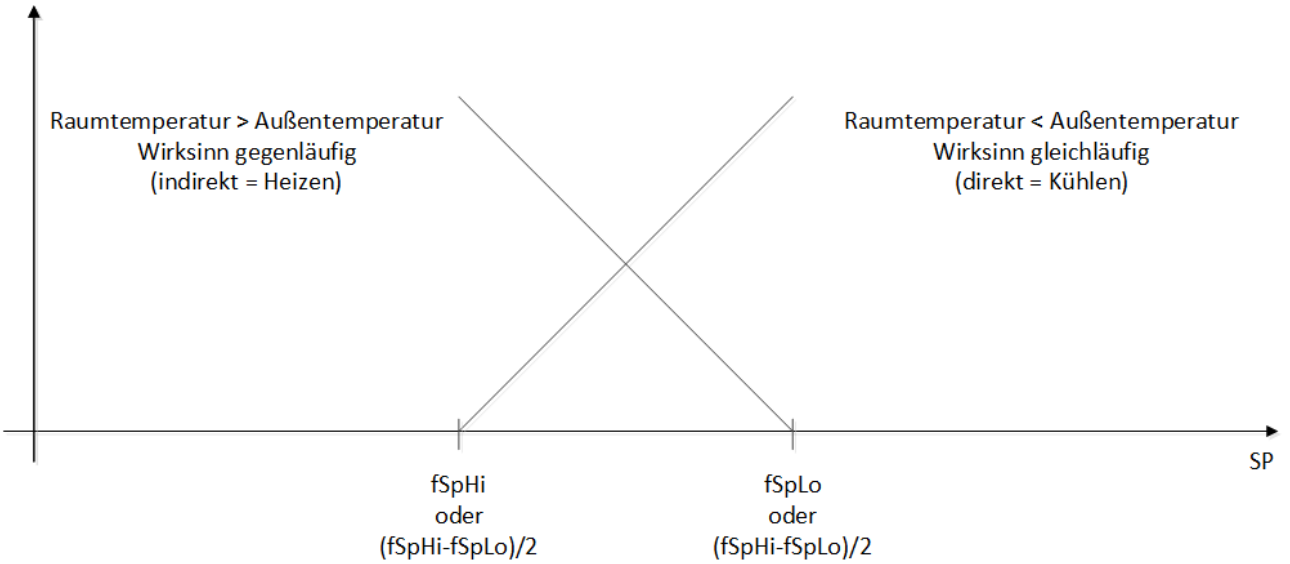
Für eine Wärmerückgewinnung wird der Sollwert *fSpRcv* wahlweise aus dem Mittelwert des oberen und unteren Sollwertes, *fSpHi* und *fSpLo* berechnet oder aber in Abhängigkeit vom Wirksinn der Wärmerückgewinnung (WRG).
Die Methode wird durch die Eingangsvariable *bSlcnSpRcv* definiert:

b SlcnSpRcv	fSpRcv
TRUE	Mittelwert aus *fSpLo* und *fSpHi*
FALSE	Wirksinnabhängig, definiert durch Eingang bActRcv

Im Falle der wirksinnabhängigen Definition des Sollwertes gilt dann:

bActRcv	Wirksinn	fSpRcv
TRUE	direkt (Kühlen)	*fSpHi*
FALSE	indirekt (Heizen)	*fSpLo*

Wärmerückgewinnung (WRG)



Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  fSpHi        : REAL;
  fSpLo        : REAL;
  fSpMax       : REAL;
  fSpMin       : REAL;
  bActnRcv     : BOOL;
  bSlcnSpRcv   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Bausteinfreigabe. Wenn bEn = FALSE, stehen alle Ausgabeparameter auf 0.0.
fSpHi	REAL	Zu überprüfender Eingabewert des oberen Sollwertes.
fSpLo	REAL	Zu überprüfender Eingabewert des unteren Sollwertes.
fSpMax	REAL	Maximaler Sollwert
fSpMin	REAL	Minimaler Sollwert.
bActnRcv	BOOL	Wirkrichtung der nachgeschalteten Wärmerückgewinnung.
bSlcnSpRcv	BOOL	Sollwert-Wahl der nachgeschalteten Wärmerückgewinnung.

Ausgänge

```
VAR_OUTPUT
  fPrSpHi     : REAL;
  fPrSpLo     : REAL;
  fSpRcv      : REAL;
  bErr        : BOOL;
  sErrDescr   : T_MAXSTRING;
END_VAR
```

Name	Typ	Beschreibung
fPrSpHi	REAL	Ausgabewert des oberen Sollwertes.
fPrSpLo	REAL	Ausgabewert des unteren Sollwertes.
fSpRcv	REAL	Ausgabewert des resultierenden Sollwertes für die Wärmerückgewinnung.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

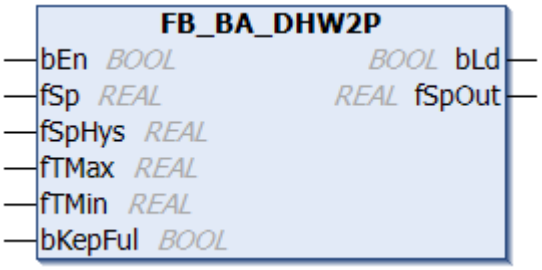
Fehlerbeschreibung
01: Warnung: Die Sollwerte sind nicht in einer logischen Reihenfolge: Entweder (*fSpMin* >= *fSpMax*) OR (*fSpHi* < *fSpMin*) OR (*fSpLo* > *fSpMax*)

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.2 Domestic Hot Water

6.1.2.2.3.1.2.1 FB_BA_DHW2P

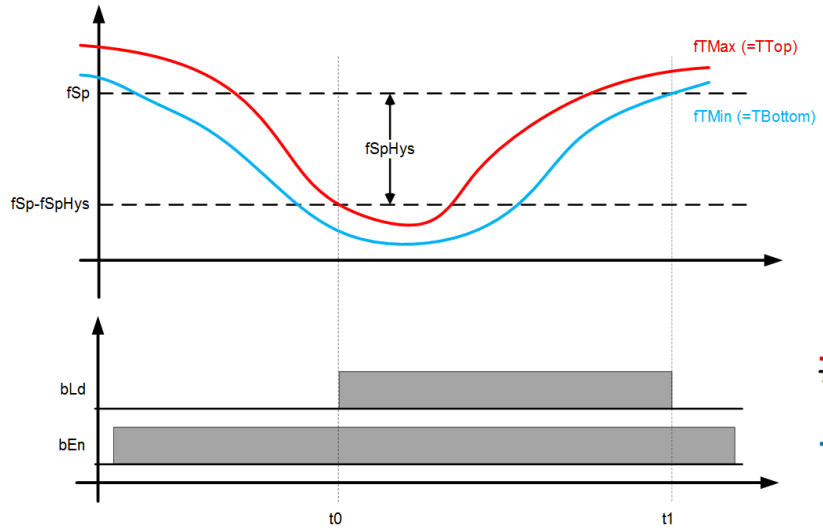


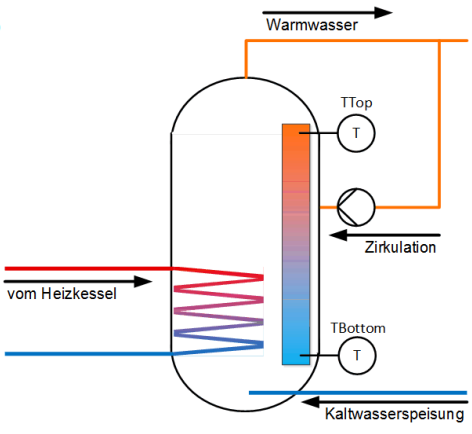
Der Funktionsbaustein FB_BA_DHW2P steuert die Ladung eines Warmwasserspeichers mittels eines Zweipunkt-Reglers. An dem Eingang *bEn* wird die Speicherladung freigegeben. Bei aktiver Speicherladung ist der Ausgang *bLd* TRUE. Mit der Variablen *fSp* wird dem Funktionsbaustein der Sollwert für die Brauchwassertemperatur übergeben. An dem Eingang *fTMin* wird eine Minimalauswahl, am Eingang *fTMax* eine Maximalauswahl aller Temperaturfühler des Warmwasserspeichers angeschlossen.
Bedingt durch die Temperaturschichtung im Warmwasserspeicher ist der oberste Fühler im Allgemeinen der mit der höchsten Temperatur und der untere derjenige mit der niedrigsten.

Die Speicherladung kann mittels der Variablen *bKepFul* in zwei Arten erfolgen:

bKepFul = FALSE

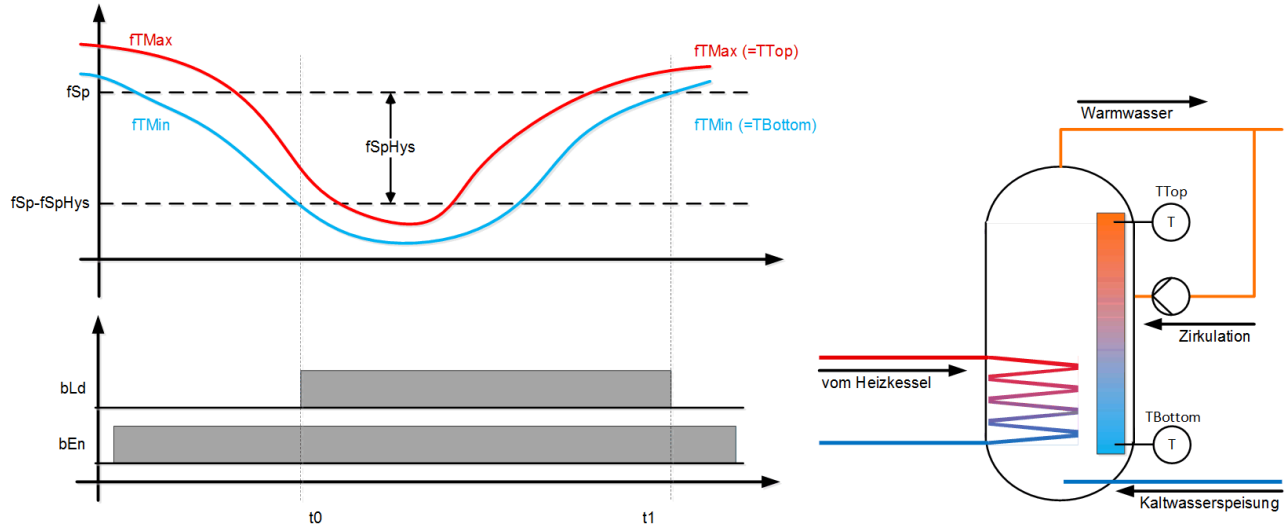
Die Ladung wird angefordert, wenn *fTMax* unter dem Wert von *fSp-fSpHys* liegt. Die Ladungsanforderung wird deaktiviert, wenn *fTMin* über dem Sollwert von *fSp* liegt.
Dadurch, dass im Allgemeinen der oberste Fühler die höchste Temperatur misst, erfolgt eine Einschaltung der Speicherladung erst bei entladenem Warmwasserspeicher.





bKepFul = TRUE

Die Ladung wird angefordert, wenn *fTMin* unter dem Wert von *fSp-fSpHys* liegt. Die Ladungsanforderung wird deaktiviert, wenn *fTMin* wieder über dem Sollwert ist.
Durch die Minimalauswahl aller Speichertemperaturen, wird der kälteste Punkt des Speichers für die Regelung verwendet. Eine Nachladung erfolgt, sobald der Speicher nicht mehr ganz gefüllt ist.



Eingänge

```
VAR_INPUT
  bEn       : BOOL;
  fSp       : REAL;
  fSpHys    : REAL;
  fTMax     : REAL;
  fTMin     : REAL;
  bKepFul   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe Boiler Ladung
fSp	REAL	Temperatursollwert des Brauchwassers [°C].
fSpHys	REAL	Hysterese, empfohlen 1°K bis 5°K.
fTMax	REAL	Maximalauswahl aller Speichertemperaturfühler [°C].
fTMin	REAL	Minimalauswahl aller Speichertemperaturfühler [°C].
bKepFul	BOOL	Steuertemperaturanwahl: FALSE = mit *fTMax* wird *bLd* gefordert, mit *fTMin* ausgeschaltet. TRUE = *fTMin* allein steuert das Ein-Ausschalten von *bLd*.

Ausgänge

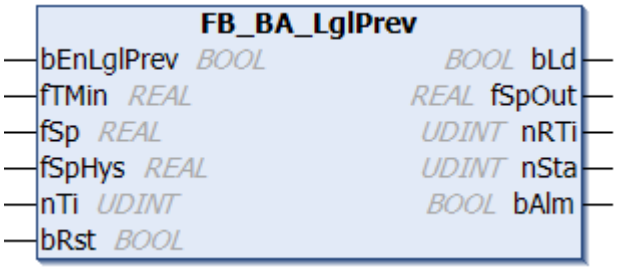
```
VAR_OUTPUT
  bLd     : BOOL;
  fSpOut  : REAL;
END_VAR
```

Name	Typ	Beschreibung
bLd	BOOL	Freigabe des Ladebetriebes.
fSpOut	REAL	Sollwertweitergabe an Ladeschaltung: *fSpOut* = *fSp* (Eingang) wenn der Baustein aktiviert ist. *fSpOut* = 0 wenn der Baustein nicht aktiviert ist.

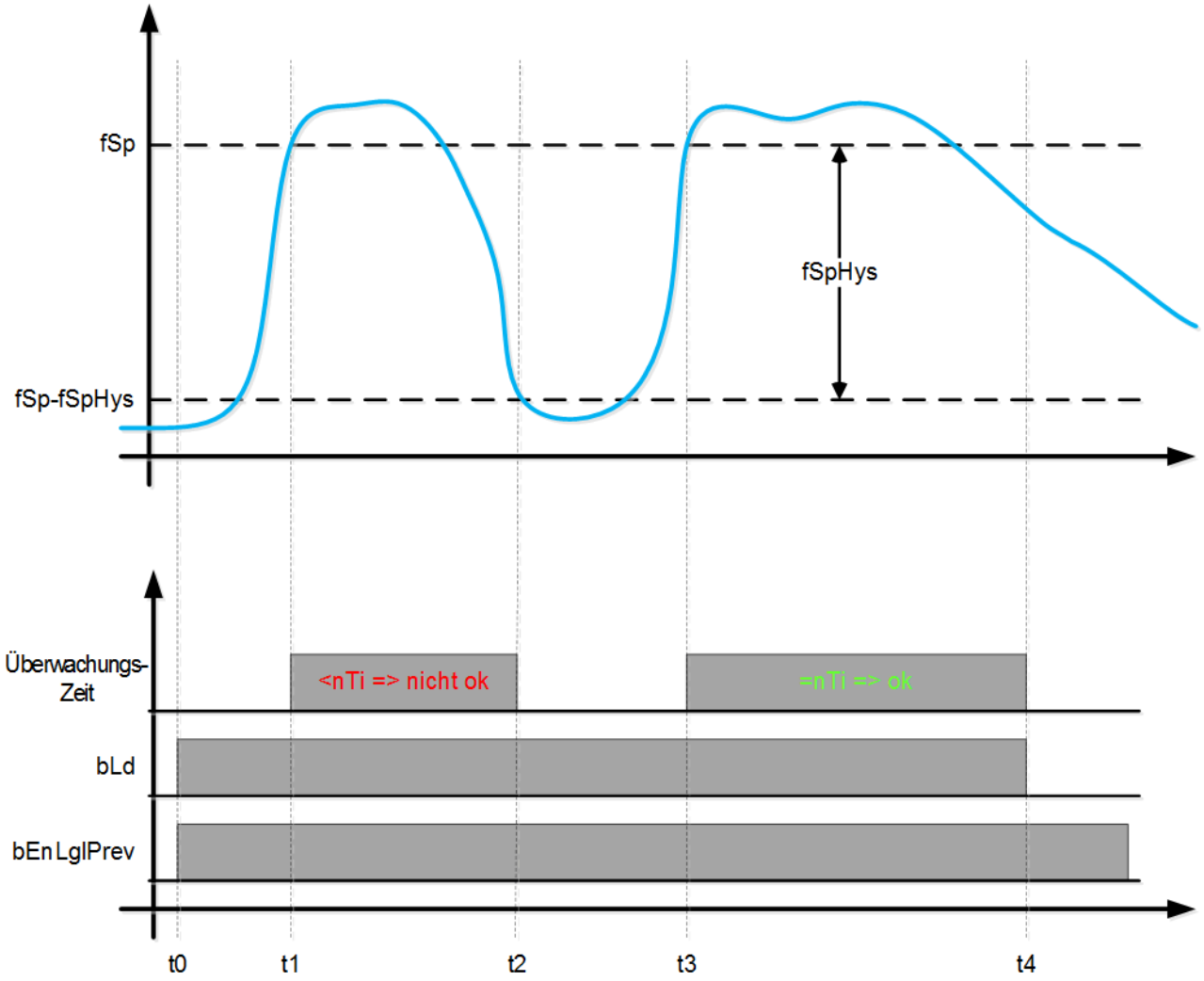
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.2.2 FB_BA_LglPrev



Der Funktionsbaustein FB_BA_LglPrev dient zur Desinfektion des Brauchwassers und zum Abtöten von Legionellen. Die Freigabe des Desinfektionsbetriebes erfolgt an dem Eingang *bEnLglPrev* durch die Anschaltung eines Zeitschaltprogrammes. Zu empfehlen ist die Durchführung der Desinfektion mindestens einmal in der Woche (nachts). Die Temperatur sollte mindestens 70°C betragen. Das Einschaltintervall an *bEnLglPrev* muss dafür ausreichend lang gewählt werden. Der Ausgang *bLd* aktiviert dabei die Speicherladung.
Bei Warmwasserspeichern mit mehreren Temperaturfühlern muss an *fTMin* eine Min-Auswahl aller Fühler angeschlossen werden.
Überschreitet *fTMin* den Wert von *fSp,* wird ein Überwachungstimer mit einer Zeit von *nTi_sec* [s] gestartet. Bleibt die minimale Speichertemperatur *fTMin* bis zum Ablauf des Timers oberhalb von *fSp - fSpHys* ist eine ausreichende Aufheizung des Speichers erfolgt. Bei einer vorhandenen Zirkulation muss der Ausgang *bLd* mit der Freigabe der Zirkulationspumpe verknüpft sein, damit auch die Wasserleitung innerhalb des Brauchwassersystems desinfiziert wird. Ist die Temperatur jedoch während des Desinfektionsprozesses unter *fSp -fSpHys* gefallen, muss der Desinfektionsprozess neu gestartet werden bis die Zeit von *nTi* einmal komplett abgelaufen ist. Bei einer erfolgreichen Desinfektion wird der Ausgang *bLd* zurückgesetzt.
Wurde während der Bausteinaktivierung (*bEnLglPrev)* kein vollständiger Desinfektionsprozess abgeschlossen, wird dieses mit dem Ausgang *bAlm* signalisiert. Der Ausgang muss mit *bRst* zurückgesetzt werden.



Erläuterung zur Grafik:

t0 Starten des Legionellenprogramms und Schalten des Ausgangs *bLd*. Aufheizen des Warmwasserspeichers.

t1 Der Speicher hat die Temperatur *fSp* erreicht. Der Timer für die Aufheizzeit wird gestartet.

t2 Die minimale Speichertemperatur ist unterhalb von f*Sp -fSpHys* gefallen. Der Timer für die Aufheizzeit wird wieder zurückgesetzt.

t3 Die Temperatur steigt erneut über *fSp* und der Aufheiztimer wird wieder gestartet.

t4 Die Minimale Speichertemperatur war über den Zeitraum von *nTi* hinweg oberhalb der Grenze *fSp - fSpHys*, die Desinfektion war erfolgreich. *bLd* wird zurückgesetzt und der Warmwasserspeicher geht wieder in den Normalbetrieb.

Eingänge

```
VAR_INPUT
  bEnLglPrev  : BOOL;
  fTMin       : REAL;
  fSp         : REAL;
  fSpHys      : REAL;
  nTi         : UDINT;
  bRst        : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bEnLglPrev	BOOL	Freigabe des Desinfektionsbetriebes über ein Zeitschaltprogamm.
fTMin	REAL	Minimale Speichertemperatur [°C]. Minimalauswahl von Temperaturfühler oben und Temperaturfühler unten.
fSp	REAL	Sollwert Desinfektion [°C]
fSpHys	REAL	Temperaturspreizung [°K] untere Grenze - wird immer absolut gerechnet.
nTi	UDINT	Überwachungszeitraum [s].
bRst	BOOL	Rücksetzen des Legionellen Alarms.

Ausgänge

```
VAR_OUTPUT
  bLd     : BOOL;
  fSpOut  : REAL;
  nRTi    : UDINT;
  nSta    : UDINT;
  bAlm    : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bLd	BOOL	Anti-Legionellenbetrieb aktiv.
fSpOut	REAL	Sollwertweitergabe an Ladeschaltung: *fSp* (Eingang), wenn der Baustein aktiviert ist 0, wenn der Baustein nicht aktiviert ist
nRTi	UDINT	Countdown Timer Desinfektionsbetrieb.
nSta	UDINT	Status Desinfektionsprogramm: 1. Desinfektionsbetrieb erfolgreich. 2. Desinfektion erfolgreich abgeschlossen. Zum Abschluss und für eine neue Aktivierung der Legionellenvorsorge muss *bEnLglPrev* FALSE sein. 3. Desinfektionsbetrieb aktiv. 4. Desinfektion nicht erfolgreich. Alarm steht an. 5. Desinfektion nicht erfolgreich, der Alarm wurde quittiert. 6. Neustart der Steuerung bzw. es wurde noch kein Legionellenbetrieb angefordert.
bAlm	BOOL	Temperatursollwert wurde nicht durchgängig über den Zeitraum von nTi hinweg erreicht, sodass eine ausreichende Desinfektion nicht gewährleistet ist.

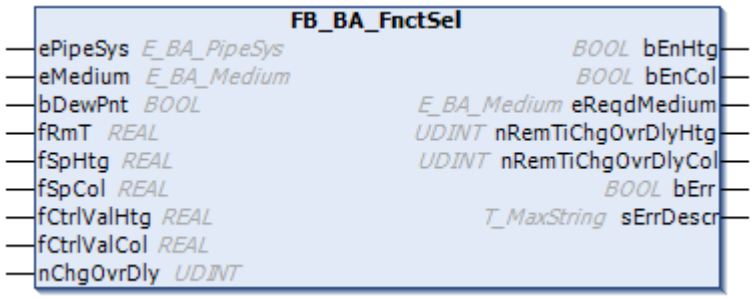
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3 Room Automation

6.1.2.2.3.1.3.1 Heating Cooling Functions

6.1.2.2.3.1.3.1.1 FB_BA_FnctSel



Der Funktionsbaustein FB_BA_FnctSel dient der Freigabe des Heiz- oder Kühlbetriebs in einem Raum. Dabei spielt die Art des Verteilungsnetzes eine große Rolle:
Handelt es sich um ein Zweileitersystem, so können alle Räume der Anlage immer nur entweder beheizt oder gekühlt werden.
In einem Vierleitersystem hingegen kann die Klimatisierung der Räume bedarfsgerecht erfolgen, d.h. ein Teil der Räume kann beheizt, der andere von derselben Anlage gekühlt werden.

Der Funktionsbaustein, der wie bereits erwähnt pro Raum Anwendung findet, wählt damit je nach vorhandenem Rohrleitungssystem seine Regler unterschiedlich an:

Zweileiternetz

Das Zweileitersystem ist angewählt, wenn am Eingang der Bausteins *ePipeSys.e2Pipe* eingestellt ist. Da alle Räume der Anlage nur entweder beheizt oder gekühlt werden können, wird die Anwahl von zentraler Stelle für alle Räume gleich über den Eingang *eMedium* vorgegeben. Ist *eMedium* FALSE, so wird der Raum-Heizregler angewählt - ist der Eingang TRUE, so ist es der Kühlregler. Die Reglerfreigaben *bEnHtg* und *bEnCol* werden immer gegenseitig verzögert um *nChgOvrDly* [s] ausgegeben, das heißt: erst wenn die Freigabe Kühlen *bEnCol* für *nChgOvrDly* FALSE ist, kann die Freigabe Heizen gegeben werden und umgekehrt. Zusätzlich zum Ablauf dieser Umschaltzeit wird darauf geachtet, dass der abzuschaltende Regler 0.0 ausgibt. Dies geschieht durch Rückmeldung an den Eingängen *fCtrlValHtg* und *fCtrlValCol*. So wird ein drastischer Wechsel von Heizen auf Kühlen und umgekehrt vermieden.

Vierleiternetz

Das Vierleitersystem ist angewählt, wenn am Eingang der Bausteins *ePipeSys.e4Pipe* eingestellt ist. Die Regleranwahl kann nun für alle Räume nach Bedarf unterschiedlich sein und richtet sich daher nach der Raumtemperatur *fRmT* und den eingestellten Sollwerten *fSpHtg* für Heizen und *fSpCol* für Kühlen. Übersteigt die Raumtemperatur den Kühlsollwert, so wird der Kühlregler freigegeben (*bEnCol*), fällt sie unter den Heizsollwert, wird der Heizregler freigegeben (*bEnHtg*). Befindet sich die Temperatur zwischen beiden Sollwerten, so sind beide Regler abgeschaltet (energieneutrale Zone). Auch hier werden die Reglerfreigaben *bEnHtg* und *bEnCol* immer gegenseitig verzögert um *nChgOvrDly* [s] ausgegeben (siehe Zweileiternetz) und zusätzlich zum Ablauf dieser Umschaltzeit darauf geachtet, dass der abzuschaltende Regler 0.0 ausgibt. Dies geschieht durch Rückmeldung an den Eingängen *fCtrlValHtg* und *fCtrlValCol*. So wird ein drastischer Wechsel von Heizen auf Kühlen und umgekehrt vermieden, falls die Umschaltzeit nicht ausreichend hoch bemessen ist.

Taupunktwächter (bDewPnt)

In beiden Systemen - Zwei- und Vierleiter - hat der Taupunktwächter die Aufgabe, das Kühlen unmittelbar zu verriegeln.

Programmablauf

Der Baustein kennt 3 Zustände:

1. Warten auf Freigabe Heizen bzw. Kühlen

2. Freigabe Heizen
3. Freigabe Kühlen

Im ersten Schritt wartet der Baustein auf die Erfüllung der Bedingungen, die für das Heizen bzw. das Kühlen notwendig sind:

Heizen	Kühlen
Ausgabe Kühlregler = 0 (*fCtrlValCol*)	Ausgabe Heizregler = 0 (*fCtrlValHtg*)
Raumtemperatur (*fRmT*) < Sollwert Heizen (*fSpHtg*)	Raumtemperatur (*fRmT*) > Sollwert Kühlen (*fSpCol*)
Reglerfreigabe Kühlen (*bEnCol*) steht mindestens für die Umschaltzeit *nChgOvrDly* [s] auf FALSE	Reglerfreigabe Heizen (*bEnHtg*) steht mindestens für die Umschaltzeit *nChgOvrDly* [s] auf FALSE
Vierleitersystem ist angewählt (*ePipesys* = *E_BA_PipeSys.4Pipe*) oder Zweileitersystem ist angewählt und es steht Heizmedium an (*ePipesys* = *E_BA_PipeSys.2Pipe* AND bMedium = FALSE)	Vierleitersystem ist angewählt (*ePipesys* = *E_BA_PipeSys.e4Pipe*) oder Zweileitersystem ist angewählt und es steht Kühlmedium an (*ePipesys* = *E_BA_PipeSys.e2Pipe* AND bMedium = TRUE)
	Der Taupunktwächter spricht nicht an (*bDewPnt* = TRUE)

Ist nun eine Bedingungskette erfüllt, springt der Baustein in den jeweiligen Zustand Heizen oder Kühlen und beendet diesen erst, wenn der entsprechende Regler am Bausteineingang (*fCtrlValHtg*/*fCtrlValCol*) 0 ausgibt. Damit ist sichergestellt, dass immer nur ein Regler aktiv ist, auch wenn eine zu hohe Reglerausgabe, z. B. Heizen, u. U. ein kurzfristiges Kühlen erforderlich machen würde (Überschwingen). Das Heizen, bzw. Kühlen wird dann erst abgeschlossen, wenn kein Bedarf mehr ansteht.

Allerdings gibt es 3 Ausnahmen, bei denen das Heizen bzw. Kühlen unmittelbar unterbrochen wird:

1. Im Zweileitersystem (*ePipeSys* = *E_BA_PipeSys.2Pipe*) wird geheizt (*bEnHtg*), es wurde jedoch auf Kühlmedium umgestellt (*eMedium* = *E_BA_Medium.eCoolMedium*)
2. Im Zweileitersystem (*ePipeSys* = *E_BA_PipeSys.2Pipe*) wird gekühlt (*bEnCol*), es wurde jedoch auf Heizmedium umgestellt (*eMedium* = *E_BA_Medium.eHeatMedium*)
3. Im Kühlbetrieb (Zwei- oder Vierleitersystem) hat der Taupunktwächter angesprochen (*bDewPnt*=TRUE)

In diesen Fällen werden die Heiz- und Kühlfreigaben weggenommen und zurück in den Wartezustand gesprungen.

Ist eine der beiden Reglerfreigaben gegeben und der entsprechende Regler reagiert nicht, das heißt er bleibt für die Zeit *nChgOvrDly* [s] weiterhin auf "0",
so springt der Baustein selbstständig in den ersten Schritt "Warten auf Freigabe Heizen bzw. Kühlen" zurück.

Es handelt sich hierbei um eine Notfunktion, falls ein Temperatursensor springt und eine falsche Auswahl getroffen wird, die dann durch den gewählten Regler nicht erfüllt werden kann. Ein Beispiel wäre, wenn beim SPS-Start eine Sensorfunktion 0° ausgibt, dadurch Heizen angewählt wird und die Sensorfunktion dann aufgrund einer programmierten SPS-Startverzögerung einen Temperaturwert annimmt, der den Kühlregler fordert. Ohne diese Notfunktion würde vergeblich gewartet werden, dass der Heizregler einen Wert größer "0" annimmt.

Bedarfsmitteilung (*eReqdMedium*)

Um der Anlage mitzuteilen, wie der aktuelle Bedarf an Wärme oder Kühlung ist, wird am Bausteinausgang, also pro Raum, in Abhängigkeit der Ist- und Solltemperatur eine Bedarfskennung ausgegeben. Diese kann zentral gesammelt und ausgewertet werden. Die Auswertung erfolgt immer, also unabhängig vom vorhandenen Netzsystem (Zwei- oder Vierleiter).

eReqdMedium	Medium	Raumtemperatur
1	Kein Medium ist angefordert	*fRmT* > *fSpHtg* UND *fRmT* < *fSpCol*
2	Heizmedium ist angefordert	*fRmT* < *fSpHtg*
3	Kühlmedium ist angefordert	*fRmT* > *fSpCol*

Fehlerbehandlung

Der Heiz-Sollwert darf nicht größer oder gleich dem Kühl-Sollwert sein, da damit ein Temperaturbereich entstehen würde, in dem gleichzeitig Heiz- und Kühlbedarf entsteht. Da der Baustein jedoch immer nur eine Freigabe, Heizen oder Kühlen ausgibt, ist der Fall anlagentechnisch unbedenklich. Es wird für diesen Fall nur eine Warnmeldung (*bErr* = TRUE, *sErrDescr* = Warnmeldung) ausgegeben - der Baustein unterbricht seine Abarbeitung nicht.

Eingänge

```
VAR_INPUT
  ePipeSys          : E_BA_PipeSys;
  eMedium           : E_BA_Medium;
  bDewPnt           : BOOL;
  fRmT              : REAL;
  fSpHtg            : REAL;
  fSpCol            : REAL;
  fCtrlValHtg       : REAL;
  fCtrlValCol       : REAL;
  nChgOvrDel        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
ePipeSys	E_BA_PipeSys [▶ 272]	Rohrleitersystem (2Pipe, 4Pipe) der Anlage.
eMedium	E_BA_Medium [▶ 272]	Auswahl des Mediums für das gesamte Zweileiternetz (NoMedium, HeatMedium, CoolMedium).
bDewPnt	BOOL	Taupunktwächter: wenn *bDewPnt* = FALSE, dann wird der Kühlregler gesperrt.
fRmT	REAL	Raumtemperatur
fSpHtg	REAL	Errechneter Sollwert der Zulufttemperatur für den Erhitzer.
fSpCol	REAL	Errechneter Sollwert der Zulufttemperatur für den Kühler.
fCtrlValHtg	REAL	Aktueller Ausgabewert vom Heizregler. Wird intern als Umschaltkriterium von Heizen auf Kühlen verwendet: *fCtrlValHtg* muss 0 sein.
fCtrlValCol	REAL	Aktueller Ausgabewert vom Kühlregler. Wird intern als Umschaltkriterium von Kühlen auf Heizen verwendet: *fCtrlValCol* muss 0 sein.
nChgOvrDel	UDINT	Umschaltverzögerung [s] von Heizen auf Kühlen und umgekehrt. Intern begrenzt auf einen Minimalwert von 0.

Ausgänge

```
VAR_OUTPUT
  bEnHtg              : BOOL;
  bEnCol              : BOOL;
  eReqdMedium         : E_BA_Medium;
  nRemTiChgOvrDlyHtg  : UDINT;
  nRemTiChgOvrDlyCol  : UDINT;
  bErr                : BOOL;
  sErrDescr           : T_MAXSTRING;
END_VAR
```

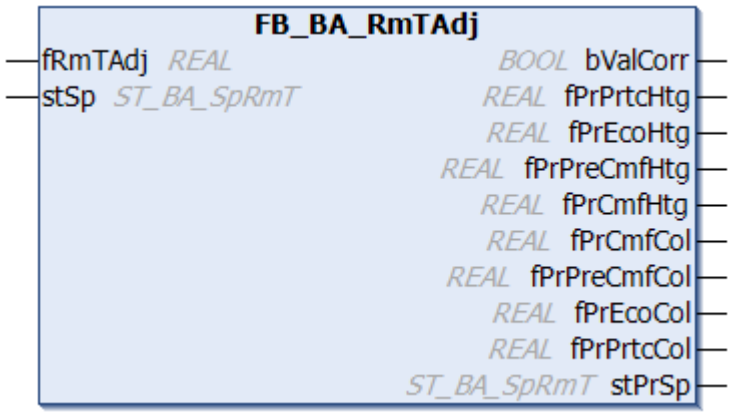
Name	Typ	Beschreibung
bEnHtg	BOOL	Freigabe des Reglers Heizen.
bEnCol	BOOL	Freigabe des Reglers Kühlen.
eReqdMedium	E_BA_Medium [▶ 272]	Angefordertes Medium (siehe Bedarfsermittlung).
nRemTiChgOvrDlyHtg	UDINT	Countdown [s] Umschaltverzögerung von Kühlen auf Heizen.
nRemTiChgOvrDlyCol	UDINT	Countdown [s] Umschaltverzögerung von Heizen auf Kühlen.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Warnung: Der Sollwert Heizen (Setpoint Heating) ist größer als oder gleich dem Sollwert Kühlen (Setpoint Cooling)

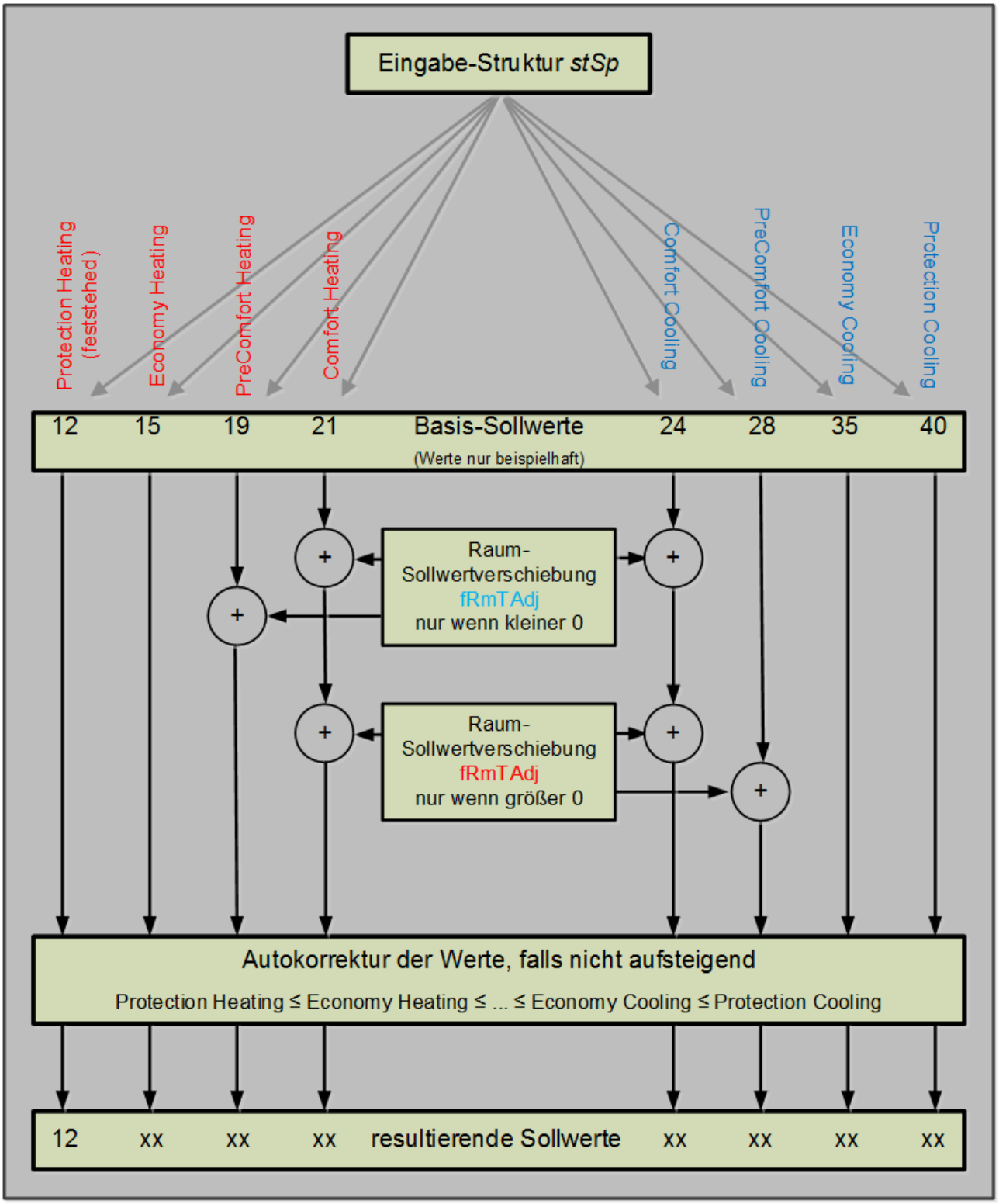
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.1.2 FB_BA_RmTAdj



Der Funktionsbaustein FB_BA_RmTAdj dient der Anpassung des Raumtemperatursollwertes vom Nutzer im Raum. Er verschiebt die Sollwerte am Eingang des Funktionsbausteins in Abhängigkeit eines Offsets *fRmTAdj*, wie in der folgenden Skizze dargestellt. Am Eingang *fRmTAdj* kann für die Sollwertkorrektur z. B. der Wert eines Widerstandspotenziometers oder eines busfähigen Feldgerätes verwendet werden.



Ist der Einstellwert *fRmTAdj* größer als Null, so wird eine Raumtemperatur-Erwärmung gewünscht: Der Wert Comfort Heating wird um den Wert *fRmTAdj* angehoben. Gleichzeitig werden auch die Werte Comfort Cooling und Precomfort Cooling erhöht. Ist der Wert *fRmTAdj* hingegen kleiner als Null, so wird eine Raumtemperatur-Absenkung erwünscht. Analog zum Erwärmungsfall werden nun die Werte Comfort Cooling, Comfort Heating und PreComfort Heating um den Wert *fRmTAdj* herabgesetzt.

Auto-Korrektur

Die Temperaturanpassung ist dazu gedacht, die Werte in kleinem Rahmen zu beeinflussen. Natürlich können beliebige Eingangswerte eingegeben werden. Eine Heizungsanlage wird jedoch nur dann sinnvoll funktionieren, wenn die Sollwerte in folgender Reihenfolge aufsteigende Werte besitzen:

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

Die Auto-Korrektur arbeitet nach folgendem Prinzip: Beginnend mit dem Wert Economy Heating wird geprüft, ob dieser Wert kleiner als der niedrigere Wert Protection Heating ist. Ist das der Fall, wird der Wert Economy Heating dem Wert Protection Heating gleichgesetzt. Es folgt dann die Prüfung, ob der Wert Precomfort Heating kleiner ist als Economy Heating und so weiter bis zur Prüfung, ob Protection Cooling kleiner ist als Economy Cooling. Sind ein oder mehrere Werte korrigiert worden, so wird dies am Ausgang *bValCorr* mit einem TRUE-Signal angezeigt.

Eingänge

```
VAR_INPUT
  fRmTAdj    : REAL;
  stSp       : ST_BA_SpRmT;
END_VAR
```

Name	Typ	Beschreibung
fRmTAdj	REAL	Raumtemperatur-Verschiebungswert.
stSp	ST_BA_SpRmT [▶ 277]	Eingabe-Struktur der Sollwerte.

Ausgänge

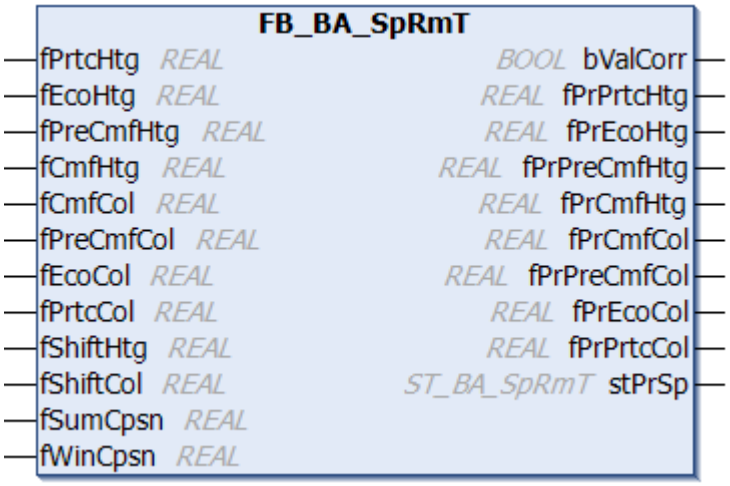
```
VAR_OUTPUT
  bValCorr       : BOOL;
  fPrPrtcHtg     : REAL;
  fPrEcoHtg      : REAL;
  fPrPreCmfHtg   : REAL;
  fPrCmfHtg      : REAL;
  fPrPrtcCol     : REAL;
  fPrEcoCol      : REAL;
  fPrPreCmfCol   : REAL;
  fPrCmfCol      : REAL;
  stPrSp         : ST_BA_SpRmT;
END_VAR
```

Name	Typ	Beschreibung
bValCorr	BOOL	Autokorrektur der Werte wurde durchgeführt, siehe oben.
fPrPrtcHtg	REAL	Resultierender Sollwert „Protection Heating“.
fPrEcoHtg	REAL	Resultierender Sollwert „Economy Heating“.
fPrPreCmfHtg	REAL	Resultierender Sollwert „PreComfort Heating“.
fPrCmfCol	REAL	Resultierender Sollwert „Comfort Cooling“.
fPrPreCmfCol	REAL	Resultierender Sollwert „PreComfort Cooling“.
fPrEcoCol	REAL	Resultierender Sollwert Economy Cooling.
fPrPrtcCol	REAL	Resultierender Sollwert „Protection Cooling“.
stPrSp	ST_BA_SpRmT [▶ 277]	Zusammengefasste Ausgabe der resultierenden Werte in einer Struktur.

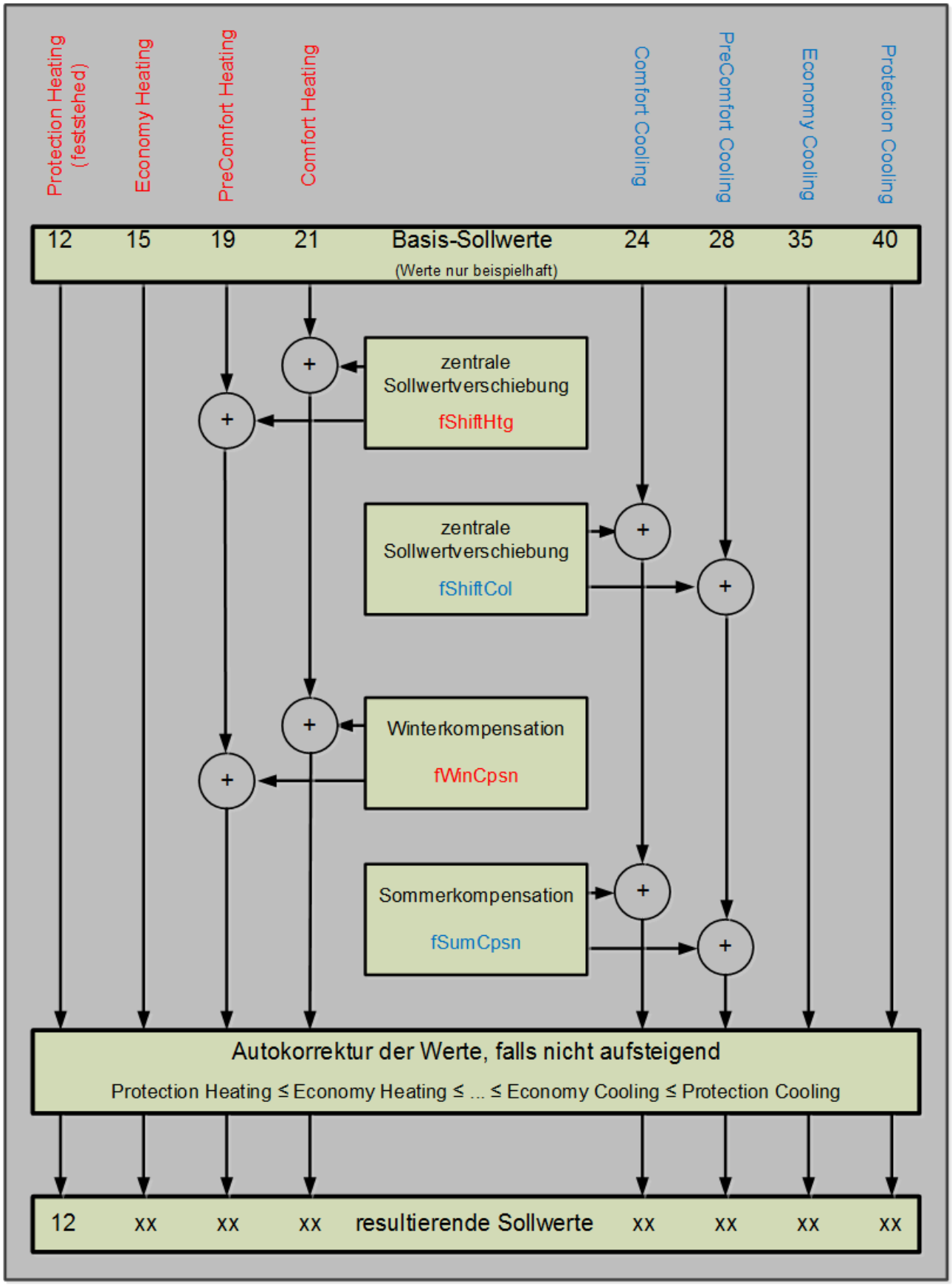
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.1.3 FB_BA_SpRmT



Der Funktionsbaustein FB_BA_SpRmT weist den Energieniveaus Protection, Economy, PreComfort und Comfort jeweils einen Sollwert für den Kühl- und Heizbetrieb zu.
Die folgende Grafik beschreibt das Verhalten des Funktionsbausteins, wobei die eingetragenen Werte als Beispiel zu sehen sind:



Die Comfort- und PreComfort-Werte des Heizbetriebes werden mit dem Parameter *fShiftHtg* als zentrale Sollwertverschiebung beaufschlagt. Unabhängig davon wird noch Winterkompensation *fWinCpsn* hinzuaddiert.
Für den Kühlbetrieb gilt analog: Die Comfort- und PreComfort-Werte werden mit dem Parameter *fShiftCol* und unabhängig davon mit dem Wert der Sommerkompensation *fSumCpsn* beaufschlagt.

Auto-Korrektur

Die Temperaturanpassung ist dazu gedacht, die Werte in kleinem Rahmen zu beeinflussen. Natürlich können beliebige Eingangswerte eingegeben werden. Eine Heizungsanlage wird jedoch nur dann sinnvoll funktionieren, wenn die Sollwerte in folgender Reihenfolge aufsteigende Werte besitzen:

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

Die Auto-Korrektur arbeitet nach folgendem Prinzip: Beginnend mit dem Wert Economy Heating wird geprüft, ob dieser Wert kleiner als der niedrigere Wert Protection Heating ist. Ist das der Fall, wird der Wert Economy Heating dem Wert Protection Heating gleichgesetzt. Es folgt dann die Prüfung, ob der Wert Precomfort Heating kleiner ist als Economy Heating und so weiter bis zur Prüfung, ob Protection Cooling kleiner ist als Economy Cooling. Sind ein oder mehrere Werte korrigiert worden, so wird dies am Ausgang *bValCorr* mit einem TRUE-Signal angezeigt.

Eingänge

```
VAR_INPUT
  fPrtcHtg    : REAL;
  fEcoHtg     : REAL;
  fPreCmfHtg  : REAL;
  fCmfHtg     : REAL;
  fCmfCol     : REAL;
  fPreCmfCol  : REAL;
  fEcoCol     : REAL;
  fPrtcCol    : REAL;
  fShiftHtg   : REAL;
  fShiftCol   : REAL;
  fSumCpsn    : REAL;
  fWrWinCpsn  : REAL;
END_VAR
```

Name	Typ	Beschreibung
fPrtcHtg	REAL	Basis-Sollwert „Protection Heating“.
fEcoHtg	REAL	Basis-Sollwert „Economy Heating“.
fPreCmfHtg	REAL	Basis-Sollwert „PreComfort Heating“.
fCmfHtg	REAL	Basis-Sollwert „Comfort Heating“.
fCmfCol	REAL	Basis-Sollwert „Comfort Cooling“.
fPreCmfCol	REAL	Basis-Sollwert „PreComfort Cooling“.
fEcoCo	REAL	Basis-Sollwert „Economy Cooling“.
fPrtcCol	REAL	Basis-Sollwert „Protection Cooling“.
fShiftHtg	REAL	Sollwertverschiebung „Heizen“.
fShiftCol	REAL	Sollwertverschiebung „Kühlen“.
fSumCpsn	REAL	Wert Sommerkompensation.
fWinCpsn	REAL	Wert Winterkompensation.

Ausgänge

```
VAR_OUTPUT
  bValCorr      : BOOL;
  fPrPrtcHtg    : REAL;
  fPrEcoHtg     : REAL;
  fPrPreCmfHtg  : REAL;
  fPrCmfHtg     : REAL;
  fPrCmfCol     : REAL;
```

```
  fPrPreCmfCol   : REAL;
  fPrEcoCol      : REAL;
  fPrPrtcCol     : REAL;
  stPrSp         : ST_BA_SpRmT;
END_VAR
```

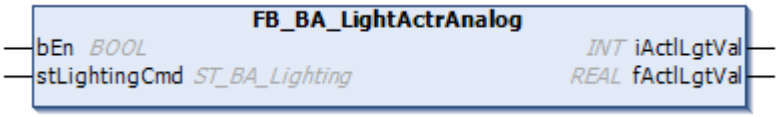
Name	Typ	Beschreibung
bValCorr	BOOL	Autokorrektur der Werte wurde durchgeführt, siehe oben.
fPrPrtcHtg	REAL	Resultierender Sollwert „Protection Heating“.
fPrEcoHtg	REAL	Resultierender Sollwert „Economy Heating“.
fPrPreCmfHtg	REAL	Resultierender Sollwert „PreComfort Heating“.
fPrCmfCol	REAL	Resultierender Sollwert „Comfort Cooling“.
fPrPreCmfCol	REAL	Resultierender Sollwert „PreComfort Cooling“.
fPrEcoCol	REAL	Resultierender Sollwert Economy Cooling.
fPrPrtcCol	REAL	Resultierender Sollwert „Protection Cooling“.
stPrSp	ST_BA_SpRmT [▶ 277]	Zusammengefasste Ausgabe der resultierenden Werte in einer Struktur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.2 Lighting

6.1.2.2.3.1.3.2.1 FB_BA_LightActrAnalog



Der Funktionsbaustein FB_BA_LightActrAnalog dient zur Ansteuerung eines analogen Lichtaktors.
Die Ausgabe erfolgt in 0…100% und entsprechend umgerechnet in 0…32767 (positiver Integer-Bereich).
Der Baustein eignet sich damit für den Einsatz mit den Dimmerklemmen (z. B. KL2751, KL2761).

Funktion

Ist der Baustein nicht freigegeben (*bEn* = FALSE) so sind die Ausgänge *fActlLgtVal* = 0 und *iActlLgtVal* = 0.
Im freigegebenen Zustand wird der Lichthelligkeitswert des Eingangstelegramms (*stLightingCmd.fLgtVal*) direkt an den Ausgang *fActlLgtVal* weiter gereicht. Derselbe Wert wird multipliziert mit 327,65 an den Ausgang *iActlLgtVal* gegeben.

Eingänge

```
VAR_INPUT
  bEn              : BOOL;
  stLightingCmd    : ST_BA_Lighting;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteins und Aktivierung der Funktion.
stLightingCmd	ST_BA_Lighting [▶ 277]	Lichtsteuertelegramm

Ausgänge

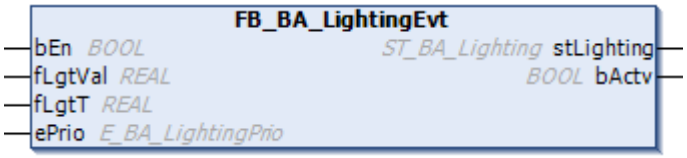
```
VAR_OUTPUT
  iActlLgtVal     : INT;
  fActlLgtVal     : REAL;;
END_VAR
```

Name	Typ	Beschreibung
iActlLgtVal	INT	Aktueller Lichtwert im positiven Integer-Bereich (0… 32767).
fActlLgtVal	REAL	Aktueller Lichtwert in Prozent [%].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.2.2 FB_BA_LightingEvt



Der Funktionsbaustein FB_BA_LightingEvt dient zum Stellen des Lichtwertes und der Farbtemperatur bei einem beliebigen Ereignis. Er kann beispielsweise genutzt werden, um die Beleuchtung beim Nachtwächterrundgang oder für die Gebäudereinigung zu schalten.

Wenn der Funktionsbaustein über den Eingang *bEn* aktiviert ist, so wird der Aktiv-Merker im Lichtsteuertelegramm (*bActv* in stLighting) am Ausgang *stLighting* gesetzt. Die an den Eingangsvariablen eingetragenen Werte *fLgtVal* für den Lichtwert [%] und *fLgtT* für die Lichttemperatur [K] werden in diesem Telegramm weiter gereicht. Ist die Funktion durch Rücksetzen von *bEn* nicht mehr aktiv, so wird der Aktiv-Merker im Lichtsteuertelegramm stLighting zurück und die Werte für Helligkeit und Farbtemperatur auf "0" gesetzt. Mit einem Telegrammauswahlbaustein (z.B. FB_BA_LightingTgmSel8) kann durch das Rücksetzen eine Funktion niedrigerer Priorität die Steuerung übernehmen.

Die Priorität (E_BA_LightingPrio) des Ausgabetelegrammes ist über den Parameter *ePrio* (VAR_INPUT CONSTANT PERSISTENT) definierbar. Sie ist auf *eManualActuator* voreingestellt.

Informationen zu vererbten Elementen

Der Baustein erbt, von der internen Basisklasse *FB_BA_BaseLightingEvt*, die einen Telegrammzähler zur Ermittlung des zuletzt gesendeten Befehls beinhaltet.

Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  fLgtVal      : REAL;
  fLgtT        : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Ein TRUE-Signal an diesem Eingang aktiviert den Baustein und übergibt die eingetragenen Sollwerte im Licht-Stelltelegramm ST_BA_Lighting [▶ 277] zusammen mit dem Aktivmerker. Ein FALSE-Signal setzt den Aktivmerker wieder zurück und den Lichtwert auf Null.
fLgtVal	REAL	Lichtwert [%]
fLgtT	REAL	Lichttemperatur [K]

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio          : E_BA_LightingPrio := E_BA_LightingPrio.eScene1;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_LightingPrio [▶ 273]	Priorität E_BA_LightingPrio des Telegrammes, voreingestellt auf *eScene1*.

Ausgänge

```
VAR_OUTPUT
  stLighting    : ST_BA_Lighting;
  bActv         : BOOL;
END_VAR
```

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Licht-Stelltelegramm
bActv	BOOL	Aktiv

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.2.3 FB_BA_LightingTgmSel8 / FB_BA_LightingTgmSel4



Die Funktion der Bausteine ist stellvertretend am FB_BA_LightingTgmSel8 erläutert

Die Funktionsbausteine dienen zur Prioritätssteuerung für bis zu 4 bzw. bis zu 8 Lichtsteuertelegramme (*stLighting_Prio1* … *stLighting_Prio4,* bzw. *stLighting_Prio1* … *stLighting_Prio8*) des Typs ST_BA_Lighting.

Das aktive Telegramm der höchsten Priorität wird am Ausgang stLighting ausgegeben. "Aktiv" bedeutet, dass innerhalb der Struktur des Positioniertelegramms die Variable *bActv* gesetzt ist. Die Priorität ist dabei innerhalb der Telegrammstruktur als *ePrio* hinterlegt, wobei die Priorität als höher gilt, je geringer der Wert von *ePrio* ist.

Bei Telegrammen gleicher Priorität gilt das zuletzt veränderte (last writer wins), ermittelt durch die Variable *nEvtInc*.

Dieser Baustein ist so zu programmieren, dass immer eines der angelegten Telegramme aktiv ist. Falls ein Telegramm nicht aktiv ist, wird am Ausgang ein Leertelegramm ausgegeben, d.h. *fLgtVal* = 0, *fLgtT* = 0, *bDimUp* = FALSE, *bDimDwn* = FALSE, *bDimMod* = FALSE, *bActv* = FALSE.

Eingänge

```
VAR_INPUT
  stLightingTgm_1    : ST_BA_Lighting;
  stLightingTgm_2    : ST_BA_Lighting;
  stLightingTgm_3    : ST_BA_Lighting;
  stLightingTgm_4    : ST_BA_Lighting;
  stLightingTgm_5    : ST_BA_Lighting;
  stLightingTgm_6    : ST_BA_Lighting;
  stLightingTgm_7    : ST_BA_Lighting;
  stLightingTgm_8    : ST_BA_Lighting;
END_VAR
```

Name	Typ	Beschreibung
stLightingTgm_N	ST_BA_Lighting [▶ 277]	Telegrammeingänge

Ausgänge

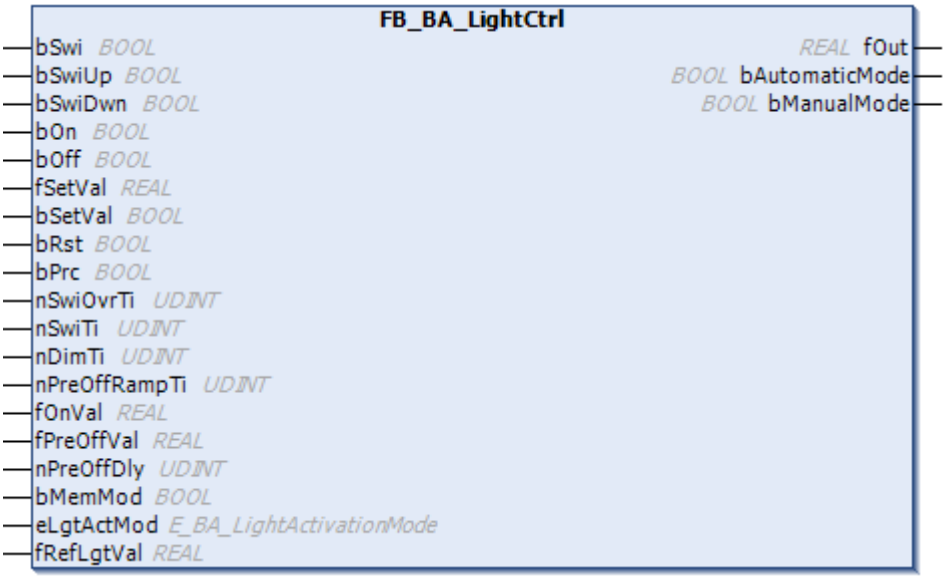
```
VAR_OUTPUT
  stLighting      : ST_BA_Lighting;
  nNumActvTgm     : UINT;
  ePrioActvTgm    : E_BA_LightingPrio;
END_VAR
```

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Resulierendes Telegramm
nNumActvTgm	UINT	Zeigt an, welcher Eingang resultiert, wenn z.B. *stLightingTgm_3* durchgereicht wird, ist *nNumActvTgm* = 3. Ist *nNumActvTgm* = 0, so ist kein Telegramm aktiv.
ePrioActvTgm	E_BA_LightingPrio [▶ 273]	Dieser Ausgang gibt die Priorität des aktiven Telegrammes an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.2.4 FB_BA_LightCtrl



Der Funktionsbaustein ist ein universeller Baustein zur Realisierung verschiedener Lichtfunktionen.

Er dient der Steuerung einer oder mehrerer Leuchten.

Er ermöglicht die manuelle Steuerung der Beleuchtung via Schalter bzw. Raumbediengerät oder die automatische Steuerung einer Lichtgruppe mittels Präsenzerkennung. Durch entsprechende Beschaltung des Bausteins lässt er sich für die folgenden Funktionen einsetzen:

- Automatiklicht präsenzabhängig
- Manuelle Schaltung einer Beleuchtung mit Dimmfunktion
- Kombination aus Automatiklicht via Präsenz und manueller Steuerung.

Der Automatikbetrieb der Beleuchtung via Präsenz lässt sich mit der manuellen Bedienung der Beleuchtung über die Eingänge *bSwi*, *bSwiUp*, *bSwiDwn*, *bOn*, *bOff* kombinieren. Gültig ist dann die letzte Statusänderung von einem der genannten Eingänge.

Ist die Beleuchtung über die Erkennung von Präsenz am Eingang *bPrc* eingeschaltet, dann kann die Beleuchtung über die Eingänge *bSwi*, *bSwiUp* oder *bSwiDwn* ausgeschaltet bzw. verändert werden. Der manuelle Eingriff ist bis zu einer fallenden Flanke an *bPrc* gültig.

Bei einer erneuten steigenden Flanke an dem Eingang *bPrc* wird die Beleuchtung erneut im Automatikbetrieb eingeschaltet.

Die automatische Einschaltung der Beleuchtung via *bPrc* erfolgt allerdings nur, wenn sich die Lichtautomatik in der Betriebsart Vollautomatik befindet. Die automatische Einschaltung der Beleuchtung via *bPrc* erfolgt allerdings nur, wenn sich die Lichtautomatik in der Betriebsart Vollautomatik befindet.

Ist der aktuelle Stellwert der Beleuchtung das Ergebnis einer manuellen Bedienung, befindet sich der Funktionsbaustein in der manuellen Betriebsart. Dieses wird an dem Ausgang *bManualMode* angezeigt. Ist der aktuelle Stellwert der Beleuchtung das Ergebnis einer positiven oder negativen Flanke am Eingang *bPrc*, dann befindet sich die Beleuchtung in der Betriebsart Automatik *bAutomatikMode* = TRUE. Der aktuelle Stellwert für die Lichtgruppe wird am Ausgang *fOut* von 0% bis 100% ausgegeben.

Eingänge

```
VAR_INPUT
  bSwi               : BOOL;
  bSwiUp             : BOOL;
  bSwiDwn            : BOOL;
  bOn                : BOOL;
  bOff               : BOOL;
  fSetVal            : REAL;
  bSetVal            : BOOL;
  bRst               : BOOL;
  bPrc               : BOOL;
  nSwiOvrTi          : UDINT := 250;
  nSwiTi             : UDINT := 2;
  nDimTi             : UDINT := 5;
  nPreOffRampTi      : UDINT := 10;
  fOnVal             : REAL  := 100;
  fPreOffVal         : REAL  := 20;
  nPreOffDly         : UDINT := 20;
  bMemMod            : BOOL;
  eLgtActMod         : E_BA_LightActivationMode;
  fRefLgtVal         : REAL  := -1;
END_VAR
```

Name	Typ	Beschreibung
bSwi	BOOL	Ein kurzer Tastendruck an dem Eingang deaktiviert die Beleuchtung wenn sie gerade eingeschaltet ist. Wenn die Beleuchtung aus ist, wird sie durch einen kurzen Tastendruck eingeschaltet. Ein langer Tastendruck dimmt die Leuchten der Lichtgruppe heller oder dunkler.
bSwiUp	BOOL	Ein kurzer Tastendruck schaltet die Beleuchtung ein. Ein langer Tastendruck dimmt die Beleuchtung heller.
bSwiDwn	BOOL	Ein kurzer Tastendruck schaltet die Beleuchtung aus. Ein langer Tastendruck dimmt die Beleuchtung dunkler.
bOn	BOOL	Unabhängig vom aktuellen Status der Beleuchtung wird das Licht von einer Flanke am Eingang auf 100 % geschaltet.
bOff	BOOL	Unabhängig vom aktuellen Status der Beleuchtung wird das Licht von einer Flanke am Eingang auf 0 % geschaltet.
fSetVal	REAL	Handvorgabewert. Der Wert am Eingang *fSetVal* wird durch eine steigende Flanke an *bSetVal* als Lichtausgabewert übernommen.
bSetVal	BOOL	Übernahme des Handvorgabewertes.
bRst	BOOL	Dieser Eingang setzt den manuellen und automatischen Betrieb des Funktionsbausteins zurück. Die Beleuchtung wird damit ausgeschaltet. Der Ausgang *fOut* folgt dabei einer Abschaltrampe. Siehe hierzu *fPreOffVal*, *nPreOffDly* und *nPreOffRampT*.
bPrc	BOOL	Präsenzmeldeeingang. Ist der Baustein im Vollautomatikbetrieb, so kann über diesen Eingang eine steigende Flanke die Funktion aktivieren und eine fallende Flanke deaktivieren. Das Licht wird dabei ein- bzw. ausgeschaltet. Im Halbautomatikbetrieb deaktiviert nur eine fallende Flanke an diesem Eingang den Baustein.
nSwiOvrTi	UDINT	Zeit [ms] zur Unterscheidung zwischen kurzem- und langem Tastendruck.
nSwiTi	UDINT	Rampe für die Schaltfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100 %.
nDimTi	UDINT	Rampe für die Dimmfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100 %.
nPreOffRampTi	UDINT	Rampe, mit der vor dem Abschalten auf einen Basiswert *fPreOffVal* gefahren wird.
fOnVal	REAL	Einschaltwert, falls nicht über den Eingang *bMemMod* die Möglichkeit des „Memory-Mode" angewählt ist.
fPreOffVal / nPreOffDly	REAL / UDINT	Basiswert und Haltezeit auf diesem Wert vor Ausschalten der Gesamtfunktion. Unterschreitet der aktuelle Lichtwert den Basiswert bereits, so ist diese Funktion nicht aktiv und es wird unmittelbar abgeschaltet.
bMemMod	BOOL	Memory-Mode: Beim Einschalten des Lichts im Handbetrieb nimmt das Licht den Wert an, den die Funktion vor dem letzten Ausschalten innehatte. Ist der "Memory-Mode" nicht aktiv, so ist der Einschaltwert über *fOnValMan* definiert.

Name	Typ	Beschreibung
eLgtActMod	E_BA_LightActivationMode [▶ 272]	Aktivierungsmodus der Lichtsteuerfunktion. Vollautomatik: Die Lichtsteuerfunktion wird über erkannte Präsenz aktiviert und über wegfallende Präsenz deaktiviert. Tastendrücke können die Funktion auch aktivieren. Halbautomatik: Aktivierung der Lichtsteuerfunktion nur über Tastendruck, über wegfallende Präsenz ist sie automatisch deaktiviert.
fRefLgtVal	REAL	Lichtwert eines einzelnen Lichtaktors, der die Lichtzone repräsentiert. Dieser Eingang dient der Beurteilung, ob im Handbetrieb mit dem nächsten Tastendruck ein- oder ausgeschaltet wird oder ab welchem Wert gedimmt werden soll. Dieser Eingang ist mit "-1" vorinitialisiert. Bei einer Nichtbelegung wird dieses erkannt und davon ausgegangen, dass diese Funktion die alleinige zur Ansteuerung der Leuchten ist. Damit wird der Ausgabewert *fOut* als Lichtwert zur Beurteilung der nächsten Schaltaktionen herangezogen.

Ausgänge

```
VAR_OUTPUT
  fOut              : REAL;
  bAutomaticMode    : BOOL;
  bManualMode       : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fOut	REAL	Lichtausgabewert in 0…100 %.
bAutomaticMode	BOOL	Der Baustein wurde über eine steigende Flanke am Präsenzeingang aktiviert, was nur im Aktivierungsmodus "Vollautomatik" möglich ist.
bManualMode	BOOL	Der Baustein wurde durch einen Taster (*bOn*, *bSwi*, *bSwiUp*, *bSwiDwn*) oder durch das Setzen auf einen Wert (*fSetValMan*/*bSetValMan*) aktiviert bzw. übersteuert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.2.5 FB_BA_LightCtrlValue



Dieser Baustein dient dazu, einen Einschaltwert (0..100%) für eine Konstantlichtregelung zu bestimmen, so dass diese mit einer Helligkeit nahe des Sollwertes aufstartet und das Licht nicht lange nachsteuern muss.

Funktion

Die Funktionen dieses Bausteines sind nur verfügbar, wenn er mit einem TRUE-Signal an *bEn* freigegeben ist. Anderenfalls nimmt der Ausgang *fOnValue* den Standardwert am Eingang *fOnValDefault* an. Ist *fOnValDefault* nicht beschaltet, so wird er per Initialisierung auf 50 gesetzt.

Einlernen

Im freigegebenen Zustand (*bEn* = TRUE) lässt sich der Baustein einlernen. Mit einer steigenden Flanke an *bTeach* wird eine Schrittkette gestartet.

Das Licht wird über das Lichtsteuertelegramm *stTeachLighting* zunächst ausgeschaltet. Danach wird für die Zeit *nBrtnsAdjTi* in Sekunden gewartet, so dass der Helligkeitssensor die Lichtverhältnisse richtig erfassen kann. Der Helligkeitswert, der über den Eingang *fBrtns* eingelesen wird, wird gespeichert. Das Licht wird nun auf 100% gestellt und es wird wieder *nBrtnsAdjTi* in Sekunden gewartet. Die Differenz des nun erfassten Helligkeitswertes zum gespeicherten Wert ist der mögliche Zuwachs durch das Kunstlicht und wird als persistente Ausgangsvariable *fBrtnsGain* gespeichert angezeigt. Der Lichtwert wird zum Abschluss der Routine noch einmal auf 0 gesetzt.

Das Lichtsteuertelegramm *stTeachLighting* ist nur während dieser Einlernphase aktiv und hat mit *ePrio* := *E_BA_LightingPrio.eMaintenance* eine hohe Priorität über den Hand oder Automatikbefehlen, die keinen Alarm darstellen. Damit erfüllt dieser Baustein die Bedingung, parallel zur Konstantlichtregelung zu stehen - die jeweiligen Telegramme können schlussendlich über einen Prioritätenschalter geführt werden.

Darüber hinaus ist während der Einlernphase der Ausgang *bTeaching* gesetzt. Dieser kann dazu genutzt werden, Helligkeitssensoren, die kein kontinuierliches Signal ausgeben, schneller abzufragen und damit den Einlernvorgang zu verkürzen.

Arbeitsweise

Ist der Baustein freigegeben (*bEn* = TRUE) und mindestens einmal eingelernt, so dass der Ausgangswert *fBrtnsGain* einen Wert größer Null angenommen hat, so wird kontinuierlich ein Eingangswert *fOnValue* errechnet:

Die Differenz von Soll- und Istwert, *fSP* zu *fBrtns*, wird per Dreisatz dem Zuwachs durch das Kunstlicht *fBrtnsGain* gegenübergestellt. Damit ergibt sich der notwendige prozentuale Helligkeitsaufschlag. Dieser wird dem aktuellen Lichtwert, der am Eingang *fRefLgtVal* anliegt, aufaddiert.

Ein nicht eingelernter Baustein wird dadurch erkannt, dass der Ausgang *fBrtnsGain* noch auf 0 steht. In diesem Fall wird auch bei gegebener Bausteinfreigabe (*bEn* = TRUE) der Standardwert *fOnValDefault* ausgegeben.

fRefLgtVal

Lichtwert eines einzelnen Lichtaktors, der die Lichtzone repräsentiert. Mit dem Helligkeits-Istwert und -Sollwert sowie dem ermittelten Helligkeitszuwachs *fBrtnsGain* lässt sich lediglich der prozentuale Aufschlag ermitteln, der notwendig ist, um die Konstantlichtregelung nahe am Sollwert aufzustarten. Kennt man den aktuellen Lichtwert, so ergibt sich aus der Summe der absolute Einschaltwert.

In den meisten Anwendungen wird der Lichtwert vor Einschalten 0% sein. Durch die Lichtsteuerung über Prioritäten ist es jedoch möglich, dass eine Lichtgruppe vor Einschalten der Konstantlichtregelung von einer niedrigeren Priorität angesteuert wird und damit schon auf einem gewissen Niveau steht.

Daher ist die Kenntnis des aktuellen Lichtstellgröße wichtig.

Eingänge

```
VAR_INPUT
  bEn                : BOOL;
  bTeach             : BOOL;
  fBrtns             : REAL;
  fSP                : REAL;
  nBrtnsAdjTi        : UDINT := 5;
  fOnValDefault      : REAL = 50;
  fRefLgtVal         : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Ein TRUE-Signal an diesem Eingang aktiviert den Baustein. Ist der Baustein noch nicht eingelernt, d.h. der ermittelte Helligkeitszuwachs am Ausgang *fBrtnsGain* ist noch 0, so wird am Ausgang *fOnValue* dauerhaft der Standardwert *fOnValDefault* angenommen. Dieser Wert wird auch gesetzt, wenn der Baustein nicht freigegeben ist (*bEn* = FALSE).
bTeach	BOOL	Eine steigende Flanke an diesem Eingang startet den Einlernvorgang.
fBrtns	REAL	Eingang für den Helligkeitsmesswert.
fSP	REAL	Eingang für den Helligkeitssollwert.
nBrtnsAdjTi	UDINT	Wartezeit in Sekunden, die beim Einlernvorgang nach einem Lichtwechsel gewartet wird, um einen korrekten Wert vom Helligkeitssensor zu empfangen.
fOnValDefault	REAL	Standardausgabewert [0..100%] der am Ausgang *fOnVal* durchgereicht wird, sollte der Baustein entweder nicht freigegeben (*bEn* = FALSE) oder nicht eingelernt sein (*fBrtnsGain* = 0).
fRefLgtVal	REAL	Lichtwert [%] eines einzelnen Lichtaktors, der die Lichtzone repräsentiert.

Ausgänge

```
VAR_OUTPUT
  fOnValue          : REAL;
  bTeaching         : BOOL;
  stTeachLighting   : ST_BA_Lighting;
END_VAR
```

Name	Typ	Beschreibung
fOnValue	REAL	Ermittelter Einschaltwert. Ist der Baustein nicht freigegeben oder nicht eingelernt, so wird hier der Standardwert *fOnValDefault* ausgegeben.
bTeaching	BOOL	Ein TRUE-Signal an diesem Ausgang zeigt an, dass der Einlernvorgang aktiv ist.
stTeachLighting	ST_BA_Lighting [▶ 277]	Lichtsteuertelegramm mit der voreingestellten Priorität *E_BA_LightingPrio.eMaintenance*.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio             : BYTE := E_BA_LightingPrio.eMaintenance;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_LightingPrio [▶ 273]	Priorität des Telegrammes, voreingestellt auf *E_BA_LightingPrio.eMaintenance*.

Ausgänge PERSISTENT

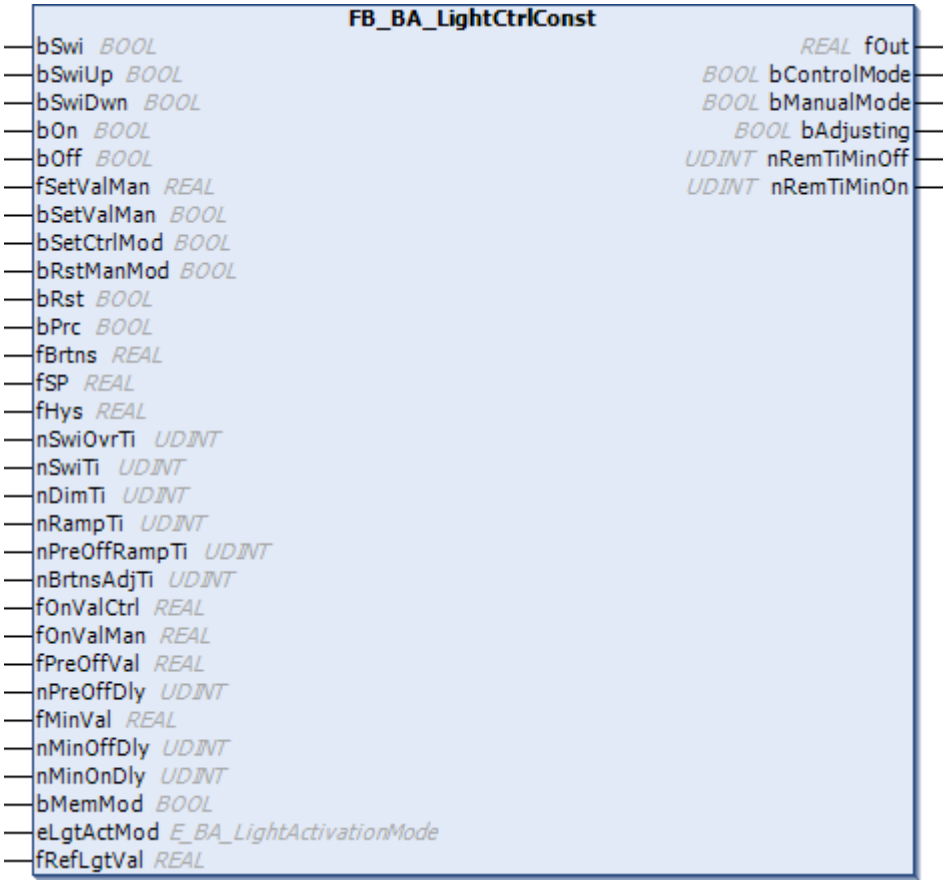
```
VAR_OUTPUT PERSISTENT
  fBrtnsGain        : REAL;
END_VAR
```

Name	Typ	Beschreibung
fBrtnsGain	REAL	Ermittelter Helligkeitszuwachs. Dieser Wert ist bezogen auf den Unterschied von einer Helligkeit (*fBrtns*) bei einem Lichtwert von 0% zu einer Helligkeit bei einem Lichtwert von 100%.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.55	Tc3_BA2 ab v5.3.19.0

6.1.2.2.3.1.3.2.6 FB_BA_LightCtrlConst



Dieser Baustein stellt eine Konstantlichtregelung mit manueller Übersteuerung dar.

Er kann sich, wie in VDI 3813 Blatt 2 beschrieben, in den folgenden Betriebsstatus befinden:

- Deaktiviert: Ausgänge *bControlMode* = FALSE und *bManualMode* = FALSE
- Regelbetrieb: Ausgänge *bControlMode* = TRUE und *bManualMode* = FALSE
- Handübersteuerung: Ausgänge *bControlMode* = FALSE und *bManualMode* = TRUE

In den aktiven Zuständen (Regelbetrieb, Handübersteuerung) kann der Lichtausgabewert *fOut* beliebige Werte von 0…100% annehmen, im nicht aktiven Zustand ist er fest auf 0%.

Die Unterscheidung "aktiv" und "nicht aktiv" ist dann von Bedeutung, wenn mehrere Lichtsteuerfunktionen im Wechsel auf Aktoren zugreifen.

Damit eignet der Baustein sich für das Zusammenspiel mit anderen Lichtfunktionen in einer Prioritätsauswahl, in der der Ausgang *fOut* in Abhängigkeit der Betriebsstatusausgänge als gültiger Steuerwert angesehen wird.

Aufgrund des möglichen Zusammenspiels mit anderen Lichtsteuerfunktionen arbeitet die Lichtsteuerfunktion mit einem Eingang für den Helligkeitsstatus einer Referenzleuchte. Steuern mehrere Lichtfunktionen unabhängig voneinander per Prioritätssteuerung dieselbe Leuchte bzw. dieselbe Lichtgruppe an, so ist dem Baustein nicht bekannt, ob er der aktive ist. Mit Hilfe des Referenzeinganges kann jedoch immer vor jeder Schaltaktion auf den aktuellen Wert synchronisiert werden und vor einer Umschalt- (toggle-) Aktion beurteilt werden, ob das Licht im nächsten Schritt ein- oder ausgeschaltet werden soll.

Ist sichergestellt, dass die Funktion die alleinige ist, die das Licht ansteuert, so ist der Eingang nicht zu belegen. Er ist mit dem Wert -1 vorinitialisiert, was dem Baustein anzeigt, dass der Eingang nicht verknüpft ist.

Das bedeutet aber auch, dass es nicht möglich ist, online eine Eingangsverknüpfung zu löschen, ohne die Funktionsweise des Bausteines zu verfälschen!

Funktion

Vom nicht aktivierten Zustand aus gesehen arbeitet dieser Baustein in zwei Stufen: Kurze Tastendrücke an *bSwi*, *bSwiUp* oder *bSwiDown* sowie steigende Flanken an *bOn*, *bSetCtrlMod* oder *bPrc* schalten die Funktion zunächst in den Regelbetrieb. Der Präsenzmeldeeingang *bPrc* ist dabei jedoch nur im Vollautomatikmodus aktiv.

Der Betriebsmodus Vollautomatik oder Halbautomatik lässt sich am Eingang (eMode [▶ 272]) parametrieren.

Einschaltverhalten

Die Lichtregelung wird zunächst auf den Wert *fOnValCtrl* geschaltet.

Das Schalten des Lichtes erfolgt dabei mit einer Rampe, die unter *nSwiTi* in Sekunden bezogen auf 0 bis 100% angegeben ist.

Ist der gewählte Lichtwert erreicht, so verharrt die Regelung für die Zeit *nBrtnsAdjTi* [s] auf diesem Wert, damit der Helligkeitssensor den korrekten Wert erfasst.

Bereits während dieser Wartezeit kann die Funktion in **Handübersteuerung** geschaltet werden.

Regelverhalten

Die Regelung versucht nun diesen Sollwert zu halten: erhöht sich der Tageslichteinfall, wird der Lichtstellwert am Ausgang *fOut* reduziert. Wird die Gesamthelligkeit schwächer, so wird der Lichtstellwert erhöht. Die Änderung folgt dabei ebenfalls einer Rampe, welche unter *nRampTi* in Sekunden bezogen auf 0 bis 100% angegeben ist. Die Konstantlichtregelung ist damit ein einfacher I-Regler.

Um zu verhindern, dass der Rampenbaustein ständig neue Lichtwerte vorgibt, kann am Eingang *fHys* ein Hysteresebereich angegeben werden.

Die Konstantlichtregelung steuert den Ausgang soweit hoch oder herunter, bis die gemessene Helligkeit den Zielwert erreicht bzw. leicht über- oder unterschreitet. Das Licht gilt dann als ausgeregelt. Erst wenn die gemessene Helligkeit den Sollwert um *fHys*/2 über- oder unterschreitet, wird erneut nachgeregelt. Bereich und Einheit der Hysterese richten sich nach dem Helligkeitssensor.

Minimalwert der Regelung

Änderungen im unteren Helligkeitsbereich werden häufig als störend empfunden.

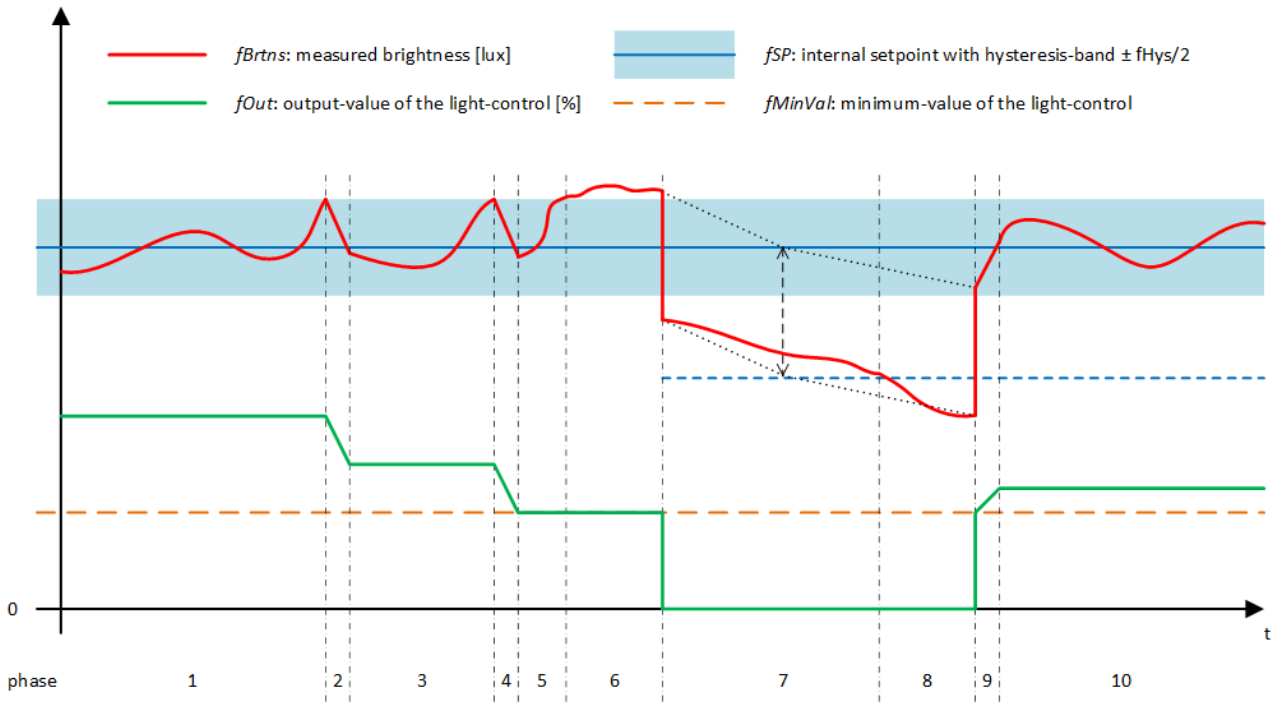
Daher kann es sinnvoll sein, die Lichtregelung auf einen Minimal-Stellwert zu begrenzen und bei sehr hoher Außenhelligkeit das Licht dann ganz abzuschalten.

Hat die Regelung ihren Minimalstellwert *fMinVal* erreicht und soll noch weiter herunter regeln, so verharrt sie auf dem Minimalwert für die Zeit *nOffDly* [s] und schaltet dann auf den Ausgabewert 0.

Vor und nach dem Schalten wird jeweils der gemessene Helligkeitswert gespeichert. Die Differenz, welche sich daraus ergibt, ist der Helligkeitszuwachs durch Einschalten auf den Minimalwert.

Als neuer Schwellwert für das Wiedereinschalten gilt nun der Regelungssollwert minus dem Helligkeitszuwachs. Erreicht oder unterschreitet die gemessene Helligkeit diesen Schwellwert für die Zeit *nOnDly* [s], so wird das Licht wieder eingeschaltet und danach auf den Sollwert geregelt.

Die Wahl dieses Schwellwertes soll das folgende Diagramm verdeutlichen:



1. Die gemessene Helligkeit *fBrtns* bewegt sich zunächst innerhalb der Toleranzgrenzen um den Sollwert *fSP*. Zum Ende hin überschreitet die Helligkeit jedoch den Toleranzbereich.
2. Die Lichtregelung dimmt das Licht (Stellwert *fOut*) so weit herunter, bis der Lichtsollwert erreicht bzw. in dem SPS-Zyklus gerade unterschritten wird. Der gemessene Lichtwert *fBrtns* kann demnach etwas unter dem Sollwert liegen, was jedoch in Kauf genommen wird, um ständiges Nachregeln zu vermeiden.
3. Die gemessene Helligkeit *fBrtns* bewegt sich wieder innerhalb der Toleranzgrenzen, wie in Phase 1.
4. Es wird erneut geregelt, diesmal auf den minimalen Ausgabestellwert, wodurch die gesamte gemessene Helligkeit jedoch wieder in den Toleranzbereich rückt.
5. Die gemessene Helligkeit steigt durch Außenhelligkeit erneut an und überschreitet den Toleranzbereich.
6. Die gemessene Helligkeit *fBrtns* bleibt weiterhin oberhalb des Toleranzbereiches, die interne Zeit *nOffDly* [s] läuft ab.
7. Das Licht wird abgeschaltet (*fOut* = 0). Die Differenz der gemessenen Helligkeit vor und nach dem Abschalten wird als Helligkeitszugewinn durch Einschalten des Lichtes auf den Minimalstellwert intern gemerkt. Der neue Schwellwert, der nun unterschritten werden muss, ist der Regelungssollwert minus dem Helligkeitszugewinn. Der Wahl dieses Schwellwertes liegen folgende Überlegungen zugrunde:

- Ein Wiedereinschalten des Lichtes darf nicht zur Folge haben, dass die gemessene Helligkeit danach wieder oberhalb des Toleranzbereiches liegt, da dies u.U. ein ständiges Ein- und Ausschalten zur Folge hätte.
- Dieser Schwellwert ist realistisch, d.h. > 0: zu Beginn von Phase 5 war die Außenhelligkeit schon derart hoch, dass auf den Minimalwert der Regelung gestellt wurde. Vor Abschalten des Lichts in Phase 7 ist noch einmal mehr Außenhelligkeit hinzugekommen. Im Diagramm ist leicht erkennbar, dass der neue Schwellwert dann erreicht ist, wenn der Zuwachs an Außenhelligkeit von Beginn der Phase 5 bis Ende Phase 6 wieder vergangen ist – ein realistisches Szenario. Sicherheitshalber wird jedoch intern nicht nur auf Unterschreiten, sondern auch noch auf Erreichen des Schwellwertes (kleiner oder gleich) geprüft.

1. Der Schwellwert wird für die Zeit *nOnDly* [s] unterschritten.
2. Das Licht wird wieder zunächst auf den Minimalwert geschaltet und danach so lange hochgedimmt, bis die gemessene Helligkeit dem Sollwert *fSP* entspricht.
3. Die gemessene Helligkeit *fBrtns* bewegt sich wieder innerhalb der Toleranzgrenzen um den Sollwert *fSP*.

Handübersteuerung

Aus dem Regelbetrieb heraus lassen folgende Aktionen die Funktion in den Handbetrieb (Ausgänge *bControlMode* = FALSE und *bManualMode* = TRUE) wechseln:

- Kurze Tastendrücke an den Eingängen *bSwi*, *bSwiUp* und *bSwiDwn* schalten das Licht wechselseitig ein und aus.
- Mit *bOn* und *bOff* wird gezielt ein- bzw. ausgeschaltet.
- Ein langer Tastendruck am Eingang *bSwi* dimmt das Licht wechselseitig auf und ab. Das Dimmen erfolgt über eine Rampe, die unter *nDimTi* in Sekunden bezogen auf 0 bis 100% angegeben ist.
- Ein langer Tastendruck am Eingang *bSwiUp* dimmt das Licht gezielt auf. Das Dimmen erfolgt über eine Rampe, die unter *nDimTi* in Sekunden bezogen auf 0 bis 100% angegeben ist.
- Ein langer Tastendruck am Eingang *bSwiDwn* dimmt das Licht gezielt ab. Das Dimmen erfolgt über eine Rampe, die unter *nDimTi* in Sekunden bezogen auf 0 bis 100% angegeben ist.
- Eine Änderung am Eingang *fSetValMan*. Dieser Wert wird dann als neuer Lichtausgabewert übernommen.

"Aus" im Handbetrieb bedeutet dabei, dass die Gesamtfunktion weiterhin aktiv ist!

Mit einer positiven Flanke an *bRstManMod* oder *bSetCtrlMod* wird aus der Handübersteuerung wieder in den Regelbetrieb geschaltet.

Memory-Mode

Ist der Memory-Mode am Eingang *bMemMod* aktiviert, so wird sich intern der Wert gemerkt, den der Baustein vor dem letzten manuellen Abschalten innehatte. Beim nächsten Einschalten im Handmodus wird dieser Wert dann übernommen.

Ist Memory-Mode deaktiviert, erfolgt das Einschalten im Handmodus immer mit dem Wert *fOnValMan*.

Deaktivieren der Funktion

Aus dem Regelbetrieb oder Handbetrieb heraus lassen zwei Ereignisse die Funktion in den deaktivierten Zustand (Ausgänge *bControlMode* = FALSE und *bManualMode* = FALSE) wechseln:

- eine fallende Flanke am Eingang *bPrc*.
- ein TRUE-Signal am Eingang *bRst*. Dieser Eingang ist für die zentrale Deaktivierung des Bausteins gedacht oder für den Fall, dass kein Präsenzmelder vorhanden ist.

Bei der Deaktivierung wird über eine Rampe *nPreOffRampTi* (in Sekunden bezogen auf 100% bis 0%) auf einen Basiswert *fPreOffVal* herabgedimmt. Dieser Wert steht für die Zeit *nPreOffDly* [s] am Ausgang an, bevor ausgeschaltet wird. Der Funktionsbaustein ist dann deaktiviert.

Während der Ausschaltphase kann der Baustein wieder aktiviert werden, wobei er dann wieder in den Regelbetrieb übergeht.

Die Aktivierung erfolgt dabei über Kurze Tastendrücke an *bSwi*, *bSwiUp* oder *bSwiDown* sowie steigende Flanken an *bOn* oder *bPrc*. Der Präsenzmeldeeingang *bPrc* ist dabei jedoch nur im Vollautomatikmodus aktiv.



Ist der Lichtausgabewert vor der Deaktivierung bereits kleiner als der erwähnte Basiswert, so wird die Abschaltroutine übersprungen und unmittelbar ausgeschaltet.

Eingänge

```
VAR_INPUT
  bSwi              : BOOL;
  bSwiUp            : BOOL;
  bSwiDwn           : BOOL;
  bOn               : BOOL;
  bOff              : BOOL;
  fSetValMan        : REAL;
  bSetValMan        : BOOL;
  bSetCtrlMod       : BOOL;
```

```
  bRstManMod        : BOOL;
  bRst              : BOOL;
  bPrc              : BOOL;
  fBrtns            : REAL;
  fSP               : REAL;
  fHys              : REAL := 50;
  nSwiOvrTi         : UDINT := 250;
  nSwiTi            : UDINT := 2;
  nDimTi            : UDINT := 5;
  nRampTi           : UDINT := 60;
  nPreOffRampTi     : UDINT := 10;
  nBrtnsAdjTi       : UDINT := 5;
  fOnValCtrl        : REAL := 50;
  fOnValMan         : REAL  := 100;
  fPreOffVal        : REAL  := 20;
  nPreOffDly        : UDINT := 20;
  fMinVal           : REAL;
  nMinOffDly        : UDINT := 300;
  nMinOnDly         : UDINT := 300;
  bMemMod           : BOOL;
  eLgtActMod        : E_BA_LightActivationMode;
  fRefLgtVal        : REAL  := -1;
END_VAR
```

Name	Typ	Beschreibung
bSwi	BOOL	Ist der Konstantlichtregelbaustein noch nicht aktiv, so schaltet ein kurzer Tastendruck zunächst die Regelung ein, danach lassen weitere Signale den Baustein in die Handfunktion übergehen und es gilt: kurzer Tastendruck: an/aus, lange Tastendrücke: wechselweise hoch- oder herunterdimmen.
bSwiUp	BOOL	Ist der Konstantlichtregelbaustein noch nicht aktiv, so schaltet ein kurzer Tastendruck zunächst die Regelung ein, danach lassen weitere Signale den Baustein in die Handfunktion übergehen und es gilt: kurzer Tastendruck: an/aus, lange Tastendrücke: gezielt hochdimmen.
bSwiDwn	BOOL	Ist der Konstantlichtregelbaustein noch nicht aktiv, so schaltet ein kurzer Tastendruck zunächst die Regelung ein, danach lassen weitere Signale den Baustein in die Handfunktion übergehen und es gilt: kurzer Tastendruck: an/aus, lange Tastendrücke: gezielt herabdimmen.
bOn	BOOL	Ist der Konstantlichtregelbaustein noch nicht aktiv, so schaltet ein TRUE-Signal zunächst die Regelung ein, danach bezieht sich dieser Eingang nur auf das Einschalten im Hand-Modus.
bOff	BOOL	Schaltet das Licht aus, der Konstantlichtregelbaustein befindet sich weiterhin im Handbetrieb. War der Konstantlichtregelbaustein vorher im Regelbetrieb, so befindet er sich nun im Handbetrieb.
fSetValMan	REAL	Handvorgabewert. Die Werte am Eingang *fSetValMan* werden bei durch eine steigende Flanke an *bSetValMan* als Lichtausgabewert übernommen. Der Baustein geht durch das Setzen direkt in den Handbetrieb über, egal, ob er vorher im Regelbetrieb oder im deaktivierten Zustand war.
bSetValMan	BOOL	Übernahme des Handvorgabewertes, siehe *fSetValMan*.
bSetCtrlMod	BOOL	Eine steigende Flanke versetzt den Baustein direkt in den Regelbetrieb, unabhängig davon, ob er vorher im deaktivierten Zustand oder im Handbetrieb war.
bRstManMod	BOOL	Dieser Eingang lässt den Konstantlichtregelbaustein, sollte er sich im Handbetrieb befinden, zurück in den Automatikmodus wechseln.
bRst	BOOL	Dieser Eingang schaltet den Konstantlichtregelbaustein ab. Die Abschaltung erfolgt über eine Rampe und Verweilzeit auf einem Basislichtwert, siehe unten: *fPreOffVal*, *nPreOffDly* und *nPreOffRampT*.
bPrc	BOOL	Präsenzmeldeeingang. Ist der Konstantlichtregelbaustein im Vollautomatikbetrieb konfiguriert, so kann über diesen Eingang eine steigende Flanke die Funktion aktivieren, falls sie vorher nicht aktiv war und eine fallende Flanke deaktivieren. Im Halbautomatikbetrieb deaktiviert nur eine fallende Flanke an diesem Eingang den Konstantlichtregelbaustein. "Aktivieren" bedeutet hierbei, dass die Funktion um Handbetrieb eingeschaltet wird (Ausgänge *bControlMode* = TRUE und *bManualMode* = FALSE). Deaktivieren bedeutet: die Funktion ist weder im Hand-, noch im Regelbetrieb aktiv (Ausgänge *bControlMode* = FALSE und *bManualMode* = FALSE).
fBrtns	REAL	Aktuelle Helligkeit für die Konstantlichtautomatik: Bereich und Einheit richten sich nach dem Helligkeitssensor.

Name	Typ	Beschreibung
fSP	REAL	Helligkeits-Sollwert für die Konstantlichtautomatik: Es wird ein Abgleich mit *fBrtnsSen* angestrebt. Bereich und Einheit richten sich daher nach dem Helligkeitssensor.
fHys	REAL	Konstantlicht-Automatik: Hystereseband. Die Konstantlichtregelung steuert den Ausgang soweit hoch bzw. herunter, bis die gemessene Helligkeit den Zielwert erreicht bzw. leicht über- oder unterschreitet. Das Licht gilt dann als ausgeregelt. Erst wenn die gemessene Helligkeit den Sollwert um *fHys*/2 über- oder unterschreitet, wird erneut nachgeregelt. Bereich und Einheit richten sich nach dem Helligkeitssensor, siehe oben: *fBrtnsSP* und *fBrtnsSen*.
nSwiOvrTi	UDINT	Zeit [ms] zur Unterscheidung zwischen kurzem- und langem Tastendruck.
nSwiTi	UDINT	Rampe für die Schaltfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100 %.
nDimTi	UDINT	Rampe für die Dimmfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100 %.
nRampTi	UDINT	Regelrampe der Konstantlichtautomatik in Sekunden, bezogen auf ein Dimmen von 0 bis 100%.
nPreOffRampTi	UDINT	Rampe, mit der vor dem Abschalten auf einen Basiswert *fPreOffVal* gefahren wird.
nBrtnsAdjTi	UDINT	Wartezeit [s] nach Einschalten des Lichts, damit der Helligkeitssensor den korrekten Wert erfasst.
fOnValCtrl	REAL	Einschaltwert für den Regelbetrieb, sollte dieser bei vorher ausgeschalteter Funktion aktiviert werden.
fOnValMan	REAL	Einschaltwert der Handfunktion, sollte nicht über den Parameter *eOperationalMode* "Memory-Mode" angewählt sein.
fPreOffVal / nPreOffDly	REAL / UDINT	Basiswert und Haltezeit auf diesem Wert vor Ausschalten der Gesamtfunktion. Unterschreitet der aktuelle Lichtwert den Basiswert bereits, so ist diese Funktion nicht aktiv und es wird unmittelbar abgeschaltet.
fMinVal	REAL	Konstantlichtautomatik: Mindestausgabewert. Wird dieser Wert intern unterschritten (d.h. es ist hell genug, dass kein künstliches Licht benötigt wird), so schaltet die Konstantlichtregelung das Licht nach Ablauf von *nOffDly* [s] ab. Ist das Licht ausgeschaltet und die Regelung erkennt wieder Bedarf an Licht über dem Minimumwert, so schaltet die Regelung nach *nOnDly* [s] das Licht wieder auf zunächst *fMinVal* ein.
nMinOffDly	UDINT	Konstantlichtautomatik: Abschaltwartezeit in Sekunden, siehe *fMinVal*.
nMinOnDly	UDINT	Konstantlichtautomatik: Einschaltwartezeit in Sekunden, siehe *fMinVal*.
bMemMod	BOOL	Memory-Mode: Beim Einschalten des Lichts im Handbetrieb nimmt das Licht den Wert an, den die Funktion vor dem letzten Ausschalten innehatte. Ist der "Memory-Mode" nicht aktiv, so ist der Einschaltwert über *fOnValMan* definiert.

Name	Typ	Beschreibung
eLgtActMod	E_BA_LightActivationMode [▶ 272]	Aktivierungsmodus der Lichtsteuerfunktion. Vollautomatik: Die Lichtsteuerfunktion wird über erkannte Präsenz aktiviert und über wegfallende Präsenz deaktiviert. Tastendrücke können die Funktion auch aktivieren. Halbautomatik: Aktivierung der Lichtsteuerfunktion nur über Tastendruck, über wegfallende Präsenz ist sie automatisch deaktiviert.
fRefLgtVal	REAL	Lichtwert eines einzelnen Lichtaktors, der die Lichtzone repräsentiert. Dieser Eingang dient der Beurteilung, ob im Handbetrieb mit dem nächsten Tastendruck ein- oder ausgeschaltet wird oder ab welchem Wert gedimmt werden soll. Dieser Eingang ist mit "-1" vorinitialisiert. Bei einer Nichtbelegung wird dieses erkannt und davon ausgegangen, dass diese Funktion die alleinige zur Ansteuerung der Leuchten ist. Damit wird der Ausgabewert *fOut* als Lichtwert zur Beurteilung der nächsten Schaltaktionen herangezogen.

Ausgänge

```
VAR_OUTPUT
  fOut              : REAL;
  bControlMode      : BOOL;
  bManualMode       : BOOL;
  bAdjusting        : BOOL;
  nRemTiminOff      : UDINT;
  nRemTiminOn       : UDINT;
END_VAR
```

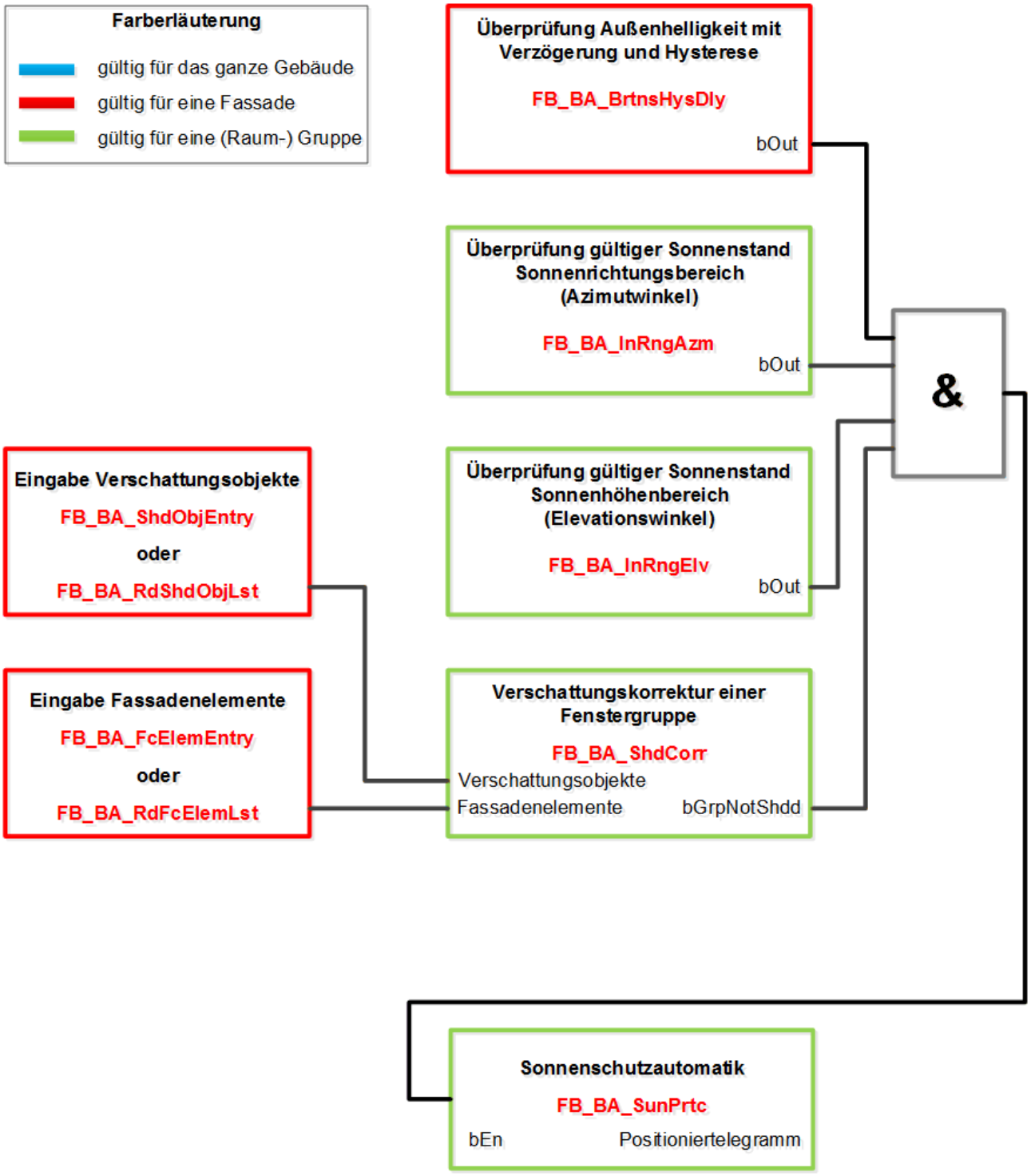
Name	Typ	Beschreibung
fOut	REAL	Lichtausgabewert in 0…100 %.
bControlMode	BOOL	Zeigt an, ob der Baustein im Regelbetrieb arbeitet.
bManualMode	BOOL	Zeigt an, ob der Baustein im Handbetrieb arbeitet.
bAdjusting	BOOL	Der Konstantlichtregelbaustein befindet sich im Ausregelbetrieb. Dieses Signal kann dazu genutzt werden, Helligkeitssensoren, welche nicht analog, sondern über eine Kommunikation arbeiten, häufiger abzufragen und damit ein günstigeres Regelverhalten zu erzielen.
nRemTiMinOff	UDINT	Kontrollausgang: Die Regelausgabe befindet sich am Minimalwert und es läuft der Timer bis zum Schalten auf "0%" ab. Dieser Ausgang zeigt die verbleibenden Sekunden.
nRemTiMinOn	UDINT	Kontrollausgang: Die Konstantlichtregelung hat das Licht wegen genügend Umgebungslicht abgeschaltet, nun ist jedoch wieder Lichtbedarf vorhanden und es läuft der Timer bis zum Wiedereinschalten ab. Dieser Ausgang zeigt die verbleibenden Sekunden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.55	Tc3_BA2 ab v5.3.19.0

6.1.2.2.3.1.3.3 Sun Protection

6.1.2.2.3.1.3.3.1 Übersicht Verschattungskorrektur



6.1.2.2.3.1.3.3.2 Verschattungskorrektur: Grundlagen und Definitionen

Die Verschattungskorrektur ist in Verbindung mit der Sonnenautomatik oder Lamellennachführung nutzbar. Die Funktion prüft, ob ein Fenster oder eine Gruppe von Fenstern, die z. B. einem Raum zugeordnet sind, temporär durch umliegende Bebauung oder eigene Gebäudeteile verschattet werden. Für Fenster, welche im Schatten umliegender Gebäude oder Bäume stehen, ist ein Sonnenschutz nicht notwendig, unter Umständen sogar störend. Die Verschattungskorrektur ermittelt anhand von eingetragenen Daten der

Fassade und ihrer Umgebung, welche Teile der Fassade verschattet werden. Damit ist es dann möglich, für einzelne Fenster oder Fenstergruppen zu entscheiden, ob der Sonnenschutz aktiv sein soll.
Neben dem aktuellen Sonnenstand hängt die Verschattung der einzelnen Fenster von drei Dingen ab:

- der Ausrichtung der Fassade
- der Lage der Fenster
- der Positionierung der Verschattungsobjekte

Die folgenden Abbildungen sollen diese Zusammenhänge erläutern und die einzutragenden Parameter vorstellen.

Ausrichtung der Fassade

Betrachtung von oben

Für die reine Betrachtung des Schattenwurfes auf die Fassade ist ein zweidimensionales Koordinatensystem erforderlich, daher wurden die x- und y-Achse auf die Fassade gelegt. Der Nullpunkt liegt dabei im Fußpunkt links unten, so als würde die Fassade von vorne betrachtet. Zur Bemessung der verschattenden Objekte kommt dann noch die Z-Komponente hinzu. Deren Achse weist von der Fassade weg und hat denselben Nullpunkt, wie die x- und y-Achse.

Der horizontale Sonnenstand (Azimutwinkel) ist auf der Nordhalbkugel per Definition von der Nordrichtung aus bemessen. Die Fassadenausrichtung richtet sich ebenfalls nach der Nordrichtung, wobei die Blickrichtung aus einem Fenster der Fassade gilt:

Blickrichtung	Fassadenausrichtung
Nord	β=0°
Ost	β=90°
Süd	β=180°
West	β=270°

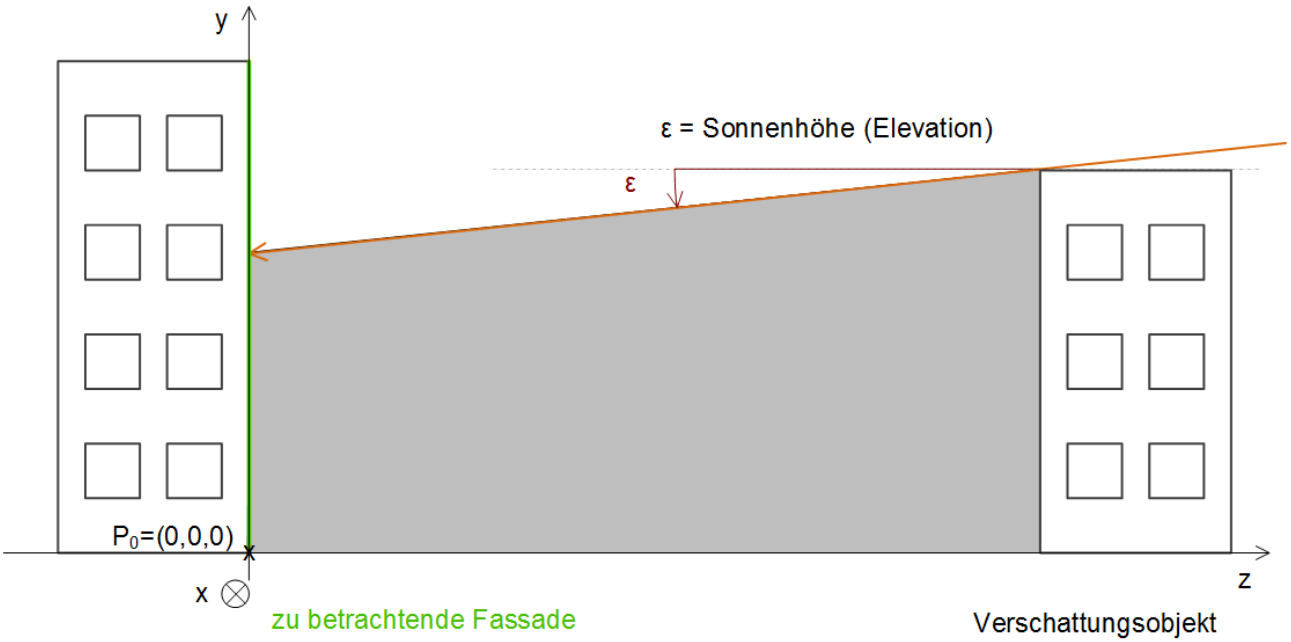
Auf der Südhalbkugel ist der Sonnenverlauf umgekehrt: Sie geht zwar auch im Osten auf, hat ihren Mittagsstand jedoch im Norden. Die Fassadenausrichtung wird diesem Verlauf angepasst:

Blickrichtung	Fassadenausrichtung
Süd	β=0°
Ost	β=90°
Nord	β=180°
West	β=270°

Die weiteren Ausführungen beziehen sich jedoch der Einfachheit halber nur auf die Anwendung auf der Nordhalbkugel. Die Berechnungen auf der Südhalbkugel gestalten sich analog und werden bei der Parametrierung des Bausteines FB_BA_ShdCorr [▶ 596] (Verschattungskorrektur) durch einen booleschen Eingang, *bSouth*, aktiviert

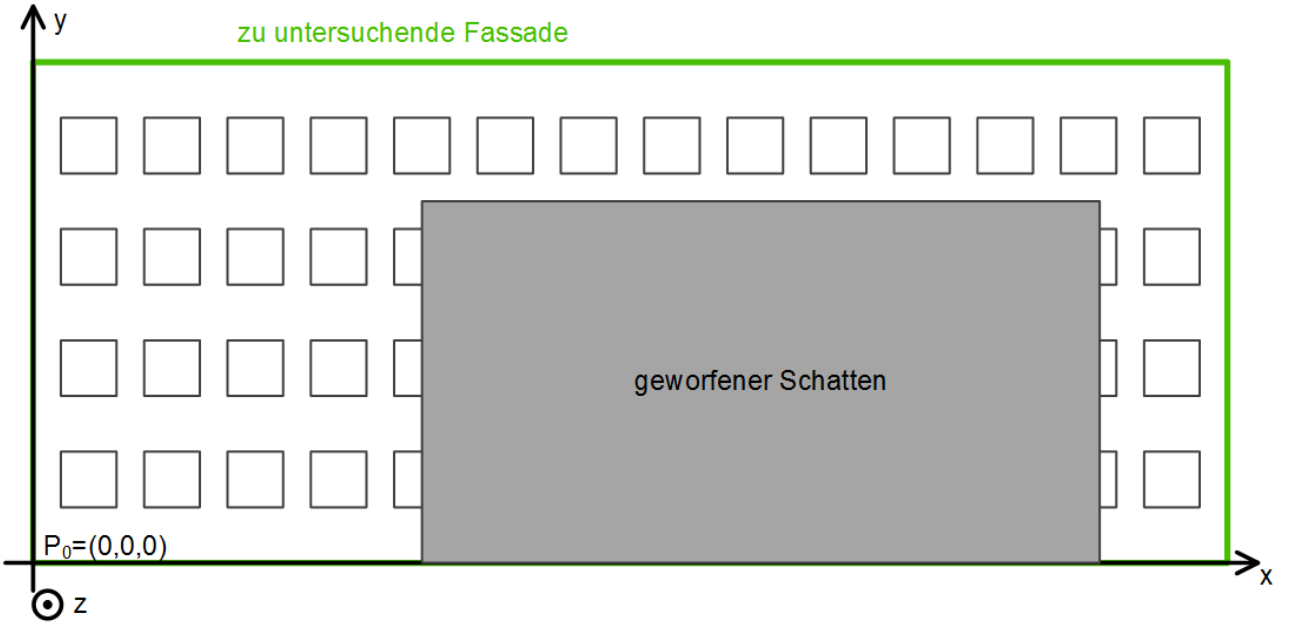
Die beiden folgenden Abbildungen sollen die Lage des Ursprungspunktes P_0 sowie die Ausrichtung des Koordinatensystems weiter verdeutlichen:

Betrachtung von der Seite



Anhand dieser Abbildung lässt sich auch der Elevationswinkel (Sonnenhöhe) darstellen: per Definition ist dieser bei Sonnenaufgang 0° (horizontaler Lichteinfall) und kann maximal 90° erreichen, dies jedoch nur an Orten innerhalb des nördlichen und südlichen Wendekreises.

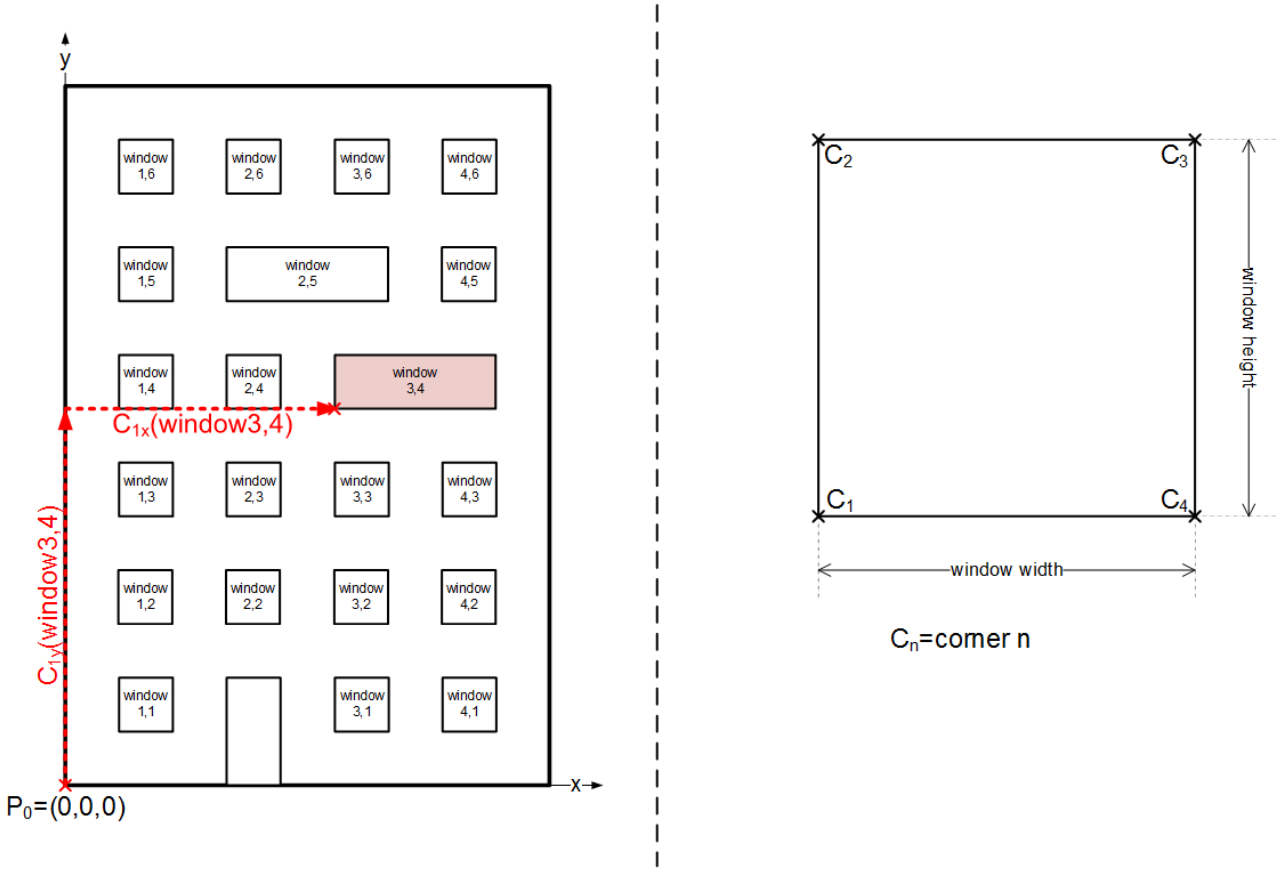
Betrachtung von vorne



Hier ist die Lage des Koordinatenursprungs, P_0, am linken unteren Fußpunkt der Fassade noch einmal besonders deutlich. Darüber hinaus ist die x-y-Ausrichtung dargestellt, die später für den Eintrag der Fensterelemente wichtig ist.

Lage der Fenster

Die Lage der Fenster wird durch die Angabe ihres linken unteren Eckpunktes in Bezug auf das Fassaden-Koordinatensystems definiert. Da ein Fenster plan auf der Fassade liegt, ist die Eingabe auf die x- und die y-Koordinate beschränkt.



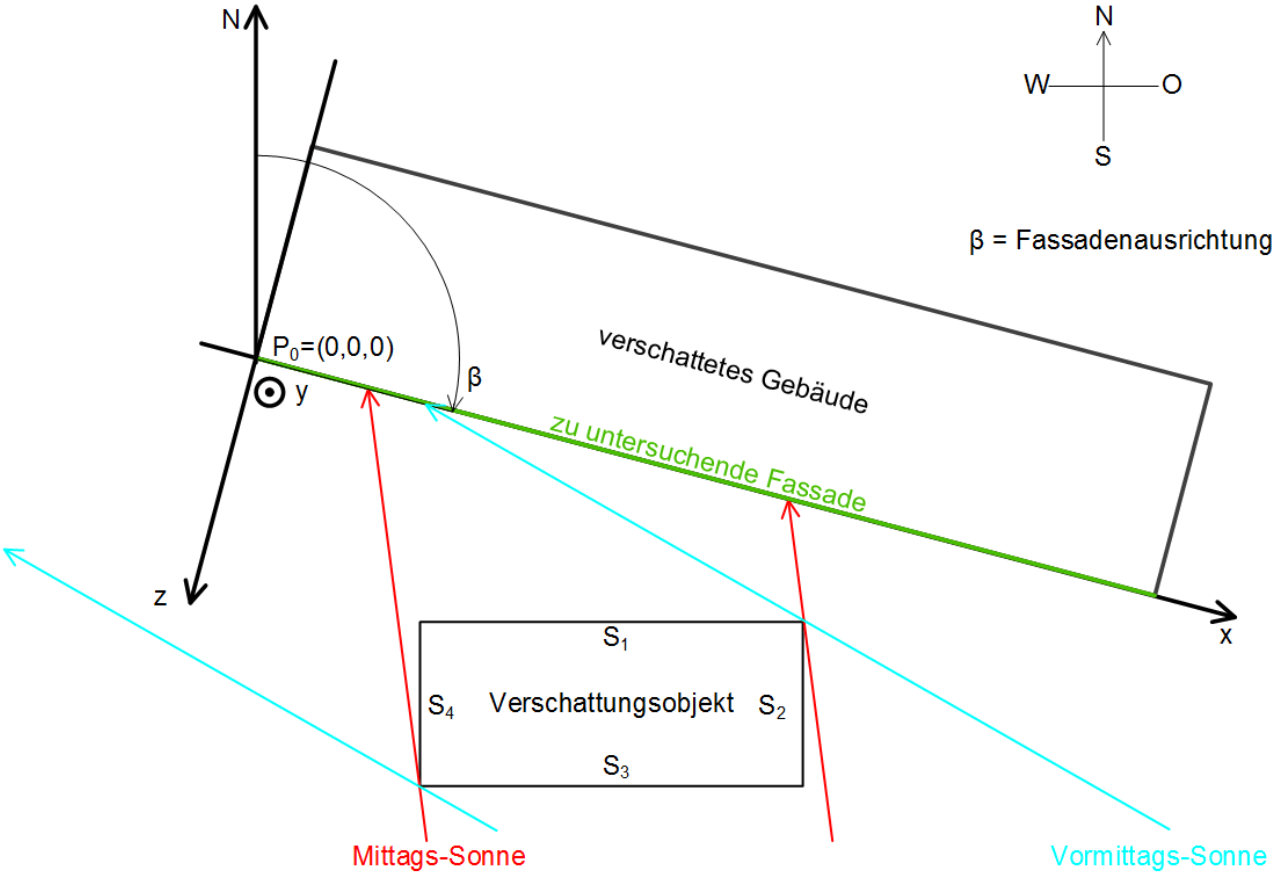
Zusätzlich sind die Fensterbreite und die Fensterhöhe anzugeben.

Aus den eingetragenen Werten wird intern die Lage jedes Fenstereckpunktes auf der Fassade ermittelt. Ein Fenster gilt dann als verschattet, wenn alle Eckpunkte im Schatten liegen.

Positionierung der Verschattungsobjekte

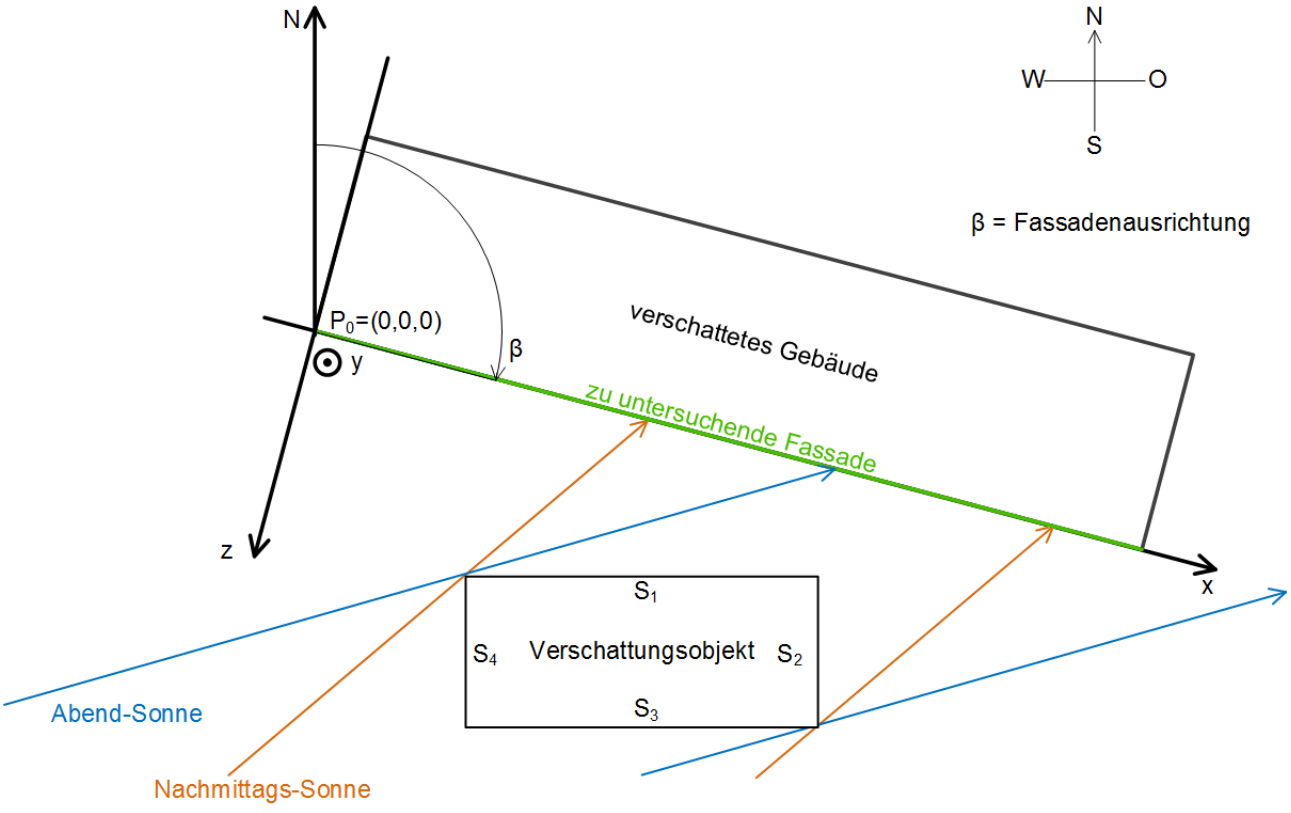
Bei der Beschreibung der Verschattungsobjekte wird zwischen eckigen Objekten (Gebäude, Pfeiler) und Objekten, die annähernd kugelförmig sind (z.B. Bäume), unterschieden. Eckige Objekte lassen sich ihrem Schattenwurf nach in viereckige schattenwerfende Fassaden unterteilen, wobei zu beachten ist, welche über den Tag hinweg den Hauptschatten werfen:

Morgens/Mittags

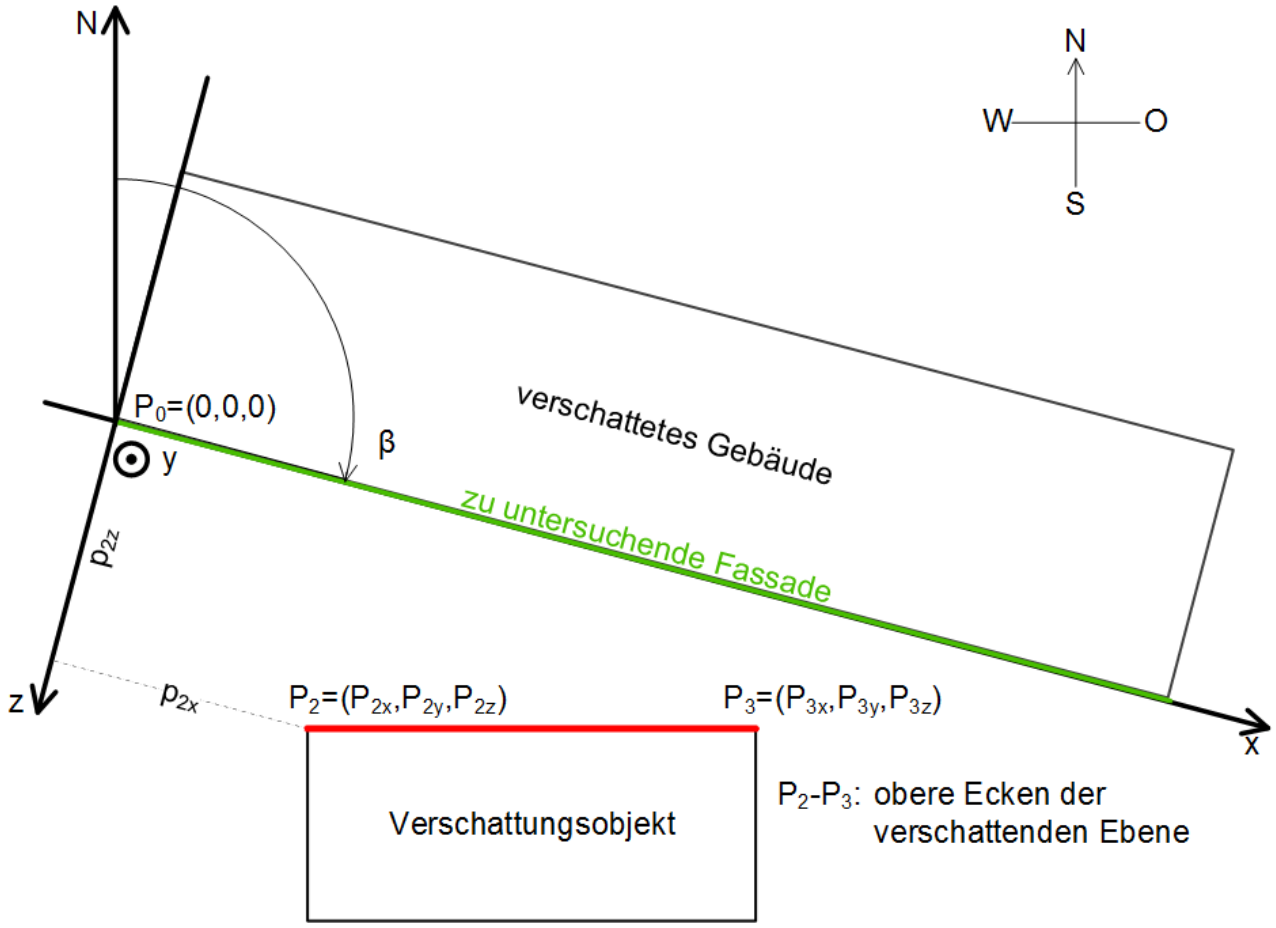


Morgens und mittags ist der geworfene Schatten hauptsächlich durch die Seiten S_1 und S_4 gegeben, eine Betrachtung von S_2 und S_3, sollten sie nicht höher sein, ist nicht notwendig.

Nachmittags/Abends



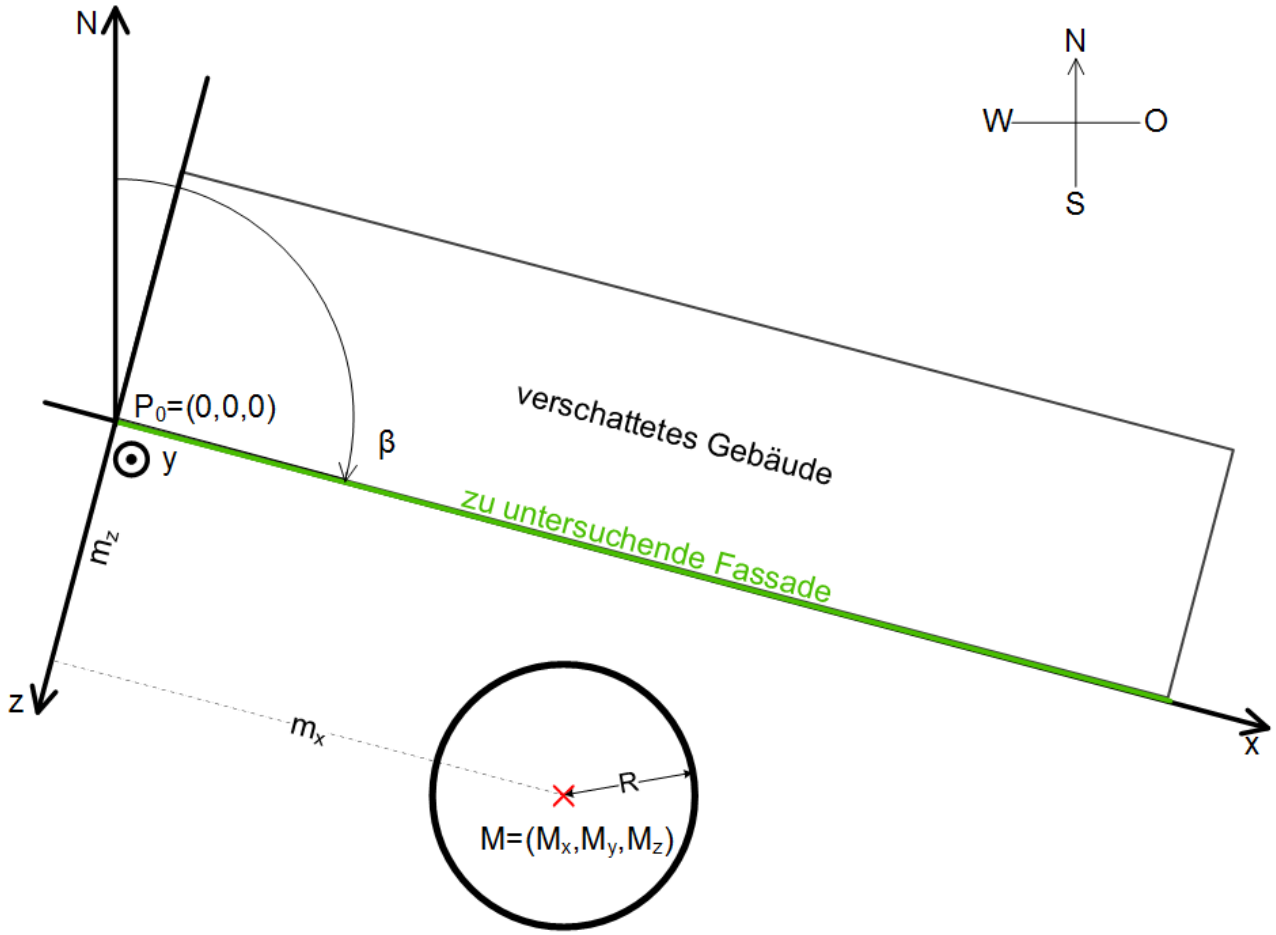
Am Nachmittag und am Abend lässt sich der Summen-Schatten allein durch die Betrachtung von S_1 und S_2 ermitteln. Es reicht also in diesem Fall S_1, S_2 und S_4 als Schattenwerfer anzugeben. Die Eingabe erfolgt dabei anhand der vier Eckpunkte bzw. deren Koordinaten in Bezug auf den Fassaden-Nullpunkt:



In dieser Skizze sind wegen der Draufsicht nur die oberen Punkte, P_2 und P_3 dargestellt. Der untere Punkt P_1 liegt unter P_2 und P_4 unter P_3.

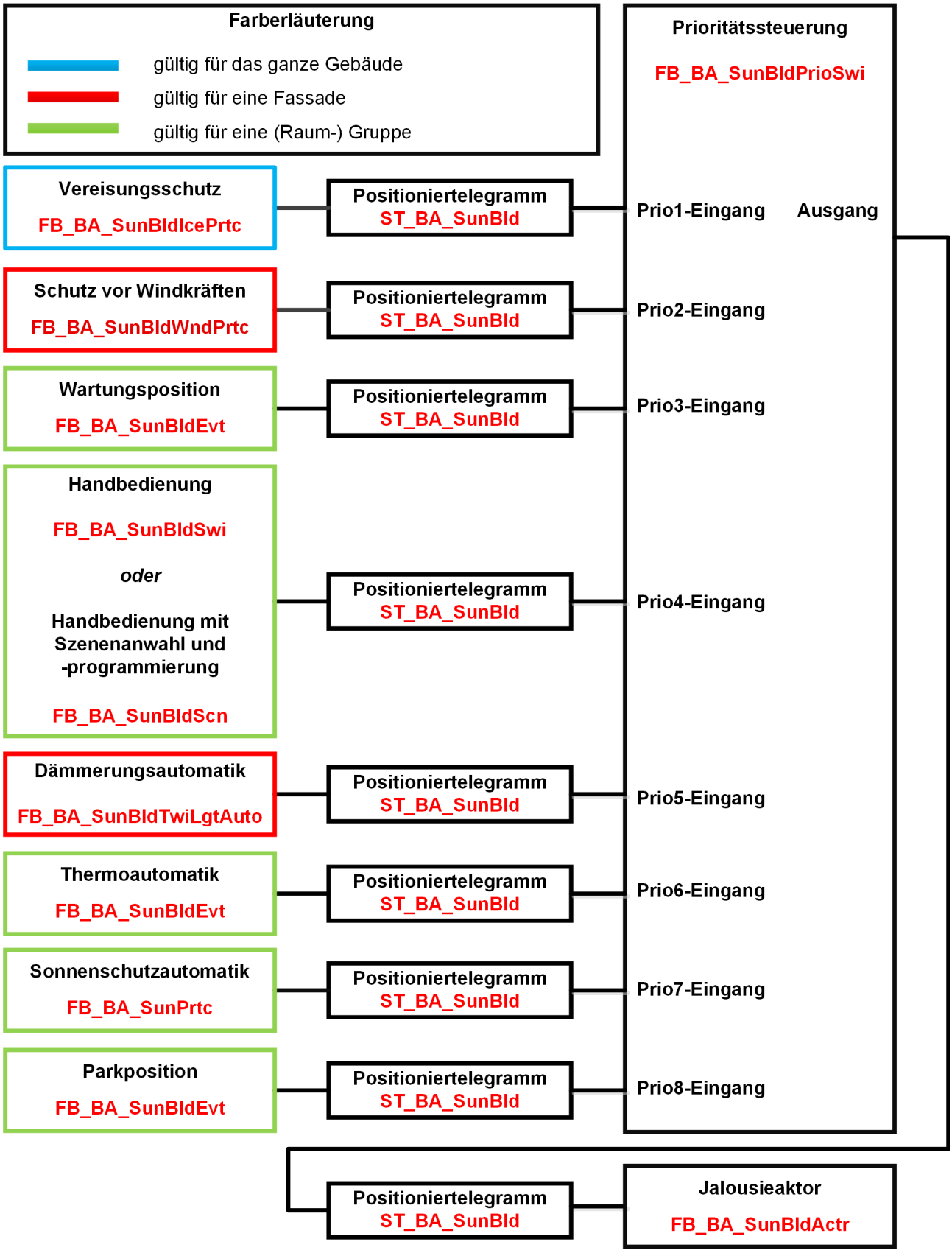
Die Eingabe von verschattenden Kugelelementen erfolgt durch die Eingabe des Kugelmittelpunktes und des Radius:

Kugelelemente



Eine "Einteilung" des Kugelelementes, wie beim eckigen Gebäude, ist nicht notwendig, da der Schattenwurf einer Kugel sich nur in seiner Richtung, nicht aber in seiner Größe ändert.

6.1.2.2.3.1.3.3.3 Übersicht Sonnenschutzautomatik



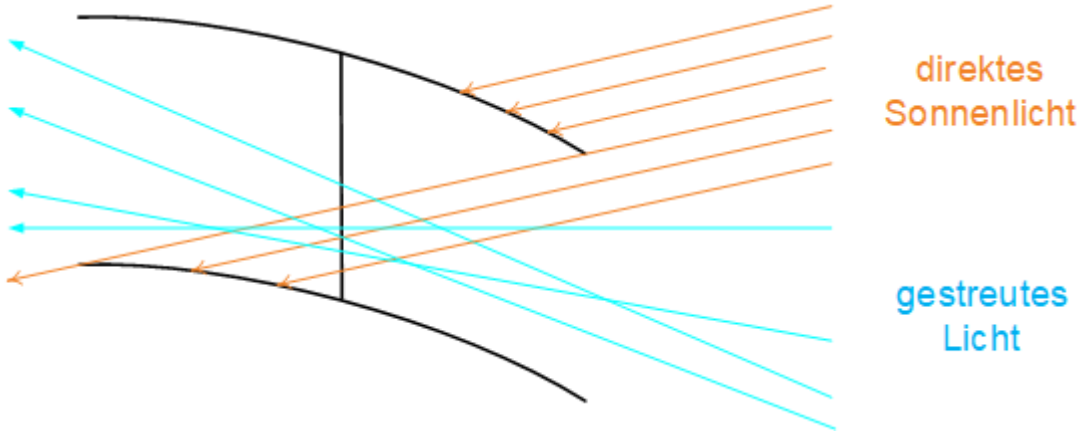
6.1.2.2.3.1.3.3.4 Sonnenschutz: Grundlagen und Definitionen

Direkter Einfall von Tageslicht wird von in Räumen befindlichen Personen als störend angesehen. Auf der anderen Seite jedoch empfindet der Mensch das natürliche Licht als angenehmer im Vergleich zum künstlichen Licht. Zwei Möglichkeiten des Blendschutzes sollen hier vorgestellt werden:

- Lamellennachführung
- Höhenverstellung

Lamellennachführung

Eine Jalousie mit Lamellen, welche sich nachführen lassen, bietet hier die Möglichkeit eines intelligenten Sonnenschutzes. Dabei wird die Stellung der Lamellen zyklisch dem aktuellen Sonnenstand angepasst, so dass kein direktes Tageslicht durch die Jalousien fällt, jedoch möglichst viel diffuses Tageslicht genutzt werden kann.



Die Abbildung zeigt, dass gestreutes Licht von unten noch einfallen kann, während vom direkten Tageslicht gerade nichts mehr, bzw. theoretisch nur noch ein Strahl hindurchtritt. Zur Berechnung des Lamellenwinkels sind folgende Parameter notwendig:

- die aktuelle Sonnenhöhe (Elevationswinkel)
- der Sonnenstand, d.h. der Azimut Winkel
- die Fassadenausrichtung
- die Lamellenbreite
- der Lamellenabstand

Effektiver Elevationswinkel

Wird die Jalousie im Schnitt wie oben betrachtet, so hängt der Lichteinfallswinkel nicht allein von der Sonnenhöhe (Elevation) ab, sondern auch von der Sonnenrichtung:

- Sind Fassadenausrichtung und Sonnenstand (Azimut) gleich, trifft die Sonne also direkt auf die Fassade, so ist der effektive Lichteinfallswinkel gleich dem aktuellen Elevationswinkel.
- Fällt die Sonne jedoch von der Sonnenrichtung aus gesehen schräg auf die Fassade, so ist bei gleichem Elevationswinkel der effektive Winkel größer.

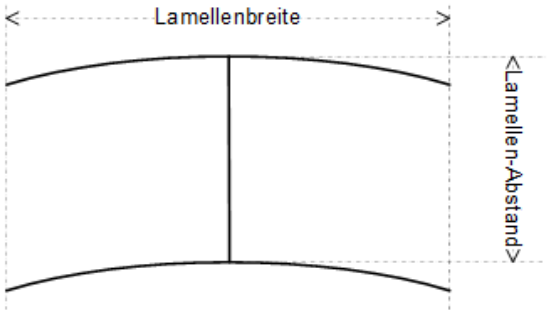
Dieser Zusammenhang kann leicht mit einem rechtwinkligen Zeichendreieck, welches hochkant auf dem Tisch steht, verdeutlicht werden: Direkt von der Seite ist ein Dreieck mit zwei 45°-Winkeln und einem 90° Winkel zu sehen. Wird das Dreieck gedreht, so wird die auf dem Tisch liegende Seite scheinbar kürzer und die beiden ehemaligen 45°-Winkel ändern sich. Das Dreieck scheint steiler zu werden.

Im Folgenden soll daher vom „effektiven Elevationswinkel“ gesprochen werden, der Teil des Lichtes, welcher direkt auf die Jalousie trifft.

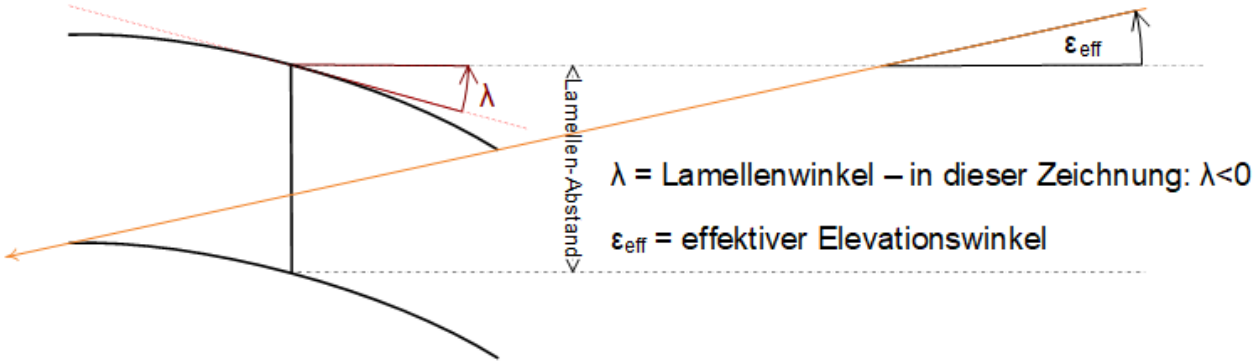
Die folgenden drei Bilder zeigen den Zusammenhang zwischen effektivem Elevationswinkel und den Jalousiemaßen und wie sich der resultierende Lamellenwinkel λ im Laufe des Tages ändert:

Lamellenwinkel

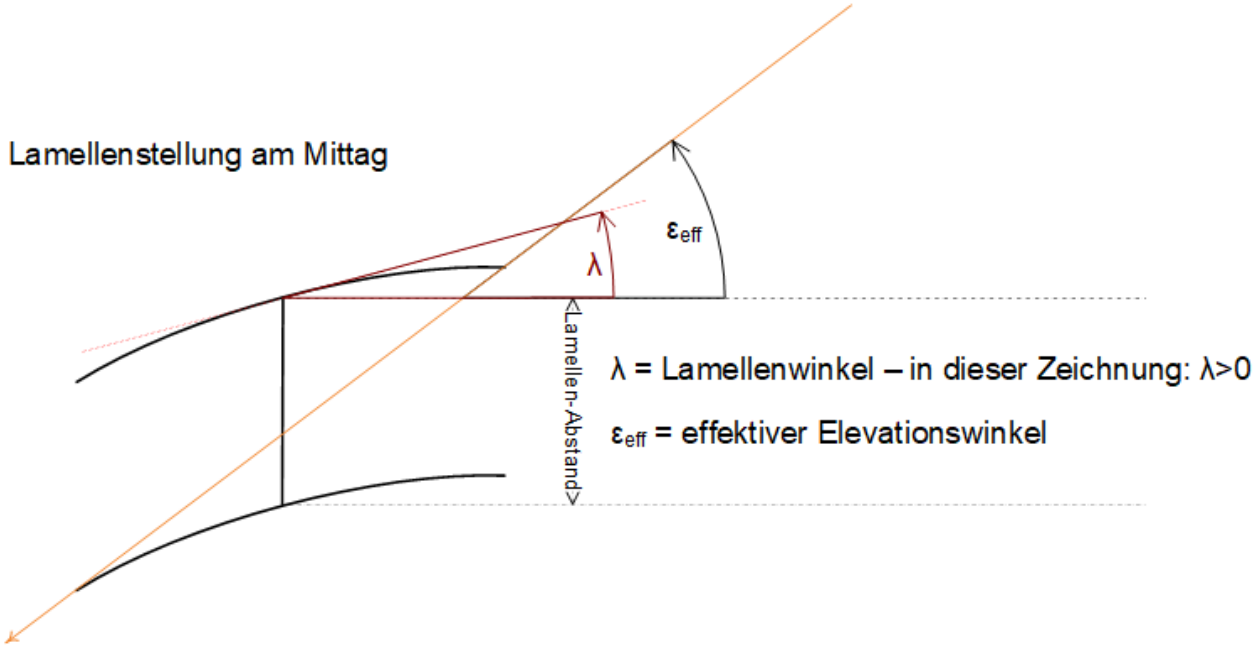
Lamellen bei einen Winkel von λ=0



Lamellenstellung am Morgen und am Abend



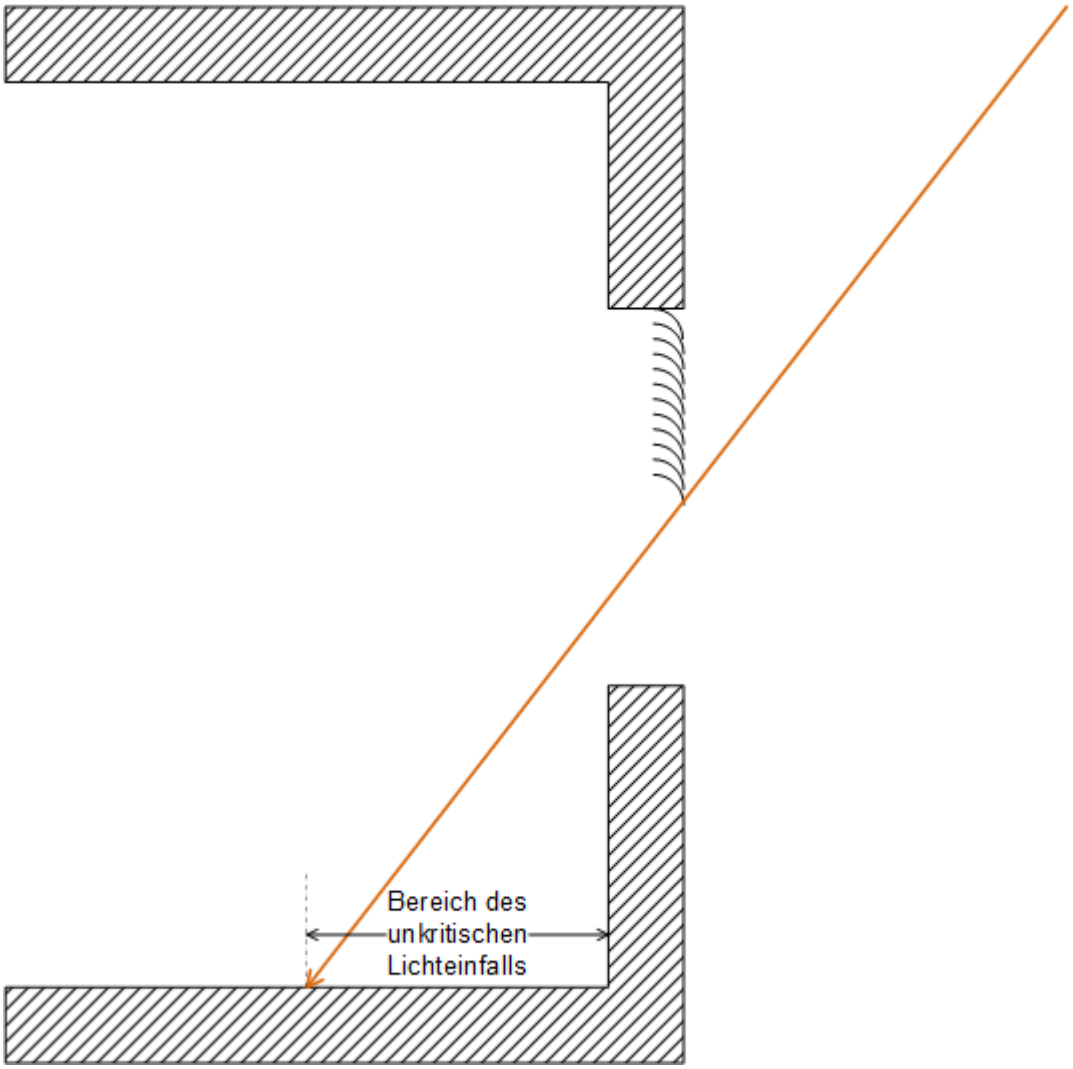
Lamellenstellung am Mittag



Tiefergehende Informationen zu dieser Thematik enthält das Kapitel Effektiver Elevationswinkel [▶ 345].

Höhenverstellung

Bei einem hohen Sonnenstand in der Mittagszeit dringen die direkten Sonnenstrahlen nicht in die volle Tiefe des Raumes ein. Wenn direkte Sonnenstrahlen im Bereich der Fensterbrüstung als unkritisch betrachtet werden, kann die Höhe des Sonnenschutzes automatisch so angepasst werden, dass die Sonnenstrahlen immer nur bis zu einer unkritischen Tiefe in den Raum eindringen.

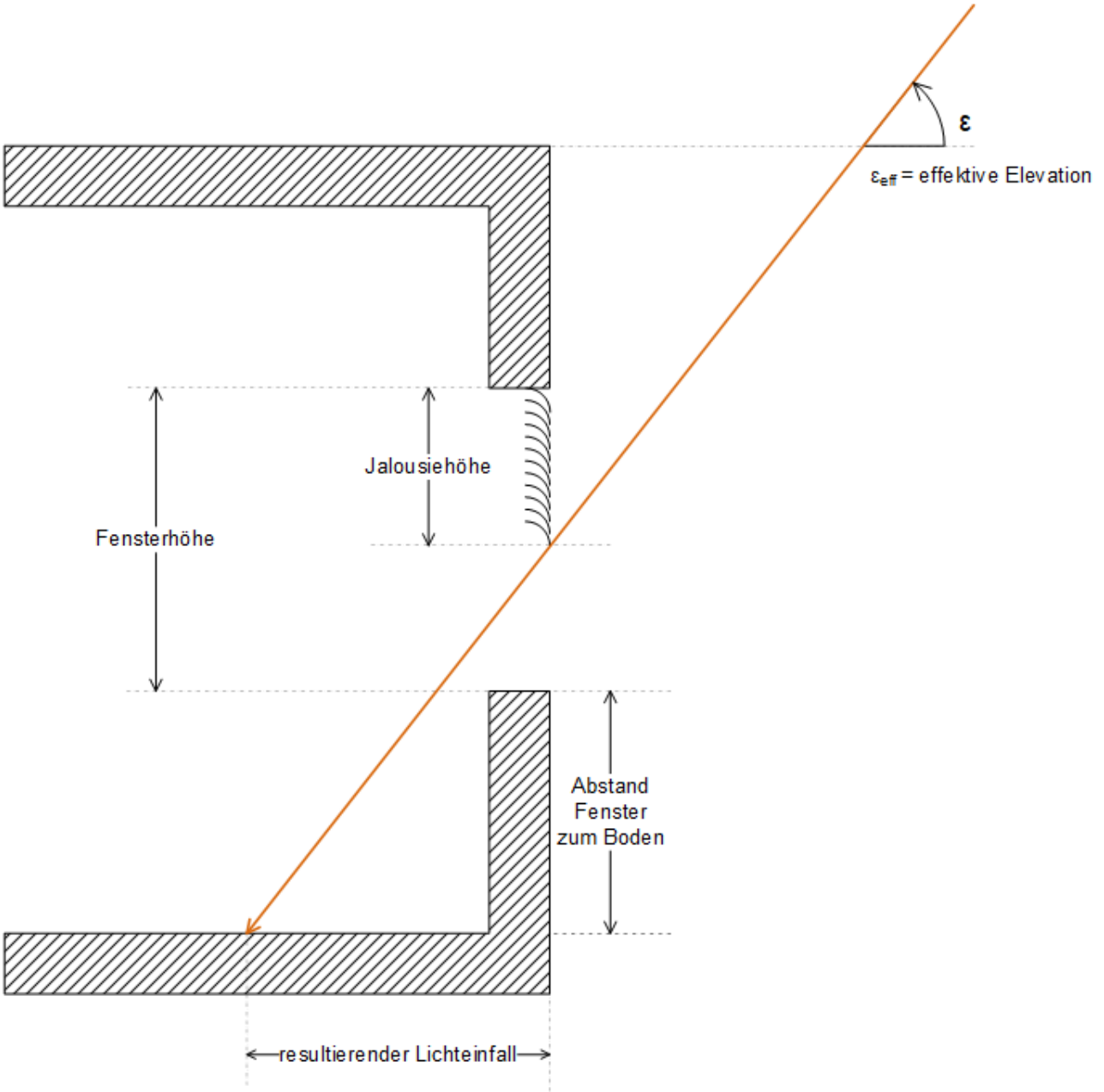


Um zu jedem Zeitpunkt die entsprechende Jalousiehöhe berechnen zu können, die sicherstellt, dass die Sonneneinstrahlung einen gewissen Wert nicht überschreitet, sind folgende Werte nötig.

Zur Berechnung der jeweiligen Jalousiehöhe erforderlich:

- Sonnenhöhe (Elevation)
- Fensterhöhe
- Abstand Fenster zum Boden

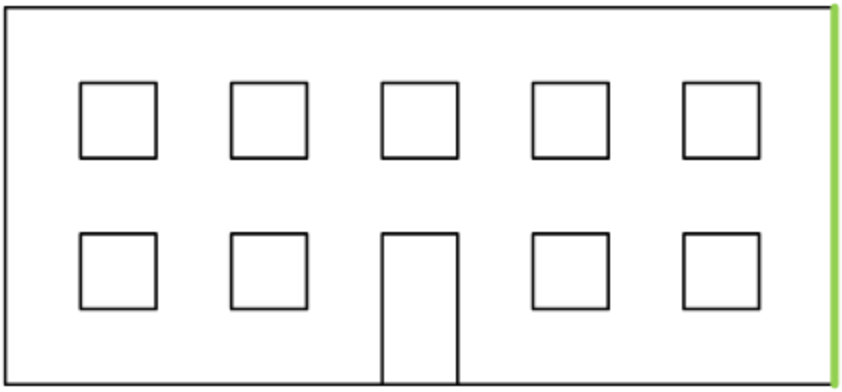
Die folgende Abbildung veranschaulicht, wo diese Parameter einzuordnen sind:



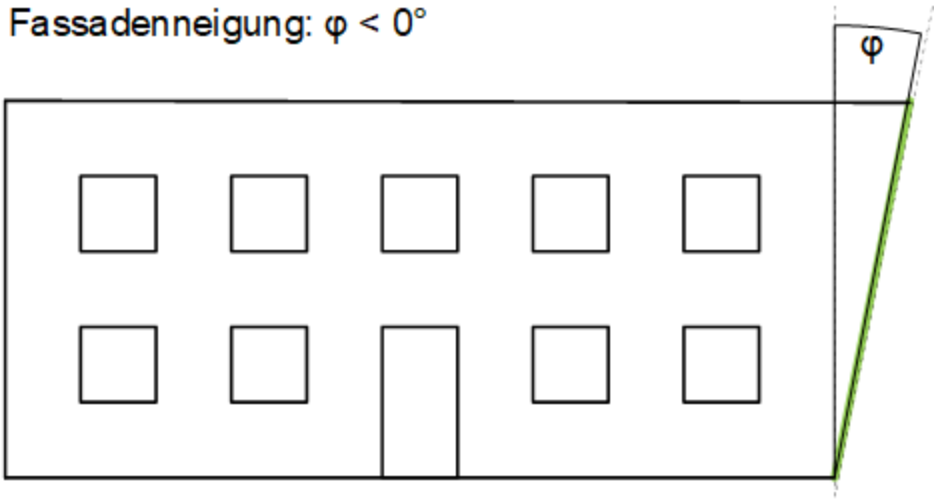
Einfluss der Fassadenneigung

Bei den beiden beschriebenen Methoden des Sonnenschutzes wurde davon ausgegangen, dass die Fassade und damit die Fenster senkrecht zum Boden stehen. Bei einer geneigten Fassade jedoch ändert sich der Lichteinfall, so dass dieser Einfluss berücksichtigt wird. Die Fassadenneigung wird wie folgt definiert:

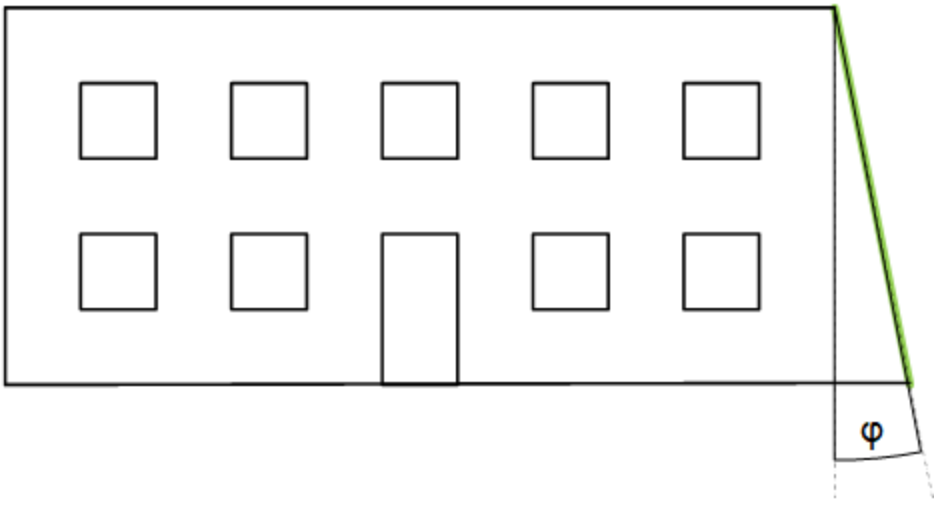
Fassadenneigung: φ = 0°



Fassadenneigung: φ < 0°

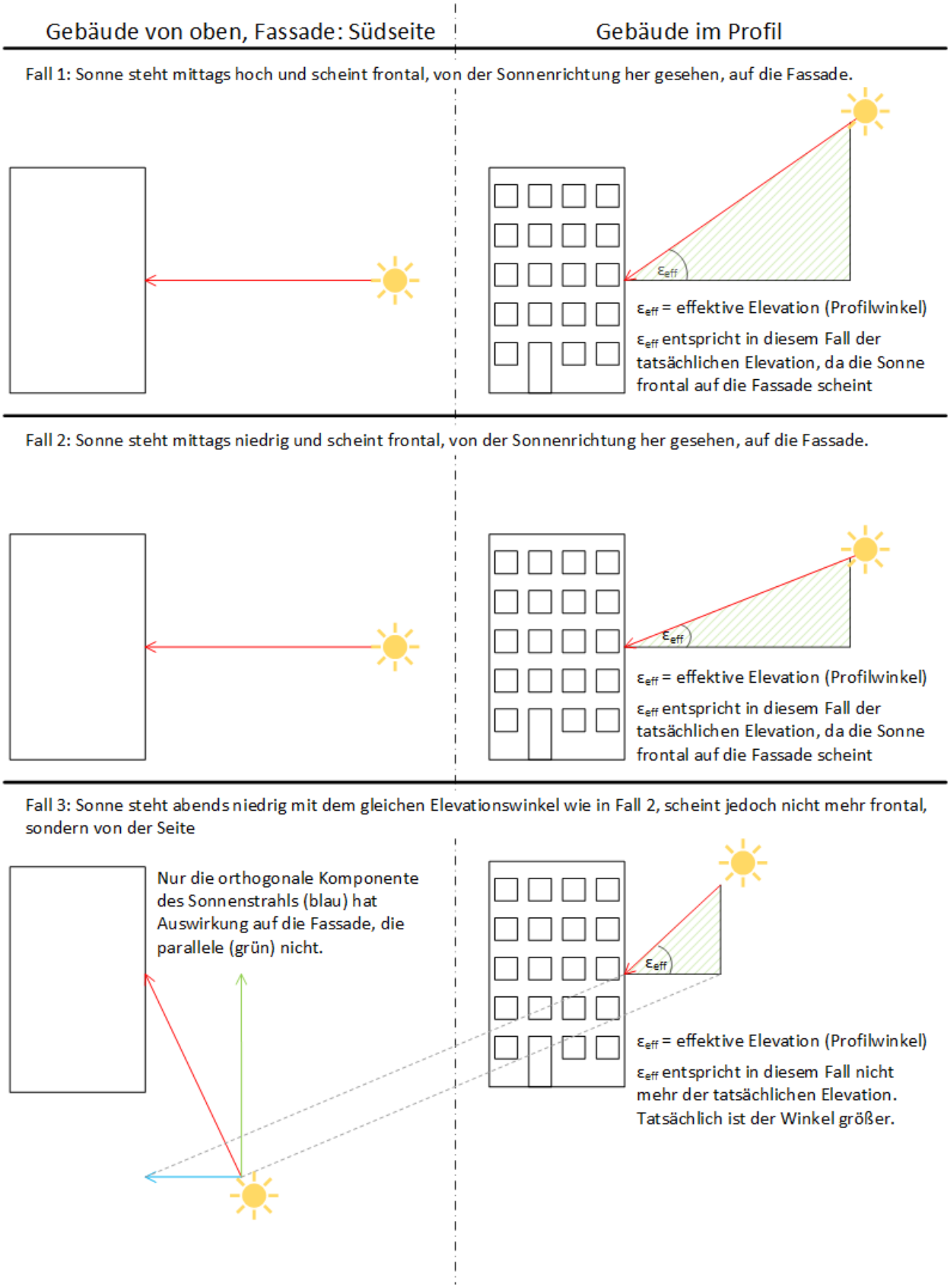


Fassadenneigung: φ > 0°



6.1.2.2.3.1.3.3.4.1 Effektiver Elevationswinkel

Dieses Kapitel beschreibt das Verhältnis vom Sonneneinfall auf die Fassade zum Elevationswinkel.



Fall 1 zeigt einen typischen Sonneneinfall mittags auf eine Südfassade.

Die Sonne steht im Vergleich zu morgens und abends hoch.

Das grün schraffierte Dreieck stellt den Elevationswinkel ε dar.

Von der Sonnenrichtung (Azimut) her gesehen wirkt nur die orthogonale Komponente direkt auf die Fassade!

Daher ist in diesem Fall der Elevationswinkel ε gleich dem für die Berechnung effektiven Elevationswinkel ε_{eff} bzw. dem Profilwinkel.

Fall 2 zeigt einen typischen Sonneneinfall mittags auf eine Südfassade, jedoch niedriger.

Das grün schraffierte Dreieck stellt den Elevationswinkel ε dar, der im Vergleich zu Fall1 kleiner ausfällt.

Auch hier ist der Elevationswinkel ε gleich dem für die Berechnung effektiven Elevationswinkel ε_{eff} bzw. dem Profilwinkel.

Fall 3 zeigt einen typischen Sonneneinfall abends auf eine Südfassade. Der Elevationswinkel ist gleich dem in Fall 2. Damit soll Fall 3 die zeitliche Fortsetzung von Fall1 am selben Tag darstellen.

Das grün schraffierte Dreieck aus Fall 2 (Gebäude im Profil) ist nun nach vorne hin gekippt.

Es stellt jedoch die orthogonale Beleuchtungskomponente dar. Der Elevationswinkel für diese Komponente ist sichtbar größer als in Fall 2, wo die Sonne orthogonal auf die Fassade scheint.

6.1.2.2.3.1.3.3.5 Liste der Verschattungselemente

Die Daten aller Verschattungsobjekte (Gebäudeteile, Bäume, etc.) pro Fassade werden innerhalb des Programms in einem Feld von Strukturelementen des Typs ST_BA_ShdObj [▶ 279] hinterlegt.

Die Verschattungskorrektur FB_BA_ShdCorr [▶ 375] liest die Informationen dieser Liste. Der Verwaltungsbaustein FB_BA_ShdObjEntry [▶ 378] jedoch liest und beschreibt sie als Ein-/Ausgangsvariable.

Es empfiehlt sich daher, diese Liste global zu deklarieren:

```
VAR_GLOBAL
     aShdObj : ARRAY[1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Die Variable *nSunPrt_MaxShdObj* stellt dabei die Obergrenze der zur Verfügung stehenden Elemente dar und ist global als Konstante innerhalb der Programmbibliothek deklariert:

```
VAR_GLOBAL CONSTANT
     nSunPrt_MaxShdObj : UINT := 20;
END_VAR
```

6.1.2.2.3.1.3.3.6 Liste der Fassadenelemente

Die Daten aller Fenster (Fassadenelemente) pro Fassade werden innerhalb des Programms in einem Feld von Strukturelementen des Typs ST_BA_FcdElem [▶ 278] hinterlegt.

Der Verwaltungsbaustein FB_BA_FcdElemEntry [▶ 352] sowie die Verschattungskorrektur FB_BA_ShdCorr [▶ 375] lesen und beschreiben diese Liste (letzterer setzt die Verschattungsinformation) und greifen somit als Ein-/Ausgangsvariable auf dieses Feld zu.

Es empfiehlt sich daher, diese Liste global zu deklarieren:

```
VAR_GLOBAL
     aFcdElem : ARRAY[1..BA_Param.nSunPrt_MaxRowFcd, 1..BA_Param.nSunPrt_MaxColumnFcd] OF ST_BA_FcdE
lem;
END_VAR
```

Die Variablen *nSunPrt_MaxColumnFcd* und *nSunPrt_MaxRowFcd* definieren dabei die Obergrenzen der zur Verfügung stehenden Elemente und sind global als Konstante innerhalb der Programmbibliothek deklariert:

```
VAR_GLOBAL CONSTANT
     nSunPrt_MaxRowFcd     : UINT := 10;
     nSunPrt_MaxColumnFcd  : UINT := 20;
END_VAR
```

6.1.2.2.3.1.3.3.7 FB_BA_BldPosEntry

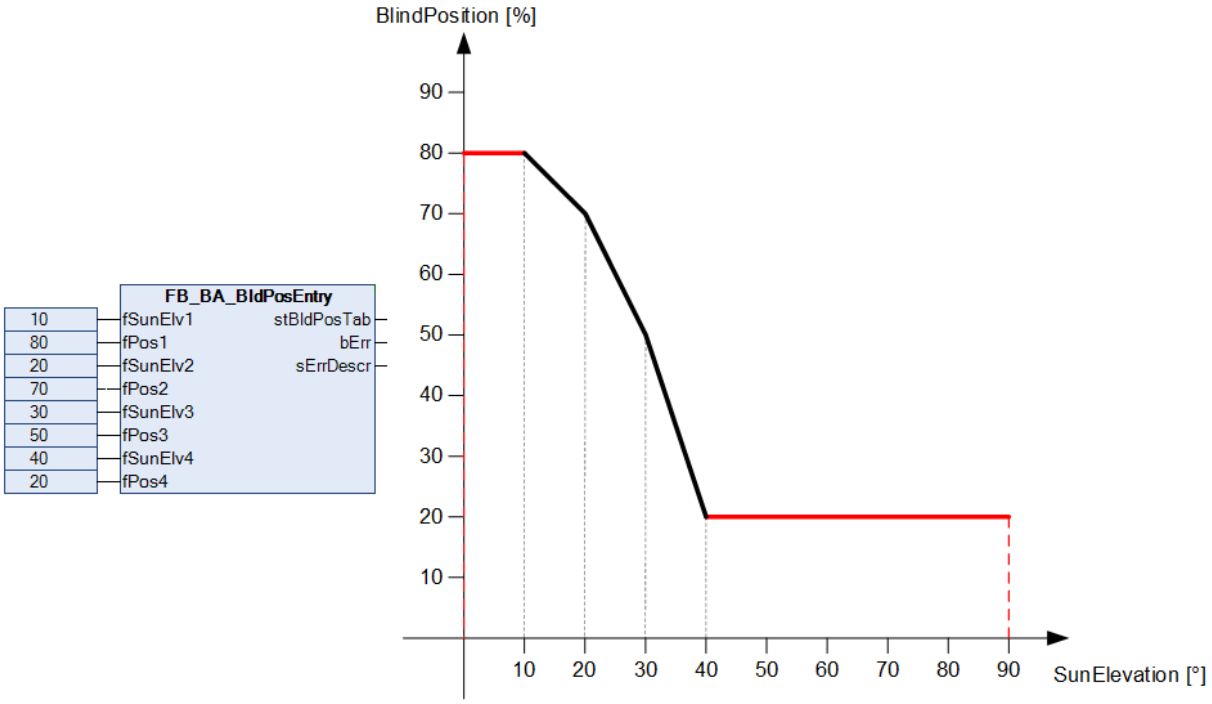


Der Funktionsbaustein FB_BA_BldPosEntry dient zur Eingabe von Stützstellen für den Baustein FB_BA_SunPrtc [▶ 406], falls dieser im Modus der Höhenpositionierung mit Hilfe einer Tabelle betrieben wird (siehe E_BA_PosMod).

Der Baustein FB_BA_SunPrtc [▶ 406] bietet neben den Betriebsarten "fixe Jalousiehöhe" und "maximaler Lichteinfall" auch die Möglichkeit, die Jalousiehöhe per Tabelleneinträge in Abhängigkeit von der Sonnenhöhe zu steuern. Durch die Eingabe mehrerer Stützpunkte wird linear interpoliert die betreffende Jalousiehöhe zum entsprechenden Sonnenstand errechnet. Da falsch eingetragene Werte jedoch zu Fehlfunktionen beim FB_BA_SunPrtc [▶ 406] führen können ist diesem der Baustein FB_BA_BldPosEntry voranzustellen. Es lassen sich vier Stützpunkte an diesem Baustein parametrieren, wobei ein fehlender Eintrag als Nulleintrag gewertet wird.
Der Baustein sortiert die eingegebenen Werte nicht selbstständig, sondern achtet darauf, dass die eingetragenen Sonnenstände der jeweiligen Stützstellen in aufsteigender Reihenfolge eingegeben wurden. Unbeabsichtigt fehlerhafte Einträge fallen dadurch schneller auf.
Die gewählten Werte für *fSunElv1* ... *fSunElv4* müssen auch eindeutig sein, es darf beispielsweise nicht gelten:
[*fSunElv1* = 10 ; *fPos1* = 50] und gleichzeitig [*fSunElv2* = 10 *; fPos2* = 30].
Das würde bedeuten, dass für ein und denselben Wert zwei verschiedene Zielwerte bestünden, was keinen eindeutigen funktionalen Zusammenhang bilden lässt.
Darüber hinaus müssen die Einträge für Sonnenstand und Jalousiehöhe im gültigen Bereich liegen.
Das bedeutet mathematisch, dass folgende Bedingungen erfüllt sein müssen:

- *fSunElv1* < *fSunElv2* < *fSunElv3* < *fSunElv4* - (Werte aufsteigend und ungleich)
- 0 ≤ *fSunElv* ≤ 90 ([°] - Gültigkeitsbereich Quellwerte)
- 0 ≤ *fPos* ≤ 100 (in Prozent - Gültigkeitsbereich Zielwerte)

Der Baustein überprüft die eingetragenen Werte auf diese Bedingungen hin und gibt bei Nichterfüllen eine Fehlermeldung aus. Zusätzlich wird der Kontrollwert *bVld* von ST_BA_BldPosTab [▶ 278] auf FALSE gesetzt.
Des Weiteren sorgt der Baustein selbstständig für ein Ausfüllen der Randbereiche: Intern wird ein weiterer Stützpunkt bei *fSunElv* = 0 mit *fPos1* und ein weiterer oberhalb von *fSunElv4* bei *fSunElv* = 90 mit *fPos4* aufgestellt. Damit wird sichergestellt, dass für alle gültigen Eingabewerte 0 ≤*fSunElv* ≤ 90 ein sinnvoller Zielwert vorhanden ist, **ohne** dass ein Eintrag für *fSunElv* = 0 und *fSunElv* = 90 vom Anwender zwingend vergeben werden muss:



Die tatsächliche Anzahl an Stützstellen, welche an den Baustein FB_BA_SunPrtc [▶ 406] übergeben wird, erhöht sich damit auf 6, siehe ST_BA_BldPosTab [▶ 278].

Die Interpolation der Werte erfolgt im Blendschutz-Baustein.

Eingänge

```
VAR_INPUT
  fSunElv1   : REAL;
  fPos1      : REAL;
  fSunElv2   : REAL;
  fPos2      : REAL;
  fSunElv3   : REAL;
  fPos3      : REAL;
  fSunElv4   : REAL;
  fPos4      : REAL;
END_VAR
```

Name	Typ	Beschreibung
fSunElv1	REAL	Sonnenstand des 1. Stützpunktes [°] (0…90).
fPos1	REAL	Jalousieposition (Grad der Schließung) des 1. Stützpunktes [%] (0…100).
fSunElv2	REAL	Sonnenstand des 2. Stützpunktes [°] (0…90).
fPos2	REAL	Jalousieposition (Grad der Schließung) des 2. Stützpunktes [%] (0…100).
fSunElv3	REAL	Sonnenstand des 3. Stützpunktes [°] (0…90).
fPos3	REAL	Jalousieposition (Grad der Schließung) des 3. Stützpunktes [%] (0…100).
fSunElv4	REAL	Sonnenstand des 4. Stützpunktes [°] (0…90).
fPos4	REAL	Jalousieposition (Grad der Schließung) des 4. Stützpunktes [%] (0…100).

Ausgänge

```
VAR_OUTPUT
  stBldPosTab  : ST_BA_BldPosTab;
  bErr         : BOOL;
  sErrDescr    : T_MAXSTRING;
END_VAR
```

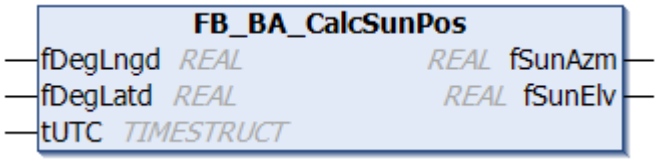
Name	Typ	Beschreibung
stBldPosTab	ST_BA_BldPosTab [▶ 278]	Übergabestruktur der Stützstellen
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Die x-Werte (Elevationswerte) in der Tabelle sind entweder nicht in aufsteigender Reihenfolge oder doppelt vorhanden.
02: Fehler: Ein eingetragener Elevationswert ist außerhalb des gültigen Bereiches von 0°…90°.
03: Fehler: Ein eingetragener Positionswert ist außerhalb des gültigen Bereiches von 0%…100%.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

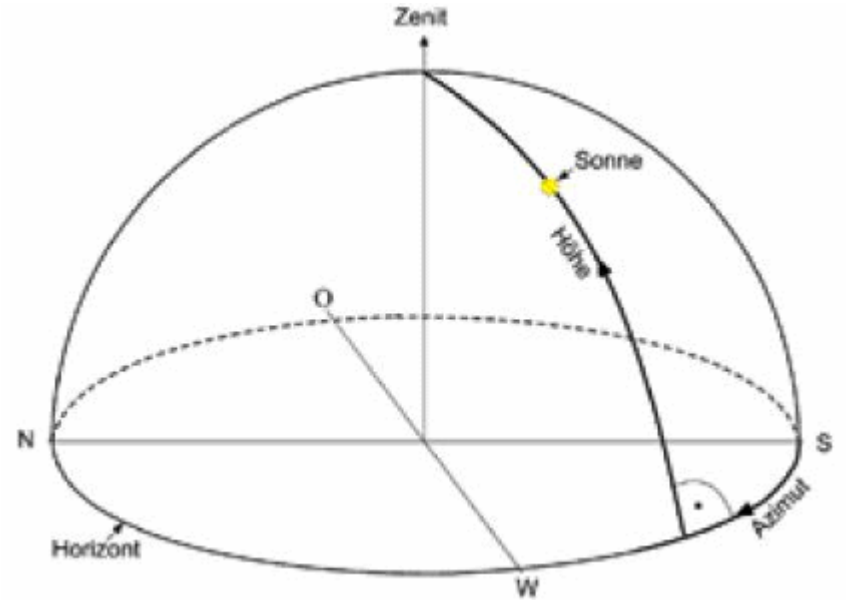
6.1.2.2.3.1.3.3.8 FB_BA_CalcSunPos



Der Funktionsbaustein FB_BA_CalcSunPos dient zur Berechnung des Sonnenstandes durch die Angabe von Datum, Uhrzeit, geografischer Länge und geografischer Breite.

Der Sonnenstand für einen gegebenen Zeitpunkt lässt sich nach gängigen Methoden mit definierter Genauigkeit berechnen. Für Anwendungen mit mäßigen Anforderungen genügt der hier vorliegende Baustein. Als Grundlage wurde hierfür der SUNAE-Algorithmus verwendet, der einen günstigen Kompromiss zwischen Genauigkeit und Rechenaufwand darstellt.

Der Stand der Sonne an einem festen Beobachtungsort wird normalerweise mit zwei Winkelangaben bestimmt. Die eine Winkelangabe zeigt die Höhe über dem Horizont an, wobei 0° bedeutet, dass sich die Sonne in der Horizontalebene des Standortes befindet und ein Wert von 90°, dass sie sich senkrecht über dem Beobachter befindet. Die andere Winkelangabe zeigt die Richtung an, in der die Sonne steht. Bei dem SUNAE-Algorithmus wird unterschieden, ob der Beobachter auf der nördlichen (Längengrad > 0) oder auf der südlichen (Längengrad < 0) Erdhalbkugel steht. Ist der Beobachtungspunkt auf der nördlichen Erdhalbkugel, so wird ein Wert von 0° für die nördliche Sonnenrichtung zugeordnet und läuft dann im Uhrzeigersinn um den Kompass, d.h. 90° ist Osten, 180° ist Süden, 270° Westen, etc. Ist der Beobachtungspunkt auf der südlichen Erdhalbkugel, so entspricht 0° der südlichen Richtung und verläuft dann entgegen dem Uhrzeigersinn, d.h. 90° ist Osten, 180° ist Norden, 270° Westen, etc.



Bei der Angabe der Uhrzeit muss die Zeit als koordinierte Weltzeit (UTC, Universal Time Coordinated - im älteren Sprachgebrauch auch GMT, Greenwich-Mean-Time) angegeben werden.

Die geografische Breite ist die im Winkelmaß (also [°]) angegebene nördliche oder südliche Entfernung eines Ortes der Erdoberfläche vom Äquator. Die Breite kann Werte von 0° (am Äquator) bis ±90° (an den Polen) annehmen. Dabei gibt ein positives Vorzeichen die nördliche Richtung und ein negatives Vorzeichen die südliche Richtung an. Die geografische Länge ist ein Winkel, der ausgehend vom Nullmeridian 0° (künstlich festgelegte Nord-Süd-Linie) Werte bis ±180° annehmen kann. Ein positives Vorzeichen gibt die Länge in östlicher Richtung und ein negatives Vorzeichen in westlicher Richtung an. Beispiele:

Ort	geographische Länge	geographische Breite
Sydney, Australien	151,2°	-33,9°
New York, USA	-74,0°	40,7°
London, England	-0,1°	51,5°
Moskau, Russland	37,6°	55,7°
Peking, China	116,3°	39,9°
Dubai, Vereinigte Arabische Emirate	55,3°	25,4°
Rio de Janeiro, Brasilien	-43,2°	-22,9°
Hawaii, USA	-155,8°	20,2°
Verl, Deutschland	8,5°	51,9°

Gibt der Baustein *FB_BA_CalcSunPos* für die Sonnenhöhe *fSunElv* einen negativen Wert zurück, so ist die Sonne nicht sichtbar. Dieses kann zur Bestimmung von Sonnenaufgang und Sonnenuntergang genutzt werden.

Eingänge

```
VAR_INPUT
  fDegLngd    : REAL;
  fDegLatd    : REAL;
  tUTC        : TIMESTRUCT;
END_VAR
```

Name	Typ	Beschreibung
fDegLngd	REAL	Geographische Länge [°]
fDegLatd	REAL	Geographische Breite [°]
tUTC	TIMESTRUCT	Eingabe der aktuellen Uhrzeit als koordinierte Weltzeit. Mit Hilfe des Bausteines FB_BA_GetTime [▶ 412] kann diese Zeit aus einem Zielsystem gelesen werden.

Ausgänge

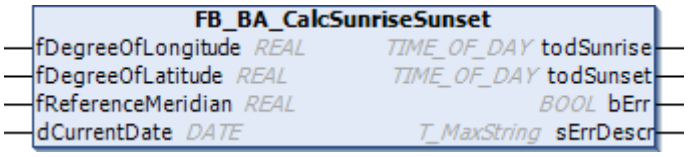
```
VAR_OUTPUT
  fSunAzm     : REAL;
  fSunElv     : REAL;
END_VAR
```

Name	Typ	Beschreibung
fSunAzm	REAL	Sonnenrichtung (nördliche Erdhalbkugel: 0° Norden ... 90° Osten ... 180° Süden ... 270° Westen ... / südliche Erdhalbkugel: 0° Süden ... 90° Osten ... 180° Norden ... 270° Westen ...).
fSunElv	REAL	Sonnenhöhe (0° horizontal ... 90° senkrecht).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.9 FB_BA_CalcSunriseSunset



Der Baustein FB_BA_CalcSunriseSunset dient der Berechnung des Sonnenaufgangs und des Sonnenuntergangs durch die Angabe der geographischen Länge, der geographischen Breite, des Bezugsmeridian und der Uhrzeit.

Die Erde ist in mehrere Zeitzonen unterteilt. Jeder Zeitzone ist ein Bezugsmeridian zugeordnet. Bezugsmeridian einiger Zeitzonen:

Zeitzone	Bezugsmeridian	Zeit
GMT (Greenwich Mean Time)	$\lambda_{GMT} = 0°$	GMT + 0h
MEZ (Mitteleuropäische Zeit)	$\lambda_{MEZ} = 15°$	GMT + 1h
MESZ (Mitteleuropäische Sommerzeit)	$\lambda_{MESZ} = 30°$	GMT + 2h

Eingänge

```
VAR_INPUT
  fDegreeOfLongitude       : REAL := 8.5;
  fDegreeOfLatitude        : REAL := 51.9;
  fReferenceMeridian       : REAL;
  dCurrentDate             : DATE;
END_VAR
```

Name	Typ	Beschreibung
fDegreeOfLongitude	REAL	Geographische Länge (Längengrad) in Grad. Östliche Längen sind positiv, westliche Längen negativ. Dieser Eingang wird intern auf -180°…180° beschränkt. Dennoch wird bei einer Eingabe außerhalb dieser Grenzen ein Fehler am Ausgang *bErr* angezeigt.
fDegreeofLatitude	REAL	Geographische Breite (Breitengrad) in Grad. Nördliche Breiten sind positiv, südliche Breiten negativ. Dieser Eingang wird intern auf -66.55°…66.55° beschränkt. Dennoch wird bei einer Eingabe außerhalb dieser Grenzen ein Fehler an Ausgang *bErr* angezeigt. **Info:** -66.55° und 66.55° bezeichnen die Breitengrade des südlichen und nördlichen Polarkreises. Über diese Grenzen hinaus kann an bestimmten Tagen die Sonne nie auf- bzw. untergehen. Eine Aussage über Sonnenauf- und Sonnenuntergang ist dann nicht möglich.
fReferenceMeridian	TIMESTRUCT	Bezugsmeridian von der Zeitzone. Östliche Meridiane sind positiv, westliche Meridiane negativ. Da sich der Bezugsmeridian auf Längengrad bezieht, ist auch diese Eingabe auf -180°…180° beschränkt. Dennoch wird bei einer Eingabe außerhalb dieser Grenzen ein Fehler am Ausgang *bErr* angezeigt.
dCurrentDate	DATE	Aktuelles Datum.

Ausgänge

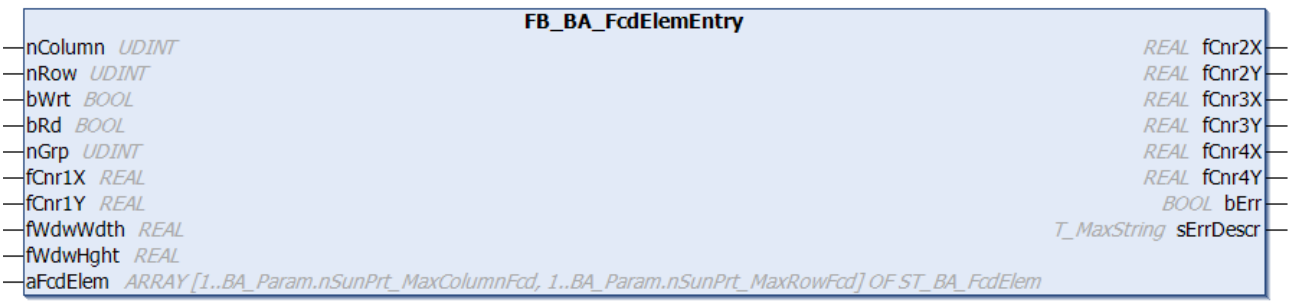
```
VAR_OUTPUT
  todSunrise     : TOD;
  todSunset      : TOD;
  bErr           : BOOL;
  sErrDescr      : T_MaxString;
END_VAR
```

Name	Typ	Beschreibung
todSunrise	TOD	Sonnenaufgang. Ausgabe der Stunde und der Minute.
todSunset	TOD	Sonnenuntergang. Ausgabe der Stunde und der Minute.
bErr	BOOL	Ergebniskontrolle für die eingegebenen Werte.
sErrDescr	T_MaxString	Enthält die Beschreibung des Fehlers, welcher intern am höchsten priorisiert ist.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.1.3.3.10 FB_BA_FcdElemEntry



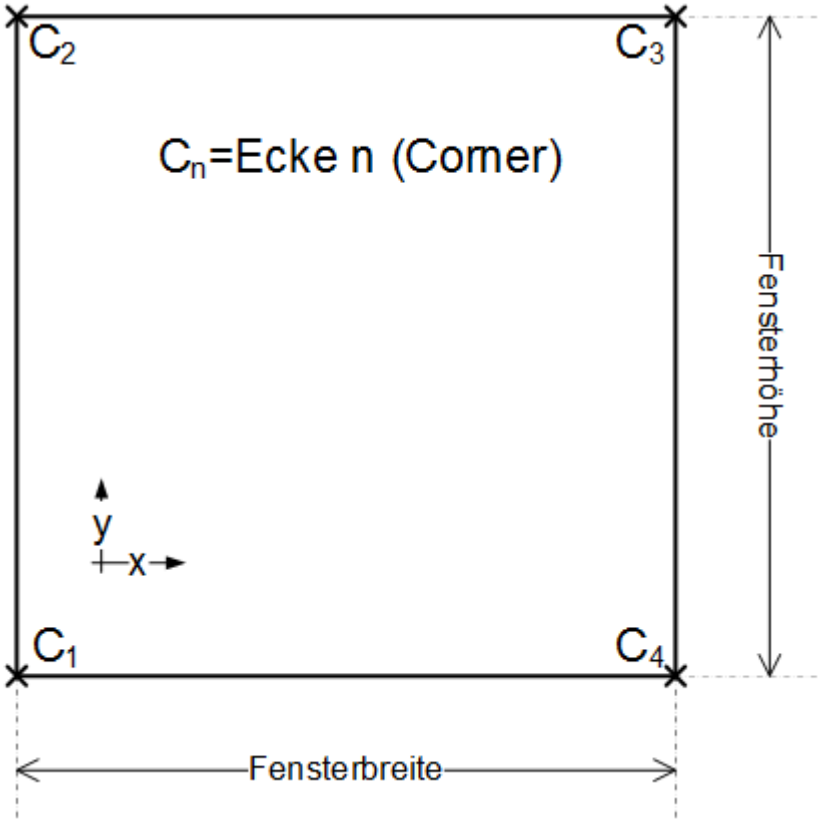
Der Funktionsbaustein FB_BA_FcdElemEntry dient zur Verwaltung aller Fassadenelemente (Fenster) einer Fassade, welche global in einer Liste von Fassadenelementen [▶ 346] hinterlegt ist. Er soll die Eingabe der Elementinformationen - auch im Hinblick auf die Nutzung der TC3 PLC HMI - erleichtern. Eine schematische Darstellung der Objekte mit Beschreibung der Koordinaten ist unter Verschattungskorrektur: Grundlagen und Definitionen [▶ 332] gegeben.

Die Deklaration der Fassadenelemente erfolgt in den globalen Variablen als zweidimensionales Feld über die Fensterspalten (Columns) und -reihen (Rows):

```
VAR_GLOBAL
     aFcdElem : ARRAY[1..Param.nSunPrt_MaxColumnFcd, 1..Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Jedes einzelne Element *arrFcdElem[x,y]* trägt die Informationen für jeweils ein Fassadenelement (ST_BA_FcdElem [▶ 278]). Dazu gehören die Gruppenzugehörigkeit, die Abmessungen (Breite, Höhe) und die Koordinaten der Eckpunkte. Der Baustein greift dabei über die IN-OUT-Variable *aFcdElem* direkt auf dieses Feld zu.

Anmerkung: Die Tatsache, dass die Koordinaten der Eckpunkte C2 bis C4 Ausgangswerte sind, ergibt sich daraus, dass sie aus den Eingabeparametern gebildet werden und für die Verwendung in einer Visualisierung zur Verfügung stehen sollen:



Alle Angaben in [m]!

fCnr2X = *fCnr1X*
fCnr2Y = *fCnr1Y* + *fWdwHght* (Fensterhöhe)
fCnr3X = *fCnr1X* + *fWdwWdth* (Fensterbreite)
fCnr3Y = *fCnr2Y*
fCnr4X = *fCnr1X* + *fWdwWdth* (Fensterbreite)
fCnr4Y = *fCnr1Y*

Die Verwendung des Bausteines erfolgt in drei Schritten:

- Auslesen
- Ändern
- Schreiben

Auslesen

Mit den Einträgen an *nColumn* und *nRow* wird das entsprechende Element aus der Liste, *aFcdElem[nColumn, nRow]*, ausgewählt. Eine steigende Flanke an *bRd* liest folgende Daten aus dem Listenelement aus:

- nGrp Gruppenzugehörigkeit,
- fCnr1X X-Koordinate des Eckpunktes1 [m]
- fCnr1Y Y-Koordinate des Eckpunktes1 [m]
- fWdwWdth Fensterbreite [m]
- fWdwHght Fensterhöhe [m]

Diese werden dann den entsprechenden Eingangsvariablen des Bausteines zugewiesen, der daraus nach dem oben erläuterten Zusammenhang die Koordinaten der Eckpunkte C2-C4 als Ausgangsvariablen errechnet. Wichtig ist hierbei, dass im Schritt des Auslesens die Eingangswerte nicht überschrieben werden. So lassen sich alle Werte zunächst in einer Visualisierung anzeigen.

Ändern

In einem nächsten Programmschritt können die aufgeführten Eingabewerte verändert werden. Die eingegebenen Werte werden dabei ständig auf Plausibilität überprüft. Der Ausgang *bErr* zeigt an, ob die Werte gültig sind (*bErr*=FALSE). Wenn die Werte ungültig sind, wird am Ausgang *sErrDescr* eine entsprechende Fehlermeldung ausgegeben. Siehe auch unten "Fehler (*bErr*=TRUE)".

Schreiben

Mit einer positiven Flanke an *bWrt* werden die parametrierten Daten in das von *nRow* und *nColumn* abhängige Feld des Arrays *aFcdElem* geschrieben, unabhängig davon, ob sie gültige Werte darstellen oder nicht. Daher ist innerhalb der Elementstruktur ST_BA_FcdElem [▶ 278] ebenfalls ein Plausibilitätsbit *bVld* vorhanden, das genau diese Information an den Baustein FB_BA_ShdCorr [▶ 375] weiterreicht und dort Fehlberechnungen vorbeugt.

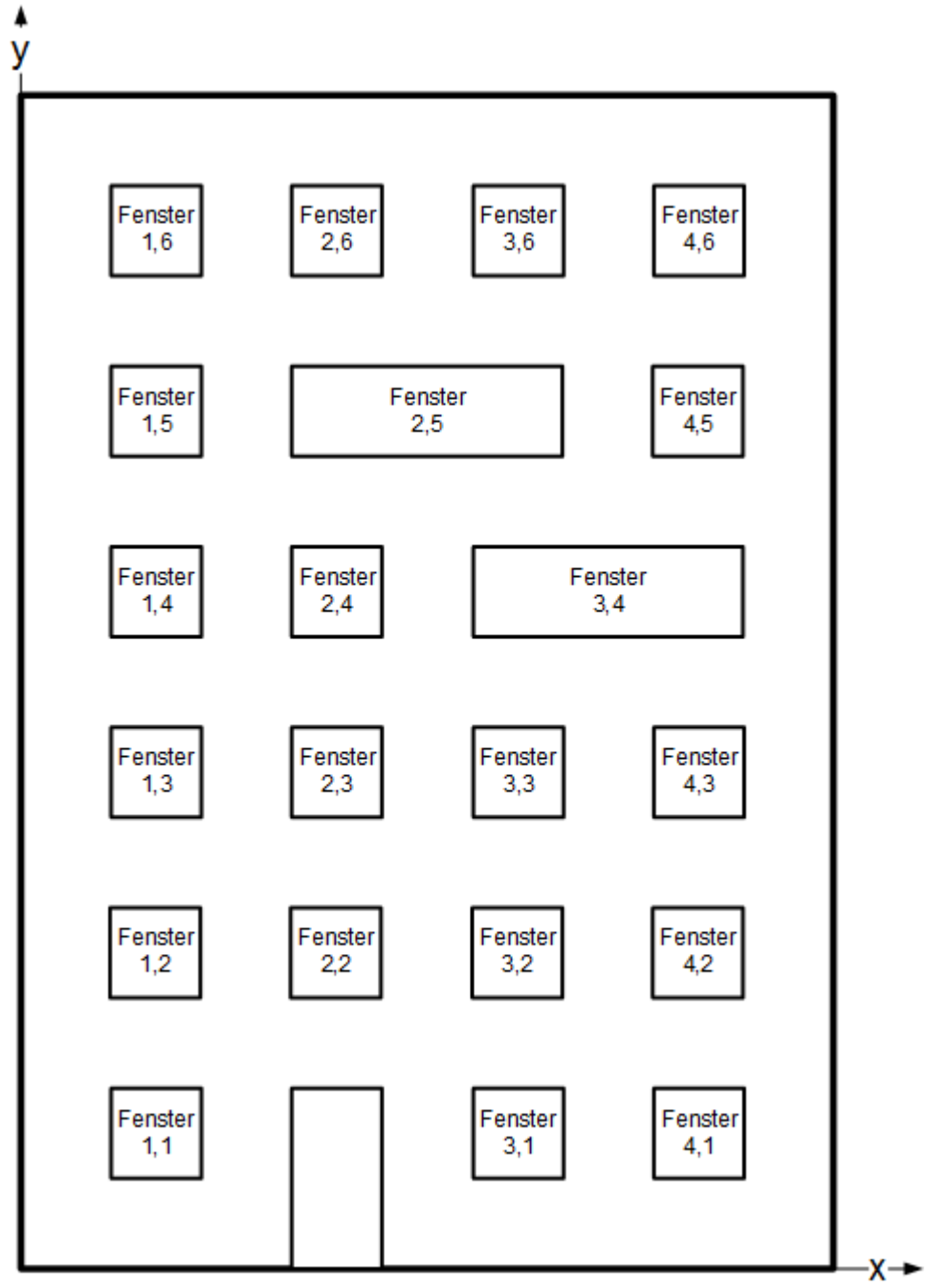
Diese Vorgehensweise soll nur als Vorschlag angesehen werden. Es ist natürlich auch möglich, den Baustein ganz normal in einem Schritt zu parametrieren und die eingetragenen Werte mit einer steigenden Flanke an *bWrt* in das entsprechende Listenelement zu schreiben.

Fehler (bErr=TRUE)

Der Baustein FB_BA_ShdCorr [▶ 375], welcher beurteilt, ob alle Fenster einer Gruppe verschattet sind, wird nur dann seine Aufgabe ausführen, wenn alle Fenster der betrachteten Gruppe gültige Einträge haben. Das bedeutet:

- *nGrp* muss größer als 0 sein
- *fCnr1X* muss größer oder gleich 0.0 sein
- *fCnr1Y* muss größer oder gleich 0.0 sein
- *fWdwWdth* muss größer als 0 sein
- *fWdwHght* muss größer als 0 sein

Ist eines dieser Kriterien nicht erfüllt, so wird dies als Falscheingabe interpretiert und der Fehlerausgang *bErr* am Bausteinausgang von *FB_BA_FcdElemEntry* gesetzt. Innerhalb des Fensterelementes ST_BA_FcdElem [▶ 278] wird das Plausibilitätsbit *bVld* auf FALSE gesetzt.
Sind hingegen **alle** Einträge eines Fassadenelementes Null, wird es als gültiges, gewollt ausgelassenes Fassadenelement angesehen:



Bei einer Fassade von 6x4 Fenstern wären hier die Elemente Fenster (2,1), Fenster (3,5) und Fenster (4,4) Leerelemente.

Eingänge

```
VAR_INPUT
  nColumn    : UDINT;
  nRow       : UDINT;
  bWrt       : BOOL;
  bRd        : BOOL;
  nGrp       : UDINT;
  fCnr1X     : REAL;
  fCnr1Y     : REAL;
  fWdwWdth   : REAL;
  fWdwHght   : REAL;
END_VAR
```

Name	Typ	Beschreibung
nColumn	UDINT	Spalten-Index des gewählten Elementes auf der Fassade. Dies bezieht sich auf die Auswahl eines Feldelementes des unter der IN-OUT-Variablen aFcdElem angelegten Arrays.
nRow	UDINT	dto. Reihen-Index. *nRow* **und** *nColumn* **dürfen nicht Null sein!** Das ergibt sich aus der Felddefinition, welche jeweils mit 1 beginnt, siehe oben.
bRd	BOOL	Mit einer positiven Flanke an diesem Eingang werden die Informationen des gewählten Elementes, *aFcdElem[nColumn, nRow]* in den Baustein gelesen und den Eingangsvariablen *nGrp* bis *fWdwHght* zugewiesen. Daraus ergeben sich dann die Ausgangsvariablen *fCnr2X* bis *fCnr4Y*. Sind zu Zeitpunkt des Auslesens schon Daten an den Eingängen *nGrp* bis *fWdwHght* angelegt, so werden die zuvor ausgelesenen Daten sofort mit diesen überschrieben.
bWrt	BOOL	Eine positive Flanke schreibt die eingegebenen sowie errechneten Werte in das gewählte Feldelement *aFcdElem[nColumn, nRow]*.
nGrp	UDINT	Gruppenzugehörigkeit. Intern begrenzt auf einen Minimalwert von 0.
fCnr1X	REAL	X-Koordinate des Eckpunktes1 [m]
fCnr1Y	REAL	Y-Koordinate des Eckpunktes1 [m]
fWdwWdth	REAL	Fensterbreite [m]
fWdwHght	REAL	Fensterhöhe [m]

Ausgänge

```
VAR_OUTPUT
  fCnr2X   : REAL;
  fCnr2Y   : REAL;
  fCnr3X   : REAL;
  fCnr3Y   : REAL;
  fCnr4X   : REAL;
  fCnr4Y   : REAL;
  bErr     : BOOL;
  sErrDesc : T_MAXSTRING;
END_VAR
```

Name	Typ	Beschreibung
fCnr2X	REAL	Ermittelte X-Koordinate des Eckpunktes 2 des Fensters [m] (siehe Funktionsbeschreibung [▶ 353])
fCnr2Y	REAL	Ermittelte Y-Koordinate des Eckpunktes 2 des Fensters [m] (siehe Funktionsbeschreibung [▶ 353])
fCnr3X	REAL	Ermittelte X-Koordinate des Eckpunktes 3 des Fensters [m] (siehe Funktionsbeschreibung [▶ 353])
fCnr2Y	REAL	Ermittelte Y-Koordinate des Eckpunktes 3 des Fensters [m] (siehe Funktionsbeschreibung [▶ 353])
fCnr4X	REAL	Ermittelte X-Koordinate des Eckpunktes 4 des Fensters [m] (siehe Funktionsbeschreibung [▶ 353])
fCnr2Y	REAL	Ermittelte Y-Koordinate des Eckpunktes 4 des Fensters [m] (siehe Funktionsbeschreibung [▶ 353])
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Indexfehler! nColumn und/oder nRow liegen außerhalb der zulässigen Grenzen 1.. nSunPrt_MaxColumnFcd bzw. 1.. nSunPrt_MaxRowFcd. Siehe Liste der Fassadenelemente.
02: Fehler: Der Gruppenindex ist 0, gleichzeitig ist jedoch ein anderer Eintrag des Fassadenelementes ungleich Null. Nur wenn alle Einträge eines Fassadenelementes Null sind, wird es als gültiges, gewollt ausgelassenes Fassadenelement angesehen, ansonsten jedoch als Fehleingabe interpretiert. **Info**: Gruppeneinträge kleiner Null werden intern auf Null begrenzt!
03: Fehler: Die X-Komponente des ersten Eckpunktes (Corner1) ist kleiner Null.
04: Fehler: Die Y-Komponente des ersten Eckpunktes (Corner1) ist kleiner Null.
05: Fehler: Die Fensterbreite ist kleiner oder gleich Null.
06: Fehler: Die Fensterhöhe ist kleiner oder gleich Null.

/ Ein-/Ausgänge

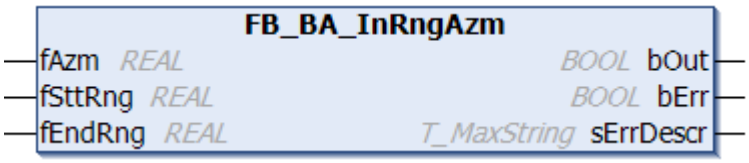
```
VAR_IN_OUT
  aFcdElem : ARRAY[1..Param.nSunPrt_MaxColumnFcd, 1..Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;
END_VAR
```

Name	Typ	Beschreibung
aFcdElem	ARRAY OF ST_BA_FcdElem [▶ 278]	Liste von Fassadenelementen [▶ 346]

Voraussetzungen

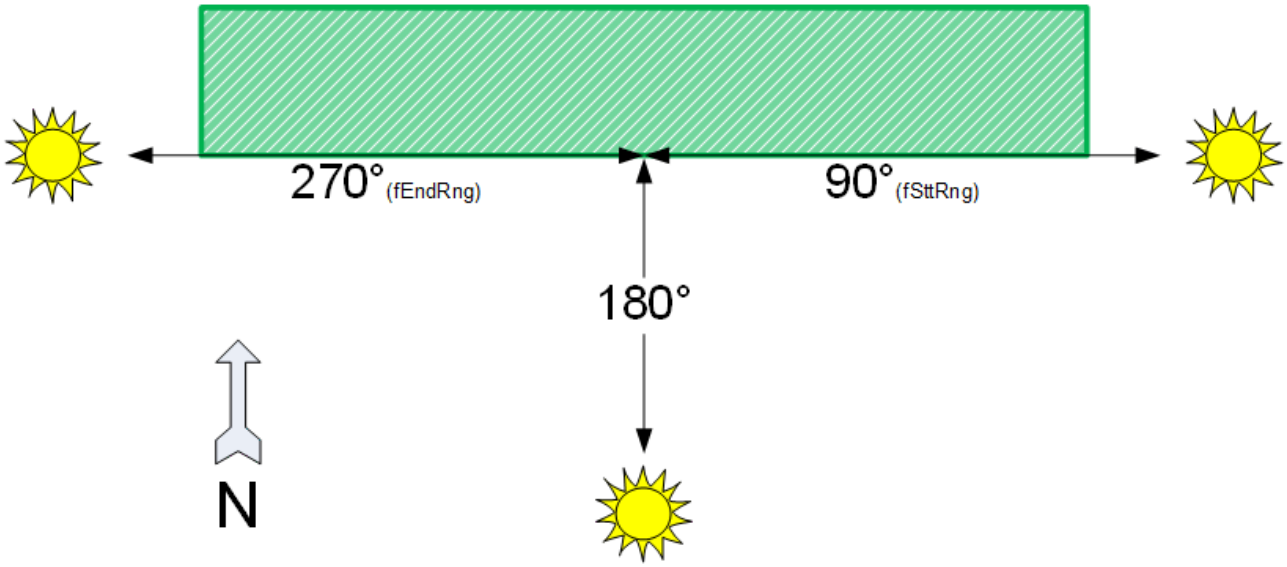
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.11 FB_BA_InRngAzm

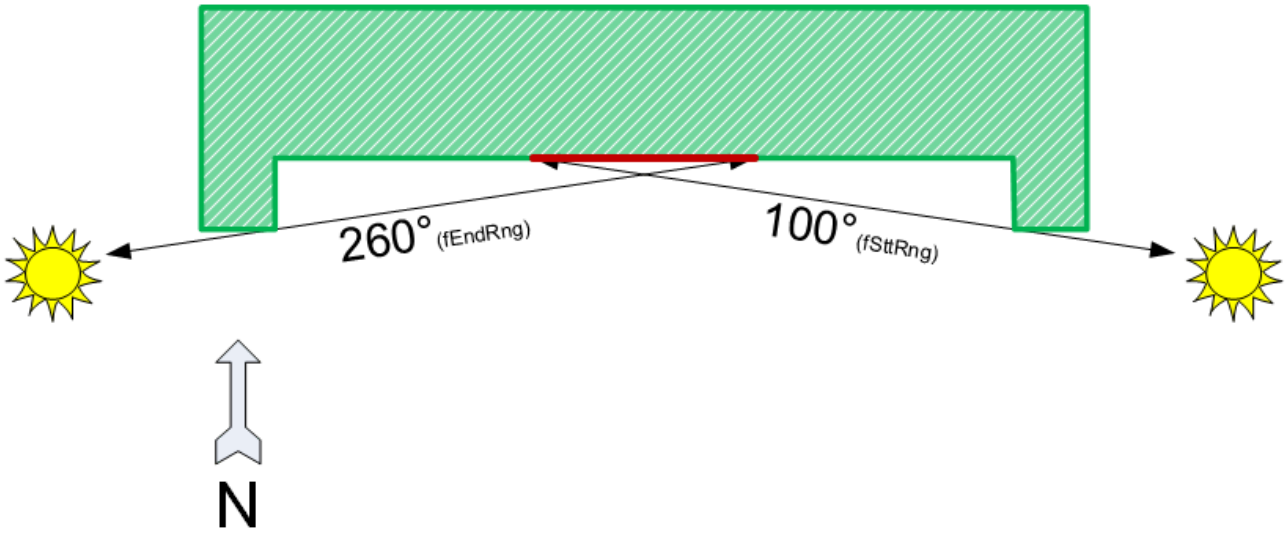


Der Funktionsbaustein FB_BA_InRngAzm prüft, ob der aktuelle Azimutwinkel (horizontaler Sonnenstand) innerhalb der eingetragenen Grenzen liegt. Wie in der Übersicht [▶ 339] erkennbar, gibt der Baustein eine zusätzliche Bewertung, ob der Sonnenschutz einer Fenstergruppe aktiviert werden soll. Daher gelten die Betrachtungen im weiteren Text immer für eine Fenstergruppe.

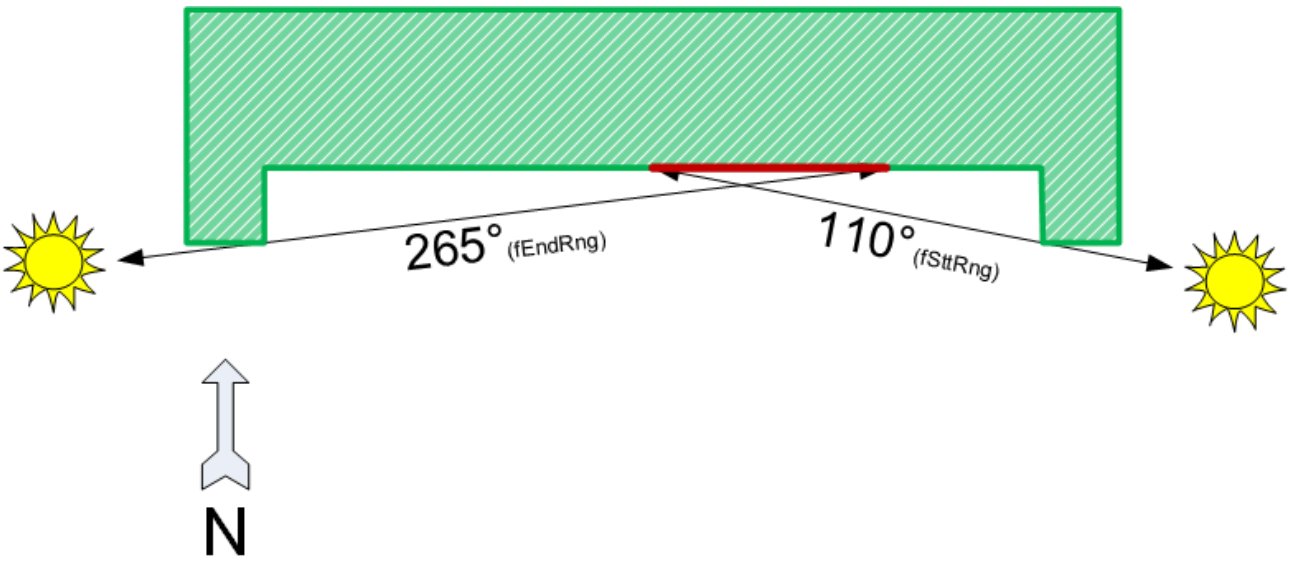
Eine glatte Fassade wird von der Sonne immer in einem Azimutwinkel von *Fassadenausrichtung-90°... Fassadenausrichtung+90°* bestrahlt.



Hat die Fassade jedoch seitliche Vorsprünge, so wird dieser Bereich eingeschränkt. Diese Einschränkung lässt sich mit Hilfe dieses Bausteines überprüfen. Dabei spielt aber auch die Lage der Fenstergruppe auf der Fassade eine Rolle. Liegt sie zentral, so ergibt sich folgende Situation (Die Werte sind dabei nur beispielhaft):



Für eine Gruppe am Rand andern sich die Werte:



Der Anfang des Bereiches *fSttRng* darf dabei größer sein als das Ende *fEndRng*, es wird dann über 0° hinaus betrachtet:

Beispiel

fAzm	10.0°
fSttRng	280.0°
fEndRng	20.0°
bOut	TRUE

Der betrachtete Bereich darf jedoch nicht größer als 180° oder gleich 0° sein, dieses wäre unrealistisch. Derartige Eingaben ergeben einen Fehler am Ausgang *bErr* - der Prüfausgang *bOut* wird dabei zusätzlich auf FALSE gesetzt.

Eingänge

```
VAR_INPUT
  fAzm     : REAL;
  fSttRng  : REAL;
  fEndRng  : REAL;
END_VAR
```

Name	Typ	Beschreibung
fAzm	REAL	Aktueller Azimutwinkel
fSttRng	REAL	Bereichsanfang [°]
fEndRng	REAL	Bereichsende [°].

Ausgänge

```
VAR_OUTPUT
  bOut       : BOOL;
  bErr       : BOOL;
  sErrDescr  : T_MAXSTRING;
END_VAR
```

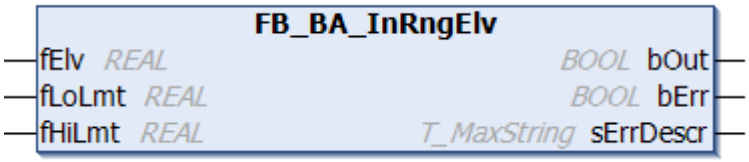
Name	Typ	Beschreibung
bOut	BOOL	Das Fassadenelement liegt in der Sonne, wenn der Ausgang TRUE ist.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: *fSttRng* bzw. *fEndRng* kleiner 0° oder größer 360°.
02: Fehler: Die Differenz von *fSttRng* und *fEndRng* ist größer als 180°. Für die Betrachtung des Sonneneinfalles auf eine Fassade ist dieser Bereich zu groß gewählt.

Voraussetzungen

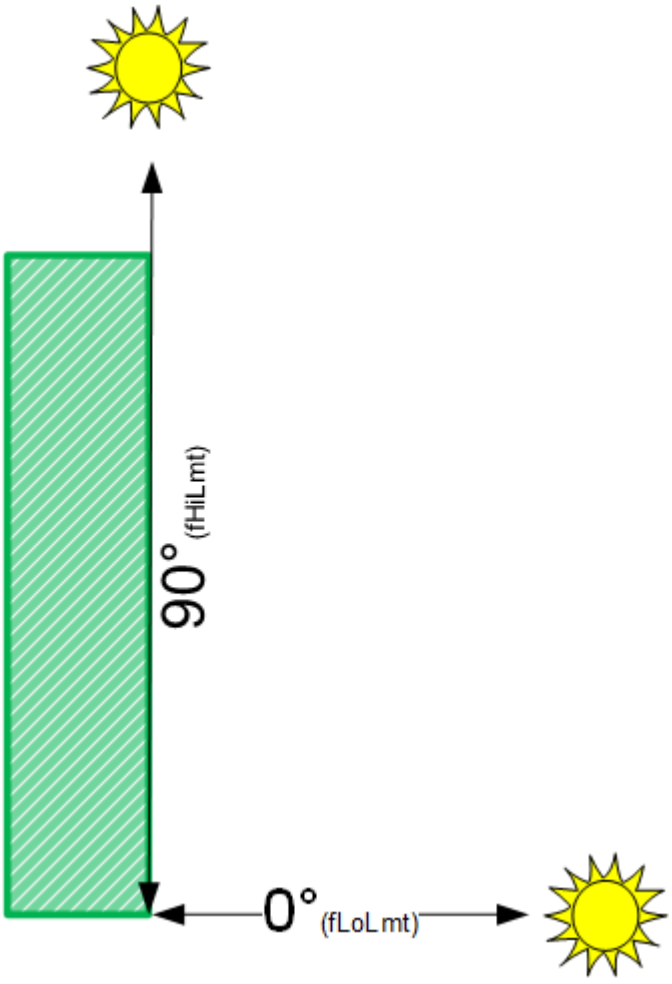
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.12 FB_BA_InRngElv

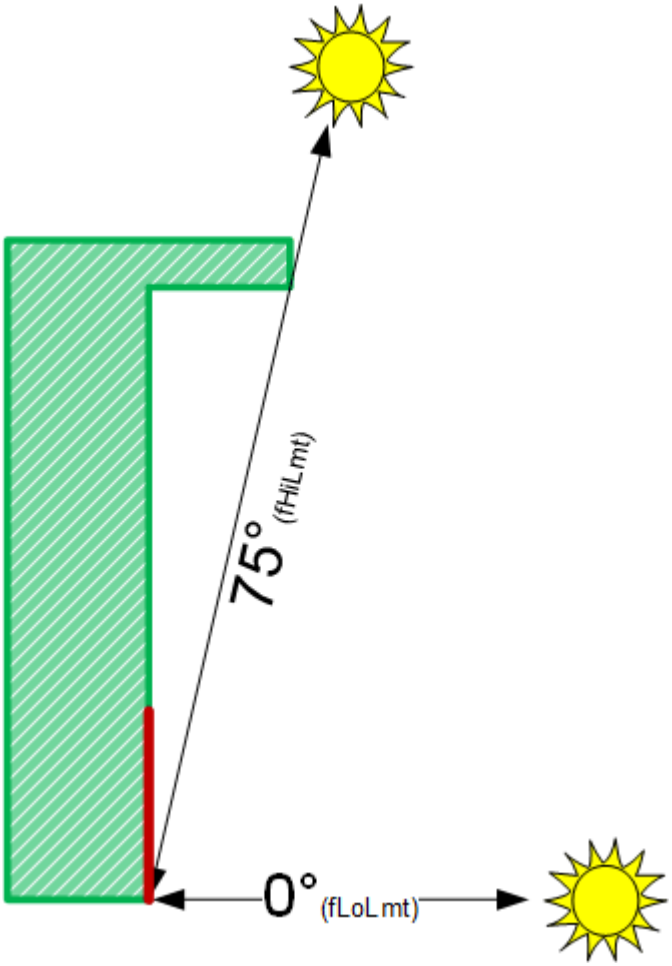


Der Funktionsbaustein FB_BA_InRngElv prüft, ob der aktuelle Elevationswinkel (vertikaler Sonnenstand) innerhalb der eingetragenen Grenzen liegt. Wie in der Übersicht [▶ 339] erkennbar, gibt der Baustein eine zusätzliche Bewertung, ob der Sonnenschutz einer Fenstergruppe aktiviert werden soll. Daher gelten die Betrachtungen im weiteren Text immer für eine Fenstergruppe.

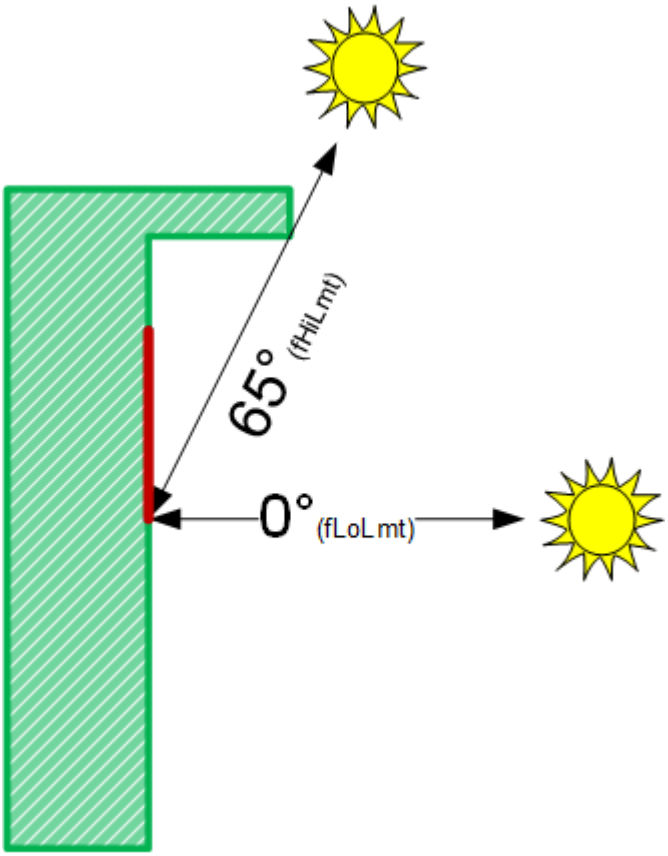
Eine normale senkrechte Fassade wird von der Sonne immer in einem Elevationswinkel von 0° bis maximal 90° bestrahlt.



Hat die Fassade jedoch Vorsprünge, so wird dieser Bereich eingeschränkt. Diese Einschränkung lässt sich mit Hilfe dieses Bausteines überprüfen. Dabei spielt aber auch die Lage der Fenstergruppe auf der Fassade eine Rolle. Liegt sie im unteren Bereich, so ergibt sich folgende Situation (Die Werte sind dabei nur beispielhaft):



Für eine Gruppe unterhalb des Vorsprunges ändern sich die Werte:



Die untere Betrachtungsgrenze, *fLoLmt*, darf dabei nicht größer oder gleich der oberen, *fHiLmt*, sein. Derartige Eingaben ergeben einen Fehler am Ausgang *bErr* - der Prüfausgang *bOut* wird dabei zusätzlich auf FALSE gesetzt.

Eingänge

```
VAR_INPUT
  fElv    : REAL;
  fLoLmt  : REAL;
  fHiLmt  : REAL;
END_VAR
```

Name	Typ	Beschreibung
fElv	REAL	Aktueller Elevationswinkel
fLoLmt	REAL	Unterer Grenzwert [°]
fHiLmt	REAL	Oberer Grenzwert [°]

Ausgänge

```
VAR_OUTPUT
  bOut       : BOOL;
  bErr       : BOOL;
  sErrDescr  : T_MAXSTRING;
END_VAR
```

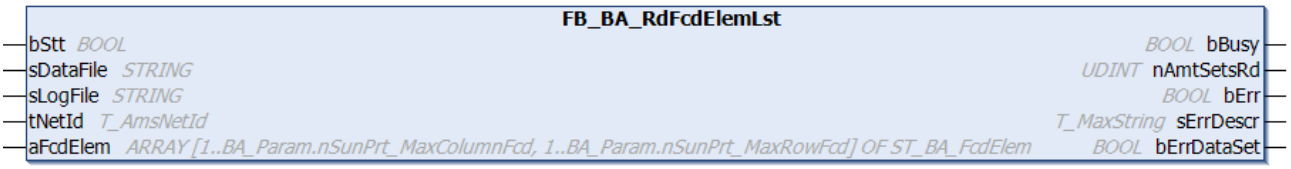
Name	Typ	Beschreibung
bOut	BOOL	Das Fassadenelement liegt in der Sonne, wenn der Ausgang TRUE ist.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: *fHiLmt* kleiner oder gleich *fLoLmt*.
02: Fehler: *fLoLmt* ist kleiner 0° oder *fHiLmt* ist größer 90°.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.13 FB_BA_RdFcdElemLst



Mit Hilfe des Funktionsbausteines FB_BA_RdFcdElemLst lassen sich Daten für Fassadenelemente (Fenster) aus einer vordefinierten Excel-Tabelle im csv-Format in die Liste der Fassadenelemente [▶ 346] importieren. Zusätzlich werden die importierten Daten auf Plausibilität überprüft und Fehler in eine Log-Datei geschrieben.

Das folgende Beispiel zeigt die Excel-Tabelle mit den Einträgen der Fensterelemente.
Alle Textfelder sind frei beschreibbar. Wichtig sind die grün markierten Felder, jede Zeile dort kennzeichnet einen Datensatz.
Folgende Regeln sind zu beachten:

- Ein Datensatz muss immer mit einem '@' beginnen.
- Die Indizes *IndexColumn* und *IndexRow* müssen innerhalb der definierten Grenzen liegen (siehe Liste von Fassadenelementen [▶ 346]). Mit diesen Indizes wird direkt beschrieben, in welches Fassadenelement der Liste *aFcdElem* die Daten des Satzes hineingelegt werden.
- Fensterbreite und Fensterhöhe müssen größer Null sein
- Die Eck-Koordinaten P1x und P1y müssen größer oder gleich Null sein.
- Jedes Fensterelement muss einer Gruppe 1..255 zugeordnet sein.
- Die Gesamtgröße der Tabelle darf systembedingt 65534 Bytes nicht überschreiten.
- Die Tabelle muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein.

Es ist nicht notwendig, alle Fensterelemente zu beschreiben, die per Definition bzw. Deklaration möglich wären. Bevor die neue Liste eingelesen wird, löscht der Baustein die gesamte alte Liste im Programm. Alle Elemente, welche dann nicht durch Einträge in der Excel-Tabelle beschrieben werden, haben reine Null-Einträge und sind dadurch als nicht vorhanden gekennzeichnet und auch nicht auswertbar, da der Baustein zur Verschattungskorrektur, FB_BA_ShdCorr [▶ 375], Elemente mit Gruppeneintrag '0' nicht annimmt.

DE_FacadeElements.xls [Kompatibilitätsmodus]

	A	B	C	D	E	F	G	H	I	J
1	Nummer	Beschreibung		IndexColum	IndexRow	Fensterbreit	Fensterhoeh	P1x	P1y	Gruppe
2				(Achse)	(Stockwerk)	[m]	[m]	[m]	[m]	
3		Text								
4	1	Beschreibung	@	1	1	1,2	1,3	1,5	1	2
5	2	Beschreibung	@	0	1	1,2	1,3	2,7	1	2
6	3	Beschreibung	@	3	1	1,2	1,3	4,4	1	2
7	4	Beschreibung	@	4	1	1,2	1,3	6,1	1	2
8	5	Beschreibung	@	5	1	1,2	1,3	7,8	1	2
9	6	Beschreibung	@	6	1	1,2	1,3	9,5	1	2
10	7	Beschreibung	@	7	1	1,2	1,3	11,2	1	2
11	8	Beschreibung	@	8	1	1,2	1,3	12,9	1	2
12	9	Beschreibung	@	9	1	1,2	1,3	14,6	1	2
13	10	Beschreibung	@	10	1	1,2	1,3	16,3	1	2
14	11	Beschreibung	@	1	1	1,2	1,3	1,5	4	2
15	12	Beschreibung	@	0	1	1,2	1,3	2,7	4	2
16	13	Beschreibung	@	3	1	1,2	1,3	4,4	4	2
17	14	Beschreibung	@	4	1	1,2	1,3	6,1	4	2
18	15	Beschreibung	@	5	1	1,2	1,3	7,8	4	2
19	16	Beschreibung	@	6	1	1,2	1,3	9,5	4	2
20	17	Beschreibung	@	7	1	1,2	1,3	11,2	4	2
21	18	Beschreibung	@	8	1	1,2	1,3	12,9	4	2
22	19	Beschreibung	@	9	1	1,2	1,3	14,6	4	2
23	20	Beschreibung	@	10	1	1,2	1,3	16,3	4	2
24	21	Beschreibung	@	1	1	1,2	1,3	1,5	7	2
25	22	Beschreibung	@	0	1	1,2	1,3	2,7	7	2
26	23	Beschreibung	@	3	1	1,2	1,3	4,4	7	2
27	24	Beschreibung	@	4	1	1,2	1,3	6,1	7	2
28	25	Beschreibung	@	5	1	1,2	1,3	7,8	7	2
29	26	Beschreibung	@	6	1	1,2	1,3	9,5	7	2
30	27	Beschreibung	@	7	1	1,2	1,3	11,2	7	2
31	28	Beschreibung	@	8	1	1,2	1,3	12,9	7	2
32	29	Beschreibung	@	9	1	1,2	1,3	14,6	7	2
33	30	Beschreibung	@	10	1	1,2	1,3	16,3	7	2
34	31	Beschreibung	@	1	1	1,2	1,3	1,5	10	2
35	32	Beschreibung	@	0	1	1,2	1,3	2,7	10	2
36	33	Beschreibung	@	3	1	1,2	1,3	4,4	10	2
37	34	Beschreibung	@	4	1	1,2	1,3	6,1	10	2
38	35	Beschreibung	@	5	1	1,2	1,3	7,8	10	2
39	36	Beschreibung	@	6	1	1,2	1,3	9,5	10	2
40	37	Beschreibung	@	7	1	1,2	1,3	11,2	10	2
41	38	Beschreibung	@	8	1	1,2	1,3	12,9	10	2
42	39	Beschreibung	@	9	1	1,2	1,3	14,6	10	2
43	40	Beschreibung	@	10	1	1,2	1,3	16,3	10	2

Log-Datei

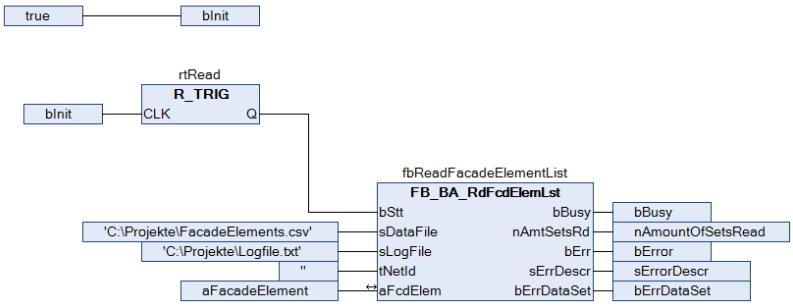
Bei jedem Start des Lesebausteines wird die Logdatei neu beschrieben und der alte Inhalt gelöscht. Ist die Log-Datei nicht vorhanden, so wird sie zunächst automatisch erstellt. Die Log-Datei enthält dann entweder eine OK Meldung oder eine Auflistung aller aufgetretenen Fehler. Fehler, welche mit dem Öffnen, Beschreiben oder Schließen der Logdatei selbst in Zusammenhang stehen, können nicht mitgeschrieben

werden. Daher ist immer auch der Ausgang *sErrDescr* des Lesebausteines zu beachten, der den letzten Fehlercode anzeigt. Da im Ablauf des Lesens die Log-Datei immer als Letztes geschlossen wird, ist ein entsprechender Hinweis im Fehlerfall gewährleistet.

Programmbeispiel

```
PROGRAM ReadFacadeElements
VAR
    bInit                       :   BOOL;
    rtRead                      :   R_TRIG;
    fbReadFacadeElementList     :   FB_BA_RdFcdElemLst;
    aFacadeElement              :   ARRAY [1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem;

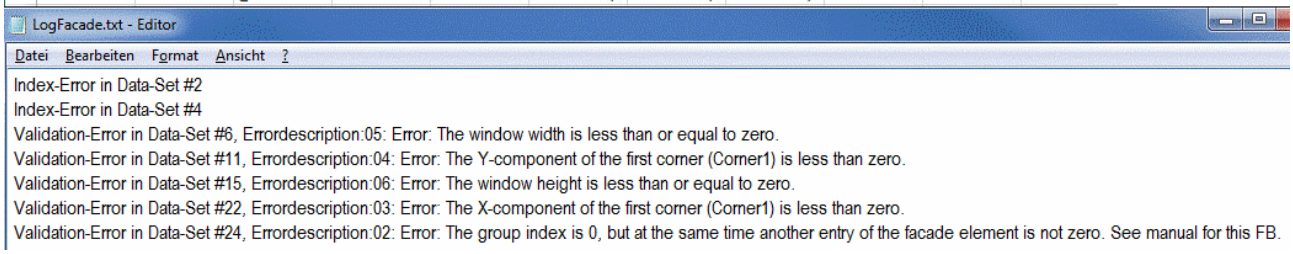
    bBusy                       :   BOOL;
    nAmountOfSetsRead           :   UDINT;
    bError                      :   BOOL;
    sErrorDescr                 :   T_MaxString;
    bErrDataSet                 :   BOOL;
END_VAR
```



In diesem Beispiel wird bei PLC-Start die Variable *bInit* zunächst auf TRUE gesetzt. Damit erhält der Eingang *bStt* am Baustein *fbReadFacadeElementList* einmalig eine steigende Flanke, welcher den Lesevorgang auslöst. Gelesen wird die Datei *FacadeElements.csv*, welche sich im Ordner *C:\Projekte* befindet. In demselben Ordner wird dann die Logdatei "Logfile.txt" hinterlegt. Ist diese Logdatei noch nicht vorhanden wird sie erstellt, andernfalls wird der bestehende Inhalt überschrieben. Das Lesen und Schreiben erfolgt auf demselben Rechner, auf dem sich auch die PLC befindet. Das wird durch den Eingang *tNetID* = '' (=lokal) definiert. Alle Daten werden in die im Programm deklarierte Liste *aFcdElem* geschrieben. Während gelesen und geschrieben wird ist der Ausgang *bBusy* auf TRUE. Der zuletzt auftretende Fehler in Bezug auf das Dateihandling wird an *sErrDescr* angezeigt, *bErr* steht dann auf TRUE. Wird hingegen ein Fehler im Datensatz erkannt, wird dies an *bErrDataSet* angezeigt und in der Log-Datei näher beschrieben. Die Anzahl der gefundenen und gelesenen Datenzeilen wird an *nAmtSetsRd* zur Kontrolle angezeigt.

In die folgende Excel Liste wurden die markierten Fehler "eingebaut". Dadurch ergibt sich das gezeigte Log-File:

	A	B	C	D	E	F	G	H	I	J	K
1	Nummer	Beschreibung		IndexColum	IndexRow	Fensterbreit	Fensterhoeh	P1x	P1y	Gruppe	
2				(Achse)	(Stockwerk)	[m]	[m]	[m]	[m]		
3		Text									
4	1	Beschreibun	@	1	1	1,2	1,3	1,5	1,4	2	
5	2	Beschreibun	@	0	1	1,2	1,3	2,7	1	2	
6	3	Beschreibun	@	3	1	1,2	1,3	4,4	1	2	
7	4	Beschreibun	@	4	-1	1,2	1,3	6,1	1	2	
8	5	Beschreibun	@	5	1	1,2	1,3	7,8	1	2	
9	6	Beschreibun	@	6	1	0	1,3	9,5	1	2	
10	7	Beschreibun	@	7	1	1,2	1,3	11,2	1	2	
11	8	Beschreibun	@	8	1	1,2	1,3	12,9	1	2	
12	9	Beschreibun	@	9	1	1,2	1,3	14,6	1	2	
13	10	Beschreibun	@	10	1	1,2	1,3	16,3	1	5	
14	11	Beschreibun	@	1	2	1,2	1,3	1	-1	5	
15	12	Beschreibun	@	2	2	1,2	1,3	2,7	3	5	
16	13	Beschreibun	@	3	2	1,2	1,3	4,4	4	5	
17	14	Beschreibun	@	4	2	1,2	1,3	4,4	4	5	
18	15	Beschreibun	@	5	2	1,2	0	7,8	4	5	
19	16	Beschreibun	@	6	2	1,2	1,3	9,5	4	5	
20	17	Beschreibun	@	7	2	1,2	1,3	11,2	4	5	
21	18	Beschreibun	@	8	2	1,2	1,3	12,9	4	5	
22	19	Beschreibun	@	9	2	1,2	1,3	14,6	4	3	
23	20	Beschreibun	@	10	2	1,2	1,3	16,3	4	3	
24											
25	31	Beschreibun	@	1	3	1,2	1,3	1	7	3	
26	32	Beschreibun	@	2	3	1,2	1,3	-1	6	3	
27	33	Beschreibun	@	3	3	1,2	1,3	4,4	7	3	
28	34	Beschreibun	@	4	3	1,2	1,3	6,1	7	0	
29	35	Beschreibun	@	5	3	1,2	1,3	7,8	7	3	
30	36	Beschreibun	@	6	3	1,2	1,3	9,5	7	3	
31	37	Beschreibu	@	7	3	1,2	1,3	11,2	7	3	
32	38	Beschreibun	@	8	3	1,2	1,3	12,9	7	7	
33	39	Beschreibun	@	9	3	1,2	1,3	14,6	7	7	
34	40	Beschreibun	@	10	3	1,2	1,3	16,3	7	7	



Der erste Fehler ist im Datensatz 2 und ein Index-Fehler, da "0" nicht erlaubt ist.
Der nächste Fehler in Datensatz 6 wurde nach Validierung der Daten mit dem intern genutzten Baustein FB_BA_ShdObjEntry [▶ 378] gefunden und näher mit einer Fehlerbeschreibung versehen. Der dritte und der vierte Fehler trat ebenfalls nach der internen Validierung auf.

Wichtig hierbei ist, dass sich die Datensatz-Nummern (hier 22 und 24) nicht nach den eingetragenen Nummern in der Liste, sondern nach den tatsächlichen laufenden Nummern richten: eingelesen wurden hier lediglich 30 Datensätze.

Eingänge

```
VAR_INPUT
  bStt       : BOOL;
  sDataFile  : STRING;
  sLogFile   : STRING;
  tNetId     : T_AmsNetId;
END_VAR
```

Name	Typ	Beschreibung
bStt	BOOL	Eine TRUE-Flanke an diesem Eingang startet den Lesevorgang.
sDataFile	STRING	Enthält den Pfad- und Dateinamen der zu öffnenden Daten-Datei. Diese muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein. Wird die Datei mit einem einfachen Text-Editor geöffnet, so müssen die Werte durch Semikola getrennt dargestellt sein. Beispiel für einen Eintrag: *sDataFile:= 'C:\Projekte\FacadeElements.csv'*
sLogFile	STRING	dto. Log-File für die auflaufenden Fehler. Diese Datei wird bei jeder neuen Bausteinaktivierung überschrieben, so dass nur aktuelle Fehler enthalten sind.
tNetId	T_AmsNetID	Hier kann ein String mit der AMS Net Id des TwinCAT-Rechners angegeben werden, auf dem die Dateien geschrieben/gelesen werden sollen. Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

Die Speicherung der Daten kann nur auf dem Steuerungsrechner selbst und den Rechnern erfolgen, die mit dem Steuerungsrechner per ADS verbunden sind. Verweise auf lokale Festplatten dieser Rechner sind möglich, nicht jedoch auf verbundene Netzwerkfestplatten.

/ Ein-/Ausgänge

```
VAR_IN_OUT
  aFcdElem : ARRAY[1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem
;
END_VAR
```

Name	Typ	Beschreibung
aFcdElem	ARRAY OF ST_BA_FcdElem [▶ 278]	Liste von Fassadenelementen [▶ 346]

Ausgänge

```
VAR_OUTPUT
  bBusy        : BOOL;
  nAmtSetsRd   : UDINT;
  bErr         : BOOL;
  sErrDescr    : T_MAXSTRING;
  bErrDataSet  : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang steht auf TRUE, solange Elemente aus der Datei ausgelesen werden.
aAmtSetsRd	UDINT	Anzahl der gelesenen Datensätze
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

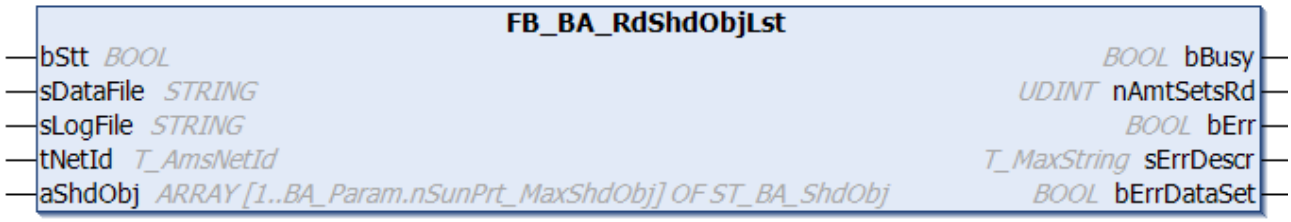
Fehlerbeschreibung
01: Dateihandling-Fehler: Öffnen der Logdatei - Es wird die ADS-Fehlernummer angegeben.
02: Dateihandling-Fehler: Öffnen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
03: Dateihandling-Fehler: Lesen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
04: Fehler: Beim Lesen der Datendatei wurde festgestellt, dass die Datei ist zu groß ist (Anzahl Bytes größer als nMaxDataFileSize)
05: Dateihandling-Fehler: Schreiben in die Logdatei - Es wird die ADS-Fehlernummer angegeben.
06: Dateihandling-Fehler: Schließen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
07: Dateihandling-Fehler: Schreiben in die Logdatei (ok-Meldung, falls keine Fehler erkannt wurden) - Es wird die ADS-Fehlernummer angegeben.
08: Dateihandling-Fehler: Schließen der Logdatei - Es wird die ADS-Fehlernummer angegeben.

Name	Typ	Beschreibung
bErrDataSet	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die gelesenen Datensätze fehlerhaft sind. Nähere Einzelheiten werden in die Log-Datei geschrieben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.14 FB_BA_RdShdObjLst



Mit Hilfe des Funktionsbausteines FB_BA_RdShdObjLst lassen sich Daten für Verschattungsobjekte aus einer vordefinierten Excel-Tabelle im csv-Format in die Liste der Verschattungsobjekte [▶ 346] importieren. Zusätzlich werden die importierten Daten auf Plausibilität überprüft und Fehler in eine Log-Datei geschrieben.

Das folgende Beispiel zeigt die Excel-Tabelle mit den Einträgen der Fensterelemente.
Alle Textfelder sind frei beschreibbar. Wichtig sind die grün markierten Felder, jede Zeile dort kennzeichnet einen Datensatz. Je nachdem ob es sich um den Typ Viereck oder Kugel handelt, haben die Spalten G bis J eine unterschiedliche Bedeutung. Die Spalten K bis M sind bei Kugeln freizulassen. Bezüglich der Vierecks-Koordinaten werden nur die relevanten Daten eingegeben und die restlichen intern berechnet (siehe FB_BA_ShdObjEntry [▶ 378]).
Folgende Regeln sind zu beachten:

- Ein Datensatz muss immer mit einem '@' beginnen.
- Die Monatseinträge dürfen nicht 0 und größer 12 sein, alle anderen Kombinationen sind möglich.
  Beispiele:
  Beginn=1, Ende=1: Verschattung im Januar.
  Beginn=1, Ende=5: Verschattung von Anfang Januar bis Ende Mai.
  Beginn=11, Ende=5: Verschattung von Anfang November bis Ende Mai (des folgenden Jahres).
- Die z-Koordinaten P1z und P3z bzw. Mz müssen größer Null sein.
- Der Radius muss größer Null sein.
- Die Gesamtgröße der Tabelle darf systembedingt 65534 Bytes nicht überschreiten.
- Die Tabelle muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein.

Es ist nicht notwendig, alle Verschattungsobjekte, welche pro Fassade möglich sind, zu beschreiben. Nur die, die in der Liste aufgeführt werden, kommen letztendlich zum Tragen.

DE_ShadingObjects.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Nummer	Beschreibung		Typ	Beginn	Ende	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Viereck	(Monat)	(Monat)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Kugel									
4		Text											
5	1	Beschreibung	@	0	1	2	-94,75	0	36,06	11	-70,71	11	68,59
6	2	Beschreibung	@	0	1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Beschreibung	@	0	1	2	62,23	0	0	14,47	62,23	14,47	8
8	4	Beschreibung	@	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Beschreibung	@	0	1	2	46	0	13	14,47	46	14,47	38,89
10	6	Beschreibung	@	0	1	2	0	0	14	9	35	9	14
11	7	Beschreibung	@	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Beschreibung	@	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Beschreibung	@	0	1	2	27,8	0	14	9,8	35	9,8	14
14													
15	10	Beschreibung	@	1	1	2	27	15	40	6			
16	11	Beschreibung	@	1	1	2	38	15	36	6			
17	12	Beschreibung	@	1	1	2	-14	4	4	1,5			
18	13	Beschreibung	@	1	1	2	-6,5	6	6	3,2			
19	14	Beschreibung	@	1	1	2	-7	9	6	1,2			
20	15	Beschreibung	@	1	1	2	-1	6	8	3,2			
21	16	Beschreibung	@	1	1	2	-1	9	8	1,2			

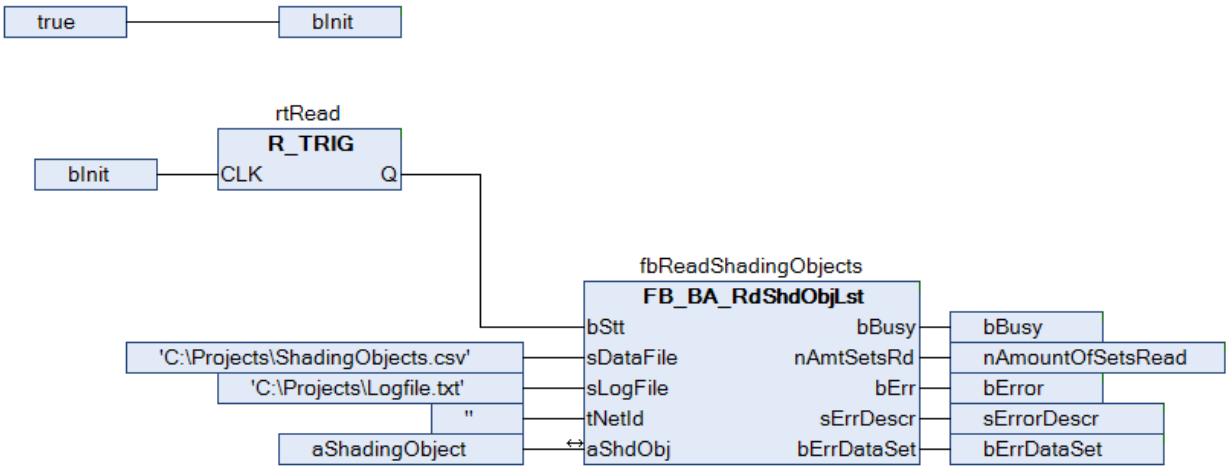
Log-Datei

Bei jedem Start des Lesebausteines wird die Logdatei neu beschrieben und der alte Inhalt gelöscht. Ist die Log-Datei nicht vorhanden, so wird sie zunächst automatisch erstellt. Die Log-Datei enthält dann entweder eine OK Meldung oder eine Auflistung aller aufgetretenen Fehler. Fehler, welche mit dem Öffnen, Beschreiben oder Schließen der Logdatei selbst in Zusammenhang stehen, können nicht mitgeschrieben werden. Daher ist immer auch der Ausgang *sErrDescr* des Lesebausteines zu beachten, der den letzten Fehlercode anzeigt. Da im Ablauf des Lesens die Log-Datei immer als Letztes geschlossen wird, ist ein entsprechender Hinweis im Fehlerfall gewährleistet.

Programmbeispiel

```
PROGRAM ReadShadingObjects
VAR
    bInit                       :   BOOL;
    rtRead                      :   R_TRIG;
    fbReadShadingObjects        :   FB_BA_RdShdObjLst;
    aShadingObject              :   ARRAY [1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;

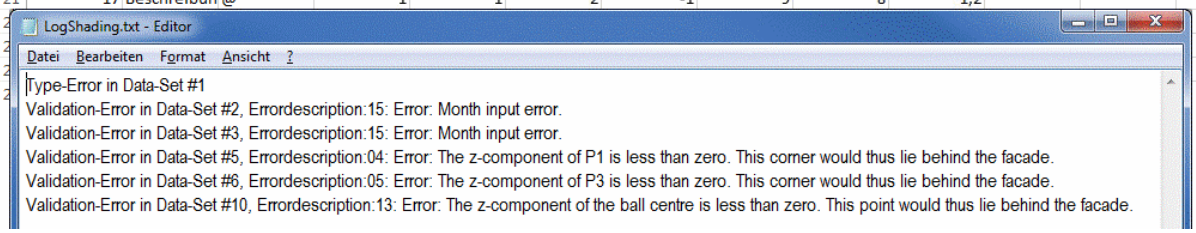
    bBusy                       :   BOOL;
    nAmountOfSetsRead           :   UDINT;
    bError                      :   BOOL;
    sErrorDescr                 :   T_MaxString;
    bErrDataSet                 :   BOOL;
END_VAR
```



In diesem Beispiel wird bei PLC-Start die Variable *bInit* zunächst auf TRUE gesetzt. Damit erhält der Eingang *bStt* am Baustein *fbReadShadingObjects* einmalig eine steigende Flanke, welche den Lesevorgang auslöst. Gelesen wird die Datei *ShadingObjects.csv*, welche sich im Ordner *C:\Projekte* befindet. In demselben Ordner wird dann die Logdatei *Logfile.txt* hinterlegt. Ist diese Logdatei noch nicht vorhanden wird sie erstellt, andernfalls wird der bestehende Inhalt überschrieben. Das Lesen und Schreiben erfolgt auf demselben Rechner, auf dem sich auch die PLC befindet. Das wird durch den Eingang *tNetID* = '' (=lokal) definiert. Alle Daten werden in die im Programm deklarierte Liste *aShdObj* geschrieben. Während gelesen und geschrieben wird ist der Ausgang *bBusy* auf TRUE. Der zuletzt auftretende Fehler in Bezug auf das Dateihandling wird an *sErrDescr* angezeigt, *bErr* steht dann auf TRUE. Wird hingegen ein Fehler im Datensatz erkannt, wird dies an *bErrDataSet* angezeigt und in der Log-Datei näher beschrieben. Die Anzahl der gefundenen und gelesenen Datenzeilen wird an *nAmtSetsRd* zur Kontrolle angezeigt.

In die folgende Excel Liste wurden die markierten Fehler eingebaut. Dadurch ergibt sich das gezeigte Log-File:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Nummer	Beschreibung		Typ	Beginn	Ende	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Viereck	(Monat)	(Monat)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Kugel									
4		Text											
5	1	Beschreibun	@	2	1	2	-94,75	-4	36,06	11	-70,71	11	68,59
6	2	Beschreibun	@	0	-1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Beschreibun	@	0	1	13	62,23	0	0	14,47	62,23	14,47	8
8	4	Beschreibun	@	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Beschreibun	@	0	1	2	46	0	-13	14,47	46	14,47	38,89
10	6	Beschreibun	@	0	1	2	0	0	14	9	35	9	-14
11	7	Beschreibun	@	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Beschreibun	@	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Beschreibun	@	0	1	2	27,8	0	14	9,8	35	9,8	14
14													
15	11	Beschreibun	@	1	1	2	27	15	-40	6			
16	12	Beschreibun	@	1	1	2	38	15	36	6			
17	13	Beschreibun	@	1	1	2	-14	4	4	1,5			
18	14	Beschreibun	@	1	1	2	-6,5	6	6	3,2			
19	15	Beschreibun	@	1	1	2	-7	9	6	1,2			
20	16	Beschreibun	@	1	1	2	-1	6	8	3,2			
21	17	Beschreibun	@	1	1	2	-1	9	8	1,2			



Der erste Fehler ist im Datensatz 3 und ein Typ-Fehler, da "2" nicht definiert ist.
Der nächste Fehler in Datensatz 6 wurde nach Validierung der Daten mit dem intern genutzten Baustein FB_BA_ShdObjEntry [▶ 378] gefunden und daher näher mit einer Fehlerbeschreibung versehen. Der dritte Fehler trat ebenfalls nach der internen Validierung auf.

Zu beachten ist, dass die fehlenden Koordinaten wie folgt ausgefüllt werden:

fP2x = fP1x; fP2z = fP1z; fP4x = fP3x; fP4y = fP1y; fP4z = fP3z;

Fehlermeldungen, die die Gleichheit zweier Punkte bemängeln, sind nach diesen Gleichungen zu untersuchen.

Wichtig hierbei ist, dass sich die Datensatz-Nummer (hier 11) nicht nach der eingetragenen Nummer in der Liste, sondern nach der tatsächlichen laufenden Nummer richtet: eingelesen wurden hier lediglich 16 Datensätze.

Eingänge

```
VAR_INPUT
  bStt        : BOOL;
  sDataFile   : STRING;
  sLogFile    : STRING;
  tNetId      : T_AmsNetId;
END_VAR
```

Name	Typ	Beschreibung
bStt	BOOL	Eine TRUE-Flanke an diesem Eingang startet den Lesevorgang.
sDataFile	STRING	Enthält den Pfad- und Dateinamen der zu öffnenden Daten-Datei. Diese muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein. Wird die Datei mit einem einfachen Text-Editor geöffnet, so müssen die Werte durch Semikola getrennt dargestellt sein. Beispiel für einen Eintrag: *sDataFile:= 'C:\Projekte\FacadeElements.csv'*
sLogFile	STRING	dto. Log-File für die auflaufenden Fehler. Diese Datei wird bei jeder neuen Bausteinaktivierung überschrieben, so dass nur aktuelle Fehler enthalten sind.
tNetId	T_AmsNetID	Hier kann ein String mit der AMS Net Id des TwinCAT-Rechners angegeben werden, auf dem die Dateien geschrieben/gelesen werden sollen. Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

Die Speicherung der Daten kann nur auf dem Steuerungsrechner selbst und den Rechnern erfolgen, die mit dem Steuerungsrechner per ADS verbunden sind. Verweise auf lokale Festplatten dieser Rechner sind möglich, nicht jedoch auf verbundene Netzwerkfestplatten.

/ Ein-/Ausgänge

```
VAR_IN_OUT
  aShdObj: ARRAY[1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Name	Typ	Beschreibung
aShdObj	ARRAY OF ST_BA_ShdObj [▶ 279]	Liste der Verschattungsobjekte [▶ 346]

Ausgänge

```
VAR_OUTPUT
  bBusy         : BOOL;
  nAmtSetsRd    : UDINT;
  bErr          : BOOL;
  sErrDescr     : T_MAXSTRING;
  bErrDataSet   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Dieser Ausgang steht auf TRUE, solange Elemente aus der Datei ausgelesen werden.
aAmtSetsRd	UDINT	Anzahl der gelesenen Datensätze
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

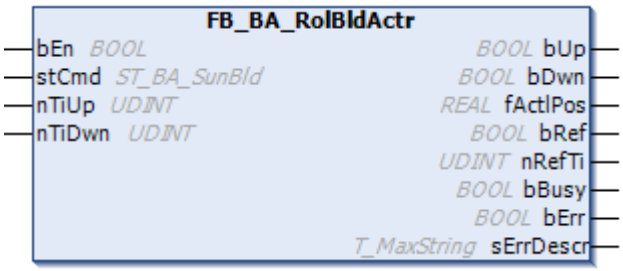
Fehlerbeschreibung
01: Dateihandling-Fehler: Öffnen der Logdatei - Es wird die ADS-Fehlernummer angegeben.
02: Dateihandling-Fehler: Öffnen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
03: Dateihandling-Fehler: Lesen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
04: Fehler: Beim Lesen der Datendatei wurde festgestellt, dass die Datei ist zu groß ist (Anzahl Bytes größer als nMaxDataFileSize)
05: Dateihandling-Fehler: Schreiben in die Logdatei - Es wird die ADS-Fehlernummer angegeben.
06: Dateihandling-Fehler: Schließen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
07: Dateihandling-Fehler: Schreiben in die Logdatei (ok-Meldung, falls keine Fehler erkannt wurden) - Es wird die ADS-Fehlernummer angegeben.
08: Dateihandling-Fehler: Schließen der Logdatei - Es wird die ADS-Fehlernummer angegeben.

Name	Typ	Beschreibung
bErrDataSet	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die gelesenen Datensätze fehlerhaft sind. Nähere Einzelheiten werden in die Log-Datei geschrieben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.15 FB_BA_RolBldActr



Der Funktionsbaustein FB_BA_RolBldActr dient zur Positionierung eines Rollladens über zwei Ausgänge: Hoch- und Herunterfahren. Über das Positioniertelegramm stSunBld [▶ 279] kann der Rollladen auf eine beliebige Position gesteuert werden. Darüber hinaus beinhaltet das Positioniertelegramm stSunBld [▶ 279] auch Handbefehle, mit denen der Rollladen individuell auf bestimmte Positionen bewegt werden kann. Diese Handbefehle werden von dem Baustein FB_BA_SunBldSwi [▶ 398] angesteuert.

Die aktuelle Höhenposition wird dabei nicht durch einen zusätzlichen Encoder eingelesen, sondern intern durch die Laufzeit des Rollladens ermittelt.
Durch die beiden unterschiedlichen Laufzeitparameter *nTiUp* (Laufzeit Rolllade hoch [ms]) und *nTiDwn* (Laufzeit Rollladen herunter [ms]) wird den unterschiedlichen Fahrcharakteristiken Rechnung getragen.

Der Baustein steuert den Rollladen grundsätzlich über die Informationen aus dem Positioniertelegramm stSunBld [▶ 279]. Ist der Automatikmodus aktiv (*bManMod* = FALSE), so wird immer die aktuelle Position angefahren, wobei Änderungen sofort berücksichtigt werden. Im Handbetrieb (*bManMod* = TRUE) steuern die Befehle *bManUp* und *bManDwn* den Rollladen.

Referenzieren

Ein sicheres Referenzieren ist gegeben, wenn der Rollladen länger als seine komplette Hochlaufzeit nach oben hin angesteuert wird. Die Position ist dann auf jeden Fall "0". Da eine Rollladenpositionierung ohne Encoder naturgemäß immer fehlerbehaftet ist, ist es wichtig möglichst oft automatisch zu referenzieren: jedes Mal, wenn die Position "0" angefahren werden soll, fährt der Rollladen zunächst ganz normal mit kontinuierlicher Positionsberechnung nach oben. Erreicht sie den errechneten Positionswert 0%, so wird der Ausgang *bUp* noch einmal für die weiterhin gehalten, und zwar noch einmal für die komplette Hochlaufzeit + 5s.

Der Referenziervorgang kann durch einen Handbefehl "herunter" oder durch eine neue Positionierung unterbrochen werden.

Beim Neustart der Anlage referenziert der Baustein in jedem Fall.

Eingänge

```
VAR_INPUT
  bEn           : BOOL;
  stSunBld      : ST_BA_Sunblind;
  nTiUp         : UDINT;
  nTiDwn        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabeeingang für den Baustein. Solange dieser Eingang auf TRUE steht, nimmt der Aktorbaustein Befehle entgegen und arbeitet, wie oben beschrieben. Ein FALSE-Signal an diesen Eingang setzt die Steuerausgänge *bUp* und *bDwn* zurück und der Funktionsbaustein verharrt in einem Ruhezustand.
stSunBld	ST_BA_SunBld [▶ 279]	Positioniertelegramm
nTiUp	UDINT	Komplette Hochfahrzeit [ms]
nTiDwn	UDINT	Komplette Zeit zum Herunterfahren [ms]

Ausgänge

```
VAR_OUTPUT
  bUp           : BOOL;
  bDwn          : BOOL;
  fActlPos      : REAL;
  bRef          : BOOL;
  nRefTi        : UDINT;
  bBusy         : BOOL;
  bErr          : BOOL;
  sErrDescr     : T_MAXSTRING
END_VAR
```

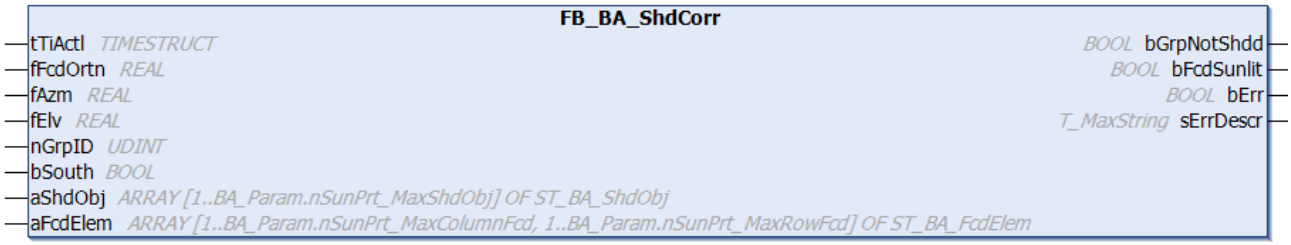
Name	Typ	Beschreibung
bUp	BOOL	Steuerausgang Rollladen hoch
bDwn	BOOL	Steuerausgang Rollladen herunter
fActlPos	READ	Aktuelle Position in Prozent.
bRef	BOOL	Die Rollladen befindet sich in der Referenzierung, d.h. für die die komplette Hochlaufzeit + 5s wird der Ausgang *bUp* gesetzt. Nur ein Handbefehl "herunter" kann den Rollladen in Gegenrichtung bewegen und diesen Modus beenden.
nRefTi	UDINT	Referenzier-Countdown-Anzeige [s]
bBusy	BOOL	Ein Positionier- oder Referenziervorgang findet statt.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Die Gesamt-Hoch- oder Herunterfahrzeit (nTiUp / nTiDwn) ist Null.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.16 FB_BA_ShdCorr



Der Funktionsbaustein FB_BA_ShdCorr dient zur Verschattungsbeurteilung einer Fenstergruppe auf einer Fassade.

Der Baustein *FB_BA_ShdCorr* berechnet für eine Gruppe von Fenstern, ob sich diese im Schatten von umliegenden Objekten befindet. Mit dem Ergebnis, welches am Ausgang *bGrpNotShdd* ausgegeben wird, kann beurteilt werden, ob ein Sonnenschutz für diese Fenstergruppe sinnvoll ist.
Dabei greift der Baustein auf zwei zu definierende Listen zu:

- Die Parameter, welche die Verschattungselemente beschreiben, die für die Fassade maßgeblich sind, auf der sich die Fenstergruppe befindet. Diese Liste der Verschattungsobjekte [▶ 346] wird als Eingangsvariable *aShdObj* an den Baustein angelegt, da die Informationen lediglich gelesen werden.
- Die Daten der Elemente (Fenster) der Fassade, in der sich die zu betrachtende Gruppe befindet. Auf diese Liste der Fassadenelemente [▶ 346] wird über die IN-OUT-Variable *aFcdElem* zugegriffen, da nicht nur die Koordinaten der Fenster gelesen werden, sondern von dem Baustein *FB_BA_ShdCorr* in diese Liste auch jeweils die Verschattungsinformation für jede Fensterecke hinterlegt wird. Diese Information kann so in weiteren Teilen des Applikationsprogrammes genutzt werden.

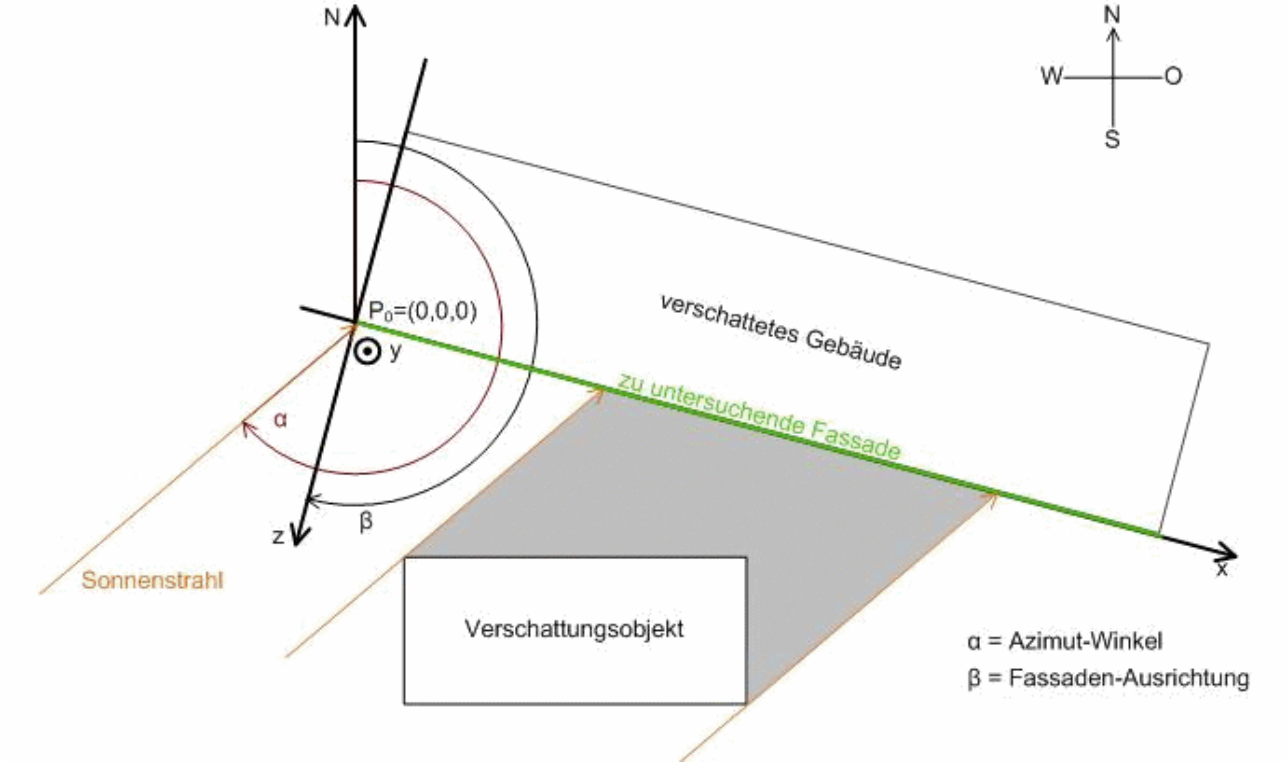
Anhand der Fassadenausrichtung (*fFcdOrtn*), der Sonnenrichtung (*fAzm*) und der Sonnenhöhe (*fElv*) kann dann für jeden Eckpunkt eines Fensters errechnet werden, ob sich dieser im Bereich eines Schattens befindet. Eine Gruppe von Fenstern gilt dann als komplett verschattet, wenn alle Eckpunkte verschattet sind.

Dabei gilt auf der Nordhalbkugel für die Fassadenausrichtung (Blick aus dem Fenster):

Blickrichtung	Fassadenausrichtung
Nord	β=0°
Ost	β=90°
Süd	β=180°
West	β=270°

Der Baustein führt seine Berechnungen nur dann durch, wenn die Sonne tatsächlich auf die Fassade scheint. Wird die in der Einleitung vorgestellte Zeichnung betrachtet, so ist dies gegeben, wenn gilt:

Fassadenausrichtung < Azimutwinkel < Fassadenausrichtung + 180°

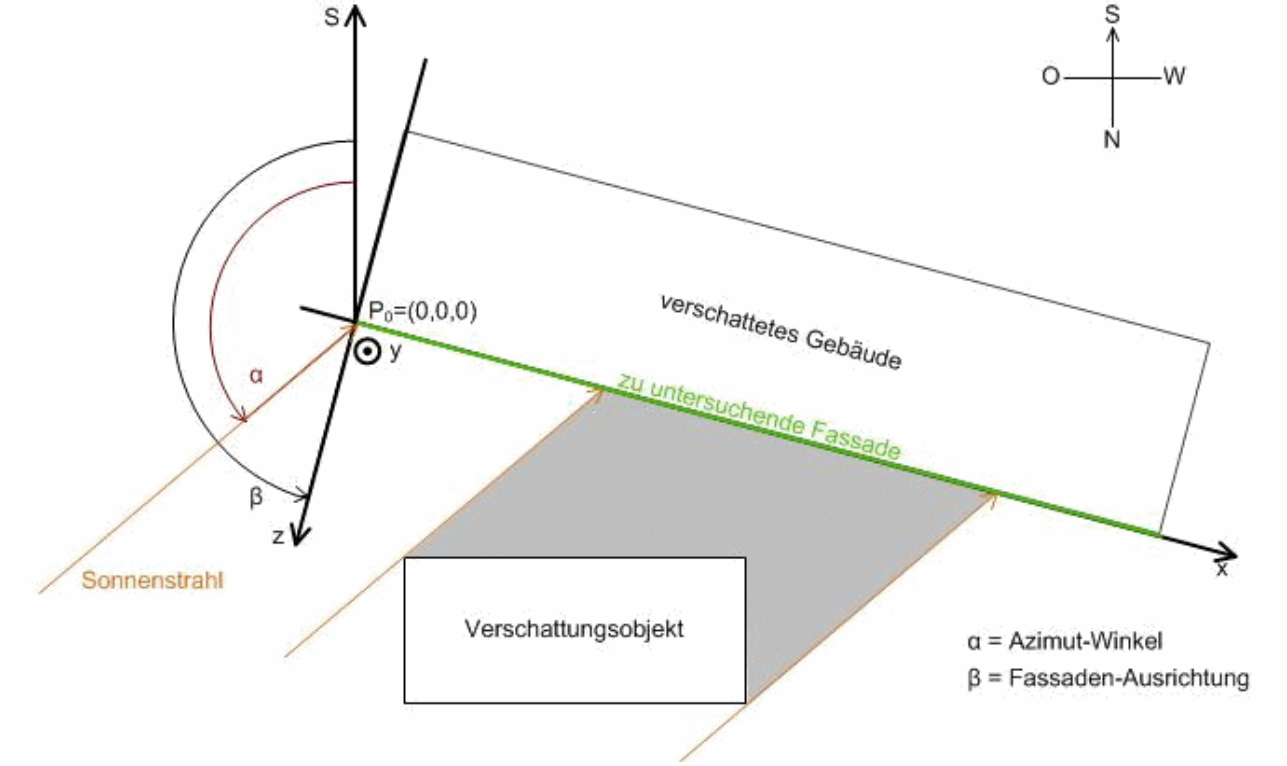


Darüber hinaus ist eine Berechnung auch dann nicht nötig, wenn die Sonne noch nicht aufgegangen ist, die Sonnenhöhe (Elevation) also unter 0° liegt. In beiden Fällen wird der Ausgang *bFcdSunlit* auf FALSE gesetzt.

Für die Südhalbkugel ändern sich die Verhältnisse. Es gilt für die Fassadenausrichtung (Blick aus dem Fenster):

Blickrichtung	Fassadenausrichtung
Süd	β=0°
Ost	β=90°
Nord	β=180°
West	β=270°

Auch ändert sich die interne Berechnung Fassade zu Sonnenstrahl:



Zur Unterscheidung der Verhältnisse - Nord- oder Südhalbkugel - ist der Eingangsparameter *bSouth* auf FALSE zu setzen (Nordhalbkugel) oder auf TRUE (Südhalbkugel)

Eingänge

```
VAR_INPUT
  tTiActl    : TIMESTRUCT;
  fFcdOrtn   : REAL;
  fAzm       : REAL;
  fElv       : REAL;
  nGrpID     : DINT;
  bSouth     : BOOL;
  aShdObj    : ARRAY[1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Name	Typ	Beschreibung
tTiActl	TIMESTRUCT	Eingabe der aktuellen Uhrzeit - hier die Ortszeit, da mithilfe dieser Zeit die verschattenden Monate berücksichtigt werden. Bei Verwendung der UTC-Zeit (bzw. GMT) kann, je nach Lage auf der Erde, der Monat mitten am Tag "umspringen".
fFcdOrtn	REAL	Fassadenausrichtung, siehe Abbildung oben.
fAzm	REAL	Sonnenrichtung zum Betrachtungszeitpunkt [°]
fElv	REAL	Sonnenhöhe zum Betrachtungszeitpunkt [°]
nGrpID	UDINT	Betrachtete Fenstergruppe. Die Gruppe 0 ist hierbei für nicht verwendete Fensterelemente reserviert (siehe FB_BA_FcdElemEntry [▶ 352]). Ein 0-Eintrag würde zu einer Fehlerausgabe führen (bErr=TRUE). Der Baustein wird dann nicht weiter abgearbeitet und *bGrpNotShdd* auf FALSE gesetzt.
bSouth	BOOL	FALSE: Berechnungen beziehen sich auf Verhältnisse der Nordhalbkugel - TRUE: auf Südhalbkugel
aShdObj	ARRAY OF ST_BA_ShdObj [▶ 279]	Liste der Verschattungsobjekte [▶ 346].

/ Ein-/Ausgänge

```
VAR_IN_OUT
  aFcdElem : ARRAY[1..BA_Param.nSunPrt_MaxColumnFcd, 1..BA_Param.nSunPrt_MaxRowFcd] OF ST_BA_FcdElem
;
END_VAR
```

Name	Typ	Beschreibung
aFcdElem	ARRAY OF ST_BA_FcdElem [▶ 278]	Liste von Fassadenelementen [▶ 346]

Ausgänge

```
VAR_OUTPUT
  bGrpNotShdd    : BOOL;
  bFcdSunlit     : BOOL;
  bErr           : BOOL;
  sErrDescr      : T_MAXSTRING;
END_VAR
```

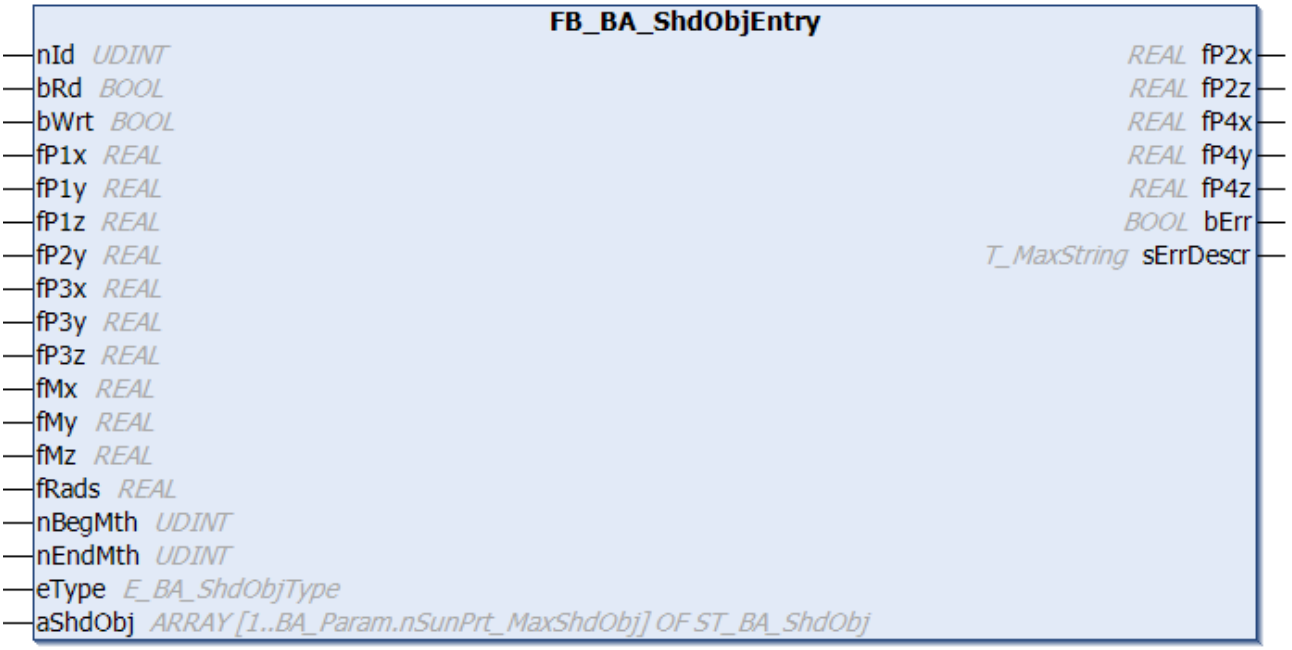
Name	Typ	Beschreibung
bGrpNotShdd	BOOL	Steht auf TRUE, solange die Fenstergruppe als nicht verschattet errechnet wird.
bFcdSunlit	BOOL	Dieser Ausgang ist dann auf TRUE gesetzt, wenn die Sonne auf die Fassade scheint. Siehe Beschreibung oben.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Indexfehler Fenstergruppe (Gruppe von Fassadenelementen): Index ist kleiner oder gleich 0.
02: Fehler: Ein Fassadenelement ist ungültig.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.17 FB_BA_ShdObjEntry



Der Funktionsbaustein FB_BA_ShdObjEntry dient zur Verwaltung aller Verschattungselemente einer Fassade, welche global in einer Liste von Verschattungselementen [▶ 346] hinterlegt sind. Er soll die Eingabe der Elementinformationen - auch im Hinblick auf die Nutzung einer Visualisierung - erleichtern. Eine schematische Darstellung der Objekte mit Beschreibung der Koordinaten ist unter Verschattungskorrektur: Grundlagen und Definitionen [▶ 332] gegeben.

Die Deklaration der Verschattungselemente erfolgt in den globalen Variablen:

```
VAR_GLOBAL
     arrShdObj : ARRAY[1..BA_Param.nSunPrt.nMaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Jedes einzelne Element *aShdObj[1]* bis *aShdObj [nMaxShdObj]* trägt die Informationen für jeweils ein Verschattungselement (ST_BA_ShdObj [▶ 279]). Dies sind neben dem gewählten Verschattungstyp (Viereck oder Kugel) die jeweils dazugehörigen Koordinaten. Im Falle eines Vierecks sind es die Eckpunkte *(fP1x*, *fP1y*, *fP1z), (fP2x*, *fP2y*, *fP2z), (fP3x*, *fP3y*, *fP3z)* und *(fP4x*, *fP4y*, *fP4z)* und im Falle einer Kugel der Mittelpunkt *(fMx*, *fMy*, *fMz)* und der Radius *fRads*. Zusätzlich kann über die Eingänge *nBegMth* und *nEndMth* die Phase der Verschattung definiert werden, was bei Objekten, wie z.B. Bäume, die im Winter kein Laub tragen, wichtig ist.

Der Baustein greift dabei über die IN-OUT-Variable *aShdObj* direkt auf das Feld dieser Informationen zu.

Anmerkung**:** Die Tatsache, dass die Vierecks Koordinaten *fP2x*, *fP2z*, *fP4x*, *fP4y*, und *fP4z* Ausgangswerte sind, ergibt sich daraus, dass sie aus den Eingabeparametern gebildet werden:

fP2x = fP1x; *fP2z = fP1z*; *fP4x = fP3x*; *fP4y = fP1y*; *fP4z = fP3z*;

Das schränkt die Eingabe eines Vierecks so weit ein, dass die seitlichen Kanten senkrecht auf dem Boden stehen (*fP2x = fP1x* und *fP4x = fP3x*), dass das Viereck keine Neigung hat (*fP2z = fP1z* und *fP4z = fP3z*) und nur nach "oben" also in positive y-Richtung eine unterschiedliche Höhe haben kann (*fP4y = fP1y*).

Die Verwendung des Bausteines erfolgt in drei Schritten:

- Auslesen
- Ändern
- Schreiben

Auslesen

Mit dem Eintrag an *nId* wird das entsprechende Element aus der Liste, *aShdObj[nId]*, ausgewählt. Eine steigende Flanke an *bRd* liest die Daten aus. Diese Werte werden den Ein- und Ausgangsvariablen des Bausteines zugewiesen. Es handelt sich hierbei um die Eingangswerte *fP1x*, *fP1y*, *fP1z*, *fP2y*, *fP3x*, *fP3y*, *fP3z*, *fMx*, *fMy*, *fMz*, *fRads* und dem Objekt-Enumerator eType und um die Ausgangswerte *fP2x*, *fP2z*, *fP4x*, *fP4y* und *fP4z*. Wichtig ist hierbei, dass im Schritt des Auslesens die Eingangswerte nicht überschrieben werden. So lassen sich alle Werte zunächst in einer Visualisierung anzeigen.

Ändern

In einem nächsten Programmschritt können dann die aufgeführten Eingabewerte verändert werden. Wenn am Eingang eType über den Wert *"eObjectTypeTetragon"* die Verwendung eines Vierecks vorgewählt ist, ergeben sich die Ausgangswerte *fP2x, fP2z, fP4x, fP4y*, und f*P4z* aus den eingegebenen Koordinaten des Vierecks, siehe oben.

Die eingegebenen Werte werden ständig auf Plausibilität überprüft. Der Ausgang *bErr* zeigt an, ob die Werte gültig sind (*bErr*=FALSE). Wenn der Wert ungültig ist, wird am Ausgang *sErrDescr* eine entsprechende Fehlermeldung ausgegeben.
Wird ein Viereck definiert, müssen lediglich die Eingänge *fP1x*, f*P1y*, f*P1z*, f*P2y*, f*P3x*, f*P3y* und f*P3z* beschrieben werden, die Eingänge *fMx*, *fMy*, *fMz* und *fRads* brauchen nicht verknüpft zu werden. Im Falle einer Kugeldefinition müssen nur *fMx*, *fMy*, *fMz* und *fRads* beschrieben werden und die Viereckkoordinaten können unverknüpft bleiben

Schreiben

Mit einer positiven Flanke an *bWrt* werden die parametrierten Daten in das Listenelement mit dem Index *nIId* geschrieben, unabhängig davon, ob sie gültige Werte darstellen oder nicht. Daher ist innerhalb der Elementstruktur ST_BA_ShdObj [▶ 279] ein Plausibilitätsbit *bVId* vorhanden, das genau diese Information an den Baustein FB_BA_ShdCorr [▶ 375] weiterreicht und dort Fehlberechnungen vorbeugt.

Diese Vorgehensweise soll nur als Vorschlag angesehen werden. Es ist auch möglich, den Baustein ganz normal in einem Schritt zu parametrieren und die eingetragenen Werte mit einer steigenden Flanke an *bWrt* in das entsprechende Listenelement zu schreiben.

Eingänge

```
VAR_INPUT
  nId          : UDINT;
  bRd          : BOOL;
  bWrt         : BOOL;
  fP1x         : REAL;
  fP1y         : REAL;
  fP1z         : REAL;
  fP2y         : REAL;
  fP3x         : REAL;
  fP3y         : REAL;
  fP3z         : REAL;
  fMx          : REAL;
  fMy          : REAL;
  fMz          : REAL;
  fRads        : REAL;
  nBegMth      : UDINT;
  nEndMth      : UDINT;
  eType        : E_BA_ShdObjType;
END_VAR
```

Name	Typ	Beschreibung
nId	UDINT	Index des gewählten Elementes. Dies bezieht sich auf die Auswahl eines Feldelementes des unter der IN-OUT-Variablen *aShdObj* angelegten Arrays. Die Variable *nId* **darf nicht Null sein!** Das ergibt sich aus der Felddefinition, welche mit 1 beginnt. Eine fehlerhafte Eingabe wird jedoch erkannt und als solche an *bErr/ sErrDescr* angezeigt.
bRd	BOOL	Mit einer positiven Flanke an diesem Eingang werden die Informationen des gewählten Elementes, *aShdObj[nId]* in den Baustein gelesen und den Eingangsvariablen *fP1x* bis eType sowie den Ausgangsvariablen *fP2x* bis *fP4z* zugewiesen. Sind zu diesem Zeitpunkt schon Daten an den Eingängen *fP1x* bis eType angelegt, so werden die zuvor ausgelesenen Daten sofort mit diesen überschrieben.
bWrt	BOOL	Eine positive Flanke schreibt die an den Eingängen *fP1x* bis *eType* angelegten sowie die ermittelten und den Ausgängen *fP2x* bis *fP4z* zugewiesenen Werte in das gewählte Feldelement *aShdObj[nId].*
fP1x	REAL	X-Koordinate des Punktes 1 des Verschattungselementes (Viereck) [m].
fP1y	REAL	Y-Koordinate des Punktes 1 des Verschattungselementes (Viereck) [m].
fP1z	REAL	Z-Koordinate des Punktes 1 des Verschattungselementes (Viereck) [m].
fP2y	REAL	Y-Koordinate des Punktes 2 des Verschattungselementes (Viereck) [m].
fP3x	REAL	X-Koordinate des Punktes 3 des Verschattungselementes (Viereck) [m].
fP3y	REAL	Y-Koordinate des Punktes 3 des Verschattungselementes (Viereck) [m].
fP3z	REAL	Z-Koordinate des Punktes 3 des Verschattungselementes (Viereck) [m].
fMx	REAL	X-Koordinate des Mittelpunktes des Verschattungselementes (Kugel) [m].
fMy	REAL	Y-Koordinate des Mittelpunktes des Verschattungselementes (Kugel) [m].
fMz	REAL	Z-Koordinate des Mittelpunktes des Verschattungselementes (Kugel) [m].
fRads	REAL	Radius des Verschattungselementes (Kugel) [m].
nBegMth	UDINT	Anfang der Verschattungsperiode (Monatszahl).
nEndMth	UDINT	Ende der Verschattungsperiode (Monatszahl).
eType	E_BA_ShdObjType [▶ 274]	Gewählter Elementtyp: Viereck oder Kugel.

Bemerkung zur Verschattungsperiode:

Die Monatseinträge dürfen nicht 0 und größer 12 sein, andernfalls sind alle Kombinationen möglich.
Beispiele:
Beginn=1, Ende=1: Verschattung im Januar.
Beginn=1, Ende=5: Verschattung von Anfang Januar bis Ende Mai.
Beginn=11, Ende=5: Verschattung von Anfang November bis Ende Mai (des folgenden Jahres).

/ Ein-/Ausgänge

```
VAR_IN_OUT
  aShdObj: ARRAY[1..BA_Param.nSunPrt_MaxShdObj] OF ST_BA_ShdObj;
END_VAR
```

Name	Typ	Beschreibung
aShdObj	ARRAY OF ST_BA_ShdObj [▶ 279]	Liste der Verschattungsobjekte [▶ 346]

Ausgänge

```
VAR_OUTPUT
  fP2x       : REAL;
  fP2z       : REAL;
  fP4x       : REAL;
  fP4y       : REAL;
  fP4z       : REAL;
  bErr       : BOOL;
  sErrDescr  : T_MAXSTRING;
END_VAR
```

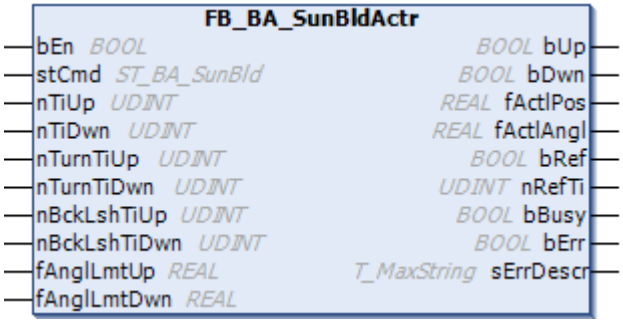
Name	Typ	Beschreibung
fP2x	REAL	Ermittelte X-Koordinate des Punktes 2 des Verschattungselementes (Viereck) [m] (siehe "Anmerkung [▶ 379]" oben).
fP2Z	REAL	Ermittelte Z-Koordinate des Punktes 2 des Verschattungselementes (Viereck) [m] (siehe "Anmerkung [▶ 379]" oben
fP4x	REAL	Ermittelte X-Koordinate des Punktes 4 des Verschattungselementes (Viereck) [m] (siehe "Anmerkung [▶ 379]" oben).
fP4y	REAL	Ermittelte Y-Koordinate des Punktes 4 des Verschattungselementes (Viereck) [m] (siehe "Anmerkung [▶ 379]" oben).
fP4z	REAL	Ermittelte Z-Koordinate des Punktes 4 des Verschattungselementes (Viereck) [m] (siehe "Anmerkung [▶ 379]" oben).
bErr	BOOL	Enthält die Fehlerbeschreibung.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Der Eingang *nId* liegt außerhalb der zulässigen Grenzen 1..*nMaxShdObj*.
02 Fehler: Die Summe der Winkel des Vierecks ist nicht 360°. Das bedeutet, dass die Eckpunkte nicht in der Reihenfolge P1, P2, P3 und P4, sondern P1, P3, P2 und P4 sind. Dies ergibt ein über Kreuz geschlagenes Viereck.
03: Fehler: Die Eckpunkte des Vierecks liegen nicht auf derselben Ebene.
04: Fehler: Die z-Komponente von P1 ist kleiner Null. Damit läge dieser Eckpunkt hinter der Fassade.
05: Fehler: Die z-Komponente von P3 ist kleiner Null. Damit läge dieser Eckpunkt hinter der Fassade.
06: Fehler: P1 ist gleich P2. Damit ist das eingetragene Objekt kein Viereck.
07: Fehler: P1 ist gleich P3. Damit ist das eingetragene Objekt kein Viereck.
08: Fehler: P1 ist gleich P4. Damit ist das eingetragene Objekt kein Viereck.
09: Fehler: P2 ist gleich P3. Damit ist das eingetragene Objekt kein Viereck.
10: Fehler: P2 ist gleich P4. Damit ist das eingetragene Objekt kein Viereck.
11: Fehler: P3 ist gleich P4. Damit ist das eingetragene Objekt kein Viereck.
12: Fehler: Der eingetragene Radius ist gleich Null.
13: Fehler: Die z-Komponente des Kugelmittelpunktes ist kleiner Null. Damit läge dieser Punkt hinter der Fassade.
14: Fehler: Fehler Objekttyp *eType* - weder Viereck noch Kugel.
15: Fehler: Fehler Monatseingabe.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

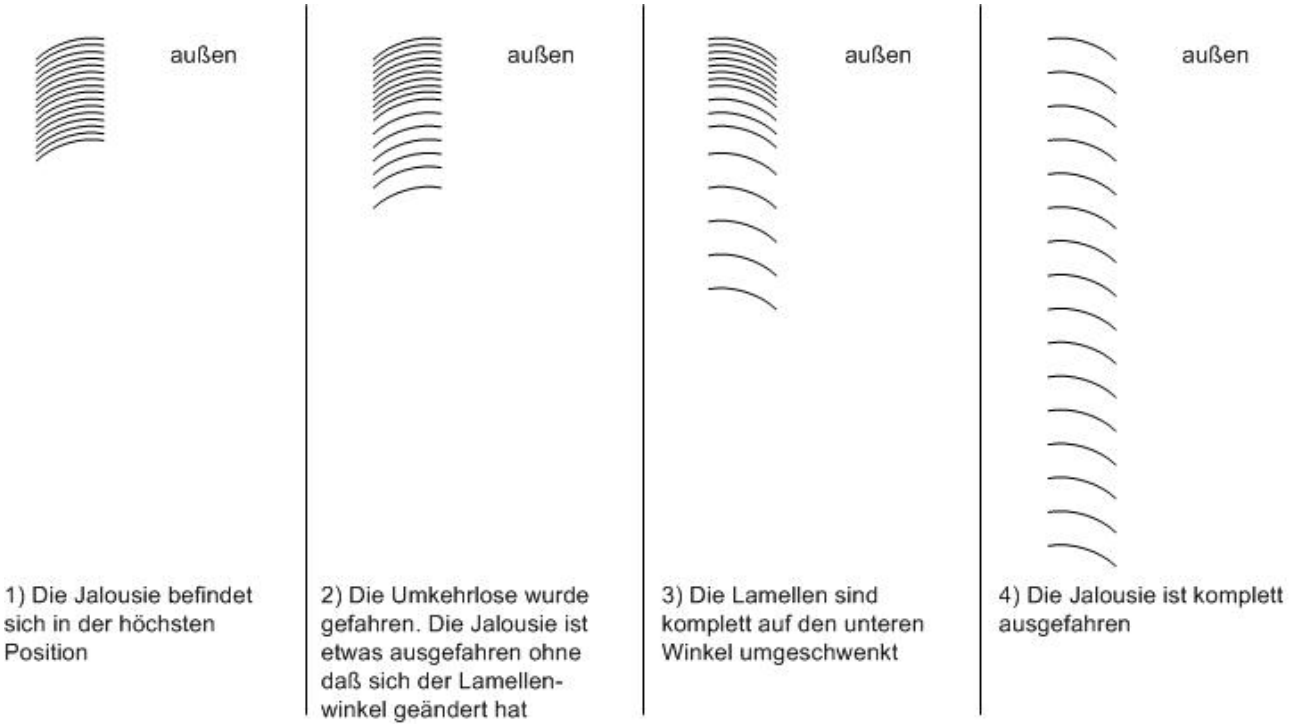
6.1.2.2.3.1.3.3.18 FB_BA_SunBldActr



Der Funktionsbaustein FB_BA_SunBldActr dient zur Positionierung einer Lamellen-Jalousie über zwei Ausgänge: hoch- und herunterfahren. Über das Positioniertelegramm stSunBld [▶ 279] kann die Jalousie auf eine beliebige (Höhen-) Position und einen Lamellenwinkel gefahren werden. Darüber hinaus beinhaltet das Positioniertelegramm stSunBld [▶ 279] auch Handbefehle, mit denen die Jalousie individuell auf bestimmte Stellungen bewegt werden kann. Diese Handbefehle werden von dem Baustein FB_BA_SunBldSwi [▶ 398] angesteuert.

Die aktuelle Höhenposition und der Lamellenwinkel werden dabei nicht durch einen zusätzlichen Encoder eingelesen, sondern intern durch die Laufzeit der Jalousie ermittelt. Der Berechnung liegt folgendes Fahrprofil zugrunde (von der höchsten und niedrigsten Position der Jalousie aus betrachtet):

Fahrprofil abwärts:

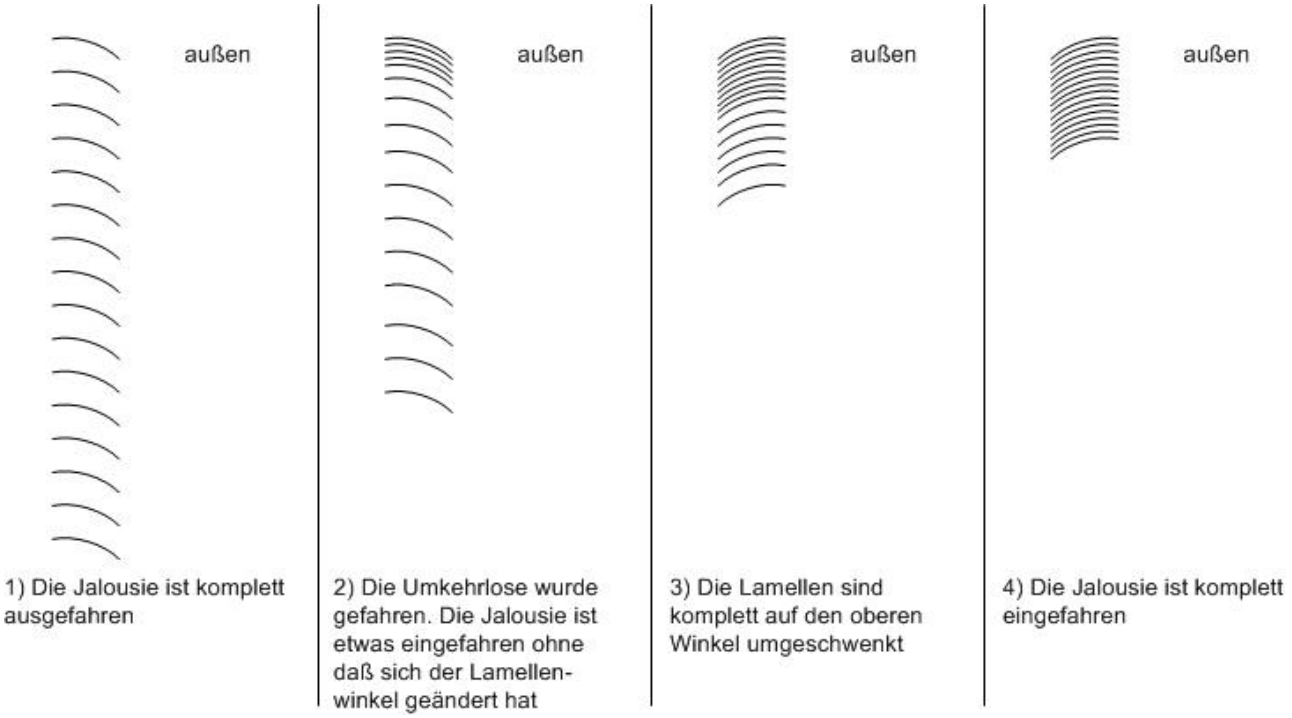


Nähere Erläuterung zu den Begriffen "Umkehrlose" und "Umschwenken", hier in der Abwärtsbewegung:

Die Jalousie beschreibt ihre Abwärtsbewegung normalerweise mit dem Lamellentiefpunkt nach außen gerichtet, wie in Bild 3).
Befindet sich die Jalousie in einer Ausgangsstellung mit dem Tiefpunkt nach innen (d.h. nach Abschluss einer Aufwärtsbewegung), so vergeht bei einer erneuten Abwärtsfahrt eine gewisse Zeit bis die Lamellen

vom "Tiefpunkt innen" bis zu "Tiefpunkt außen" zu schwenken beginnen. Während dieser Zeit ändert sich der Lamellenwinkel nicht, die Jalousie fährt nur herunter (Bild 1 und Bild 2). Diese Zeit ist ein wichtiger Parameter zur Bewegungsberechnung und wird am Baustein unter *nBckLshTiDwn* [ms] eingetragen. Da an einem x-beliebigen Punkt nach einer beliebig langen Jalousiebewegung nicht bekannt ist, ob schon etwas von der Umkehrlosen ausgefahren wurde, ist die Umkehrlose der Abwärtsbewegung bzw. deren Ausfahrzeit am sichersten zu messen, wenn die Jalousie zunächst ganz nach oben gefahren wurde. Ein weiterer wichtiger Parameter ist die Zeitspanne des darauffolgenden Umschwenkens der Lamellen von "Tiefpunkt innen" bis zu "Tiefpunkt außen". Diese Zeit ist als *nTurnTiDwn* [ms] am Baustein einzutragen.

Fahrprofil aufwärts:



Nähere Erläuterung zu den Begriffen "Umkehrlose" und "Umschwenken", hier in der Aufwärtsbewegung:

Dieser Sachverhalt ist analog zur oben beschriebenen Abwärtsbewegung: Die Jalousie beschreibt ihre Aufwärtsbewegung normalerweise mit dem Lamellentiefpunkt nach innen gerichtet, wie in Bild 3).
Befindet sich die Jalousie in einer Ausgangsstellung mit dem Tiefpunkt nach außen (d.h. nach Abschluss einer Abwärtsbewegung), so vergeht bei einer erneuten Aufwärtsfahrt eine gewisse Zeit bis die Lamellen vom "Tiefpunkt außen" bis zu "Tiefpunkt innen" zu schwenken beginnen. Während dieser Zeit ändert sich der Lamellenwinkel nicht, die Jalousie fährt nur herauf (Bild1 und Bild 2). Auch diese Zeit ist ein wichtiger Parameter zur Bewegungsberechnung und wird am Baustein unter *nBckLshTiUp* [ms] eingetragen. Da an einem x-beliebigen Punkt nach einer beliebig langen Jalousiebewegung nicht bekannt ist, ob schon etwas von der Umkehrlosen ausgefahren wurde, ist die Umkehrlose der Aufwärtsbewegung bzw. deren Ausfahrzeit am sichersten zu messen, wenn die Jalousie zunächst ganz nach unten gefahren wurde. Ein weiterer wichtiger Parameter ist die Zeitspanne des darauffolgenden Umschwenkens der Lamellen von "Tiefpunkt außen" bis zu "Tiefpunkt innen". Diese Zeit ist als *nTurnTiUp* [ms] am Baustein einzutragen.

Parametrierung

Zur Berechnung der (Höhen-)Position und des Lamellenwinkels sind nun jeweils für die Auf- und Abwärtsbewegung folgende Zeiten zu ermitteln:

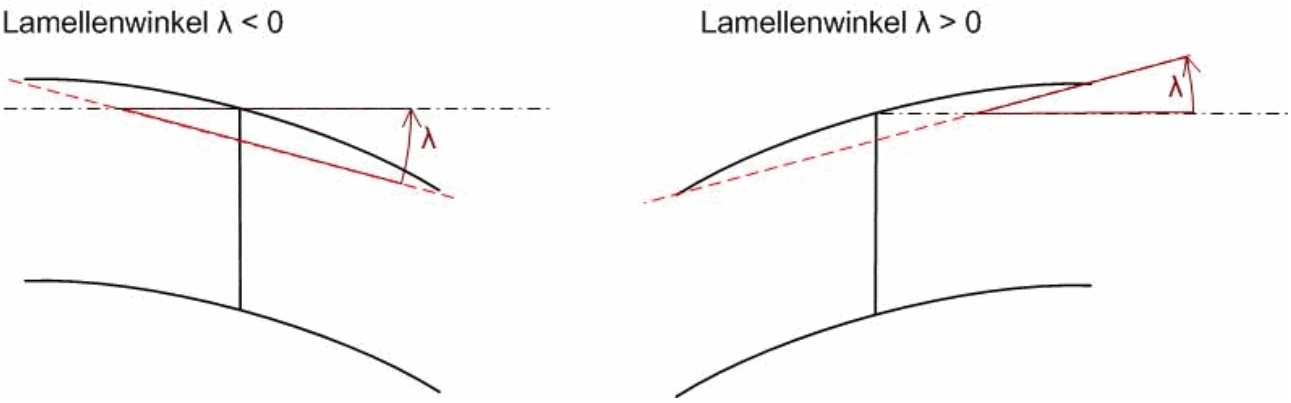
- die Verfahrdauer der Umkehrlose (*nBckLshTiUp* / *nBckLshTiDwn* [ms])
- die Verfahrdauer des Umschwenkens (*nTurnTiUp* / *nTurnTiDwn* [ms])
- die Gesamt-Verfahrdauer (*nTiUp* / *nTiDwn* [ms])

Des Weiteren sind zur Berechnung erforderlich:

- der höchste Lamellenwinkel nach einem Umschwenken nach oben (*fAnglLmtUp* [°])

- der niedrigste Lamellenwinkel nach einem Umschwenken nach unten (*fAnglLmtDwn* [°])

Der Lamellenwinkel λ ist dabei durch eine gedachte Gerade durch die Endpunkte der Lamelle zur Horizontalen definiert.



Funktionsweise

Der Baustein steuert die Jalousie grundsätzlich über die Informationen, aus dem Positioniertelegramm stSunBld [▶ 702]. Ist der Automatikmodus aktiv (*bManMod* = FALSE), so wird immer die aktuelle Position und Lamellenwinkel angefahren, wobei Änderungen sofort berücksichtigt werden. Die Positionierung auf die Höhe hat dabei Vorrang: Es wird zunächst die eingegebene Höhe und danach der Lamellenwinkel angefahren. Aus Gründen der Einfachheit bleibt dabei der Positionsfehler durch das Winkel-Verfahren unberücksichtigt. Im Handbetrieb (*bManMod* = TRUE) steuern die Befehle *bManUp* und *bManDwn* die Jalousie.
Bei einem Wechsel vom Hand- in den Automatikmodus wird in jedem Fall ein Automatik-Fahrbefehl ausgelöst.

Referenzieren

Ein sicheres Referenzieren ist gegeben, wenn die Jalousie länger als ihre komplette Hochlaufzeit nach oben hin angesteuert wird. Die Position ist dann auf jeden Fall "0" und der Lamellenwinkel auf seinem Maximum. Da eine Jalousiepositionierung ohne Encoder naturgemäß immer fehlerbehaftet ist, ist es wichtig möglichst oft automatisch zu referenzieren: jedes Mal, wenn die Position "0" angefahren werden soll (der Winkel spielt dabei keine Rolle) fährt die Jalousie zunächst ganz normal mit kontinuierlicher Positionsberechnung nach oben. Erreicht sie den errechneten Positionswert 0 %, so wird der Ausgang *bUp* gehalten, und zwar noch einmal für die komplette Hochlaufzeit + 5 s.

Der Referenziervorgang kann durch einen Handbefehl "herunter" oder durch eine neue Positionierung unterbrochen werden.

Beim Neustart der Anlage referenziert der Baustein in jedem Fall.

Zielgenauigkeit

Da der Baustein die Position der Jalousie nur über Laufzeiten ermittelt, spielt die Zykluszeit der SPS-Task eine entscheidende Rolle bei der Positioniergenauigkeit. Besteht beispielsweise eine Umschwenkzeit von 1 s für einen Lamellen-Winkelbereich von - 70° bis 10°, so liegt bei einer Zykluszeit von 50 ms die Genauigkeit bei +/- 4°.

Eingänge

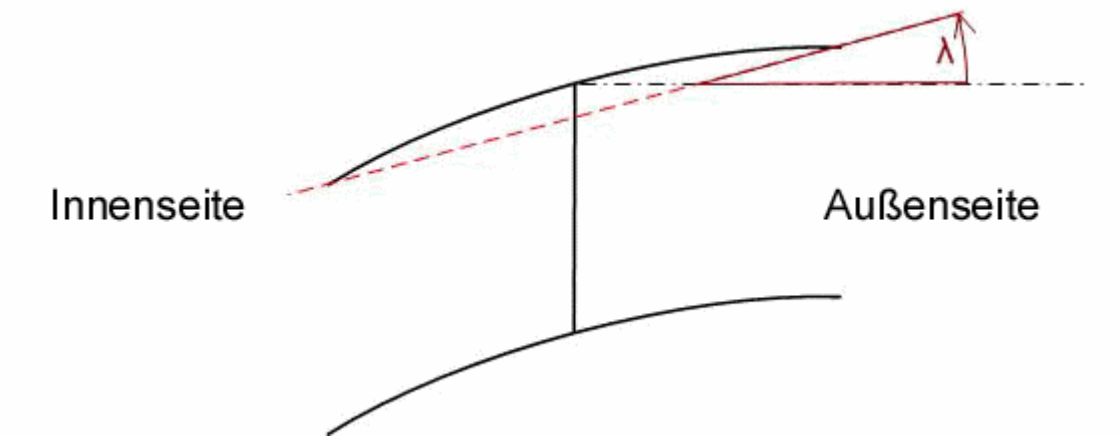
```
VAR_INPUT
  bEn               : BOOL;
  stCmd             : ST_BA_SunBld;
  nTiUp             : UDINT;
  nTiDwn            : UDINT;
  nTurnTiUp         : UDINT;
  nTurnTiDwn        : UDINT;
  nBckLshTiUp       : UDINT;
  nBckLshTiDwn      : UDINT;
```

```
  fAnglLmtUp          : REAL;
  fAnglLmtDwn         : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabeeingang für den Baustein. Solange dieser Eingang auf TRUE steht, nimmt der Aktorbaustein Befehle entgegen und arbeitet, wie oben beschrieben. Ein FALSE-Signal an diesen Eingang setzt die Steuerausgänge *bUp* und *bDwn* zurück und der Funktionsbaustein verharrt in einem Ruhezustand.
stCmd	ST_BA_SunBld [▶ 279]	Positioniertelegramm
nTiUp	UDINT	Komplette Hochfahrzeit [ms]
nTiDwn	UDINT	Komplette Zeit zum Herunterfahren [ms].
nTurnTiUp	UDINT	Zeit zum Umschwenken der Lamellen in obere Richtung [ms].
nTurnTiDwn	UDINT	Zeit zum Umschwenken der Lamellen in untere Richtung [ms].
nBckLshTiUp	UDINT	Zeit zum Ausfahren der Umkehrlose in obere Richtung [ms]. Diese Eingabe ist intern auf einen Minimalwert von 0 begrenzt.
nBckLshTiDwn	UDINT	Zeit zum Ausfahren der Umkehrlose in untere Richtung [ms]. Diese Eingabe ist intern auf einen Minimalwert von 0 begrenzt.
fAnglLmtUp	REAL	Höchste Stellung der Lamellen [°].

Diese Stellung ist erreicht, wenn die Jalousie ganz hochgefahren ist.

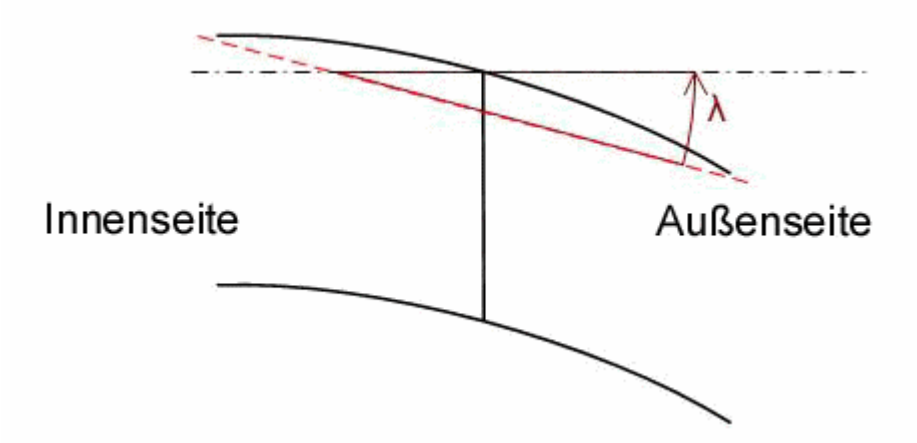
Der Lamellenwinkel λ, so wie er einleitend definiert ist, ist dann typischerweise größer Null.



Name	Typ	Beschreibung
fAnglLmtDwn	REAL	Niedrigste Stellung der Lamellen [°].

Diese Stellung ist erreicht, wenn die Jalousie ganz heruntergefahren ist.

Der Lamellenwinkel λ, so wie er einleitend definiert ist, ist dann typischerweise kleiner Null.



Ausgänge

```
VAR_OUTPUT
  bUp           : BOOL;
  bDwn          : BOOL;
  fActlPos      : REAL;
  fActlAngl     : REAL;
  bRef          : BOOL;
  nRefTi        : UDINT;
  bBusy         : BOOL;
  bErr          : BOOL;
  sErrDesc      : T_MAXSTRING;
END_VAR
```

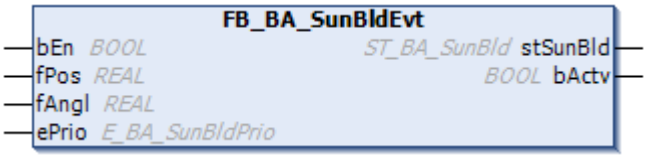
Name	Typ	Beschreibung
bUp	BOOL	Steuerausgang Jalousie hoch
bDwn	BOOL	Steuerausgang Jalousie herunter
fActlPos	REAL	Aktuelle Position in Prozent
fActlAngl	REAL	Aktueller Lamellenwinkel [°]
bRef	BOOL	Die Jalousie befindet sich in der Referenzierung, d.h. für die die komplette Hochlaufzeit + 5s wird der Ausgang *bUp* gesetzt. Nur ein Handbefehl "herunter" kann die Jalousie in Gegenrichtung bewegen und diesen Modus beenden.
nRefTi	UDINT	Referenzier-Countdown-Anzeige [s]
bBusy	BOOL	Ein Positionier- oder Referenziervorgang findet statt.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Hoch/Herunter-Timer = 0.
02: Fehler: Turning/Wendetimer = 0.
03: Fehler: Lamellenwinkelgrenzen: Die obere Grenze ist kleiner oder gleich der unteren Grenze (*fAnglLmtUp* <= *fAnglLmtDwn*).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.19 FB_BA_SunBldEvt



Der Funktionsbaustein FB_BA_SunBldEvt dient zur Positions- und Winkelvorgabe bei einem beliebigen Ereignis. Sie kann beispielsweise genutzt werden, um eine Parkposition anzufahren oder im Wartungsfall die Jalousie hochfahren zu lassen.

Die Funktion wird über den Eingang *bEn* aktiviert. Ist dies der Fall, so wird der Aktiv-Merker *bActv* im Positioniertelegramm stSunBld [▶ 279] am Ausgang gesetzt und und die an den In-Out-Variablen eingetragenen Werte *fPos* für die Jalousiehöhe [%] und *fAngl* für den Lamellenwinkel [°] in diesem Telegramm weitergereicht. Ist die Funktion durch Rücksetzen von *bEn* nicht mehr aktiv, so wird der Aktiv-Merker im Positioniertelegramm stSunBld [▶ 279] zurück und die Positionen für Höhe und Winkel auf "0" gesetzt. Mit dem Prioritätenbaustein (z.B. FB_BA_SunBldPrioSwi4 [▶ 612]) kann durch das Rücksetzen eine Funktion niedrigerer Priorität die Steuerung übernehmen.

Eingänge

```
VAR_INPUT
  bEn      : BOOL;
  fPos     : REAL;
  fAngl    : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Ein TRUE-Signal an diesem Eingang aktiviert den Baustein und übergibt die eingetragenen Sollwerte im Positioniertelegramm ST_BA_SunBld [▶ 279] zusammen mit dem Aktivmerker. Ein FALSE-Signal setzt den Aktivmerker wieder zurück, sowie Position und Winkel auf Null.
fPos	REAL	Höhenposition der Jalousie [%] im Falle einer Aktivierung.
fAngl	REAL	Lamellenwinkel der Jalousie [°] im Falle einer Aktivierung.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio    : E_BA_SunBldPrio := E_BA_SunBldPrio.eScene1;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des aktiven Telegramms.

Ausgänge

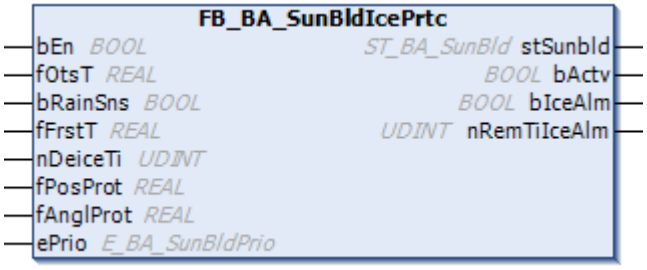
```
VAR_OUTPUT
  stSunBld   : ST_BA_SunBld;
  bActv      : BOOL;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabestruktur der Jalousiestellungen.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.20 FB_BA_SunBldIcePrtc



Der Funktionsbaustein FB_BA_SunBldIcePrtc behandelt den richtungsunabhängigen Vereisungsschutz.

Der Witterungsschutz hat bei der Jalousiesteuerung die höchste Priorität (siehe Übersicht [▶ 339]) und soll sicherstellen, dass die Jalousie durch Eis beschädigt wird.

Eine bevorstehende Vereisung wird dadurch erkannt, dass während einer Niederschlagserkennung an *bRainSns* die gemessene Außentemperatur *fOtsT* unterhalb des Frost-Grenzwertes *fFrstT* liegt. Dieses Ereignis wird intern gespeichert und bleibt dann so lange bestehen, bis sichergestellt ist, dass das Eis wieder abgetaut ist. Dazu muss die Außentemperatur den Frost-Grenzwert für die eingetragene Enteisungszeit *nDeiceTi* [s] überschritten haben. Aus Sicherheitsgründen wird das Vereisungsereignis persistent, also über einen SPS-Ausfall hinweg gespeichert. Fällt die Steuerung also während einer Ver- bzw. Enteisungsperiode aus, so gilt die Jalousie nach Wiederanlauf der Steuerung als neu vereist und der Enteisungszeitmesser startet wieder.

Bei Vereisungsgefahr wird die Jalousie in die Schutzposition gefahren die durch *fPosProt* (Höhenposition [%]) und *fAnglProt* (Lamellenwinkel [°]) vorgegeben wird.

Eingänge

```
VAR_INPUT
  bEn               : BOOL;
  fOtsT             : REAL;
  bRainSns          : BOOL;
  fFrstT            : REAL;
  nDeiceTi          : UDINT;
  fPosProt          : REAL;
  fAnglProt         : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Ein TRUE-Signal an diesem Eingang aktiviert den Baustein und übergibt die eingetragenen Sollwerte im Positioniertelegramm ST_BA_SunBld [▶ 279] zusammen mit dem Aktivmerker. Ein FALSE-Signal setzt den Aktivmerker wieder zurück, sowie Position und Winkel auf Null. *bActv* steht auf FALSE. Das bedeutet, dass über die Prioritätssteuerung eine andere Funktionalität die Jalousieansteuerung übernimmt.
fOtsT	REAL	Außentemperatur [°C]
bRainSns	BOOL	Eingang für einen Niederschlagssensor.
fFrstT	REAL	Vereisungstemperatur-Grenzwert [°] Celsius. Dieser Wert darf nicht größer als 0 sein. Anderenfalls wird ein Fehler ausgegeben.
nDeiceTi	UDINT	Zeit zum Abtauen der Jalousie nach Vereisung [s]. Danach wird der Vereisungsalarm zurückgesetzt.
fPosProt	REAL	Höhenposition der Jalousie [%] im Schutzfall.
fAnglProt	REAL	Lamellenwinkel der Jalousie [°] im Schutzfall.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio    : E_BA_SunBldPrio := E_BA_SunBldPrio.eIce;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des aktiven Telegramms.

Ausgänge

```
VAR_OUTPUT
  stSunBld           : ST_BA_SunBld;
  bActv              : BOOL;
  bIceAlm            : BOOL;
  nRemTiIceAlm       : UDINT;
END_VAR
```

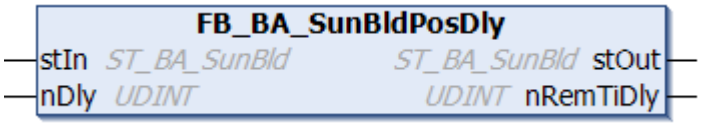
Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabestruktur der Jalousiestellungen.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.
bIceAlm	BOOL	Zeigt den Vereisungsalarm an.
nRemTiIceAlm	UDINT	Bei aufkommenden Vereisungsfall (*bIceAlm* = TRUE) wird dieser Sekundenzähler auf die Enteisungszeit gesetzt. Sobald die Temperatur über dem eingetragenen Frostpunkt (*fFrstT*) liegt, werden hier die verbleibenden Sekunden bis zur Entwarnung (*bIceAlm* = FALSE) angezeigt. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.

Wenn ein Fehler ansteht, wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.21 FB_BA_SunBldPosDly



Der Funktionsbaustein verzögert Positionsänderungen von Automatik-Befehlen.

Wenn durch ein Ereignis, zum Beispiel Wetterschutz, zu viele Jalousieantriebe gleichzeitig gestartet werden, ist es möglich, dass die Summe der hohen Motor-Anzugsströme Sicherungen auslösen. Es ist daher zu empfehlen, die Jalousieantriebe kurz hintereinander zu starten, um den Gesamtstrom zu minimieren.

Dieser Baustein gibt Automatikbefehle vom Eingangstelegramm stIn [▶ 279] verzögert an das Ausgangstelegramm stOut [▶ 279] weiter. Er beachtet dazu drei Fälle

1. die Jalousieposition *fPos* hat sich im Automatikmodus verändert (*bManMode* = FALSE im Telegramm *stIn*)
2. der Lamellenwinkel *fAngl* hat sich im Automatikmodus verändert (*bManMode* = FALSE im Telegramm *stIn*)
3. der Hand-Modus wurde gerade verlassen, d.h. der Automatikmodus gerade aktiv (fallende Flanke *bManMode* im Telegramm *stIn*)

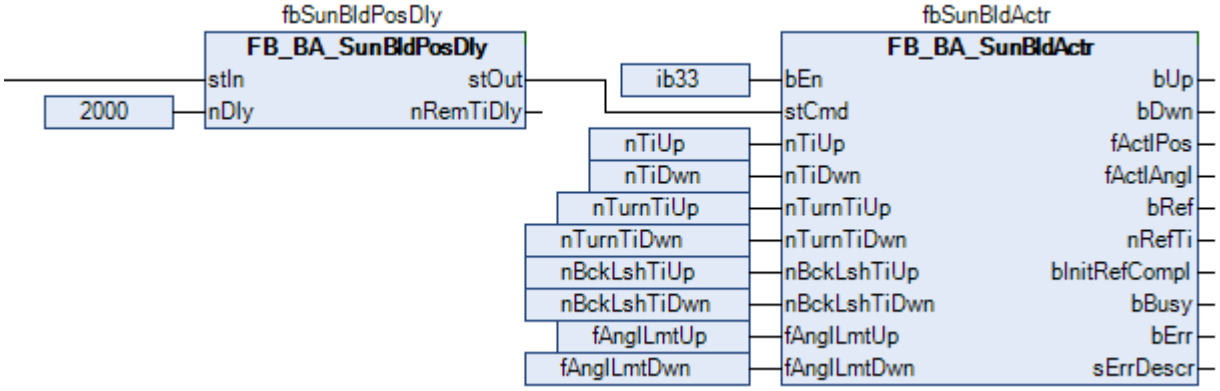
Das Ausgangstelegramm *stOut*, ist immer eine direkte Kopie des Eingangstelegramms *stIn*. In diesen drei Fällen jedoch wird das Ausgangstelegramm *stOut* für die Zeit von *nDly* [ms] festgesetzt.

Dadurch wird die über den Baustein FB_BA_SunBldActr [▶ 383] angesteuerte Jalousie für den Zeitraum der Verzögerung auf ihrer Position gehalten. Jede weitere Änderung nach den oben genannten Kriterien innerhalb der Verzögerungszeit startet den Zeitgeber neu.

Ein Wechsel auf Hand im Eingangstelegramm (*bManMode* = TRUE) jedoch löscht den Warte-Timer unmittelbar. Das (Hand-)Telegramm wird unverzögert durchgereicht. So werden **nur** Automatiktelegramme verzögert.

Anwendung

Vorzugsweise direkt vor dem Jalousie-Aktorbaustein:



Eingänge

```
VAR_INPUT
  stIn        : ST_BA_Sunblind;
  nDly        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
stIn	ST_BA_SunBld [▶ 279]	Eingangs-Positioniertelegramm.
nDly	UDINT	Verzögerungszeit des Aktiv-Bits im Positioniertelegramm [ms].

Ausgänge

```
VAR_OUTPUT
  stOut           : ST_BA_Sunblind;
  nRemTiDly       : UDINT;
END_VAR
```

Name	Typ	Beschreibung
stOut	ST_BA_SunBld [▶ 279]	Ausgangs-Positioniertelegramm.
nRemTiDly	UDINT	Anzeigeausgang abgelaufene Verzögerungszeit [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.22 FB_BA_SunBldPosHMI



Der Funktionsbaustein FB_BA_SunBldPosHMI dient zur Positions- und Winkelvorgabe über eine Bedienoberfläche. Mit jeder steigenden Flanke an *bStart* wird ein aktives Positioniertelegramm mit der Zielposition *fPos* und dem Lamellenwinkel *fAngl* am Ausgang *stSunBld* ausgegeben.

Der Eingang *bRstManFnct* dient zum Rücksetzen des Ausgabetelegrammes. Dabei wird das Telegramm wie folgt beschrieben:

```
stSunBld.bActv := FALSE;
stSunBld.fPos  := 0.0;
stSunBld.fAngl := 0.0;
```

Der Eingang *bRstManFnct* wirkt statisch: solange er auf TRUE steht, wird das Ausgabetelegramm auf die oben erwähnten Werte gesetzt.

Außerdem wird der Bausteinausgang *bActv* auf FALSE gesetzt. Dieser soll anzeigen, ob ein aktives Telegramm ausgegeben wird.

Eingänge

```
VAR_INPUT
  fPos          : REAL;
  fAngl         : REAL;
  bStart        : BOOL;
  bRstManFnct   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fPos	REAL	Eingabe der Zielposition
fAngl	REAL	Eingabe des Lamellenwinkels
bStart	BOOL	Eine steigende Flanke an diesem Eingang gibt ein aktives Telegramm mit der eingetragenen Zielposition *fPos* und dem Lamellenwinkel *fAngl* aus.
bRstManFnct	BOOL	Ein TRUE an diesem Eingang löscht das Telegramm am Ausgang (*stSunBld.bActv* = FALSE, *bActv* = FALSE, *stSunBld.fPos* = 0, *stSunBld.fAngl* = 0) und es werden keine neuen Werte durchgereicht. Der Ausgang *bActv* geht dann ebenfalls auf FALSE.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio  : E_BA_SunBldPrio := E_BA_SunBldPrio.eManualActuator;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des Ausgabetelegramms.

Ausgänge

```
VAR_OUTPUT
  stSunBld  : ST_BA_SunBld;
  bActv     : BOOL;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.23 FB_BA_SunBldPrioSwi4



Der Funktionsbaustein FB_BA_SunBldPrioSwi4 dient zur Prioritätssteuerung für bis zu 4 Positioniertelegramme (*stSunBld_Prio1* ... *stSunBld_Prio4*) des Typs ST_BA_SunBld [▶ 279] aus verschiedenen Steuerbausteinen.

Dabei hat das Telegramm an *stSunBld_Prio1* die höchste und das an *stSunBld_Prio4* die niedrigste Priorität. Das aktive Telegramm der höchsten Priorität wird am Ausgang *stSunBld* ausgegeben. Aktiv bedeutet, dass innerhalb der Struktur des Positioniertelegramms die Variable *bActv* gesetzt ist.

Dieser Baustein ist so zu programmieren, dass immer eines der angelegten Telegramme aktiv ist. Falls kein Telegramm aktiv ist, wird am Ausgang ein Leertelegramm ausgegeben, d.h. *fPos* = 0, *fAngl* = 0, *bManUp* = FALSE, *bManDwn* = FALSE, *bManMod* = FALSE, *bActv* = FALSE. Da der Jalousiebaustein FB_BA_SunBldActr [▶ 383] bzw. der Rollladenbaustein FB_BA_RolBldActr [▶ 373] seinerseits nicht auf den

Merker *bActv* achtet, würde dieses Telegramm als Fahrbefehl auf Position "0", also vollständig geöffnet, gewertet werden. Das Fehlen eines aktiven Telegramms stellt damit kein Sicherheitsrisiko für die Jalousie dar.

Eingänge

```
VAR_INPUT
  stSunBld_Prio1    : ST_BA_SunBld;
  stSunBld_Prio2    : ST_BA_SunBld;
  stSunBld_Prio3    : ST_BA_SunBld;
  stSunBld_Prio4    : ST_BA_SunBld;
END_VAR
```

Name	Typ	Beschreibung
stSunBld_PrioN	ST_BA_SunBld [▶ 279]	Zur Auswahl stehende Positioniertelegramme. Dabei hat *stSunBld_Prio1* die höchste und *stSunBld_Prio4* die niedrigste Priorität.

Ausgänge

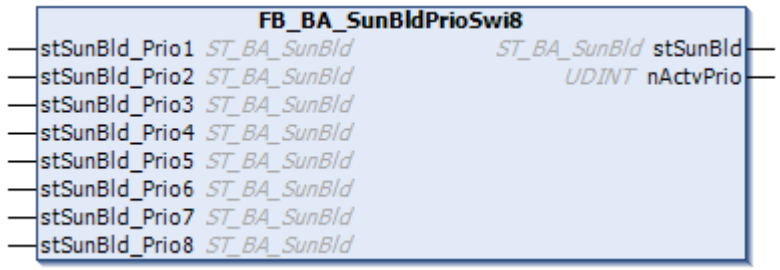
```
VAR_OUTPUT
  stSunBld     : ST_BA_SunBld;
  nActvPrio    : UDINT;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
nActvPrio	UDINT	Aktives Positioniertelegramm. Ist keines aktiv, so wird "0" ausgegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.24 FB_BA_SunBldPrioSwi8



Der Funktionsbaustein FB_BA_SunBldPrioSwi8 dient zur Prioritätssteuerung für bis zu 8 Positioniertelegramme (*stSunBld_Prio1* ... *stSunBld_Prio8*) des Typs ST_BA_SunBld [▶ 279] aus verschiedenen Steuerbausteinen.

Dabei hat das Telegramm an *stSunBld_Prio1* die höchste und das an *stSunBld_Prio8* die niedrigste Priorität. Das aktive Telegramm der höchsten Priorität wird am Ausgang *stSunBld* ausgegeben. Aktiv bedeutet, dass innerhalb der Struktur des Positioniertelegramms die Variable *bActv* gesetzt ist.

Dieser Baustein ist so zu programmieren, dass immer eines der angelegten Telegramme aktiv ist. Falls kein Telegramm aktiv ist, wird am Ausgang ein Leertelegramm ausgegeben, d.h. *fPos* = 0, *fAngl* = 0, *bManUp* = FALSE, *bManDwn* = FALSE, *bManMod* = FALSE, *bActv* = FALSE. Da der Jalousiebaustein FB_BA_SunBldActr [▶ 383] bzw. der Rollladenbaustein FB_BA_RolBldActr [▶ 373] seinerseits nicht auf den Merker *bActv* achtet, würde dieses Telegramm als Fahrbefehl auf Position "0", also vollständig geöffnet, gewertet werden. Das Fehlen eines aktiven Telegramms stellt damit kein Sicherheitsrisiko für die Jalousie dar.

Eingänge

```
VAR_INPUT
  stSunBld_Prio1    : ST_BA_SunBld;
  stSunBld_Prio2    : ST_BA_SunBld;
  stSunBld_Prio3    : ST_BA_SunBld;
  stSunBld_Prio4    : ST_BA_SunBld;
  stSunBld_Prio5    : ST_BA_SunBld;
  stSunBld_Prio6    : ST_BA_SunBld;
  stSunBld_Prio7    : ST_BA_SunBld;
  stSunBld_Prio8    : ST_BA_SunBld;
END_VAR
```

Name	Typ	Beschreibung
stSunBld_PrioN	ST_BA_SunBld [▶ 279]	Zur Auswahl stehende Positioniertelegramme. Dabei hat *stSunBld_Prio1* die höchste und *stSunBld_Prio8* die niedrigste Priorität.

Ausgänge

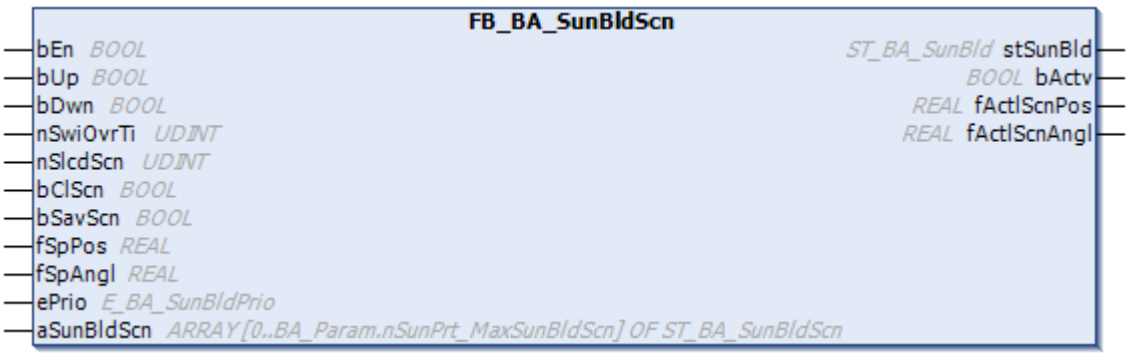
```
VAR_OUTPUT
  stSunBld     : ST_BA_SunBld;
  nActvPrio    : UDINT;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
nActvPrio	UDINT	Aktives Positioniertelegramm. Ist keines aktiv, so wird "0" ausgegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.25 FB_BA_SunBldScn



Der Funktionsbaustein FB_BA_SunBldScn stellt eine Erweiterung der Handbedienung FB_BA_SunBldSwi [▶ 398] um eine Szenen-Speicher- und Aufruf-Funktionalität dar. Damit lässt sich die Jalousieansteuerung FB_BA_SunBldActr [▶ 383] bzw. die Rollladenansteuerung FB_BA_RolBldActr [▶ 373] sowohl im Handbedienmodus ansteuern, als auch zuvor gespeicherte Positionen (Szenen) direkt anfahren. Es können bis zu 21 Szenen gespeichert werden.

Betrieb

Der Funktionsbaustein steuert über die Befehlseingänge *bUp* und *bDwn* den Jalousiebaustein FB_BA_SunBldActr [▶ 383] bzw. den Rollladenbaustein FB_BA_RolBldActr [▶ 373] im Handbetrieb an, wobei *bUp* Vorrang hat. Die Befehle werden an die jeweiligen Kommandos *bManUp* und *bManDwn* des

Positioniertelegramms [▶ 279] weitergereicht. Ist ein Befehlseingang länger als die eingetragene Zeit *nSwiOvrTi* [ms] aktiviert, so geht der entsprechende Steuerbefehl in Selbsthaltung. Ein erneutes Aktivieren eines Befehlseinganges löscht diese Selbsthaltung wieder.

Eine steigende Flanke an *bSavScn* speichert die aktuelle Position und den Lamellenwinkel in die unter *nSlcdScn* angewählte Szene. Dieser Vorgang ist jederzeit möglich, auch während einer aktiven Positionierung.

Eine steigende Flanke an *bClScn* ruft diejenige Szene ab, welche am Eingang *nSlcdScn* anliegt. Dieser Szenenabruf kann durch Handbefehle an *bUp* und *bDwn* unterbrochen werden.

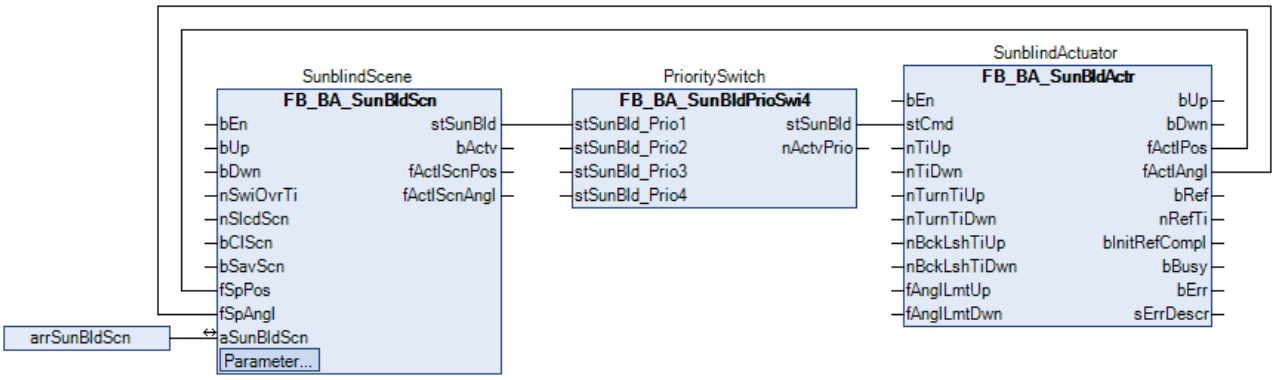
Die Handsteuerung und die Szenenpositionierung erfolgen jeweils mit der Priorität, welche unter *ePrio* vorgewählt ist.

Ist die Funktion durch Rücksetzen von *bEn* nicht mehr aktiv, so wird der Aktiv-Merker im Positioniertelegramm stSunBld [▶ 279] zurück und die Positionen für Höhe und Winkel auf "0" gesetzt. Mit dem Prioritätenbaustein (z.B. *FB_BA_SunBldPrioSwi4*) kann durch das Rücksetzen eine Funktion niedrigerer Priorität die Steuerung übernehmen.

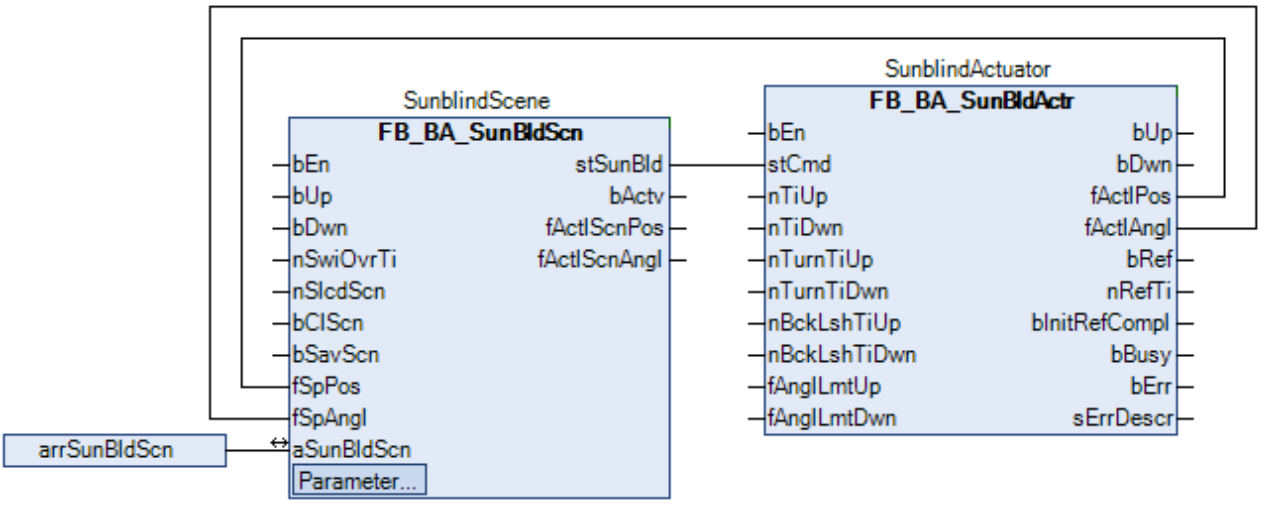
Verknüpfung an Jalousiebaustein

Der Szenenanwahlbaustein kann, wie der "normale" Handbedienbaustein FB_BA_SunBldSwi [▶ 398], entweder über eine voran gestellte Prioritätssteuerung FB_BA_SunBldPrioSwi4 bzw. FB_BA_SunBldPrioSwi8 oder aber direkt an den Jalousiebaustein angeschlossen werden. Die Verbindung erfolgt dabei über das Positioniertelegramm ST_BA_Sunbld [▶ 279]. Des Weiteren benötigt der Szenenbaustein die aktuelle Position aus dem Jalousiebaustein der Referenz-Jalousie:

Verwendung einer Prioritätssteuerung:



Direkte Beschaltung:



Eingänge

```
VAR_INPUT
  bEn             : BOOL;
  bUp             : BOOL;
  bDwn            : BOOL;
  nSwiOvrTi       : UDINT;
  nSlcdScn        : UDINT;
  bClScn          : BOOL;
  bSavScn         : BOOL;
  fSpPos          : REAL;
  fSpAngl         : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_Sunbld [▶ 279] werden für Position und Winkel jeweils 0 ausgegeben - *bManMod* und *bActv* stehen jeweils auf FALSE. Das bedeutet für eine Beschaltung mit Prioritätssteuerung, dass eine andere Funktionalität die Jalousieansteuerung übernimmt. Eine direkte Beschaltung hingegen lässt die Jalousie direkt auf Position 0, also ganz nach oben fahren, da der Aktorbaustein das Bit *bActv* selbst nicht auswertet.
bUp	BOOL	Befehlseingang Jalousie hoch.
bDwn	BOOL	Befehlseingang Jalousie herunter.
nSwiOvrTi	UDINT	Zeit [ms] bis bei dauerhaft aktiviertem Befehlseingang der entsprechende Handbefehl im Positioniertelegramm ST_BA_Sunbld [▶ 279] in Selbsthaltung geht. Intern begrenzt auf einen Minimalwert von 0.
nSlcdScn	UDINT	Angewählte Szene, welche entweder gespeichert (*bSavScn*) oder aufgerufen (*bClScn*) werden soll. Intern begrenzt auf einen Minimalwert von 0 bis BA_Param.nSunPrt_MaxSunBldScn [▶ 289]
bClScn	BOOL	Angewählte Szene aufrufen
bSavScn	BOOL	Angewählte Szene speichern
fSpPos	REAL	Sollposition [%], welche in der gewählten Szene gesichert werden soll. Ist mit der Ist-Position des Aktorbausteins FB_BA_SunBldActr bzw. FB_BA_RolBldActr der Referenz-Jalousie/Rolllade zu verknüpfen, um dadurch eine vorher manuell angefahrene Position speichern zu können. Intern begrenzt auf Werte von 0 bis 100.
fSpAngl	REAL	dto. Lamellenwinkel [°]

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio     : E_BA_SunBldPrio := E_BA_SunBldPrio.eScene1;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des aktiven Telegramms.

/ Ein-/Ausgänge

```
VAR_IN_OUT
  aSunBldScn : ARRAY[0..BA_Param.nSunPrt_MaxSunBldScn] OF ST_BA_SunBldScn;
END_VAR
```

Name	Typ	Beschreibung
aSunBldScn	ARRAY OF ST_BA_SunBldScn [▶ 280]	Tabelle mit den Szeneneinträgen

Ausgänge

```
VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  bActv         : BOOL;
  fActlScnPos   : REAL;
  fActlScnAngl  : REAL;
END_VAR
```

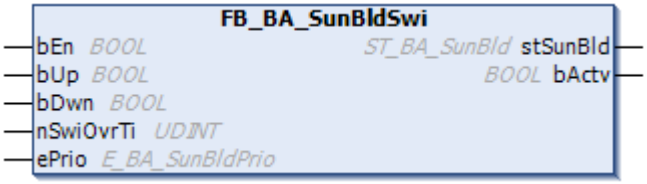
Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.
fActlScnPos	REAL	Zeigt die gespeicherte relative Jalousiehöhenposition [%] der aktuell angewählten Szene an.
fActlScnAngl	REAL	dto. Lammellenwinkel [°]

Wenn ein Fehler ansteht, wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.26 FB_BA_SunBldSwi



Mit Hilfe des Funktionsbausteines FB_BA_SunBldSwi lassen sich die Jalousieansteuerung FB_BA_SunBldActr [▶ 383] bzw. die Rollladensteuerung FB_BA_RolBldActr [▶ 373] im Handbedienmodus ansteuern. Die Verbindung erfolgt dabei über das Positioniertelegramm ST_BA_Sunbld [▶ 279] entweder direkt oder mit einer zusätzlichen Prioritätssteuerung.

Betrieb

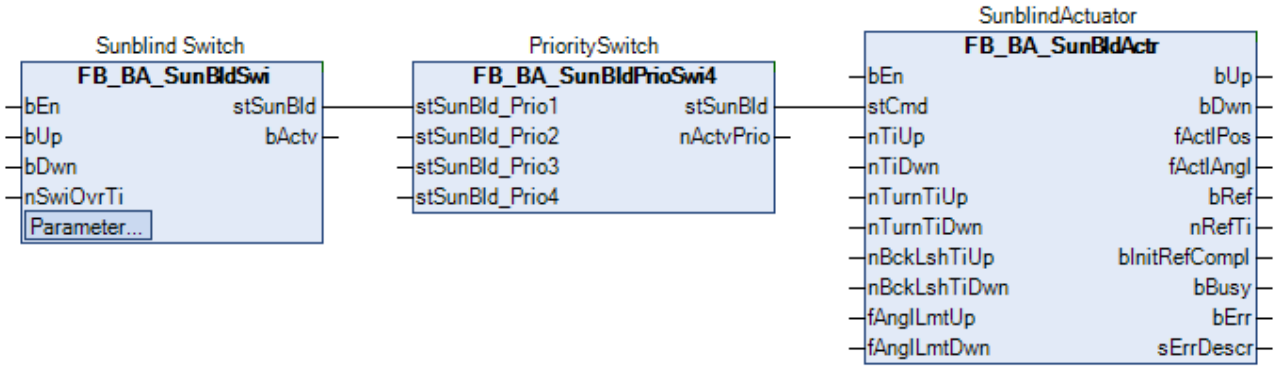
Der Funktionsbaustein steuert über die Befehlseingänge *bUp* und *bDwn* den Jalousiebaustein FB_BA_SunBldActr [▶ 383] bzw. den Rollladenbaustein FB_BA_RolBldActr [▶ 373] im Handbetrieb an, wobei *bUp* Vorrang hat. Die Befehle werden an die jeweiligen Kommandos *bManUp* und *bManDwn* des Positioniertelegramms weitergereicht. Ist ein Befehlseingang länger als die eingetragene Zeit *nSwiOvrTi* [ms] aktiviert, so geht der entsprechende Steuerbefehl in Selbsthaltung. Ein erneutes Aktivieren eines Befehlseinganges löscht diese Selbsthaltung wieder.
Wird der Baustein durch den Eingang *bEn* = TRUE aktiviert, so wird im Positioniertelegramm das Bit *bActv* sofort gesetzt. Damit meldet der Baustein am Prioritätsschalter (siehe …) seinen Vorrang gegenüber niedrigen Prioritäten an. Gleichzeitig wird auch das Bit *bManMod* im Positioniertelegramm gesetzt, um den angeschossenen Aktor Bausteinen zu melden, dass diese auf Handbefehle reagieren sollen.

Bei Deaktivierung des Bausteines durch *bEn* = FALSE werden beide Bits, *bActv* und *bManMod* wieder auf FALSE gesetzt.

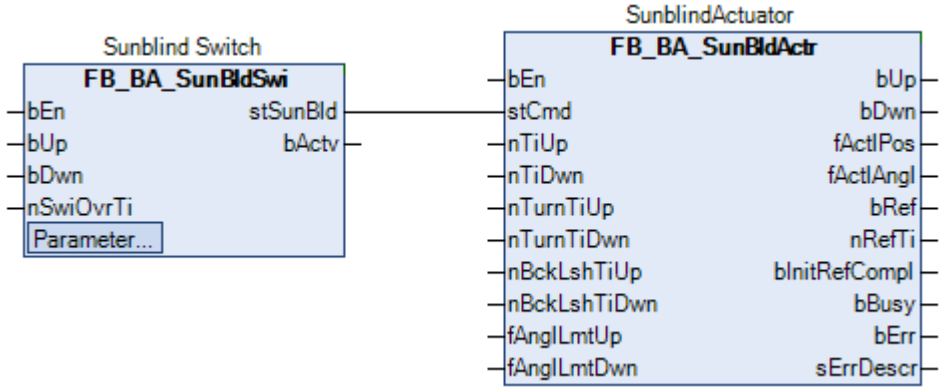
Verknüpfung an Jalousiebaustein

Der Handbedienbaustein kann entweder über eine voran gestellte Prioritätssteuerung FB_BA_... oder aber direkt an den Jalousiebaustein angeschlossen werden. Die Verbindung erfolgt dabei über das Positioniertelegramm ST_BA_Sunbld [▶ 279].

Verwendung einer Prioritätssteuerung:



Direkte Beschaltung:



Eingänge

```
VAR_INPUT
  bEn             : BOOL;
  bUp             : BOOL;
  bDwn            : BOOL;
  nSwiOvrTi       : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_Sunbld [▶ 279] werden für Position und Winkel jeweils 0 ausgegeben - *bManMod* und *bActv* stehen jeweils auf FALSE. Das bedeutet für eine Beschaltung mit Prioritätssteuerung, dass eine andere Funktionalität die Jalousieansteuerung übernimmt. Eine direkte Beschaltung hingegen lässt die Jalousie direkt auf Position 0, also ganz nach oben fahren, da der Aktorbaustein das Bit *bActv* selbst nicht auswertet.
bUp	BOOL	Befehlseingang Jalousie hoch.
bDwn	BOOL	Befehlseingang Jalousie herunter.
nSwiOvrTi	UDINT	Zeit [ms] bis bei dauerhaft aktiviertem Befehlseingang der entsprechende Handbefehl im Positioniertelegramm ST_BA_Sunbld [▶ 279] in Selbsthaltung geht. Intern begrenzt auf einen Minimalwert von 0.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio    : E_BA_SunBldPrio := E_BA_SunBldPrio.eManualActuator;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des aktiven Telegramms.

Ausgänge

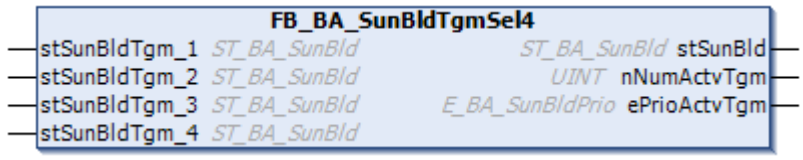
```
VAR_OUTPUT
  stSunBld  : ST_BA_SunBld;
  bActv     : BOOL;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.27 FB_BA_SunBldTgmSel4



Der Funktionsbaustein FB_BA_SunBldTgmSel4 dient zur Prioritätssteuerung für bis zu 4 Positioniertelegramme verschiedener Steuerbausteine (*stSunBld_Prio1 ... stSunBld_Prio4*) des Typs ST_BA_SunBld [▶ 279].

Das aktive Telegramm der höchsten Priorität wird am Ausgang *stSunBld* ausgegeben. Aktiv bedeutet, dass innerhalb der Struktur des Positioniertelegramms die Variable *bActv* gesetzt ist. Die Priorität ist dabei innerhalb der Telegrammstruktur als *ePrio* hinterlegt. Je kleiner der Wert von *ePrio* ist, desto höher ist die Priorität.

Bei Telegrammen gleicher Priorität gilt das zuletzt veränderte (last writer wins), ermittelt durch die Variable *nEvtIncSunBld* in der globalen Variablenliste BA_Globals [▶ 288].

Dieser Baustein ist so zu programmieren, dass immer eines der angelegten Telegramme aktiv ist. Falls ein Telegramm nicht aktiv ist, wird am Ausgang ein Leertelegramm ausgegeben:

```
fPos    = 0,
fAngl   = 0,
bManUp  = FALSE,
bManDwn = FALSE,
bManMod = FALSE,
bActv   = FALSE;
```

Da der Jalousiebaustein FB_BA_SunBldActr [▶ 383] bzw. der Rollladenbaustein FB_BA_RolBldActr [▶ 373] seinerseits nicht auf den Merker *bActv* achtet, würde dieses Telegramm als Fahrbefehl auf Position "0", also vollständig geöffnet, gewertet werden. Das Fehlen eines aktiven Telegramms stellt damit kein Sicherheitsrisiko für die Jalousie dar.

Eingänge

```
VAR_INPUT
  stSunBldTgm_1   : ST_BA_SunBld;
  stSunBldTgm_2   : ST_BA_SunBld;
  stSunBldTgm_3   : ST_BA_SunBld;
  stSunBldTgm_4   : ST_BA_SunBld;
END_VAR
```

Name	Typ	Beschreibung
stSunBldTgm_1… stSunBldTgm_4	ST_BA_SunBld [▶ 279]	Telegrammeingänge

Ausgänge

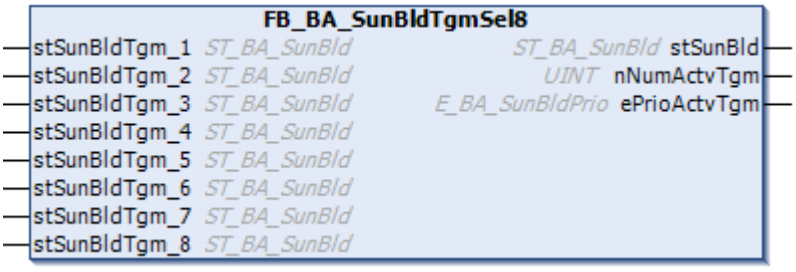
```
VAR_OUTPUT
  stSunBld       : ST_BA_SunBld;
  nNumActvTgm    : UINT;
  ePrioActvTgm   : E_BA_SunBldPrio;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
nNumActvTgm	UINT	Zeigt an, welcher Eingang gültig ist, wenn z.B. *stSunBldTgm_3* durchgereicht wird, ist *nNumActvTgm* = 3. Ist *nNumActvTgm* = 0, so ist kein Telegramm aktiv.
ePrioActvTgm	E_BA_SunBldPrio [▶ 274]	Dieser Ausgang gibt die Priorität des aktiven Telegramms an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.28 FB_BA_SunBldTgmSel8



Der Funktionsbaustein FB_BA_SunBldTgmSel8 dient zur Prioritätssteuerung für bis zu 8 Positioniertelegramme verschiedener Steuerbausteine (*stSunBld_Prio1 ... stSunBld_Prio8*) des Typs ST_BA_SunBld [▶ 279].

Das aktive Telegramm der höchsten Priorität wird am Ausgang *stSunBld* ausgegeben. Aktiv bedeutet, dass innerhalb der Struktur des Positioniertelegramms die Variable *bActv* gesetzt ist. Die Priorität ist dabei innerhalb der Telegrammstruktur als *ePrio* hinterlegt. Je kleiner der Wert von *ePrio* ist, desto höher ist die Priorität.

Bei Telegrammen gleicher Priorität gilt das zuletzt veränderte (last writer wins), ermittelt durch die Zählvariable *nEvtIncLight* in der globalen Variablenliste BA_Globals [▶ 288].

Dieser Baustein ist so zu programmieren, dass immer eines der angelegten Telegramme aktiv ist. Falls ein Telegramm nicht aktiv ist, wird am Ausgang ein Leertelegramm ausgegeben:

```
fPos    = 0,
fAngl   = 0,
bManUp  = FALSE,
bManDwn = FALSE,
bManMod = FALSE,
bActv   = FALSE;
```

Da der Jalousiebaustein FB_BA_SunBldActr [▶ 383] bzw. der Rollladenbaustein FB_BA_RolBldActr [▶ 373] seinerseits nicht auf den Merker *bActv* achtet, würde dieses Telegramm als Fahrbefehl auf Position "0", also vollständig geöffnet, gewertet werden. Das Fehlen eines aktiven Telegramms stellt damit kein Sicherheitsrisiko für die Jalousie dar.

Eingänge

```
VAR_INPUT
  stSunBldTgm_1  : ST_BA_SunBld;
  stSunBldTgm_2  : ST_BA_SunBld;
  stSunBldTgm_3  : ST_BA_SunBld;
  stSunBldTgm_4  : ST_BA_SunBld;
  stSunBldTgm_5  : ST_BA_SunBld;
  stSunBldTgm_6  : ST_BA_SunBld;
  stSunBldTgm_7  : ST_BA_SunBld;
  stSunBldTgm_8  : ST_BA_SunBld;
END_VAR
```

Name	Typ	Beschreibung
stSunBldTgm_1… stSunBldTgm_8	ST_BA_SunBld [▶ 279]	Telegrammeingänge

Ausgänge

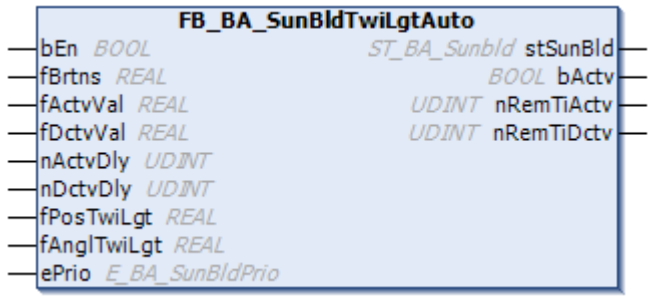
```
VAR_OUTPUT
  stSunBld      : ST_BA_SunBld;
  nNumActvTgm   : UINT;
  ePrioActvTgm  : E_BA_SunBldPrio;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
nNumActvTgm	UINT	Zeigt an, welcher Eingang gültig ist, wenn z.B. *stSunBldTgm_3* durchgereicht wird, ist *nNumActvTgm* = 3. Ist *nNumActvTgm* = 0, so ist kein Telegramm aktiv.
ePrioActvTgm	E_BA_SunBldPrio [▶ 274]	Dieser Ausgang gibt die Priorität des aktiven Telegramms an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.29 FB_BA_SunBldTwiLgtAuto



Der Funktionsbaustein FB_BA_SunBldTwiLgtAuto steuert die Jalousie, wenn die Außenhelligkeit einen Grenzwert unterschritten hat.

Die Dämmerungsautomatik arbeitet mit einer Werte- und einer zeitlichen Hysterese: Unterschreitet der Außenhelligkeitswert *fBrtns* [lx] für die Zeit *nActvDly* [s] den Wert *fActvVal* [lx], so ist der Baustein aktiv und wird die an den Eingangsvariablen angegebenen Jalousiepositionen *fPosTwiLgt* (Höhe [%]) und *fAnglTwiLgt* (Lamellenwinkel [°]) am Ausgang im Positioniertelegramm ST_BA_Sunbld [▶ 279] bereitstellen. Überschreitet die Außenhelligkeit hingegen für die Zeit *nDctvDly* [s] den Wert *fDctvVal* [lx], so ist die Automatik nicht mehr aktiv. Der Aktiv-Merker im Positioniertelegramm ST_BA_Sunbld [▶ 279] wird zurück und die Positionen für Höhe und Winkel auf "0" gesetzt. Eine Funktion niedrigerer Priorität kann dann die Steuerung übernehmen.

Eingänge

```
VAR_INPUT
  bEn             : BOOL;
  fBrtns          : REAL;
  fActvVal        : REAL;
  fDctvVal        : REAL;
  nActvDly        : UDINT;
  nDctvDly        : UDINT;
  fPosTwiLgt      : REAL;
  fAnglTwiLgt     : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_SunBld [▶ 279] werden für Position und Winkel jeweils 0 ausgegeben - *bManMod* und *bActv* stehen jeweils auf FALSE. Das bedeutet, dass über die Prioritätssteuerung eine andere Funktionalität die Jalousieansteuerung übernimmt. Eine direkte Beschaltung hingegen lässt die Jalousie direkt auf Position 0, also ganz nach oben fahren, da der Aktorbaustein das Bit *bActv* selbst nicht auswertet.
fBrtns	REAL	Außenhelligkeit [lx].
fActvVal	REAL	Aktivierungsgrenzwert [lx]. Der Wert *fActvVal* wird intern begrenzt auf Werte von 0 bis *fDctvVal*.
fDctvVal	REAL	Deaktivierungsgrenzwert [lx]. Intern begrenzt auf einen Minimalwert von 0.
nActvDly	UDINT	Aktivierungsverzögerung [s]. Intern begrenzt auf einen Minimalwert von 0.
nDctvDly	UDINT	Deaktivierungsverzögerung [s]. Intern begrenzt auf einen Minimalwert von 0.
fPosTwiLgt	REAL	Höhenposition der Jalousie [%], wenn die Dämmerungsautomatik aktiv ist. Intern begrenzt auf Werte von 0 bis 100.
fAnglTwiLgt	REAL	Lamellenwinkel der Jalousie [°], wenn die Dämmerungsautomatik aktiv ist

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio    : E_BA_SunBldPrio := E_BA_SunBldPrio.eGroupTwiLightAuto;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des aktiven Telegramms.

Ausgänge

```
VAR_OUTPUT
  stSunBld        : ST_BA_SunBld;
  bActv           : BOOL;
```

```
  nRemTiActv        : UDINT;
  nRemTiDctv        : UDINT;
END_VAR
```

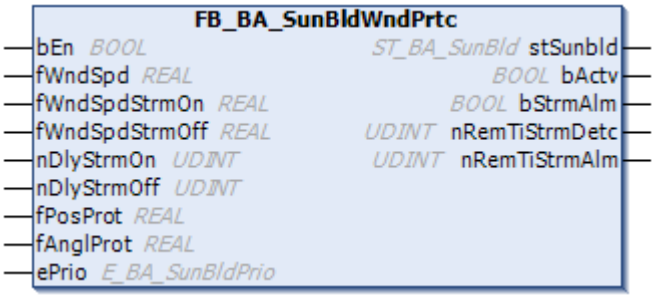
Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.
nRemTiActv	UDINT	Zeigt die verbleibende Zeit an nach Unterschreitung des Schaltwertes *fActvVal* bis zur Aktivierung der Automatik [s] an. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.
nRemTiDctv	UDINT	Zeigt die verbleibende Zeit an nach Überschreitung des Schaltwertes *fDctvVal* bis zur Abschaltung der Automatik [s] an. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.

Wenn ein Fehler ansteht, wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.30 FB_BA_SunBldWndPrtc



Der Funktionsbaustein FB_BA_SunBldWndPrtc behandelt den richtungsabhängigen Windschutz.

Der Witterungsschutz hat bei der Jalousiesteuerung die höchste Priorität (siehe Übersicht [▶ 339]) und soll sicherstellen, dass die Jalousie weder durch Eis noch durch Wind beschädigt wird.

Liegt die gemessene Windgeschwindigkeit für die Zeit *nDlyStrmOn* [s] über dem Wert *fWndSpdStrmOn*, so wird davon ausgegangen, dass ein Sturm unmittelbar bevorsteht. Erst wenn die Windgeschwindigkeit den Wert *fWndSpdStrmOff* für die Zeit *nDlyStrmOff* [s] unterschreitet, gilt der Sturm als abgeflaut und das Fahren der Jalousie als sicher. Aus Sicherheitsgründen wird auch das Sturm-Ereignis persistent gespeichert. Fällt die Steuerung also während eines Sturmes aus, so wird nach Wiederanlauf der Steuerung der Ablauf-Zeitgeber von neuem gestartet.

Bei Windgefahr-Fällen wird die Jalousie in die Schutzposition gefahren die durch *fPosProt* (Höhenposition [%]) und *fAnglProt* (Lamellenwinkel [°]) vorgegeben wird.

Eingänge

```
VAR_INPUT
  bEn               : BOOL;
  fWndSpd           : REAL;
  fWndSpdStrmOn     : REAL;
```

```
  fWndSpdStrmOff     : REAL;
  nDlyStrmOn         : UDINT;
  nDlyStrmOff        : UDINT;
  fPosProt           : REAL;
  fAnglProt          : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_Sunbld [▶ 279] werden für Position und Winkel jeweils 0 ausgegeben - *bManMod* und *bActv* stehen jeweils auf FALSE. Das bedeutet für eine Beschaltung mit Prioritätssteuerung, dass eine andere Funktionalität die Jalousieansteuerung übernimmt.
fWndSpd	REAL	Windgeschwindigkeit. Die Einheit der Eingabe ist beliebig, jedoch ist es wichtig, dass es keine Werte kleiner als 0 gibt, und die Werte mit zunehmender Geschwindigkeit größer werden.
fWndSpdStrmOn	REAL	Windgeschwindigkeits-Grenzwert zur Aktivierung des Sturmalarmes. Dieser Wert darf nicht kleiner als 0 sein und muss oberhalb des Wertes für die Deaktivierung liegen. Anderenfalls wird ein Fehler ausgegeben. Die Einheit der Eingabe muss die gleiche sein wie die des Eingangs *fWndSpd*. Ein Wert größer als dieser Grenzwert löst nach der eingetragenen Zeit *nDlyStrmOn* den Alarm aus.
fWndSpdStrmOff	REAL	Windgeschwindigkeits-Grenzwert zur Deaktivierung des Sturmalarmes. Dieser Wert darf nicht kleiner als 0 sein und muss unterhalb des Wertes für die Aktivierung liegen. Anderenfalls wird ein Fehler ausgegeben. Die Einheit der Eingabe muss die gleiche sein wie die des Einganges *fWndSpd*. Ein Wert kleiner oder gleich diesem Grenzwert setzt nach der eingetragenen Zeit *nDlyStrmOff* den Alarm zurück.
nDlyStrmOn	UDINT	Verzögerungszeit zur Auslösung des Sturmalarmes [s].
nDlyStrmOff	UDINT	Verzögerungszeit zum Rücksetzen des Sturmalarmes [s].
fPosProt	REAL	Höhenposition der Jalousie [%] im Schutzfall.
fAnglProt	REAL	Lamellenwinkel der Jalousie [°] im Schutzfall

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio    : E_BA_SunBldPrio := E_BA_SunBldPrio.eStorm;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des aktiven Telegramms.

Ausgänge

```
VAR_OUTPUT
  stSunBld             : ST_BA_SunBld;
  bActv                : BOOL;
  bStrmAlm             : BOOL;
  nRemTiStrmDetc       : UDINT;
  nRemTiStrmAlm        : UDINT;
END_VAR
```

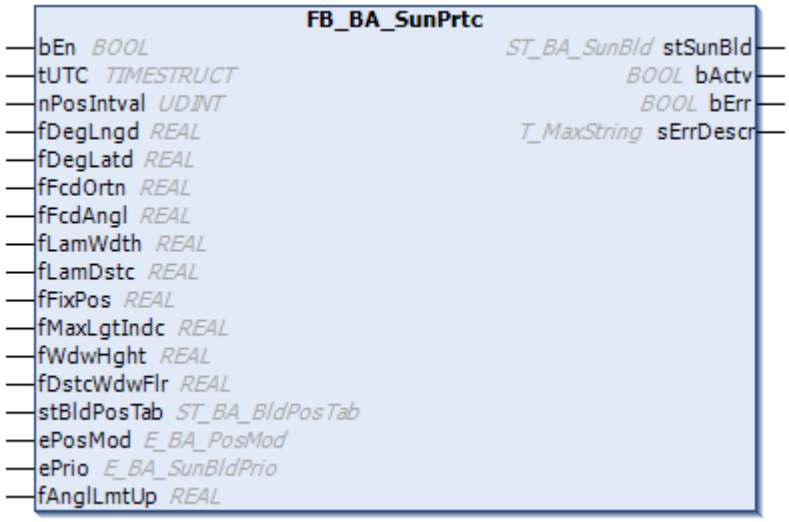
Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.
bStrmAlm	BOOL	Zeigt den Sturmalarm an
nRemTiStrmDetc	UDINT	Im unkritischen Fall zeigt dieser Sekundenzähler konstant die Alarmverzögerungszeit *nDlyStrmOn* an. Liegt die gemessene Windstärke *fWndSpd* über dem Aktivierungsgrenzwert *fWndSpdStrmOn*, so werden die Sekunden bis zum Alarm heruntergezählt. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.
nRemTiStrmAlm	UDINT	Sobald der Sturmalarm ausgelöst wird, zeigt dieser Sekundenzähler zunächst konstant die Deaktivierungsverzögerungszeit *nDlyStrmOff* des Sturmalarmes an. Sinkt die gemessene Windstärke *fWndSpd* unter den Deaktivierungsgrenzwert *fWndSpdStrmOff*, so werden die Sekunden bis zur Entwarnung (*bStrmAlm*=FALSE) heruntergezählt. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0

Wenn ein Fehler ansteht, wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.3.3.31 FB_BA_SunPrtc



Der Funktionsbaustein FB_BA_SunPrtc dient zur Blendschutz Steuerung mit Hilfe einer Lamellen-Jalousie.

Der Blendschutz wird durch die Variation des Lamellenwinkels und durch die Position der Jalousiehöhe realisiert.

Der Lamellenwinkel wird in Abhängigkeit des Sonnenstandes so eingestellt, dass direkte Blendung unterbunden, aber trotzdem ein maximaler Einfall natürlichen Lichts möglich ist.

Für das Variieren der Jalousiehöhe stehen drei verschieden Betriebsarten zur Verfügung.

1. Die Jalousie fährt bei aktiven Sonnenschutz in eine fest vorgegebene Höhe. Der Wert für die Höhe wird mit der Variablen *fFixPos* vorgegeben.
2. Die Position der Jalousie wird in Abhängigkeit des Sonnenstandes variiert. Die Position wird in der Tabelle (ST_BA_BldPosTab [▶ 278]) vorgegeben. Siehe auch Beschreibung des FB_BA_BldPosEntry [▶ 572].
3. Die Höhe der Jalousie wird anhand der Fenstergeometrie so berechnet, dass die Sonnenstrahlen bis zu einer vorgegebenen Tiefe in den Raum einfallen. Die Einfalltiefe der Sonnenstrahlen wird mit der Variablen *fMaxLgtIndc* definiert.

Damit das Neupositionieren des Lamellenwinkels nicht zu häufig erfolgt, kann mittels der Variablen *nPosIntval* [Min] ein Zeitintervall bestimmt werden, innerhalb dessen keine Anpassung des Lamellenwinkels erfolgt. Um trotzdem eine Blendwirkung auszuschließen, wird der Winkel immer weiter geändert, dass er bis zum Ablauf des Zeitintervalls ausreicht.

Für die Positionierung der Jalousie und der Einstellung des Lamellenwinkels müssen die folgenden Bedingungen erfüllt sein.

- 1. Der Eingang *bEn* muss TRUE sein.
- 2. Die Sonne muss aufgegangen sein. (Elevation > 0)
- 3. Der Baustein ist richtig parametriert (*bErr* = False)

Eingänge

```
VAR_INPUT
  bEn               : BOOL;
  tUTC              : TIMESTRUCT;
  nPosIntval        : UDINT;
  fDegLngd          : REAL;
  fDegLatd          : REAL;
  fFcdOrtn          : REAL;
  fFcdAngl          : REAL;
  fLamWdth          : REAL;
  fLamDstc          : REAL;
  fFixPos           : REAL;
  fMaxLgtIndc       : REAL;
  fWdwHght          : REAL;
  fDstcWdwFlr       : REAL;
  stBldPosTab       : ST_BA_BldPosTab;
  ePosMod           : E_BA_PosMod := E_BA_PosMod.eFix;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Wenn dieser Eingang auf FALSE gesetzt wird, ist die Positionierung nicht aktiv, d.h. in der Positionier-Struktur *stSunBld* vom Typ ST_BA_SunBld [▶ 279] wird das aktiv-Bit (*bActv*) zurückgesetzt und der Baustein selbst verharrt in einem Stillstands-Modus. Ist der Baustein hingegen aktiviert, so ist das aktiv-Bit auf TRUE und der Baustein gibt in der Positionierstruktur zu den entsprechenden Zeiten seine Stellwerte durch (*fPos*, *fAngl*).
stUTC	TIMESTRUCT	Eingabe der aktuellen Uhrzeit als koordinierte Weltzeit (UTC, Coordinates Universal Time - im älteren Sprachgebrauch auch GMT, Greenwich-Mean-Time) (siehe TIMESTRUCT). Mit Hilfe des Bausteines FB_BA_GetTime [▶ 631] kann diese Zeit aus einem Zielsystem gelesen werden.

Ein Zeitrücksprung von mehr als 300 s führt, wenn die Jalousie nach den o.a. Kriterien in der Sonne steht und der Blendschutz aktiv ist, zu einer sofortigen Neupositionierung. Diese Funktionalität ist eingefügt worden, um einen nachvollziehbaren Programmablauf zu gewährleisten.

Name	Typ	Beschreibung
nPosIntval	UDINT	Positionierintervall in Minuten - Zeitspanne zwischen zwei Ausgaben von Jalousiestellungen. Gültiger Bereich: 1 min...720 min.
fDegLngd	REAL	Geographische Länge (Längengrad) [°]. Gültiger Bereich: -180°...180°.
fDegLatd	REAL	Geographische Breite (Breitengrad) [°]. Gültiger Bereich: -90°...90°.
fFcdOrtn	REAL	Fassadenausrichtung [°].

Dabei gilt auf der Nordhalbkugel für die Fassadenausrichtung (Blick aus dem Fenster):

Blickrichtung	Fassadenausrichtung
Nord	β=0°
Ost	β=90°
Süd	β=180°
West	β=270°

Für die Südhalbkugel gilt hingegen:

Blickrichtung	Fassadenausrichtung
Süd	β=0°
Ost	β=90°
Nord	β=180°
West	β=270°

Name	Typ	Beschreibung
fFcdAngl	REAL	Fassadenneigung [°] (siehe Fassadenneigung [▶ 343]).
fLamWdth	REAL	Breite der Lamellen in mm (siehe Skizze [▶ 340]).
fLamDstc	REAL	Lamellenabstand in mm (siehe Skizze [▶ 340]).
fFixPos	REAL	Fixe (konstante) Jalousiehöhe [0…100 %]. Gültig, wenn *ePosMod* = *ePosModFix* (siehe E_BA_PosMod [▶ 698]).
fMaxLgtIndc	REAL	Maximal gewünschter Lichteinfall in mm gemessen ab Außenseite der Wand (siehe Höhenverstellung [▶ 569]). Mit Hilfe der Parameter *fWdwHght* und *fDstcWdwFlr* wird in Abhängigkeit des Sonnenstandes errechnet, wie hoch die Jalousie stehen muss, damit der Lichteinfall den Wert *fMaxLgtIndc* nicht überschreitet. Gültig, wenn *ePosMod* = *ePosModeMaxIncidence* (siehe E_BA_PosMod).
fWdwHght	REAL	Fensterhöhe in mm zur Errechnung der Jalousiehöhe, wenn der Modus "Maximal gewünschter Lichteinfall" gewählt ist.
fDstcWdwFlr	REAL	Abstand Boden - Fenstersims in mm zur Errechnung des der Jalousiehöhe, wenn der Modus "Maximal gewünschter Lichteinfall" gewählt ist.
stBldPosTab	ST_BA_BldPosTab [▶ 278]	Tabelle von 6 Stützpunkten, davon 4 parametrierbar, aus denen dann durch lineare Interpolation eine Jalousieposition in Abhängigkeit des Sonnenstandes gegeben wird. Gültig, wenn *ePosMod* = *ePosModFix* (siehe E_BA_PosMod). Weitere Beschreibung siehe FB_BA_BldPosEntry [▶ 572].
ePosMod	E_BA_PosMod	Auswahl des Positioniermodus

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  ePrio        : E_BA_SunBldPrio := E_BA_SunBldPrio.eSunProtection;
  fAnglLmtUp   : REAL;
END_VAR
```

Name	Typ	Beschreibung
ePrio	E_BA_SunBldPrio [▶ 274]	Priorität des aktiven Telegramms.
fAnglLmtUp	REAL	Lamellen müssen nicht über 0° geöffnet werden.

Ausgänge

```
VAR_OUTPUT
  stSunBld     : ST_BA_SunBld;
  bActv        : BOOL;
  bErr         : BOOL;
  sErrorDescr  : T_MAXSTRING;
END_VAR
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Ausgabetelegramm für die Position und den Winkel der Lamelle.
bActv	BOOL	Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 279] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.
bErr	BOOL	Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Die Dauer des Positionierintervalls ist kleiner oder gleich Null oder überschreitet 720 min.
02: Fehler: Der eingetragene Längengrad ist nicht im gültigen Bereich von -180°…180°.
03: Fehler: Der eingetragene Breitengrad ist nicht im gültigen Bereich von -90°…90°.
04: Fehler: Der eingetragene Wert für die Fassadenneigung *fFcdAngl* ist außerhalb des gültigen Bereiches von -90°…90°.
05: Fehler: Der Wert für den Lamellenabstand (*fLamDstc*) ist größer oder gleich dem Wert für die Lamellenbreite (*fLamWdth*). Dies stellt keine "gültige" Jalousie dar, da die Lamellen nicht vollständig schließen könnten. Mathematisch gesehen würde dieses zu Fehlern führen.
06: Fehler: Der eingetragene Wert für die Lamellenbreite *fLamWdth* ist gleich Null.
07: Fehler: Der eingetragene Wert für den Lamellenabstand *fLamDstc* ist gleich Null.
08: Fehler: Der eingetragene Wert für die fixe Jalousiehöhe (*fFixPos*) ist größer als 100 oder kleiner als 0. Gleichzeitig ist Positionierung "fixe Jalousiehöhe" angewählt – *ePosMod* = ePosModFix.
09: Fehler: Das Bit "Werte gültig" (*bVld*) in der Positioniertabelle *stBldPosTab* ist nicht gesetzt - ungültige Werte: siehe FB_BA_BldPosEntry. Gleichzeitig ist Positionierung "Tabelle" angewählt – *ePosMod* = ePosModTab.
10: Fehler: Der eingetragene Wert für den maximal gewünschten Lichteinfall *fMaxLgtIndc* ist kleiner oder gleich Null. Gleichzeitig ist Positionierung "maximaler Lichteinfall" angewählt – *ePosMod* = ePosModMaxIndc.
11: Fehler: Der eingetragene Wert für die Fensterhöhe *fWdwHght* ist kleiner oder gleich Null. Gleichzeitig ist Positionierung "maximaler Lichteinfall" angewählt – *ePosMod* = ePosModMaxIndc.
12: Fehler: Der eingetragene Abstand Fensterunterkante zu Boden *fDstcWdwFlr* ist kleiner Null. Gleichzeitig ist Positionierung "maximaler Lichteinfall" angewählt – *ePosMod* = ePosModMaxIndc.
13: Fehler: nicht gültiger Positioniermodus am Eingang *ePosMod* eingetragen.

Wenn ein Fehler ansteht, wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.4 System

6.1.2.2.3.1.4.1 FB_BA_CnvtTiSt



Mit dem Funktionsbaustein FB_BA_CnvtTiSt können die einzelnen Bestandteile einer Zeitstruktur zu einer Struktur zusammengefasst werden.

> Der Baustein verfügt über keine Überprüfung fehlerhafter Eingaben, wie z.B. eine Stundeneingabe von 99. Diese Überprüfung erfolgt sinnvollerweise in den angeschlossenen Bausteinen, die ihrerseits die Zeitstruktur ohnehin überprüfen müssen. Die zulässigen Grenzen werden hier jedoch bei der Variablenerläuterung aufgezeigt.

Eingänge

```
VAR_INPUT
  nYear         : WORD;
  nMonth        : WORD;
  nDay          : WORD;
  nHour         : WORD;
  nMinute       : WORD;
  nSecond       : WORD;
  nMilliseconds : WORD;
END_VAR
```

Name	Typ	Beschreibung
nYear	WORD	Das Jahr (1970…2106).
nMonth	WORD	Der Monat (1…12).
nDay	WORD	Der Tag im Monat (1…31).
nHour	WORD	Die Stunde (0…23).
nMinute	WORD	Die Minute (0…59).
nSecond	WORD	Die Sekunde (0…59).
nMilliseconds	WORD	Die Millisekunde (0…999).

Ausgänge

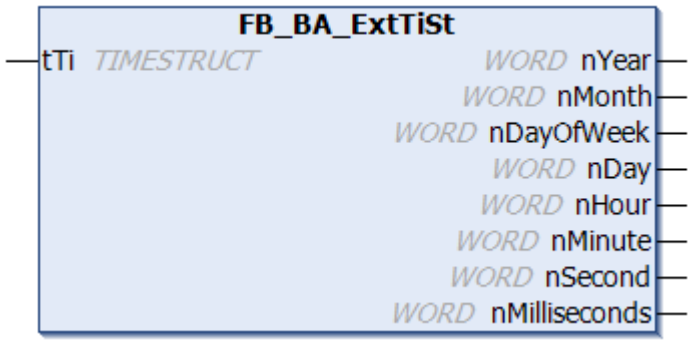
```
VAR_OUTPUT
  bNewData  : BOOL;
  tTi       : TIMESTRUCT;
END_VAR
```

Name	Typ	Beschreibung
bNewData	BOOL	Der Ausgang ist in dem Zyklus TRUE, in dem sich die Eingangsvariablen verändert haben.
tTi	TIMESTRUCT	Ausgabe Zeitstruktur

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.4.2 FB_BA_ExtTiSt



Der Funktionsbaustein FB_BA_ExtTiSt löst eine Zeitstruktur in die verschiedenen Bestandteile auf, um sie so etwa für Zeitbedingungen nutzbar zu machen.

Eingänge

```
VAR_INPUT
  tTi      : TIMESTRUCT;
END_VAR
```

Name	Typ	Beschreibung
tTi	TIMESTRUCT	Eingabe Zeitstruktur

VAR_OUTPUT

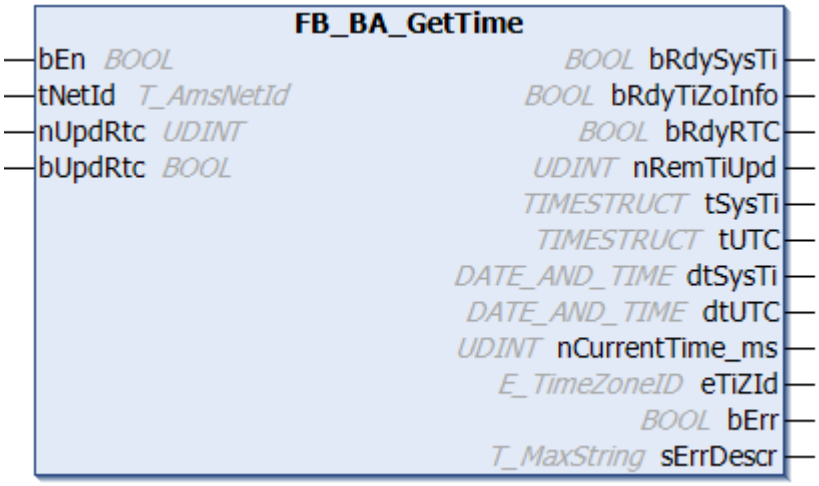
```
VAR_OUTPUT
  nYear          : WORD;
  nMonth         : WORD;
  nDayOfWeek     : WORD;
  nDay           : WORD;
  nHour          : WORD;
  nMinute        : WORD;
  nSecond        : WORD;
  nMilliseconds  : WORD;
END_VAR
```

Name	Typ	Beschreibung
nYear	WORD	Das Jahr (1970…2106).
nMonth	WORD	Der Monat (1…12).
nDayOfWeek	WORD	Der Wochentag (0 (So)…0 (Sa)).
nDay	WORD	Der Tag im Monat (1…31).
nHour	WORD	Die Stunde (0…23).
nMinute	WORD	Die Minute (0…59).
nSecond	WORD	Die Sekunde (0…59).
nMilliseconds	WORD	Die Millisekunde (0…999).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.4.3 FB_BA_GetTime



Mit dem Funktionsbaustein FB_BA_GetTime kann eine interne Uhr (Real Time Clock RTC) in der TwinCAT SPS realisiert werden. Die RTC-Uhr wird mit der Freigabe des Funktionsbausteines über *bEn* mit der aktuellen NT-Systemzeit initialisiert. Es wird ein Systemtakt der CPU benutzt, um die aktuelle RTC-Zeit zu berechnen. Der Funktionsbaustein muss in jedem Zyklus der SPS einmal aufgerufen werden, damit die aktuelle Zeit berechnet werden kann. Intern wird in dem Funktionsbaustein eine Instanz der Funktionsbausteine NT_GetTime, FB_GetTimeZoneInformation und RTC_EX2 aufgerufen. Die Ausgabe der Zeit erfolgt an den Ausgängen *tSysTi* für die gelesene Systemzeit und *tUtc* für die koordinierte Weltzeit (Coordinated Universal Time). Diese wird intern aus der Systemzeit und der Zeitzone ermittelt. Ist die Systemzeit und/oder die Zeitzone fehlerhaft eingegeben worden, so wird auch die UTC-Zeit nicht richtig sein.

Die Systemzeit wird über den einzustellenden Timer *nUpdRTC* [s] zyklisch ausgelesen und damit die interne RTC-Uhr synchronisiert. In demselben Zyklus werden auch die Zeitinformationen (Zeitzone, Zeitverschiebung zu UTC und Sommer-/ Winterzeit) ausgelesen. Der Ausgang *nRemTiUpd* zeigt die verbleibenden Sekunden bis zum nächsten Lesezyklus. Die ausgegebenen Zeitstrukturen *dtSysTi* und *dtUtc* können mit Hilfe des Bausteines FB_BA_ExtTiSt weiter in die Bestandteile - Tag, Monat, Stunde, Minute etc. - aufgelöst werden.

Information zum Lese-/Wartezyklus

Während des Lesezyklus fallen die Ausgänge *bRdySysTi* und *bRdyTiZoInfo* auf FALSE, der Enumerator *eTiZId* zeigt 0 = *eTimeZoneID_Unknown*. Wurde erfolgreich gelesen, so gehen die Ausgänge wieder auf TRUE bzw. zeigen die jeweilige Information Sommer- oder Winterzeit an, sofern sie verfügbar ist. War das Lesen nicht erfolgreich - es wird intern 5s lang auf eine Antwort gewartet - so bleiben die Ausgänge auf FALSE bzw. 0 stehen und ein erneuter Wartezyklus bis zum nächsten Lesezyklus wird gestartet. Im Fehlerfall wird die interne RTC-Uhr zwar nicht synchronisiert und kann immer noch eine richtige Zeit anzeigen, die Zeitinformationen jedoch können fehlerhaft sein und damit auch die UTC-Zeit. Fehler während des Lesezyklus schlagen sich in jedem Fall in der Anzeige an *bErr* und *sErrDescr* nieder. Der Countdown-Ausgang *nRemTiUpd* wird erst dann neu gestartet, wenn der Wartezyklus beginnt.

Eingänge

```
VAR_INPUT
  bEn             : BOOL;
  tNetId          : T_AmsNetId;;
  nUpdRtc         : UDINT;
  bUpdRtc         : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteins. Ist *bEn* = TRUE, so wird die RTC-Uhr mit der NT-Systemzeit initialisiert.
tNetId	T_AmsNetID	Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden dessen NT-Systemzeit als Zeitbasis gelesen werden soll. Für den lokalen Rechner kann auch ein Leerstring angegeben werden.
nUpdRtc	UDINT	Zeitangabe [s], mit der die RTC-Uhr mit der NT-Systemzeit regelmäßig synchronisiert wird. Dieser Wert wird intern auf ein Minimum von 5s begrenzt, um die Abarbeitung der internen Bausteine zu gewährleisten.
bUpdRtc	BOOL	Parallel zu der Zeitangabe *nUpdRtc* kann die RTC-Uhr über eine positive Flanke an diesem Eingang synchronisiert werden

Ausgänge

```
VAR_OUTPUT
  bRdySysTi          : BOOL;
  bRdyTiZoInfo       : BOOL;
  bRdyRTC            : BOOL;
  nRemTiUpd          : UDINT;
  tSysTi             : TIMESTRUCT;
  tUTC               : TIMESTRUCT;
  dtSysTi            : DT;
  dtUTC              : DT;
  nCurrentTime_ms    : UDINT
  eTiZId             : E_TimeZoneID;
  bErr               : BOOL;
  sErrDescr          : T_MAXSTRING;
END_VAR
```

Name	Typ	Beschreibung
bRdySysTi	BOOL	Die Systemzeit wurde erfolgreich aus dem Zielsystem gelesen.
bRdyTiZoInfo	BOOL	Die zusätzlichen Zeitinformationen (Zeitzone, Zeitverschiebung zu UTC und Sommer-/ Winterzeit) wurden erfolgreich gelesen.
bRdyRTC	BOOL	Wurde der Funktionsbaustein mindestens einmal initialisiert, so wird dieser Ausgang gesetzt. Ist dieser Ausgang gesetzt, dann sind die Werte für das Datum, Uhrzeit und Millisekunden an den Ausgängen gültig.
nRemTiUpd	UDINT	Countdown zur nächsten Synchronisation bzw. Aktualisierung der Zeitinformationen.
tSysTi	TIMESTRUCT	Systemzeit des ausgelesenen Zielsystems. Die Zeitstruktur kann mit Hilfe des Bausteines FB_BA_ExtTiSt weiter in die Bestandteile - Tag, Monat, Stunde, Minute etc. - aufgelöst werden. **Info**: Ist der Baustein nicht aktiviert (*bEn* = FALSE), so zeigt der Ausgang *stSysTi* in seinen Teilelementen (Tag Monat, etc.) jeweils 0 an.
tUTC	TIMESTRUCT	Koordinierte Weltzeit. Diese wird intern aus der Systemzeit und den gelesenen Zeitinformationen des Zielsystems ermittelt. Die Zeitstruktur kann mit Hilfe des Bausteines FB_BA_ExtTiSt weiter in die Bestandteile - Tag, Monat, Stunde, Minute etc. - aufgelöst werden. **Info**: Ist der Baustein nicht aktiviert (*bEn*=FALSE), so zeigt der Ausgang *tUTC* in seinen Teilelementen (Tag Monat, etc.) jeweils 0 an.
dtSysTi / dtUTC	DT	Wie *stSysTi* / *stUTC*, jedoch im DATE-AND-TIME-Format: Jahr-Monat-Tag-Stunden-Minuten-Sekunden. **Info**: Ist der Baustein nicht aktiviert (*bEn* = FALSE), so zeigen die Ausgänge *dtSysTi* und *dtUTC* jeweils DT#1970-01-01-00:00 an, da dies die Untergrenze ist und es den Nullen in der Strukturdarstellung von *dtSysTi* / *dtUTC* entspricht.
nCurrentTime	UDINT	Aktuelle Tageszeit [ms]
eTiZId	E_TimeZoneID	Enumerator zur Sommer-/ Winterzeitinformation.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Warnung: ADS-Fehler beim Lesen der Zeit (NT_GetTime). Es wird die ADS-Fehlernummer angegeben.
02: Warnung: ADS-Fehler beim Lesen der Zeitzoneninformation (FB_GetTimeZoneInformation). Es wird die ADS-Fehlernummer angegeben.

Voraussetzungen

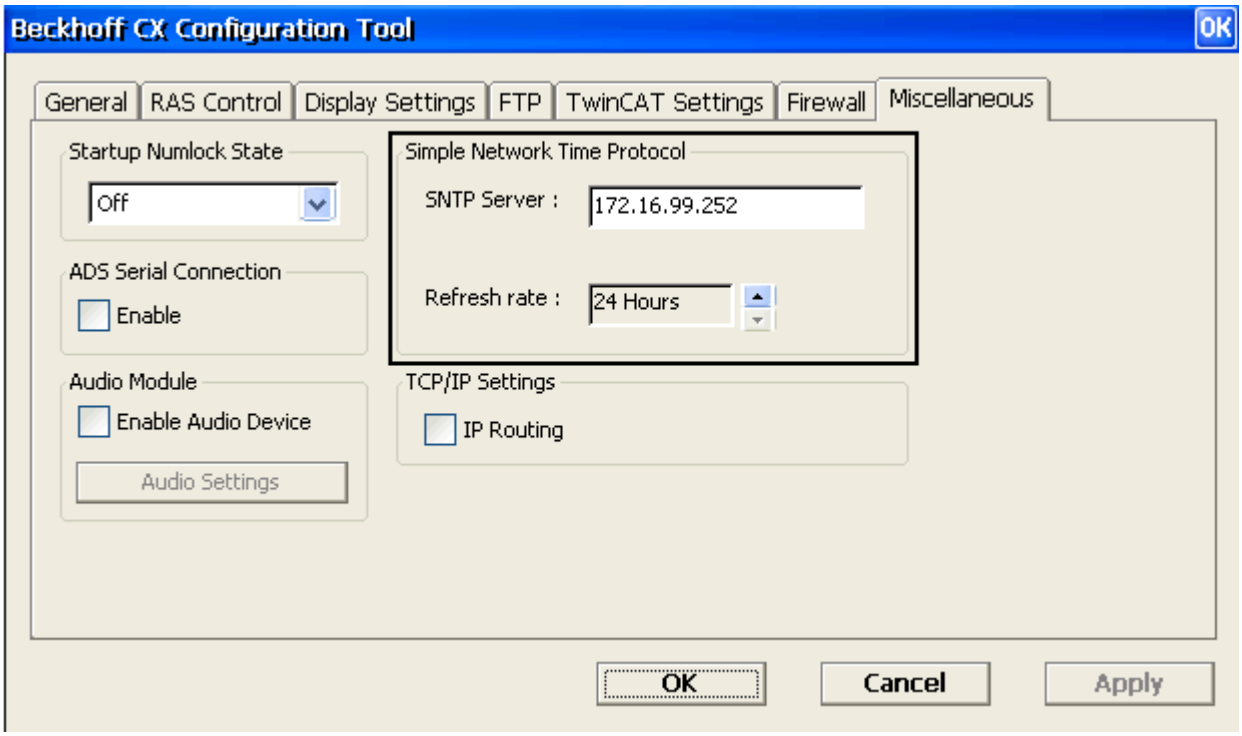
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.4.4 FB_BA_SetTime



Mit dem Funktionsbaustein FB_BA_SetTime kann die lokale NT-Systemzeit und das Datum eines TwinCAT-Systems gesetzt werden (die lokale NT-Systemzeit wird in der Taskleiste eingeblendet). Die Systemzeit wird über die Struktur *tSysTi* vorgegeben.
Intern wird in dem Funktionsbaustein eine Instanz des Funktionsbausteins NT_SetLocalTime aus der TcUtilities-Bibliothek aufgerufen.

Die lokale NT-Systemzeit können Sie auch mit der Hilfe des SNTP-Protokolls mit einer Referenzzeit synchronisieren. Mehr Informationen dazu finden Sie im Beckhoff Information System unter: Beckhoff Information System > Embedded-PC > Betriebssysteme > CE > SNTP: Simple Network Time Protocol



Eingänge

```
VAR_INPUT
  bSet          : BOOL;
  tNetId        : T_AmsNetId;
  tSysTi        : TIMESTRUCT;
  nTiOut        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bSet	BOOL	Aktivierung des Funktionsbausteins mit einer steigenden Flanke.
tNetId	T_AmsNetID	Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden dessen lokale NT-Systemzeit gesetzt werden soll. Für den lokalen Rechner kann auch ein Leerstring *tNetId* := ''; angegeben werden.
tSysTi	TIMESTRUCT	Struktur mit der neuen lokalen NT-Systemzeit. Ist die Zeit als Struktur nicht vorhanden, so empfiehlt sich die Verwendung des Bausteines FB_BA_CnvtTiSt [▶ 410], welcher die Teilvariablen von Datum und Uhrzeit in einer Struktur zusammenführt.
nTiOut	UDINT	Gibt die Timeout-Zeit [s] an, die bei der Ausführung nicht überschritten werden darf.

Ausgänge

```
VAR_OUTPUT
  bBusy           : BOOL;
  bErr            : BOOL;
  sErrDescr       : T_MAXSTRING;
END_VAR
```

Name	Typ	Beschreibung
bBusy	BOOL	Bei der Aktivierung des Funktionsbausteins über eine steigende Flanke an *bSet* wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn entweder die zu übertragene Systemzeit fehlerhaft ist oder aber ein ADS-Fehler in der Übertragung stattfindet.
sErrDescr	T_MAXSTRING	Enthält die Fehlerbeschreibung

Fehlerbeschreibung
01: Fehler: Fehler Bereichsüberschreitung Jahr
02: Fehler: Fehler Bereichsüberschreitung Monat
03: Fehler: Fehler Bereichsüberschreitung Monatstag
04: Fehler: Fehler Bereichsüberschreitung Stunde
05: Fehler: Fehler Bereichsüberschreitung Minute
06: Fehler: Fehler Bereichsüberschreitung Sekunde
07: Fehler: Fehler Bereichsüberschreitung Millisekunde
08: Warnung: Ein ADS-Fehler ist während des Setzens der Zeit aufgetreten (NT_SetLocalTime). Es wird die ADS-Fehlernummer angegeben.

Grenzen der Zeitvorgabe

Die angelegte Zeitstruktur *stSysTi* wird bausteinintern auf Grenzen geprüft (siehe TIMESTRUCT).

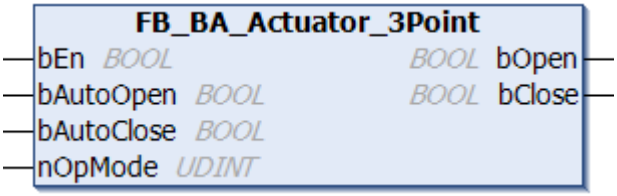
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5 Universal

6.1.2.2.3.1.5.1 Actuators

6.1.2.2.3.1.5.1.1 FB_BA_Actuator_3Point



Der Funktionsbaustein FB_BA_Actuator_3Point dient der Steuerung eines 3-Punkt-Aktuators z. B. einer 3-Punktklappe oder eines 3-Punkt-Ventils.

Am Ausgang *bOpen* wird der Befehl zum Auffahren des Aktors angeschlossen.

Am Ausgang *bClose* wird der Befehl zum Zufahren des Aktors angeschlossen.

Im Automatikbetrieb (*nOpMode* = 0) werden die Stellbefehle von *bCmdOpen* und *bCmdClose* direkt an die Ausgänge *bOpen* und *bClose* weitergeleitet.

Mit Hilfe des Eingangs *nOpMode* wird die Betriebsart des 3-Punkt-Aktors bestimmt:

- 0 = Automatik
- 1 = Stop (bOpen = bClose = FALSE)
- 2 = Zu
- 3 = Auf

Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  bAutoOpen    : BOOL;
  bAutoClose   : BOOL;
  nOpMode      : UDINT
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins.
bAutoOpen	BOOL	Befehl zum Auffahren des Aktors
bAutoClose	BOOL	Befehl zum Zufahren des Aktors.
nOpMode	UDINT	Auswahl der Betriebsart (0 = Automatik, 1 = Stopp (bOpen = bClose = FALSE), 2 = Zu, 3 = Auf).

Ausgänge

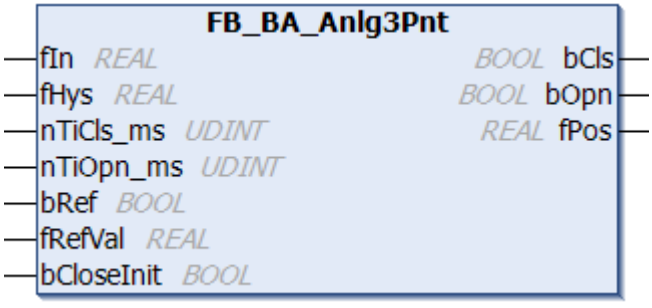
```
VAR_OUTPUT
  bOpen   : BOOL;
  bClose  : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bOpen	BOOL	Steuerausgang öffnen
bClose	BOOL	Steuerausgang schließen

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.1.2 FB_BA_Anlg3Pnt



Der Funktionsbaustein FB_BA_Anlg3Pnt ist für die Ansteuerung von Dreipunktstellantrieben zur Betätigung von Ventilen oder Klappen vorgesehen. Ein stetiges Stellsignal für die Positionierung eines Stellantriebs wird in die binären Befehle für das Auf- und Zufahren umgewandelt.
Ist die Abweichung zwischen dem Positionssollwert *fIn* und dem errechneten Positionsistwert *fPos* des Stellorgans größer als der eingestellte Schwellwert *fHys / 2*, dann beginnt der Funktionsbaustein abhängig von dem Betrag der Regelabweichung durch Schalten der Ausgänge *bOpn* oder *bCls* die Position zu korrigieren:

	bOpn	bCls
fIn - fPos > fHys / 2	TRUE	FALSE
fIn - fPos < - fHys / 2	FALSE	TRUE

Erreicht der Baustein durch einen entsprechenden Eingangswert *fIn* eine Endlage *fOut* = 0 oder *fOut* = 100, so bleibt der entsprechende Schaltausgang dauerhaft gesetzt, um diese Endlage auch am Ventil bzw. der Klappe sicher zu erreichen:

	bOpn	bCls
fOut = 0	FALSE	dauerhaft TRUE
fOut = 100	dauerhaft TRUE	FALSE

Eine eventuelle Abschaltung des Dauersignals muss durch äußere Programmierung des Anwenders erfolgen.

Die Eingabe *fIn* wird intern automatisch auf den Bereich von 0…100 % begrenzt.
Dies gilt ebenfalls für die Eingaben *fHys* und *fRefVal*. Die Verfahrzeiten *nTiCls* sowie *nTiOpn* sind beide nach unten hin auf 10 (Millisekunden) begrenzt.
Eine steigende Flanke an *bRef* löst einen Referenzierbefehl (Setzen der errechneten Istposition auf *fRefVal*) aus.
Falls der Antrieb über Endlagenschalter verfügt, können diese auch direkt mittels Digitaleingang erfasst und zur Referenzierung an *bRef* verwendet werden.

Eingänge

```
VAR_INPUT
  fIn          : REAL;
  fHys         : REAL;
  nTiCls       : UDINT;
  nTiOpn       : UDINT;
  bRef         : BOOL;
  fRefVal      : REAL;
  bCloseInit   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fIn	REAL	Sollwert für die Position des Stellantriebs [0…100 %]. Intern begrenzt auf Werte von 0 bis 100.
fHys	REAL	Hysterese für die Position des Stellantriebs [0…100 %]. Intern begrenzt auf Werte von 0 bis 100.
nTiCls	UDINT	Fahrzeit des Stellantriebs von auf nach zu [ms]. Intern begrenzt auf Werte von 0 bis 100.
nTiOpn	UDINT	Fahrzeit des Stellantriebs von zu nach auf [ms]. Intern begrenzt auf Werte von 0 bis 100.
bRef	BOOL	Flanke referenziert den internen Positionsspeicher des Antriebs auf Wert von *fRefVal* [0…100 %].
fRefVal	REAL	Wert für die Referenzierung des Stellantrieb mit *bRef* [0…100 %]. Intern begrenzt auf Werte von 0 bis 100.
bCloseInit	BOOL	Wenn dieser Eingang TRUE ist, dann ist der Ausgang *bCls* für die Zeit udiTiOpn_ms TRUE.

Ausgänge

```
VAR_OUTPUT
  bCls     : BOOL;
  bOpn     : BOOL;
  fPos     : REAL;
END_VAR
```

Name	Typ	Beschreibung
bCls	BOOL	Ausgang für das Zufahren des Stellantriebs.
bOpn	BOOL	Ausgang für das Auffahren des Stellantriebs.
fPos	REAL	aktuelle errechnete Position des Stellantriebs [0…100 %].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.1.3 FB_BA_AntBlkg



Der Funktionsbaustein FB_BA_AntBlkg verhindert das Blockieren von Pumpen oder Stellantrieben nach längeren Zeiten ohne Bewegung durch die Ausgabe eines Einschaltimpulses.
Eine Impulsausgabe erfolgt generell nur, wenn der Funktionsbaustein an *bEn* frei geschaltet ist.

Die maximale Dauer des Stillstands bis zur Ausgabe eines Impulses wird durch den Wert der Variable *nOffMin* bestimmt. Zur Erfassung der Stillstandzeit muss der Eingang *bFdb* mit der Betriebsrückmeldung des Aggregates verbunden werden. Die Länge des Impulses wird mit der Variablen *nImplLngt* parametriert. Für diese Funktion muss die Betriebsart E_BA_AntBlkgMode.eOffTime eingestellt sein.

Falls der Blockierschutz-Impuls nicht in Abhängigkeit der Stillstandszeit, sondern zyklisch von einem Zeitschaltplan ausgehen sollen, ist hierfür der Eingang *bExtReq* zu verwenden. Bei einer steigenden Flanke an *bExtReq* wird unmittelbar ein Impuls an *bQ* ausgegeben. Für diese Funktion muss die Betriebsart E_BA_AntBlkgMode.eExternalRequest eingestellt sein.

Eingänge

```
VAR_INPUT
  bEn               : BOOL;
  bFdb              : BOOL;
  bExtReq           : BOOL;
  bLock             : BOOL;
  nOffMin           : UDINT;
  nImplLngt         : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	*bEn* ist die allgemeine Freigabe des Funktionsbausteins. Wenn *bEn* FALSE ist, so ist der Meldeausgang *bQ* ebenfalls FALSE.
bFdb	BOOL	Eingang zum Anschluss des Feedbacksignals eines Motors oder Ventils. Dieser Eingang wird nur in der Betriebsart *E_BA_AntBlkgMode.eOffTime* berücksichtigt.
bExtReq	BOOL	Aktiv in der Betriebsart E_BA_AntBlkgMode.eExternalRequest. Externe Anforderung eines Impulses zum Beispiel von einem Zeitschaltplan. Mit einer steigenden Flanke wird der Blockierschutz-Impuls gestartet.
bLock	BOOL	Aktiv in den Betriebsarten *E_BA_AntBlkgMode.eExternalRequest* oder *E_BA_AntBlkgMode.eOffTime*. Um zu verhindern, dass z.B. die Pumpe und das Ventil eines Erhitzers gleichzeitig einen Puls bekommt, wird die Ausgabe des Pulses immer so lange unterdrückt, bis *bLock* wieder FALSE ist. Wird *bLock* während der Ausgabe eines Blockierschutz-Pulses TRUE, dann wird der Blockierschutz-Impuls unterbrochen. Nachdem *bLock* wieder FALSE ist, wird der Blockierschutz-Impuls neu gestartet.
nOffMin	UDINT	Mindestausschaltzeit des Aktors ohne Bewegung des Motors oder Ventils [s].
nImplLngt	UDINT	Länge des Blockierschutz-Impulses [s] an *bQ*.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  eMode    : E_BA_AntBlkgMode := E_BA_AntBlkgMode.eOffTime;
END_VAR
```

Name	Typ	Beschreibung
eMode	E_BA_AntBlkgMode	Eingabe der Betriebsart des Funktionsbausteines

E_BA_AntBlkgMode.eOff - die Betriebsart *eOff* ist vergleichbar mit dem Eingang *bEn*. Ist diese Betriebsart aktiv, so ist der Impulsausgang *bQ* FALSE.

E_BA_AntBlkgMode.eExternalRequest - Externe Anforderung eines Impulses zum Beispiel von einem Zeitschaltplan. Mit einer steigenden Flanke wird der Blockierschutz-Impuls gestartet.

E_BA_AntBlkgMode.eOffTime - Mindestausschaltzeit des Aktors ohne Bewegung des Motors oder Ventils (*bFdb* = FALSE). Nach Ablauf des Timers wird der Blockierschutz-Impuls gestartet.

Ausgänge

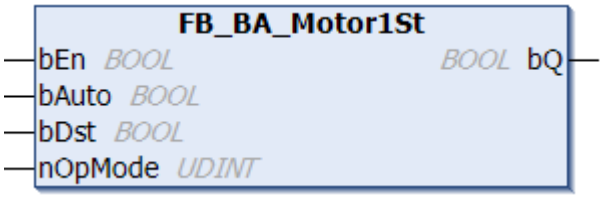
```
VAR_OUTPUT
  bQ                : BOOL;
  nRemOffMin        : UDINT;
  nRemImplLngt      : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ	BOOL	Ausgang zur Ausgabe des Blockierschutz-Impulses.
nRemOffMin	UDINT	Verbleibende Zeit [s] bis zur Ausgabe des nächsten Impulses bei andauerndem Stillstand.
nRemImplLngt	UDINT	Verbleibende Restzeit [s] des Impulses an *bQ*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.1.4 FB_BA_Motor1St



Der Funktionsbaustein FB_BA_Motor1St dient zur Steuerung eines einfachen einstufigen Motors.

Der Eingang *bEn* dient der allgemeinen Freigabe des Motors.

Über den Eingang *nOpMode* wird die Betriebsart des Motors eingestellt:

- 0 = Automatik
- 1 = Hand Aus
- 2 = Hand Ein

Im Automatikbetrieb (*nOpMode* = 0) lässt sich der Motor über den Eingang *bAuto* bedienen (*bAuto* = *bQ* =TRUE).

An *bDst* wird die Sammlung aller möglichen Störungen eines Motors angeschlossen.

Eingänge

```
VAR_INPUT
  bEn       : BOOL;
  bAuto     : BOOL;
  bDst      : BOOL;
  nOpMode   : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe Motor.
bAutoOpen	BOOL	Anforderung des Aktors im Automatikbetrieb (*nOpMode* = 0).
bDst	BOOL	Eingang zur Sammlung der möglichen Motorstörungen.
nOpMode	UDINT	Auswahl der Betriebsart (0 = Automatik, 1 = Hand Aus, 2 = Hand Ein).

Ausgänge

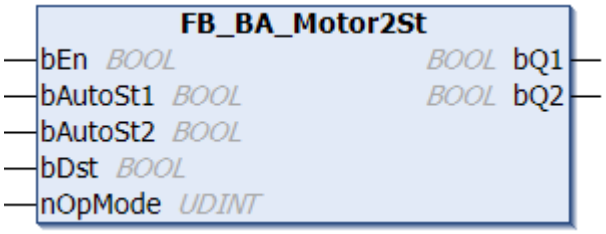
```
VAR_OUTPUT
  bQ   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bQ	BOOL	Steuerausgang

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.1.5 FB_BA_Motor2St



Der Funktionsbaustein FB_BA_Motor2St dient zur Steuerung eines einfachen zweistufigen Motors.

Der Eingang *bEn* dient der allgemeinen Freigabe des Motors.

Über den Eingang *nOpMode* wird die Betriebsart des Motors eingestellt:

- 0 = Automatik
- 1 = Hand Aus
- 2 = Hand Stufe 1
- 3 = Hand Stufe 2

Im Automatikbetrieb (*nOpMode* = 0) lässt sich die gewünschte Stufe über die Eingänge *bAutoSt1* (Stufe 1) und *bAutoSt2* (Stufe 2) einstellen.

An *bDst* wird die Sammlung aller möglichen Störungen eines Motors angeschlossen.

Eingänge

```
VAR_INPUT
  bEn       : BOOL;
  bAutoSt1  : BOOL;
  bAutoSt2  : BOOL;
  bDst      : BOOL;
  nOpMode   : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe Motor
bAutoSt1	BOOL	Anforderung des Aktors auf Stufe 1 im Automatikbetrieb (*nOpMode* = 0).
bAutoSt2	BOOL	Anforderung des Aktors auf Stufe 2 im Automatikbetrieb (*nOpMode* = 0).
bDst	BOOL	Eingang zur Sammlung der möglichen Motorstörungen.
nOpMode	UDINT	Auswahl der Betriebsart (0 = Automatik, 1 = Hand Aus, 2 = Hand Stufe 1, 3 = Hand Stufe 2).

Ausgänge

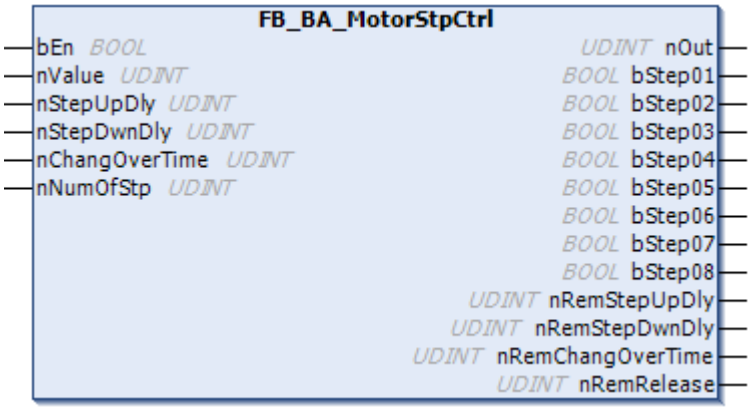
```
VAR_OUTPUT
  bQ1   : BOOL;
  bQ2   : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bQ1	BOOL	Steuerausgang Stufe 1
bQ2	BOOL	Steuerausgang Stufe 2

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

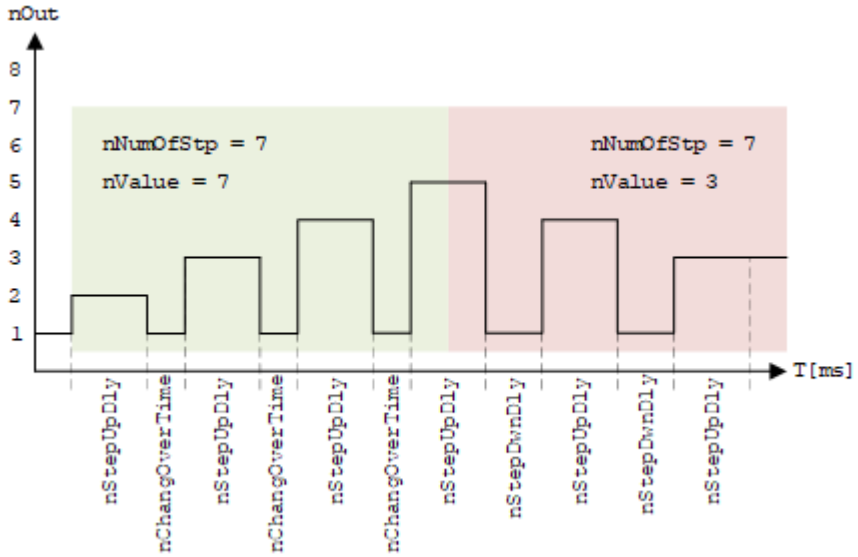
6.1.2.2.3.1.5.1.6 FB_BA_MotorStpCtrl



Der Funktionsbaustein FB_BA_MotorStpCtrl dient zur Steuerung von mehrstufigen Antrieben. Gestartet wird immer in Stufe 1 und je nach Anforderung durch *nValue* wird schrittweise in die nächsthöheren Stufen geschaltet. Das Hoch - und Runterschalten in die einzelnen Stufen wird durch die 3 Zeitangaben *nStepUpDly*, *nStepDwnDly* und *nChangOverTime* beeinflusst.

Bei einem Neustart oder durch Wegnahme der Freigabe ist ein Neuanlauf des Antriebs für die Zeitdauer von (*nOut* * *nStepDwnDly*) gesperrt. Angezeigt wird dieser Zeitverlauf durch die Ausgangsvariable *nRemRelease*. Intern wird der letzte aktive Zustand von *nOut* persistent gespeichert für die Berechnung der Sperrzeit.

Beispiel



Fehlerbehandlung

Die Limitierung des Eingabewertes *nNumOfStp* wird intern überwacht und korrigiert.

nNumOfStp < 1 wird auf den Wert 1 angepasst und über das Property *ErrorDescription* wird eine detaillierte Beschreibung ausgegeben.

nNumOfStp > 9 wird auf den Wert 9 angepasst und über das Property *ErrorDescription* wird eine detaillierte Beschreibung ausgegeben.

Zusätzlich wird eine Warnungsmeldung in dem Fenster Fehlerliste des TwinCAT Programmiertools ausgegeben.

Die Limitierung des Eingabewertes *nValue* wird intern überwacht und korrigiert.

nValue < 1 wird auf den Wert 1 angepasst und über das Property *ErrorDescription* wird eine detaillierte Beschreibung ausgegeben.

nValue > 9 wird auf den Wert 9 angepasst und über das Property *ErrorDescription* wird eine detaillierte Beschreibung ausgegeben.

Zusätzlich wird eine Warnungsmeldung in dem Fenster Fehlerliste des TwinCAT Programmiertools ausgegeben.

Eingänge

```
VAR_INPUT
  bEn       : BOOL;
  nValue    : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins. Wenn *bEn* FALSE ist, so sind sämtliche Ausgangsvariablen FALSE oder haben den Wert 0.
nValue	UDINT	Anzusteuernde Stufe von 1 bis 9.

nValue	Anforderung
1	Aus
2	bStep01
3	bStep02
4	bStep03
5	bStep04
6	bStep05
7	bStep06
8	bStep07
9	bStep08

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  nStepUpDly        : UDINT := 3000;
  nStepDwnDly       : UDINT := 1000;
  nChangOverTime    : UDINT := 100;
  nNumOfStp         : UDINT := 4;
END_VAR
```

Name	Typ	Beschreibung
nStepUpDly	UDINT	Mindesteinschaltzeit der jeweiligen Stufe [ms].
nStepDwnDly	UDINT	Rückschaltzeit bzw. Ausschaltzeit der Stufen [ms].
nChangOverTime	UDINT	Zeitverzögerung für die Umschaltphase beim Hochschalten [ms] zwischen den Stufen, um die Motorwicklungen zu schonen. Während dieser Zeit sind sämtliche Ausgänge FALSE und *nOut* = 1.
nNumOfStp	UDINT	Eingabe der Anzahl der benötigten Stufen. Die Eingabe ist begrenzt von 1 bis 9.

Ausgänge

```
VAR_OUTPUT
  nOut                : UDINT;
  bStep01             : BOOL;
  bStep02             : BOOL;
  bStep03             : BOOL;
  bStep04             : BOOL;
  bStep05             : BOOL;
  bStep06             : BOOL;
  bStep07             : BOOL;
  bStep08             : BOOL;
  nRemStepUpDly       : UDINT;
```

```
  nRemStepDwnDly       : UDINT;
  nRemChangOverTime    : UDINT;
  nRemRelease          : UDINT;
END_VAR
```

Name	Typ	Beschreibung
nOut	UDINT	Ausgabe der aktuell gültigen Stufe von 1 bis 9.
bStep0N	BOOL	Ausgabe der Stufe N in Abhängigkeit von dem stufigen Ausgangssignal *nOut*.
nRemStepUpDly	UDINT	Verbleibende Restzeit der Mindesteinschaltzeit der jeweiligen Stufe [ms].
nRemStepDwnDly	UDINT	Verbleibende Restzeit der Rückschaltzeit bzw. Ausschaltzeit der Stufen [ms].
RemChangOverTime	UDINT	Verbleibende Restzeit der Umschaltzeit beim Hoch- und Runterschalten [ms].
nRemRelease	UDINT	Verbleibende Restzeit der internen Sperrung des Funktionsbausteins nach einem Neustart oder durch Wegnahme der Freigabe *bEn* [ms].

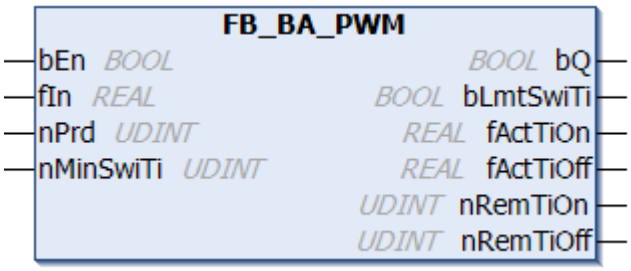
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ErrorDescription	T_MaxString	Get	Gibt eine detaillierte Beschreibung von Fehlern heraus, siehe Fehlerbehandlung [▶ 423].
NumberOfSteps	UDINT	Get	Gibt den aktuellen oder korrigierten Wert der Anzahl der Stufen aus. Im Fehlerfall von nNumOfStp gibt es eine detaillierte Beschreibung des Fehlers unter Fehlerbehandlung [▶ 423].
Step	STRING	Get	Ausgabe der aktuellen Stufe in Abhängigkeit von *nOut*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.1.7 FB_BA_PWM



Der Funktionsbaustein FB_BA_PWM errechnet aus einem analogen Eingangssignal *rIn* (0…100 %, **intern fest begrenzt**) und dem der Periodendauer *nPrd* [s] eine Ein- und eine Ausschaltzeit *fActTiOn* und *fActTiOff* [s].

Dabei entspricht:

- 100 % am Eingang einer Einschaltzeit *fActTiOn* von der gesamten Periodendauer *nPrd* und einer Ausschaltzeit *fActTiOff* von 0 s

- 0 % am Eingang einer Einschaltzeit *fActTiOn* von 0 s und einer Ausschaltzeit *fActTiOff* von der gesamten Periodendauer *nPrd*.

Darüber hinaus gibt es die Möglichkeit über *nMinSwiTi* [s] die Schaltdauer nach unten hin zu begrenzen, um Schäden an Antrieben durch zu kurze Stellimpulse zu vermeiden. Dieses Verhalten gilt jedoch nur für 0> *fIn* >100!

Bei *fIn* = 0 bzw. 100 bleibt der Ausgang *bQ* jedoch gelöscht, bzw. gesetzt, Nach Ablauf der Periodenzeit wird erneut das aktuelle Eingangssignal bewertet. Steht es immer noch auf 0 bzw. 100 erfolgt kein Zustandswechsel von *bQ*.

Schaltverhalten

1. Ein FALSE-Signal am Eingang *bEn* setzt den Baustein außer Funktion und *bQ* auf FALSE. Es werden lediglich die Ein-und Ausschaltzeiten kontinuierlich berechnet und an den Ausgängen *fActTiOn* / *fActTiOff* [s] angezeigt.
2. Eine steigende Flanke am Eingang *bEn* aktiviert den Baustein: Er springt zunächst in einen Entscheidungs-Schritt. Je nach vorherigem Zustand vom Schaltausgang *bQ* wird nun in den umschaltenden Schritt gesprungen. Steht der Eingang *fIn* jedoch auf 0, so erfolgt ein unmittelbarer Sprung in den Aus-Schritt (*bQ*=FALSE) bzw. bei *fIn*=100 in den EIN-Schritt (*bQ*=TRUE), ohne auf den vorherigen Zustand von *bQ* zu achten. Die Mindest-Schaltzeit ist für diese beiden Fälle deaktiviert.
3. In dem jeweils aktiven Schritt (EIN oder AUS) läuft nun ein Countdown-Timer mit dem aktuell errechneten Startwert ab, der sich aus dem Puls-Pause-Verhältnis ergibt. Der Ein- bzw. Aus-Schritt wird mit der errechneten Zeit durchlaufen, egal, ob sich währenddessen das Puls-Pause-Verhältnis ändert. Der jeweilige Countdown wird an den Ausgängen *nRemTiOn* / n*RemTiOff* in vollen Sekunden angezeigt.
4. Nach Ablauf des Ein- bzw. des Aus-Schrittes erfolgt ein Rücksprung in den Entscheidungsschritt (Punkt 2).

Eingänge

```
VAR_INPUT
  bEn             : BOOL;
  fIn             : REAL;
  nPrd            : UDINT;
  nMinSwiTi       : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung der Puls-Weiten-Modulation.
fIn	REAL	Eingangssignal, intern auf 0…100 % fest begrenzt.
nPrd	UDINT	Periodenzeit [s]. Intern begrenzt auf einen Minimalwert von 0.
nMinSwiTi	UDINT	Minimale Einschaltzeit [s], zur Vermeidung zu kurzer Pulse. Intern begrenzt auf Werte von 0 bis *nPrd*.

Ausgänge

```
VAR_OUTPUT
  bQ              : BOOL;
  bLmtSwiTi       : BOOL;
  fActTiOn        : REAL;
  fActTiOff       : REAL;
  nRemTiOn        : UDINT;
  nRemTiOff       : UDINT;
END_VAR
```

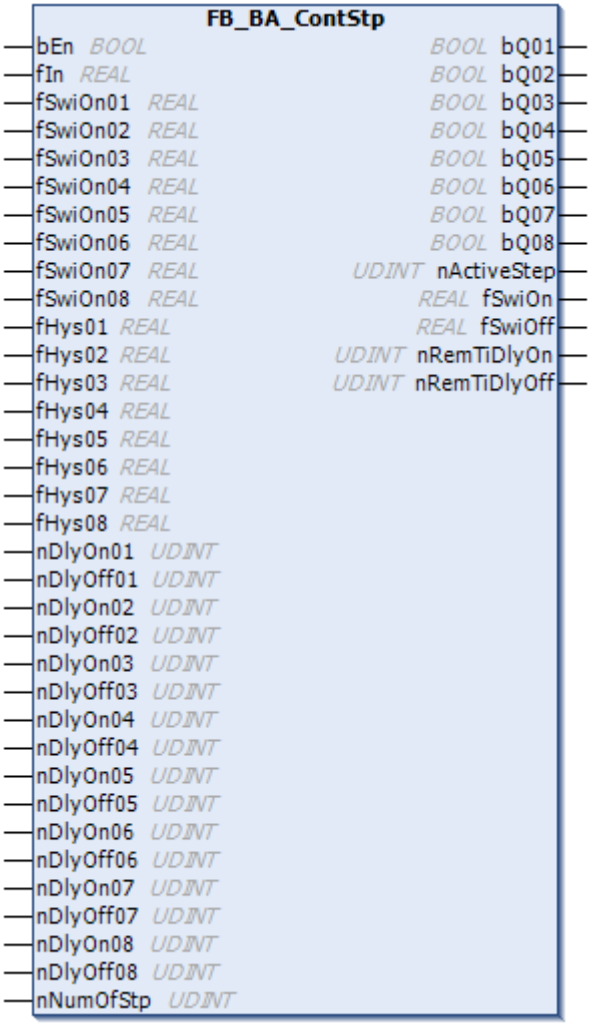
Name	Typ	Beschreibung
bQ	BOOL	PWM-Ausgang.
bLmtSwiTi	BOOL	Informationsausgang, dass das Eingangssignal so niedrig ist, dass mit der minimalen Einschaltzeit begrenzt wird.
fActTiOn	REAL	Informationsausgang: errechnete Einschaltzeit [s].
fActTiOff	REAL	Informationsausgang: errechnete Ausschaltzeit [s].
nRemTiOn	UDINT	Countdown Einschalt-Timer [s].
nRemTiOff	UDINT	Countdown Ausschalt-Timer [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

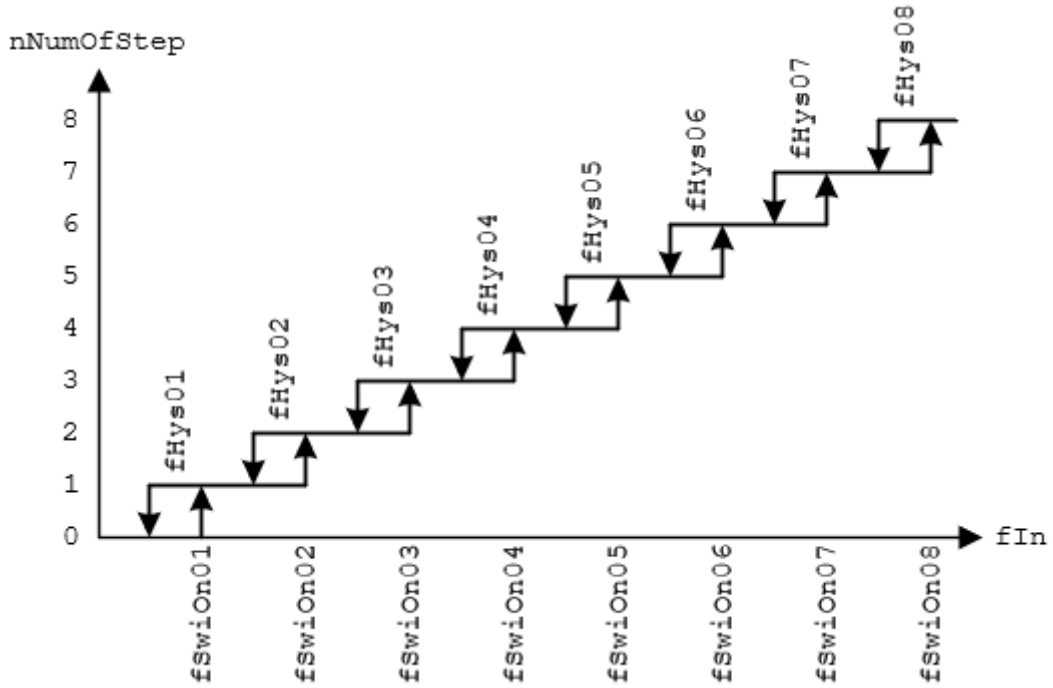
6.1.2.2.3.1.5.2 Control

6.1.2.2.3.1.5.2.1 FB_BA_ContStp



Der Funktionsbaustein FB_BA_ContStp ermittelt abhängig vom stetigen Eingangssignal *fIn* die resultierenden Schaltstufen eines mehrstufigen Aggregats.

nActive-Step	nNumOf-Step	fSwiOn	fSwiOff	nRemTiDlyOn	nRemTiDlyOff	bQ01	bQ02	bQ03	bQ04	bQ05	bQ06	bQ07	bQ08
0	0	fSwiOn01	fSwiOn01 - fHys01	nDlyOn01		FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	> = 1	fSwiOn02	fSwiOn01 - fHys01	nDlyOn02	nDlyOff01	**TRUE**	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	> = 2	fSwiOn03	fSwiOn02 - fHys02	nDlyOn03	nDlyOff02	**TRUE**	**TRUE**	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	> = 3	fSwiOn04	fSwiOn03 - fHys03	nDlyOn04	nDlyOff03	**TRUE**	**TRUE**	**TRUE**	FALSE	FALSE	FALSE	FALSE	FALSE
4	> = 4	fSwiOn05	fSwiOn04 - fHys04	nDlyOn05	nDlyOff04	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALSE	FALSE	FALSE	FALSE
5	> = 5	fSwiOn06	fSwiOn05 - fHys05	nDlyOn06	nDlyOff05	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALSE	FALSE	FALSE
6	> = 6	fSwiOn07	fSwiOn06 - fHys06	nDlyOn07	nDlyOff06	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALSE	FALSE
7	> = 7	fSwiOn08	fSwiOn07 - fHys07	nDlyOn08	nDlyOff07	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	FALSE
8	8	fSwiOn08	fSwiOn08 - fHys08		nDlyOff08	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**	**TRUE**



Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  fIn          : REAL;
  fSwiOn01     : REAL;
```

```
  fSwiOn02    : REAL;
  fSwiOn03    : REAL;
  fSwiOn04    : REAL;
  fSwiOn05    : REAL;
  fSwiOn06    : REAL;
  fSwiOn07    : REAL;
  fSwiOn08    : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins. Wenn *bEn* FALSE ist, so sind sämtliche Meldeausgänge *bQ0N* ebenfalls FALSE.
fIn	REAL	Stetiger Eingangswert, von dem die Schaltzustände abgeleitet werden.
fSwiOn0N	REAL	Einschaltpunkt Stufe 0N.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  fHys01       : REAL := 5;
  fHys02       : REAL := 5;
  fHys03       : REAL := 5;
  fHys04       : REAL := 5;
  fHys05       : REAL := 5;
  fHys06       : REAL := 5;
  fHys07       : REAL := 5;
  fHys08       : REAL := 5;
  nDlyOn01     : UDINT;
  nDlyOff01    : UDINT;
  nDlyOn02     : UDINT;
  nDlyOff02    : UDINT;
  nDlyOn03     : UDINT;
  nDlyOff03    : UDINT;
  nDlyOn04     : UDINT;
  nDlyOff04    : UDINT;
  nDlyOn05     : UDINT;
  nDlyOff05    : UDINT;
  nDlyOn06     : UDINT;
  nDlyOff06    : UDINT;
  nDlyOn07     : UDINT;
  nDlyOff07    : UDINT;
  nDlyOn08     : UDINT;
  nDlyOff08    : UDINT;
  nNumOfStp    : UDINT := 4;
END_VAR
```

Name	Typ	Beschreibung
fHys0N	REAL	Absolutwert Hysterese Stufe 0N.
nDlyOn0N	UDINT	Einschaltverzögerung Stufe 0N.
nDlyOff0N	UDINT	Ausschaltverzögerung Stufe 0N.
nNumOfStp	UDINT	Eingabe der Anzahl der benötigten Stufen. Die Eingabe ist begrenzt von 0 bis 8.

Ausgänge

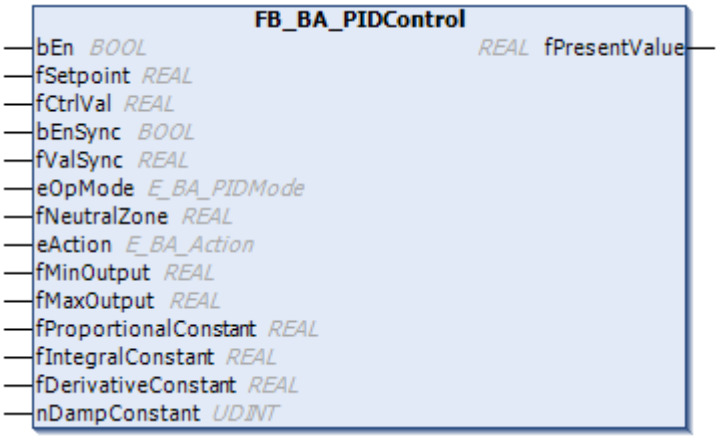
```
VAR_OUTPUT
  bQ01           : BOOL;
  bQ02           : BOOL;
  bQ03           : BOOL;
  bQ04           : BOOL;
  bQ05           : BOOL;
  bQ06           : BOOL;
  bQ07           : BOOL;
  bQ08           : BOOL;
  nActiveStep    : UDINT;
  fSwiOn         : REAL;
  fSwiOff        : REAL;
  nRemTiDlyOn    : UDINT;
  nRemTiDlyOff   : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ0N	BOOL	Anzeige des Status Stufe 0N. Die Stufe kann erst aktiv sein, wenn die davor liegenden Stufen TRUE sind. TRUE = EIN; FALSE = AUS
nActiveStep	UDINT	Zeigt an, wie viele Stufen eingeschaltet sind.
fSwiOn	REAL	Zeigt den nächsten Einschaltpunkt an.
fSwiOff	REAL	Zeigt den nächsten Ausschaltpunkt an.
nRemTiDlyOn	UDINT	Ist der Einschaltpunkt für die Umschaltung auf die nächste Stufe erreicht, so wird hier die verbleibende Zeit der Einschaltverzögerung angezeigt.
nRemTiDlyOff	UDINT	Ist der Ausschaltpunkt für das Herunterschalten in die nächste Stufe erreicht, wird hier die verbleibende Zeit der Ausschaltverzögerung angezeigt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.2.2 FB_BA_PIDControl



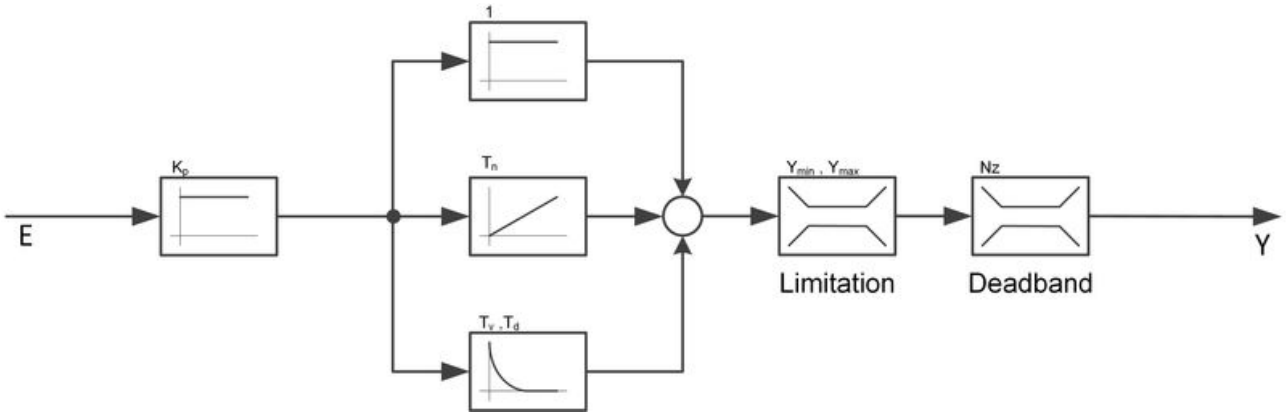
Der Funktionsbaustein FB_BA_PIDControl ist ein Universal-PID-Regler.

Der Regler ist intern in zwei aufeinander folgende Teile gegliedert:

- der Regler selbst, wie in den unten aufgeführten Wirkungsplänen als P-, I- und D-Anteil dargestellt mit einer Ausgangsbegrenzung (Limitation).
- einem Totbandglied (neutrale Zone), welches die Ausgangsänderungen des Reglers mit einer Hysterese beaufschlagt.

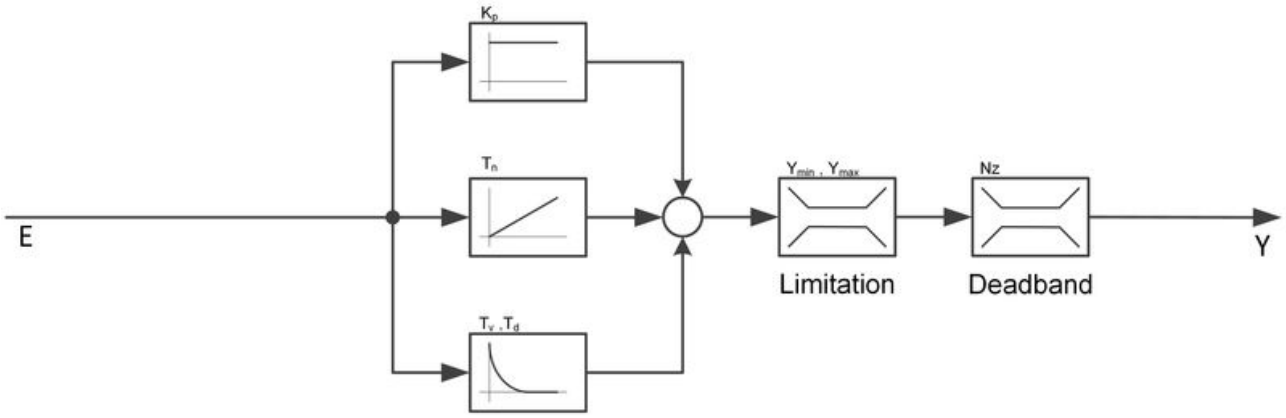
Betriebsmodus "P-Anteil vorgelagert":

(*eOpMode* = E_BA_PIDMode.eP1ID)



Betriebsmodus "Parallelstruktur":

(*eOpMode* = E_BA_PIDMode.ePID)



Wirksinn

Mit *eAction* = E_BA_Action.eReverse wird der Wirksinn des Reglers so umgekehrt, dass eine Regelabweichung kleiner als 0 eine Stellgrößenänderung ins Positive bewirkt. Dies wird dadurch erreicht, dass die Regelabweichung negativ berechnet wird:

eAction	*fE* (Regelabweichung)	Wirksinn
E_BA_Action.eDirect	*fCtrlVal-fSetpoint* (Istwert-Sollwert)	direkt (Kühlen)
E_BA_Action.eReverse	*fSetpoint-fCtrlVal* (Sollwert-Istwert)	indirekt (Heizen)

Passiv-Verhalten (bEn = FALSE)

Die Ausgänge werden wie folgt gesetzt:

fPresentValue	0.0

Die internen Werte für den P-, I-, und D-Anteil werden auf 0 gesetzt, ebenso die Werte für den I- und D-Anteil vom vorhergehenden Zyklus. Damit wird die Stellgröße bei einem Neustart im ersten Zyklus ohne Vergangenheitswerte berechnet.

Aktiv-Verhalten (bEn = TRUE)

Im ersten Zyklus werden I- und D-Anteil wie bereits erwähnt ohne Vergangenheitswerte berechnet und frei von alten Werten aufgestartet.

Anti-Reset-Windup

Ist der I-Anteil aktiv, so sorgt der Regler dafür, dass dieser festgehalten wird, sollte der Regler Ausgang *rY* über die Grenzen *fMinOutput* oder *fMaxOutput* hinausgehen wollen. Innerhalb des Reglers wird in jedem Zyklus eine Vorberechnung des Regler Ausgangs gemacht.

Anti-Reset-Windup an Min Grenze

Ist die Vorberechnung kleiner als die untere Ausgangsgrenze *fMinOutput*, so wird der I-Anteil daran gehindert weiter zu fallen und auf den Wert des letzten PLC-Zyklus begrenzt. Ein Ansteigen des I-Anteils bleibt jedoch möglich.

Anti-Reset-Windup an Max Grenze

Ist die Vorberechnung hingegen größer als die obere Grenze *fMaxOutput*, so wird der I-Anteil daran gehindert weiter zu steigen und ebenfalls auf den Wert des letzten PLC-Zyklus begrenzt. In diesem Fall bleibt ein Abfallen des I-Anteiles weiter möglich.

Synchronisationen

Es gibt verschiedene Fälle in denen Regler Ausgang nicht nur begrenzt, sondern durch Manipulation des I-Anteils (wenn nicht aktiv, dann des D-Anteils bzw. des P-Anteils) auf einen neuen Wert synchronisiert werden muss. Diese Fälle sind wegen der möglichen Gleichzeitigkeit priorisiert:

Prio	Beschreibung	Bedingungen	Bemerkungen
1	Synchronisation über *bEnSync*/*fValSync*	Regler freigegeben – *bEn* = TRUE	Ein positives Signal an *bEnSync* setzt den I-Anteil so, dass die Stellgröße den Wert *fValSync* annimmt. Diese Methode kann, wenn *bEn* und *bEnSync* gleichzeitig gesetzt werden, zum Setzen eines Initialwertes genutzt werden, von dem aus die Regelung "losläuft". Ist der I-Anteil nicht aktiv, so wird der D-Anteil entsprechend gesetzt. Zu beachten ist, dass intern nur die steigende Flanke von *bEnSync* ausgewertet wird, da es sich um eine Setz-Aktion handelt. Für ein erneutes Synchronisieren, etwa auf einen Übergabewert, muss am Eingang *bEnSync* ein erneutes TRUE-Signal angelegt werden.
2	Bereichssynchronisation *fMinOutput*	Regler freigegeben – *bEn* = TRUE *fMinOutput* <> fMinOutput_1 (letzterZyklus) fY_Test < *fMinOutput*	Hat sich die untere Bereichsgrenze geändert und die vorberechnete Regler Ausgabe ist nun kleiner als *fMinOutput* so wird auf *fMinOutput* synchronisiert.
3	Bereichssynchronisation *fMaxOutput*	Regler freigegeben – *bEn* = TRUE *fMaxOutput* <> fMaxOutput_1 (letzterZyklus) fY_Test > *fMaxOutput*	Hat sich die obere Bereichsgrenze geändert und die vorberechnete Regler Ausgabe ist nun größer als *fMaxOutput*, so wird auf *fMaxOutput* synchronisiert
4	Synchronisation bei Wirksinnumkehr	Regler freigegeben – *bEn* = TRUE *eAction* <> eAction_1 (letzter Zyklus)	Es wird so synchronisiert, dass der Ausgang den Wert VOR der Umkehr innehat: *fPresentValue* = fPresentValue_1 (letzter Zyklus)
5	Anti-Reset-Windup	Regler freigegeben – *bEn* = TRUE	Siehe Anti-Reset-Windup

Neutrale Zone

Ein Wert von *fNeutralZone* > 0.0 gibt die Funktion der neutralen Zone (Deadband) frei. Ein Wert gleich Null deaktiviert das Totbandglied und die Werte am Eingang werden direkt durchgereicht.

Ist beim aktiven Regler die Änderung am Eingang des Gliedes in einem SPS-Zyklus im Vergleich zum vorhergehenden SPS-Zyklus kleiner als *fNeutralZone* / 2, so wird der Ausgang auf dem Wert des vorhergehenden Zyklus gehalten, bis die Änderung größer oder gleich *fNeutralZone* / 2 ist.

Durch diese Funktion sollen unnötig viele Stellimpulse vermieden werden.

Syntax

```
VAR_INPUT
  bEn                          : BOOL;
  fSetpoint                    : REAL;
  fCtrlVal                     : REAL;
  bEnSync                      : BOOL;
  fValSync                     : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory':='Config'}
  eOpMode                      : E_BA_PIDMode := E_BA_PIDMode.ePID;
  {attribute 'parameterCategory':='Deadband'}
  fNeutralZone                 : REAL := 0.0;
  {attribute 'parameterCategory':='Config'}
  eAction                      : E_BA_Action := E_BA_Action.eReverse;
  {attribute 'parameterCategory':='Config'}
  fMinOutput                   : REAL := 0;
  {attribute 'parameterCategory':='Config'}
  fMaxOutput                   : REAL := 100;
  {attribute 'parameterCategory':='Config'}
  fProportionalConstant        : REAL := 3;
  {attribute 'parameterCategory':='Config'}
  {attribute 'parameterUnit':= 's,ms'}
  fIntegralConstant            : REAL;
  {attribute 'parameterCategory':='Config'}
  {attribute 'parameterUnit':= 's,ms'}
  fDerivativeConstant          : REAL;
  {attribute 'parameterCategory':='Config'}
  {attribute 'parameterUnit':= 's'}
  nDampConstant                : UDINT;
END_VAR
VAR_OUTPUT
  fPresentValue                : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Reglers
fSetpoint	REAL	Sollwert der Regelstrecke
fCtrlVal	REAL	Istwert der Regelstrecke
bEnSync	BOOL	Synchronisiert den Regler auf den Wert von *fValSync*.
fValSync	REAL	Synchronisationswert. Der Wert *fValSync* wird intern begrenzt auf Werte von *fMinOutput* bis *fMaxOutput*.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eOpMode	E_BA_PIDMode	Wirkungsweise des Reglers: PID-Modus oder P-ID-Modus.
fNeutralZone	REAL	Totband
eAction	E_BA_Action	Wirksinn des Reglers
fMinOutput	TIME	Untere Ausgabebegrenzung des Reglers [%]. Der Wert *fMinOutput* wird nach oben hin durch *fMaxOutput* begrenzt.
fMaxOutput	TIME	Obere Ausgabebegrenzung des Reglers [%].
fProportionalConstant	REAL	Regler Verstärkung. Wirkt nur auf den P-Anteil. Intern begrenzt auf einen Minimalwert von 0.
fIntegralConstant	REAL	Nachstellzeit des I-Anteiles [s,ms]. Ein Nullwert an diesem Parameter schaltet den I-Anteil ab.
fDerivativeConstant	REAL	Vorhaltezeit des D-Anteiles [s,ms]. Ein Nullwert an diesem Parameter schaltet den D-Anteil ab.
nDampConstant	UDINT	Dämpfungszeit des D-Anteiles [s].

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Stellgröße. Bereich durch *fYMin* und *fYMax* eingeschränkt.

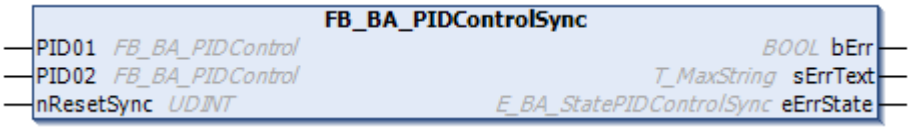
Eigenschaften

Name	Typ	Zugriff	Beschreibung
AntiResetWindup	BOOL	Get	Der Regler befindet sich im Status Anti-Reset-Windup.
ControlDeviation	REAL	Get	Regelabweichung
IsMaxLimit	BOOL	Get	Die Regler Ausgabe *fPresentValue* ist an der oberen Ausgabegrenze *fMaxOutput*.
IsMinLimit	BOOL	Get	Die Regler Ausgabe *fPresentValue* ist an der untere Ausgabegrenze *fMinOutput*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_BA2 ab v5.4.2.0

6.1.2.2.3.1.5.2.3 FB_BA_PIDControlSync



Der Funktionsbaustein FB_BA_PIDControlSync dient der Parameter-Synchronisation für 2 Regler vom Typ FB_BA_PIDControl [▶ 430]. Dieser Baustein kann beispielsweise zum Abgleich von Führungsreglern dienen, welche zur Erzeugung der Zuluft Sollwerte für eine raumlufttechnische Anlage eingesetzt werden.

Die folgenden Parameter des FB_BA_PIDControl [▶ 430] werden synchronisiert:

- eOpMode
- eActionRm
- fProportionalConstant
- fIntegralConstant
- fDerivativeConstant
- fMaxOutputRm
- fMinOutputRm
- nDampConstant
- fNeutralZone

Fehlererkennung

Die unten aufgeführten Fehlermeldungen werden durch den FB_BA_PIDControlSync erkannt.

Die Fehlermeldungen werden in der Entwicklungsumgebung von TwinCAT 3 in dem Fenster "Fehlerliste" ausgegeben. Dieses kann unter dem Menüpunkt Ansicht aktiviert werden.

Die Fehlertexte werden über das Property *ErrText* und den Ausgang *sErrText* ausgegeben.

Zusätzlich werden die Meldungen durch das Enum *eErrState* angezeigt.

Fehlermeldungen

Meldetext Deutsch	Meldetext Englisch
'Synchronisation *eAction* fehlerhaft'	'Synchronization *eAction* faulty'
'Synchronisation *nDampConstant* fehlerhaft'	'Synchronization *nDampConstant* faulty'
'Synchronisation *fDerivativeConstant* fehlerhaft'	'Synchronization *fDerivativeConstant* faulty'
'Synchronisation *fIntegralConstant* fehlerhaft'	'Synchronization *fIntegralConstant* faulty'
'Synchronisation *fMaxOutput* fehlerhaft'	'Synchronization *fMaxOutput* faulty'
'Synchronisation *fMinOutput* fehlerhaft'	'Synchronization *fMinOutput* faulty'
'Synchronisation *fNeutralZone* fehlerhaft'	'Synchronization *fNeutralZone* faulty'
'Synchronisation *eOpMode* fehlerhaft'	'Synchronization *eOpMode* faulty'
'Synchronisation *fProportionalConstant* fehlerhaft'	'Synchronization *fProportionalConstant* faulty'

Syntax

```
FUNCTION_BLOCK FB_BA_LoopSync
VAR_OUTPUT
  bErr          : BOOL;
  sErrText      : T_MaxString;
  eErrState     : E_BA_StatePIDControlSync;
END_VAR
VAR_IN_OUT
  PID01         : FB_BA_PIDControl;
  PID02         : FB_BA_PIDControl;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bErr	BOOL	Der Ausgang zeigt an, wenn ein Fehler bei der Synchronisation aufgetreten ist.
sErrText	T_MaxString	Die Variable zeigt den Status der Synchronisation in Textform [▶ 435] an.
eErrState	E_BA_StatePIDControlSync [▶ 276]	Die Enumeration zeigt den Status der Synchronisation an.

Ein- Ausgänge

Name	Typ	Beschreibung
Loop1	FB_BA_PIDControl [▶ 430]	Referenz auf den Regler Nr. 1 des Parameterabgleichs.
Loop2	FB_BA_PIDControl [▶ 430]	Referenz auf den Regler Nr. 2 des Parameterabgleichs.

Eigenschaften

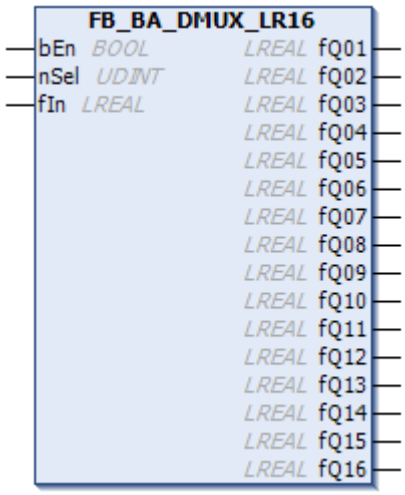
Name	Typ	Zugriff	Beschreibung
ErrText	T_MaxString	Get	Das Property *ErrText* zeigt die Fehlertexte von *sErrText* an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_BA2 ab v5.4.2.0

6.1.2.2.3.1.5.3 General Control Functions

6.1.2.2.3.1.5.3.1 FB_BA_DMUX_XX



Die Demultiplexer-Bausteine existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Ausgangsgrößen (4, 8, 12 und 16), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_DMUX_LR16 beschrieben.

Der Funktionsbaustein FB_BA_DMUX_XX gibt im aktivierten Zustand (*bEn* = TRUE) den Wert am Eingang *fIn* an demjenigen Ausgang *fQ01..fQ16* aus dessen Nummer am Eingang *nSel* eingetragen ist. Alle anderen Ausgänge werden auf 0 gesetzt (bei den booleschen Demultiplexern auf FALSE).

Beispiel:

Eingänge	Ausgänge
bEn = TRUE	fQ01 = 0.0
nSel = 5	fQ02 = 0.0
fIn = 32.5	fQ03 = 0.0
	fQ04 = 0.0
	fQ05 = 32.5
	fQ06 = 0.0
	fQ07 = 0.0
	fQ08 = 0.0
	fQ09 = 0.0
	fQ10 = 0.0
	fQ11 = 0.0
	fQ12 = 0.0
	fQ13 = 0.0
	fQ14 = 0.0
	fQ15 = 0.0
	fQ16 = 0.0

Ist der eingetragene Wert an *nSel* größer als die Anzahl der Ausgänge, so wird am "höchsten" Ausgang der Wert von *fIn* ausgegeben:

Eingänge	Ausgänge
bEn = TRUE	fQ01 = 0.0
nSel = 25	fQ02 = 0.0
fIn = 32.5	fQ03 = 0.0
	fQ04 = 0.0
	fQ05 = 0.0
	fQ06 = 0.0
	fQ07 = 0.0
	fQ08 = 0.0
	fQ09 = 0.0
	fQ10 = 0.0
	fQ11 = 0.0
	fQ12 = 0.0
	fQ13 = 0.0
	fQ14 = 0.0
	fQ15 = 0.0
	fQ16 = 32.5

Bei *bEn*=FALSE wird an allen Ausgängen 0.0 bzw. FALSE bei den booleschen Demultiplexern ausgegeben.

Eingänge

```
VAR_INPUT
  bEn     : BOOL;
  nSel    : UDINT;
  fIn     : LREAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung der Bausteinfunktion
nSel	UDINT	Nummer des Ausgangs *fQ01...fQ16*, der den Wert des Eingangs *fIn* annehmen soll.
fIn	LREAL	Wert, der zur Ausgabe kommen soll.

Ausgänge

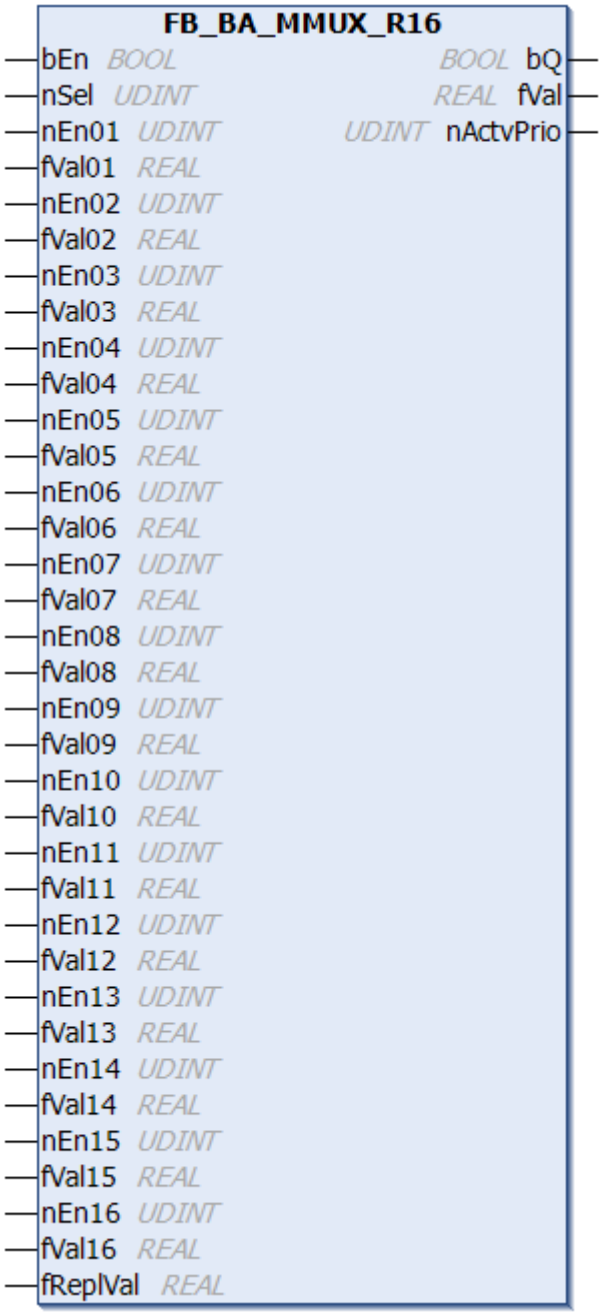
```
VAR_OUTPUT
  fQ01  : LREAL;
  fQ02  : LREAL;
  fQ03  : LREAL;
  fQ04  : LREAL;
  fQ05  : LREAL;
  fQ06  : LREAL;
  fQ07  : LREAL;
  fQ08  : LREAL;
  fQ09  : LREAL;
  fQ10  : LREAL;
  fQ11  : LREAL;
  fQ12  : LREAL;
  fQ13  : LREAL;
  fQ14  : LREAL;
  fQ15  : LREAL;
  fQ16  : LREAL;
END_VAR
```

Name	Typ	Beschreibung
fQ01...fQ16	LREAL	Wertausgänge

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.2 FB_BA_MMUX_XX



Der Funktionsbaustein FB_BA_MMUX_XX schaltet in Abhängigkeit eines Selektors und der entsprechenden Eingangs-Selektorbedingung einen Eingangswert auf den Ausgang.

Die Multiplexer-Bausteine existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Eingangsgrößen (4, 8, 12, 16 und 24), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_MMUX_R16 beschrieben.

Der Funktionsbaustein schaltet im aktivierten Zustand (*bEn* = TRUE) in Abhängigkeit eines Selektors *nSel* und der entsprechenden Eingangs-Selektorbedingung *nEnxx* einen von den Eingangswerten *fValxx* auf den Ausgang *fVal*.

Wenn mehrere Eingangs-Selektorbedingungen *nEn01*...*nEn16* gleich sind und der Selektor *nSel* einer Bedingung entspricht, dann wird der Eingangswert *fVal01*... *fVal16* der tiefsten aktiven Selektorbedingung auf den Ausgang *fVal* geschaltet. *nEn01* ist die tiefste, *nEn16* die höchste Selektorbedingung.

Die Ausgangsvariable *bQ* zeigt an, dass der Selektor *nSel* mit einer Eingangs-Selektorbedingung *nEnxx* übereinstimmt.
Die Ausgangsvariable *nActvPrio* zeigt die aktive Selektorbedingung an.
Ist keine Selektorbedingung aktiv, so wird *fReplVal* an *fVal* ausgegeben. *bQ* ist dann FALSE und *nActvPrio* zeigt eine 255 an.

Beispiel:

Eingänge		**Ausgang**	
Variable	**Wert**	**Variable**	**Wert**
bEn	TRUE	bQ	TRUE
nSel	5	fVal	1,123
nEn01	4	nActvPrio	7
fVal01	123		
nEn02			
fVal02			
nEn03	3		
fVal03	321		
nEn04			
fVal04			
nEn05	8		
fVal05	345		
nEn06			
fVal06			
nEn07	5		
fVal07	1,123		
nEn08			
fVal08			
nEn09	5		
fVal09	5,4321		
nEn10			
fVal10			
nEn11			
fVal11			
nEn12			
fVal12			
nEn13			
fVal13			
nEn14			
fVal14			
nEn15			
fVal15			
nEn16			
fVal16			
fReplVal			

Wenn keine aktive Priorität ansteht, dann wird an dem Ausgang *nActvPrio* der Wert der globalen Konstanten BA_Globals.nNoActivePrio [▶ 288] ausgegeben.

Eingänge

```
VAR_INPUT
  bEn         : BOOL;
  nSel        : UDINT;
  nEn01       : UDINT := BA_Globals.nNoActvPrio;
  fVal01      : REAL;
  nEn02       : UDINT := BA_Globals.nNoActvPrio;
  fVal02      : REAL;
  nEn03       : UDINT := BA_Globals.nNoActvPrio;
  fVal03      : REAL;
  nEn04       : UDINT := BA_Globals.nNoActvPrio;
  fVal04      : REAL;
  nEn05       : UDINT := BA_Globals.nNoActvPrio;
  fVal05      : REAL;
  nEn06       : UDINT := BA_Globals.nNoActvPrio;
  fVal06      : REAL;
  nEn07       : UDINT := BA_Globals.nNoActvPrio;
  fVal07      : REAL;
  nEn08       : UDINT := BA_Globals.nNoActvPrio;
  fVal08      : REAL;
  nEn09       : UDINT := BA_Globals.nNoActvPrio;
  fVal09      : REAL;
  nEn10       : UDINT := BA_Globals.nNoActvPrio;
  fVal10      : REAL;
  nEn11       : UDINT := BA_Globals.nNoActvPrio;
  fVal11      : REAL;
  nEn12       : UDINT := BA_Globals.nNoActvPrio;
  fVal12      : REAL;
  nEn13       : UDINT := BA_Globals.nNoActvPrio;
  fVal13      : REAL;
  nEn14       : UDINT := BA_Globals.nNoActvPrio;
  fVal14      : REAL;
  nEn15       : UDINT := BA_Globals.nNoActvPrio;
  fVal15      : REAL;
  nEn16       : UDINT := BA_Globals.nNoActvPrio;
  fVal16      : REAL;
  fReplVal    : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung der Bausteinfunktion
nSel	UDINT	Selektor. Intern begrenzt auf Werte von 0 bis 4294967294.
nEn01…nEn16	UDINT	Eingangswerte, aus denen gewählt werden soll.
fReplVal	REAL	Ersatzwert, wenn keine Eingangs-Selektorbedingung aktiv ist.

Ausgänge

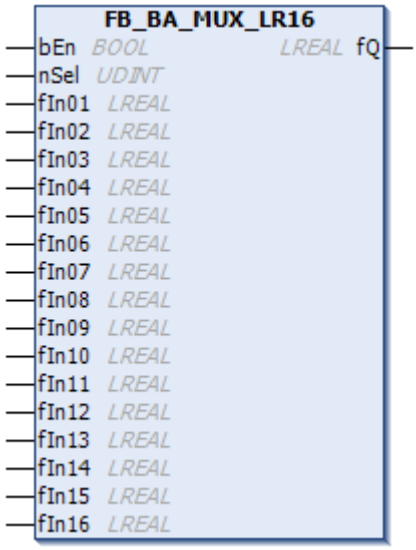
```
VAR_OUTPUT
  bQ          : BOOL;
  fVal        : REAL;
  nActvPrio   : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ	BOOL	Ist TRUE, wenn der Selektor *nSel* mit einer Eingangs-Selektorbedingung *nEnxx* übereinstimmt.
fVal	REAL	Wert der gewählten Eingangs-Selektorbedingung.
nActvPrio	UDINT	Zeigt an welche Eingangs-Selektorbedingung aktiv ist. Wenn keine aktive Priorität ansteht, dann wird an dem Ausgang *nActvPrio* der Wert der globalen Konstanten BA_Globals.nNoActivePrio ausgegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.3 FB_BA_MUX_XX



Die Multiplexer-Bausteine existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Eingangsgrößen (4, 8, 12 und 16), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_MUX_LR16 beschrieben.

Der Funktionsbaustein FB_BA_MUX_XX gibt im aktivierten Zustand (*bEn*=TRUE) denjenigen Eingangswert *fIn01..fIn16* am Ausgang *fQ* aus, dessen Nummer am Eingang *nSel* eingetragen ist.
Beispiel:

Eingänge	Ausgang
bEn = TRUE	fQ = 16.5
nSel = 5	
fIn01 = 15.9	
fIn02 = 32.5	
fIn03 = 17.4	
fIn04 = 5.84	
fIn05 = 9.56	
fIn06 = 16.5	
fIn07 = 32.781	
fIn08 = 25.4	
fIn09 = 44.5	
fIn10 = 66.1	
fIn11 = 45.5	
fIn12 = 83.3	
fIn13 = 54.56	
fIn14 = 33.8	
fIn15 = 98.5	
fIn16 = 71.3	

Ist der eingetragene Wert an *nSel* größer als die Anzahl der Eingänge, so wird der "höchstrangige" Eingang an *fQ* ausgegeben:

Eingänge	Ausgang
bEn = TRUE	fQ = 2.3
nSel = 25	
fIn01 = 15.9	
fIn02 = 32.5	
fIn03 = 17.4	
fIn04 = 5.84	
fIn05 = 9.56	
fIn06 = 16.5	
fIn07 = 32.781	
fIn08 = 25.4	
fIn09 = 44.5	
fIn10 = 66.1	
fIn11 = 45.5	
fIn12 = 83.3	
fIn13 = 54.56	
fIn14 = 33.8	
fIn15 = 98.5	
fIn16 = 71.3	

Bei *bEn*=FALSE wird am Ausgang *fQ* 0.0 bzw. FALSE bei den booleschen Multiplexern ausgegeben.

Eingänge

```
VAR_INPUT
  bEn    : BOOL;
  nSel   : UDINT;
  fIn01  : LREAL;
  fIn02  : LREAL;
  fIn03  : LREAL;
  fIn04  : LREAL;
  fIn05  : LREAL;
  fIn06  : LREAL;
  fIn07  : LREAL;
  fIn08  : LREAL;
  fIn09  : LREAL;
  fIn10  : LREAL;
  fIn11  : LREAL;
  fIn12  : LREAL;
  fIn13  : LREAL;
  fIn14  : LREAL;
  fIn15  : LREAL;
  fIn16  : LREAL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung der Bausteinfunktion
nSel	UDINT	Nummer des Eingangs, dessen Wert am Ausgang *fQ* ausgegeben werden soll.
f01...f16	LREAL	Eingangswerte aus denen gewählt werden soll.

Ausgänge

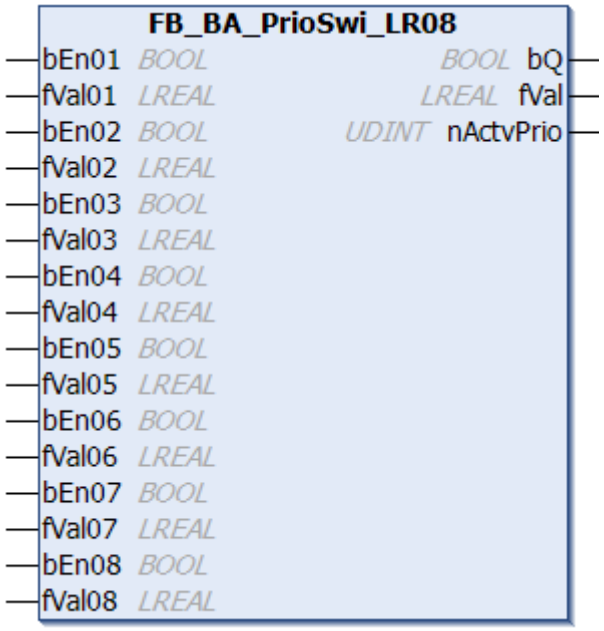
```
VAR_OUTPUT
  fQ   : LREAL;
END_VAR
```

Name	Typ	Beschreibung
fQ	LREAL	Wert des gewählten Eingangs.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.4 FB_BA_PrioSwi_XX



Die Prioritätenschalter existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Ausgangsgrößen (4, 8, 12 und 16 bzw. 24), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Funktionsbaustein FB_BA_PrioSwi_LR08 beschrieben.

Zur Selektion verschiedener Werte stehen Prioritätenschalter zur Verfügung. Zum Ausgang *fVal* wird derjenige Wert mit der höchsten Priorität durchgeschaltet dessen Eingang *bEnxx* TRUE ist.

Beispiel:

Eingänge			Ausgänge		
bEn01	FALSE		bQ	TRUE	
fVal01		32.5	fVal		5.84
bEn02	FALSE		nActvPrio		3
fVal02		17.4			
bEn03	TRUE				
fVal03		5.84			
bEn04	TRUE				
fVal04		9.56			
bEn05	FALSE				
fVal05		16.5			
bEn06	TRUE				
fVal06		32.781			
bEn07	FALSE				
fVal07		25.4			
bEn08	TRUE				
fVal08		44.5			

Ist keine der Prioritäten freigegeben, so fällt der Ausgang *bQ* auf FALSE. An den Ausgängen *fVal* und *nActvPrio* wird jeweils 0 ausgegeben. Bei einem booleschen Prioritätenschalter wird an dem Ausgang *bVal* dann FALSE ausgegeben.

Eingänge			Ausgänge		
bEn01	FALSE		bQ	FALSE	
fVal01		32.5	fVal		0.0
bEn02	FALSE		nActvPrio		0
fVal02		17.4			
bEn03	FALSE				
fVal03		5.84			
bEn04	FALSE				
fVal04		9.56			
bEn05	FALSE				
fVal05		16.5			
bEn06	FALSE				
fVal06		32.781			
bEn07	FALSE				
fVal07		25.4			
bEn08	FALSE				
fVal08		44.5			

Wenn keine aktive Priorität ansteht, dann wird am Ausgang *nActvPrio* der Wert der globalen Konstanten nNoActivePrio [▶ 127] ausgegeben.

Eingänge

```
VAR_INPUT
  bEn01   : BOOL;
  fVal01  : LREAL;
  bEn02   : BOOL;
  fVal02  : LREAL;
  bEn03   : BOOL;
  fVal03  : LREAL;
  bEn04   : BOOL;
  fVal04  : LREAL;
  bEn05   : BOOL;
  fVal05  : LREAL;
  bEn06   : BOOL;
  fVal06  : LREAL;
  bEn07   : BOOL;
  fVal07  : LREAL;
  bEn08   : BOOL;
  fVal08  : LREAL;
END_VAR
```

Name	Typ	Beschreibung
bEn01...bEn08	BOOL	Freigabe des Prioritätswertes
fVal01...fVal08	LREAL	Prioritätswert

Ausgänge

```
VAR_OUTPUT
  bQ         : BOOL;
  fVal       : LREAL;
  nActvPrio  : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ	BOOL	Ausgabe, ob überhaupt eine Priorität freigegeben ist.
fVal	LREAL	Ausgabe des Wertes der aktuellen (höchsten) Priorität, die freigegeben ist.
nActvPrio	UDINT	Aktuelle (höchste) Priorität, die freigegeben ist.

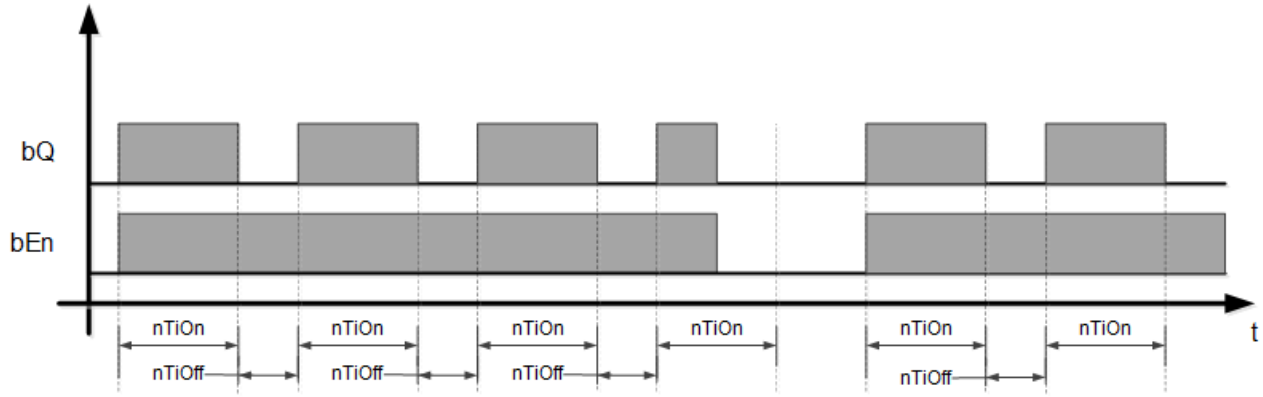
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.5 FB_BA_Blink



Der Funktionsbaustein FB_BA_Blink ist ein Oszillator mit einstellbarer Puls- und Pausenzeit, *nTiOn* und *nTiOff* [ms]. Er wird mit einem TRUE-Signal an *bEn* freigegeben und beginnt mit der Puls-Phase.



nTiNextSwi ist ein Countdown [s] zum nächsten Wechsel von *bQ*.

Eingänge

```
VAR_INPUT
  bEn           : BOOL;
  nTiOn         : UDINT;
  nTiOff        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Bausteinfreigabe
nTiOn	UDINT	Puls-Zeit [ms]
nTiOff	UDINT	Pausenzeit [ms]

Ausgänge

```
VAR_OUTPUT
  bQ              : BOOL;
  nTiNextSwi      : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ	BOOL	Oszillatorausgang
nTiNextSwi	UDINT	Countdown bis zum nächsten Wechsel von *bQ* [s]

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.6 FB_BA_FIFO04



Der Funktionsbaustein FB_BA_FIFO04 ermöglicht eine Folgesteuerung von bis zu vier Aggregaten mit automatischem Wechsel der Einschaltreihenfolge nach Betriebsstunden.
Der Funktionsbaustein steht in zwei Ausführungen zur Verfügung: für eine Sequenz von vier und von acht [▶ 447] Aggregaten.
In der Abfolge haben die Aggregate mit den wenigen Betriebsstunden Vorrang vor den mit vielen Betriebsstunden.

Eine steigende Flanke an *bChg* erzwingt einen Folgewechsel. Die Aggregate mit den geringsten Betriebsstunden werden in der FIFO nach vorne gesetzt und somit vorrangig eingeschaltet.
In der Folge werden nur Aggregate eingetragen, welche an den Eingängen *bEn01*…*bEn04* frei gegeben sind. *nNum* gibt die Anzahl der angeforderten Aggregate an.
An den Eingängen *nActvTi01* bis *nActvTi04* werden die Betriebsstunden der Aggregate eingetragen. Werden diese Eingänge alle konstant auf Null gesetzt, erfolgt der Folgewechsel nur in Abhängigkeit von *bChg* zyklisch gesteuert.
Dabei fällt immer das erste Aggregat aus dem FIFO heraus, die anderen werden aufgeschoben, und das erste Aggregat hinten in dem FIFO wieder angehängt. Somit rotieren die Aggregate in ihrer Folge.

Eingänge

```
VAR_INPUT
  bEn           : BOOL;
  nNum          : UDINT;
  bChg          : BOOL;
  bEn01         : BOOL;
  bEn02         : BOOL;
  bEn03         : BOOL;
  bEn04         : BOOL;
  nActvTi01     : UDINT;
  nActvTi02     : UDINT;
  nActvTi03     : UDINT;
  nActvTi04     : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Bausteinfreigabe
nNum	UDINT	Anzahl der Aggregate
bChg	BOOL	Folgewechsel erzwingen
bEn01...bEn04	BOOL	Freigabe Aggregat 1...Freigabe Aggregat 4.
nActvTi01...nActvTi04	UDINT	Betriebsstunden Aggregat 1...Betriebsstunden Aggregat 4.

Ausgänge

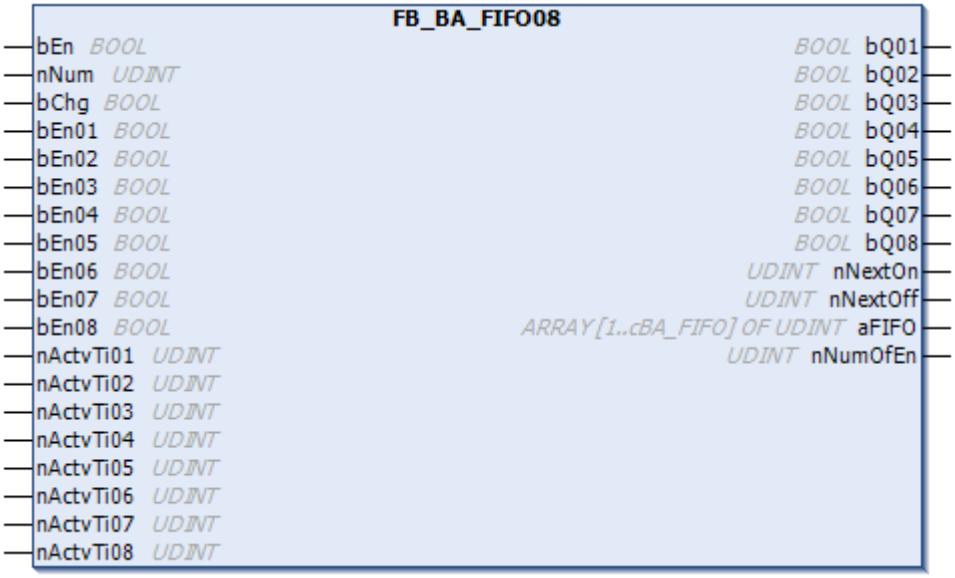
```
VAR_OUTPUT
  bQ01        : BOOL;
  bQ02        : BOOL;
  bQ03        : BOOL;
  bQ04        : BOOL;
  nNextOn     : UDINT;
  nNextOff    : UDINT;
  aFIFO       : ARRAY [1..4] OF UDINT;
  nNumOfEn    : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ01...bQ04	BOOL	Schaltet Aggregat 1…4.
nNextOn	UDINT	Nummer des Aggregates, welches als nächstes eingeschaltet wird.
nNextOff	UDINT	Nummer des Aggregates, welches als nächstes ausgeschaltet wird.
aFIFO	ARRAY OF UDINT	FIFO-Speicher als Feld.
nNumOfEn	UDINT	Anzahl Teilnehmer in Abhängigkeit der einzelnen Freigaben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.7 FB_BA_FIFO08



Der Funktionsbaustein FB_BA_FIFO08 ermöglicht eine Folgesteuerung von bis zu acht Aggregaten mit automatischem Wechsel der Einschaltreihenfolge nach Betriebsstunden.
Der Funktionsbaustein steht in zwei Ausführungen zur Verfügung: für eine Sequenz von vier [▶ 446] und von acht Aggregaten.

In der Abfolge haben die Aggregate mit den wenigen Betriebsstunden Vorrang vor den mit vielen Betriebsstunden.

Eine steigende Flanke an *bChg* erzwingt einen Folgewechsel. Die Aggregate mit den geringsten Betriebsstunden werden in der FIFO nach vorne gesetzt und somit vorrangig eingeschaltet.
In der Folge werden nur Aggregate eingetragen, welche an den Eingängen *bEn01…bEn08* frei gegeben sind. *nNum* gibt die Anzahl der angeforderten Aggregate an.
An den Eingängen *nActvTi01* bis *nActvTi08* werden die Betriebsstunden der Aggregate eingetragen. Werden diese Eingänge alle konstant auf Null gesetzt, erfolgt der Folgewechsel nur in Abhängigkeit von *bChg*

zyklisch gesteuert.
Dabei fällt immer das erste Aggregat aus dem FIFO heraus, die anderen werden aufgeschoben, und das erste Aggregat hinten in dem FIFO wieder angehängt. Somit rotieren die Aggregate in ihrer Folge.

Eingänge

```
VAR_INPUT
  bEn           : BOOL;
  nNum          : UDINT;
  bChg          : BOOL;
  bEn01         : BOOL;
  bEn02         : BOOL;
  bEn03         : BOOL;
  bEn04         : BOOL;
  bEn05         : BOOL;
  bEn06         : BOOL;
  bEn07         : BOOL;
  bEn08         : BOOL;
  nActvTi01     : UDINT;
  nActvTi02     : UDINT;
  nActvTi03     : UDINT;
  nActvTi04     : UDINT;
  nActvTi05     : UDINT;
  nActvTi06     : UDINT;
  nActvTi07     : UDINT;
  nActvTi08     : UDINT;
EN_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Bausteinfreigabe
nNum	UDINT	Anzahl der Aggregate
bChg	BOOL	Folgewechsel erzwingen
bEn01...bEn08	BOOL	Freigabe Aggregat 1...Freigabe Aggregat 8.
nActvTi01...nActvTi08	UDINT	Betriebsstunden Aggregat 1...Betriebsstunden Aggregat 8.

Ausgänge

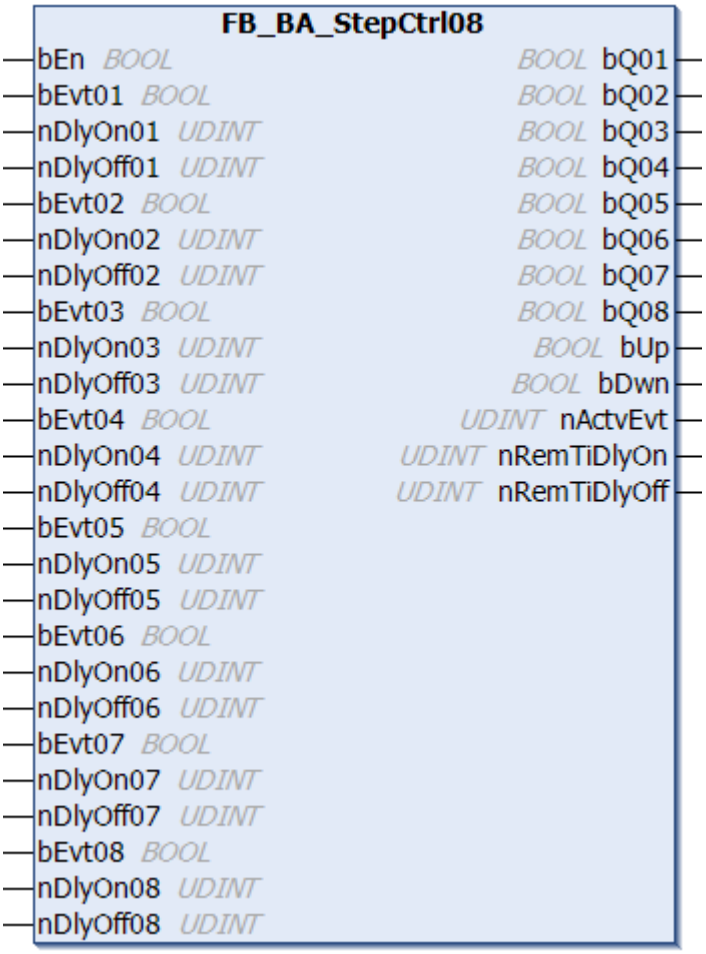
```
VAR_OUTPUT
  bQ01          : BOOL;
  bQ02          : BOOL;
  bQ03          : BOOL;
  bQ04          : BOOL;
  bQ05          : BOOL;
  bQ06          : BOOL;
  bQ07          : BOOL;
  bQ08          : BOOL;
  nNextOn       : UDINT;
  nNextOff      : UDINT;
  aFIFO         : ARRAY [1..8] OF UDINT;
  nNumOfEn      : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ01...bQ08	BOOL	Schaltet Aggregat 1…8.
nNextOn	UDINT	Nummer des Aggregates, welches als nächstes eingeschaltet wird.
nNextOff	UDINT	Nummer des Aggregates, welches als nächstes ausgeschaltet wird.
aFIFO	ARRAY OF UDINT	FIFO-Speicher als Feld.
nNumOfEn	UDINT	Anzahl Teilnehmer in Abhängigkeit der einzelnen Freigaben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.8 FB_BA_StepCtrl08

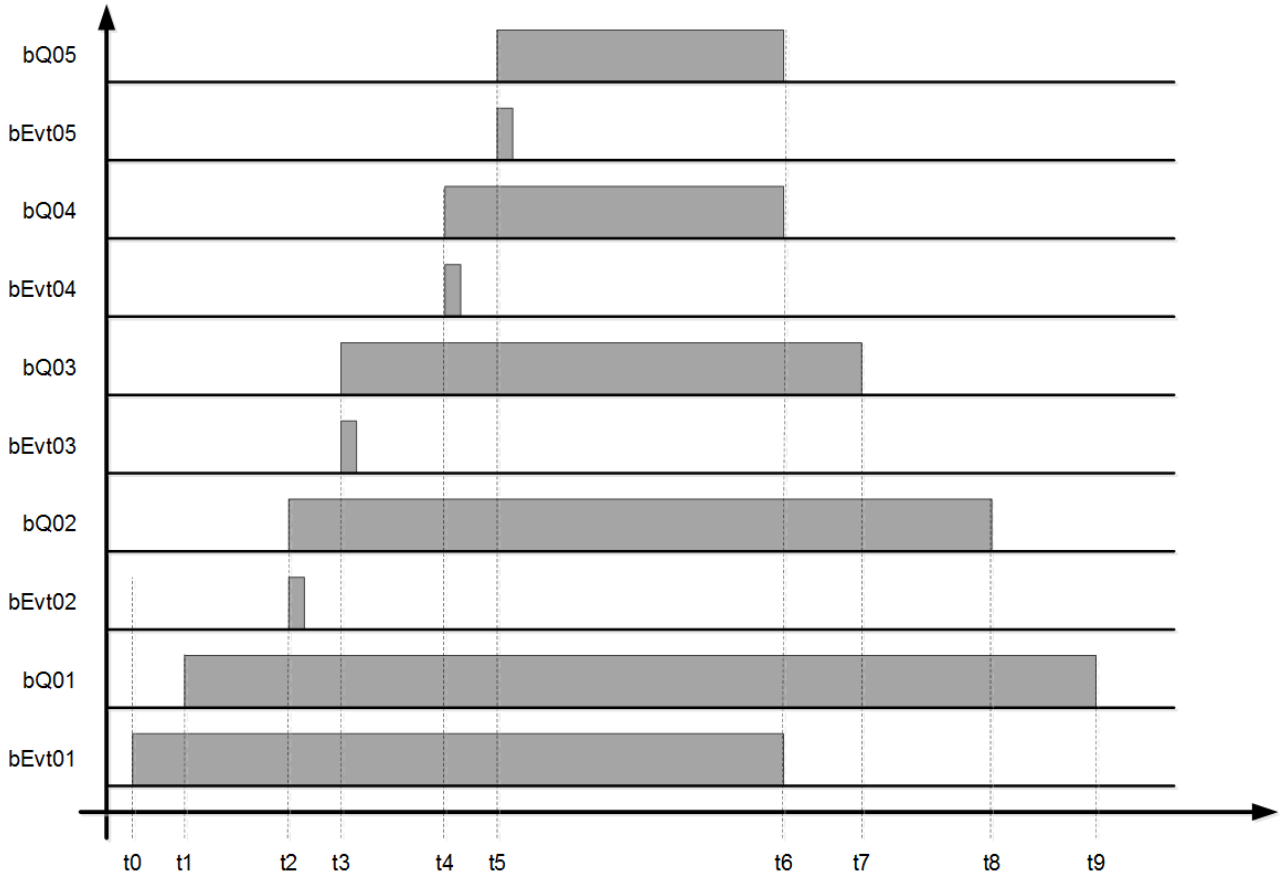


Der Funktionsbaustein FB_BA_StepCtrl08 dient zur Ausgabe sequenziell folgender Steuerungsbefehle. Eine typische Anwendung dieses Bausteins ist der Startvorgang einer raumlufttechnischen Anlage. *bEn* ist für die generelle Freigabe des Funktionsbausteins. Bei *bEn* = FALSE werden alle Ausgänge von *bQ01* bis *bQ08* auf FALSE gesetzt. Das Starten der Steuersequenz erfolgt am Eingang *bEvt01*. Nach dem Ablauf des Timers *nDlyOn01,* siehe Parameter, wird der zugehörige Ausgang *bQ01* gesetzt. Weitere Stufen schalten nach einer steigenden Flanke an den Eingängen *bEvt02* bis *bEvt08* jeweils Zeitverzögert über die Timer *nDlyOn02* bis *nDlyOn08* zu. Wird *bEvt01* bei hochgelaufener Steuerkette FALSE dann schaltet die Steuersequenz in umgekehrter Reihenfolge zurück. Das Abschalten der Ausgänge wird durch die Timer *nDlyOff01* bis *nDlyOff08*, siehe Parameter, verzögert.

Die Ausgänge *bUp* und *bDwn* zeigen an, ob sich die Steuerkette im aufsteigenden oder fallenden Zustand befindet. Die Variable *nActvEvt* zeigt an in welchen Schritt sich die Steuerkette aktuell befindet. Dabei bedeutet "0", dass die Schrittkette nicht aktiv ist.

Der Ausgang *nRemTiDlyOn* zeigt beim Hochschalten der Steuerkette die verbleibende Zeit bis zum nächsten Schritt an. Der Ausgang *nRemTiDlyOff* zeigt beim Runterschalten der Steuerkette die verbleibende Zeit bis zum Umschalten in den nächstniedrigeren Schritt an.

Beispiel



- t0 Einschalten der Schrittkette
- t1 Einschalten Schritt 1 *nDlyOn01* = t1 - t0
- t2 Ereignis Freigabe Schritt 2, Einschalten Schritt 2, *nDlyOn02* = 0
- t3 Ereignis Freigabe Schritt 3, Einschalten Schritt 3, *nDlyOn03* = 0
- t4 Ereignis Freigabe Schritt 4, Einschalten Schritt 4, *nDlyOn04* = 0
- t5 Ereignis Freigabe Schritt 5, Einschalten Schritt 5, *nDlyOn05* = 0
- t6 Abschalten der Schrittkette, Abschalten Schritt 5, Abschalten Schritt 4; *nDlyOff05* = 0, *nDlyOff04* = 0
- t7 Abschalten Schritt 3, *nDlyOff03* = t7 -t6
- t8 Abschalten Schritt 2, *nDlyOff02* = t8 -t7
- t9 Abschalten Schritt 1, *nDlyOff01* = t9 -t8

EIngänge

```
VAR_INPUT
  bEn             : BOOL;
  bEvt01          : BOOL;
  nDlyOn01        : UDINT;
  nDlyOff01       : UDINT;
  bEvt02          : BOOL;
  nDlyOn02        : UDINT;
  nDlyOff02       : UDINT;
  bEvt03          : BOOL;
  nDlyOn03        : UDINT;
  nDlyOff03       : UDINT;
  bEvt04          : BOOL;
  nDlyOn04        : UDINT;
  nDlyOff04       : UDINT;
  bEvt05          : BOOL;
  nDlyOn05        : UDINT;
  nDlyOff05       : UDINT;
  bEvt06          : BOOL;
  nDlyOn06        : UDINT;
  nDlyOff06       : UDINT;
```

```
  bEvt07          : BOOL;
  nDlyOn07        : UDINT;
  nDlyOff07       : UDINT;
  bEvt08          : BOOL;
  nDlyOn08        : UDINT;
  nDlyOff08       : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Bausteinfreigabe
bEvt01…08	BOOL	Einschaltbefehl für Schritt 1 bis 8.
nDlyOn01…08	UDINT	Einschaltverzögerung für Ausgang *bQ01…08* [s]
nDlyOff01…08	UDINT	Ausschaltverzögerung für Ausgang *bQ01…08* [s]

Ausgänge

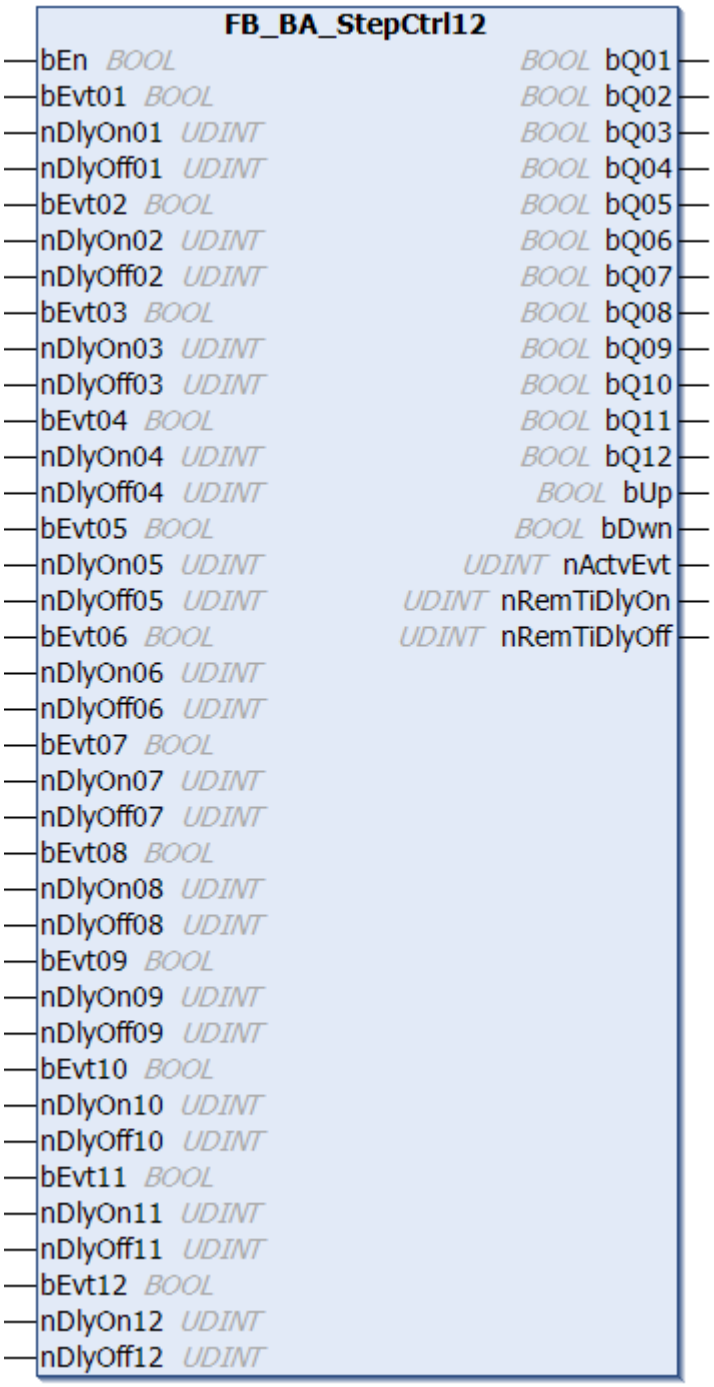
```
VAR_OUTPUT
  bQ01                : BOOL;
  bQ02                : BOOL;
  bQ03                : BOOL;
  bQ04                : BOOL;
  bQ05                : BOOL;
  bQ06                : BOOL;
  bQ07                : BOOL;
  bQ08                : BOOL;
  bUp                 : BOOL;
  bDwn                : BOOL;
  nActvEvt            : UDINT;
  nRemTiDlyOn         : UDINT;
  nRemTiDlyOff        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ01...bQ08	BOOL	Schritt 1 bis 8 Ein
bUp	BOOL	Steuerkette ist im aufsteigenden Zustand.
bDwn	BOOL	Steuerkette ist im fallenden Zustand.
nActvEvt	UDINT	Aktiver Schritt, Anzeige 0…8, "0" bedeutet nicht aktive Schrittkette.
nRemTiDlyOn	UDINT	Restzeit bis zum Hochschalten in den nächsten Schritt [s].
nRemTiDlyOff	UDINT	Restzeit bis zum Runterschalten in den vorherigen Schritt [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.9 FB_BA_StepCtrl12

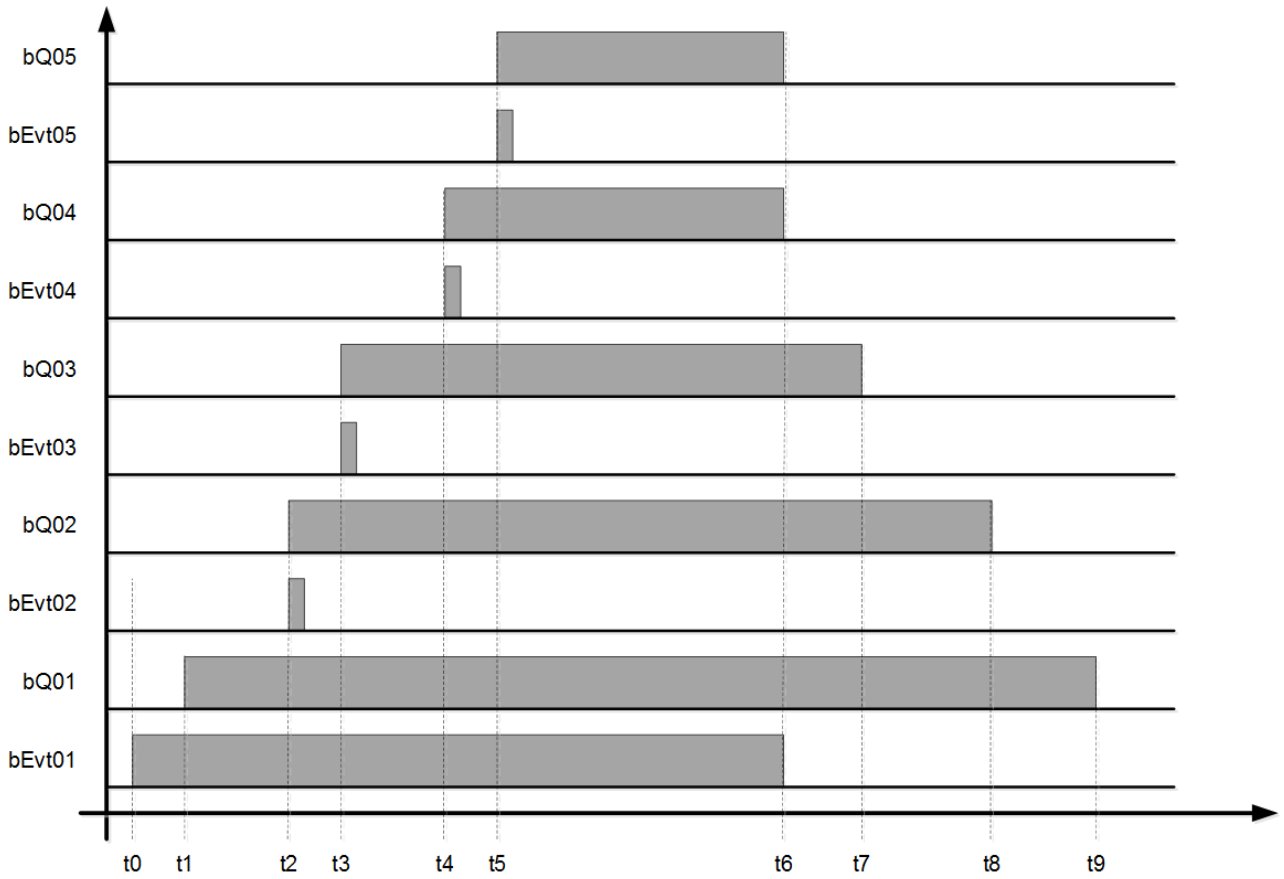


Der Funktionsbaustein FB_BA_StepCtrl12 dient zur Ausgabe sequenziell folgender Steuerungsbefehle. Eine typische Anwendung dieses Bausteins ist der Startvorgang einer raumlufttechnischen Anlage. *bEn* ist für die generelle Freigabe des Funktionsbausteins. Bei *bEn* = FALSE werden alle Ausgänge von *bQ01* bis *bQ12* auf FALSE gesetzt. Das Starten der Steuersequenz erfolgt am Eingang *bEvt01*. Nach dem Ablauf des Timers *nDlyOn01,* siehe Parameter, wird der zugehörige Ausgang *bQ01* gesetzt. Weitere Stufen schalten nach einer steigenden Flanke an den Eingängen *bEvt02* bis *bEvt12* jeweils zeitverzögert über die Timer *nDlyOn02* bis *nDlyOn12* zu. Wird *bEvt01* bei hochgelaufener Steuerkette FALSE dann schaltet die Steuersequenz in umgekehrter Reihenfolge zurück. Das Abschalten der Ausgänge wird durch die Timer *nDlyOff01* bis *nDlyOff12*, siehe Parameter, verzögert.

Die Ausgänge *bUp* und *bDwn* zeigen an, ob sich die Steuerkette im aufsteigenden oder fallenden Zustand befindet. Die Variable *nActvEvt* zeigt an in welchen Schritt sich die Steuerkette aktuell befindet. Dabei bedeutet "0", dass die Schrittkette nicht aktiv ist.

Der Ausgang *nRemTiDlyOn* zeigt beim Hochschalten der Steuerkette die verbleibende Zeit bis zum nächsten Schritt an. Der Ausgang *nRemTiDlyOff* zeigt beim Runterschalten der Steuerkette die verbleibende Zeit bis zum Umschalten in den nächst-niedrigeren Schritt an.

Beispiel



- t0 Einschalten der Schrittkette
- t1 Einschalten Schritt 1 *nDlyOn01* = t1 - t0
- t2 Ereignis Freigabe Schritt 2, Einschalten Schritt 2, *nDlyOn02* = 0
- t3 Ereignis Freigabe Schritt 3, Einschalten Schritt 3, *nDlyOn03* = 0
- t4 Ereignis Freigabe Schritt 4, Einschalten Schritt 4, *nDlyOn04* = 0
- t5 Ereignis Freigabe Schritt 5, Einschalten Schritt 5, *nDlyOn05* = 0
- t6 Abschalten der Schrittkette, Abschalten Schritt 5, Abschalten Schritt 4; *nDlyOff05* = 0, *nDlyOff04* = 0
- t7 Abschalten Schritt 3, *nDlyOff03* = t7 -t6
- t8 Abschalten Schritt 2, *nDlyOff02* = t8 -t7
- t9 Abschalten Schritt 1, *nDlyOff01* = t9 -t8

Eingänge

```
VAR_INPUT
  bEn               : BOOL;
  bEvt01            : BOOL;
  nDlyOn01          : UDINT;
  nDlyOff01         : UDINT;
  bEvt02            : BOOL;
  nDlyOn02          : UDINT;
  nDlyOff02         : UDINT;
  bEvt03            : BOOL;
  nDlyOn03          : UDINT;
  nDlyOff03         : UDINT;
  bEvt04            : BOOL;
  nDlyOn04          : UDINT;
  nDlyOff04         : UDINT;
  bEvt05            : BOOL;
```

```
  nDlyOn05          : UDINT;
  nDlyOff05         : UDINT;
  bEvt06            : BOOL;
  nDlyOn06          : UDINT;
  nDlyOff06         : UDINT;
  bEvt07            : BOOL;
  nDlyOn07          : UDINT;
  nDlyOff07         : UDINT;
  bEvt08            : BOOL;
  nDlyOn08          : UDINT;
  nDlyOff08         : UDINT;
  bEvt09            : BOOL;
  nDlyOn09          : UDINT;
  nDlyOff09         : UDINT;
  bEvt10            : BOOL;
  nDlyOn10          : UDINT;
  nDlyOff10         : UDINT;
  bEvt11            : BOOL;
  nDlyOn11          : UDINT;
  nDlyOff11         : UDINT;
  bEvt12            : BOOL;
  nDlyOn12          : UDINT;
  nDlyOff12         : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Bausteinfreigabe
bEvt01…012	BOOL	Einschaltbefehl für Schritt 1 bis 12.
nDlyOn01…12	UDINT	Einschaltverzögerung für Ausgang *bQ01…12* [s]
nDlyOff01…12	UDINT	Ausschaltverzögerung für Ausgang *bQ01…12* [s]

Ausgänge

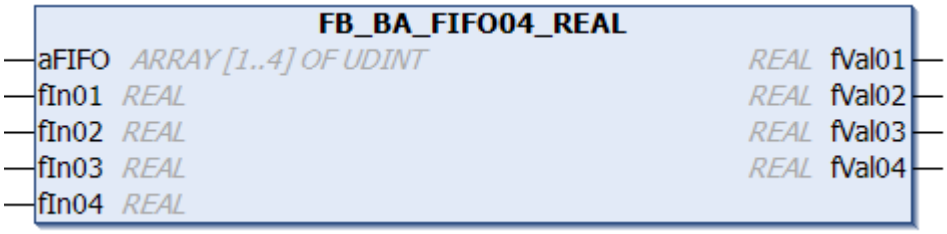
```
VAR_OUTPUT
  bQ01              : BOOL;
  bQ02              : BOOL;
  bQ03              : BOOL;
  bQ04              : BOOL;
  bQ05              : BOOL;
  bQ06              : BOOL;
  bQ07              : BOOL;
  bQ08              : BOOL;
  bQ09              : BOOL;
  bQ10              : BOOL;
  bQ11              : BOOL;
  bQ12              : BOOL;
  bUp               : BOOL;
  bDwn              : BOOL;
  nActvEvt          : UDINT;
  nRemTiDlyOn       : UDINT;
  nRemTiDlyOff      : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ01...bQ12	BOOL	Schritt 1 bis 12 Ein
bUp	BOOL	Steuerkette ist im aufsteigenden Zustand.
bDwn	BOOL	Steuerkette ist im fallenden Zustand.
nActvEvt	UDINT	Aktiver Schritt, Anzeige 0…12, "0" bedeutet nicht aktive Schrittkette.
nRemTiDlyOn	UDINT	Restzeit bis zum Hochschalten in den nächsten Schritt [s].
nRemTiDlyOff	UDINT	Restzeit bis zum Runterschalten in den vorherigen Schritt [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.10 FB_BA_FIFO04_XX



Der Funktionsbaustein FB_BA_FIFO04_XX dient zum Auswerten des FiFo-Speichers aus dem FB_BA_FIFO04 [▶ 446]. Die Eingänge werden nach FIFO-Tabelle auf die entsprechenden Ausgänge des verwendeten Bausteins *FB_BA_FIFO04_BOOL* oder *FB_BA_FIFO04_REAL* verknüpft.

Beispiel:

Im Beispiel steht in dem Array: 4,3,1,2,0,0,0,0. Als Ergebnis wird im *FB_BA_FIFO04_REAL* folgendes ausgegeben:

fIn01 auf den Ausgang *fVal04*

fIn02 auf den Ausgang f*Val03*

fIn03 auf den Ausgang f*Val01*

fIn04 auf den Ausgang f*Val02*

Eingänge

```
VAR_INPUT
  aFIFO        : Array [1..4] OF UDINT;
  fIn01…fIn04  : REAL;
END_VAR
```

Name	Typ	Beschreibung
aFIFO	ARRAY OF UDINT	Enthält die Zuordnungstabelle mit maximal acht Werten. Der erste Wert gibt an wohin der erste Eingang kopiert wird, der zweite Wert gibt an wohin der zweite Eingang kopiert wird, etc. Bei "0" wird keine Zuordnung vorgenommen.
fIn01…fIn04	REAL	Sollwerte die verknüpft werden sollen.

Ausgänge

```
VAR_OUTPUT
  fVal01…fVal04   : REAL;
END_VAR
```

Name	Typ	Beschreibung
fVal01…fVal04	REAL	Aktor Sollwert, laut FIFO-Tabelle verknüpfter Eingangswert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.11 FB_BA_FIFO08_XX



Der Funktionsbaustein FB_BA_FIFO08_XX dient zum Auswerten des FiFo-Speichers aus dem FB_BA_FIFO08 [▶ 447]. Die Eingänge werden nach FIFO-Tabelle auf die entsprechenden Ausgänge des verwendeten Bausteins *FB_BA_FIFO08_BOOL* oder *FB_BA_FIFO08_REAL* verknüpft.

Beispiel:

Im Beispiel steht in dem Array: 4,3,1,2,0,0,0,0. Als Ergebnis wird im *FB_BA_FIFO08_REAL* ausgegeben:

fIn01 auf den Ausgang f*Val04*

fIn02 auf den Ausgang f*Val03*

fIn03 auf den Ausgang f*Val01*

fIn04 auf den Ausgang f*Val02*

Eingänge

```
VAR_INPUT
  aFIFO          : Array [1..8] OF UDINT;
  fIn01…fIn08    : REAL;
END_VAR
```

Name	Typ	Beschreibung
aFIFO	ARRAY OF UDINT	Enthält die Zuordnungstabelle mit maximal acht Werten. Der erste Wert gibt an wohin der erste Eingang kopiert wird, der zweite Wert gibt an wohin der zweite Eingang kopiert wird, etc. Bei "0" wird keine Zuordnung vorgenommen.
fIn01…fIn08	REAL	Sollwerte die verknüpft werden sollen.

Ausgänge

```
VAR_OUTPUT
  fVal01…fVal08   : REAL;
END_VAR
```

Name	Typ	Beschreibung
fVal01…fVal08	REAL	Aktor Sollwert, laut FIFO-Tabelle verknüpfter Eingangswert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.3.12 LastWriterWins

6.1.2.2.3.1.5.3.12.1 FB_BA_LastWriterWins_R04



Der Funktionsbaustein schaltet gibt jeweils denjenigen Eingangswert *fInxx* am Ausgang fQ aus, welcher sich als letztes geändert hat.

Am Ausgang *nActInput* wird angezeigt, welcher Eingang gerade ausgegeben wird.

Solange sich am Baustein nichts ändert (die SPS ist gerade erst gestartet) und alle Eingänge auf "0" stehen, so wird an den Ausgängen *fQ* und *nActInput* ebenfalls "0" ausgegeben.

Ändern sich in einem SPS-Zyklus mehrere Eingänge gleichzeitig, so hat der Eingang mit der geringeren Ordnungszahl Priorität, z.B.: *fIn01* vor *fIn02*.

Eingänge

```
VAR_INPUT
  aFIFO          : Array [1..8] OF UDINT;
  fIn01…fIn08    : REAL;
END_VAR
```

Name	Typ	Beschreibung
fIn01…fIn04	REAL	Eingangswerte.

Ausgänge

```
VAR_OUTPUT
  fQ           : REAL;
  nActInput    : USINT;
END_VAR
```

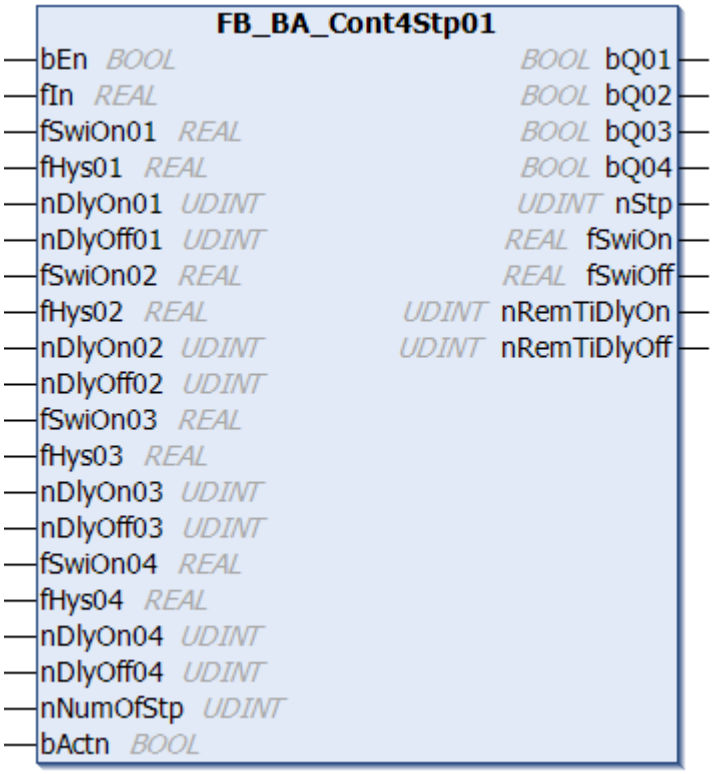
Name	Typ	Beschreibung
fQ	REAL	Ausgangswert.
nActInput	USINT	Zeigt an, welcher Eingangswert gerade ausgegeben wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.55	Tc3_BA2 ab v5.3.19.0

6.1.2.2.3.1.5.4 Hysteresis 2-Point-Control

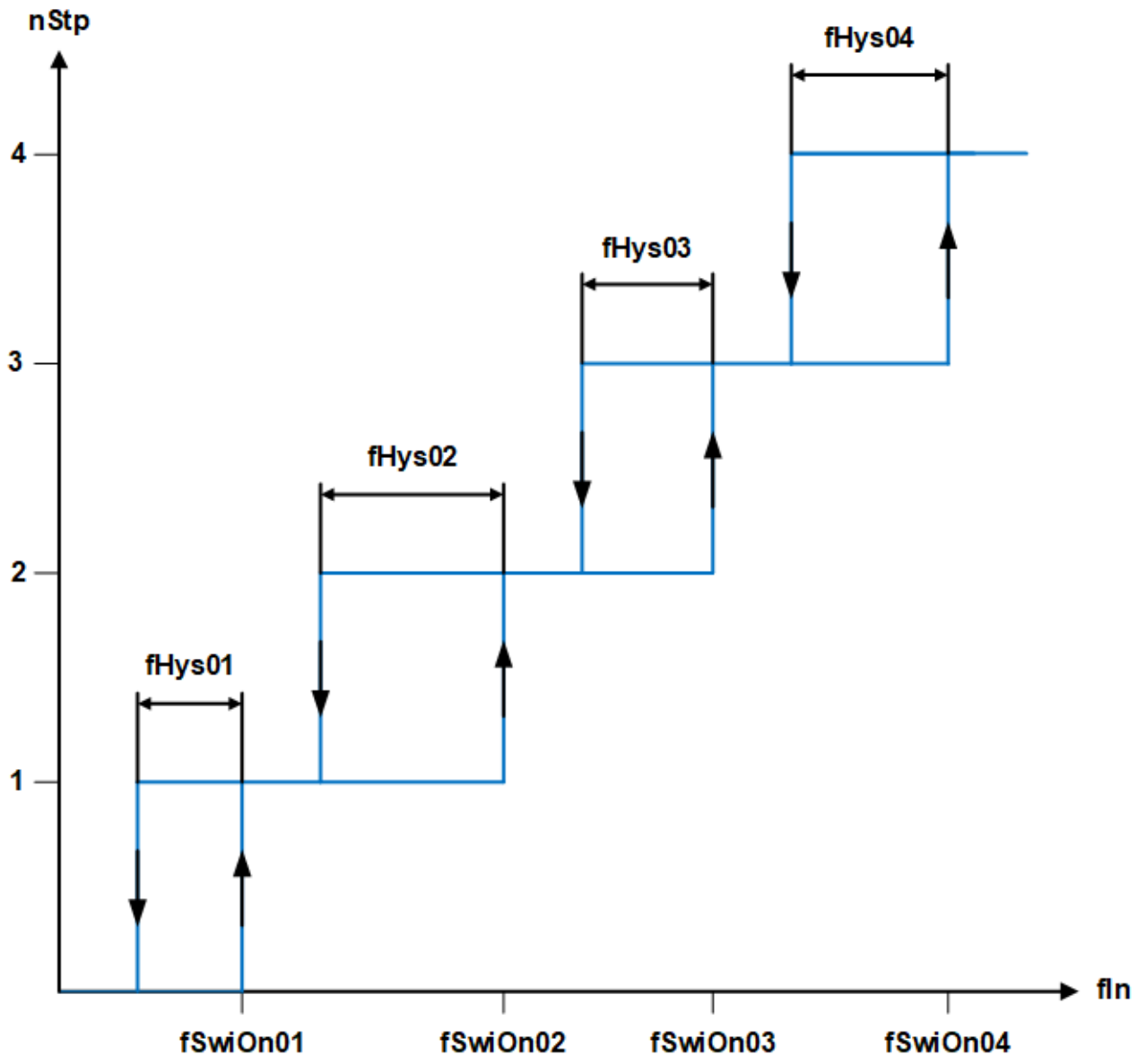
6.1.2.2.3.1.5.4.1 FB_BA_Cont4Stp01



Der Funktionsbaustein FB_BA_Cont4Stp01 ermittelt abhängig vom Eingangssignal die resultierenden Schaltstufen eines mehrstufigen Aggregats.
Es sind vier Einschaltschwellen und vier Hysteresen parametrierbar.

Diagramm 01

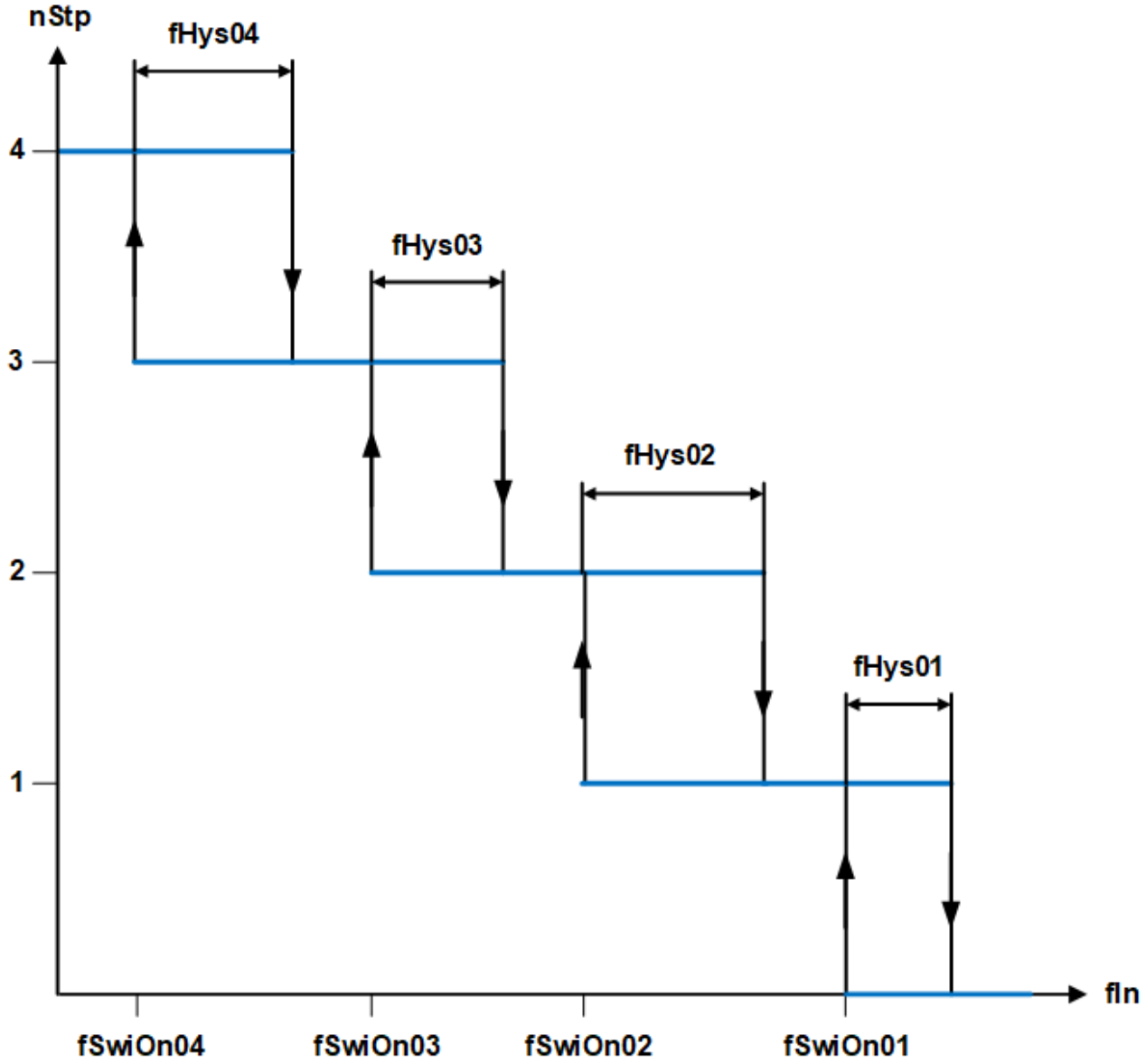
Wirksinn Parameter *bActn* = FALSE = Reverse = Heizen



nStp	nNumOf Stp	fSwiOn	fSwiOff	nRem-TiDlyOn	nRem-TiDlyOff	bQ01	bQ02	bQ03	bQ04
0	0	fSwiOn01	fSwiOn01 - fHys01	nDlyOn01	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	fSwiOn02	fSwiOn01 - fHys01	nDlyOn02	nDlyOff01	TRUE	FALSE	FALSE	FALSE
2	>= 2	fSwiOn03	fSwiOn02 - fHys02	nDlyOn03	nDlyOff02	TRUE	TRUE	FALSE	FALSE
3	>= 3	fSwiOn04	fSwiOn03 - fHys03	nDlyOn04	nDlyOff03	TRUE	TRUE	TRUE	FALSE
4	>= 4	fSwiOn04	fSwiOn04 - fHys04	0	nDlyOff04	TRUE	TRUE	TRUE	TRUE

Diagramm 02

Wirksinn Parameter *bActn* =TRUE = Direct = Kühlen



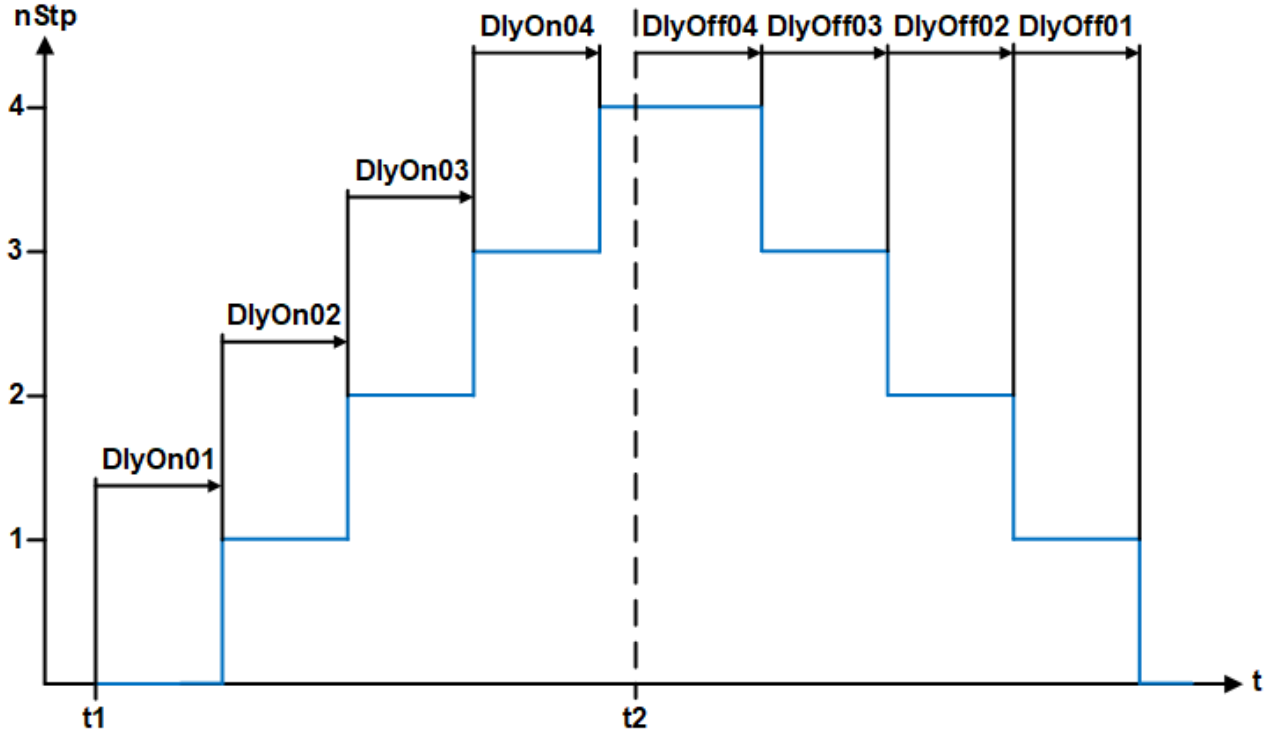
nStp	nNumOf Stp	fSwiOn	fSwiOff	nRem-TiDlyOn	nRem-TiDlyOff	bQ01	bQ02	bQ03	bQ04
0	0	fSwiOn01	fSwiOn01 + fHys01	nDlyOn01	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	fSwiOn02	fSwiOn01 + fHys01	nDlyOn02	nDlyOff01	TRUE	FALSE	FALSE	FALSE
2	>= 2	fSwiOn03	fSwiOn02 + fHys02	nDlyOn03	nDlyOff02	TRUE	TRUE	FALSE	FALSE
3	>= 3	fSwiOn04	fSwiOn03 + fHys03	nDlyOn04	nDlyOff03	TRUE	TRUE	TRUE	FALSE
4	4	fSwiOn04	fSwiOn04 + fHys04	0	nDlyOff04	TRUE	TRUE	TRUE	TRUE

Diagramm 03

Zeitverhalten der Ein- und Ausschaltverzögerungen

Zum Zeitpunkt t1 springt *fIn* von *fSwiOn01* auf *fSwiOn04*

Zum Zeitpunkt t2 springt *fIn* von *fSwiOn04* auf *fSwiOn01 – fHys01*



Eingänge

```
VAR_INPUT
  bEn             : BOOL;
  fIn             : REAL;
  fSwiOn01        : REAL
  fHys01          : REAL;
  nDlyOn01        : UDINT;
  nDlyOff01       : UDINT;
  fSwiOn02        : REAL;
  fHys02          : REAL;
  nDlyOn02        : UDINT;
  nDlyOff02       : UDINT;
  fSwiOn03        : REAL;
  fHys03          : REAL;
  nDlyOn03        : UDINT;
  nDlyOff03       : UDINT;
  fSwiOn04        : REAL;
  fHys04          : REAL;
  nDlyOn04        : UDINT;
  nDlyOff04       : UDINT;
  nNumOfStp       : UDINT;
  bActn           : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Generelle Freigabe des Funktionsbausteins. Ist *bEn* = FALSE, so sind sämtliche Ausgänge auf 0 gesetzt.
fIn	REAL	Eingangswert von dem der Schaltzustand abgeleitet wird.
fSwiOn01	REAL	Einschaltpunkt Stufe 01
fHys01	REAL	Absolutwert Hysterese Stufe 01
nDlyOn01	UDINT	Einschaltverzögerung Stufe 01
nDlyOff01	UDINT	Ausschaltverzögerung Stufe 01
fSwiOn02	REAL	Einschaltpunkt Stufe 02
fHys02	REAL	Absolutwert Hysterese Stufe 02
nDlyOn02	UDINT	Einschaltverzögerung Stufe 02
nDlyOff02	UDINT	Ausschaltverzögerung Stufe 02
fSwiOn03	REAL	Einschaltpunkt Stufe 03
fHys03	REAL	Absolutwert Hysterese Stufe 03
nDlyOn03	UDINT	Einschaltverzögerung Stufe 03
nDlyOff03	UDINT	Ausschaltverzögerung Stufe 03
fSwiOn04	REAL	Einschaltpunkt Stufe 04
fHys04	REAL	Absolutwert Hysterese Stufe 04
nDlyOn04	UDINT	Einschaltverzögerung Stufe 04
nDlyOff04	UDINT	Ausschaltverzögerung Stufe 04
nNumOfStp	UDINT	Eingabe der Anzahl der Stufen, die benötigt werden. Die Eingabe ist begrenzt von 0 bis 4.
bActn	BOOL	Eingangsvariable mit der der Wirksinn des Stufenschalters bestimmt wird. TRUE = Direct = Kühlen; FALSE = Reverse = Heizen

Ausgänge

```
VAR_OUTPUT
  bQ01               : BOOL;
  bQ02               : BOOL;
  bQ03               : BOOL;
  bQ04               : BOOL;
  nStp               : UDINT;
  fSwiOn             : REAL;
  fSwiOff            : REAL;
  nRemTiDlyOn        : UDINT;
  nRemTiDlyOff       : UDINT;
END_VAR
```

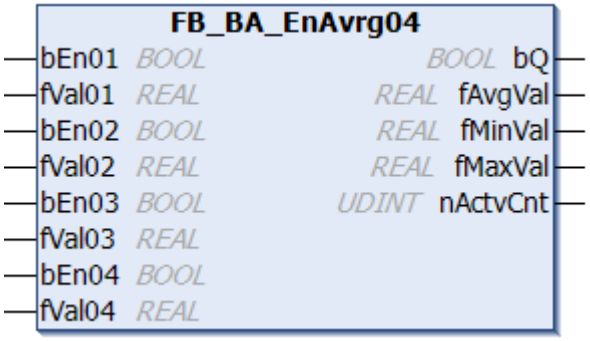
Name	Typ	Beschreibung
bQ01	BOOL	Anzeige des Status Stufe 01 TRUE = EIN; FALSE = AUS nStp >= 1
bQ02	BOOL	Anzeige des Status Stufe 02 TRUE = EIN; FALSE = AUS nStp >= 2
bQ03	BOOL	Anzeige des Status Stufe 03 TRUE = EIN; FALSE = AUS nStp >= 3
bQ04	BOOL	Anzeige des Status Stufe 04 TRUE = EIN; FALSE = AUS nStp >= 4
nStp	UDINT	Anzeige in welcher Stufe sich der Stufenschalter befindet
fSwiOn	REAL	Anzeige des nächsten Einschaltpunktes
fSwiOff	REAL	Anzeige des nächsten Ausschaltpunktes
nRemTiDlyOn	UDINT	Ist der Einschaltpunkt für das Weiterschalten auf die nächste Stufe erfüllt, dann wird hier der Ablauf der Einschaltverzögerungszeit angezeigt.
nRemTiDlyOff	UDINT	Ist der Ausschaltpunkt für das Herunterschalten auf die nächste Stufe erfüllt, dann wird hier der Ablauf der Ausschaltverzögerungszeit angezeigt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5 Mathematics

6.1.2.2.3.1.5.5.1 FB_BA_EnAvrg0X



Der Funktionsbaustein berechnet den arithmetischen Mittelwert aus den freigegebenen Eingangswerten. Der Baustein ist für die Varianten von 2, 4 und 8 Eingangswerten vorhanden. Die folgende Dokumentation bezieht sich auf den *FB_BA_EnAvrg04*.

Eingänge

```
VAR_INPUT
  bEn01…bEn04       : BOOL;
  fVal01…fVal04     : REAL;
END_VAR
```

Name	Typ	Beschreibung
bEn01…bEn04	BOOL	Generelle Freigabe einer Mittelwertberechnung. Ist *bEn0x* = FALSE, so wird der entsprechende Eingangswert nicht zur Mittelwertbildung einbezogen.
fVal01…fVal04	REAL	An die Variablen *fVal0* bis *fVal0x* werden die Werte angelegt aus denen der Mittelwert berechnet werden soll.

Ausgänge

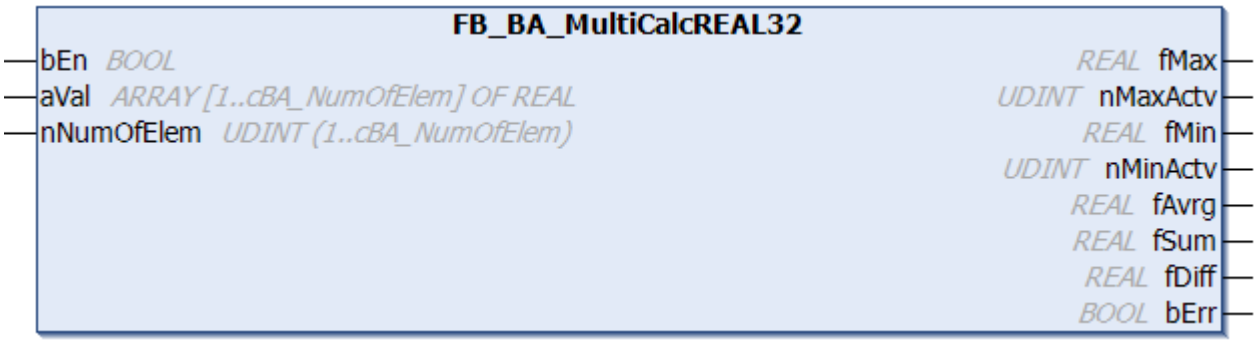
```
VAR_OUTPUT
  fAvgVal      : REAL;
  fMinVal      : REAL;
  fMaxVal      : REAL;
  nActvCnt     : UDINT;
END_VAR
```

Name	Typ	Beschreibung
fAvgVal	REAL	Errechneter arithmetischer Mittelwert.
fMinVal	REAL	Kleinster Eingangswert.
fMaxVal	REAL	Größter Eingangswert.
nActvCnt	UDINT	Anzahl der berücksichtigten Eingangswerte zur Mittelwertbildung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5.2 FB_BA_MultiCalcREAL32



Der Multi-Kalkulations-Baustein FB_BA_MultiCalcREAL32 existiert für den Variablentypen REAL.

Der Funktionsbaustein ermittelt im aktivierten Zustand (*bEn* = TRUE) aus den Eingangswerten *aVal* folgendes:

- den Maximalwert aller Eingänge *fMax*
- den Eingang, an dem dieser Maximalwert anliegt *nMaxActv*
- den Minimalwert aller Eingänge *fMin*
- den Eingang, an dem dieser Minimalwert anliegt *nMinActv*
- den Mittelwert aller Eingänge *fAvrg*
- die Summe aller Eingänge *fSum*
- die Differenz zwischen dem Maximal- und Minimalwert *fDiff*

Sollen nicht alle Eingänge zur Berechnung kommen, so lässt sich die Anzahl durch einen Eintrag an *nNumOfElem* einschränken: mit *nNumOfElem* = 6 beispielsweise werden die Berechnungen nur für die ersten sechs Einträge von *aVal* durchgeführt.
Ein Eintrag größer als 32 wird automatisch auf 32 limitiert, ein Eintrag kleiner als 1 automatisch auf 1.

Beispiel:

Eingänge	Ausgang
bEn = TRUE	fMax = 32
aVal[1] = 32	nMaxActv = 1
aVal[2] = 17	fMin = 5
aVal[3] = 5	nMinActv = 3
aVal[4] = 9	fAvrg = 18.5
aVal[5] = 16	fSum = 111
aVal[6] = 32	fDiff = 27
aVal[7] = 25	
aVal[8] = 44	
nNumOfElem = 6	

Bei *bEn* = FALSE wird an allen Ausgängen 0 ausgegeben.

Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  aVal         : ARRAY [1..???] of REAL;
  nNumOfElem   : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung der Bausteinfunktion.
aVal	ARRAY OF REAL	Feld mit den zu berechnenden Werten.
nNumOfElem	UDINT	Anzahl der Eingangswerte, mit denen gerechnet wird.

Ausgänge

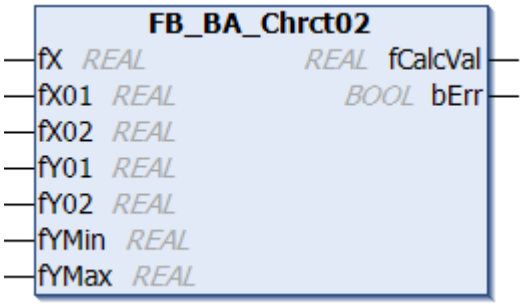
```
VAR_OUTPUT
  fMax         : REAL;
  nMaxActv     : UDINT;
  fMin         : REAL;
  nMinActv     : UDINT;
  fAvrg        : REAL;
  fSum         : REAL;
  fDiff        : REAL;
  bErr         : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fMax	REAL	Maximalwert von allen Eingängen.
nMaxActv	UDINT	Eingang, an dem der Maximalwert anliegt.
fMin	REAL	Minimalwert von allen Eingängen.
nMinActv	UDINT	Eingang, an dem der Minimalwert anliegt.
fAvrg	REAL	Mittelwert aller Eingänge
fSum	REAL	Summe aller Eingänge
fDiff	REAL	Differenz zwischen dem Maximal- und Minimalwert.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

Voraussetzungen

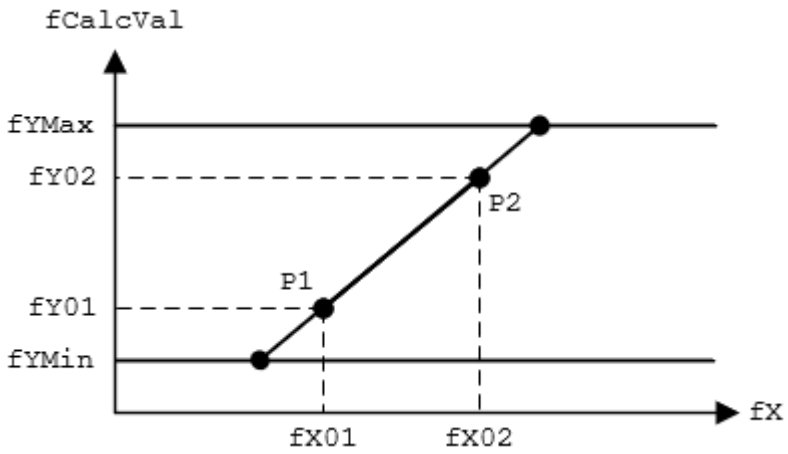
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5.3 FB_BA_Chrct02



Der Funktionsbaustein FB_BA_Chrct02 stellt eine Linear-Interpolation mit 2 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*fX01/fY01*] und [*fX02/fY02*] bestimmt.

Der errechnete Ausgangswert *fCalcVal* wird durch *fYMin* bzw. *fYMax* limitiert.



Fehlerbehandlung

Die Eingabewerte für *fX0[n+1]* müssen immer größer als von *fX0[n]* sein.

Im Fehlerfall zeigt die Variable *bErr* dieses an. Das Property *ErrorDescription* gibt eine detaillierte Beschreibung heraus.

Der Eingabewerte für *fYMin* darf nicht größer sein als *fYMax*.

Eingänge

```
VAR_INPUT
  fX     : REAL;
  fX01   : REAL;
  fX02   : REAL;
  fY01   : REAL;
  fY02   : REAL;
  fYMin  : REAL;
  fYMax  : REAL;
END_VAR
```

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.
fX0N	REAL	X-Wert für Stützpunkt PN.
fY0N	REAL	Y-Wert für Stützpunkt PN.
fYMin	REAL	Unterer Grenzwert von *fCalcVal*. Die Eingabe für den unteren Grenzwert ist auf den globalen Parameter BAComn_Global.fMinReal aus der Tc3_BA2_Common Bibliothek limitiert.
fYMax	REAL	Oberer Grenzwert von *fCalcVal*. Die Eingabe für den oberen Grenzwert ist auf den globalen Parameter BAComn_Global.fMaxReal aus der Tc3_BA2_Common Bibliothek limitiert.

Ausgänge

```
VAR_OUTPUT
  fCalcVal   : REAL;
  bErr       : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fCalcVal	REAL	Errechneter Ausgangswert der Kennlinie.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

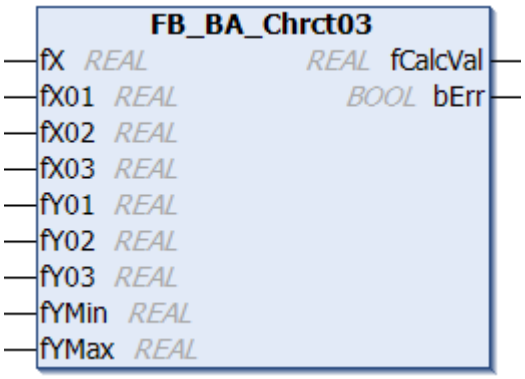
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ErrorDescription	T_MaxString	Get	Gibt eine detaillierte Beschreibung von Fehlern heraus, siehe Fehlerbehandlung [▶ 466].

Voraussetzungen

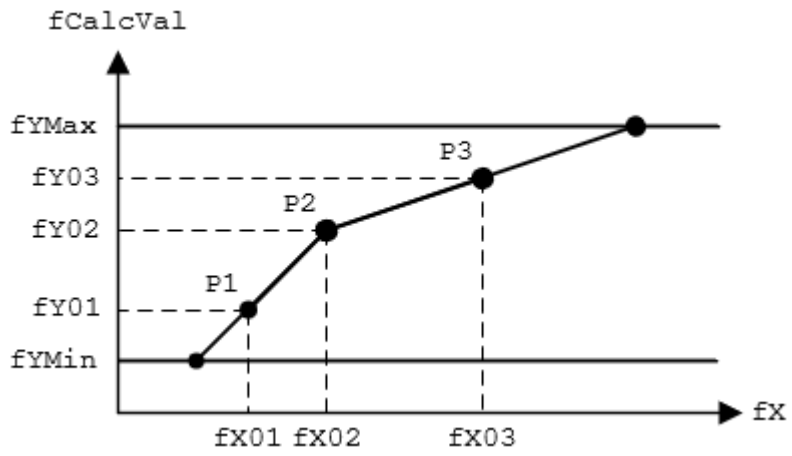
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5.4 FB_BA_Chrtc03



Der Funktionsbaustein FB_BA_Chrct03 stellt eine Linear-Interpolation mit 3 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*fX01/fY01*], [*fX02/fY02*] und [*fX03/fY03*] bestimmt.

Der errechnete Ausgangswert *fCalcVal* wird durch *fYMin* bzw. *fYMax* limitiert.



Fehlerbehandlung

Die Eingabewerte für *fX0[n+1]* müssen immer größer als von *fX0[n]* sein.

Im Fehlerfall zeigt die Variable *bErr* dieses an. Das Property *ErrorDescription* gibt eine detaillierte Beschreibung heraus.

Der Eingabewerte für *fYMin* darf nicht größer sein als *fYMax*.

Eingänge

```
VAR_INPUT
  fX     : REAL;
  fX01   : REAL;
  fX02   : REAL;
  fX03   : REAL;
  fY01   : REAL;
  fY02   : REAL;
  fY03   : REAL;
  fYMin  : REAL;
  fYMax  : REAL;
END_VAR
```

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.
fX0N	REAL	X-Wert für Stützpunkt PN.
fY0N	REAL	Y-Wert für Stützpunkt PN.
fYMin	REAL	Unterer Grenzwert von *fCalcVal*. Die Eingabe für den unteren Grenzwert ist auf den globalen Parameter BAComn_Global.fMinReal aus der Tc3_BA2_Common Bibliothek limitiert.
fYMax	REAL	Oberer Grenzwert von *fCalcVal*. Die Eingabe für den oberen Grenzwert ist auf den globalen Parameter BAComn_Global.fMaxReal aus der Tc3_BA2_Common Bibliothek limitiert.

Ausgänge

```
VAR_OUTPUT
  fCalcVal   : REAL;
  bErr       : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fCalcVal	REAL	Errechneter Ausgangswert der Kennlinie.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

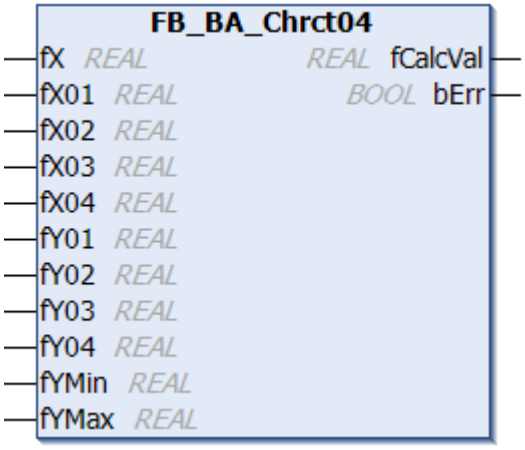
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ErrorDescription	T_MaxString	Get	Gibt eine detaillierte Beschreibung von Fehlern heraus, siehe Fehlerbehandlung [▶ 468].

Voraussetzungen

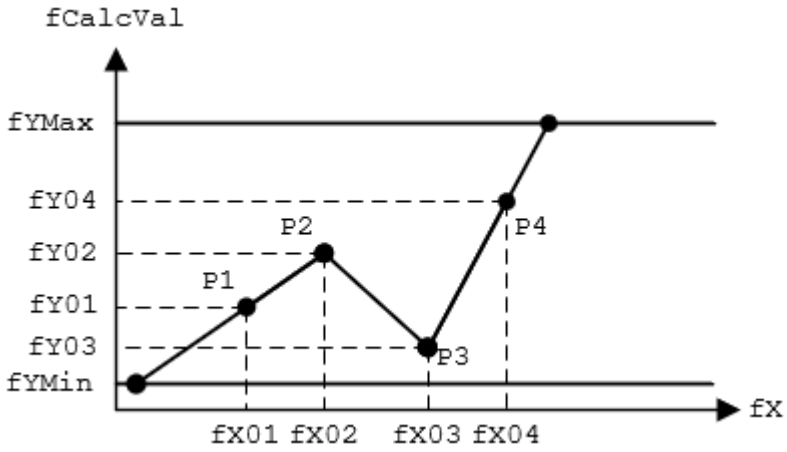
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5.5 FB_BA_Chrct04



Der Funktionsbaustein FB_BA_Chrct04 stellt eine Linear-Interpolation mit 4 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*fX01/fY01*], [*fX02/fY02*], [*fX03/fY03*] und [*fX04/fY04*] bestimmt.

Der errechnete Ausgangswert *fCalcVal* wird durch *fYMin* bzw. *fYMax* limitiert.



Fehlerbehandlung

Die Eingabewerte für *fX0[n+1]* müssen immer größer als von *fX0[n]* sein.

Im Fehlerfall zeigt die Variable *bErr* dieses an. Das Property *ErrorDescription* gibt eine detaillierte Beschreibung heraus.

Der Eingabewerte für *fYMin* darf nicht größer sein als *fYMax*.

Eingänge

```
VAR_INPUT
  fX      : REAL;
  fX01    : REAL;
  fX02    : REAL;
  fX03    : REAL;
  fX04    : REAL;
  fY01    : REAL;
  fY02    : REAL;
  fY03    : REAL;
  fY04    : REAL;
  fYMin   : REAL;
  fYMax   : REAL;
END_VAR
```

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.
fX0N	REAL	X-Wert für Stützpunkt PN.
fY0N	REAL	Y-Wert für Stützpunkt PN.
fYMin	REAL	Unterer Grenzwert von *fCalcVal*. Die Eingabe für den unteren Grenzwert ist auf den globalen Parameter BAComn_Global.fMinReal aus der Tc3_BA2_Common Bibliothek limitiert.
fYMax	REAL	Oberer Grenzwert von *fCalcVal*. Die Eingabe für den oberen Grenzwert ist auf den globalen Parameter BAComn_Global.fMaxReal aus der Tc3_BA2_Common Bibliothek limitiert.

Ausgänge

```
VAR_OUTPUT
  fCalcVal    : REAL;
  bErr        : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fCalcVal	REAL	Errechneter Ausgangswert der Kennlinie.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

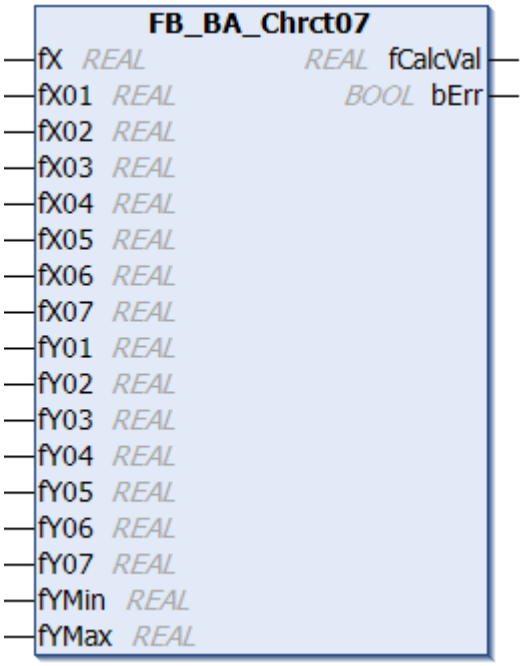
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ErrorDescription	T_MaxString	Get	Gibt eine detaillierte Beschreibung von Fehlern heraus, siehe Fehlerbehandlung [▶ 469].

Voraussetzungen

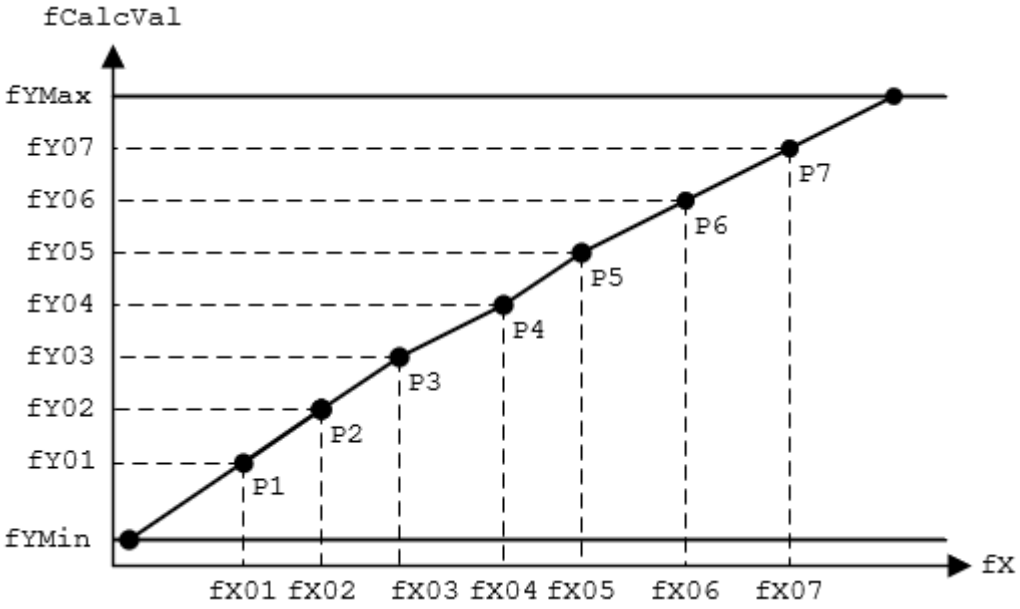
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5.6 FB_BA_Chrct07



Der Funktionsbaustein FB_BA_Chrct07 stellt eine Linear-Interpolation mit 7 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*fX01/fY01*], [*fX02/fY02*], [*fX03/fY03*] … [*fX07/fY07*] bestimmt.

Der errechnete Ausgangswert *fCalcVal* wird durch *fYMin* bzw. *fYMax* limitiert.



Fehlerbehandlung

Die Eingabewerte für *fX0[n+1]* müssen immer größer als von *fX0[n]* sein.

Im Fehlerfall zeigt die Variable *bErr* dieses an. Das Property *ErrorDescription* gibt eine detaillierte Beschreibung heraus.

Der Eingabewerte für *fYMin* darf nicht größer sein als *fYMax*.

Eingänge

```
VAR_INPUT
  fX     : REAL;
  fX01   : REAL;
  fX02   : REAL;
  fX03   : REAL;
  fX04   : REAL;
  fX05   : REAL;
  fX06   : REAL;
  fX07   : REAL;
  fY01   : REAL;
  fY02   : REAL;
  fY03   : REAL;
  fY04   : REAL;
  fY05   : REAL;
  fY06   : REAL;
  fY07   : REAL;
  fYMin  : REAL;
  fYMax  : REAL;
END_VAR
```

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.
fX0N	REAL	X-Wert für Stützpunkt PN.
fY0N	REAL	Y-Wert für Stützpunkt PN.
fYMin	REAL	Unterer Grenzwert von *fCalcVal*. Die Eingabe für den unteren Grenzwert ist auf den globalen Parameter BAComn_Global.fMinReal aus der Tc3_BA2_Common Bibliothek limitiert.
fYMax	REAL	Oberer Grenzwert von *fCalcVal*. Die Eingabe für den oberen Grenzwert ist auf den globalen Parameter BAComn_Global.fMaxReal aus der Tc3_BA2_Common Bibliothek limitiert.

Ausgänge

```
VAR_OUTPUT
  fCalcVal   : REAL;
  bErr       : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fCalcVal	REAL	Errechneter Ausgangswert der Kennlinie.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

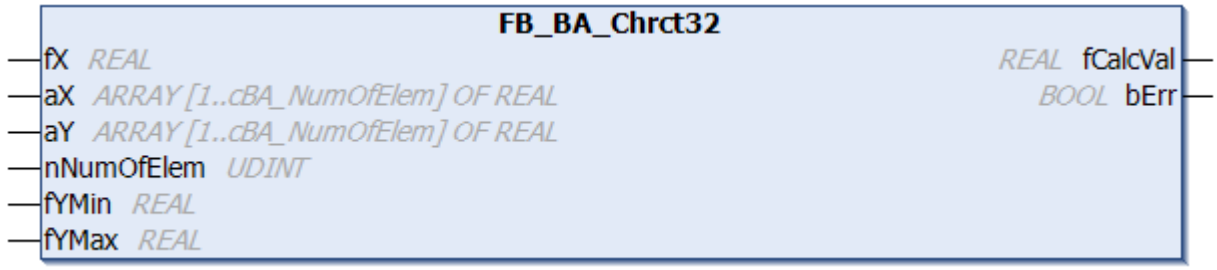
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ErrorDescription	T_MaxString	Get	Gibt eine detaillierte Beschreibung von Fehlern heraus, siehe Fehlerbehandlung [▶ 471].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

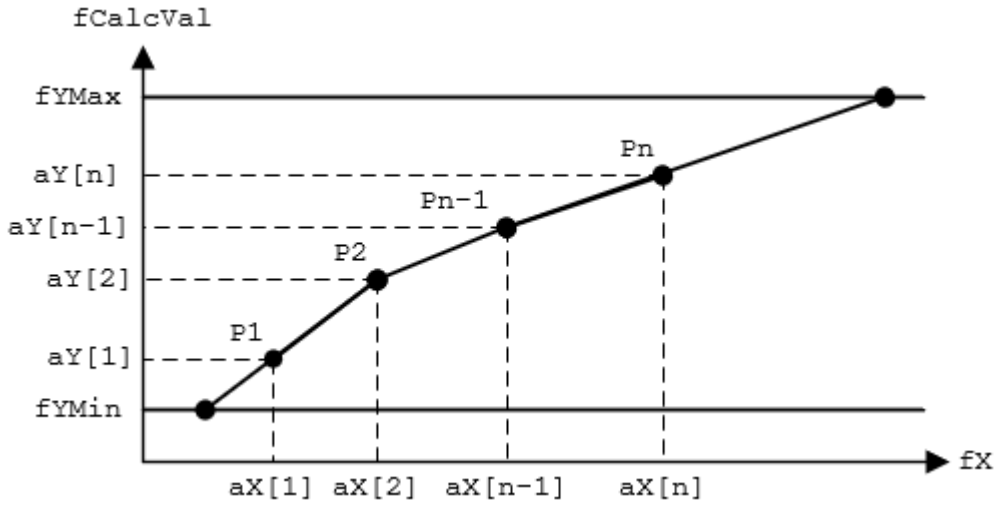
6.1.2.2.3.1.5.5.7 FB_BA_Chrct32



Der Funktionsbaustein FB_BA_Chrct32 stellt eine Linear-Interpolation mit bis zu 32 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Stützstellen werden hier, anders als bei den "kleineren" Interpolationsbausteinen FB_BA_Chrct02 [▶ 466], FB_BA_Chrct04 [▶ 469] und FB_BA_Chrct07 [▶ 471] der Übersicht halber über Feldvariablen [*aX[1]/aY[1]* bis [*aX[n]/aY[n]*] bestimmt.

Die Eingangsvariable *nNumOfElem* bestimmt die Anzahl der Stützstellen aus den Feldbereichen *aX/aY*.

Der errechnete Ausgangswert *fCalcVal* wird durch *fYMin* bzw. *fYMax* limitiert.



Fehlerbehandlung

Die Eingabewerte für *aX[n+1]* müssen immer größer als von *aX[n]* sein.

Im Fehlerfall zeigt die Variable *bErr* dieses an. Das Property *ErrorDescription* gibt eine detaillierte Beschreibung heraus.

Der Eingabewerte für *fYMin* darf nicht größer sein als *fYMax*.

Im Fehlerfall zeigt die Variable *bErr* dieses an. Das Property *ErrorDescription* gibt eine detaillierte Beschreibung heraus.

Eingänge

```
VAR_INPUT
  fX           : REAL;
  aX           : ARRAY [1..FB_BA_Chrct32.cBA_NumOfElem] OF REAL;
  aY           : ARRAY [1..FB_BA_Chrct32.cBA_NumOfElem] OF REAL;
  nNumOfElem   : DINT(2..32);
  fYMin        : REAL;
  fYMax        : REAL;
END_VAR
```

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.
aX	ARRAY OF REAL	X-Wert für die Stützpunkte.
aY	ARRAY OF REAL	Y-Wert für die Stützpunkte.
nNumOfElem	DINT	Anzahl der Stützstellen. Intern limitiert auf den Bereich 2 … 32.
fYMin	REAL	Unterer Grenzwert von *fCalcVal*. Die Eingabe für den unteren Grenzwert ist auf den globalen Parameter BAComn_Global.fMinReal aus der Tc3_BA2_Common Bibliothek limitiert.
fYMax	REAL	Oberer Grenzwert von *fCalcVal*. Die Eingabe für den oberen Grenzwert ist auf den globalen Parameter BAComn_Global.fMaxReal aus der Tc3_BA2_Common Bibliothek limitiert.

Ausgänge

```
VAR_OUTPUT
  fCalcVal    : REAL;
  bErr        : BOOL;
END_VAR
```

Name	Typ	Beschreibung
fCalcVal	REAL	Errechneter Ausgangswert der Kennlinie.
bErr	BOOL	Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

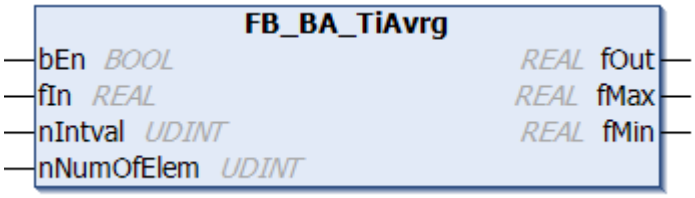
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ErrorDescription	T_MaxString	Get	Gibt eine detaillierte Beschreibung von Fehlern heraus, siehe Fehlerbehandlung [▶ 471].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5.8 FB_BA_TiAvrg



Der Funktionsbaustein FB_BA_TiAvrg berechnet den Mittelwert aus einer Folge von zurückliegenden Eingangswerten.

Die Funktion erfasst Eingangswerte *fIn* in definierten Zeitintervallen *nIntval* und berechnet den Mittelwert der erfassten Werte. Nach jedem Zeitintervall steht ein neuer Mittelwert *fOut* zur Verfügung.

Die zu mittelnden Werte werden in einen FIFO-Speicher geschrieben. Dieser Speicher ist auf maximal 512 Einträge *nNumOfElem* beschränkt. Nach Ablauf des Zeitintervalls *nIntval* wird ein neuer Wert in den Speicher geschrieben und anschließend der Mittelwert gebildet. Sind die Einträge des FIFO-Speichers = *nNumOfElem*, so wird der älteste Eintrag gelöscht, damit ein neuer Wert zur Mittelwertberechnung aufgenommen wird.

Interval	nNumOfE-lem	fIn	FIFOSpeicher	fOut	fMax	fMin
1	5	2	2 / 1	2	2	2
2	5	4	(4+2) / 2	3	4	2
3	5	5	(5+4+2) / 3	3.667	5	2
4	5	6	(6+5+4+2) / 4	4.25	6	2
5	5	7	(7+6+5+4+2) / 5	4.8	7	2
6	5	9	(9+7+6+5+4) / 5	6.2	8	2
7	5	14	(14+9+7+6+5) / 5	8.2	14	2
8	5	1	(1+14+9+7+6) / 5	7.4	14	1
9	5	-4	(-4+1+14+9+7) / 5	5.4	14	-4
10	5	-12	((-12)+(-4)+1+14+9) / 5	1.6	14	-12

Eingänge

```
VAR_INPUT
  bEn           : BOOL;
  fIn           : REAL;
  nIntval       : UDINT;
  nNumOfElem    : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteins. FALSE bedeutet, dass die Ausgangsvariablen den Wert 0 haben und der Inhalt des FIFO-Speichers gelöscht wird.
fIn	REAL	Feld mit den zu berechnenden Werten.
nIntVal	UDINT	Zeitintervall [s] für das Schreiben neuer Werte in den FIFO. Intern begrenzt auf einen Wert von 1 bis 2147483.
nNumOfElem	UDINT	Größe des FIFO-Wertespeichers. Eine Änderung setzt die vorangegangene Mittelwertbildung zurück. Intern begrenzt auf einen Wert von 1 bis 512.

Ausgänge

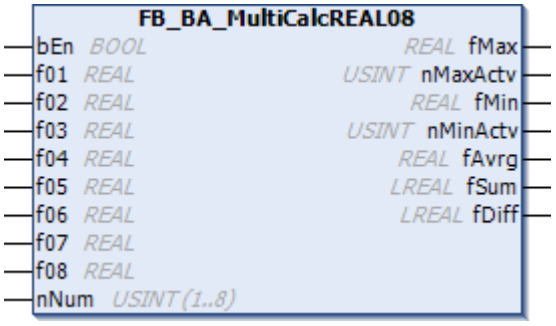
```
Ausgänge
  fOut       : REAL;
  fMax       : REAL;
  fMin       : REAL;
END_VAR
```

Name	Typ	Beschreibung
fOut	REAL	Berechneter Mittelwert
fMax	REAL	Größter Wert im FIFO-Speicher
fMin	REAL	Kleinster Wert im FIFO-Speicher

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.5.9 FB_BA_MultiCalcREAL08



Der Funktionsbaustein FB_BA_MultiCalcREAL08 berechnet im aktivierten Zustand (*bEn* = TRUE) aus den 8 Eingangswerten *f01…f08* folgende Werte:

1. den Maximalwert von allen Eingängen *fMax*
2. den Eingang, an dem dieser Maximalwert anliegt *nMaxActv*
3. den Minimalwert von allen Eingängen *fMin*
4. den Eingang, an dem dieser Minimalwert anliegt *nMinActv*
5. den Mittelwert aller Eingänge *fAvrg*
6. die Summe aller Eingänge *fSum*
7. die Differenz zwischen dem Maximal- und Minimalwert *fDiff*

Sollen nicht alle Eingänge zur Berechnung kommen, so lässt sich die Anzahl durch einen Eintrag an *nNum* einschränken: mit *nNum* = 6 werden beispielsweise die Berechnungen nur für die Eingänge f01...f06 durchgeführt.

Ein Eintrag größer als 8 wird automatisch auf 8 limitiert, ein Eintrag kleiner als 1 automatisch auf 1.

Beispiel:

Eingänge	Ausgang
bEn = TRUE	fMax = 32
f01 = 32	nMaxActv = 1
f02 = 17	fMin = 5
f03 = 5	nMinActv = 3
f04 = 9	fAvrg = 18.5
f05 = 16	fSum = 111
f06 = 32	fDiff = 27
f07 = 25	
f08 = 44	
nNum = 6	

Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  f01…f08      : REAL;
  nNum         : USINT(1..8);
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung der Bausteinfunktion.
f01...f08	REAL	Eingangswerte, aus denen berechnet werden soll.
nNum	USINT	Anzahl der Eingangswerte, mit denen gerechnet wird.

Ausgänge

```
VAR_OUTPUT
  fMax        : REAL;
  nMaxActv    : USINT;
  fMin        : REAL;
  nMinActv    : USINT;
  fAvrg       : REAL;
  fSum        : REAL;
  fDiff       : REAL;
END_VAR
```

Name	Typ	Beschreibung
fMax	REAL	Maximalwert von allen Eingängen.
nMaxActv	USINT	Eingang, an dem der Maximalwert anliegt.
fMin	REAL	Minimalwert von allen Eingängen.
nMinActv	UDSNT	Eingang, an dem der Minimalwert anliegt.
fAvrg	REAL	Mittelwert aller Eingänge
fSum	REAL	Summe aller Eingänge
fDiff	REAL	Differenz zwischen dem Maximal- und Minimalwert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.1.5.6 Monitoring Functions

6.1.2.2.3.1.5.6.1 FB_BA_FdbCtrlBinary



Der Funktionsbaustein FB_BA_FdbCtrlBinary dient der Rückmeldeüberwachung eines Aktors mittels digitaler Rückmeldung. Anwendungsbeispiele des Funktionsbausteins sind zum Beispiel eine Betriebsrückmeldeüberwachung, eine Prozessrückmeldeüberwachung oder die Laufüberwachung eines Antriebs mittels Endlagenschalter.

Der Funktionsbaustein arbeitet in 2 Schritten, um z.B. einen Endlagenschalter zu überwachen. Schritt 1 beinhaltet die Überwachung während der Fahrzeit des Aktors. Schritt 2 überwacht den geöffneten Zustand des Aktors.

Schritt 1: Steht am Eingang **bActuator** ein TRUE an, dann läuft die Verfahrzeit des Aktors *nFdbDelay/ nRemTiFdbDelay* ab. Steht innerhalb dieser Zeit am Eingang *bSwitch* kein TRUE an, so wird dies über den Ausgang bQ angezeigt. Wird *bSwitch* aber innerhalb der Zeitangabe *nFdbDelay* TRUE, so ist Schritt 1 abgearbeitet und Schritt 2 wird aktiv.

Schritt 2: *bActuator* und *bSwitch* sind TRUE. Wird *bActuator* FALSE, so wird Schritt 1 wieder aktiv. Wird *bSwitch* FALSE und innerhalb der Zeitangabe *nInterruptionDelay/nRemTiInterruptionDelay* nicht wieder TRUE, so wird dies über den Ausgang *bQ* angezeigt.

Zusätzlich wird die Ruhestellung des Aktors überwacht. Das heißt, wenn *bActuator* nicht aktiv ist und das Feedback Signal *bSwitch* ein TRUE anzeigt, dann wird nach Ablauf der Zeit *nFdbDelay* der Ausgang *bQ* auf TRUE gesetzt.

bQ signalisiert, dass die Überwachung des Rückmeldesignals eine Störung hat.

Eingänge

```
VAR_INPUT
  bEn                    : BOOL;
  bActuator              : BOOL;
  bSwitch                : BOOL;
  nFdbDelay              : UDINT;
  nInterruptionDelay     : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins. Wenn *bEn* FALSE ist, so ist der Meldeausgang *bQ* ebenfalls FALSE.
bActuator	BOOL	An diesem Eingang wird der Schaltaktorausgang des zu überwachenden Aggregates angeschlossen.
bSwitch	BOOL	Dient dem Anschluss des Rückmelde-Signals, z. B. eines Differenzdruckschalters, Strömungswächters oder Endlagenschalters.
nFdbDelay	UDINT	Ansprechverzögerung [s] der Überwachungsfunktion beim Start des Aktors. Die Eingabe der Zeitangabe ist auf den globalen Parameter *BAComn_Global.udiMaxSecInMilli* aus der Tc3_BA2_Common Bibliothek limitiert (siehe BAComn_Global).
nInterruptionDelay	UDINT	Verzögerungszeit z. B. eines Endlagenschalters im geöffneten Zustand in [s]. Die Eingabe der Zeitangabe ist auf den globalen Parameter *BAComn_Global.udiMaxSecInMilli* aus der Tc3_BA2_Common Bibliothek limitiert, siehe (BAComn_Global).

Ausgänge

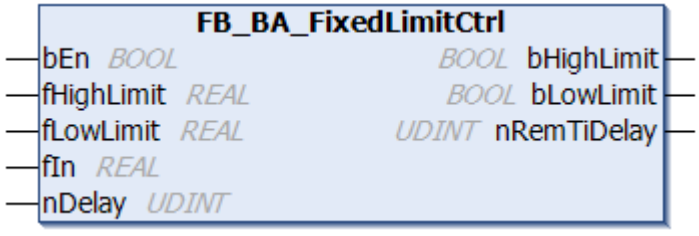
```
VAR_OUTPUT
  bQ                     : BOOL;
  nRemTiFdbDelay         : UDINT;
  nRemTiInterruptionDelay  : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bQ	BOOL	Mit bQ wird signalisiert, dass die Überwachung des Feedback Signals eine Störung anzeigt.
nRemTiFdbDelay	UDINT	Verbleibende Zeit [s] bis der Ausgang *bQ* gesetzt wird. Die Vorgabe kommt von *nFdbDelay*.
nRemTiInterruptionDelay	UDINT	Verbleibende Zeit [s] bis der Ausgang *bQ* gesetzt wird. Die Vorgabe kommt von *nInterruptionDelay*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.6.2 FB_BA_FixedLimitCtrl



Der Funktionsbaustein FB_BA_FixedLimitCtrl dient zur Überwachung eines festen Wertes.

Um den Wert zu überwachenden Wert *fIn* herum wird ein Toleranzbereich definiert.
Der Toleranzbereich ergibt sich aus einem oberen Grenzwert *fHighLimit* und einem unteren Grenzwert *fLowLimit.*

Übersteigt der Wert *fIn* den oberen Grenzwert des Toleranzbereiches, dann wird der Ausgang *bHighLimit* gesetzt. Eine Ansprechverzögerung des Ausgangs *bHighLimit* kann mit der Zeitvariable *nDelay* parametriert werden.

Unterschreitet der Wert *fIn* den unteren Grenzwert des Toleranzbereiches, dann wird der Ausgang *bLowLimit* gesetzt. Eine Ansprechverzögerung des Ausgangs *bLowLimit* kann mit der Zeitvariable *nDelay* parametriert werden.

Ist *fLowLimit* > *fHighLimit*, so wird intern *fLowLimit* auf *fHighLimit* korrigiert.

Eingänge

```
VAR_INPUT
  bEn          : BOOL;
  fHighLimit   : REAL := 32;
  fLowLimit    : REAL := 16;
  fIn          : REAL;
  nDelay       : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins. Wenn *bEn* FALSE ist, so sind die Meldeausgänge *bHighLimit* und *bLowLimit* ebenfalls FALSE.
fHighLimit	REAL	Vorgabe oberer Grenzwert.
fLowLimit	REAL	Vorgabe unterer Grenzwert.
fIn	REAL	Zu überwachender Eingangswert.
nDelay	UDINT	Ansprechverzögerung [s] der Ausgänge *bHighLimit/ bLowLimit*. Die Eingabe der Zeitangabe ist auf den globalen Parameter *BAComn_Global.udiMaxSecInMilli* aus der Tc3_BA2_Common Bibliothek limitiert (siehe BAComn_Global).

Ausgänge

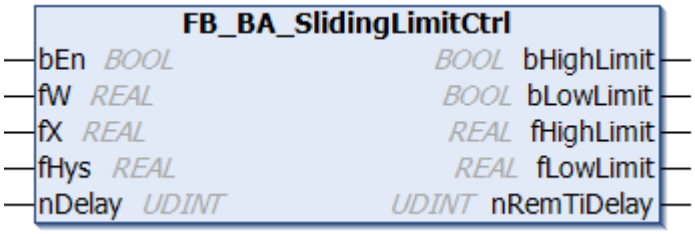
```
VAR_OUTPUT
  bHighLimit         : BOOL;
  bLowLimit          : BOOL;
  nRemTiDelay        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bHighLimit	BOOL	Oberer Grenzwert erreicht.
bLowLimit	BOOL	Unterer Grenzwert erreicht.
nRemTiDelay	UDINT	Verbleibende Zeit nach Überschreitung eines Grenzwertes bis entweder der Ausgang *bHighLimit* oder *bLowLimit* ansprechen.

Voraussetzungen

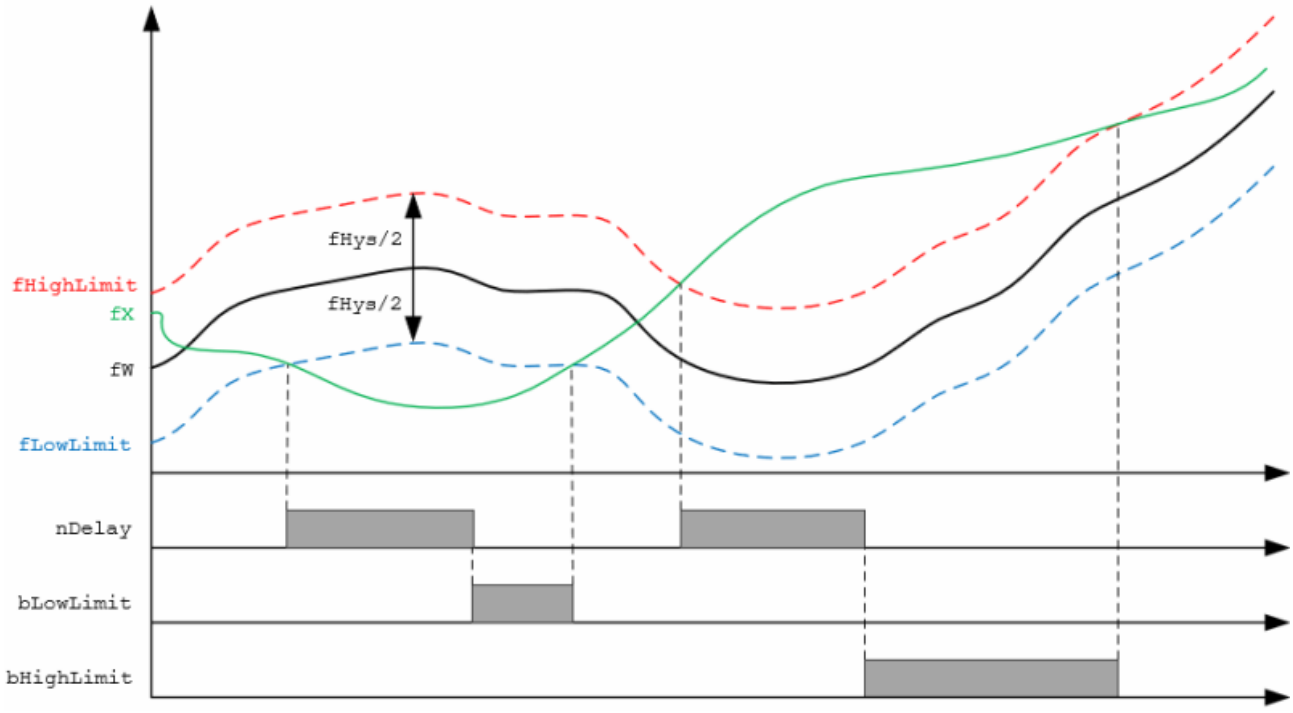
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.6.3 FB_BA_SlidingLimitCtrl



Der Funktionsbaustein FB_BA_SlidingLimitCtrl dient zur Überwachung eines gleitenden Sollwertes.
Der Eingang *bEn* dient der allgemeinen Freigabe des Bausteins.

Um die Funktion einer Regelung zu überprüfen, wird der Istwert mit dem Sollwert der Regelstrecke verglichen.
Ist die Abweichung von Soll- und Istwert innerhalb des Toleranzbereiches *fHys*, dann ist die Regelung in Ordnung. Weicht der Istwert über einen längeren Zeitraum um einen Betrag außerhalb dieses Toleranzbereiches von dem Sollwert ab, dann wird der Timer *nDelay* gestartet. Nach dem Ablauf des Timers, bei bleibender Regelabweichung, wird entweder der Ausgang *bLowLimit* oder *bHighLimit* TRUE. des Funktionsbausteins eine Meldung ausgegeben.



Eingänge

```
VAR_INPUT
  bEn           : BOOL;
  fW            : REAL;
  fX            : REAL;
  fHys          : REAL;
  nDelay        : UDINT;
ENDVAR
```

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins. Wenn *bEn* FALSE ist, so sind die Meldeausgänge *bHighLimit* und *bLowLimit* ebenfalls FALSE.
fW	REAL	Sollwert
fX	REAL	Istwert
fHys	REAL	Hysterese
nDelay	UDINT	Ansprechverzögerung [s] der Ausgänge *bHighLimit*/ *bLowLimit*. Die Eingabe der Zeitangabe ist auf den globalen Parameter *BAComn_Global.udiMaxSecInMilli* aus der Tc3_BA2_Common Bibliothek limitiert (siehe BAComn_Global).

Ausgänge

```
VAR_OUTPUT
  bHighLimit          : BOOL;
  bLowLimit           : BOOL;
  fHighLimit          : REAL;
  fLowLimit           : REAL;
  nRemTiDelay         : UDINT;
END_VAR
```

Name	Typ	Beschreibung
bHighLimit	BOOL	Oberer Grenzwert erreicht.
bLowLimit	BOOL	Unterer Grenzwert erreicht.
fHighLimit	REAL	Ausgabe des oberen Grenzwerts. fHighLimit = fW + (fHys / 2)
fLowLimit	REAL	Ausgabe des unteren Grenzwerts. *fLowLimit* = *fW* - (*fHys* / 2)
nRemTiDelay	UDINT	Verbleibende Zeit nach Überschreitung eines Grenzwertes bis einer der Ausgänge *bHighLimit* oder *bLowLimit* gesetzt wird. Die Vorgabe kommt von *nDelay*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.7 StepControl

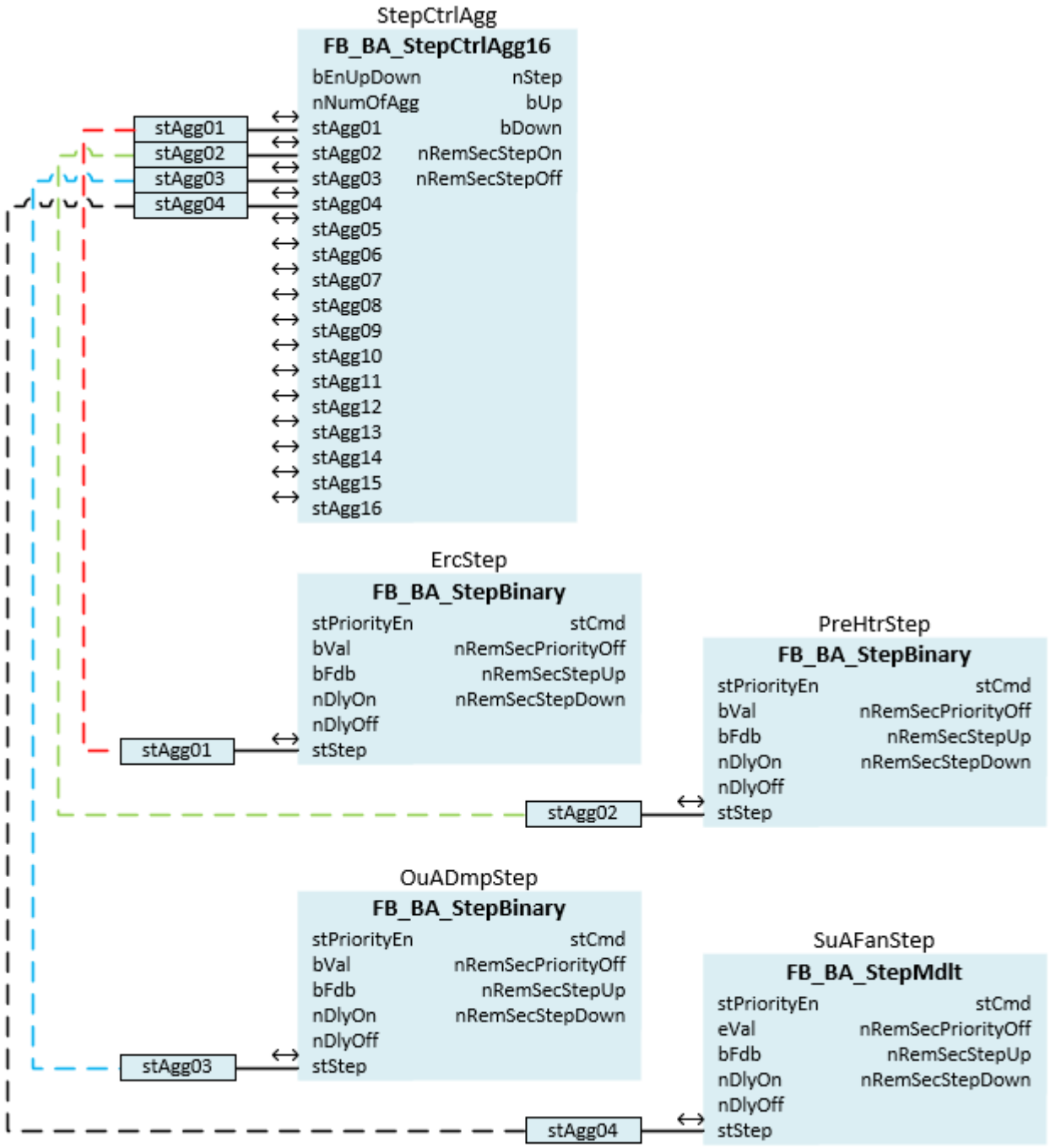
6.1.2.2.3.1.5.7.1 FB_BA_StepCtrlAgg16



Der Funktionsbaustein FB_BA_StepCtrlAgg16 stellt die übergeordnete Steuereinheit einer Schrittkettensteuerung von Aggregaten dar.

Mit der Steuereinheit *FB_BA_StepCtrlAgg16* können einzelne Aggregate einer Anlage in einer bestimmten Reihenfolge sequenziell ein- (stAgg01 > stAgg02 > stAgg03 > stAgg04 ... stAgg16) bzw. ausgeschaltet (stAgg16 > stAgg15 > stAgg14 > stAgg13 ... stAgg01) werden.

Der Datenaustausch zwischen den Empfangsbausteinen der Schrittkettensteuerung (FB_BA_StepBinary [▶ 484], FB_BA_StepMdlt [▶ 486]) und der Steuereinheit *FB_BA_StepCtrlAgg16* geschieht über die Daten- und Kommandostrukturen *stAggxx*.



Illustration

```
FUNCTION_BLOCK FB_BA_StepCtrlAgg16
VAR_INPUT
  bEnUpDown          : BOOL;
  nNumOfAgg          : UDINT;
END_VAR
VAR_OUTPUT
  nStep              : UDINT;
  bUp                : BOOL;
  bDown              : BOOL;
  nRemSecStepOn      : UDINT;
  nRemSecStepOff     : UDINT;
END_VAR
VAR_IN_OUT
  stAgg01-16         : ST_BA_Step;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnUpDown	BOOL	Eine steigende Flanke bedeutet eine steigende Einschaltreihenfolge der Aggregate von 1 nach 16. FALSE bedeutet eine fallende Ausschaltreihenfolge vom höchsten, aktiven Aggregat Richtung 0. 0 bedeutet, dass kein Aggregat der Schrittkettensteuerung aktiv ist.
nNumOfAgg	UDINT	Eingabe der Anzahl der Aggregate der Schrittkettensteuerung. Eine Limitierung lässt nur Werte von 1 - 16 zu.

Ausgänge

Name	Typ	Beschreibung
nStep	UDINT	Anzeige in welchem Schritt sich die Schrittkettensteuerung befindet. 0 bedeutet, dass kein Aggregat der Schrittkettensteuerung aktiv ist.
bUp	BOOL	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung im steigenden Zustand befindet.
bDown	BOOL	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung im fallenden Zustand befindet.
nRemSecStepUp	UDINT	Countdown Schaltung in die nächsthöhere Stufe [s].
nRemSecStepDown	UDINT	Countdown Schaltung in die nächstniedrigere Stufe [s].

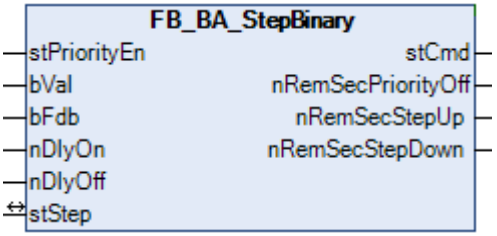
Ein - Ausgänge

Name	Typ	Beschreibung
stAgg01-16	ST_BA_Step [▶ 282]	Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem Steuerbaustein *FB_BA_SequenceLinkBase*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.7.2 FB_BA_StepBinary



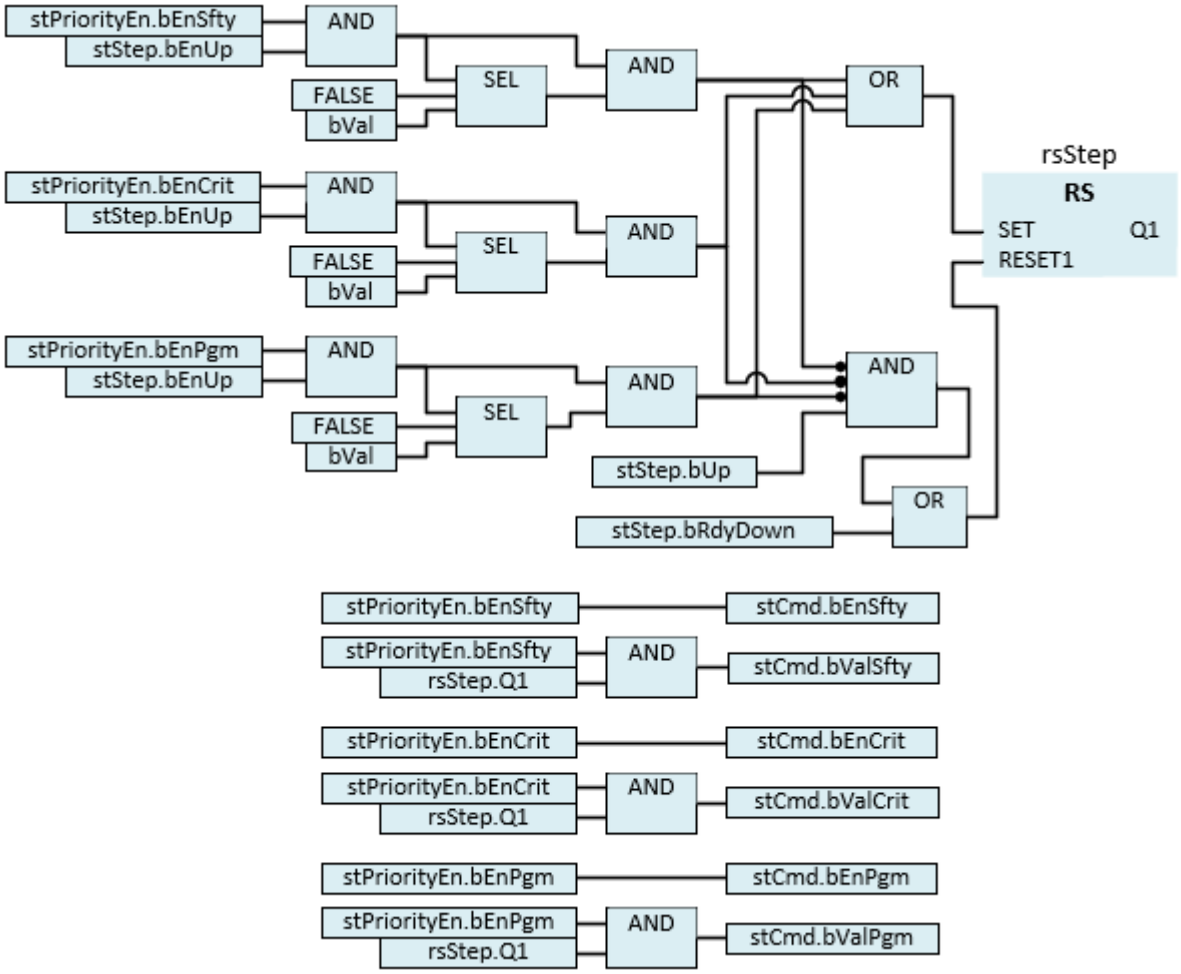
Der Funktionsbaustein ist Teil einer Schrittkettensteuerung.

Er kommuniziert über die Daten- und Kommandostruktur *stStep* mit der übergeordneten Steuereinheit der Schrittkettensteuerung FB_BA_StepCtrlAgg16 [▶ 482].

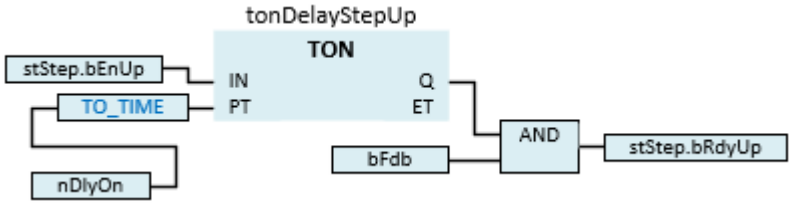
Die Kommandostruktur *stPriorityEn* beinhaltet die Freigabe der Anlage und der Prioritäten.

Die Kommandostruktur *stCmd* steuert das angeschlossene Aggregat an.

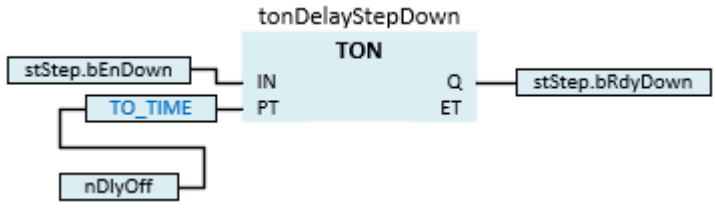
Einschaltbedingungen des angeschlossenen Aggregates



Weiterschaltbedingung innerhalb der Schrittkette



Ausschaltbedingungen des aktiven Aggregates innerhalb der Schrittkette



Illustration

```
FUNCTION_BLOCK FB_BA_StepBinary
VAR_INPUT
  stPriorityEn       : ST_BA_PriorityEn;
  bVal               : BOOL;
  bFdb               : BOOL;
  nDlyOn             : UDINT;
  nDlyOff            : UDINT;
END_VAR
```

```
VAR_OUTPUT
  stCmd               : ST_BA_Binary;
  nRemSecStepUp       : UDINT;
  nRemSecStepDown     : UDINT;
END_VAR
VAR_IN_OUT
  stStep              : ST_BA_Step;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stPriorityEn	ST_BA_PriorityEn [▶ 283]	Die Kommandostruktur beinhaltet die Freigabe der Anlage und der Prioritäten „Safety“, „Critical“ und „Program“.
bVal	BOOL	Einschaltwert für das angeschlossene Aggregat.
bFdb	BOOL	Rückmeldung des angeschlossenen Aggregats. Die Rückmeldung wird zur Weiterschaltung in den nächsten Schritt benötigt. *bFdb* wird nur berücksichtigt, wenn das Aggregat das Aktive in der Schrittkettensteuerung ist.
nDlyOn	UDINT	Zeitangabe Einschaltverzögerung [s]. Die Zeitangabe wird zur Weiterschaltung in den nächsten Schritt benötigt.
nDlyOff	UDINT	Zeitangabe Ausschaltverzögerung [s]. Die Zeitangabe wird zur Ausschaltung des aktiven Schrittes benötigt.

Ausgänge

Name	Typ	Beschreibung
stCmd	ST_BA_Binary [▶ 282]	Die Kommandostruktur *stCmd* übermittelt dem angeschlossenen Aggregat die Freigaben und Schaltwerte der Prioritäten.
nRemSecStepUp	UDINT	Countdown Schaltung in die nächsthöhere Stufe [s].
nRemSecStepDown	UDINT	Countdown Schaltung in die nächstniedrigere Stufe [s].

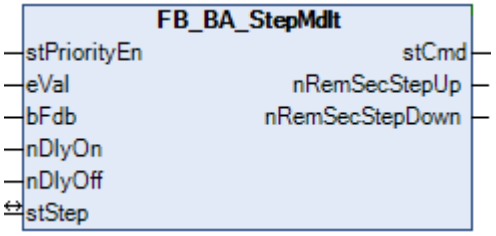
/ Ein - Ausgänge

Name	Typ	Beschreibung
stStep	ST_BA_Step [▶ 282]	Daten- und Befehlsstruktur zwischen dem Schrittkettenbaustein *FB_BA_StepBinary* und dem Steuerbaustein der Schrittkette FB_BA_StepCtrlAgg16 [▶ 482].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.7.3 FB_BA_StepMdlt



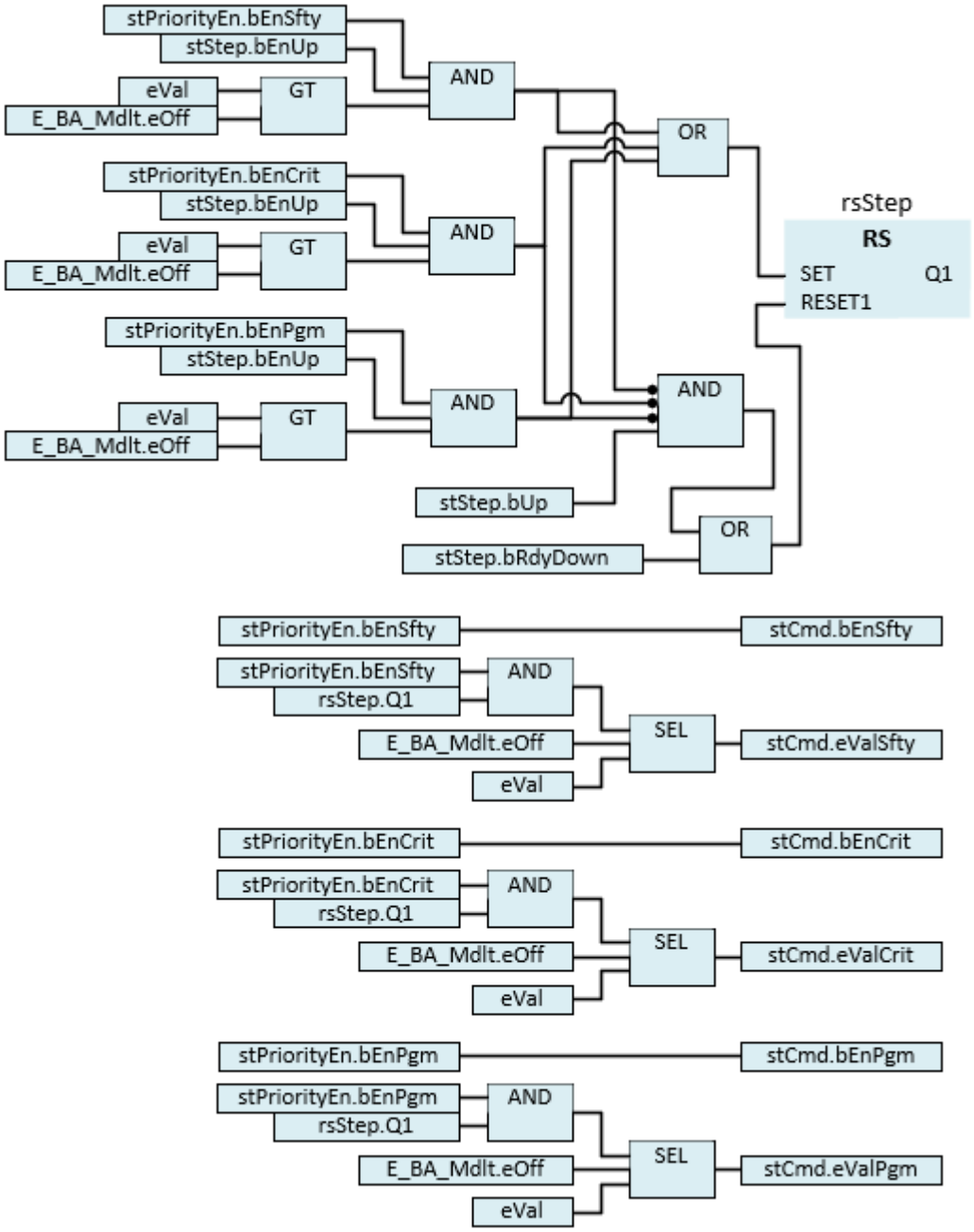
Der Funktionsbaustein ist Teil einer Schrittkettensteuerung.

Er kommuniziert über die Daten- und Kommandostruktur *stStep* mit der übergeordneten Steuereinheit der Schrittkettensteuerung FB_BA_StepCtrlAgg16 [▶ 482].

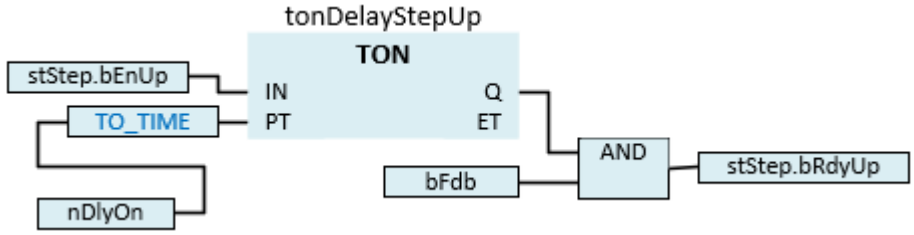
Die Kommandostruktur *stPriorityEn* beinhaltet die Freigabe der Anlage und der Prioritäten.

Die Kommandostruktur *stCmd* steuert das angeschlossene Aggregat an.

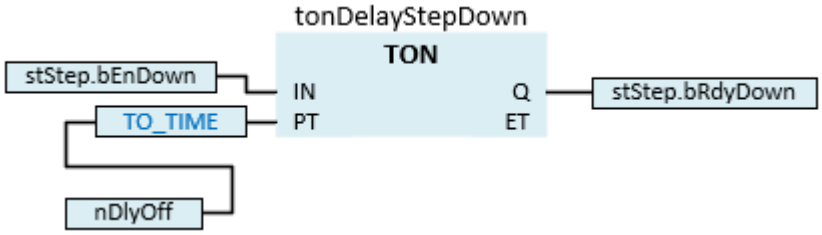
Einschaltbedingung des angeschlossenen Aggregates



Weiterschaltbedingung innerhalb der Schrittkette



Ausschaltbedingung des aktiven Aggregates innerhalb der Schrittkette



Eingänge

Name	Typ	Beschreibung
stPriorityEn	ST_BA_PriorityEn [▶ 283]	Die Kommandostruktur beinhaltet die Freigabe der Anlage und der Prioritäten „Safety“, „Critical“ und „Program“.
eVal	E_BA_Mdlt [▶ 275]	Einschaltwert für das angeschlossene Aggregat.
bFdb	BOOL	Rückmeldung des angeschlossenen Aggregats. Die Rückmeldung wird zur Weiterschaltung in den nächsten Schritt benötigt. *bFdb* wird nur berücksichtigt, wenn das Aggregat das Aktive in der Schrittkettensteuerung ist.
nDlyOn	UDINT	Zeitangabe Einschaltverzögerung [s]. Die Zeitangabe wird zur Weiterschaltung in den nächsten Schritt benötigt.
nDlyOff	UDINT	Zeitangabe Ausschaltverzögerung [s]. Die Zeitangabe wird zur Ausschaltung des aktiven Schrittes benötigt.

Ausgänge

Name	Typ	Beschreibung
stCmd	ST_BA_Mdlt [▶ 281]	Die Kommandostruktur *stCmd* übermittelt dem angeschlossenen Aggregat die Freigaben und Schaltwerte der Prioritäten.
nRemSecStepUp	UDINT	Countdown Schaltung in die nächsthöhere Stufe [s].
nRemSecStepDown	UDINT	Countdown Schaltung in die nächstniedrigere Stufe [s].

Ein - Ausgänge

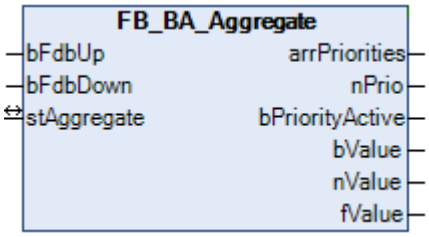
Name	Typ	Beschreibung
stStep	ST_BA_Step [▶ 282]	Daten- und Befehlsstruktur zwischen dem Schrittkettenbaustein *FB_BA_StepBinary* und dem Steuerbaustein der Schrittkette FB_BA_StepCtrlAgg16 [▶ 482].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.1.5.8 PlantControl

6.1.2.2.3.1.5.8.1 FB_BA_Aggregate



Der Aggregatebaustein repräsentiert einen Empfangsbaustein einer Schrittkettensteuerung.

Er kommuniziert über die Daten- und Kommandostruktur *stAggregate* mit der übergeordnete Anlagensteuereinheit der Schrittkettensteuerung FB_BA_PlantControl [▶ 491].

Der Funktionsbaustein dient der Auswertung der Struktur *stAggregate*. Das Ergebnis wird an den Ein- und Ausgängen des Funktionsbausteins ausgegeben.

Syntax

```
FUNCTION_BLOCK FB_BA_Aggregate
VAR_INPUT
  bFdbUp              : BOOL;
  bFdbDown            : BOOL;
END_VAR
VAR_OUTPUT
  arrPriorities       : ARRAY [0..BA_Param.nMaxPriority] OF BOOL;
  nPrio               : UDINT;
  bPriorityActive     : BOOL;
  bValue              : BOOL;
  nValue              : UDINT;
  fValue              : REAL;
END_VAR
VAR_IN_OUT
  stAggregate         : ST_BA_Aggregate;
END_VAR
```

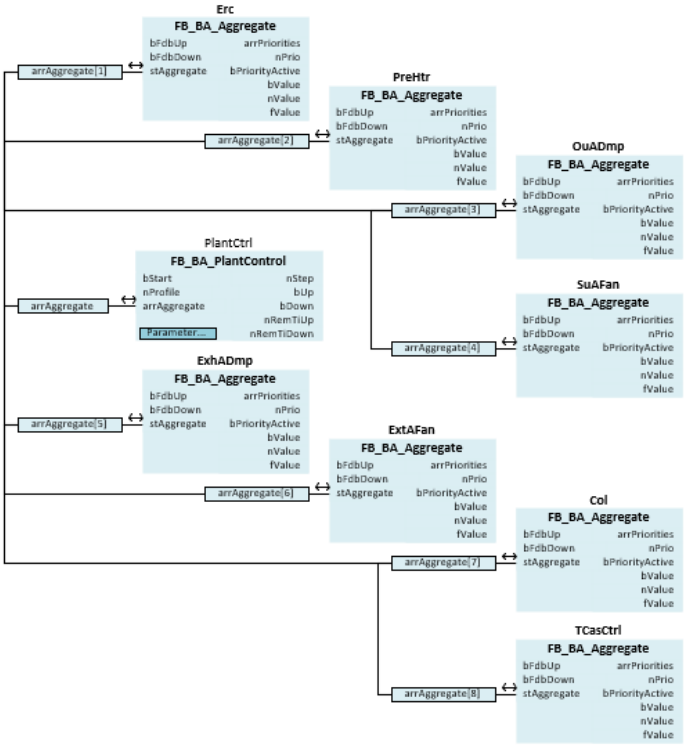
Eingänge

Name	Typ	Beschreibung
bFdbUp	BOOL	An dem Eingang wird eine Rückmeldung des Aggregats angelegt. Sie dient innerhalb der Schrittkettensteuerung dem Hochschalten in den nächsthöheren Schritt. Die Übertragung der Rückmeldung an die Anlagensteuereinheit geschieht über die Daten- und Kommandostruktur *stAggregate*.
bFdbDown	BOOL	An dem Eingang wird eine Rückmeldung des Aggregates angelegt. Sie dient dem Runterschalten in den nächstunteren Schritt. Die Übertragung der Rückmeldung an die Anlagensteuereinheit geschieht über die Daten- und Kommandostruktur *stAggregate*.

Ausgänge

Name	Typ	Beschreibung
arrPriorities	BOOL	Das binäre Array gibt die Freigaben der Prioritäten aus. Es ist immer nur eine Priorität aktiv. *nPrio* zeigt an, welche es ist.
nPrio	UDINT	Anzeige welche Priorität aktiv ist.
bPriorityActive	BOOL	Eine Priorität ist freigegeben.
bValue	BOOL	Binärer Schaltwert für das Aggregat.
nValue	UDINT	Mehrstufiger Schaltwert für das Aggregat.
fValue	REAL	Analoger Stellwert für das Aggregat.

Ein - Ausgänge

Name	Typ	Beschreibung
stAggregate	ST_BA_Aggregate [▶ 283]	Die Aggregatestruktur dient der bidirektionalen Kommunikation zwischen dem angeschlossen Aggregat FB_BA_Aggregate und der Anlagensteuerung FB_BA_PlantControl [▶ 491]. 

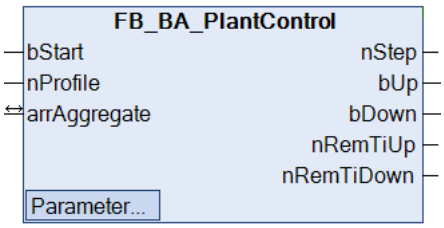
Eigenschaften

Name	Typ	Zugriff	Beschreibung
Down	BOOL	Get	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung in fallendem Zustand befindet.
OpMode	UDINT	Get	Zeigt die aktuelle Betriebsart der Anlagensteuerung an.
Prio	UDINT	Get	Zeigt an, welche Priorität die aktive in der Anlagensteuerung ist.
Step	UDINT	Get	Zeigt den aktuellen Schritt der Anlagensteuerung an.
Up	BOOL	Get	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung im steigendem Zustand befindet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.55	Tc3_BA2 ab v5.3.19.0

6.1.2.2.3.1.5.8.2 FB_BA_PlantControl



Der Funktionsbaustein realisiert das sequenzielle Aufstarten einer Anlage.

Beim Starten werden die Aggregate in einer bestimmten Reihenfolge hintereinander eingeschaltet.

Beim Abschalten der Anlage werden die Aggregate in umgekehrter Reihenfolge ausgeschaltet.

Aus den Schritten der Startsequenz und den Profilen mit den spezifischen Freigaben einzelner Aggregate ergibt sich eine Schaltmatrix.

Schaltmatrix

arrStepDescription		arrProfileDescription	[1]	[2]	[3]	[4]	[5]	[6]
		sDescription	Auto	NightCooling	CoolDownProtection	OverHeatingProtection	ForcedVentilation	CriticalOff
		nPrio	Priority.nProgra	Priority.nProgram	Priority.nProgram	Priority.nProgram	Priority.nCritical	Priority.nCritical
		arrProfileParameter						
		[Profile,Step]						
[1]	Erc	bValue	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[2]	PreHeater PreRinse	bValue	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[3]	DamperOuA	bValue	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[4]	FanSupplyAir	bValue	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		nValue	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eMax	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eOf
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[5]	DamperExhA	bValue	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[6]	FanExtractAir	bValue	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		nValue	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eMax	E_BA_Mdlt.eOn	E_BA_Mdlt.eMax	E_BA_Mdlt.eOf
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[7]	Cooler	bValue	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
		nValue	0	0	0	0	0	0
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[8]	Enabling TemperatureControl	bValue	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		nValue	E_BA_Mdlt.eOn	E_BA_Mdlt.eOff	E_BA_Mdlt.eMax	E_BA_Mdlt.eMin	E_BA_Mdlt.eOn	E_BA_Mdlt.eOf
		fValue	0	0	0	0	0	0
		fDelayUp	0	0	0	0	0	0
		fDelayDown	0	0	0	0	0	0
		bEnFdbUp	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		bEnFdbDown	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

In der vertikal rot eingerahmten Spalte befinden sich die Anlagenschritte.

Die Startsequenz der Anlage beginnt mit dem Element [1] ERC, Energierückgewinnung. Es folgt das Element [2] PreHeater PreRinse, Vorerhitzer Vorspülen usw.

Die Startreihenfolge der Aggregate ist für alle Anlagenprofile gleich.

In dem horizontal grün eingerahmten Bereich befinden sich die Beschreibungen der Anlagenprofile. Mit den Profilen können Aggregate hinsichtlich der projektspezifischen Anforderungen kommandiert werden. Ein Anlagenprofil hat jeweils eine eigene Schrittkette innerhalb der die Kommandierung der Aggregate erfolgt. Die Eingangsvariable *nProfile* bestimmt das Anlagenprofil bzw. die Spalte in der Steuermatrix.

In dem gelb markierten Bereich befindet sich die Kommandierung des Aggregates ERC (Energierückgewinnung) für das Profil Auto (Normalbetrieb). Da es sich um die binäre Freigabe des Aggregates handelt, ist *bValue* = TRUE. Mehrstufige Aggregate können mit der Variable *nValue* und stetige Aggregate mit der Variablen *fValue* kommandiert werden.

In dem blau eingerahmten Feld befinden sich Variablen zur Parametrierung der Startsequenz. Mit diesen Parametern können für jedes Profil unterschiedliche Timer und Bedingungen für das Hoch- und Runterschalten der Startsequenz definiert werden.

In diesem Beispiel ist *bEnFdbUp* = TRUE. Somit wird eine Rückmeldung des Aggregates PreHeater erwartet, bevor die Startsequenz hochschaltet und das nächste Aggregat mit dem Index [3] DmpOuA (Außenluftklappe) einschaltet.

Das Aufstarten der Sequenz wird in Abhängigkeit der Eingangsvariablen *bStart* aktiviert. Eine steigende Flanke an *bStart* beginnt mit dem Kommandieren des Aggregates mit dem Index [1] und fährt so lange mit dem Startprozess fort, bis die höchste Stufe der Schaltsequenzen erreicht ist.

Im Bild ist das Starten der Sequenz in dem Profil Auto mit dem grünen Pfeil illustriert und beinhaltet 8 Schaltsequenzen.

Eine fallende Flanke an *bStart* löst das Herunterfahren der Startsequenz in umgekehrter Reihenfolge aus. Im Bild ist diese durch den roten Pfeil dargestellt.

Wird *bStart* während des Startvorgangs FALSE (fallende Flanke), läuft die Sequenz von dem jeweils gerade aktiven Schritt bis zur vollständigen Abschaltung aller Aggregate wieder herunter.

Im Bild ist diese durch den roten Pfeil dargestellt.

HINWEIS

Verriegelung der Aggregate

Wenn während des Rauf- oder Runterfahrens die Umschaltung von einem Profil auf ein anderes erfolgt, werden die Kommandierungen der neuen Profile unmittelbar an die Aggregate ausgegeben bis zum Anlagenschritt *nStep*. Die Aggregate oberhalb der aktuellen Sequenz (> *nStep*) bekommen die folgenden Kommandierungen: *bValue* = FALSE, *nValue* = 0 und *fValue* = 0.0

Notwendige Verriegelungen bezüglich der Ein- und Ausschaltreihenfolge sind dadurch nicht immer sicher gewährleistet.

Es ist deshalb durch externe Verrieglungen der Aggregate dafür zu sorgen, dass keine kritischen Schaltzustände entstehen. Alternativ dazu kann die jeweils aktive Betriebsart in ihrer Sequenz zunächst komplett heruntergefahren werden, bevor durch *nProfile* auf ein neues Profil umgeschaltet wird.

Syntax

```
FUNCTION_BLOCK FB_BA_PlantCtrl
VAR_INPUT
  bStart                    : BOOL;
  nProfile                  : UDINT;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  arrStepDescription        : ARRAY [1..MAX(1,BA_Param.nMaxAggregate)] OF T_MaxString;
  arrProfileDescription     : ARRAY [1..MAX(1,BA_Param.nMaxProfile)] OF ST_BA_ProfielDescription;
  arrProfileParameter       : ARRAY [1..MAX(1,BA_Param.nMaxProfile),1..MAX(1,BA_Param.nMaxAggregate)]
 OF ST_BA_ProfileParameter;
END_VAR
VAR_OUTPUT
  nStep                  : UDINT;
  bUp                    : BOOL;
  bDown                  : BOOL;
  nRemTiUp               : UDINT;
```

```
  nRemTiDown            : UDINT;
END_VAR
VAR_IN_OUT
  arrAggregate          : ARRAY [1..MAX(1,BA_Param.nMaxAggregate)] OF ST_BA_Aggregate;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bStart	BOOL	Eine steigende Flanke startet die Sequenz. Bei einer fallenden Flanke wird die Sequenz in umgekehrter Reihenfolge heruntergefahren.
nProfile	UDINT	Über den Eingang wird der Steuermatrix das aktuelle, gültige Profil übergeben.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
arrStepDescription	ARRAY [1..MAX(1,BA_Param.nMaxAggregate [▶ 289])] OF T_MaxString	In dem Array wird der Klartext der Anlagenschritte angegeben. Diese Angabe dient rein der Anzeige. In der Schaltmatrix deutet der rot eingerahmte Bereich darauf hin. Initialisierung Klartext Anlagenschritte, siehe Beispiel 1.
arrProfileDescription	ARRAY [1..MAX(1,BA_Param.nMaxProfile [▶ 289])] OF ST_BA_ProfileDescription [▶ 284]	In dem Array werden die Profile der Steuermatrix definiert. In der Schaltmatrix deutet der grün eingerahmte Bereich darauf hin. Initialisierung der Profil Beschreibung, siehe Beispiel 2.
arrProfileParameter	ARRAY [1..MAX(1,BA_Param.nMaxProfile [▶ 289]),1..MAX(1,BA_Param.nMaxAggregate [▶ 289])] OF ST_BA_ProfileParameter [▶ 285]	Das zweidimensionale Array beinhaltet alle Parameter für die Kommandierung der Aggregate und die Parametrierung der Schrittketten. Jedes Element des Arrays beinhaltet die Profil-Parameter-Struktur. In der Schaltmatrix deutet der orange und blau eingerahmte Bereich auf einen Teil des Arrays hin. Initialisierung der Parameter für das Profil Auto, siehe Beispiel 3.

Beispiele

Beispiel 1: Initialisierung Klartext Anlagenschritte

```
PlantControl.arrStepDescription[1]  := txtEvent_EnergyRecovery;
PlantControl.arrStepDescription[2]  := txtEvent_PreRinse;
PlantControl.arrStepDescription[3]  := txtEvent_DamperOutsideAir;
PlantControl.arrStepDescription[4]  := txtEvent_SupplyAirFan;
PlantControl.arrStepDescription[5]  := txtEvent_DamperExhaustAir;
PlantControl.arrStepDescription[6]  := txtEvent_ExtractAirFan;
PlantControl.arrStepDescription[7]  := txtEvent_Cooler;
PlantControl.arrStepDescription[8]  := txtEvent_EnableTemperatureControl;
```

Beispiel 2: Initialisierung der Profil Beschreibung

```
PlantControl.arrProfileDescription[1].sDescription  := txtEvent_Auto;
PlantControl.arrProfileDescription[1].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[2].sDescription  := txtEvent_NightCooling;
PlantControl.arrProfileDescription[2].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[3].sDescription  := txtEvent_CoolDownProtection;
PlantControl.arrProfileDescription[3].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[4].sDescription  := txtEvent_OverHeatingProtection;
PlantControl.arrProfileDescription[4].nPrio         := Priority.nProgram;

PlantControl.arrProfileDescription[5].sDescription  := txtEvent_ForcedVentilation;
PlantControl.arrProfileDescription[5].nPrio         := Priority.nCritical;

PlantControl.arrProfileDescription[6].sDescription  := txtEvent_CriticalOff;
PlantControl.arrProfileDescription[6].nPrio         := Priority.nCritical;

PlantControl.arrProfileDescription[7].sDescription  := txtEvent_SmokeExtraction_P;
PlantControl.arrProfileDescription[7].nPrio         := Priority.nLifeSafety;

PlantControl.arrProfileDescription[8].sDescription  := txtEvent_LifeSafetyOff;
PlantControl.arrProfileDescription[8].nPrio         := Priority.nLifeSafety;
```

Beispiel 3: Initialisierung der Parameter für das Profil Auto

Der erste Index des Arrays steht für das Profil *nProfile*, der zweite für den Schritt in der Startsequenz respektive für das Aggregat.

```
// Profile Auto
// PlantControl.arrProfileParameter[Profile,Step]
// Step 1 = Erc,
// Step 2 = PreHtr
// Step 3 = DamperOuA
// Step 4 = FanSupplyAir
// Step 5 = DamperExhA
// Step 6 = FanExtractAir
// Step 7 = Cooler
// Step 8 = EnablingTemperatureControl

PlantControl.arrProfileParameter[1,1].bValue            := TRUE;
PlantControl.arrProfileParameter[1,1].nValue            := 0;
PlantControl.arrProfileParameter[1,1].fValue            := 0;
PlantControl.arrProfileParameter[1,1].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,1].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,1].bEnFdbUp          := FALSE;
PlantControl.arrProfileParameter[1,1].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,2].bValue            := TRUE;
PlantControl.arrProfileParameter[1,2].nValue            := 0;
PlantControl.arrProfileParameter[1,2].fValue            := 0;
PlantControl.arrProfileParameter[1,2].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,2].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,2].bEnFdbUp          := TRUE;
PlantControl.arrProfileParameter[1,2].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,3].bValue            := TRUE;
PlantControl.arrProfileParameter[1,3].nValue            := 0;
PlantControl.arrProfileParameter[1,3].fValue            := 0;
PlantControl.arrProfileParameter[1,3].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,3].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,3].bEnFdbUp          := TRUE;
PlantControl.arrProfileParameter[1,3].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,4].bValue            := FALSE;
PlantControl.arrProfileParameter[1,4].nValue            := E_BA_Mdlt.eOn;
PlantControl.arrProfileParameter[1,4].fValue            := 0;
PlantControl.arrProfileParameter[1,4].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,4].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,4].bEnFdbUp          := FALSE;
PlantControl.arrProfileParameter[1,4].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,5].bValue            := TRUE;
PlantControl.arrProfileParameter[1,5].nValue            := 0;
PlantControl.arrProfileParameter[1,5].fValue            := 0;
PlantControl.arrProfileParameter[1,5].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,5].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,5].bEnFdbUp          := TRUE;
PlantControl.arrProfileParameter[1,5].bEnFdbDown        := FALSE;

PlantControl.arrProfileParameter[1,6].bValue            := FALSE;
PlantControl.arrProfileParameter[1,6].nValue            := E_BA_Mdlt.eOn;
PlantControl.arrProfileParameter[1,6].fValue            := 0;
PlantControl.arrProfileParameter[1,6].fDelayUp          := 0;
PlantControl.arrProfileParameter[1,6].fDelayDown        := 0;
PlantControl.arrProfileParameter[1,6].bEnFdbUp          := FALSE;
PlantControl.arrProfileParameter[1,6].bEnFdbDown        := FALSE;
```

Ausgänge

Name	Typ	Beschreibung
nStep	UDINT	Mit der Variable wird angezeigt in welchem Schritt sich die Sequenz befindet. *nStep* = 0 bedeutet, dass kein Aggregat der Schrittkettensteuerung aktiv ist.
bUp	BOOL	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung im steigenden Zustand befindet.
bDown	BOOL	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung im fallenden Zustand befindet.
nRemTiUp	UDINT	Countdown Einschaltung in die nächste Stufe [s].
nRemTiDown	UDINT	Countdown Ausschaltung der aktiven Stufe [s].

/ Ein - Ausgänge

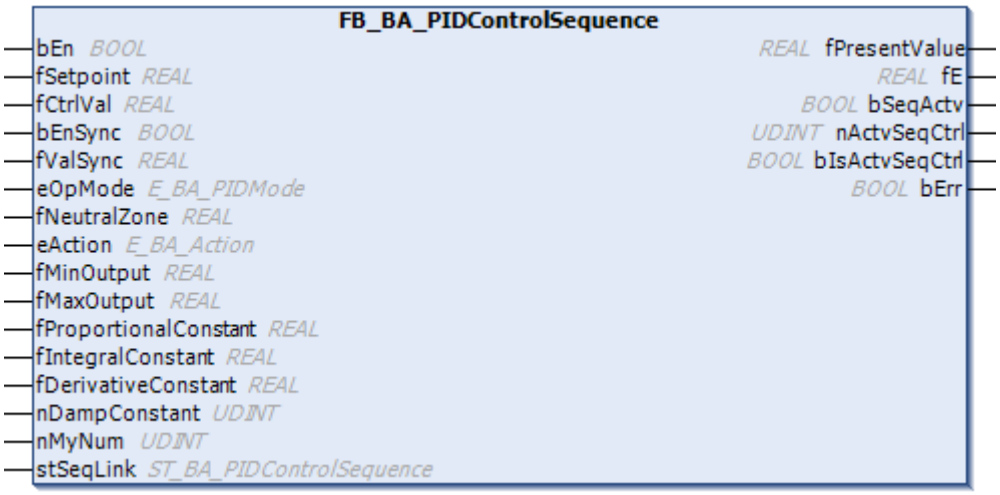
Name	Typ	Beschreibung
arrAggregate	ST_BA_Aggregate [▶ 283]	Das Array beinhaltet für jedes Aggregat eine bidirektionale Daten- und Kommandostruktur. Die Reihenfolge der Aggregate in der Schaltmatrix wird durch die Anschlussreihenfolge der Strukturen bestimmt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.55	Tc3_BA2 ab v5.3.19.0

6.1.2.2.3.1.5.9 Sequence

6.1.2.2.3.1.5.9.1 FB_BA_PIDControlSequence



PID-Sequenzregler als Bestandteil einer Sequenz.

Mit Erhalt der Freigabe *bEn* = TRUE wird dem übergeordneten Kontrollbaustein FB_BA_SequenceLinkBase [▶ 175] mitgeteilt, dass der Sequenzregler betriebsbereit für die Sequenzsteuerung ist.

Der Datenaustausch zwischen den Sequenzreglern *FB_BA_SeqCtrl* und dem Kontrollbaustein FB_BA_SequenceLinkBase [▶ 175] geschieht über die Struktur stSeqLink [▶ 125].

Ausgabewert fPresentValue

Der Funktionsbaustein ermittelt abhängig von den Freigaben *bEn* und *stSequenceLink.bEnSeqLink*, dem Wirksinn *eActionPgm* / *eActionRm* und der Sequenznummer *nActvSeqCtrl* / *nMyNum* den daraus resultierenden Wert am Ausgang *fPresentValue*.

bEn	stSequence-Link.bEnSeqLink	eActionPgm/eAc-tionRm	nActvSeqCtrl/nMy-Num	fPresentValue
TRUE	FALSE	E_BA_Action.eReverse		fMaxOutputPgm / fMaxOutputRm
TRUE	FALSE	E_BA_Action.eDirect		fMinOutputPgm / fMinOutputRm
FALSE	FALSE			0
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl > nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eReverse	nActvSeqCtrl < nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl < nMyNum	fMinOutputPgm / fMinOutputRm
TRUE	TRUE	E_BA_Action.eDirect	nActvSeqCtrl > nMyNum	fMaxOutputPgm / fMaxOutputRm
TRUE	TRUE		nActvSeqCtrl = nMyNum	FB_BA_Loop.fPresentValue

Fehlererkennung

Die unten aufgeführten Fehlermeldungen werden durch den *FB_BA_SeqCtrl* erkannt.

Das x in den Textmeldungen steht für eine nummerische Angabe eines Sequenzreglers.

Die Fehlermeldungen werden in der Entwicklungsumgebung von Tc3 in dem Fenster "Fehlerliste" ausgegeben. Dieses kann unter dem Menüpunkt Ansicht aktiviert werden.

Die Fehlertexte werden über die Properties *ErrorParamMaxSeqCtrl* und *ErrorSeqNumMultiple* ausgegeben.

Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1

Diese Meldung ist die einzige, welche die Sequenzsteuerung deaktiviert.

Deutsch	Englisch
Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1	Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1
Sequenznummer nMyNum = x mehrfach vergeben	Sequence number nMyNum = x assigned multiple times

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_Loop [▶ 225]

Syntax

```
FUNCTION_BLOCK FB_BA_SeqCtrl EXTENDS FB_BA_Loop
VAR_INPUT
  nMyNum            : UDINT;
END_VAR
VAR_OUTPUT
  fE                : REAL;
```

```
  bSeqActv          : BOOL;
  nActvSeqCtrl      : UDINT;
  bIsActvSeqCtrl    : BOOL;
  bErr              : BOOL;
END_VAR
VAR_IN_OUT
  stSeqLink         : ST_BA_SeqLink;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nMyNum	UDINT	Meine Nummer in der Sequenz. Diese Zahl darf nur einmal in einer Sequenzsteuerung vorhanden sein. Eine interne Limitierung lässt nur Werte von 1 - XBA_Param.nMaxSeqCtrl [▶ 128] zu.

Ausgänge

Name	Typ	Beschreibung
fE	REAL	Zeigt die Regelabweichung des Sequenzreglers an. Diese ist abhängig vom Wirksinn des Sequenzreglers. E_BA_Action.eDirect -> fE = fX-fW E_BA_Action.eReverse -> fE = fW-fX
bSeqActv	BOOL	Der Sequenzregler ist in der Regelsequenz eingebunden.
nActvSeqCtrl	UDINT	Nummer des aktiven Sequenzreglers.
bIsActvSeqCtrl	BOOL	Zeigt an, dass der Sequenzregler der aktive in der Sequenzregelung ist. Das Property *IsActvSeqCtrl* zeigt diese Meldung ebenfalls an.
bErr	BOOL	Zeigt an, dass ein Fehler erkannt wurde. Detailliertere Informationen zeigen die Properties *ErrorParamMaxSeqCtrl und SeqNumMultiple*. Unter Fehlererkennung [▶ 498] sind weitere Erläuterungen zur Fehleranalyse zu finden.

/ Ein- Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem Steuerbaustein FB_BA_SequenceLinkBase [▶ 175].

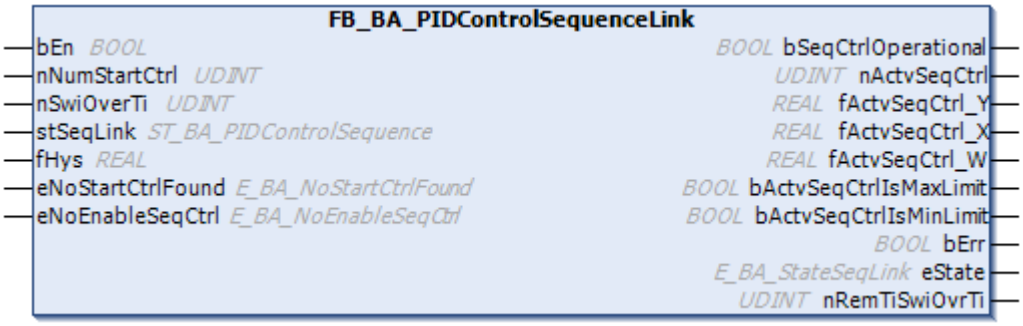
Eigenschaften

Name	Typ	Zugriff	Beschreibung
ActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des aktiven Sequenzcontrollers an.
ErrorParamMaxSeqCtrl	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1", siehe BA_Param.nMaxSeqCtrl [▶ 289]. Der globale Parameter muss auf jeden Fall angepasst werden, weil aufgrund der falschen Parametrierung die Sequenz stoppt.
SeqNumMultiple	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Sequence number nMyNum = x assigned multiple times". Eine Prüfung der angezeigten Sequenznummer ist erforderlich, weil diese mehrfach in der Sequenz vergeben wurde.
IsActvSeqCtrl	BOOL	Get	Der Sequenzregler ist der aktive in der Sequenzsteuerung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_BA2 ab v5.4.2.0

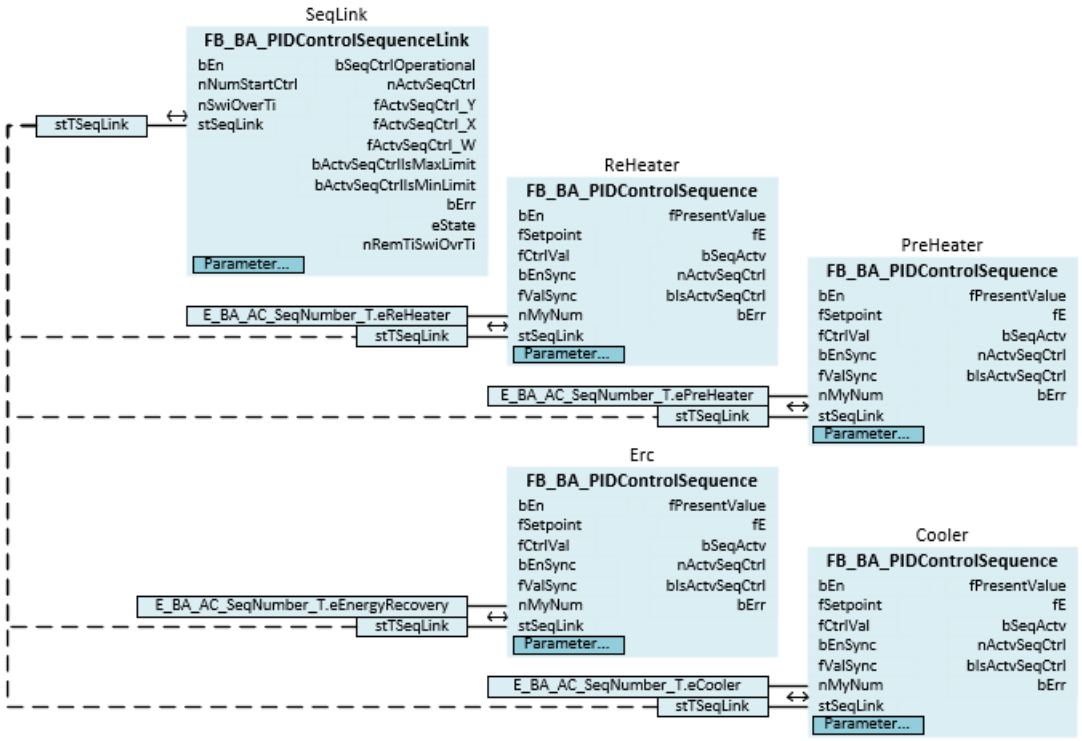
6.1.2.2.3.1.5.9.2 FB_BA_PIDControlSequenceLink



Dieser Baustein stellt die übergeordnete Verbindung einer Sequenzregelung dar.

Datenaustausch Sequenzregelung

Der Datenaustausch zwischen den Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem *FB_BA_SequenceLinkBase* geschieht über die Struktur stSeqLink [▶ 125].



Startverhalten der Sequenz

Eine TRUE-Signal am Eingang *bEn* aktiviert die komplette Sequenzregelung. Der Baustein wird zunächst denjenigen Sequenzregler aktiv schalten, welcher an *nNumStartCtrl* benannt ist. Alle anderen Sequenzregler richten ihren Ausgabewert [▶ 172] nach der Rangordnung des aktiven Reglers.

Ist der an *nNumStartCtrl* angegebene Sequenzregler nicht betriebsbereit, so wird mit dem Enum eNoStartCtrlFound die Vorgehensweise definiert, um einen neuen Startregler zu finden. Somit ist gewährleistet, dass die Sequenz immer startet.

Umschalten in der Sequenz

Erreicht ein Sequenzregler seinen Maximal- oder Minimalwert, so wird je nach Regler-Wirksinn auf den nächsten Regler in der Rangfolge umgeschaltet, sofern der Istwert den Sollwert des nächsten Reglers unterschreitet bzw. überschreitet.

Dabei werden vier Fälle unterschieden:

- Der noch aktive Regler hat einen direkten Wirksinn (Kühlen, E_BA_Action.eDirect) und befindet sich auf seinem Maximalwert: Es wird in der Rangfolge der nächsthöhere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers überschreitet.
- Der noch aktive Regler hat einen direkten Wirksinn (Kühlen, E_BA_Action.eDirect) und befindet sich auf seinem Minimalwert: Es wird in der Rangfolge der nächstniedrigere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers unterschreitet.
- Der noch aktive Regler hat einen indirekten Wirksinn (Heizen, E_BA_Action.eReverse) und befindet sich auf seinem Maximalwert: Es wird in der Rangfolge der nächstniedrigere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers unterschreitet.
- Der noch aktive Regler hat einen indirekten Wirksinn (Heizen, E_BA_Action.eReverse) und befindet sich auf seinem Minimalwert: Es wird in der Rangfolge der nächsthöhere Regler gewählt, wenn der Istwert den Sollwert dieses Reglers überschreitet.

Das Umschalten auf einen anderen Sequenzregler kann durch die Zeitangabe *nSwiOverTi* verzögert werden.

Betriebsbereitschaft der Sequenzregler

Verliert ein Regler in der Sequenz seine Betriebsbereitschaft (fehlende Freigabe *bEn*), so ist er für die gesamte Sequenz nicht mehr verfügbar.

Handelt es sich dabei nicht um den vorher aktiven Regler, so kann es, je nachdem welche Stellgröße dieser Regler ausgegeben hat, zu einer Temperaturänderung kommen, die durch die Regler Sequenz nach Möglichkeit wieder ausgeglichen wird.

Handelt es sich jedoch um den aktiven Sequenzregler, so wird mit der Enumeration eNoEnableSeqCtrl die Vorgehensweise definiert, um einen neuen, aktiven Sequenzregler zu finden.

Erlangt ein Sequenzregler seine Betriebsbereitschaft im laufenden Prozess, so wird der Regler der Sequenz hinzugefügt. Je nach Wirksinn E_BA_Action und eigener Sequenznummer *nMyNum* wird er in die Regler Sequenz eingereiht und seinen Minimal- bzw. Maximalwert ausgeben. Die daraus resultierende Temperaturänderung wird durch die Regler Sequenz nach Möglichkeit wieder ausgeglichen.

Fehlererkennung

Die unten aufgeführten Fehlermeldungen werden durch den *FB_BA_SequenceLinkBase* erkannt.

Das x in den Textmeldungen steht für eine numerische Angabe eines Sequenzreglers.

Die Fehlermeldungen werden in der Entwicklungsumgebung von Tc3 in dem Fenster "Fehlerliste" ausgegeben. Dieses kann unter dem Menüpunkt Ansicht aktiviert werden.

Ausgabe der Fehlertexte über das Property *StateText* ausgegeben.

Die Enumeration eState zeigt zusätzlich die Meldungen an.

Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1

Diese Meldung ist die einzige, welche die Sequenzsteuerung deaktiviert.

Deutsch	Englisch
Der nächste Sequenzsollwert ist kleiner als sein Vorgänger: x	Next sequence setpoint is smaller than its predecessor: x
Globaler Parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1	Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1
Mehrfache Wirkrichtung "eActn" in der Sequenzsteuerung	Multiple direction of action "eActn" of the sequence controller
Startsequenzregler ist nicht freigegeben: nNumStartCtrl = x	Start sequence controller is not enabled: nNumStartCtrl = x
Kein Sequenzregler ist betriebsbereit	No sequence controller is operational
Sequenznummer mehrfach vergeben: x	Sequence number assigned multiple times: x

Syntax

```
FUNCTION_BLOCK FB_BA_SequenceLinkBase
VAR_INPUT
  bEn                                  : BOOL;
  nNumStartCtrl                        : UDINT;
  {attribute 'parameterUnit':= 's'}
  nSwiOverTi                           : UDINT(0..1800);
END_VAR
VAR_OUTPUT
  bSeqCtrlOperational                  : BOOL;
  nActvSeqCtrl                         : UDINT;
  fActvSeqCtrl_Y                       : REAL;
  fActvSeqCtrl_X                       : REAL;
  fActvSeqCtrl_W                       : REAL;
  bActvSeqCtrlIsMaxLimit               : BOOL;
  bActvSeqCtrlIsMinLimit               : BOOL;
  bErr                                 : BOOL;
  eState                               : E_BA_StateSeqLink;
  {attribute 'parameterUnit':= 's'}
  nRemTiSwiOvrTi                       : UDINT;
```

```
END_VAR
VAR_IN_OUT
  stSeqLink                              : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  fHys                                   : REAL := 0.2;
  eNoStartCtrlFound                      : E_BA_NoStartCtrlFound := E_BA_NoStartCtrlFound.eLastActi
onReverse;
  eNoEnableSeqCtrl                       : E_BA_NoEnableSeqCtrl := E_BA_NoEnableSeqCtrl.eNextAction
;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Mit Erhalt der Freigabe des Funktionsbausteins wird die Sequenzsteuerung aktiviert. Zuvor müssen aber die Sequenzregler ihre Freigabe erhalten haben.
nNumStartCtrl	UDINT	Ordnungszahl des Sequenzreglers, welcher bei der Aktivierung der Sequenz der Startregler sein soll. Eine interne Limitierung lässt nur Werte von 1 - BA_Param.nMaxSeqCtrl [▶ 289] zu.
nSwiOverTi	UDINT	Das Umschalten auf einen anderen Sequenzregler kann durch die Zeitangabe verzögert werden. Eine interne Limitierung lässt nur Werte von 0 - 1800 Sekunden zu.

Ausgänge

Name	Typ	Beschreibung
bSeqCtrlOperational	BOOL	Die Sequenzsteuerung ist betriebsbereit bzw. läuft.
nActvSeqCtrl	UDINT	Die Ausgangsvariable zeigt die Nummer des aktiven Sequenzcontrollers an.
fActvSeqCtrl_Y	REAL	Stellgröße des aktiven Sequenzreglers.
fActvSeqCtrl_X	REAL	Istwert des aktiven Sequenzreglers.
fActvSeqCtrl_W	REAL	Sollwert des aktiven Sequenzreglers.
bActvSeqCtrlIsMaxLimit	BOOL	Die obere Ausgabebegrenzung des aktiven Sequenzreglers ist erreicht. Wurde noch nicht auf den nächsten Sequenzregler weitergeschaltet, so wird *bActvSeqCtrlIsMaxLimit* erst auf FALSE gesetzt, wenn *fActvSeqCtrl_Y* unterhalb von (*fActvSeqCtrl_Y* - *fHys*) gefallen ist.
bActvSeqCtrlIsMinLimit	BOOL	Die untere Ausgabebegrenzung des aktiven Sequenzreglers ist erreicht. Wurde noch nicht auf den nächsten Sequenzregler weiter geschaltet, so wird *bActvSeqCtrlIsMinLimit* erst auf FALSE gesetzt, wenn *fActvSeqCtrl_Y* oberhalb von (*fActvSeqCtrl_Y* + *fHys*) gestiegen ist.
bErr	BOOL	Zeigt an, dass ein Fehler erkannt wurde. Detailliertere Informationen zeigen das Property *StateText* an. Zusätzlich zeigt das Enum eState weitere Informationen an. Unter Fehlererkennung [▶ 502] sind weitere Erläuterungen zur Fehleranalyse zu finden.
eState	E_BA_StateSeqLink	Die Enumeration zeigt den Status des *FB_BA_SequenceLinkBase* und der Sequenzregelung an.
nRemTiSwiOvrTi	UDINT	Countdown der Umschaltung auf den nächsten Sequenzregler [s].

/ Ein- Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_SeqCtrl [▶ 172] und dem Steuerbaustein FB_BA_SequenceLinkBase [▶ 500].

Eingänge CONSTANT

Name	Typ	Beschreibung
fHys	REAL	Dieser Hysteresewert legt eine Hysterese um die obere und untere Ausgabebegrenzung *bActvSeqCtrlIsMaxLimit* / *bActvSeqCtrlIsMinLimit* des aktiven Sequenzreglers. Eine interne Limitierung lässt nur Werte von 0 - 2 zu.
eNoStartCtrlFound	E_BA_NoStartCtrlFound	Wenn während des laufenden Prozesses der aktive Sequenzregler seine Betriebsbereitschaft verliert, dann wird mit der Enumeration die Vorgehensweise definiert, um einen neuen, aktiven Sequenzregler zu finden.
eNoEnableSeqCtrl	E_BA_NoEnableSeqCtrl	Ist der Startregler an dem Eingang *nNumStartCtrl* nicht vorhanden oder freigegeben, so wird mit der Enumeration die Vorgehensweise definiert, um einen neuen Startregler zu finden.



Eigenschaften

Name	Typ	Zugriff	Beschreibung
FirstActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des ersten, betriebsbereiten Sequenzreglers an.
FirstActvSeqCtrlWithActionDirect	UDINT	Get	Das Property zeigt die Nummer des ersten, betriebsbereiten Sequenzreglers mit dem Wirksinn *E_BA_Action.eDirect* an.
LastActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des letzten, betriebsbereiten Sequenzreglers an.
LastActvSeqCtrlWithActionReverse	UDINT	Get	Das Property zeigt die Nummer des letzten, betriebsbereiten Sequenzreglers mit dem Wirksinn *E_BA_Action.eReverse* an.
NextActionActvSeqCtrl	UDINT	Get	Je nach Wirksinn des aktiven Sequenzreglers *nActvSeqCtrl* wird hier der nächste, betriebsbereite Sequenzregler angezeigt. Wirksinn Kühlen/Direct: nächster betriebsbereiter Sequenzregler mit dem Wirksinn Kühlen/Direct (nach 5 würde 6 oder höher kommen). Wenn keiner gefunden wird, dann wird *E_BA_NoEnableSeqCtrl.eLastSeqNum* umgesetzt. Wirksinn Heizen/Reverse: vorheriger betriebsbereiter Sequenzregler mit dem Wirksinn Heizen/Reverse (nach 5 würde 4 oder niedriger kommen). Wenn keiner gefunden wird, dann wird *E_BA_NoEnableSeqCtrl.eFirstSeqNum* umgesetzt.
NextActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des nächsten betriebsbereiten Sequenzreglers nach dem letzten aktiven an (nach 5 würde 6 oder höher kommen).
PreviousActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des vorherigen, betriebsbereiten Sequenzreglers nach dem letzten aktiven an (nach 5 würde 4 oder niedriger kommen).
ActvSeqCtrl	UDINT	Get	Das Property zeigt die Nummer des aktiven Sequenzcontrollers an, siehe *nActvSeqCtrl*.
ActvSeqCtrl_ControlValue	REAL	Get	Das Property zeigt die Stellgröße des aktiven Sequenzreglers an, siehe *fActvSeqCtrl_Y*.
ActvSeqCtrl_IsMaxLimit	BOOL	Get	Das Property zeigt an, dass die obere Ausgabebegrenzung des aktiven Sequenzreglers erreicht ist, siehe *bActvSeqCtrlIsMaxLimit*.
ActvSeqCtrl_IsMinLimit	BOOL	Get	Das Property zeigt an, dass die untere Ausgabebegrenzung des aktiven Sequenzreglers erreicht ist, siehe *bActvSeqCtrlIsMinLimit*.
ErrorParamMaxSeqCtrl	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Global parameter Tc3_BA2.BA_Param.nMaxSeqCtrl < 1", siehe BA_Param.nMaxSeqCtrl [▶ 289]. Der globale Parameter muss auf jeden Fall angepasst werden, weil aufgrund der falschen Parametrierung die Sequenz stoppt.
ErrorStartSeqCtrlNotOperable	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Start sequence controller is not enabled: nNumStartCtrl = x". Eine Prüfung des Startsequenzreglers, welcher mit der Eingangsvariablen *nNumStartCtrl* angegeben wurde, ist erforderlich.

Name	Typ	Zugriff	Beschreibung
MultiDirectionActionSeqCtrl	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: Multiple direction of action "eActn" of the sequence controller: x. Mehrfache Wirkrichtung der Sequenz Regler. Es wird immer vom Sequenzregler mit der kleinsten Nummer auf den nächstgrößeren geschaut. Eine Änderung der Wirkrichtung ist möglich.
NextSeqSpIsSmaller	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Next sequence setpoint is smaller than its predecessor: x". Eine Prüfung der Sollwerte der Sequenzregler nach aufsteigender Reihenfolge ist erforderlich.
NoSeqCtrlIsOperational	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "No sequence controller is operational". Eine Prüfung der Betriebsbereitschaft der Sequenzregler ist erforderlich.
SeqNumMulti	T_MaxString	Get	Das Property zeigt im Fehlerfall den folgenden Text an: "Sequence number assigned multiple times: x". Eine Prüfung der angezeigten Sequenznummer ist erforderlich, weil diese mehrfach in der Sequenz vergeben wurde.

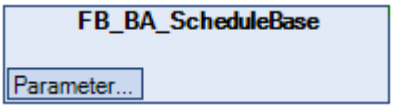
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.56	Tc3_BA2 ab v5.4.2.0

6.1.2.2.3.1.5.10 Schedule

6.1.2.2.3.1.5.10.1 Base

6.1.2.2.3.1.5.10.1.1 FB_BA_ScheduleBase



Der Basis Zeitplan repräsentiert die Parametrierung und Auswertung der abgeleiteten Zeitschalt-Funktionsbausteinen FB_BA_ScheduleAnalog [▶ 524], FB_BA_ScheduleBinary [▶ 524] und FB_BA_ScheduleMultistate [▶ 523].

Der Baustein enthält die eigentlichen Zeitschalteinträge für 24 Ausnahmetage und 7 Wochentage.

Die Zeitschalteinträge für die 24 Ausnahmetage werden in der Liste *aException* parametriert.

Die Zeitschalteinträge für die 7 Wochentage werden in der Liste *aWeek* parametriert.

Die drei abgeleiteten Zeitschalt-Funktionsbausteine FB_BA_ScheduleAnalog [▶ 524], FB_BA_ScheduleBinary [▶ 524] und FB_BA_ScheduleMultistate [▶ 523] haben eine identische Funktionalität - sie unterscheiden sich ausschließlich im Datentyp der Schaltwerte, siehe Bespielinitialisierungen.

Unterschiedliche Datentypen lassen sich innerhalb eines Zeitschalt-Bausteins nicht mischen.

Ermittlung des resultierenden Schaltbefehls

Alle zu einem bestimmten Zeitpunkt aktiven Zeitschalteinträge werden in einer internen Prioritätsmatrix zusammengefasst. Aus diesen Zeitschalteinträgen bestimmt jener mit der höchsten Schaltzeit den Schaltwert an einem der abgeleiteten Funktionsbausteine FB_BA_ScheduleAnalog [▶ 524], FB_BA_ScheduleBinary [▶ 524] oder FB_BA_ScheduleMultistate [▶ 523]. Die Auswertung der Zeitschalteinträge erfolgt zyklisch.

Ein Zeitschaltbefehl gilt jeweils nur an einem Tag (bis Mitternacht). Entweder wird ab Mitternacht ein neuer Schaltbefehl definiert oder der Default-Wert der abgeleiteten Zeitschalt-Funktionsbausteine wird am *PresentValue* ausgegeben.

Der Basis-Zeitplan ermittelt den resultierenden Schaltbefehlseintrag *PresentValue,* gemäß folgender Kriterien:

Ein aktives Ausnahme-Profil mit aktivem Zeitschalteintrag hat Vorrang vor einer Wochenzeitschaltuhr, die Wochenzeitschaltuhr wird dann nicht berücksichtigt.

Hat ein aktives Ausnahme-Profil keinen Zeitschalteintrag, so wird automatisch die Wochenzeitschaltuhr hinzugezogen. Dieser Eintrag ist gleichzusetzen mit einem aktiven Ausnahme-Profil mit aktivem Zeitschalteintrag.

Erst wenn kein Ausnahme-Profil aktiv ist, kommt die Wochenzeitschaltuhr zum Tragen und bestimmt anhand aktiver Zeitschalteinträge den *PresentValue*.

Hat auch die Wochenzeitschaltuhr keinen aktiven Zeitschalteintrag, so ist der Default-Wert = *PresentValue*.

Beispiele zur Konfiguration in Verbindung mit weiteren Schedule Bausteinen [▶ 509].

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
{region 'Variable Parameters'}
  {attribute 'parameterCategory':='Value'}
    aWeek         : T_BA_ScheduleWeek;
  {attribute 'parameterCategory':='Value'}
    aException    : T_BA_ScheduleExceptionList;
  {endregion}
END_VAR
```

Name	Typ	Beschreibung
aWeek	T_BA_ScheduleWeek [▶ 287]	Das Wochenprogramm *aWeek* enthält für jeden Wochentag (Montag bis Sonntag) je ein Tagesprofil. Ein Profil besteht aus den folgenden Eigenschaften: • Liste von Zeitschalteinträgen Bespieldeklarationen sind anhängig [▶ 509] zu finden.
aException	T_BA_ScheduleExceptionList [▶ 288]	Das Liste *aException* enthält 24 Zeitplan-Ausnahme-Profile. Ein Profil besteht aus den folgenden Eigenschaften: • Zeitraumtypen (Einfaches Datum, Datumsbereich, Woche Und Tag) • Datumswert • Liste von Zeitschalteinträgen Bespieldeklarationen sind anhängig [▶ 509] zu finden.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Beschreibung
Exception	T_BA_ScheduleExceptionList [▶ 288]	Get / Set	Public	Die Eigenschaft *Exception* ermöglicht den Schreib- und Lesezugriff auf die Ausnahmeeinträge des Zeitplanes.
Week	T_BA_ScheduleWeek [▶ 287]	Get / Set	Public	Die Eigenschaft *Week* ermöglicht den Schreib- und Lesezugriff auf das Wochenprogramm des Zeitplanes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.2.0

6.1.Beispiele für Initialisierungen
2.2.
3.1.
5.1
0.1.
1.1

Wochenzeitplan/week für FB_BA_ScheduleBinary [▶ 524]

Von Montag bis Sonntag wird eine Funktion um 6:00 Uhr eingeschaltet. Täglich um 18:00 Uhr wird sie ausgeschaltet. Falls diese Funktion nach 18 Uhr extern eingeschaltet wurde, so wird um 0:00 Uhr erneut ausgeschaltet.

```
// Monday
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.bVal           := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour          := 6;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour          := 18;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond        := 0;
// Tuesday
Schedule.aWeek[E_BA_Weekday.eTuesday,1].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].uValue.bVal          := TRUE;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nHour         := 6;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].uValue.bVal          := FALSE;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nHour         := 18;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].uValue.bVal          := FALSE;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nHour         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nSecond       := 0;
// Wednesday
Schedule.aWeek[E_BA_Weekday.eWednesday,1].eState             := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].uValue.bVal        := TRUE;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nHour       := 6;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nMinute     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nSecond     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].eState             := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].uValue.bVal        := FALSE;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nHour       := 18;
```

```
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].uValue.bVal          := FALSE;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nHour         := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nSecond       := 0;
// Thursday
Schedule.aWeek[E_BA_Weekday.eThursday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,1].uValue.bVal           := TRUE;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nHour          := 6;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,2].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nHour          := 18;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,3].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nSecond        := 0;
// Friday
Schedule.aWeek[E_BA_Weekday.eFriday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,1].uValue.bVal             := TRUE;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nHour            := 6;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,2].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nHour            := 18;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,3].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nSecond          := 0;
// Saturday
Schedule.aWeek[E_BA_Weekday.eSaturday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].uValue.bVal           := TRUE;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nHour          := 6;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nHour          := 18;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].uValue.bVal           := FALSE;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nSecond        := 0;
// Sunday
Schedule.aWeek[E_BA_Weekday.eSunday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,1].uValue.bVal             := TRUE;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nHour            := 6;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,2].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nHour            := 18;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,3].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nSecond          := 0;
```

Wochenzeitplan für FB_BA_ScheduleMultistate [▶ 523]

Von Montag bis Sonntag wird um 23:00 Uhr und 0:00 Uhr der Wert 2 vorgegeben, um 5:00 Uhr jedoch der Wert 1.

```
// Monday
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour          := 23;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.udiVal         := 1;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour          := 5;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond        := 0;
// Tuesday
Schedule.aWeek[E_BA_Weekday.eTuesday,1].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].uValue.udiVal        := 2;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nHour         := 23;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].uValue.udiVal        := 2;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nHour         := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nSecond       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].eState               := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].uValue.udiVal        := 1;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nHour         := 5;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nMinute       := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nSecond       := 0;
// Wednesday
Schedule.aWeek[E_BA_Weekday.eWednesday,1].eState             := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].uValue.udiVal      := 2;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nHour       := 23;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nMinute     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nSecond     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].eState             := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].uValue.udiVal      := 2;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nHour       := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nMinute     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nSecond     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].eState             := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].uValue.udiVal      := 1;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nHour       := 5;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nMinute     := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nSecond     := 0;
// Thursday
Schedule.aWeek[E_BA_Weekday.eThursday,1].eState              := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,1].uValue.udiVal       := 2;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nHour        := 23;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nMinute      := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nSecond      := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].eState              := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,2].uValue.udiVal       := 2;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nHour        := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nMinute      := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nSecond      := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].eState              := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,3].uValue.udiVal       := 1;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nHour        := 5;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nMinute      := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nSecond      := 0;
// Friday
Schedule.aWeek[E_BA_Weekday.eFriday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,1].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nHour          := 23;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,2].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,3].uValue.udiVal         := 1;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nHour          := 5;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nSecond        := 0;
```

```
// Saturday
Schedule.aWeek[E_BA_Weekday.eSaturday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nHour          := 23;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].uValue.udiVal         := 2;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].uValue.udiVal         := 1;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nHour          := 5;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eSaturday,3].stTime.nSecond        := 0;
// Sunday
Schedule.aWeek[E_BA_Weekday.eSunday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,1].uValue.udiVal           := 2;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nHour            := 23;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,2].uValue.udiVal           := 2;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eSunday,3].uValue.udiVal           := 1;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nHour            := 5;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eSunday,3].stTime.nSecond          := 0;
```

Kombination der Ausnahme im Datumsbereich mit einem Wochenzeitplan für FB_BA_ScheduleAnalog [▶ 524]

Dieses Beispiel stellt eine Kombination eines Datumsbereiches in Verbindung mit einer Wochenzeitschaltuhr dar.

Im Datumsbereich vom 1. März 2026 bis zum 31.September 2026 wird an jedem Tag um 0:00 Uhr der analoge Wert 14,0 vorgegeben.

Außerhalb dieses Datumsbereiches ist der Wochenzeitschaltplan aktiv. Von Montag bis Freitag wird um 6:00 Uhr der Wert 22,0, um 18:00 Uhr der Wert 16,0 und um 0:00 Uhr der Wert 14,5 ausgegeben.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay01;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 0;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.rVal                   := 14.0;

// Monday
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.rVal             := 22.0;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour            := 6;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.rVal             := 16.0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour            := 18;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.rVal             := 14.5;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond          := 0;
// Tuesday
```

```
Schedule.aWeek[E_BA_Weekday.eTuesday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].uValue.rVal             := 22.0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nHour            := 6;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,1].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].uValue.rVal             := 16.0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nHour            := 18;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].uValue.rVal             := 14.5;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eTuesday,3].stTime.nSecond          := 0;
// Wednesday
Schedule.aWeek[E_BA_Weekday.eWednesday,1].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].uValue.rVal           := 22.0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nHour          := 6;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,1].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].uValue.rVal           := 16.0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nHour          := 18;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,2].stTime.nSecond        := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].eState                := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].uValue.rVal           := 14.5;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nHour          := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nMinute        := 0;
Schedule.aWeek[E_BA_Weekday.eWednesday,3].stTime.nSecond        := 0;
// Thursday
Schedule.aWeek[E_BA_Weekday.eThursday,1].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,1].uValue.rVal            := 22.0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nHour           := 6;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,1].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,2].uValue.rVal            := 16.0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nHour           := 18;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,2].stTime.nSecond         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].eState                 := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eThursday,3].uValue.rVal            := 14.5;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nHour           := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nMinute         := 0;
Schedule.aWeek[E_BA_Weekday.eThursday,3].stTime.nSecond         := 0;
// Friday
Schedule.aWeek[E_BA_Weekday.eFriday,1].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,1].uValue.rVal              := 22.0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nHour             := 6;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,1].stTime.nSecond           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,2].uValue.rVal              := 16.0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nHour             := 18;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,2].stTime.nSecond           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].eState                   := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eFriday,3].uValue.rVal              := 14.5;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nHour             := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nMinute           := 0;
Schedule.aWeek[E_BA_Weekday.eFriday,3].stTime.nSecond           := 0;
```

Ausnahme/Exception Datum/Date

Die folgenden Beispiele zeigen mögliche Varianten für die *Ausnahme Datum* von einem Zeitschaltplan mit einem binären Zeitschalteintrag auf.

Variante 01

Die *Ausnahme Datum* ist immer aktiv. Es wird täglich um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                       := 255;
Schedule.aException[1].uDate.stDate.eMonth                      := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                        := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek                  := E_BA_WeekDay.Unspecified;
```

```
Schedule.aException[1].aEntry[1].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour              := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[1].uValue.bVal               := TRUE;
Schedule.aException[1].aEntry[2].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour              := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[2].uValue.bVal               := FALSE;
```

Variante 02

Für das gesamte Jahr 2025 wird täglich um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                               := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                  := 125;
Schedule.aException[1].uDate.stDate.eMonth                 := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                   := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek             := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour              := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[1].uValue.bVal               := TRUE;
Schedule.aException[1].aEntry[2].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour              := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[2].uValue.bVal               := FALSE;
```

Variante 03

Im Februar wird an jedem Tag um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                               := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                  := 255;
Schedule.aException[1].uDate.stDate.eMonth                 := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDate.nDay                   := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek             := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour              := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[1].uValue.bVal               := TRUE;
Schedule.aException[1].aEntry[2].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour              := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[2].uValue.bVal               := FALSE;
```

Variante 04

An jedem 29. eines Monats um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                               := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                  := 255;
Schedule.aException[1].uDate.stDate.eMonth                 := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                   := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek             := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour              := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[1].uValue.bVal               := TRUE;
Schedule.aException[1].aEntry[2].eState                    := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour              := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute            := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond            := 0;
Schedule.aException[1].aEntry[2].uValue.bVal               := FALSE;
```

Variante 05

Jeden Montag wird um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 255;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 06

Am 29. März wird um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 255;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 07

Im Jahr 2025 wird immer am 29. eines Monats um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 125;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 08

Der 29. eines Monats muss auf einem Montag liegen, damit um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet wird.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 255;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 09

Im April 2025 wird an jedem Tag um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 125;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eApril;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 10

Im April wird an jedem Montag um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 255;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eApril;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 11

Im Jahr 2030 wird an jedem Montag um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 130;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 12

Am 29. Mai 2030 wird um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                   := 130;
Schedule.aException[1].uDate.stDate.eMonth                  := E_BA_Month.eMay;
Schedule.aException[1].uDate.stDate.nDay                    := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek              := E_BA_WeekDay.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
```

```
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variante 13

Im Jahr 2030 wird im Mai an jedem Montag um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 130;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.eMay;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDate.eDayOfWeek                 := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                   := TRUE;
Schedule.aException[1].aEntry[2].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                  := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variante 14

Im Jahr 2030 muss der 29. eines Monats auf einem Montag liegen, damit um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet wird.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 130;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek                 := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                   := TRUE;
Schedule.aException[1].aEntry[2].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                  := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variante 15

Der 29. April muss auf einem Montag liegen, damit um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet wird.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 255;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.eApril;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDate.eDayOfWeek                 := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                  := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                   := TRUE;
Schedule.aException[1].aEntry[2].eState                        := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                  := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute                := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                   := FALSE;
```

Variante 16

Der 28. Februar 2025 müsste auf einem Montag liegen, damit um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet wird. Innerhalb des Bausteins wird überprüft, ob die Einschaltbedingung korrekt ist. Da der 28. Februar 2025 auf einem Freitag liegt, wird der Wochentag *eDayOfWeek* auf Freitag korrigiert, so dass am Freitag, den 28. Februar 2025 um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet wird.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDate;
Schedule.aException[1].uDate.stDate.nYear                      := 125;
Schedule.aException[1].uDate.stDate.eMonth                     := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDate.nDay                       := E_BA_Day.eDay28;
```

```
Schedule.aException[1].uDate.stDate.eDayOfWeek                := E_BA_WeekDay.eMonday;
Schedule.aException[1].aEntry[1].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                 := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                  := TRUE;
Schedule.aException[1].aEntry[2].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                 := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                  := FALSE;
```

Ausnahme/Exception Woche und Tag/WeekNDay

Die folgenden Beispiele zeigen mögliche Varianten für die *Ausnahme Woche* und *Ausnahme Tag* mit einem binären Zeitschalteintrag auf.

Variante 01

Die Ausnahme *WeekNDay* ist immer aktiv. Es wird an jedem Tag um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                  := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth                := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth          := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday              := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                 := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                  := TRUE;
Schedule.aException[1].aEntry[2].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                 := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                  := FALSE;
```

Variante 02

Im August wird an jedem Tag um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                  := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth                := E_BA_Month.eAugust;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth          := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday              := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                 := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                  := TRUE;
Schedule.aException[1].aEntry[2].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                 := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                  := FALSE;
```

Variante 03

An den Tagen 8-14 eines jeden Monats wird um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                  := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth                := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth          := E_BA_Week.eWeek2;
Schedule.aException[1].uDate.stWeekNDay.eWeekday              := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                 := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                  := TRUE;
Schedule.aException[1].aEntry[2].eState                       := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                 := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute               := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond               := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                  := FALSE;
```

Variante 04

An jedem Montag wird um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 05

An den Tagen 29-31 im März wird um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.eWeek5;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.Unspecified;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 06

An jedem Montag im März wird um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 07

Einer der Tage 29-31 eines Monats muss auf einem Montag liegen, damit um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet wird.

```
Schedule.aException[1].eType                                := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth              := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth        := E_BA_Week.eWeek5;
Schedule.aException[1].uDate.stWeekNDay.eWeekday            := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour               := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                := TRUE;
Schedule.aException[1].aEntry[2].eState                     := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour               := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute             := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond             := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                := FALSE;
```

Variante 08

Einer der Tage 1-7 im März muss auf einem Montag liegen, damit um 6:00 Uhr ein- und um 18:00 Uhr ausgeschaltet wird.

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eWeekNDay;
Schedule.aException[1].uDate.stWeekNDay.eMonth                  := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stWeekNDay.eWeekOfMonth            := E_BA_Week.eWeek1;
Schedule.aException[1].uDate.stWeekNDay.eWeekday                := E_BA_Weekday.eMonday;
Schedule.aException[1].aEntry[1].eState                         := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[1].stTime.nHour                   := 6;
Schedule.aException[1].aEntry[1].stTime.nMinute                 := 0;
Schedule.aException[1].aEntry[1].stTime.nSecond                 := 0;
Schedule.aException[1].aEntry[1].uValue.bVal                    := TRUE;
Schedule.aException[1].aEntry[2].eState                         := E_BA_SchedEntryState.eValue;
Schedule.aException[1].aEntry[2].stTime.nHour                   := 18;
Schedule.aException[1].aEntry[2].stTime.nMinute                 := 0;
Schedule.aException[1].aEntry[2].stTime.nSecond                 := 0;
Schedule.aException[1].aEntry[2].uValue.bVal                    := FALSE;
```

Autokorrektur Initialisierungen

Bei fehlerhaften Initialisierungen findet innerhalb des Funktionsbausteins eine Autokorrektur statt. Die folgenden Beispiele stellen mögliche Initialisierungen zur Autokorrektur dar.

DateRange

Beispiel 01

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 255;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek  := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 255;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth        := E_BA_Month.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay          := E_BA_Day.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek    := E_BA_WeekDay.Unspecified;
```

Automatische Korrektur, weil kein spezifischer Zeitraum angegeben ist.

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.Invalid;
```

Beispiel 02

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 127;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek  := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth        := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay          := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek    := E_BA_WeekDay.eMonday;
```

Automatische Korrektur, die Datumsbereiche werden getauscht und die Wochentage korrigiert.

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := E_BA_Day.eDay30;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek  := E_BA_WeekDay.eWednesday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 127;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth        := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay          := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek    := E_BA_WeekDay.eWednesday;
```

Beispiel 03

```
Schedule.aException[1].eType                                    := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear       := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth      := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay        := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek  := E_BA_WeekDay.eTuesday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear         := 126;
```

```
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.Unspecified;
```

Automatische Korrektur. die Wochentage werden bei Falschangabe angepasst.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eMonday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eTuesday;
```

Beispiel 04

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 222;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := 123;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := 222;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 220;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := 121;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := 145;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := 127;
```

Automatische Korrektur, bei Jahr, Monat und Tag werden die höchsten, gültigen Einträge genommen - 31.12.2100. Der Wochentag wird dem resultierenden Datum angepasst.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 200;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eDecember;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eFriday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 200;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eDecember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eFriday;
```

Beispiel 05

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 124;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay30;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.Unspecified;
```

Automatische Korrektur, die Monatstage werden angepasst. Der Wochentag wird dem resultierenden Datum angepasst.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 124;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eThursday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eFebruary;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay28;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eSaturday;
```

Beispiel 06

```
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                  := E_BA_SchedEntryState.Invalid;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.bVal             := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour            := 30;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute          := 120;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond          := 124;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHundredths      := 124;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                  := E_BA_SchedEntryState.eUndefined;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.bVal             := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour            := 11;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute          := 12;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond          := 13;
```

```
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHundredths      := 14;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                  := E_BA_SchedEntryState.eNull;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour            := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute          := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond          := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHundredths      := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,4].uValue.bVal             := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHour            := 99;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nMinute          := 111;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nSecond          := 222;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHundredths      := 100;
```

Automatische Korrektur, Sortierung beim Wochenzeitplan nach dargestellter Reihenfolge in den Einträgen: *E_BA_SchedEntryState.eValue* > *E_BA_SchedEntryState.eNull* > *E_BA_SchedEntryState.eUndefined*. Aus E_BA_SchedEntryState.Invalid wird *E_BA_SchedEntryState.eUndefined*.

Die Uhrzeiten werden angepasst.

```
Schedule.aWeek[E_BA_Weekday.eMonday,1].eState                  := E_BA_SchedEntryState.eValue;
Schedule.aWeek[E_BA_Weekday.eMonday,1].uValue.bVal             := TRUE;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHour            := 23;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nMinute          := 59;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nSecond          := 59;
Schedule.aWeek[E_BA_Weekday.eMonday,1].stTime.nHundredths      := 99;
Schedule.aWeek[E_BA_Weekday.eMonday,2].eState                  := E_BA_SchedEntryState.eNull;
Schedule.aWeek[E_BA_Weekday.eMonday,2].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHour            := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nMinute          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nSecond          := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,2].stTime.nHundredths      := 0;
Schedule.aWeek[E_BA_Weekday.eMonday,3].eState                  := E_BA_SchedEntryState.eUndefined;
Schedule.aWeek[E_BA_Weekday.eMonday,3].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHour            := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nMinute          := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nSecond          := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,3].stTime.nHundredths      := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].eState                  := E_BA_SchedEntryState.eUndefined;
Schedule.aWeek[E_BA_Weekday.eMonday,4].uValue.bVal             := FALSE;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHour            := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nMinute          := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nSecond          := 255;
Schedule.aWeek[E_BA_Weekday.eMonday,4].stTime.nHundredths      := 255;
```

Beispiel 07

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.Invalid;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 127;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eMonday;
```

Automatische Korrektur, der Datumsbereich wird getauscht und die Wochentage korrigiert.

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 126;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay30;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eWednesday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 127;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eWednesday;
```

Beispiel 08

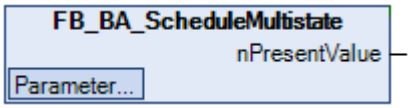
```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.Unspecified;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
```

```
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.Unspecified;
```

Automatische Korrektur, die Wochentage werden angepasst

```
Schedule.aException[1].eType                                   := E_BA_DateValChoice.eDateRange;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nYear      := 125;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eMonth     := E_BA_Month.eMarch;
Schedule.aException[1].uDate.stDateRange.stDateFrom.nDay       := E_BA_Day.eDay31;
Schedule.aException[1].uDate.stDateRange.stDateFrom.eDayOfWeek := E_BA_WeekDay.eMonday;
Schedule.aException[1].uDate.stDateRange.stDateTo.nYear        := 126;
Schedule.aException[1].uDate.stDateRange.stDateTo.eMonth       := E_BA_Month.eSetember;
Schedule.aException[1].uDate.stDateRange.stDateTo.nDay         := E_BA_Day.eDay29;
Schedule.aException[1].uDate.stDateRange.stDateTo.eDayOfWeek   := E_BA_WeekDay.eTuesday;
```

6.1.2.2.3.1.5.10.2 FB_BA_ScheduleMultistate



Der Funktionsbaustein FB_BA_ScheduleMultistate repräsentiert einen numerischen Zeitplan. Er besteht im Wesentlichen aus der Basisklasse FB_BA_ScheduleBase [▶ 507].

Der Funktionsbaustein bestimmt anhand der Basisklasse den numerischen Ausgabewert. Wird innerhalb der Basisklasse nicht der Ausgabewert *nPresentValue* bestimmt, so wird *nDefaultValue* = *nPresentValue*.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    {attribute 'parameterCategory':='Value'}
      nDefaultValue    : UDINT;
  {endregion}
END_VAR
```

Name	Typ	Beschreibung
nDefaultValue	UDINT	Der Vorgabewert *nDefaultValue* ist in folgenden Fällen aktiv: *nPresentValue* = *nDefaultValue*, wenn innerhalb der Basisklasse FB_BA_ScheduleBase [▶ 507] nicht der Ausgabewert *nPresentValue* bestimmt wird.

Ausgänge

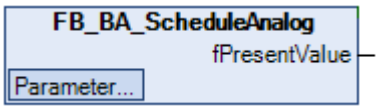
```
VAR_OUTPUT
  nPresentValue    : UDINT;
END_VAR
```

Name	Typ	Beschreibung
nPresentValue	UDINT	Aktueller Wert Er besteht im Wesentlichen aus der Basisklasse FB_BA_ScheduleBase [▶ 507]. Innerhalb der Basisklasse wird der numerische Ausgabewert für *nPresentValue* bestimmt. Wird innerhalb der Basisklasse nicht der Ausgabewert *nPresentValue* bestimmt, so ist *nDefaultValue* = *nPresentValue*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.1.5.10.3 FB_BA_ScheduleAnalog



Der Funktionsbaustein FB_BA_ScheduleAnalog repräsentiert einen analogen Zeitplan. Er besteht im Wesentlichen aus der Basisklasse FB_BA_ScheduleBase [▶ 507].

Der Funktionsbaustein bestimmt anhand der Basisklasse den analogen Ausgabewert. Wird innerhalb der Basisklasse nicht der Ausgabewert *fPresentValue* bestimmt, so wird *fDefaultValue* = *fPresentValue*.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    {attribute 'parameterCategory':='Value'}
      fDefaultValue    : REAL;
  {endregion}
END_VAR
```

Name	Typ	Beschreibung
fDefaultValue	REAL	Der Vorgabewert *fDefaultValue* ist in folgenden Fällen aktiv: *fPresentValue* = *fDefaultValue*, wenn innerhalb der Basisklasse FB_BA_ScheduleBase [▶ 507] nicht der Ausgabewert *fPresentValue* bestimmt wird.

Ausgänge

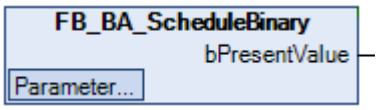
```
VAR_OUTPUT
  fPresentValue    : REAL;
END_VAR
```

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller Wert Er besteht im Wesentlichen aus der Basisklasse FB_BA_ScheduleBase [▶ 507]. Innerhalb der Basisklasse wird der numerische Ausgabewert für *fPresentValue* bestimmt. Wird innerhalb der Basisklasse nicht der Ausgabewert *fPresentValue* bestimmt, so ist *fDefaultValue* = *fPresentValue*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.1.5.10.4 FB_BA_ScheduleBinary



Der Funktionsbaustein FB_BA_ScheduleBinary repräsentiert einen binären Zeitplan. Er besteht im Wesentlichen aus der Basisklasse FB_BA_ScheduleBase [▶ 507].

Der Funktionsbaustein bestimmt anhand der Basisklasse den binären Ausgabewert. Wird innerhalb der Basisklasse nicht der Ausgabewert *bPresentValue* bestimmt, so wird *bDefaultValue* = *bPresentValue*.

Eingänge CONSTANT PERSISTENT

```
VAR_INPUT CONSTANT PERSISTENT
  {region 'Fixed Parameters'}
    {attribute 'parameterCategory':='Value'}
      bDefaultValue    : BOOL;
  {endregion}
END_VAR
```

Name	Typ	Beschreibung
bDefaultValue	BOOL	Der Vorgabewert *bDefaultValue* ist in folgenden Fällen aktiv: *bPresentValue* = *bDefaultValue*, wenn innerhalb der Basisklasse FB_BA_ScheduleBase [▶ 507] nicht der Ausgabewert *bPresentValue* bestimmt wird.

Ausgänge

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

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller Wert Er besteht im Wesentlichen aus der Basisklasse FB_BA_ScheduleBase [▶ 507]. Innerhalb der Basisklasse wird der numerische Ausgabewert für *bPresentValue* bestimmt. Wird innerhalb der Basisklasse nicht der Ausgabewert *bPresentValue* bestimmt, so ist *bDefaultValue* = *bPresentValue*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.2.0

6.1.2.2.3.2 Functions

6.1.2.2.3.2.1 Universal

6.1.2.2.3.2.1.1 Priority

6.1.2.2.3.2.1.1.1 F_BA_BinaryCrit



Die Funktion vom Rückgabetyp ST_BA_Binary [▶ 282] ermöglicht das Schreiben auf die Priorität „Critical“ des Rückgabetypen der Funktion.

In der Funktionstabelle ist zu sehen, welche Zustände am Rückgabetypen der Funktion ausgegeben werden.

bEnCrit	F_BA_Binary-Sfty-Crit.bEnSfty	F_BA_Binary-Sfty-Crit.bValSfty	F_BA_Binary-Sfty-Crit.bEnCrit	F_BA_Binary-Sfty-Crit.bValCrit	F_BA_Binary-Sfty-Crit.bEnPgm	F_BA_Binary-Sfty-Crit.bValPgm
FALSE	stCmd.bEnSfty	stCmd.bValSfty	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
TRUE	stCmd.bEnSfty	stCmd.bValSfty	**bEnCrit**	**bValCrit**	stCmd.bEnPgm	stCmd.bValPgm

Syntax

```
FUNCTION F_BA_BinaryCrit  : ST_BA_Binary
VAR_INPUT
  bEnCrit      : BOOL;
  bValCrit     : BOOL;
  stCmd        : ST_BA_Binary;
END_VAR
```

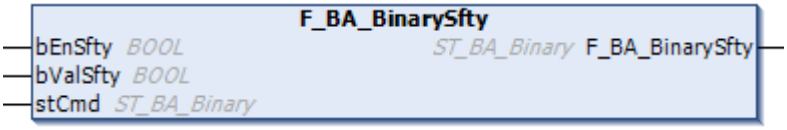
Eingänge

Name	Typ	Beschreibung
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“ des Rückgabetypen ST_BA_Binary [▶ 282].
bValCrit	BOOL	Zu beschreibender Wert der Priorität „Critical“ des Rückgabetypen ST_BA_Binary [▶ 282], wenn *bEnCrit* den Wert TRUE hat.
stCmd	ST_BA_Binary [▶ 282]	Die Kommandostruktur *stCmd* übermittelt dem angeschlossenen Aggregat die Freigaben und Schaltwerte der Prioritäten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.2.1.1.2 F_BA_BinarySfty



Die Funktion vom Rückgabetyp ST_BA_Binary [▶ 282] ermöglicht das Schreiben auf die Priorität „Safety“ des Rückgabetypen der Funktion.

In der Funktionstabelle ist zu sehen, welche Zustände am Rückgabetypen der Funktion ausgegeben werden.

bEnSfty	F_BA_Binary-Sfty-Crit.bEnSfty	F_BA_Binary-Sfty-Crit.bValSfty	F_BA_Binary-Sfty-Crit.bEnCrit	F_BA_Binary-Sfty-Crit.bValCrit	F_BA_Binary-Sfty-Crit.bEnPgm	F_BA_Binary-Sfty-Crit.bValPgm
FALSE	stCmd.bEnSfty	stCmd.bValSfty	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
TRUE	**bEnSfty**	**bValSfty**	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm

Syntax

```
FUNCTION F_BA_BinarySfty : ST_BA_Binary
VAR_INPUT
  bEnSfty      : BOOL;
```

```
  bValSfty     : BOOL;
  stCmd        : ST_BA_Binary;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“ des Rückgabetypen ST_BA_Binary [▶ 282].
bValSfty	BOOL	Zu beschreibender Wert der Priorität „Safety“ des Rückgabetypen ST_BA_Binary [▶ 282], wenn *bEnCrit* den Wert TRUE hat.
stCmd	ST_BA_Binary [▶ 282]	Die Kommandostruktur *stCmd* übermittelt dem angeschlossenen Aggregat die Freigaben und Schaltwerte der Prioritäten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.2.1.1.3 F_BA_BinarySftyCrit



Die Funktion vom Rückgabetyp ST_BA_Binary [▶ 282] ermöglicht das Schreiben auf die Prioritäten „Safety“ und „Critical“ des Rückgabetypen der Funktion.

In der Funktionstabelle ist zu sehen, welche Zustände am Rückgabetypen der Funktion ausgegeben werden.

bEnSfty	bEnCrit	F_BA_BinarySfty-Crit.bEnSfty	F_BA_BinarySfty-Crit.bValSfty	F_BA_BinarySfty-Crit.bEnCrit	F_BA_BinarySfty-Crit.bVal-Crit	F_BA_BinarySfty-Crit.bEn-Pgm	F_BA_BinarySfty-Crit.bVal-Pgm
FALSE	FALSE	stCmd.bEnSfty	stCmd.bValSfty	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
TRUE	FALSE	**bEnSfty**	**bValSfty**	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm
FALSE	**TRUE**	stCmd.bEnSfty	stCmd.bValSfty	**bEnCrit**	**bValCrit**	stCmd.bEnPgm	stCmd.bValPgm
TRUE	**TRUE**	**bEnSfty**	**bValSfty**	stCmd.bEnCrit	stCmd.bValCrit	stCmd.bEnPgm	stCmd.bValPgm

Syntax

```
FUNCTION F_BA_BinarySftyCrit : ST_BA_Binary
VAR_INPUT
  bEnSfty      : BOOL;
  bValSfty     : BOOL;
  bEnCrit      : BOOL;
  bValCrit     : BOOL;
  stCmd        : ST_BA_Binary;
END_VAR
```

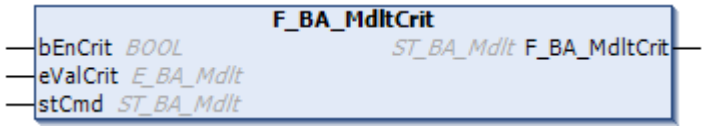
Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“ des Rückgabetypen ST_BA_Binary [▶ 282].
bValSfty	BOOL	Zu beschreibender Wert der Priorität „Safety“ des Rückgabetypen ST_BA_Binary [▶ 282], wenn *bEnCrit* den Wert TRUE hat.
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“ des Rückgabetypen ST_BA_Binary [▶ 282].
bValCrit	BOOL	Zu beschreibender Wert der Priorität „Critical“ des Rückgabetypen ST_BA_Binary [▶ 282], wenn *bEnCrit* den Wert TRUE hat.
stCmd	ST_BA_Binary [▶ 282]	Die Kommandostruktur *stCmd* übermittelt dem angeschlossenen Aggregat die Freigaben und Schaltwerte der Prioritäten.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.2.1.1.4 F_BA_MdltCrit



Die Funktion vom Rückgabetyp ST_BA_Mdlt [▶ 281] ermöglicht das Schreiben auf die Priorität „Critical“ des Rückgabetypen der Funktion.

In der Funktionstabelle ist zu sehen, welche Zustände am Rückgabetypen der Funktion ausgegeben werden.

bEnCrit	F_BA_Mdlt-Crit.bEnSfty	F_BA_Mdlt-Crit.eValSfty	F_BA_Mdlt-Crit.bEnCrit	F_BA_Mdlt-Crit.eValCrit	F_BA_Mdlt-Crit.bEnPgm	F_BA_Mdlt-Crit.eValPgm
FALSE	stCmd.bEnSfty	stCmd.eValSfty	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
TRUE	stCmd.bEnSfty	stCmd.eValSfty	**bEnCrit**	**eValCrit**	stCmd.bEnPgm	stCmd.eValPgm

Syntax

```
FUNCTION F_BA_MdltCrit : ST_BA_Mdlt
VAR_INPUT
  bEnCrit      : BOOL;
  eValCrit     : E_BA_Mdlt;
  stCmd        : ST_BA_Mdlt;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“ des Rückgabetypen ST_BA_Mdlt [▶ 281].
eValCrit	E_BA_Mdlt [▶ 275]	Zu beschreibender Wert der Priorität „Critical“ des Rückgabetypen ST_BA_Mdlt [▶ 281], wenn *bEnCrit* den Wert TRUE hat.
stCmd	ST_BA_Mdlt [▶ 281]	Die Kommandostruktur *stCmd* dient zur Ansteuerung modulierender Aggregate.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.2.1.1.5 F_BA_MdltSfty



Die Funktion vom Rückgabetyp ST_BA_Mdlt [▶ 281] ermöglicht das Schreiben auf die Priorität „Safety“ des Rückgabetypen der Funktion.

In der Funktionstabelle ist zu sehen, welche Zustände am Rückgabetypen der Funktion ausgegeben werden.

bEnSfty	F_BA_MdltSfty.bEnSfty	F_BA_MdltSfty.eValSfty	F_BA_MdltSfty.bEnCrit	F_BA_MdltSfty.eValCrit	F_BA_MdltSfty.bEnPgm	F_BA_MdltSfty.eValPgm
FALSE	stCmd.bEnSfty	stCmd.eValSfty	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
TRUE	**bEnSfty**	**eValSfty**	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm

Syntax

```
FUNCTION F_BA_MdltSfty : ST_BA_Mdlt
VAR_INPUT
  bEnSfty      : BOOL;
  eValSfty     : E_BA_Mdlt;
  stCmd        : ST_BA_Mdlt;
END_VAR
```

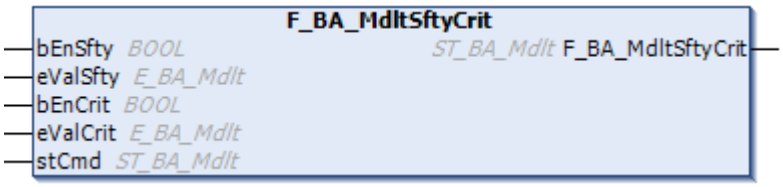
Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“ des Rückgabetypen ST_BA_Mdlt [▶ 281].
eValSfty	E_BA_Mdlt [▶ 275]	Zu beschreibender Wert der Priorität „Safety“ des Rückgabetypen ST_BA_Mdlt [▶ 281], wenn *bEnCrit* den Wert TRUE hat.
stCmd	ST_BA_Mdlt [▶ 281]	Die Kommandostruktur *stCmd* dient zur Ansteuerung modulierender Aggregate.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.2.1.1.6 F_BA_MdltSftyCrit



Die Funktion vom Rückgabetyp ST_BA_Mdlt [▶ 281] ermöglicht das Schreiben auf die Prioritäten „Safety“ und „Critical“ des Rückgabetypen der Funktion.

In der Funktionstabelle ist zu sehen, welche Zustände am Rückgabetypen der Funktion ausgegeben werden.

bEnSfty	bEnCrit	F_BA_Mdlt-Crit.bEnSfty	F_BA_Mdlt-Crit.eValSfty	F_BA_Mdlt-Crit.bEnCrit	F_BA_Mdlt-Crit.eVal-Crit	F_BA_Mdlt-Crit.bEn-Pgm	F_BA_Mdlt-Crit.eVal-Pgm
FALSE	FALSE	stCmd.bEnSfty	stCmd.eValSfty	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
FALSE	**TRUE**	stCmd.bEnSfty	stCmd.eValSfty	**bEnCrit**	**eValCrit**	stCmd.bEnPgm	stCmd.eValPgm
TRUE	FALSE	**bEnSfty**	**eValSfty**	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm
TRUE	**TRUE**	**bEnSfty**	**eValSfty**	stCmd.bEnCrit	stCmd.eValCrit	stCmd.bEnPgm	stCmd.eValPgm

Syntax

```
FUNCTION F_BA_MdltSfty : ST_BA_Mdlt
VAR_INPUT
  bEnSfty      : BOOL;
  eValSfty     : E_BA_Mdlt;
  bEnCrit      : BOOL;
  eValCrit     : E_BA_Mdlt;
  stCmd        : ST_BA_Mdlt;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnSfty	BOOL	Freigabe zum Beschreiben der Priorität „Safety“ des Rückgabetypen ST_BA_Mdlt [▶ 281].
eValSfty	E_BA_Mdlt [▶ 275]	Zu beschreibender Wert der Priorität „Safety“ des Rückgabetypen ST_BA_Mdlt [▶ 281], wenn *bEnCrit* den Wert TRUE hat.
bEnCrit	BOOL	Freigabe zum Beschreiben der Priorität „Critical“ des Rückgabetypen ST_BA_Mdlt [▶ 281].
eValCrit	E_BA_Mdlt [▶ 275]	Zu beschreibender Wert der Priorität „Critical“ des Rückgabetypen ST_BA_Mdlt [▶ 281], wenn *bEnCrit* den Wert TRUE hat.
stCmd	ST_BA_Mdlt [▶ 281]	Die Kommandostruktur *stCmd* dient zur Ansteuerung modulierender Aggregate.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.22	Tc3_BA2 ab v5.2.5.0

6.1.2.2.3.2.2 Conversion

6.1.2.2.3.2.2.1 Temperature conversion function

Temperaturumrechnungsfunktionen zwischen Kelvin, Celsius, Réaumur und Fahrenheit.

Übersicht

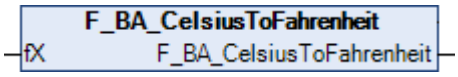
	Kelvin [K]	Grad Celsius [°C]	Grad Réaumur [°R]	Grad Fahrenheit [°F]
Absoluter Nullpunkt	0	-273,15	-218,52	-459,67
Schmelzpunkt	273,15	0	0	32
Siedepunkt	373,15	100	80	212

(Schmelz- und Siedepunkt beziehen sich auf reines Wasser)

Umrechnungsregeln

	Kelvin [K]	Grad Celsius [°C]	Grad Réaumur [°R]	Grad Fahrenheit [°F]
X = Kelvin [K]	-	= X − 273,15°C	= 4/5(X − 273,15) [°R]	-459,67
Grad Celsius [°C]	= x + 273,15K	-	0	32
Grad Réaumur [°R]		100	-	212
Grad Fahrenheit [°F]				-

6.1.2.2.3.2.2.2 F_BA_CelsiusToFahrenheit



Die Funktion F_BA_CelsiusToFahrenheit vom Rückgabetyp LREAL dient der Umrechnung einer Temperaturgröße in Grad Celsius zu einer Temperaturgröße in Grad Fahrenheit.

Syntax

```
FUNCTION F_BA_CelsiusToFahrenheit : LREAL
VAR_INPUT
  fX    : LREAL;
END_VAR
```

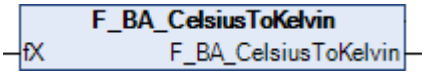
Eingänge

Name	Typ	Beschreibung
fX	LREAL	Eingabegröße in Grad Celsius.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.3 F_BA_CelsiusToKelvin



Die Funktion F_BA_CelsiusToKelvin vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Celsius zu einer Temperaturgröße in Kelvin.

Syntax

```
FUNCTION F_BA_CelsiusToKelvin : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

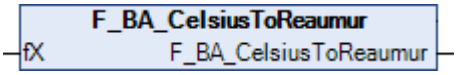
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Grad Celsius.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.4 F_BA_CelsiusToReaumur



Die Funktion F_BA_CelsiusToReaumur vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Celsius zu einer Temperaturgröße in Grad Réaumur.

Syntax

```
FUNCTION F_BA_CelsiusToReaumur : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

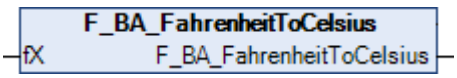
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Grad Celsius.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.5 F_BA_FahrenheitToCelsius



Die Funktion F_BA_FahrenheitToCelsius vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Fahrenheit zu einer Temperaturgröße in Grad Celsius.

Syntax

```
FUNCTION F_BA_FahrenheitToCelsius : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

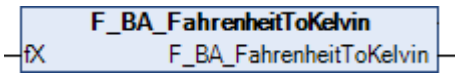
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Grad Fahrenheit.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.6 F_BA_FahrenheitToKelvin



Die Funktion F_BA_FahrenheitToKelvin vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Fahrenheit zu einer Temperaturgröße in Kelvin.

Syntax

```
FUNCTION F_BA_FahrenheitToKelvin : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

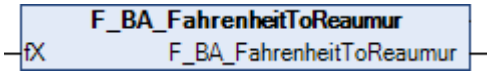
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Grad Fahrenheit.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.7 F_BA_FahrenheitToReaumur



Die Funktion F_BA_FahrenheitToReaumur vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Fahrenheit zu einer Temperaturgröße in Grad Réaumur.

Syntax

```
FUNCTION F_BA_FahrenheitToReaumur : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

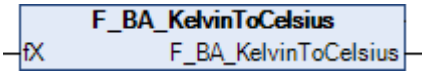
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Grad Fahrenheit.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.8 F_BA_KelvinToCelsius



Die Funktion F_BA_KelvinToCelsius vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Kelvin zu einer Temperaturgröße in Grad Celsius.

Syntax

```
FUNCTION F_BA_KelvinToCelsius : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

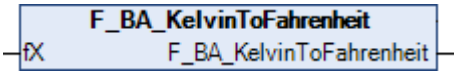
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Kelvin.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.9 F_BA_KelvinToFahrenheit



Die Funktion F_BA_KelvinToFahrenheit vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Kelvin zu einer Temperaturgröße in Grad Fahrenheit.

Syntax

```
FUNCTION F_BA_KelvinToFahrenheit : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

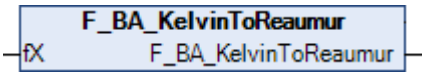
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Kelvin.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.10 F_BA_KelvinToReaumur



Die Funktion F_BA_KelvinToReaumur vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Kelvin zu einer Temperaturgröße in Grad Réaumur.

Syntax

```
FUNCTION F_BA_KelvinToReaumur : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

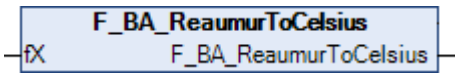
Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Kelvin.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.11 F_BA_ReaumurToCelsius



Die Funktion F_BA_ReaumurToCelsius vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Réaumur zu einer Temperaturgröße in Grad Celsius.

Syntax

```
FUNCTION F_BA_ReaumurToCelsius : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

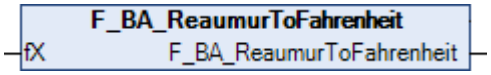
Eingänge

Name	Typ	Beschreibung
Fx	REAL	Eingabegröße in Grad Réaumur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.12 F_BA_ReaumurToFahrenheit



Die Funktion F_BA_ReaumurToFahrenheit vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Réaumur zu einer Temperaturgröße in Grad Fahrenheit.

Syntax

```
FUNCTION F_BA_ReaumurToFahrenheit : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

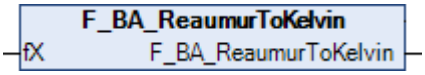
Eingänge

Name	Typ	Beschreibung
Fx	REAL	Eingabegröße in Grad Réaumur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.2.3.2.2.13 F_BA_ReaumurToKelvin



Die Funktion F_BA_ReaumurToKelvin vom Rückgabetyp REAL dient der Umrechnung einer Temperaturgröße in Grad Réaumur zu einer Temperaturgröße in Kelvin.

Syntax

```
FUNCTION F_BA_ReaumurToKelvin : REAL
VAR_INPUT
  fX    : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingabegröße in Grad Réaumur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.62	Tc3_BA2 ab v5.5.14.0

6.1.2.3 Archiv

6.1.2.3.1 Tc2_BA

Diese Bibliothek wurde wegen ihrer bewährten Funktionalitäten aus dem Supplement TS8040 aus TwinCAT 2 importiert. Daher ist die Dokumentation, bis auf die Bausteine die BACnet betreffen, identisch (siehe Tc_BA).

Die BACnet spezifischen Bausteine finden Sie nun in der Bibliothek *Tc2_BACnetRev12*.

6.1.2.3.2 Tc3_BA

Diese Bibliothek dient zur Wartung und Instandhaltung von bestehenden Projekten. Für neue Projekte verwenden Sie bitte die Bibliothek Tc3_BA2.

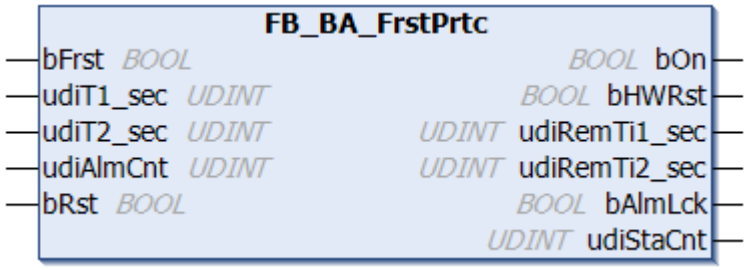
6.1.2.3.2.1 POUs

6.1.2.3.2.1.1 Klimatechnik

Funktionsbausteine

Name	Beschreibung
FB_BA_FrstPrtc [▶ 537]	Überwachung Frostalarm und Noterhitzung
FB_BA_HX [▶ 539]	Berechnung Taupunkttemperatur, spezifische Enthalpie und die absolute Feuchte
FB_BA_NgtCol [▶ 539]	Sommernachtkühlung
FB_BA_RcvMonit [▶ 541]	Funktionsbaustein zur Berechnung des Wirkungsgrades einer Energie-Rückgewinnung
FB_BA_SPSupvis [▶ 543]	Funktionsbaustein zur Verarbeitung und Kontrolle des unteren und oberen Sollwertes einer Zuluftfeuchte- oder Zulufttemperaturregelung

6.1.2.3.2.1.1.1 FB_BA_FrstPrtc



Der Funktionsbaustein dient zur Frostüberwachung eines Heizregisters in einer raumlufttechnischen Anlage.

Frostgefahr steht an, wenn der Eingang *bFrst* TRUE ist. Der Frostalarm muss im Anlagenprogramm so verknüpft werden, dass die Anlage sofort abschaltet, das Erhitzerventil öffnet und die Erhitzerpumpe eingeschaltet wird.

Bei Frostgefahr wird der Ausgang *bOn* gesetzt und *udiT1_sec* (Sekunden) gestartet. Ist nach Ablauf von *udiT1_sec* weiterhin Frostgefahr (*bFrst*=TRUE) dann bleibt *bOn* gesetzt. Ein Rücksetzen kann nur an dem Eingang *bRst* erfolgen.

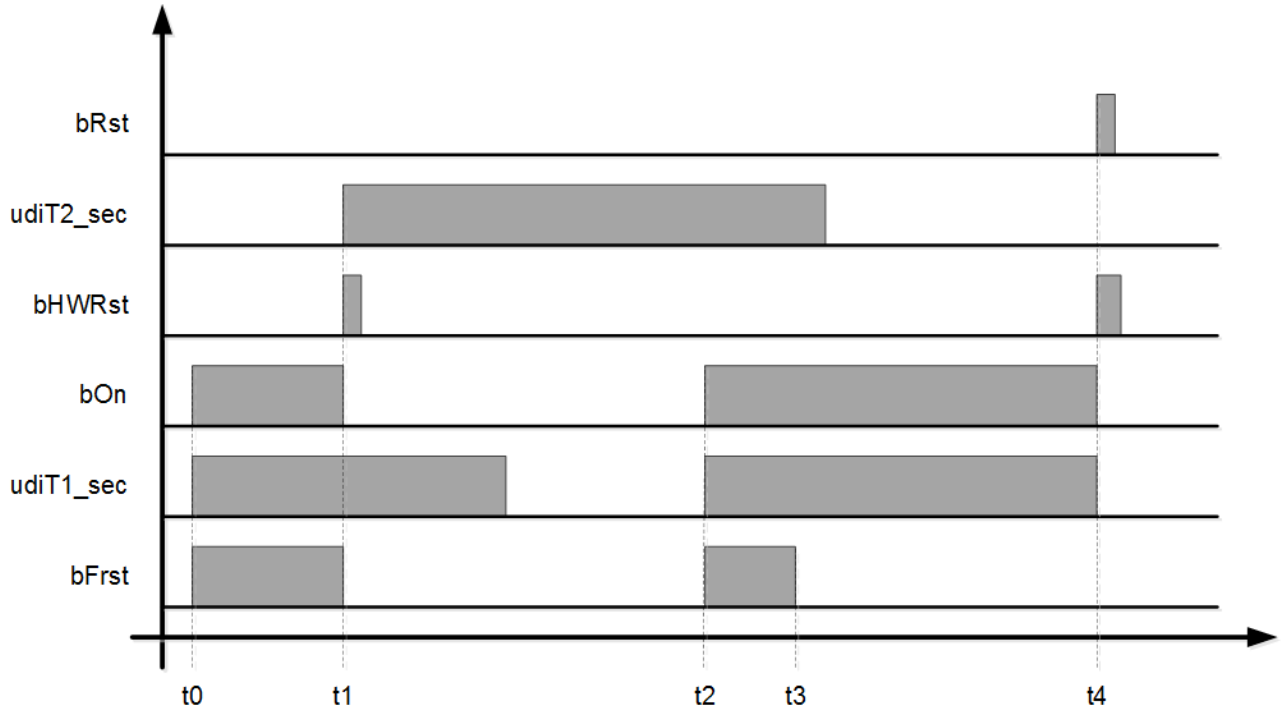
Erlischt der Frostalarm durch das Aufheizen des Heizregisters innerhalb des Zeitraums von *udiT1_sec* (*bFrst*=FALSE), so erfolgt ein automatischer Neustart der Anlage. Für den Anlagenneustart wird *bOn* FALSE und am Ausgang *bHWRst* wird ein Impuls zur Quittierung einer Selbsthaltungsschaltung im Schaltschrank ausgegeben. Mit dem Neustart wird eine zweite Überwachungszeit *udiT2_sec* (Sekunden) gestartet. Erfolgt innerhalb dieses Zeitraums erneut ein Frostalarm, so wird die Anlage endgültig verriegelt. *bOn* bleibt so lange gesetzt bis der Frostalarm beseitigt und an *bRst* quittiert ist.

Falls die Frostalarme aufeinander folgend, aber immer mit einem Zeitversatz größer als die Zeit von *udiT2_sec* erfolgen, würde theoretisch beliebig oft automatisch neu gestartet werden. Um dieses zu vermeiden werden die Neustarts innerhalb des Bausteins gezählt. Mit *udiAlmCnt* ist die Zahl der möglichen automatischen Neustarts zwischen 0 und 4 einstellbar.

Eine Quittierung am Eingang *bRst* setzt den Alarmspeicher innerhalb des Bausteins auf Null zurück.

Beispiel:

t0 = Frostalarm an Eingang *bFrst*, Alarmmeldung an Ausgang *bOn*, Starten des Timers T1 (*udiT1_sec* [s])
t1 = Frostalarm erloschen, Rücksetzen von *bOn*, Ausgabe Hardwareimpuls, Starten von Timer T2 (*udiT2_sec* [s]), Anlagenneustart
t2 = Erneuter Frostalarm innerhalb von T2, Alarmmeldung an *bOn*, Starten des Timers T1, verriegeln des Frostalarms
t3 = Frostalarm erloschen.
t4 = Quittierung des Alarms an *bRst*, Rücksetzen von *bOn*.



VAR_INPUT

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

bFrst: Anschluss für die Frostereignisse luft- und wasserseitig.

udiT1_sec: Timer zur Verzögerung von Neustarts [s]. Intern begrenzt auf einen Minimalwert von 0.

udiT2_sec: Timer Überwachungszeit [s]. Intern begrenzt auf einen Minimalwert von 0.

udiAlmCnt: Maximale Zahl der automatischen Anlagenneustarts ohne Rücksetzen. Intern begrenzt auf Werte von 0 bis 4.

bRst: Rücksetzen bzw. quittieren des Frostalarms.

VAR_OUTPUT

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

bOn: Frostalarm aktiv.

bHWRst: Ausgabe eines Impulses zur Quittierung der Frostschutzhardware.

udiRemTi1_sec: Verbleibende Zeit bis Anlagenneustart nach Frostalarm.

udiRemTi2_sec: Verbleibende Überwachungszeit.

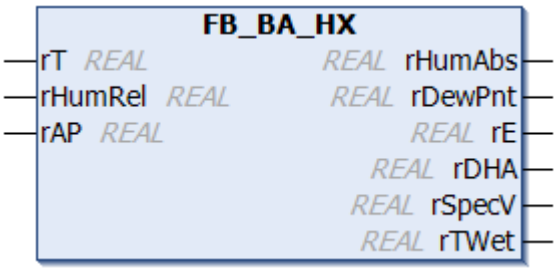
bAlmLck: Alarm Lock - gespeicherter Alarm.

udiStaCnt: Status Counter – aktuelle Anzahl nicht quittierter Fehlstarts.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.1.2 FB_BA_HX



Mit diesem Funktionsbaustein werden die Taupunkttemperatur, die spezifische Enthalpie und die absolute Feuchte berechnet. Für die Berechnung der Größen werden die Temperatur, die relative Feuchte und der barometrische Luftdruck benötigt.
Die Enthalpie ist ein Maß für die Energie eines thermodynamischen Systems.

VAR_INPUT

```
rT      : REAL;
rHumRel : REAL;
rAP     : REAL;
```

rT: Temperatur [°C].

rHumRel: Relative Feuchte [%].

rAP: Hydrostatischer Luftdruck 1013.25 hPa.

VAR_OUTPUT

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

lrHumAbs: Absolute Feuchte in g Wasser je kg trockener Luft [g/Kg].

lrDewPnt: Taupunkttemperatur [°C].

lrE: Enthalpie [kJ/kg].

lrDHA: Dichte feuchter Luft ρ [kg Gemisch/m³].

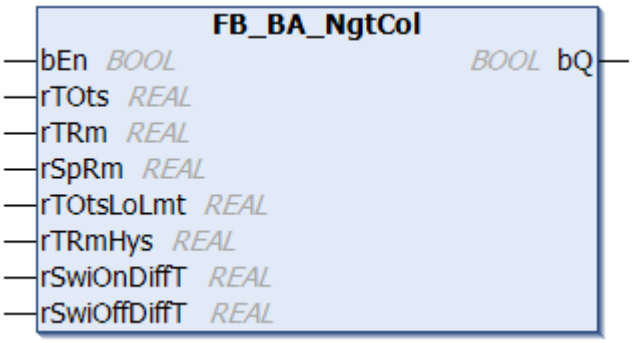
lrSpecV: spezifisches Volumen [m³/kg].

lrTWet: Feuchtkugeltemperatur [°C].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.1.3 FB_BA_NgtCol



Dieser Funktionsbaustein wird verwendet um nachts mit kühler Außenluft die tags zuvor aufgeheizten Räume herunter zu kühlen. Die Funktion der Sommernachtkühlung dient zur Verbesserung der Luftqualität und zur Einsparung elektrischer Energie. In den ersten Stunden des nächsten Sommertages wird elektrische Energie zur Kälteerzeugung gespart.

Durch Parametrierung des Funktionsbausteins *FB_BA_NgtCol* werden die Startbedingungen für die Sommernachtkühlung definiert. Der Baustein kann verwendet werden, um motorisch betätigte Fenster zu öffnen oder Klimaanlagen außerhalb ihrer normalen Betriebszeiten in den Sommernachtkühlbetrieb zu schalten.

Folgende Bedingungen müssen für die Aktivierung der Sommernachtkühlung erfüllt sein:

- Der Baustein selbst ist aktiviert (*bEn* = TRUE).
- Die Außentemperatur ist nicht zu niedrig (*rTOts* > *rTOtsLoLmt*).
- Die Außentemperatur ist im Vergleich zur Raumtemperatur ausreichend niedrig (*rTRm* - *rTOts*) > *rSwiOnDiffT.*
- Die Raumtemperatur ist so hoch, so dass es sich lohnt die Sommernachtkühlung einzuschalten. *rTRm* > *rSpRm* + *rTRmHys.*

Bei den folgenden Bedingungen wird die Sommernachtkühlung deaktiviert:

- Der Baustein selbst ist deaktiviert (*bEn* = FALSE).
- Die Außentemperatur ist zu niedrig (*rTOts* < *rTOtsLoLmt*).
- Die Außentemperatur ist im Vergleich zur Raumtemperatur zu hoch (*rTRm* - *rTOts*) < *rSwiOffDiffT.*
- Die Raumtemperatur ist niedriger als der Sollwert. *rTRm≤rSpRm.*

VAR_INPUT

```
bEn           : BOOL;
rTOts         : REAL;
rTRm          : REAL;
rSpRm         : REAL;
rTOtsLoLmt    : REAL;
rTOtsHys      : REAL;
rTRmHys       : REAL;
rSwiOnDiffT   : REAL;
rSwiOffDiffT : REAL;
```

bEn: Freigabe des Funktionsbausteins.

rTOts: Außentemperatur [°C].

rTRm: Außentemperatur [°C].

rSpRm: Raumtemperatur-Sollwert.

rTOtsLoLmt: Untere Grenze der Außentemperatur [°C], verhindert ein zu starkes Auskühlen.

rTOtsHys: Hysterese für die minimale Außentemperatur [K]. Diese Hysterese, welche nach unten hin intern auf 0.5K begrenzt ist, soll ein Flattern von *bQ* verhindern, falls die Außentemperatur genau um den Wert von *rTOtsLoLmt* pendelt.

rTRmHys: Hysterese für die Raumtemperatur [K]. Diese Hysterese, welche nach unten hin intern auf 0.5K begrenzt ist, soll ein unnötiges Schwanken von *bQ* verhindern, falls die Raumtemperatur genau um den Sollwert *rSpRm* pendelt.

rSwiOnDiffT: Differenz zwischen der Raumtemperatur und der Außentemperatur ab der die Sommernachtkühlung frei gegeben wird [K].

rSwiOffDiffT: Differenz zwischen der Raumtemperatur und der Außentemperatur ab der die Sommernachtkühlung gesperrt wird [K].

VAR_OUTPUT

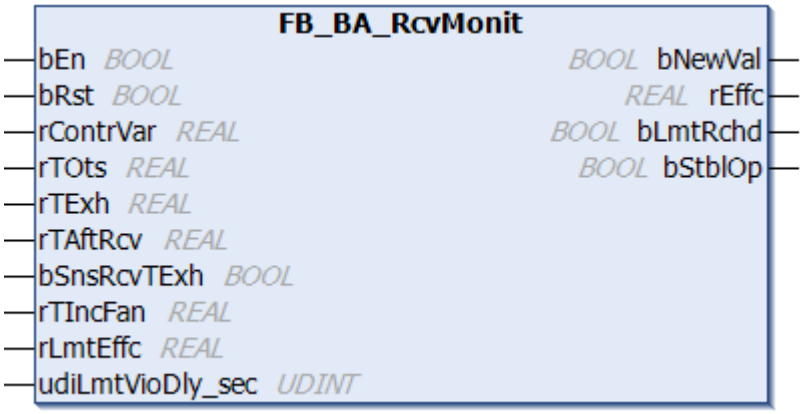
```
bQ  : BOOL;
```

bQ: Sommernachtkühlung Ein.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

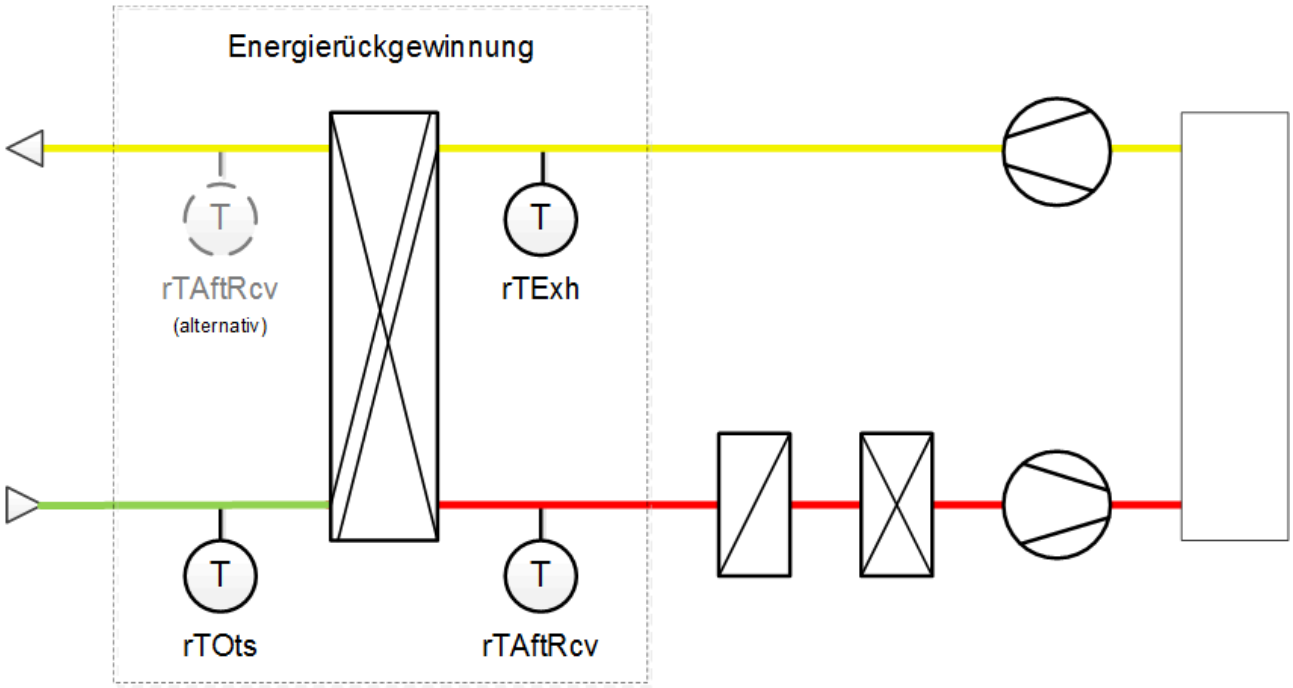
6.1.2.3.2.1.1.4 FB_BA_RcvMonit



Der Funktionsbaustein dient zur Berechnung des Wirkungsgrades einer Energie-Rückgewinnung.

Der Baustein benötigt zur Berechnung des Wirkungsgrades (der sogenannten Rückwärmzahl) folgende Temperaturmesswerte:

- Außenlufttemperatur *rTOts*
- Fortlufttemperatur *rTExh*
- Lufttemperatur der Energie-Rückgewinnung im Zuluftkanal (alternativ: im Abluftkanal) *rTAftRcv*



Der Baustein nimmt nun alle 10 s die Temperatur-Messwerte auf und bildet aus jeweils 6 Werten den minütlichen Mittelwert. Danach wird geprüft, ob die Anlage sich in einem eingeregelten, "stabilen" Zustand befindet:

- Die aufgenommenen Temperaturen Außenluft, Fortluft und Luft nach Energierückgewinnung sind annähernd konstant, d.h. keiner der 6 Einzelwerte weicht mehr als 0,5 K vom jeweiligen Mittelwert ab.
- Die Temperaturdifferenz Außenluft zu Fortluft beträgt mindestens 5 K.

Ist dies der Fall, so wird dieser Messzyklus mit einem TRUE-Signal am Ausgang *bStblOp* quittiert und der ermittelte Wirkungsgrad an *rEffc* ausgegeben. Ist der Zustand hingegen nicht "stabil", so erscheint am Ausgang *bStblOp* ein FALSE-Signal und *rEffc* wird auf 0 gesetzt.

In jedem Fall jedoch wird jeder Mess- und Auswerte-Zyklus mit einem Trigger (ein Ein-PLC-Zyklus langes TRUE-Signal) an *bNewVal* als beendet gekennzeichnet.

Freigabe (*bEn*) und Reset (*bRst*)

Der Baustein arbeitet nur dann, wenn ein TRUE-Signal an *bEn* anliegt. Andernfalls stoppt er seine Abarbeitung und alle Ausgänge werden auf FALSE bzw. 0.0 gesetzt.

Während der Abarbeitung kann ein Mess-und Auswertezyklus jederzeit durch ein TRUE-Signal an *bRst* abgebrochen werden. Alle Ausgänge werden auf FALSE bzw. 0.0 gesetzt und der Messzyklus startet automatisch neu.

Auswahl des Temperaturwertes "nach Rückgewinnung" (*bSnsRcvTExh*)

Ein FALSE-Eintrag an *bSnsRcvTExh* bedeutet, dass zur Berechnung des Wirkungsgrades die Temperaturmessung nach der Wärmerückgewinnung im **Zuluftkanal** verwendet wird.
Soll hingegen die Temperaturmessung nach der Wärmerückgewinnung im **Abluftkanal** angewendet werden, so muss an *bSnsRcvTExh* ein TRUE angelegt werden.

Grenzwertüberschreitung (*rContrVar, rLmtEffc, bLmtRchd*)

Eine Grenzwertüberschreitung liegt dann vor, wenn der ermittelte Wirkungsgrad kleiner ist als der eingetragene Grenzwert *rLmtEffc* und gleichzeitig die Stellgröße für die Wärmerückgewinnung bei 100 % liegt. Dazu muss die Stellgröße mit dem Eingang *rContrVar* verbunden werden.
Die Meldung der Grenzwertüberschreitung lässt sich durch einen Eintrag an *udiLmtVioDly_sec* [s] verzögern: Liegen die beiden Kriterien, Überschreitung und Übersteuerung länger als *udiLmtVioDly _sec* [s] vor, so wird dieses mit einem TRUE-Signal an *bLmtRchd* angezeigt.
Eine aufgetretene Warnmeldung erlischt, wenn ein kompletter Messzyklus "gute" Werte erbringt oder bei einer steigenden Flanke an *bRst* bzw. einer Deaktivierung des Bausteines.



Diese Warnmeldung tritt nur auf, wenn die Anlage in einen stabilen Betrieb ist (*bStblOp*=TRUE).

Berücksichtigung Temperaturanstieg der Abluft durch Ventilatormotor (*rTIncFan*)

Es ist möglich, dass die Abluft durch einen Ventilatormotor erwärmt und die Messung damit verfälscht wird. Diese Temperaturerhöhung kann durch *rTIncFan* angegeben werden. Intern wird dann die gemessene Ablufttemperatur um diesen Wert gemindert.

VAR_INPUT

```
bEn               : BOOL;
bRst              : BOOL;
rContrVar         : REAL;
rTOts             : REAL;
rTExh             : REAL;
rTAftRcv          : REAL;
bSnsRcvTExh       : BOOL;
rTIncFan          : REAL;
rLmtEffc          : REAL;
udiLmtVioDly_sec  : DINT;
```

bEn: Freigabe des Bausteines.

bRst: Reset - alle ermittelten Werte werden gelöscht.

rContrVar: Stellgröße für die Wärmerückgewinnung, d.h. ihr Istwert.

rTOts: Außentemperatur.

rTExh: Fortlufttemperatur.

rTAftRcv: Temperatur nach Energierückgewinnung.

bSnsRcvTExh: Temperatur am Messort nach Energierückgewinnung: FALSE -> im Zuluftkanal (SupplyAir) - TRUE -> im Abluftkanal (ExhaustAir).

rTIncFan: Temperaturanstieg durch Ventilator.

rLmtEffc: Wirkungsgrad Grenzwert.

udiLmtVioDly_sec: Zeitverzögerung Grenzwertüberschreitung [s]. Intern begrenzt auf einen Minimalwert von 0.

VAR_OUTPUT

```
bNewVal      : BOOL;
rEffc        : REAL;
bLmtRchd     : BOOL;
bStblOp      : BOOL;
```

bNewVal: Ausgabetrigger neuer Wert *rEffc*.

rEffc: Effektivität

bLmtRchd: Grenzwert erreicht

bStblOp: Stabiler Betrieb.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.1.5 FB_BA_SpSupvis



Funktionsbaustein zur Verarbeitung und Kontrolle des unteren und oberen Sollwertes einer Zuluftfeuchte- oder Zulufttemperaturregelung.

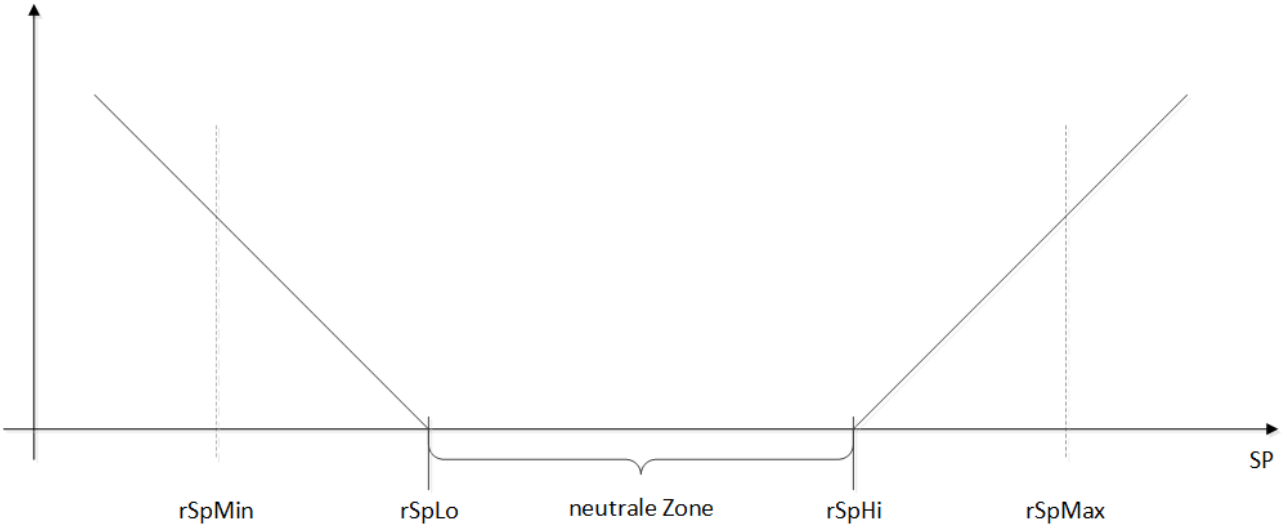
Überprüfung und Limitierung der Sollwerte

Der Funktionsbaustein limitiert und begrenzt die Sollwerte. Die beiden folgenden Tabellen stellen dar was überprüft wird und wie im Fehlerfall reagiert wird.

Überprüfung	Aktion
rSpLo > rSpHi	letzten gültigen Werte von rSpLo und rSpHi werden genommen
rSpMin >= rSpMax	letzten gültigen Werte von rSpMin und rSpMax werden genommen
rSpHi > rSpMax	rPrSpHi = rSpMax
rSpLo < rSpMin	rPrSpLo = rSpMin

Überprüfung	bErr	Aktion
rSpMin >= rSpMax	TRUE	rSpErr = ((rSpMin + rSpMax) / 2) rPrSpHi = rPrSpLo = rPrRcv = rSpErr
rSpHi < rSpMin		
rSpLo > rSpMax		

Der Abstand zwischen den Sollwerten beschreibt eine Energieneutrale Zone. Bei der Verwendung in einer Zuluftregelung würde innerhalb der Neutralen Zone nicht geheizt und nicht gekühlt werden.



Die überprüften und gegebenenfalls limitierten Sollwerte werden am Bausteinausgang als *rPrSpHi* und *rPrSpLo* (Present Setpoint) ausgegeben.

Sollwert für Wärmerückgewinnung

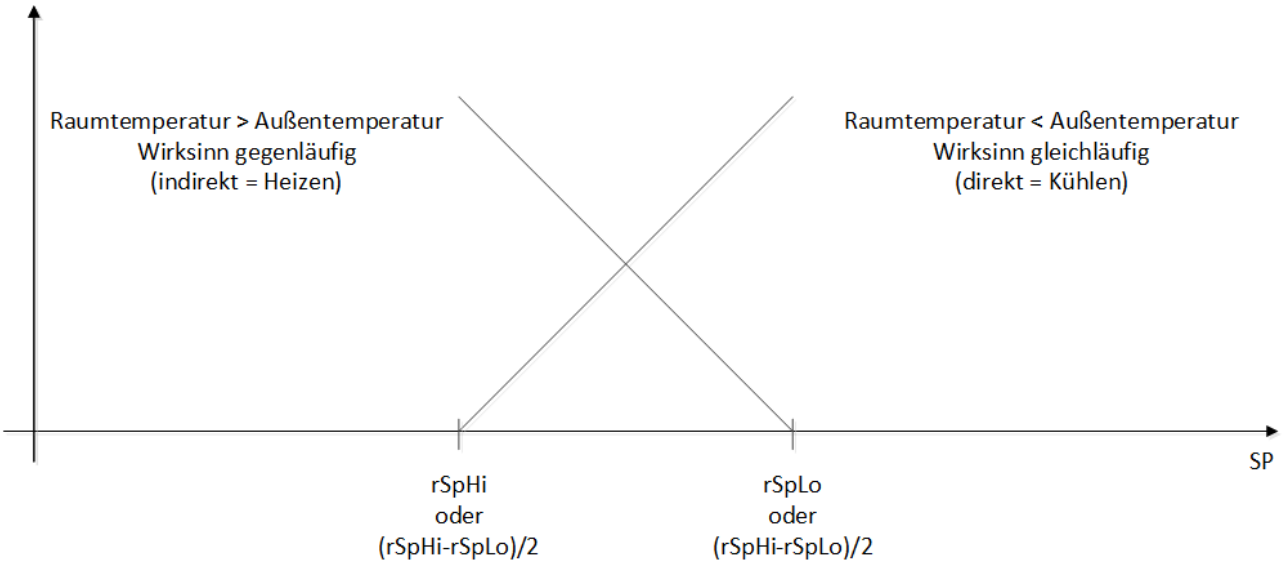
Für eine Wärmerückgewinnung wird der Sollwert *rSpRcv* wahlweise aus dem Mittelwert des oberen und unteren Sollwertes, *rSpHi* und *rSpLo* berechnet oder aber in Abhängigkeit vom Wirksinn der Wärmerückgewinnung (WRG).
Die Methode wird durch die Eingangsvariable *bSlcnSpRcv* definiert:

b SlcnSpRcv	rSpRcv
TRUE	Mittelwert aus *rSpLo* und *rSpHi*
FALSE	Wirksinnabhängig, definiert durch Eingang bActRcv

Im Falle der Wirksinnabhängigen Definition des Sollwertes gilt dann:

bActRcv	Wirksinn	rSpRcv
TRUE	direkt (Kühlen)	*rSpHi*
FALSE	indirekt (Heizen)	*rSpLo*

Wärmerückgewinnung (WRG)



VAR_INPUT

```
bEn         : BOOL;
rSpHi       : REAL;
rSpLo       : REAL;
rSpMax      : REAL;
rSpMin      : REAL;
bActnRcv    : BOOL;
bSlcnSpRcv  : BOOL;
```

bEn :Bausteinfreigabe. Wenn *bEn* = FALSE, stehen alle Ausgabeparameter auf 0.0.

rSpHi: Zu überprüfender Eingabewert des oberen Sollwertes.

rSpLo: Zu überprüfender Eingabewert des unteren Sollwertes.

rSpMax: Maximaler Sollwert.

rSpMin: Minimaler Sollwert.

bActnRcv: Wirkrichtung der nachgeschalteten Wärmerückgewinnung.

bSlcnSpRcv: Sollwert-Wahl der nachgeschalteten Wärmerückgewinnung.

VAR_OUTPUT

```
rPrSpHi    : REAL;
rPrSpLo    : REAL;
rSpRcv     : REAL;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
```

rPrSpHi: Ausgabewert des oberen Sollwertes.

rPrSpLo: Ausgabewert des unteren Sollwertes.

rSpRcv: Ausgabewert des resultierenden Sollwertes für die Wärmerückgewinnung.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Warnung: Die Sollwerte sind nicht in einer logischen Reihenfolge: Entweder (*rSpMin* >= *rSpMax*) OR (*rSpHi* < *rSpMin*) OR (*rSpLo* > *rSpMax*)

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

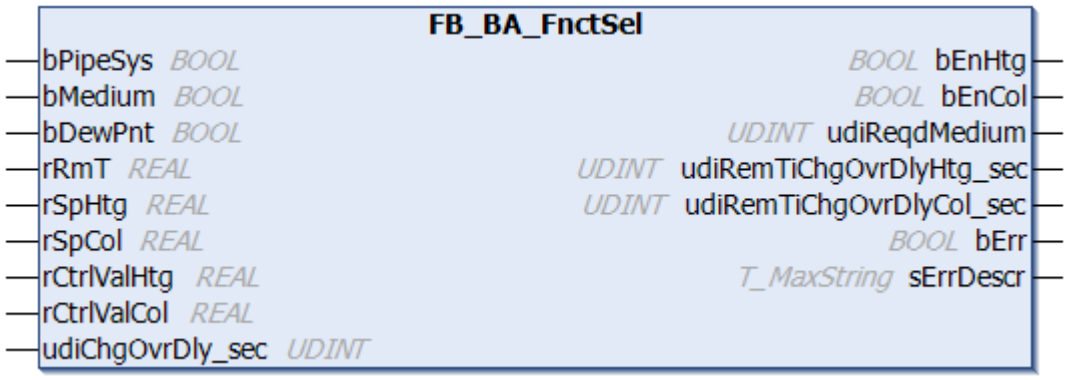
6.1.2.3.2.1.2 Raumautomation

6.1.2.3.2.1.2.1 Heizen, Kühlen

Funktionsbausteine

Name	Beschreibung
FB_BA_FnctSel [▶ 546]	Funktionsanwahl (Heizen und/oder Kühlen) in einen Zwei- oder Vierleiternetz.
FB_BA_RmTAdj [▶ 549]	Anpassung des Raumtemperatursollwertes.
FB_BA_SpRmT [▶ 552]	Anpassung des Raumtemperatursollwertes

6.1.2.3.2.1.2.1.1 FB_BA_FnctSel



Der Funktionsbaustein dient der Freigabe des Heiz- oder Kühlbetriebs in einem Raum.
Dabei spielt die Art des Verteilungsnetzes eine große Rolle:
Handelt es sich um ein Zweileitersystem, so können alle Räume der Anlage immer nur entweder beheizt oder gekühlt werden.
In einem Vierleitersystem hingegen kann die Klimatisierung der Räume bedarfsgerecht erfolgen, d.h. ein Teil der Räume kann beheizt, der andere von derselben Anlage gekühlt werden.

Der Funktionsbaustein, der wie bereits erwähnt pro Raum Anwendung findet, wählt damit je nach vorhandenem Rohrleitungssystem seine Regler unterschiedlich an:

Zweileiternetz

Das Zweileitersystem ist angewählt, wenn der Baustein am Eingang *bPipeSys* einen FALSE-Eintrag hat. Da alle Räume der Anlage nur entweder beheizt oder gekühlt werden können, wird die Anwahl von zentraler Stelle für alle Räume gleich über den Eingang *bMedium* vorgegeben. Ist *bMedium* FALSE, so wird der Raum-Heizregler angewählt - ist der Eingang TRUE, so ist es der Kühlregler. Die Reglerfreigaben *bEnHtg* und *bEnCol* werden immer gegenseitig verzögert um *udiChgOvrDly_sec* [s] ausgegeben, das heißt: erst wenn die Freigabe Kühlen *bEnCol* für *udiChgOvrDly_sec* FALSE ist, kann die Freigabe Heizen gegeben werden und umgekehrt. Zusätzlich zum Ablauf dieser Umschaltzeit wird darauf geachtet, dass der abzuschaltende Regler 0.0 ausgibt. Dies geschieht durch Rückmeldung an den Eingängen *rCtrlValHtg* und *rCtrlValCol*. So wird ein drastischer Wechsel von Heizen auf Kühlen und umgekehrt vermieden.

Vierleiternetz

Das Vierleitersystem ist angewählt, wenn der Baustein am Eingang *bPipeSys* einen TRUE-Eintrag hat. Die Regleranwahl kann nun für alle Räume nach Bedarf unterschiedlich sein und richtet sich daher nach der Raumtemperatur *rRmT* und den eingestellten Sollwerten *rSpHtg* für Heizen und *rSpCol* für Kühlen.

Übersteigt die Raumtemperatur den Kühlsollwert, so wird der Kühlregler freigegeben (*bEnCol*), fällt sie unter den Heizsollwert, wird der Heizregler freigegeben (*bEnHtg*). Befindet sich die Temperatur zwischen beiden Sollwerten, so sind beide Regler abgeschaltet (energieneutrale Zone). Auch hier werden die Reglerfreigaben *bEnHtg* und *bEnCol* immer gegenseitig verzögert um *udiChgOvrDly_sec* [s] ausgegeben (siehe Zweileiternetz) und zusätzlich zum Ablauf dieser Umschaltzeit darauf geachtet, dass der abzuschaltende Regler 0.0 ausgibt. Dies geschieht durch Rückmeldung an den Eingängen *rCtrlValHtg* und *rCtrlValCol*. So wird ein drastischer Wechsel von Heizen auf Kühlen und umgekehrt vermieden, falls die Umschaltzeit nicht ausreichend hoch bemessen ist.

Taupunktwächter (*bDewPnt*)

In beiden Systemen - Zwei- und Vierleiter - hat der Taupunktwächter die Aufgabe, das Kühlen unmittelbar zu verriegeln.

Programmablauf

Der Baustein kennt 3 Zustände:

1. Warten auf Freigabe Heizen bzw. Kühlen
2. Freigabe Heizen
3. Freigabe Kühlen

Im ersten Schritt wartet der Baustein auf die Erfüllung der Bedingungen, die für das Heizen bzw. das Kühlen notwendig sind:

Heizen	Kühlen
Ausgabe Kühlregler = 0 (*rCtrlValCol*)	Ausgabe Heizregler = 0 (*rCtrlValHtg*)
Raumtemperatur (*rRmT*) < Sollwert Heizen (*rSpHtg*)	Raumtemperatur (*rRmT*) > Sollwert Kühlen (*rSpCol*)
Reglerfreigabe Kühlen (*bEnCol*) steht mindestens für die Umschaltzeit *udiChgOvrDly_sec* [s] auf FALSE	Reglerfreigabe Heizen (*bEnHtg*) steht mindestens für die Umschaltzeit *udiChgOvrDly_sec* [s] auf FALSE
Vierleitersystem ist angewählt (*bPipesys*=TRUE) ODER Zweileitersystem ist angewählt und es steht Heizmedium an (*bPipeSys*=FALSE AND *bMedium*=FALSE)	Vierleitersystem ist angewählt (*bPipesys*=TRUE) ODER Zweileitersystem ist angewählt und es steht Kühlmedium an (*bPipeSys*=FALSE AND *bMedium*=TRUE)
	Der Taupunktwächter spricht nicht an (*bDewPnt*=TRUE)

Ist nun eine Bedingungskette erfüllt, springt der Baustein in den jeweiligen Zustand Heizen oder Kühlen und beendet diesen erst, wenn der entsprechende Regler am Bausteineingang (*rCtrlValHtg*/*rCtrlValCol*) 0 ausgibt. Damit ist sichergestellt, dass immer nur ein Regler aktiv ist, auch wenn eine zu hohe Reglerausgabe, z. B. Heizen, u. U. ein kurzfristiges Kühlen erforderlich machen würde (Überschwingen). Das Heizen, bzw. Kühlen wird dann erst abgeschlossen, wenn kein Bedarf mehr ansteht.

Allerdings gibt es 3 Ausnahmen, bei denen das Heizen bzw. Kühlen unmittelbar unterbrochen wird:

1. Im Zweileitersystem (*bPipeSys*=FALSE) wird geheizt (*bEnHtg*), es wurde jedoch auf Kühlmedium umgestellt *bMedium*=TRUE
2. Im Zweileitersystem (*bPipeSys*=FALSE) wird gekühlt (*bEnCol*), es wurde jedoch auf Heizmedium umgestellt *bMedium*=FALSE
3. Im Kühlbetrieb (Zwei- oder Vierleitersystem) hat der Taupunktwächter angesprochen (*bDewPnt*=TRUE)

In diesen Fällen werden die Heiz- und Kühlfreigaben weggenommen und zurück in den Wartezustand gesprungen.

Bedarfsmitteilung (*udireqdMedium*)

Um der Anlage mitzuteilen, wie der aktuelle Bedarf an Wärme oder Kühlung ist, wird am Bausteinausgang, also pro Raum, in Abhängigkeit der Ist- und Solltemperatur eine Bedarfskennung ausgegeben. Diese kann zentral gesammelt und ausgewertet werden. Die Auswertung erfolgt immer, also unabhängig vom vorhandenen Netzsystem (Zwei- oder Vierleiter).

udiReqdMedium	Medium	Raumtemperatur
1	kein Medium ist angefordert	*rRmT* > *rSpHtg* UND *rRmT* < *rSpCol*
2	Heizmedium ist angefordert	*rRmT* < *rSpHtg*
3	Kühlmedium ist angefordert	*rRmT* > *rSpCol*

Fehlerbehandlung

Der Heiz-Sollwert darf nicht größer oder gleich dem Kühl-Sollwert sein, da damit ein Temperaturbereich entstehen würde, in dem gleichzeitig Heiz- und Kühlbedarf entsteht. Da der Baustein jedoch immer nur eine Freigabe, Heizen oder Kühlen ausgibt, ist der Fall anlagentechnisch unbedenklich. Es wird für diesen Fall nur eine Warnmeldung (*bErr*=TRUE, *sErrDescr*=Warnmeldung) ausgegeben - der Baustein unterbricht seine Abarbeitung nicht.

VAR_INPUT

```
bPipeSys          : BOOL;
bMedium           : BOOL;
bDewPnt           : BOOL;
rRmT              : REAL;
rSpHtg            : REAL;
rSpCol            : REAL;
rCtrlValHtg       : REAL;
rCtrlValCol       : REAL;
udiChgOvrDel_sec  : UDINT;
```

bPipeSys: bei Zweileitersystem ist *bPipeSys* FALSE, bei Vierleitersystem TRUE.

bMedium: aktuelle Versorgung des gesamten Zweileiternetzes mit Kühl- oder Heizmittel. Wenn Heizmedium ansteht ist *bMedium* FALSE.

bDewPnt: Taupunktwächter: wenn *bDewPnt* = FALSE, dann wird der Kühlregler gesperrt.

rTRm: Raumtemperatur.

rSpHtg: Heiz-Sollwert.

rSpCol: Kühl-Sollwert.

rCtrlValHtg: aktueller Ausgabewert vom Heizregler. Wird intern als Umschaltkriterium von Heizen auf Kühlen verwendet: *rCtrlValHtg* muss 0 sein.

rCtrlValCol: aktueller Ausgabewert vom Kühlregler. Wird intern als Umschaltkriterium von Kühlen auf Heizen verwendet: *rCtrlValCol* muss 0 sein.

udiChgOvrDel_sec: Umschaltverzögerung [s] von Heizen auf Kühlen und umgekehrt. Intern begrenzt auf einen Minimalwert von 0.

VAR_OUTPUT

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

bEnHtg: Freigabe des Reglers Heizen.

bEnCol: Freigabe des Reglers Kühlen.

udiReqdMedium:

udiReqdMedium	Medium	Raumtemperatur
1	kein Medium ist angefordert	*rRmT* > *rSpHtg* UND *rRmT* < *rSpCol*
2	Heizmedium ist angefordert	*rRmT* < *rSpHtg*
3	Kühlmedium ist angefordert	*rRmT* > *rSpCol*

udiRemTiChgOvrDlyHtg_sec: Countdown [s] Umschaltverzögerung von Kühlen auf Heizen.

udiRemTiChgOvrDlyCol_sec: Countdown [s] Umschaltverzögerung von Heizen auf Kühlen.

bErr: Bei fehlerhafter Abarbeitung, bzw. bei Warnzuständen wird dieser Ausgang auf TRUE gesetzt.

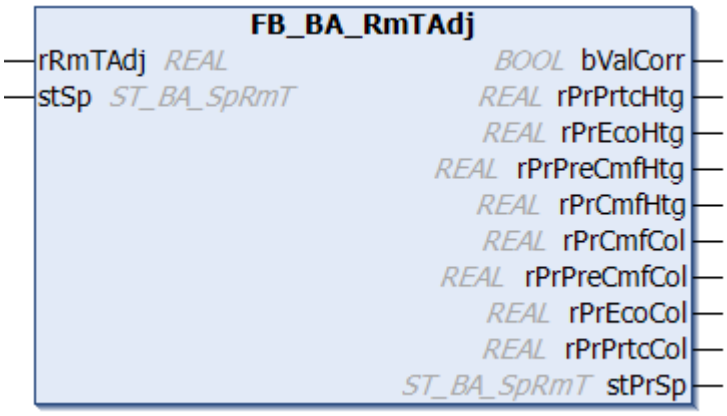
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Warnung: Der Sollwert Heizen (Setpoint Heating) ist größer als oder gleich dem Sollwert Kühlen (Setpoint Cooling)

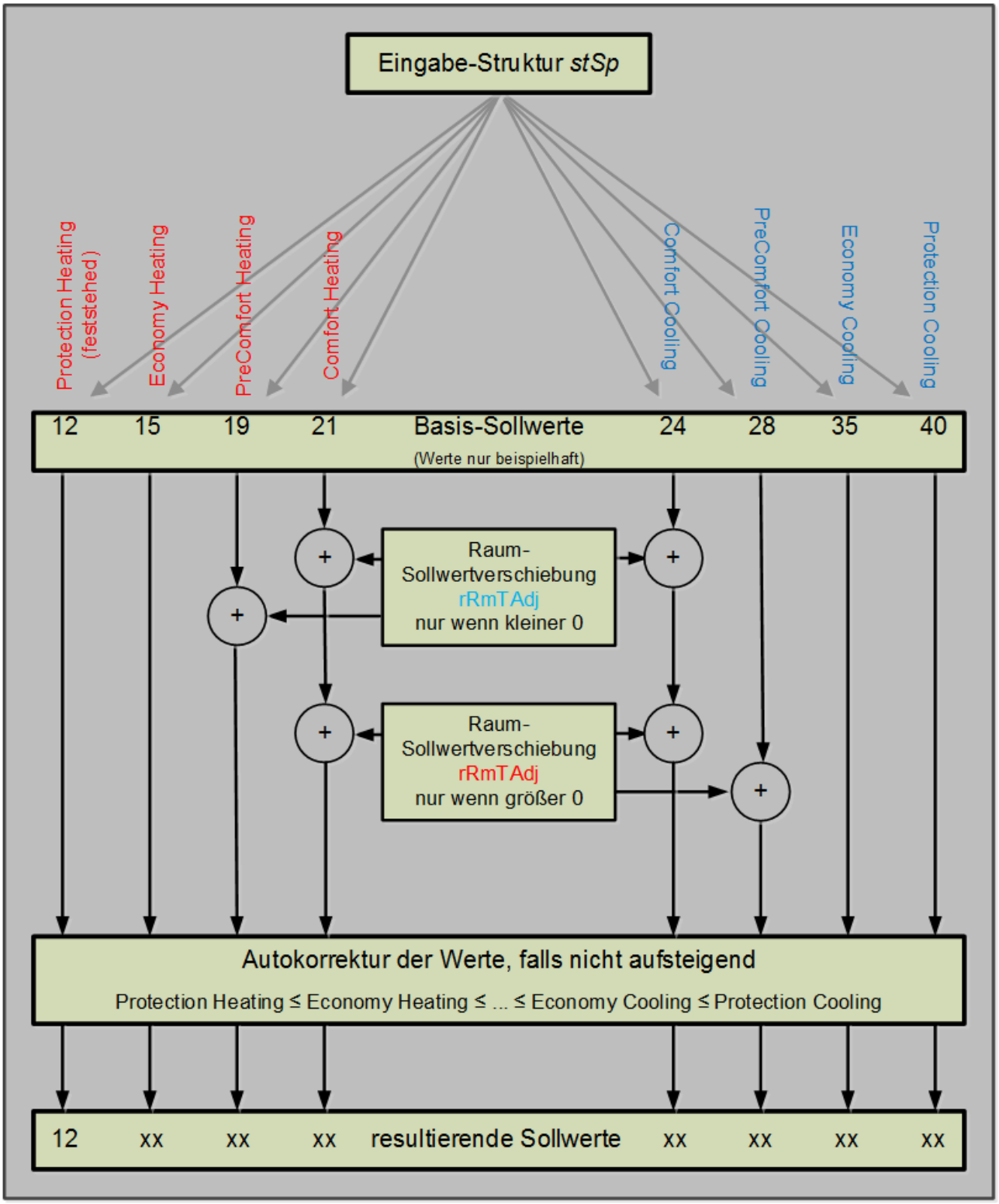
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.1.2 FB_BA_RmTAdj



Der Funktionsbaustein *FB_BA_RmTAdj* dient der Anpassung des Raumtemperatursollwertes vom Nutzer im Raum. Er verschiebt die Sollwerte am Eingang des Funktionsbausteins in Abhängigkeit eines Offsets *rRmTAdj*, wie in der folgenden Skizze dargestellt. Am Eingang *rRmTAdj* kann für die Sollwertkorrektur z. B. der Wert eines Widerstandspotentiometers oder eines busfähigen Feldgerätes verwendet werden.



Ist der Einstellwert *rRmTAdj* größer als Null, so wird eine Raumtemperatur-Erwärmung gewünscht: Der Wert Comfort Heating wird um den Wert *rRmTAdj* angehoben. Gleichzeitig werden auch die Werte Comfort Cooling und Precomfort Cooling erhöht. Ist der Wert *rRmTAdj* hingegen kleiner als Null, so wird eine Raumtemperatur-Absenkung erwünscht. Analog zum Erwärmungsfall werden nun die Werte Comfort Cooling, Comfort Heating und PreComfort Heating um den Wert *rRmTAdj* herabgesetzt.

Auto-Korrektur

Die Temperaturanpassung ist dazu gedacht, die Werte in kleinem Rahmen zu beeinflussen. Natürlich können beliebige Eingangswerte eingegeben werden. Eine Heizungsanlage wird jedoch nur dann sinnvoll funktionieren, wenn die Sollwerte in folgender Reihenfolge aufsteigende Werte besitzen:

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

Die Auto-Korrektur arbeitet nach folgendem Prinzip: Beginnend mit dem Wert Economy Heating wird geprüft, ob dieser Wert kleiner als der niedrigere Wert Protection Heating ist. Ist das der Fall, wird der Wert Economy Heating dem Wert Protection Heating gleichgesetzt. Es folgt dann die Prüfung, ob der Wert Precomfort Heating kleiner ist als Economy Heating und so weiter bis zur Prüfung, ob Protection Cooling kleiner ist als Economy Cooling. Sind ein oder mehrere Werte korrigiert worden, so wird dies am Ausgang *bValCorr* mit einem TRUE-Signal angezeigt.

VAR_INPUT

```
rRmTAdj   : REAL;
stSp      : ST_BA_SpRmT;
```

rRmTAdj: Raumtemperatur-Verschiebungswert.

stSp: Eingabe-Struktur der Sollwerte (siehe ST_BA_SpRmT [▶ 701]).

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: Autokorrektur der Werte wurde durchgeführt, siehe oben.

rPrPrtcHtg: Resultierender Sollwert Protection Heating.

rPrEcoHtg: Resultierender Sollwert Economy Heating.

rPrPreCmfHtg: Resultierender Sollwert PreComfort Heating.

rPrCmfHtg: Resultierender Sollwert Comfort Heating.

rPrCmfCol: Resultierender Sollwert Comfort Cooling.

rPrPreCmfCol: Resultierender Sollwert PreComfort Cooling.

rPrEcoCol: Resultierender Sollwert Economy Cooling.

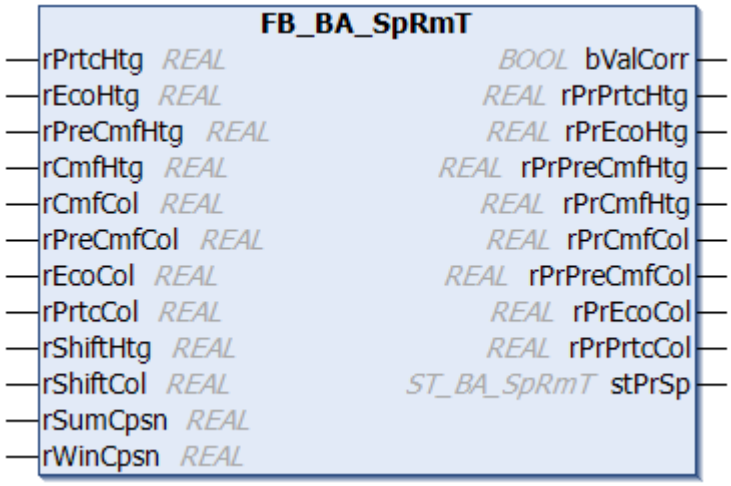
rPrPrtcCol: Resultierender Sollwert Protection Cooling.

stPrSp: Zusammengefasste Ausgabe der resultierenden Werte in einer Struktur (siehe ST_BA_SpRmT [▶ 701]).

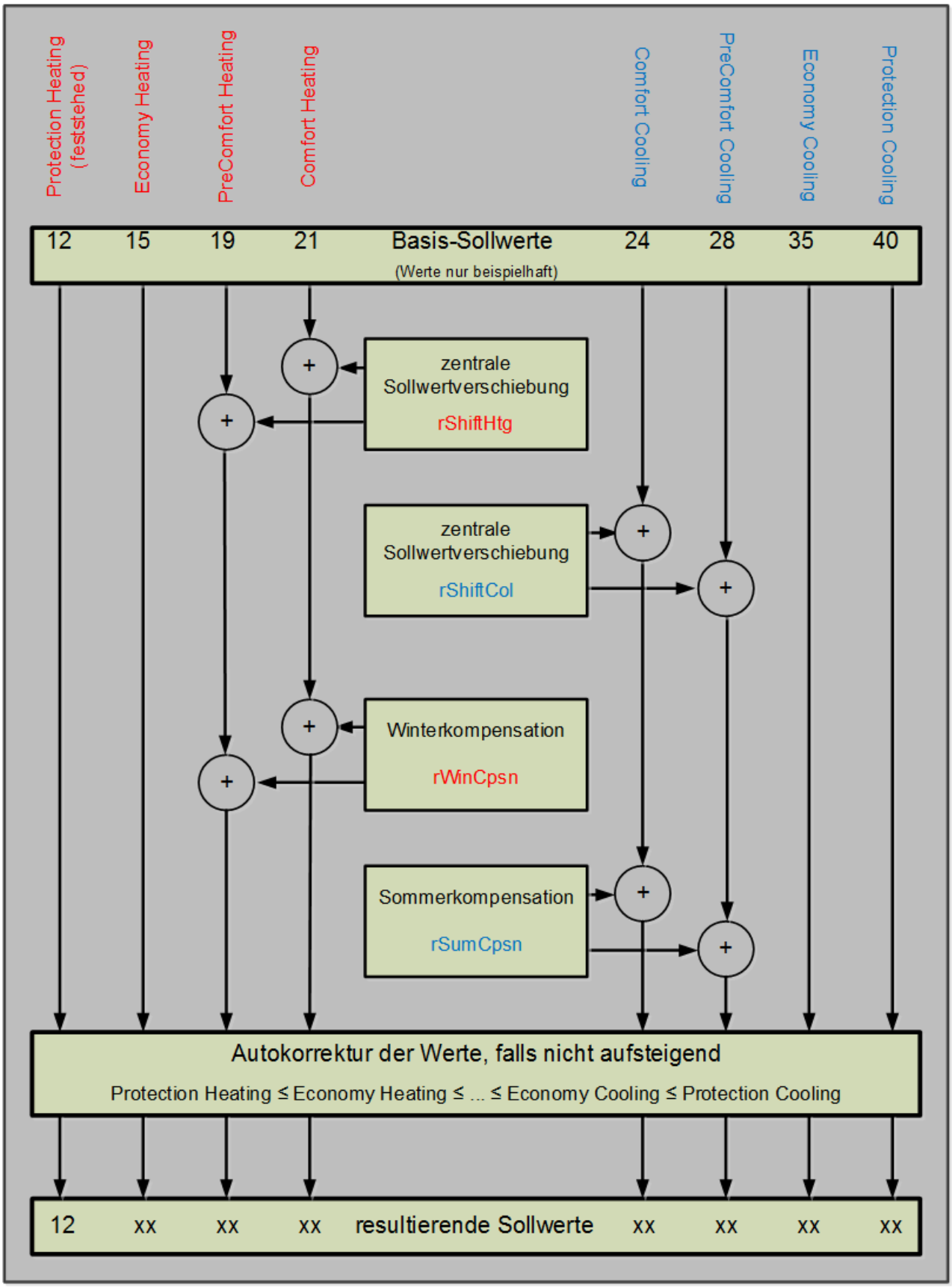
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.1.3 FB_BA_SpRmT



Der Funktionsbaustein *FB_BA_SpRmT* weist den Energieniveaus Protection, Economy, PreComfort und Comfort jeweils einen Sollwert für den Kühl- und Heizbetrieb zu.
Die folgende Grafik beschreibt das Verhalten des Funktionsbausteins, wobei die eingetragenen Werte als Beispiel zu sehen sind:



Die Comfort- und PreComfort-Werte des Heizbetriebes werden mit dem Parameter *rShiftHtg* als zentrale Sollwertverschiebung beaufschlagt. Unabhängig davon wird noch Winterkompensation *rWinCpsn* hinzuaddiert.
Für den Kühlbetrieb gilt analog: Die Comfort- und PreComfort-Werte werden mit dem Parameter *rShiftCol und* unabhängig davon mit dem Wert der Sommerkompensation *rSumCpsn* beaufschlagt.

Auto-Korrektur

Die Sollwert-Verschiebung ist dazu gedacht, die Werte in kleinem Rahmen zu beeinflussen. Natürlich können beliebige Eingangswerte eingegeben werden. Eine Heizungsanlage wird jedoch nur dann sinnvoll funktionieren, wenn die Sollwerte in folgender Reihenfolge aufsteigende Werte besitzen:

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

Die Auto-Korrektur arbeitet nach folgendem Prinzip: Beginnend mit dem Wert Economy Heating wird geprüft, ob dieser Wert kleiner als der niedrigere Wert Protection Heating ist. Ist das der Fall, wird der Wert Economy Heating dem Wert Protection Heating gleichgesetzt. Es folgt dann die Prüfung, ob der Wert Precomfort Heating kleiner ist als Economy Heating und so weiter bis zur Prüfung, ob Protection Cooling kleiner ist als Economy Cooling. Sind ein oder mehrere Werte korrigiert worden, so wird dies am Ausgang *bValCorr* mit einem TRUE-Signal angezeigt.

VAR_INPUT

```
rSumCpsn   : REAL;
rWrWinCpsn : REAL;
```

rSumCpsn: Wert Sommerkompensation

rWinCpsn: Wert Winterkompensation

VAR_OUTPUT

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

bValCorr: Autokorrektur: mindestens einer der resultierenden Sollwerte wurde so angepasst, dass die Werte weiterhin monoton steigend sind.

rPrPrtcHtg: Resultierender Sollwert Protection Heating.

rPrEcoHtg: Resultierender Sollwert Economy Heating.

rPrPreCmfHtg: Resultierender Sollwert PreComfort Heating.

rPrCmfHtg: Resultierender Sollwert Comfort Heating.

rPrCmfCol: Resultierender Sollwert Comfort Cooling.

rPrPreCmfCol: Resultierender Sollwert PreComfort Cooling.

rPrEcoCol: Resultierender Sollwert Economy Cooling.

rPrPrtcCol: Resultierender Sollwert Protection Cooling.

stPrSp: Zusammengefasste Ausgabe der resultierenden Werte in einer Struktur (siehe ST_BA_SpRmT [▶ 701]).

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: Sollwertverschiebung Kühlen.

rShiftHtg: Sollwertverschiebung Heizen.

rPrtcCol: Basis-Sollwert Protection Cooling.

rEcoCol: Basis-Sollwert Economy Cooling.

rPreCmfCol: Basis-Sollwert PreComfort Cooling.

rCmfCol: Basis-Sollwert Comfort Cooling.

rCmfHtg: Basis-Sollwert Comfort Heating.

rPreCmfHtg: Basis-Sollwert PreComfort Heating.

rEcoHtg: Basis-Sollwert Economy Heating.

rPrtcHtg: Basis-Sollwert Protection Heating.

Voraussetzungen

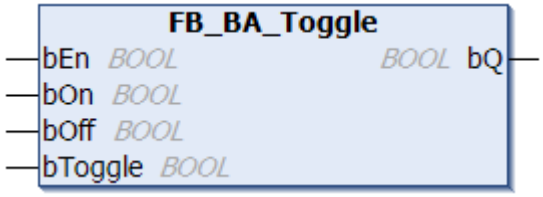
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.2 Beleuchtung

Funktionsbausteine

Name	Beschreibung
FB_BA_Toggle [▶ 555]	Schalten von Leuchten.

6.1.2.3.2.1.2.2.1 FB_BA_Toggle



Der Funktionsbaustein dient zum Ein- und Ausschalten eines Aktors.

Der Eingang *bEn* dient der allgemeinen Freigabe des Bausteins.
Durch eine positive Flanke an dem Eingang *bOn* wird der Ausgang *bQ* gesetzt. Zurückgesetzt wird der Ausgang durch eine positive Flanke an dem Eingang *bOff*. Wird an *bToggle* eine positive Flanke angelegt, so wird der Ausgang negiert; also von An nach Aus, bzw. von Aus nach An umgeschaltet.

VAR_INPUT

```
bEn     : BOOL;
bOn     : BOOL;
bOff    : BOOL;
bToggle : BOOL;
```

bEn: Freigabe Baustein.

bOn: Schaltet den Ausgang ein.

bOff: Schaltet den Ausgang aus.

bToggle: Negiert den aktuellen Zustand des Ausgangs.

VAR_OUTPUT

```
bQ    : BOOL;
```

bQ: Steuerausgang.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3 Verschattung

Übersicht Verschattungskorrektur [▶ 558]

Verschattungskorrektur: Grundlagen und Definitionen [▶ 558]

Übersicht Sonnenschutzautomatik [▶ 566]

Sonnenschutz: Grundlagen und Definitionen [▶ 567]

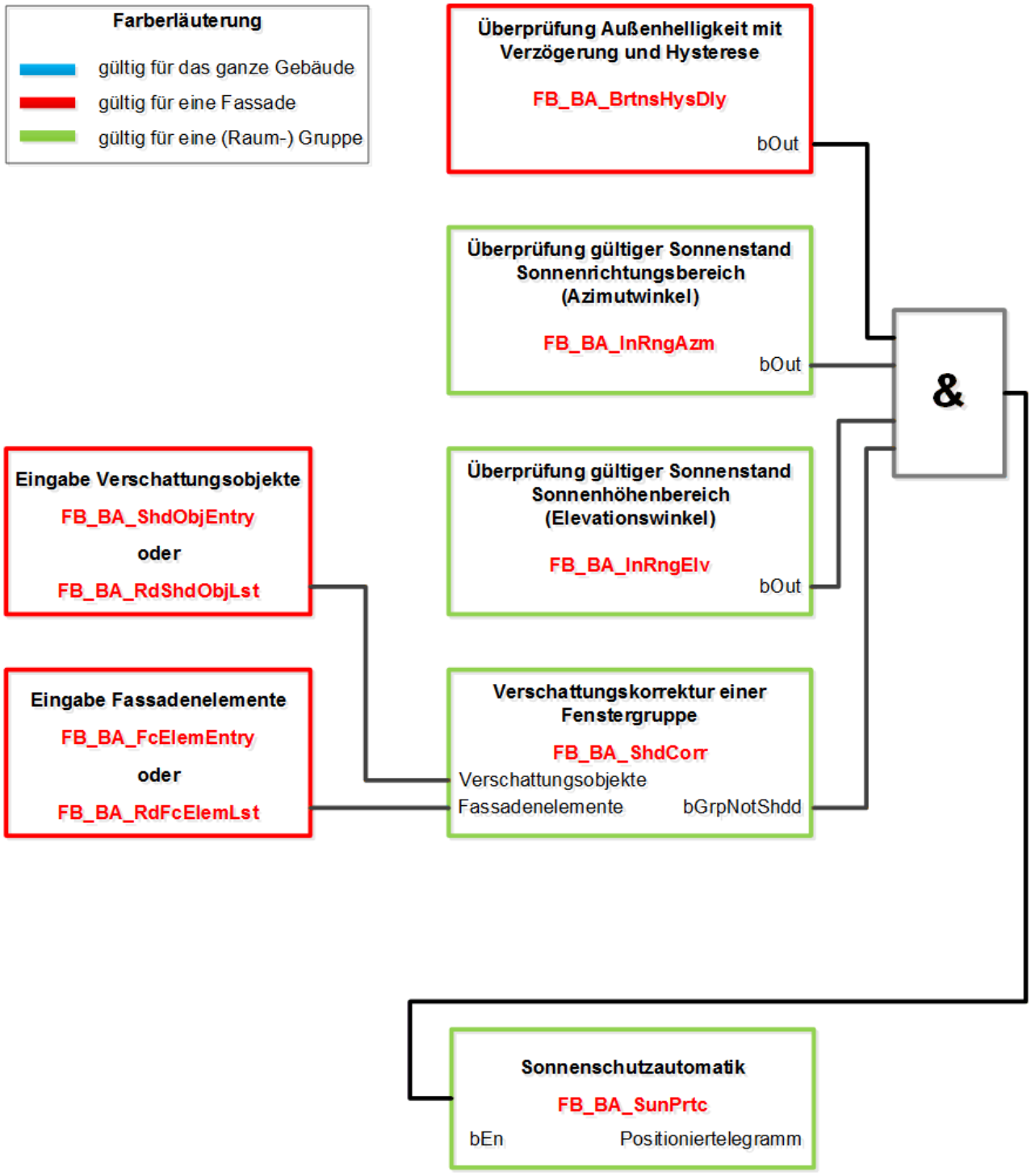
Liste der Verschattungselemente [▶ 572]

Liste der Fassadenelemente [▶ 572]

Funktionsbausteine

Name	Beschreibung
FB_BA_BldPosEntry [▶ 572]	Sonnenschutzfunktion: Eingabe von Jalousiepositionen.
FB_BA_CalcSunPos [▶ 574]	Berechnung Sonnenstand
FB_BA_FcdElemEntry [▶ 576]	Verschattungskorrektur: Eingabe von Fassadenelementen per Baustein.
FB_BA_InRngAzm [▶ 581]	Überprüfung gültiger Sonnenstand Sonnenrichtungsbereich (Azimuthwinkel)
FB_BA_InRngElv [▶ 583]	Überprüfung gültiger Sonnenstand Sonnenhöhenbereich (Elevationswinkel)
FB_BA_RdFcdElemLst [▶ 586]	Verschattungskorrektur: Eingabe von Fassadenelementen per Datenliste (csv).
FB_BA_RdShdObjLst [▶ 590]	Verschattungskorrektur: Eingabe von Verschattungsobjekten per Datenliste (csv).
FB_BA_RolBldActr [▶ 594]	Rollladen-Aktor
FB_BA_ShdCorr [▶ 596]	Verschattungskorrektur-Baustein
FB_BA_ShdObjEntry [▶ 599]	Verschattungskorrektur: Eingabe von Verschattungsobjekten per Baustein.
FB_BA_SunBldActr [▶ 603]	Jalousie-Aktor
FB_BA_SunBldEvt [▶ 608]	Ausgabe eines vorgegebenen prozentualen Jalousieposition und –stellung
FB_BA_SunBldIcePrtc [▶ 609]	Vereisungsschutz
FB_BA_SunBldPosDly [▶ 610]	Einschaltverzögerung von Jalousien/Jalousiegruppen
FB_BA_SunBldPrioSwi4 [▶ 612]	Prioritätssteuerung, 4 Eingänge
FB_BA_SunBldPrioSwi8 [▶ 613]	Prioritätssteuerung, 8 Eingänge
FB_BA_SunBldScn [▶ 614]	Handbedienung mit Szenenanwahl und -programmierung
FB_BA_SunBldSwi [▶ 616]	Handbedienung
FB_BA_SunBldTwiLgtAuto [▶ 618]	Dämmerungsautomatik
FB_BA_SunBldWndPrtc [▶ 619]	Schutz gegen Schaden durch Wind
FB_BA_SunPrtc [▶ 621]	Sonnenschutzfunktion, siehe Übersicht Sonnenschutzautomatik (Verschattungskorrektur)

6.1.2.3.2.1.2.3.1 Übersicht Verschattungskorrektur



6.1.2.3.2.1.2.3.2 Verschattungskorrektur: Grundlagen und Definitionen

Die Verschattungskorrektur ist in Verbindung mit der Sonnenautomatik oder Lamellennachführung nutzbar. Die Funktion prüft, ob ein Fenster oder eine Gruppe von Fenstern, die z. B. einem Raum zugeordnet sind, temporär durch umliegende Bebauung oder eigene Gebäudeteile verschattet werden. Für Fenster, welche im Schatten umliegender Gebäude oder Bäume stehen ist ein Sonnenschutz nicht notwendig, unter Umständen sogar störend. Die Verschattungskorrektur ermittelt anhand von eingetragenen Daten der Fassade und ihrer Umgebung, welche Teile der Fassade verschattet werden. Damit ist es dann möglich, für einzelne Fenster oder Fenstergruppen zu entscheiden, ob der Sonnenschutz aktiv sein soll.
Neben dem aktuellen Sonnenstand hängt die Verschattung der einzelnen Fenster von drei Dingen ab:

- der Ausrichtung der Fassade

- der Lage der Fenster
- der Positionierung der Verschattungsobjekte

Die folgenden Abbildungen sollen diese Zusammenhänge erläutern und die einzutragenden Parameter vorstellen.

Ausrichtung der Fassade

Betrachtung von oben

Für die reine Betrachtung des Schattenwurfes auf die Fassade ist letztendlich ein zweidimensionales Koordinatensystem erforderlich, daher wurden die x- und y-Achse auf die Fassade gelegt. Der Nullpunkt liegt dabei im Fußpunkt links unten, so als würde die Fassade von vorne betrachtet. Zur Bemessung der verschattenden Objekte kommt dann noch die Z-Komponente hinzu. Deren Achse weist von der Fassade weg und hat denselben Nullpunkt, wie die x- und y-Achse.

Der horizontale Sonnenstand (Azimutwinkel) ist auf der Nordhalbkugel per Definition von der Nordrichtung aus bemessen. Die Fassadenausrichtung richtet sich ebenfalls nach der Nordrichtung, wobei die Blickrichtung aus einem Fenster der Fassade gilt:

Blickrichtung	Fassadenausrichtung
Nord	β=0°
Ost	β=90°
Süd	β=180°
West	β=270°

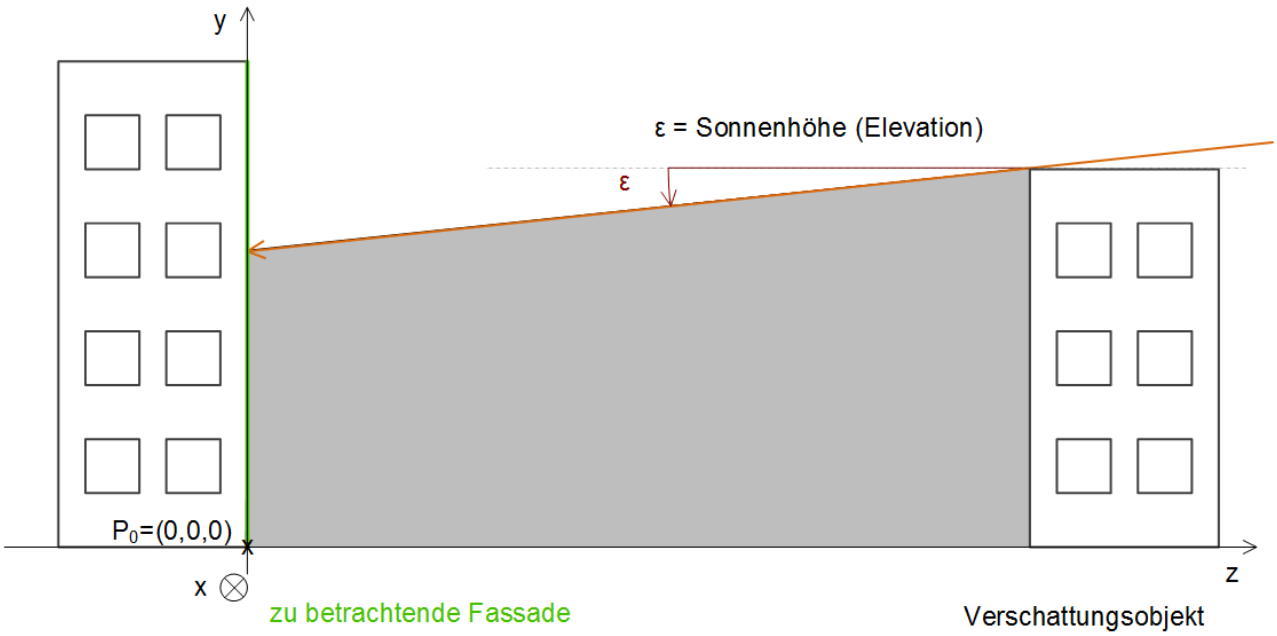
Auf der Südhalbkugel ist der Sonnenverlauf anders herum: Sie geht zwar auch im Osten auf, hat ihren Mittagsstand jedoch im Norden. Die Fassadenausrichtung wird diesem Verlauf angepasst:

Blickrichtung	Fassadenausrichtung
Süd	β=0°
Ost	β=90°
Nord	β=180°
West	β=270°

Die weiteren Ausführungen beziehen sich jedoch der Einfachheit halber nur auf die Anwendung auf der Nordhalbkugel. Die Berechnungen auf der Südhalbkugel gestalten sich analog und werden bei der Parametrierung des Bausteines FB_BA_ShdCorr [▶ 596] (Verschattungskorrektur) durch einen booleschen Eingang, *bSouth*, aktiviert

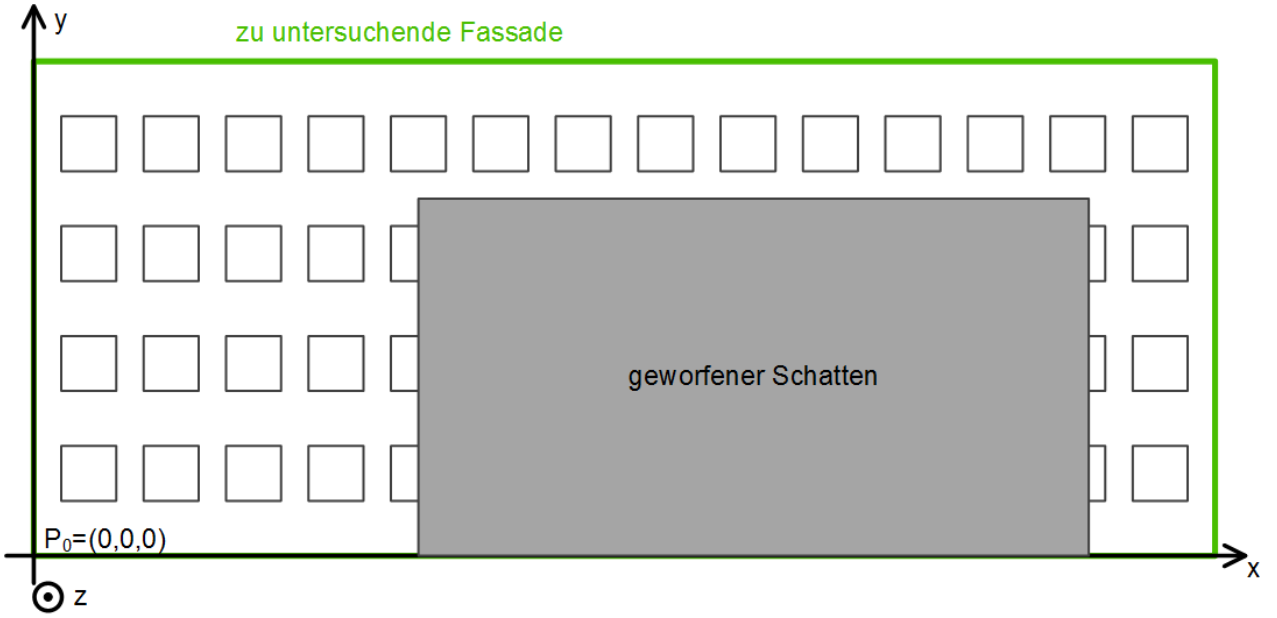
Die beiden folgenden Abbildungen sollen die Lage des Ursprungspunktes P_0 sowie die Ausrichtung des Koordinatensystems weiter verdeutlichen:

Betrachtung von der Seite



Anhand dieser Abbildung lässt sich auch der Elevationswinkel (Sonnenhöhe) darstellen: per Definition ist dieser bei Sonnenaufgang 0° (horizontaler Lichteinfall) und kann maximal 90° erreichen, dies jedoch nur an Orten innerhalb des nördlichen und südlichen Wendekreises.

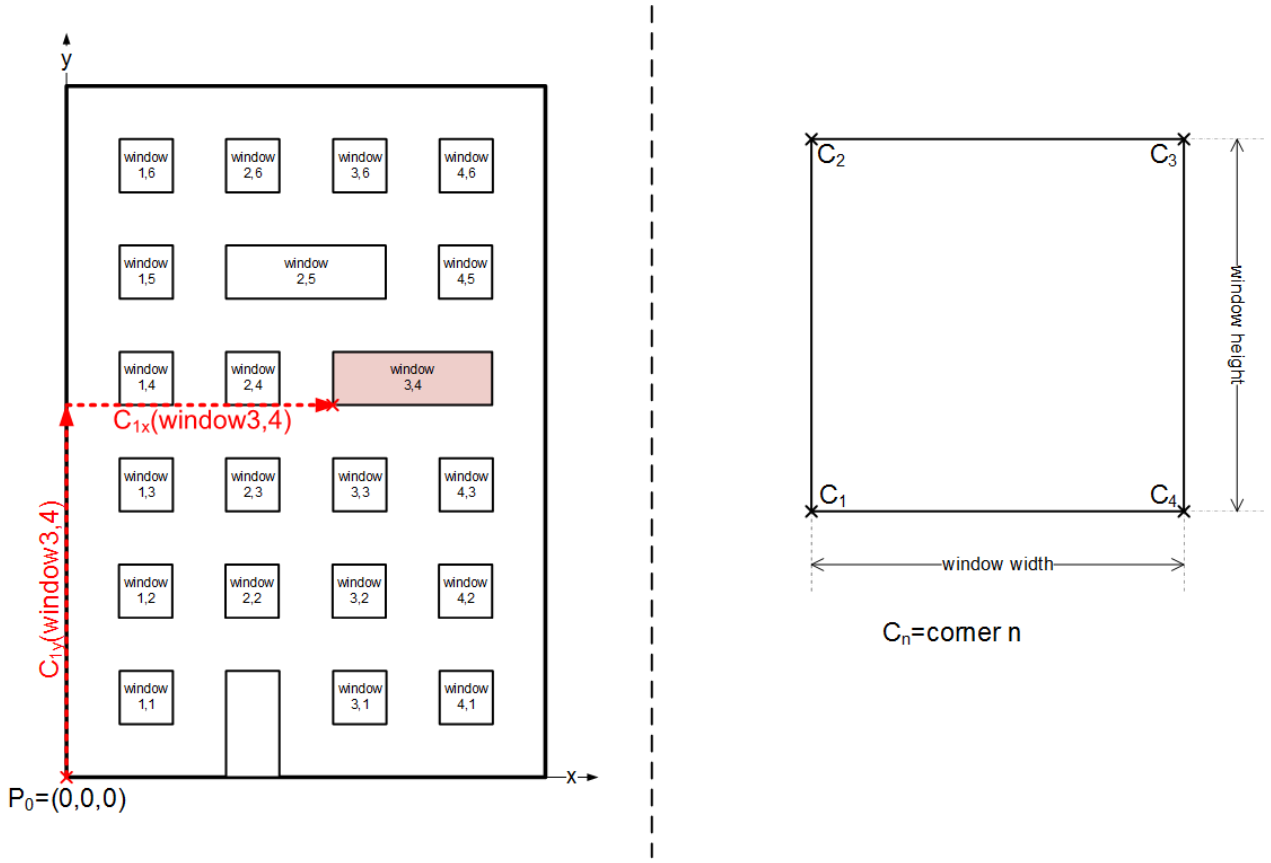
Betrachtung von vorne



Hier ist die Lage des Koordinatenursprungs, P_0, am linken unteren Fußpunkt der Fassade noch einmal besonders deutlich. Darüber hinaus ist die x-y-Ausrichtung dargestellt, die später für den Eintrag der Fensterelemente wichtig ist.

Lage der Fenster

Die Lage der Fenster wird durch die Angabe ihres linken unteren Eckpunktes in Bezug auf das Fassaden-Koordinatensystems definiert. Da ein Fenster plan auf der Fassade liegt, ist die Eingabe auf die x- und die y-Koordinate beschränkt.



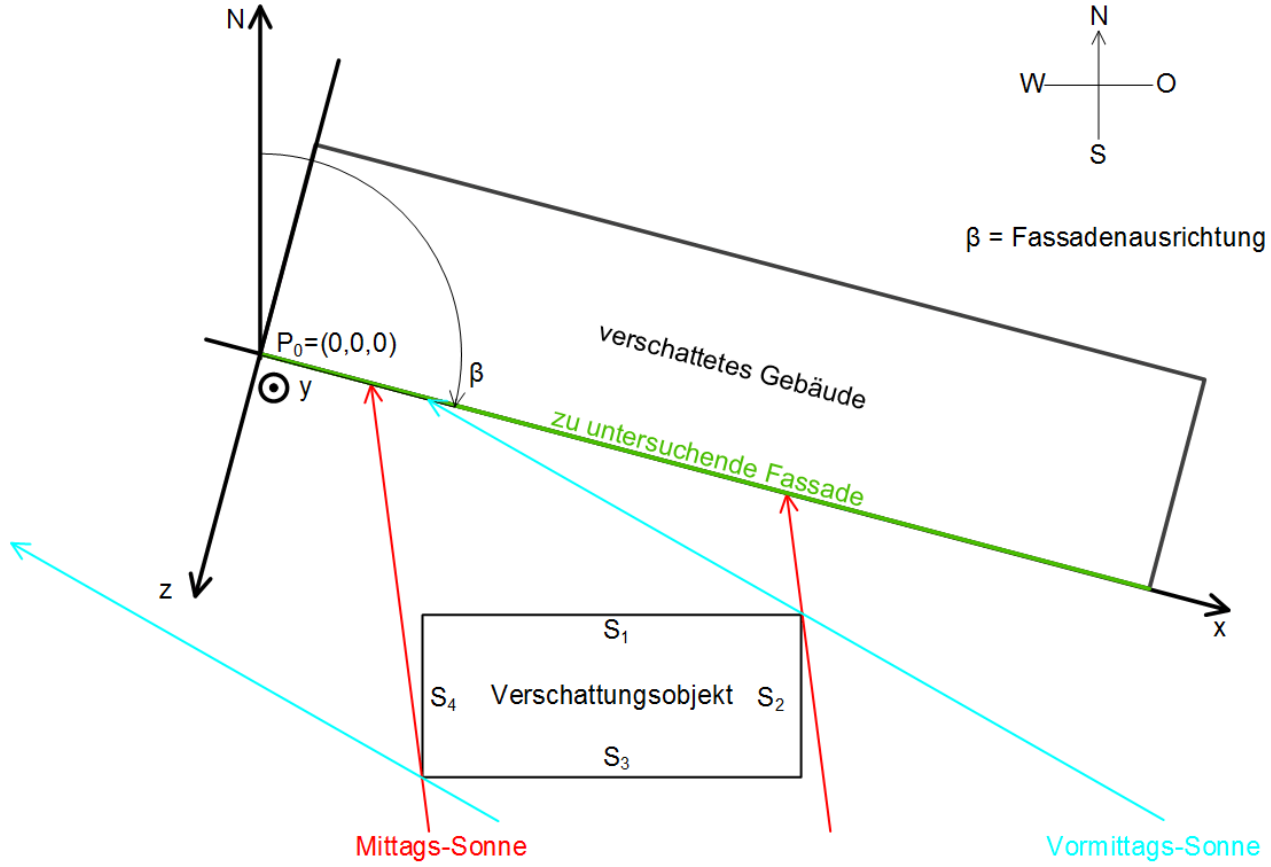
Zusätzlich sind die Fensterbreite und die Fensterhöhe anzugeben.

Aus den eingetragenen Werten wird intern die Lage jedes Fenstereckpunktes auf der Fassade ermittelt. Ein Fenster gilt dann als verschattet, wenn alle Eckpunkte im Schatten liegen.

Positionierung der Verschattungsobjekte

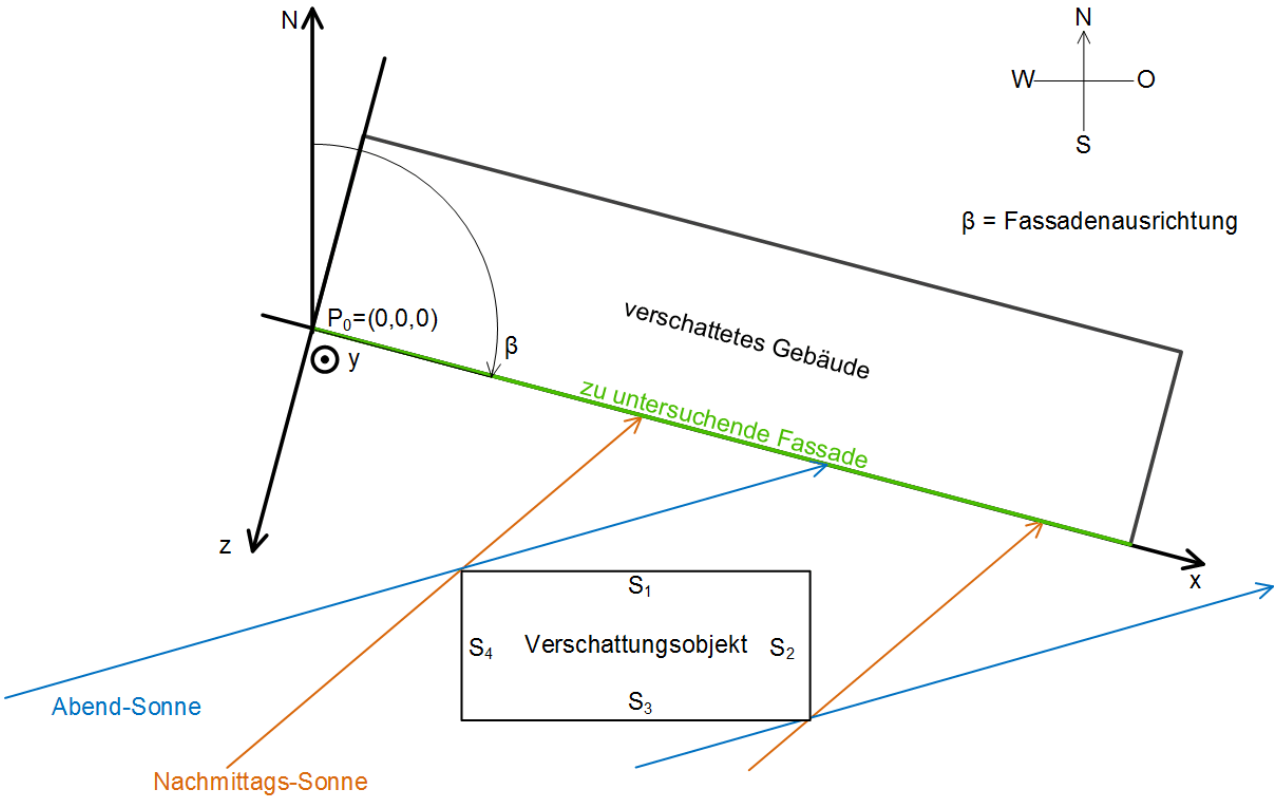
Bei der Beschreibung der Verschattungsobjekte wird zwischen eckigen Objekten (Gebäude, Pfeiler) und Objekten, die annähernd kugelförmig sind (z.B. Bäume), unterschieden. Eckige Objekte lassen sich ihrem Schattenwurf nach in viereckige schattenwerfende Fassaden unterteilen, wobei überlegt werden muss, welche über den Tag hinweg den Hauptschatten werfen:

Morgens/Mittags

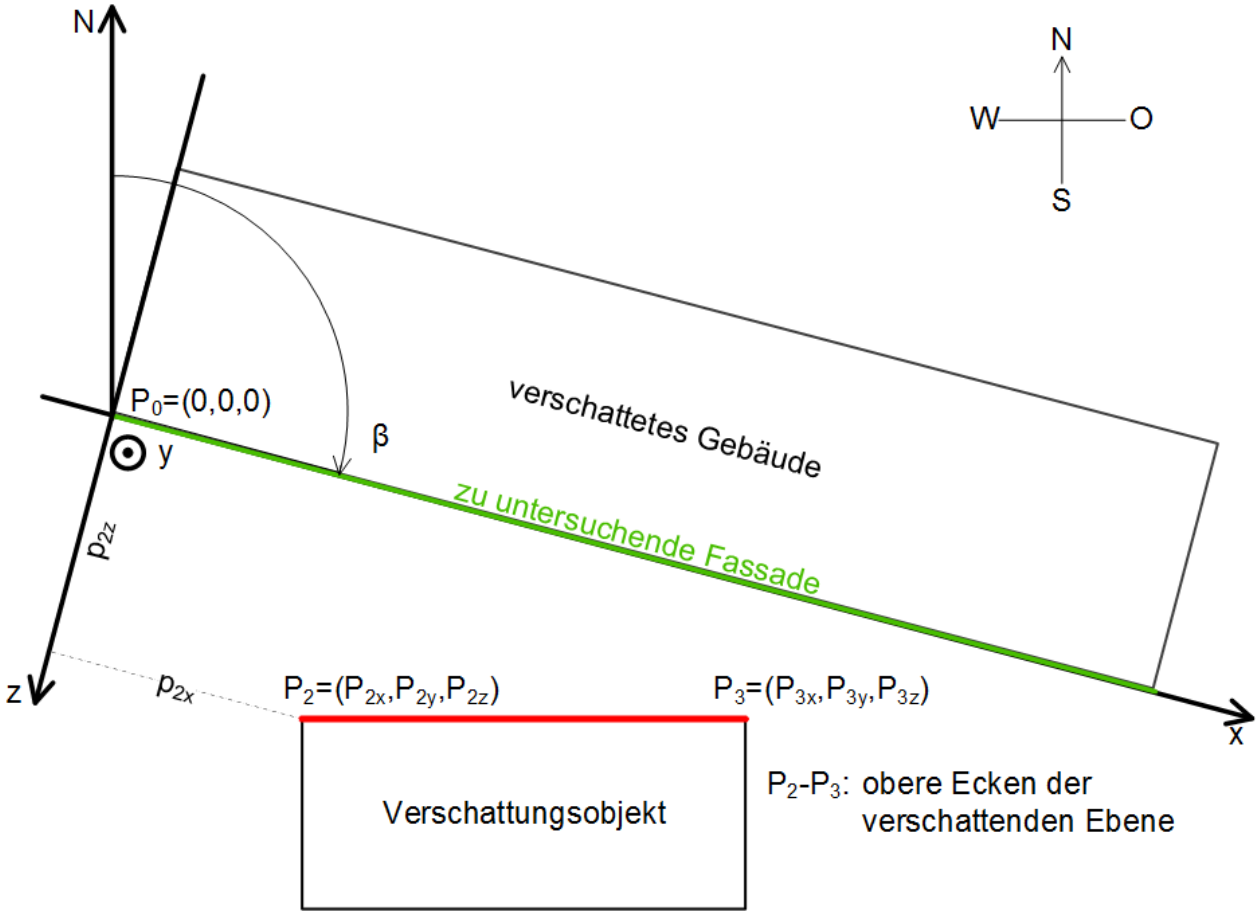


Morgens und mittags ist der geworfene Schatten hauptsächlich durch die Seiten S_1 und S_4 gegeben, eine Betrachtung von S_2 und S_3, sollten sie nicht höher sein, ist nicht notwenig.

Nachmittags/Abends



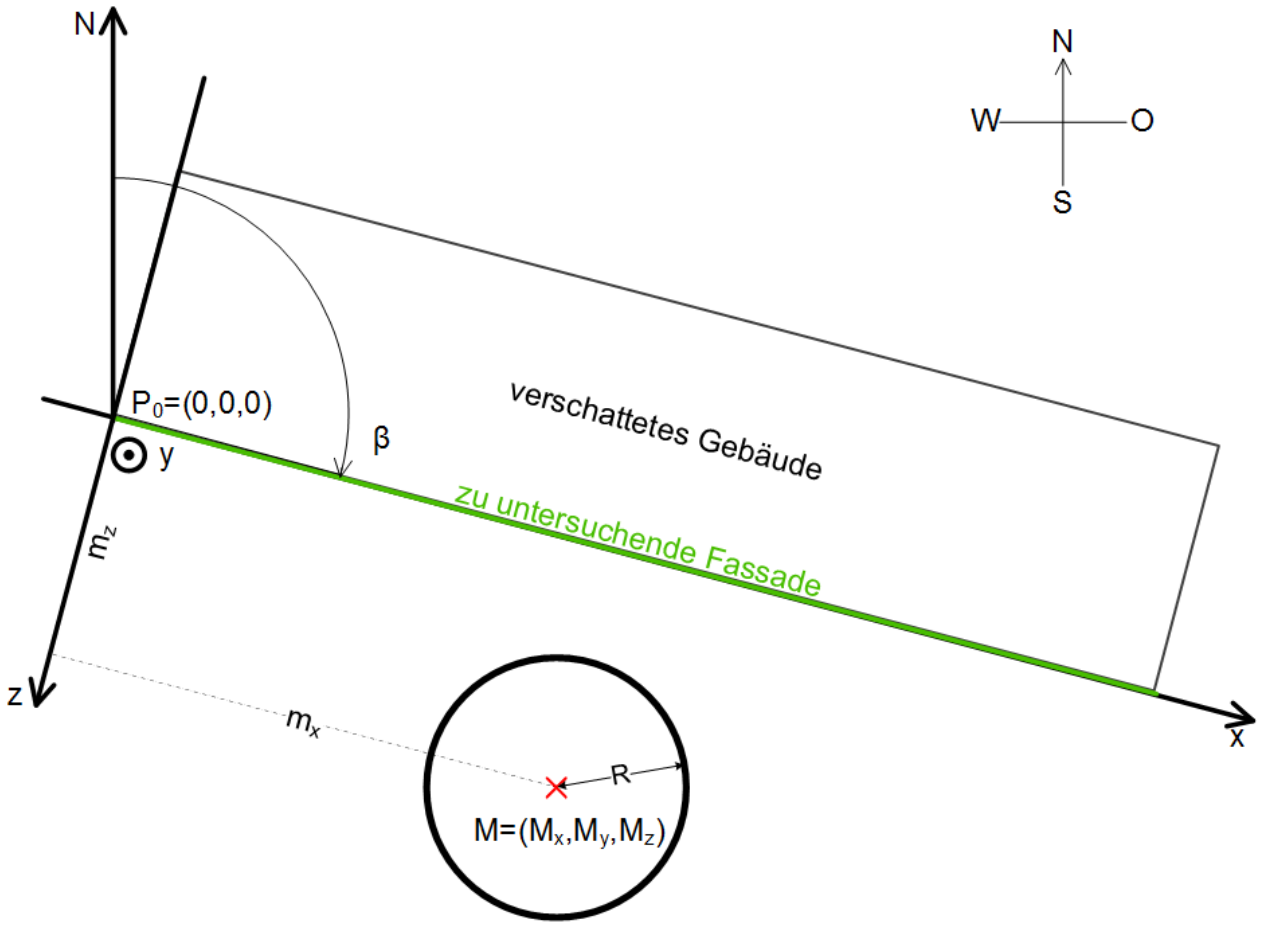
Am Nachmittag und am Abend lässt sich der Summen-Schatten allein durch die Betrachtung von S_1 und S_2 ermitteln. Es reicht also in diesem Fall S_1, S_2 und S_4 als Schattenwerfer anzugeben. Die Eingabe erfolgt dabei anhand der vier Eckpunkte bzw. deren Koordinaten in Bezug auf den Fassaden-Nullpunkt:



In dieser Skizze sind wegen der Draufsicht nur die oberen Punkte, P_2 und P_3 dargestellt. Der untere Punkt P_1 liegt unter P_2 und P_4 unter P_3.

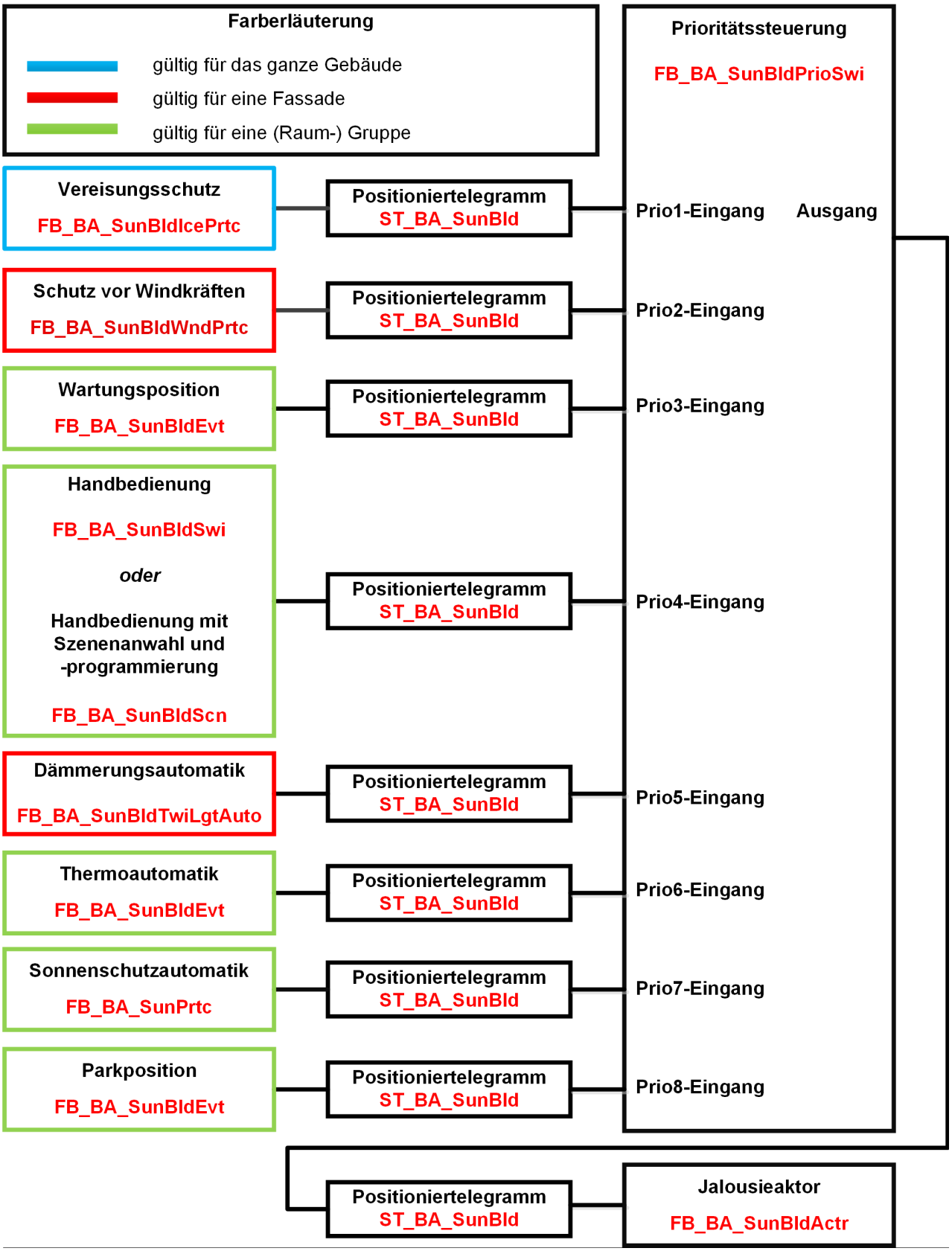
Die Eingabe von verschattenden Kugelelementen erfolgt durch die Eingabe des Kugelmittelpunktes und des Radius:

Kugelelemente



Eine "Einteilung" des Kugelelementes, wie beim eckigen Gebäude, ist nicht notwendig, da der Schattenwurf einer Kugel sich nur in seiner Richtung, nicht aber in seiner Größe ändert.

6.1.2.3.2.1.2.3.3 Übersicht Sonnenschutzautomatik



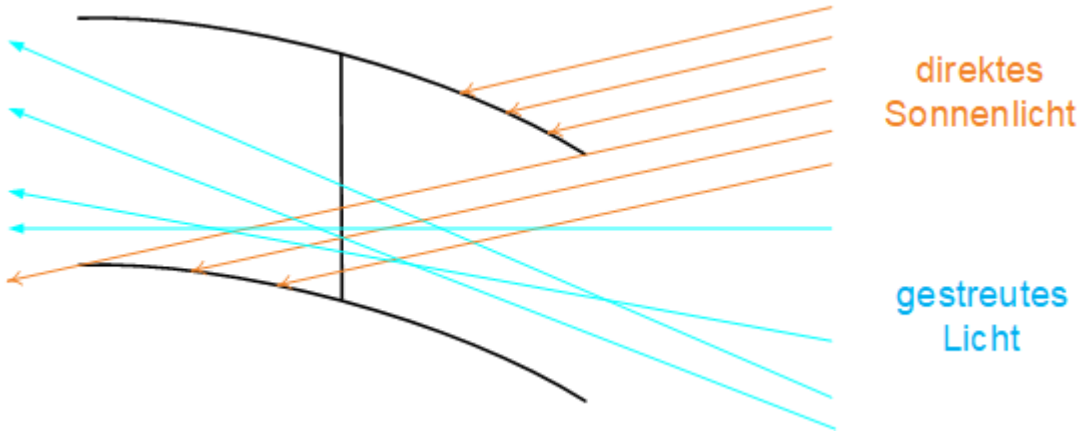
6.1.2.3.2.1.2.3.4 Sonnenschutz: Grundlagen und Definitionen

Direkter Einfall von Tageslicht wird von in Räumen befindlichen Personen als störend angesehen. Auf der anderen Seite jedoch empfindet der Mensch das natürliche Licht als angenehmer im Vergleich zum künstlichen Licht. Zwei Möglichkeiten des Blendschutzes sollen hier vorgestellt werden:

- Lamellennachführung
- Höhenverstellung

Lamellennachführung

Eine Jalousie mit Lamellen, welche sich nachführen lassen, bietet hier die Möglichkeit eines intelligenten Sonnenschutzes. Dabei wird die Stellung der Lamellen zyklisch dem aktuellen Sonnenstand angepasst, so dass kein direktes Tageslicht durch die Jalousien fällt, jedoch möglichst viel diffuses Tageslicht genutzt werden kann.



Die Abbildung zeigt, dass gestreutes Licht von unten noch einfallen kann, während vom direkten Tageslicht gerade nichts mehr, bzw. theoretisch nur noch ein Strahl hindurchtritt. Zur Berechnung des Lamellenwinkels sind folgende Parameter notwendig:

- die aktuelle Sonnenhöhe (Elevationswinkel)
- der Sonnenstand, d.h. der Azimut Winkel
- die Fassadenausrichtung
- die Lamellenbreite
- der Lamellenabstand

effektiver Elevationswinkel

Wird die Jalousie im Schnitt wie oben betrachtet, so hängt der Lichteinfallswinkel nicht allein von der Sonnenhöhe (Elevation) ab, sondern auch von der Sonnenrichtung:

- Sind Fassadenausrichtung und Sonnenstand (Azimut) gleich, trifft die Sonne also direkt auf die Fassade, so ist der effektive Lichteinfallswinkel gleich dem aktuellen Elevationswinkel.
- Fällt die Sonne jedoch von der Sonnenrichtung aus gesehen schräg auf die Fassade, so ist bei gleichem Elevationswinkel der effektive Winkel größer.

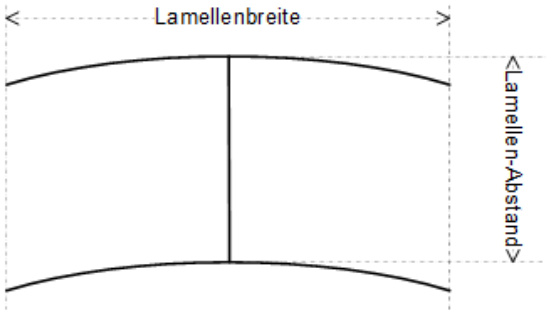
Dieser Zusammenhang kann leicht mit einem rechtwinkligen Zeichendreieck, welches hochkant auf dem Tisch steht, verdeutlicht werden: Direkt von der Seite ist ein Dreieck mit zwei 45°-Winkeln und einem 90° Winkel zu sehen. Wird das Dreieck gedreht, so wird die auf dem Tisch liegende Seite scheinbar kürzer und die beiden ehemaligen 45°-Winkel ändern sich. Das Dreieck scheint steiler zu werden.

Im Folgenden soll daher vom „effektiven Elevationswinkel“ gesprochen werden, der Teil des Lichtes, welcher direkt auf die Jalousie trifft.

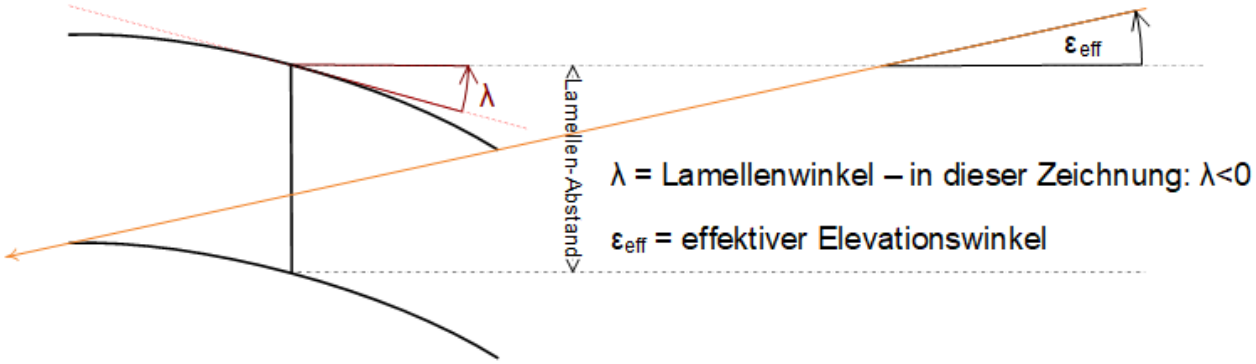
Die folgenden drei Bilder zeigen den Zusammenhang zwischen effektivem Elevationswinkel und den Jalousiemaßen und wie sich der resultierende Lamellenwinkel λ im Laufe des Tages ändert:

Lamellenwinkel

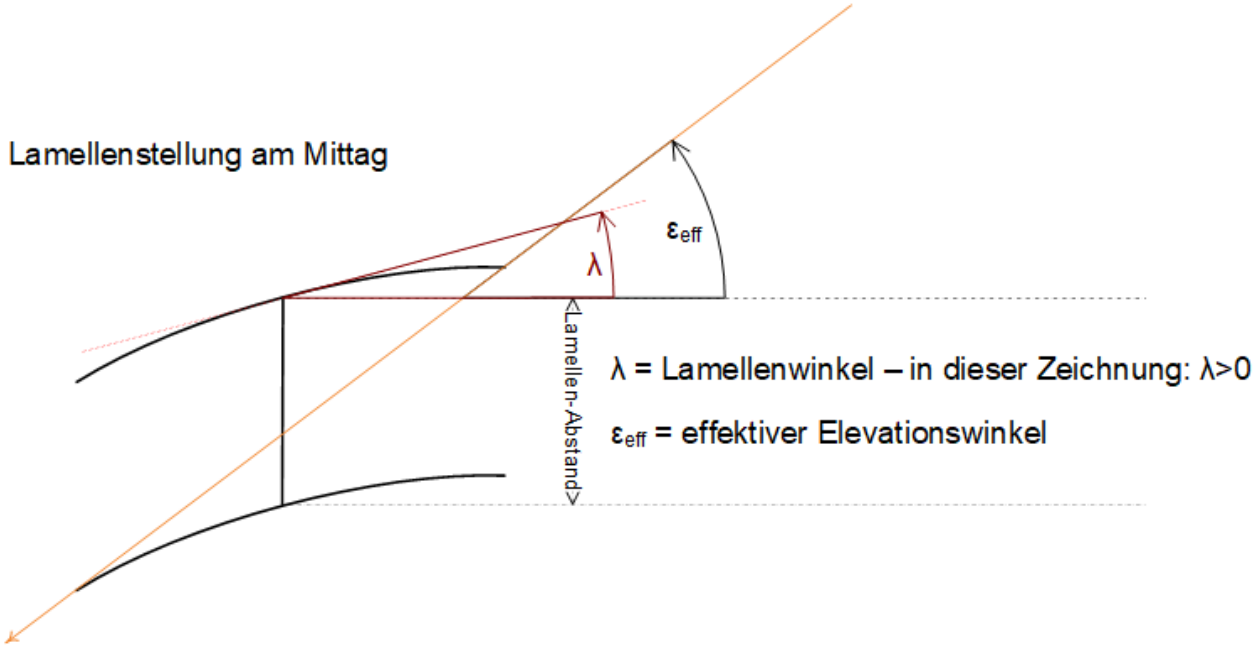
Lamellen bei einen Winkel von λ=0



Lamellenstellung am Morgen und am Abend

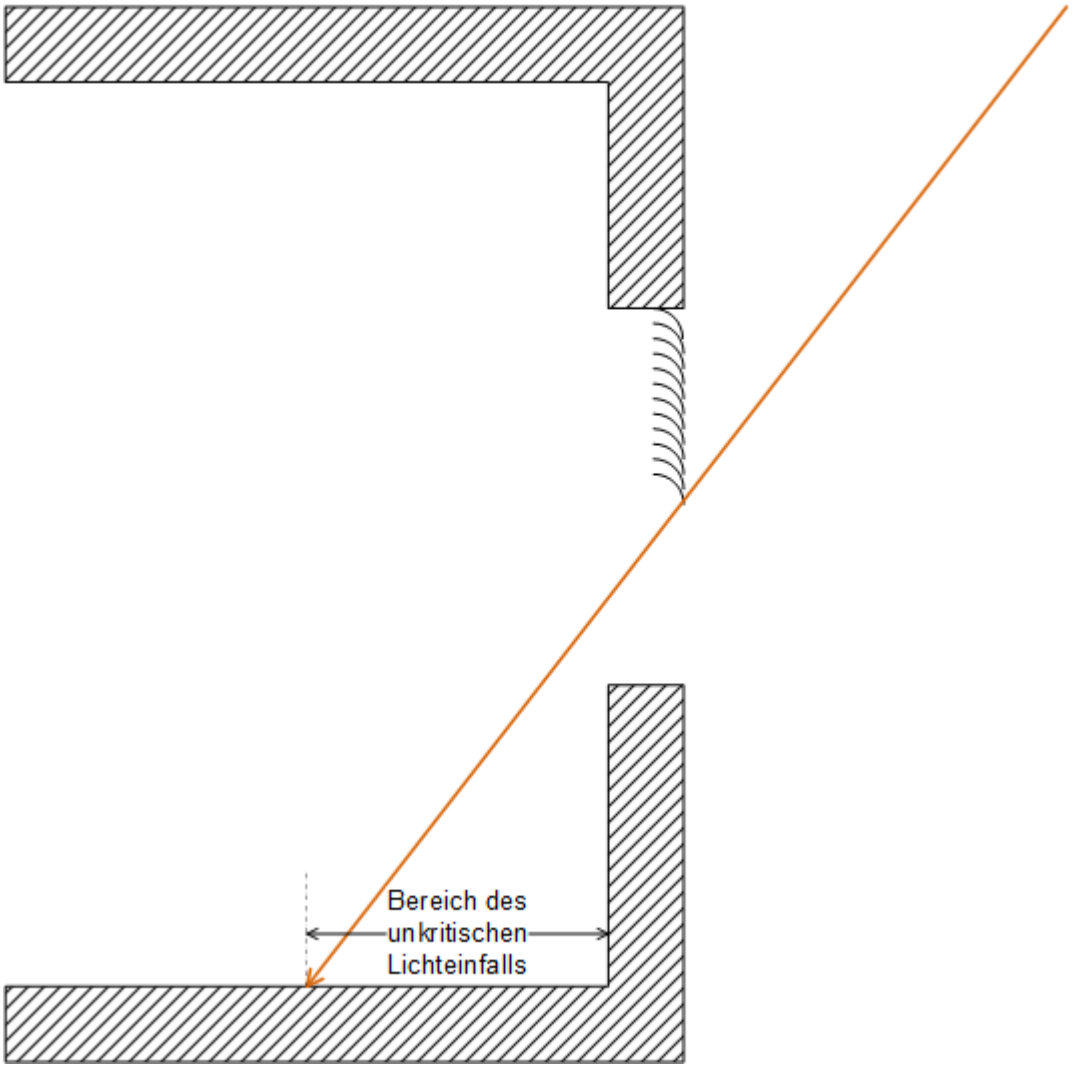


Lamellenstellung am Mittag



Höhenverstellung

Bei einem hohen Sonnenstand in der Mittagszeit dringen die direkten Sonnenstrahlen nicht in die volle Tiefe des Raumes ein. Wenn direkte Sonnenstrahlen im Bereich der Fensterbrüstung als unkritisch betrachtet werden, kann die Höhe des Sonnenschutzes automatisch so angepasst werden, dass die Sonnenstrahlen immer nur bis zu einer unkritischen Tiefe in den Raum eindringen.

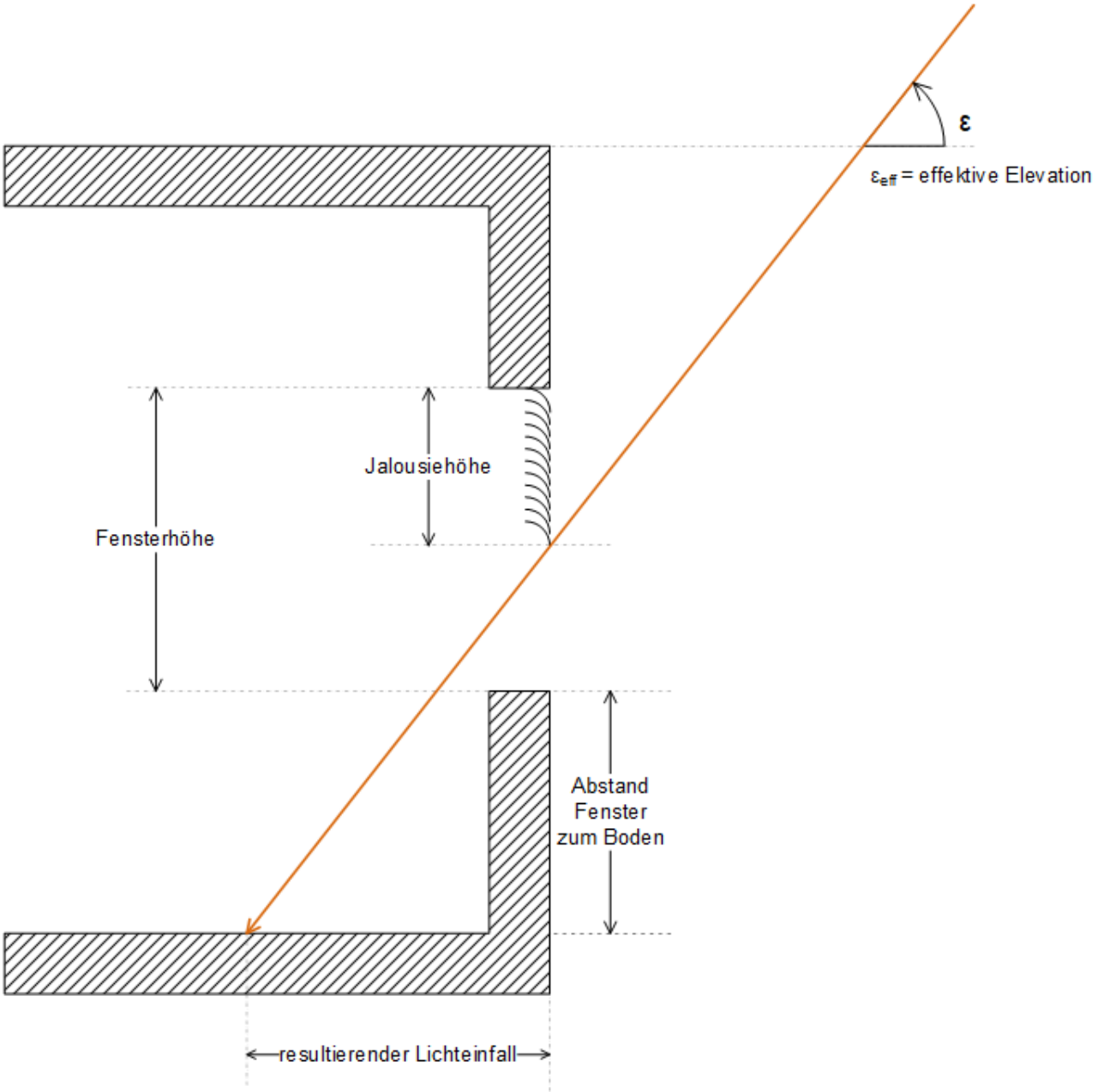


Um zu jedem Zeitpunkt die entsprechende Jalousiehöhe berechnen zu können, die sicherstellt, dass die Sonneneinstrahlung einen gewissen Wert nicht überschreitet, sind folgende Werte nötig.

Zur Berechnung der jeweiligen Jalousiehöhe erforderlich:

- Sonnenhöhe (Elevation)
- Fensterhöhe
- Abstand Fenster zum Boden

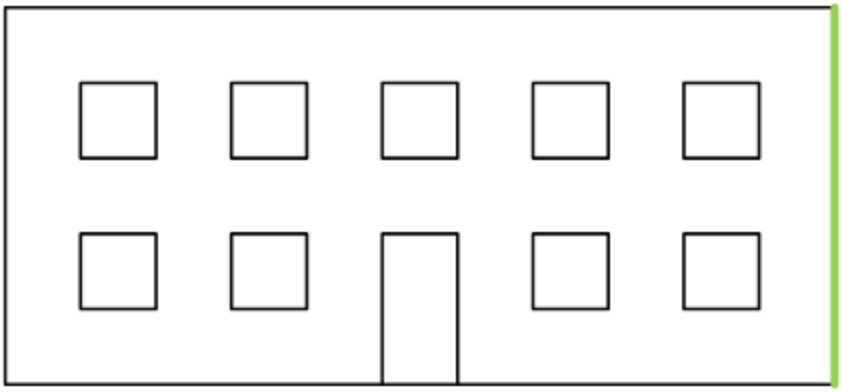
Die folgende Abbildung veranschaulicht, wo diese Parameter einzuordnen sind:



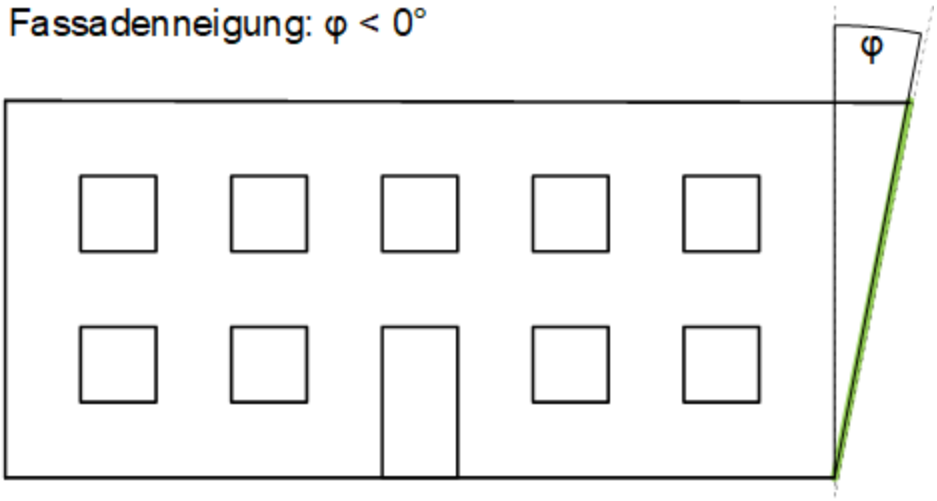
Einfluss der Fassadenneigung

Bei den beiden beschriebenen Methoden des Sonnenschutzes wurde davon ausgegangen, dass die Fassade und damit die Fenster senkrecht zum Boden stehen. Bei einer geneigten Fassade jedoch ändert sich der Lichteinfall, so dass dieser Einfluss mit berücksichtigt wird. Die Fassadenneigung wird wie folgt definiert:

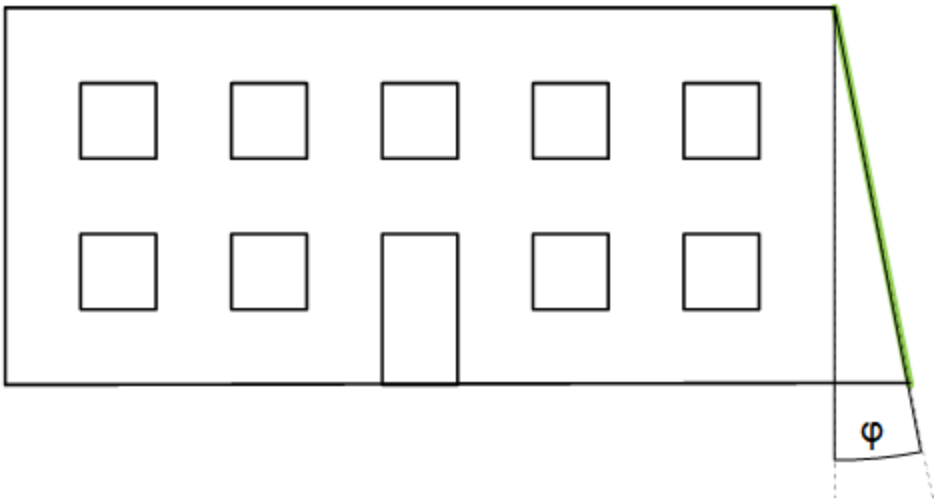
Fassadenneigung: φ = 0°



Fassadenneigung: φ < 0°



Fassadenneigung: φ > 0°



6.1.2.3.2.1.2.3.5 Liste der Verschattungselemente

Die Daten aller Verschattungsobjekte (Gebäudeteile, Bäume, etc.) pro Fassade werden innerhalb des Programms in einem Feld von Strukturelementen des Typs ST_BA_ShdObj [▶ 701] hinterlegt.

Die Verschattungskorrektur FB_BA_ShdCorr [▶ 596] liest die Informationen dieser Liste. Der Verwaltungsbaustein FB_BA_ShdObjEntry [▶ 599] jedoch liest und beschreibt sie als Ein-/Ausgangsvariable. Es empfiehlt sich daher, diese Liste global zu deklarieren:

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

Die Variable *gBA_cMaxShdObj* stellt dabei die Obergrenze der zur Verfügung stehenden Elemente dar und ist global als Konstante innerhalb der Programmbibliothek deklariert:

```
VAR_GLOBAL CONSTANT
     gBA_cMaxShdObj  : INT := 20;
END_VAR
```

6.1.2.3.2.1.2.3.6 Liste der Fassadenelemente

Die Daten aller Fenster (Fassadenelemente) pro Fassade werden innerhalb des Programms in einem Feld von Strukturelementen des Typs ST_BA_FcdElem [▶ 700] hinterlegt.

Der Verwaltungsbaustein FB_BA_FcdElemEntry [▶ 576] sowie die Verschattungskorrektur FB_BA_ShdCorr [▶ 596] lesen und beschreiben diese Liste (letzterer setzt die Verschattungsinformation) und greifen somit als Ein-/Ausgangsvariable auf dieses Feld zu.
Es empfiehlt sich daher, diese Liste global zu deklarieren:

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

Die Variablen *uiMaxColumnFcd* und *uiMaxRowFcd* definieren dabei die Obergrenzen der zur Verfügung stehenden Elemente und sind global als Konstante innerhalb der Programmbibliothek deklariert:

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

6.1.2.3.2.1.2.3.7 FB_BA_BldPosEntry



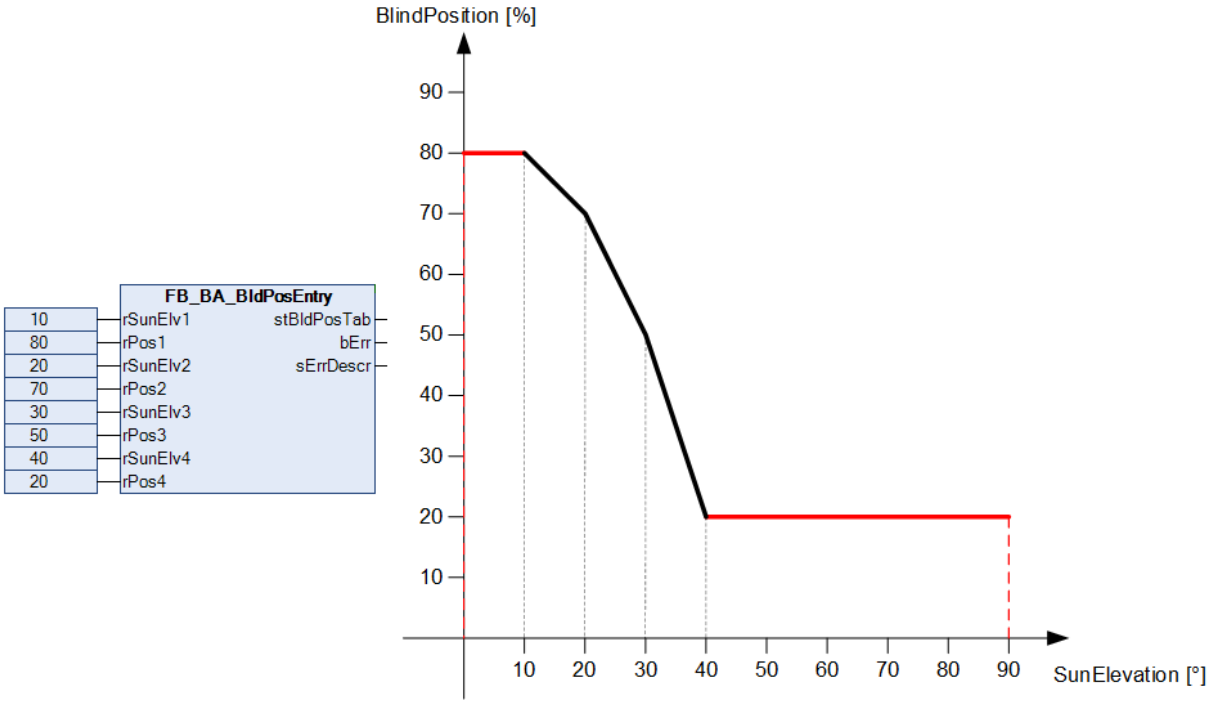
Dieser Baustein dient zur Eingabe von Stützstellen für den Baustein FB_BA_SunPrtc [▶ 621], falls dieser im Modus der Höhenpositionierung mit Hilfe einer Tabelle betrieben wird, siehe E_BA_PosMod [▶ 698].

Der Baustein FB_BA_SunPrtc [▶ 621] bietet neben den Betriebsarten "fixe Jalousiehöhe" und "maximaler Lichteinfall" auch die Möglichkeit, die Jalousiehöhe per Tabelleneinträge in Abhängigkeit von der Sonnenhöhe zu steuern. Durch die Eingabe mehrerer Stützpunkte wird linear interpoliert die betreffende Jalousiehöhe zum entsprechenden Sonnenstand errechnet. Da falsch eingetragene Werte jedoch zu Fehlfunktionen beim FB_BA_SunPrtc [▶ 621] führen können ist diesem der Baustein FB_BA_BldPosEntry

voranzustellen. Es lassen sich vier Stützpunkte an diesem Baustein parametrieren, wobei ein fehlender Eintrag als Nulleintrag gewertet wird.
Der Baustein sortiert die eingegebenen Werte nicht selbstständig, sondern achtet darauf, dass die eingetragenen Sonnenstände der jeweiligen Stützstellen in aufsteigender Reihenfolge eingegeben wurden. Unbeabsichtigt fehlerhafte Einträge fallen dadurch schneller auf.
Die gewählten Werte für *rSunElv1* .. *rSunElv4* müssen auch eindeutig sein, es darf beispielsweise nicht gelten:
[*rSunElv1* = 10 ; *rPos1* = 50] und gleichzeitig [*rSunElv2* = 10 *; rPos2* = 30].
Das würde bedeuten, dass für ein und denselben Wert zwei verschiedene Zielwerte bestünden, was keinen eindeutigen funktionalen Zusammenhang bilden lässt.
Darüber hinaus müssen die Einträge für Sonnenstand und Jalousiehöhe im gültigen Bereich liegen.
Das bedeutet mathematisch, dass folgende Bedingungen erfüllt sein müssen:

- rSunElv1 < *rSunElv2* < *rSunElv3* < *rSunElv4* - (Werte aufsteigend und ungleich)
- 0 ≤ *rSunElv* ≤ 90 ([°] - Gültigkeitsbereich Quellwerte)
- 0 ≤ *rPos* ≤ 100 (in Prozent - Gültigkeitsbereich Zielwerte)

Der Baustein überprüft die eingetragenen Werte auf diese Bedingungen hin und gibt bei Nichterfüllen Fehlermeldung aus. Zusätzlich wird der Wert *bValid* von ST_BA_BldPosTab [▶ 699] auf FALSE gesetzt.
Des Weiteren sorgt der Baustein selbstständig für ein Ausfüllen der Randbereiche: Intern wird ein weiterer Stützpunkt bei *rSunElv* = 0 mit *rBldPos1*und ein weiterer oberhalb von *rSunElv4* bei *rSunElv* = 90 mit *rBldPos4* aufgestellt. Damit wird sichergestellt, dass für alle gültigen Eingabewerte 0 ≤*rSunElv* ≤ 90 ein sinnvoller Zielwert vorhanden ist, **ohne** dass ein Eintrag für *rSunElv* = 0 und *rSunElv* = 90 vom Anwender zwingend vergeben werden muss:



Die tatsächliche Anzahl an Stützstellen, welche an den Baustein FB_BA_SunPrtc [▶ 621] übergeben wird, erhöht sich damit auf 6, siehe ST_BA_BldPosTab [▶ 699].

Die Interpolation der Werte erfolgt im Blendschutz-Baustein.

VAR_INPUT

```
rSunElv1 : REAL;
rPos1    : REAL;
rSunElv2 : REAL;
rPos2    : REAL;
rSunElv3 : REAL;
rPos3    : REAL;
rSunElv4 : REAL;
rPos4    : REAL;
```

rSunElv1: Sonnenstand des 1. Stützpunktes (0…90 °).

rPos1: Jalousieposition (Grad der Schließung) des 1. Stützpunktes (0…100 %).

rSunElv2: Sonnenstand des 2. Stützpunktes (0…90 °).

rPos2: Jalousieposition (Grad der Schließung) des 2. Stützpunktes (0…100 %).

rSunElv3: Sonnenstand des 3. Stützpunktes (0…90 °).

rPos3: Jalousieposition (Grad der Schließung) des 3. Stützpunktes (0…100 %).

rSunElv4: Sonnenstand des 4. Stützpunktes (0…90 °).

rPos4: Jalousieposition (Grad der Schließung) des 4. Stützpunktes (0…100 %).

VAR_OUTPUT

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

stBldPosTab: Übergabestruktur der Stützstellen, siehe ST_BA_BldPosTab [▶ 699].

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

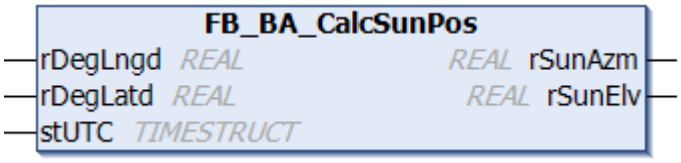
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Die x-Werte (Elevationswerte) in der Tabelle sind entweder nicht in aufsteigender Reihenfolge oder doppelt vorhanden.
02: Fehler: Ein eingetragener Elevationswert ist außerhalb des gültigen Bereiches von 0°...90°.
03: Fehler: Ein eingetragener Positionswert ist außerhalb des gültigen Bereiches von 0%...100%.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

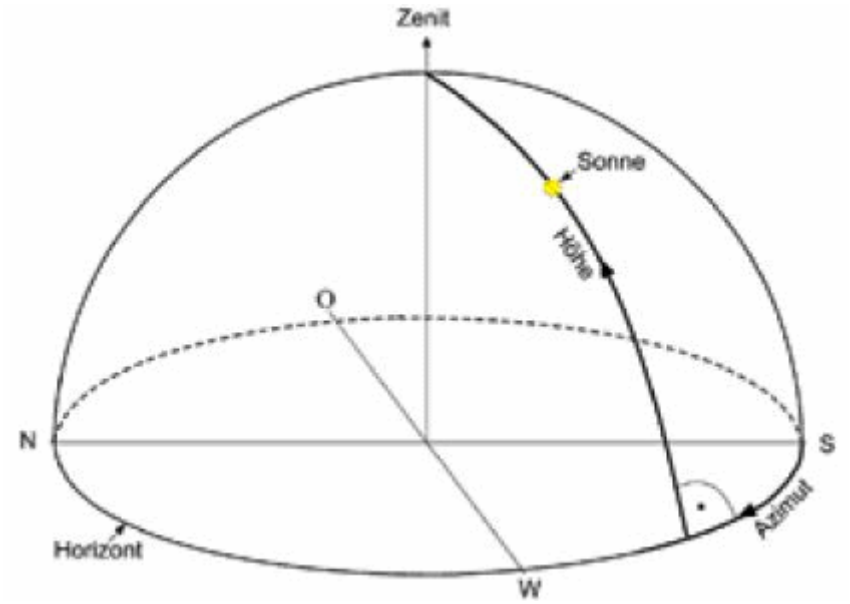
6.1.2.3.2.1.2.3.8 FB_BA_CalcSunPos



Berechnung des Sonnenstandes durch die Angabe von Datum, Uhrzeit, geografischer Länge und geografischer Breite.

Der Sonnenstand für einen gegebenen Zeitpunkt lässt sich nach gängigen Methoden mit definierter Genauigkeit berechnen. Für Anwendungen mit mäßigen Anforderungen genügt der hier vorliegende Baustein. Als Grundlage wurde hierfür der SUNAE-Algorithmus verwendet, der einen günstigen Kompromiss zwischen Genauigkeit und Rechenaufwand darstellt.

Der Stand der Sonne an einem festen Beobachtungsort wird normalerweise mit zwei Winkelangaben bestimmt. Die eine Winkelangabe zeigt die Höhe über dem Horizont an, wobei 0° bedeutet, dass sich die Sonne in der Horizontalebene des Standortes befindet und ein Wert von 90°, dass sie sich senkrecht über dem Beobachter befindet. Die andere Winkelangabe zeigt die Richtung an, in der die Sonne steht. Bei dem SUNAE-Algorithmus wird unterschieden, ob der Beobachter auf der nördlichen (Längengrad > 0) oder auf der südlichen (Längengrad < 0) Erdhalbkugel steht. Ist der Beobachtungspunkt auf der nördlichen Erdhalbkugel, so wird ein Wert von 0° für die nördliche Sonnenrichtung zugeordnet und läuft dann im Uhrzeigersinn um den Kompass, d.h. 90° ist Osten, 180° ist Süden, 270° Westen, etc. Ist der Beobachtungspunkt auf der südlichen Erdhalbkugel, so entspricht 0° der südlichen Richtung und verläuft dann entgegen dem Uhrzeigersinn, d.h. 90° ist Osten, 180° ist Norden, 270° Westen, etc.



Bei der Angabe der Uhrzeit muss die Zeit als koordinierte Weltzeit (UTC, Universal Time Coordinated - im älteren Sprachgebrauch auch GMT, Greenwich-Mean-Time) angegeben werden.

Die geografische Breite ist die im Winkelmaß (also [°]) angegebene nördliche oder südliche Entfernung eines Ortes der Erdoberfläche vom Äquator. Die Breite kann Werte von 0° (am Äquator) bis ±90° (an den Polen) annehmen. Dabei gibt ein positives Vorzeichen die nördliche Richtung und ein negatives Vorzeichen die südliche Richtung an. Die geografische Länge ist ein Winkel, der ausgehend vom Nullmeridian 0° (künstlich festgelegte Nord-Süd-Linie) Werte bis ±180° annehmen kann. Ein positives Vorzeichen gibt die Länge in östlicher Richtung und ein negatives Vorzeichen in westlicher Richtung an. Beispiele:

Ort	geographische Länge	geographische Breite
Sydney, Australien	151,2°	-33,9°
New York, USA	-74,0°	40,7°
London, England	-0,1°	51,5°
Moskau, Russland	37,6°	55,7°
Peking, China	116,3°	39,9°
Dubai, Vereinigte Arabische Emirate	55,3°	25,4°
Rio de Janeiro, Brasilien	-43,2°	-22,9°
Hawaii, USA	-155,8°	20,2°
Verl, Deutschland	8,5°	51,9°

Gibt der Baustein *FB_BA_CalcSunPos* für die Sonnenhöhe *rSunElv* einen negativen Wert zurück, so ist die Sonne nicht sichtbar. Dieses kann zur Bestimmung von Sonnenaufgang und Sonnenuntergang genutzt werden.

VAR_INPUT

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

rDegLngd: geographische Länge [°].

rDegLatd: geographische Breite [°].

stUTC: Eingabe der aktuellen Uhrzeit als koordinierte Weltzeit (siehe TIMESTRUCT). Mit Hilfe des Bausteines FB_BA_GetTime [▶ 631] kann diese Zeit aus einem Zielsystem gelesen werden.

VAR_OUTPUT

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

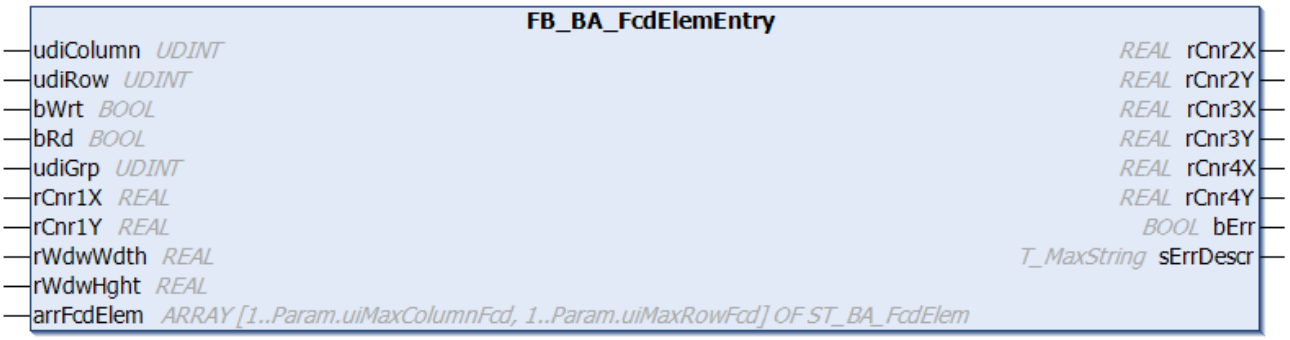
rSunAzm: Sonnenrichtung (nördliche Erdhalbkugel: 0° Norden ... 90° Osten ... 180° Süden ... 270° Westen ... / südliche Erdhalbkugel: 0° Süden ... 90° Osten ... 180° Norden ... 270° Westen ...).

rSunElv: Sonnenhöhe (0° horizontal ... 90° senkrecht).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.9 FB_BA_FcdElemEntry



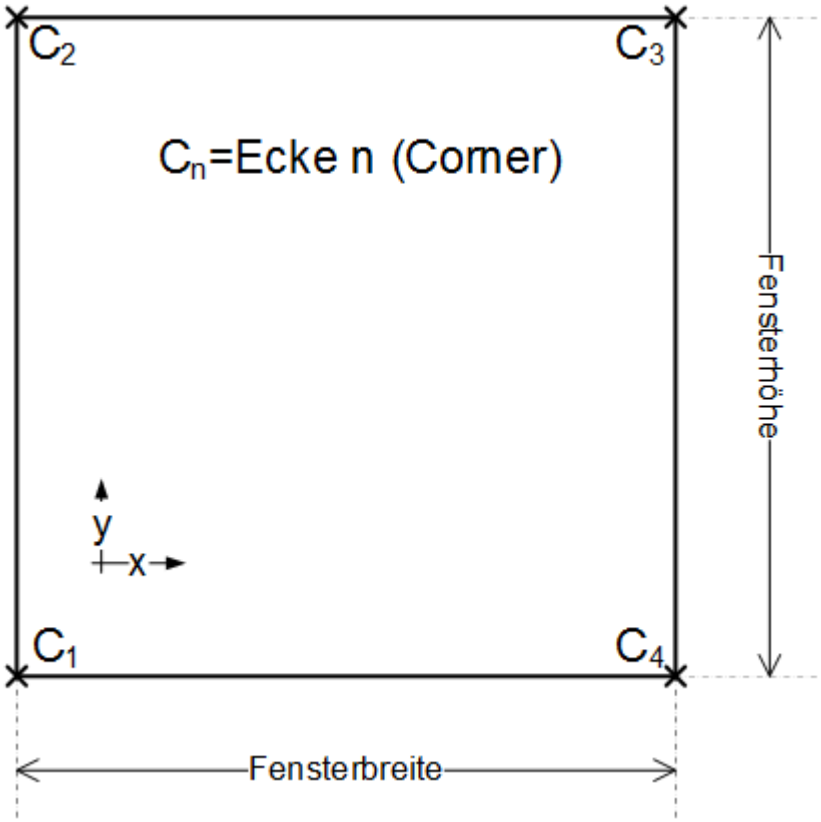
Dieser Baustein dient zur Verwaltung aller Fassadenelemente (Fenster) einer Fassade, welche global in einer Liste von Fassadenelementen [▶ 572] hinterlegt ist. Er soll die Eingabe der Elementinformationen - auch im Hinblick auf die Nutzung der TC3 PLC HMI - erleichtern. Eine schematische Darstellung der Objekte mit Beschreibung der Koordinaten ist unter Verschattungskorrektur: Grundlagen und Definitionen [▶ 558] gegeben.

Die Deklaration der Fassadenelemente erfolgt in den globalen Variablen als zweidimensionales Feld über die Fensterspalten (Columns) und -reihen (Rows):

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

Jedes einzelne Element *arrFcdElem[x,y]* trägt die Informationen für jeweils ein Fassadenelement (ST_BA_FcdElem [▶ 700]). Dazu gehören die Gruppenzugehörigkeit, die Abmessungen (Breite, Höhe) und die Koordinaten der Eckpunkte. Der Baustein greift dabei über die IN-OUT-Variable *arrFcdElem* direkt auf dieses Feld zu.

Anmerkung: Die Tatsache, dass die Koordinaten der Eckpunkte C2 bis C4 Ausgangswerte sind, ergibt sich daraus, dass sie aus den Eingabeparametern gebildet werden und für die Verwendung in einer Visualisierung zur Verfügung stehen sollen:



Alle Angaben in [m]!

rCnr2X = *rCnr1X*
rCnr2Y = *rCnr1Y* + *rWdwHght* (Fensterhöhe)
rCnr3X = *rCnr1X* + *rWdwWdth* (Fensterbreite)
rCnr3Y = *rCnr2Y*
rCnr4X = *rCnr1X* + *rWdwWdth* (Fensterbreite)
rCnr4Y = *rCnr1Y*

Die Verwendung des Bausteines erfolgt in drei Schritten:

- Auslesen
- Ändern
- Schreiben

Auslesen

Mit den Einträgen an *udiColumn* und *udiRow* wird das entsprechende Element aus der Liste, *arrFcdElem[udiColumn, udiRow]*, ausgewählt. Eine steigende Flanke an *bRd* liest folgende Daten aus dem Listenelement aus:

- udiGrp Gruppenzugehörigkeit,
- rCnr1X X-Koordinate des Eckpunktes1 [m]
- rCnr1Y Y-Koordinate des Eckpunktes1 [m]
- rWdwWdth Fensterbreite [m]
- rWdwHght Fensterhöhe [m]

Diese werden dann den entsprechenden Eingangsvariablen des Bausteines zugewiesen, der daraus nach dem oben erläuterten Zusammenhang die Koordinaten der Eckpunkte C2-C4 als Ausgangsvariablen errechnet. Wichtig ist hierbei, dass im Schritt des Auslesens die Eingangswerte nicht überschrieben werden. So lassen sich alle Werte zunächst in einer Visualisierung anzeigen.

Ändern

In einem nächsten Programmschritt können die aufgeführten Eingabewerte verändert werden. Die eingegebenen Werte werden dabei ständig auf Plausibilität überprüft. Der Ausgang *bErr* zeigt an, ob die Werte gültig sind (*bErr*=FALSE). Wenn die Werte ungültig sind, wird am Ausgang *sErrDescr* eine entsprechende Fehlermeldung ausgegeben. Siehe auch unten "Fehler (*bErr*=TRUE)".

Schreiben

Mit einer positiven Flanke an *bWrt* werden die parametrierten Daten in das Listenelement mit dem Index *nId* geschrieben, unabhängig davon, ob sie gültige Werte darstellen oder nicht. Daher ist innerhalb der Elementstruktur ST_BA_FcdElem [▶ 700] ebenfalls ein Plausibilitätsbit *bVld* vorhanden, das genau diese Information an den Baustein FB_BA_ShdCorr [▶ 596] weiterreicht und dort Fehlberechnungen vorbeugt.

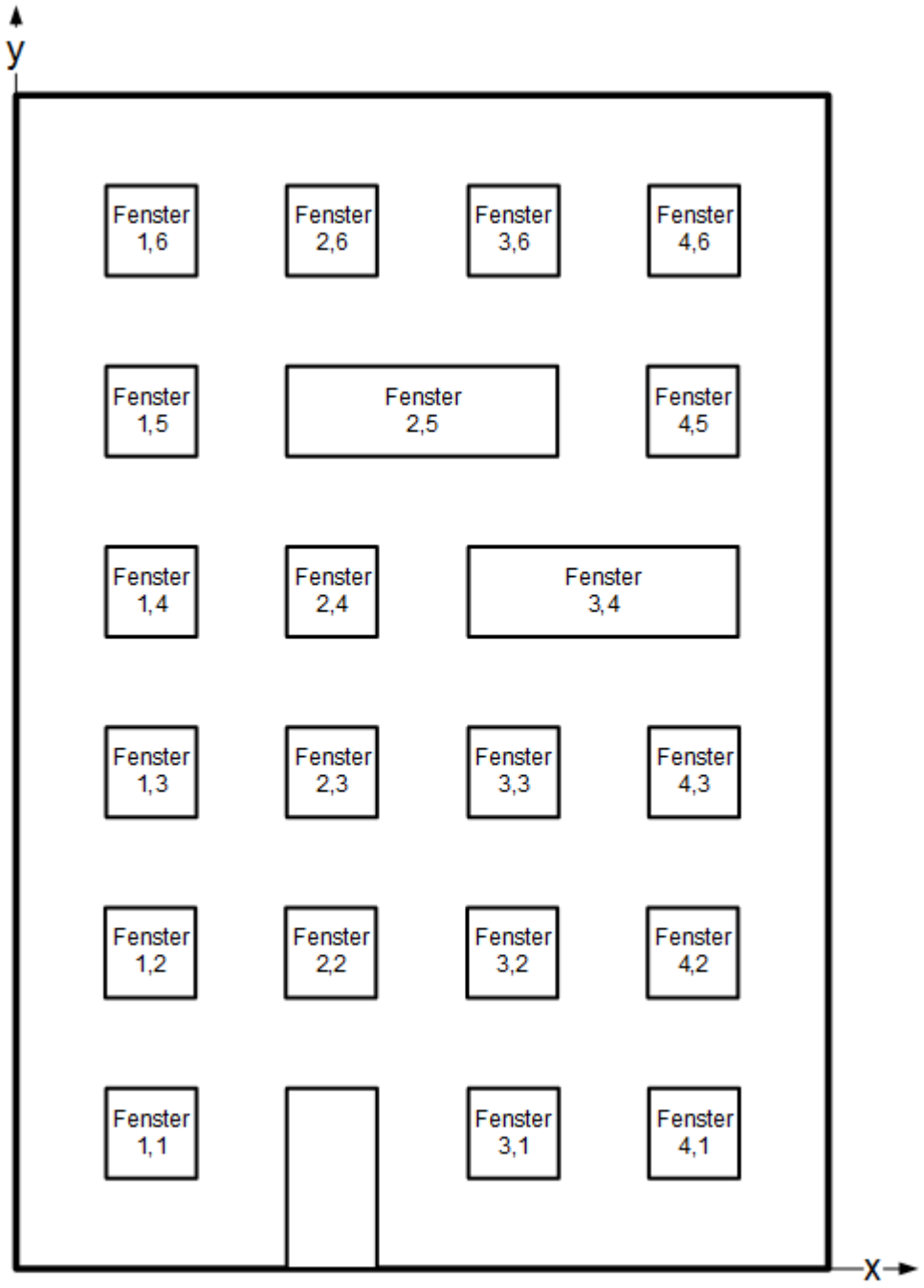
Diese Vorgehensweise soll nur als Vorschlag angesehen werden. Es ist natürlich auch möglich, den Baustein ganz normal in einem Schritt zu parametrieren und die eingetragenen Werte mit einer steigenden Flanke an *bWrt* in das entsprechende Listenelement zu schreiben.

Fehler (bErr=TRUE)

Der Baustein FB_BA_ShdCorr [▶ 596], welcher beurteilt ob alle Fenster einer Gruppe verschattet sind, wird nur dann seine Aufgabe ausführen, wenn alle Fenster der betrachteten Gruppe gültige Einträge haben.
Das bedeutet:

- udiGrp muss größer als 0 sein
- rCnr1X muss größer oder gleich 0.0 sein
- rCnr1Y muss größer oder gleich 0.0 sein
- rWdwWdth muss größer als 0 sein
- rWdwHght muss größer als 0 sein

Ist eines dieser Kriterien nicht erfüllt, so wird dies als Falscheingabe interpretiert und der Fehlerausgang *bErr* am Bausteinausgang von FB_BA_FcdElemEntry gesetzt. Innerhalb des Fensterelementes ST_BA_FcdElem [▶ 700] wird das Plausibilitätsbit *bVld* auf FALSE gesetzt.
Sind hingegen **alle** Einträge eines Fassadenelementes Null, wird es als gültiges, gewollt ausgelassenes Fassadenelement angesehen:



Bei einer Fassade von 6x4 Fenstern wären hier die Elemente Fenster (2,1), Fenster (3,5) und Fenster (4,4) Leerelemente.

VAR_INPUT

```
udiColumn : UDINT;
udiRow    : UDINT;
bWrt      : BOOL;
bRd       : BOOL;
udiGrp    : UDINT;
rCnr1X    : REAL;
rCnr1Y    : REAL;
rWdwWdth  : REAL;
rWdwHght  : REAL;
```

udiColumn: Spalten-Index des gewählten Elementes auf der Fassade. Dies bezieht sich auf die Auswahl eines Feldelementes des unter der IN-OUT-Variablen *arrFcdElem* angelegten Arrays.

udiRow: dto. Reihen-Index. *udiRow* **und** *udiColumn* **dürfen nicht Null sein!** Das ergibt sich aus der Felddefinition, welche jeweils mit 1 beginnt, siehe oben.

bRd: Mit einer positiven Flanke an diesem Eingang werden die Informationen des gewählten Elementes, *arrFcdElem[udiColumn,udiRow]* in den Baustein gelesen und den Eingangsvariablen *diGrp* bis *rWdwHght* zugewiesen. Daraus ergeben sich dann die Ausgangsvariablen *rCnr2X* bis *rCnr4Y*. Sind zu Zeitpunkt des Auslesens schon Daten an den Eingängen *diGrp* bis *rWdwHght* angelegt, so werden die zuvor ausgelesenen Daten sofort mit diesen überschrieben.

bWrt: Eine positive Flanke schreibt die eingegebenen sowie errechneten Werte in das gewählte Feldelement *arrFcdElem[udiColumn, udiRow]*.

udiGrp: Gruppenzugehörigkeit. Intern begrenzt auf einen Minimalwert von 0.

rCnr1X: X-Koordinate des Eckpunktes1 [m].

rCnr1Y: Y-Koordinate des Eckpunktes1 [m].

rWdwWdth: Fensterbreite [m].

rWdwHght: Fensterhöhe [m].

VAR_OUTPUT

```
rCnr2X   : REAL;
rCnr2Y   : REAL;
rCnr3X   : REAL;
rCnr3Y   : REAL;
rCnr4X   : REAL;
rCnr4Y   : REAL;
bErr     : BOOL;
sErrDesc : T_MAXSTRING;
```

rCnr2X: Ermittelte X-Koordinate des Eckpunktes 2 des Fensters [m]. Siehe "Anmerkung [▶ 576]" oben.

rCnr2Y: Ermittelte Y-Koordinate des Eckpunktes 2 des Fensters [m]. Siehe "Anmerkung [▶ 576]" oben.

rCnr3X: Ermittelte X-Koordinate des Eckpunktes 3 des Fensters [m]. Siehe "Anmerkung [▶ 576]" oben.

rCnr3Y: Ermittelte Y-Koordinate des Eckpunktes 3 des Fensters [m]. Siehe "Anmerkung [▶ 576]" oben.

rCnr4X: Ermittelte X-Koordinate des Eckpunktes 4 des Fensters [m]. Siehe "Anmerkung [▶ 576]" oben.

rCnr4Y: Ermittelte Y-Koordinate des Eckpunktes 4 des Fensters [m]. Siehe "Anmerkung [▶ 576]" oben.

bErr: Ergebniskontrolle für die eingegebenen Werte.

sErrDesc: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Indexfehler! udiColumn und/oder udiRow liegen außerhalb der zulässigen Grenzen 1.. uiMaxColumnFcd bzw. 1.. uiMaxColumnFcd. Siehe Liste der Fassadenelemente.
02: Fehler: Der Gruppenindex ist 0, gleichzeitig ist jedoch ein anderer Eintrag des Fassadenelementes ungleich Null. Nur wenn alle Einträge eines Fassadenelementes Null sind, wird es als gültiges, gewollt ausgelassenes Fassadenelement angesehen, ansonsten jedoch als Fehleingabe interpretiert. HINWEIS: Gruppeneinträge kleiner Null werden intern auf Null begrenzt!
03: Fehler: Die X-Komponente des ersten Eckpunktes (Corner1) ist kleiner Null.
04: Fehler: Die Y-Komponente des ersten Eckpunktes (Corner1) ist kleiner Null.
05: Fehler: Die Fensterbreite ist kleiner oder gleich Null.
06: Fehler: Die Fensterhöhe ist kleiner oder gleich Null.

VAR_IN_OUT

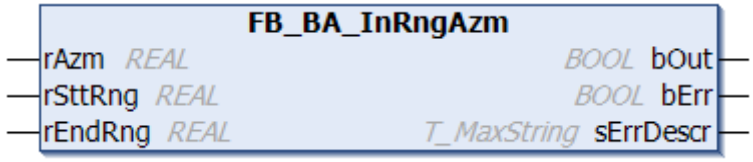
```
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: Liste von Fassadenelementen (siehe Liste der Fassadenelemente [▶ 572]).

Voraussetzungen

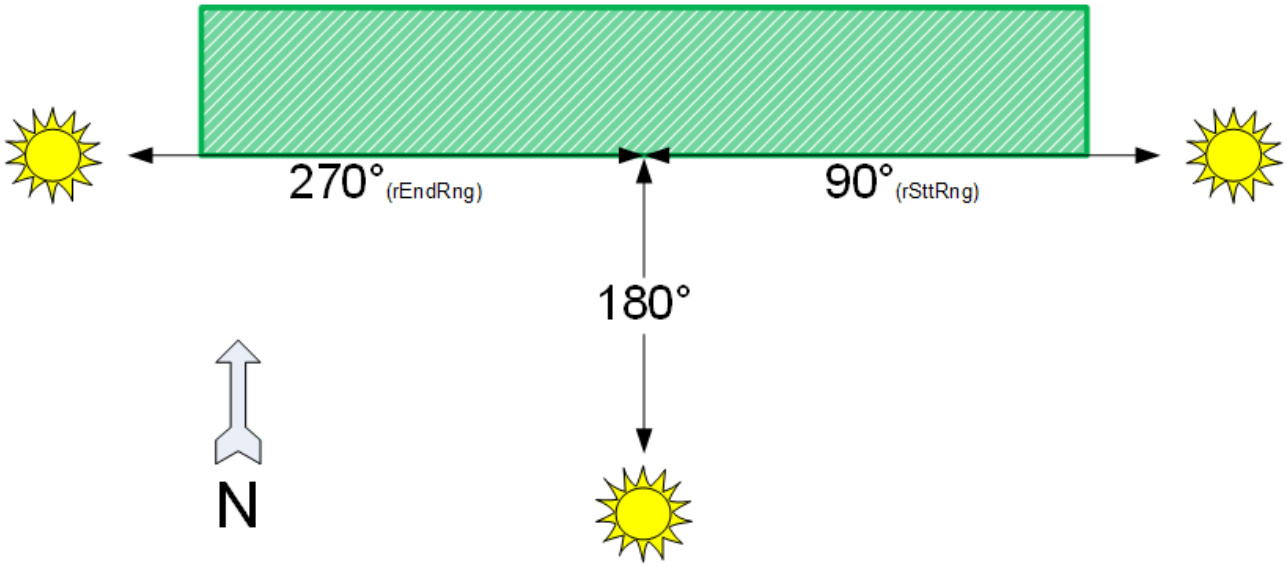
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.10 FB_BA_InRngAzm

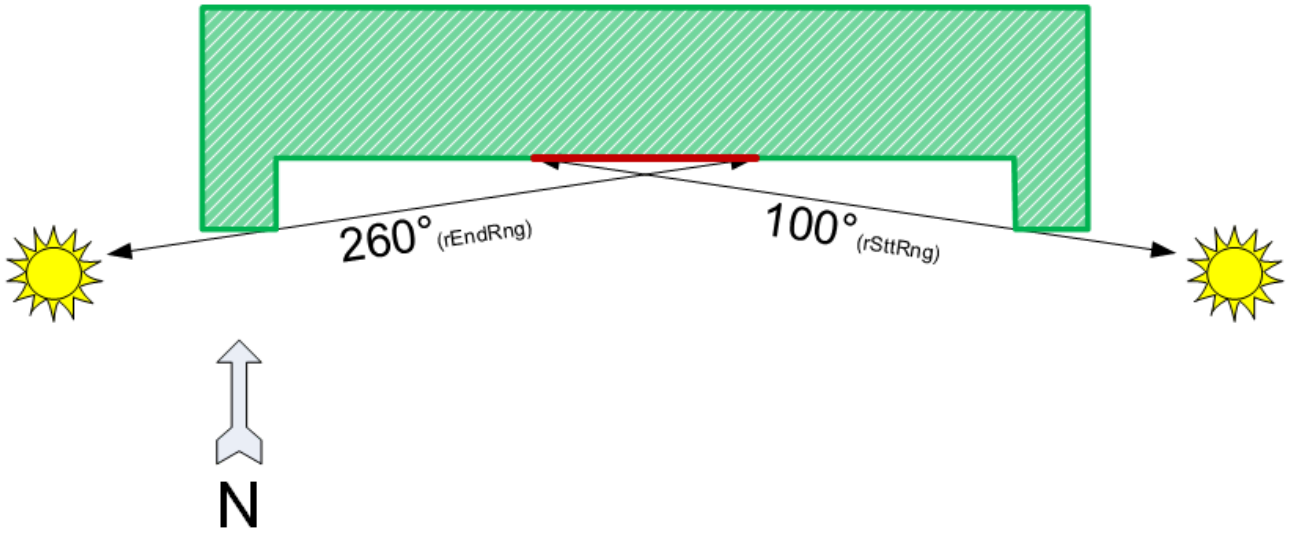


Dieser Baustein prüft, ob der aktuelle Azimutwinkel (horizontaler Sonnenstand) innerhalb der eingetragenen Grenzen liegt. Wie in der Übersicht [▶ 566] erkennbar, gibt der Baustein eine zusätzliche Bewertung, ob der Sonnenschutz einer Fenstergruppe aktiviert werden soll. Daher gelten die Betrachtungen im weiteren Text immer für eine Fenstergruppe.

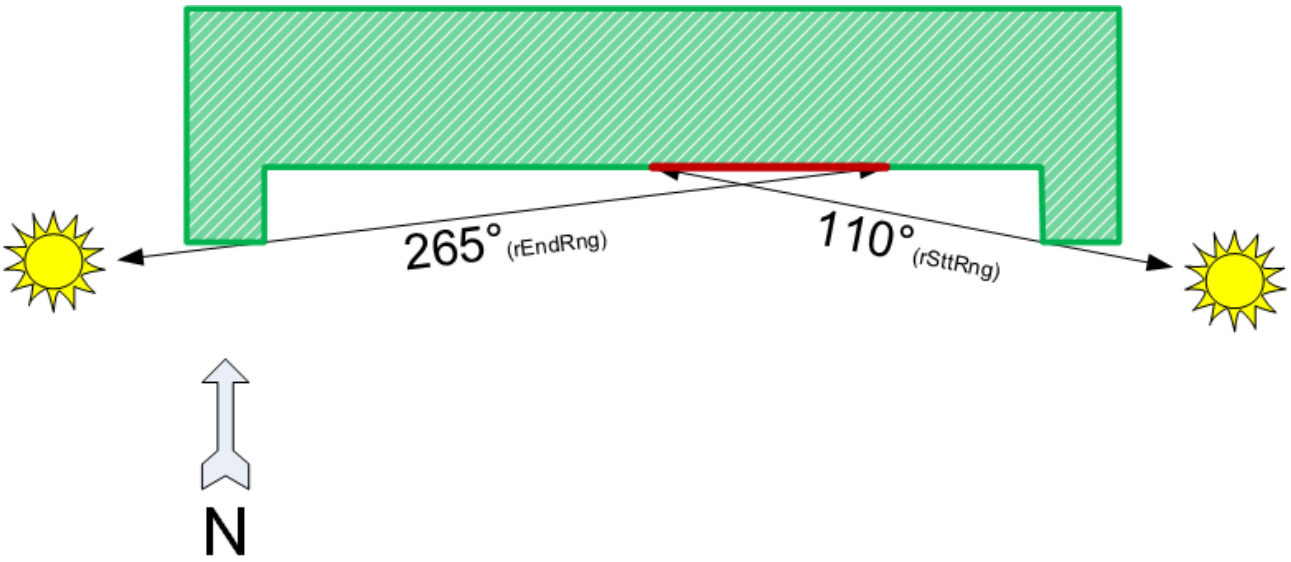
Eine glatte Fassade wird von der Sonne immer in einem Azimutwinkel von *Fassadenausrichtung-90°... Fassadenausrichtung+90°* bestrahlt.



Hat die Fassade jedoch seitliche Vorsprünge, so wird dieser Bereich eingeschränkt. Diese Einschränkung lässt sich mit Hilfe dieses Bausteines überprüfen. Dabei spielt aber auch die Lage der Fenstergruppe auf der Fassade eine Rolle. Liegt sie zentral, so ergibt sich folgende Situation (Die Werte sind dabei nur beispielhaft):



Für eine Gruppe am Rand andern sich die Werte:



Der Anfang des Bereiches *rSttRng* darf dabei größer sein als das Ende *rEndRng*, es wird dann über 0° hinaus betrachtet:

Beispiel

rAzm	10.0°
rSttRng	280.0°
rEndRng	20.0°
bOut	**TRUE**

Der betrachtete Bereich darf jedoch nicht größer als 180° oder gleich 0° sein, dieses wäre unrealistisch. Derartige Eingaben ergeben einen Fehler am Ausgang *bErr* - der Prüfausgang *bOut* wird dabei zusätzlich auf FALSE gesetzt.

VAR_INPUT

```
rAzm    : REAL;
rSttRng : REAL;
rEndRng : REAL;
```

rAzm: Aktueller Azimutwinkel.

rSttRng: Bereichsanfang [°].

rEndRng: Bereichsende [°].

VAR_OUTPUT

```
bOut       : BOOL;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
```

bOut: Das Fassadenelement liegt in der Sonne wenn der Ausgang TRUE ist.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

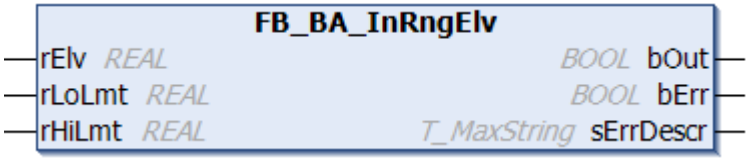
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: *rSttRng* bzw. *rEndRng* kleiner 0° oder größer 360°.
02: Fehler: Die Differenz von *rSttRng* und *rEndRng* ist größer als 180Grad. Für die Betrachtung des Sonneneinfalles auf eine Fassade ist dieser Bereich zu groß gewählt.

Voraussetzungen

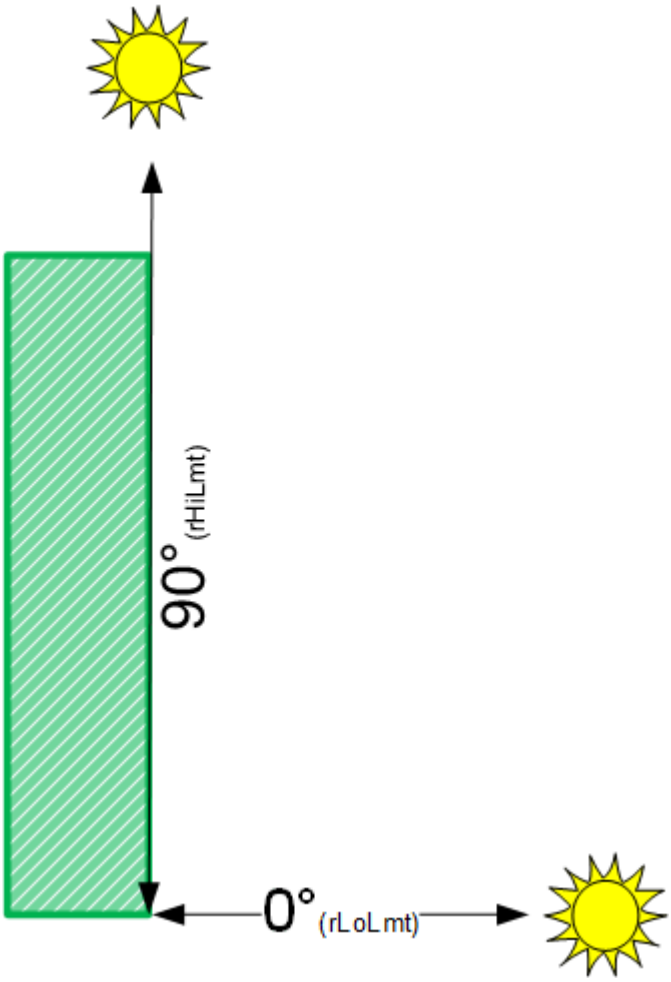
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.11 FB_BA_InRngElv

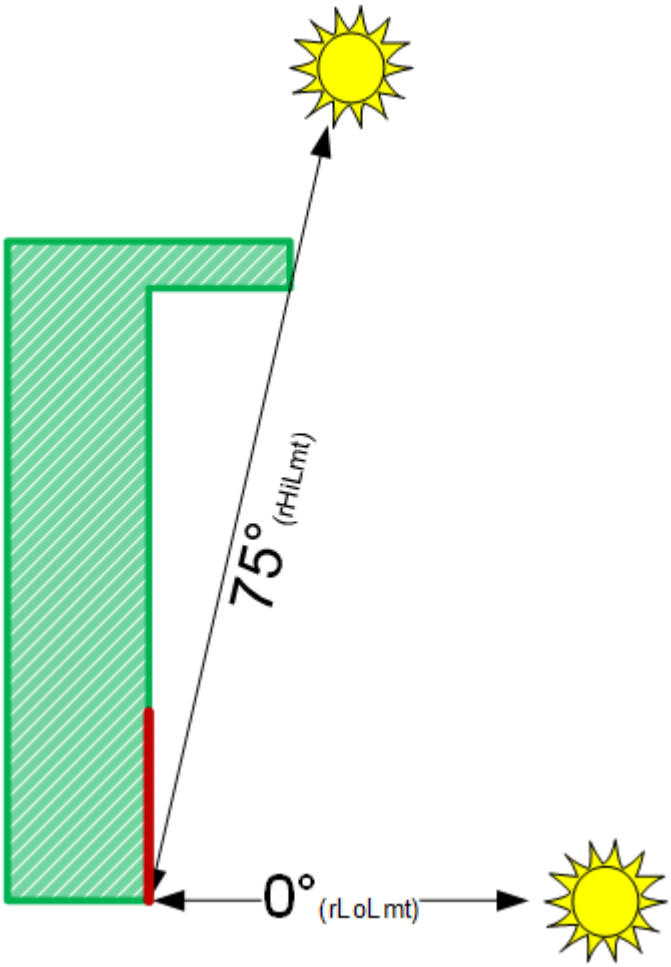


Dieser Baustein prüft, ob der aktuelle Elevationswinkel (vertikaler Sonnenstand) innerhalb der eingetragenen Grenzen liegt. Wie in der Übersicht [▶ 566] erkennbar, gibt der Baustein eine zusätzliche Bewertung, ob der Sonnenschutz einer Fenstergruppe aktiviert werden soll. Daher gelten die Betrachtungen im weiteren Text immer für eine Fenstergruppe.

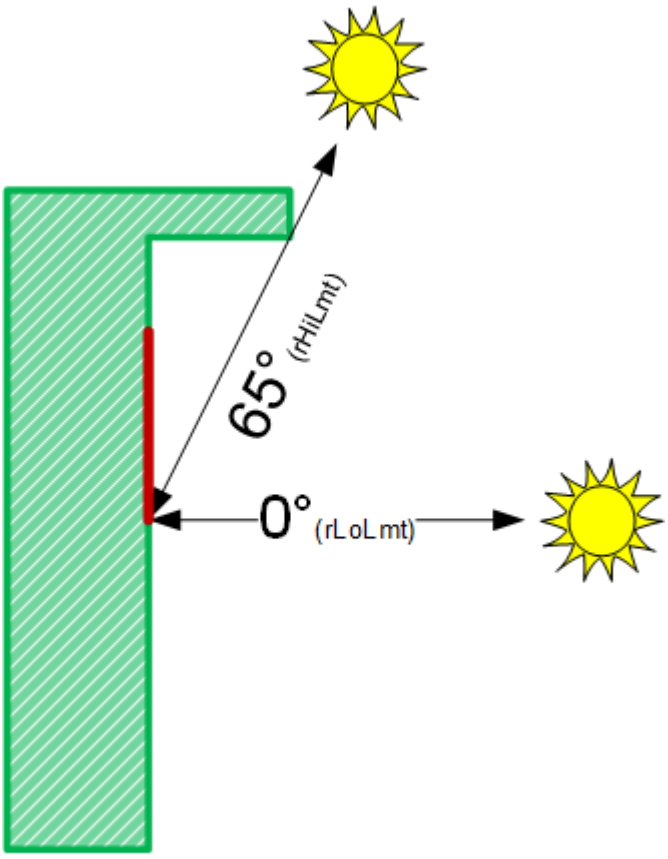
Eine normale senkrechte Fassade wird von der Sonne immer in einem Elevationswinkel von 0° bis maximal 90° bestrahlt.



Hat die Fassade jedoch Vorsprünge, so wird dieser Bereich eingeschränkt. Diese Einschränkung lässt sich mit Hilfe dieses Bausteines überprüfen. Dabei spielt aber auch die Lage der Fenstergruppe auf der Fassade eine Rolle. Liegt sie im unteren Bereich, so ergibt sich folgende Situation (Die Werte sind dabei nur beispielhaft):



Für eine Gruppe unterhalb des Vorsprunges ändern sich die Werte:



Die untere Betrachtungsgrenze, *rLoLmt*, darf dabei nicht größer oder gleich der oberen, *rHiLmt*, sein. Derartige Eingaben ergeben einen Fehler am Ausgang *bErr* - der Prüfausgang *bOut* wird dabei zusätzlich auf FALSE gesetzt.

VAR_INPUT

```
rElv   : REAL;
rLoLmt : REAL;
rHiLmt : REAL;
```

rElv: Aktueller Elevationswinkel [°].

rLoLmt: Unterer Grenzwert [°].

rHiLmt: Oberer Grenzwert [°].

VAR_OUTPUT

```
bOut      : BOOL;
bErr      : BOOL;
sErrDescr : T_MAXSTRING;
```

bOut: Das Fassadenelement liegt in der Sonne

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

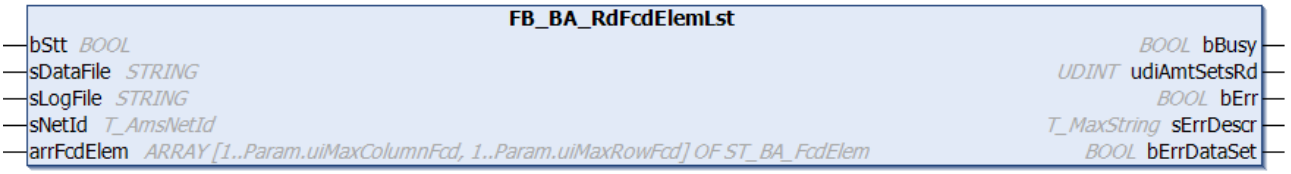
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: *rHiLmt* kleiner oder gleich *rLoLmt*.
02: Fehler: *rLoLmt* ist kleiner 0° oder *rHiLmt* ist größer 90°.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.12 FB_BA_RdFcdElemLst



Mit Hilfe dieses Bausteines lassen sich Daten für Fassadenelemente (Fenster) aus einer vordefinierten Excel-Tabelle im csv-Format in die Liste der Fassadenelemente [▶ 572] importieren. Zusätzlich werden die importierten Daten auf Plausibilität überprüft und Fehler in eine Log-Datei geschrieben.

Das folgende Beispiel zeigt die Excel-Tabelle mit den Einträgen der Fensterelemente.
Alle Textfelder sind frei beschreibbar. Wichtig sind die grün markierten Felder, jede Zeile dort kennzeichnet einen Datensatz.
Folgende Regeln sind zu beachten:

- Ein Datensatz muss immer mit einem '@' beginnen.
- Die Indizes *IndexColumn* und *IndexRow* müssen innerhalb der definierten Grenzen liegen, siehe Liste von Fassadenelementen [▶ 572]. Mit diesen Indizes wird direkt beschrieben, in welches Fassadenelement der Liste *arrFcdElem* die Daten des Satzes hineingelegt werden.
- Fensterbreite und Fensterhöhe müssen größer Null sein
- Die Eck-Koordinaten P1x und P1y müssen größer oder gleich Null sein.
- Jedes Fensterelement muss einer Gruppe 1..255 zugeordnet sein.
- Die Gesamtgröße der Tabelle darf systembedingt 65534 Bytes nicht überschreiten.

- Die Tabelle muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein.

Es ist nicht notwendig, alle Fensterelemente zu beschreiben, die per Definition bzw. Deklaration möglich wären. Bevor die neue Liste eingelesen wird, löscht der Baustein die gesamte alte Liste im Programm. Alle Elemente, welche dann nicht durch Einträge in der Excel-Tabelle beschrieben werden, haben reine Null-Einträge und sind dadurch als nicht vorhanden gekennzeichnet und auch nicht auswertbar, da der Baustein zur Verschattungskorrektur, FB_BA_ShdCorr [▶ 596], Elemente mit Gruppeneintrag '0' nicht annimmt.

DE_FacadeElements.xls [Kompatibilitätsmodus]

	A	B	C	D	E	F	G	H	I	J
1	Nummer	Beschreibung		IndexColumı	IndexRow	Fensterbreit	Fensterhoeh	P1x	P1y	Gruppe
2				(Achse)	(Stockwerk)	[m]	[m]	[m]	[m]	
3		Text								
4	1	Beschreibung	@	1	1	1,2	1,3	1,5	1	2
5	2	Beschreibung	@	0	1	1,2	1,3	2,7	1	2
6	3	Beschreibung	@	3	1	1,2	1,3	4,4	1	2
7	4	Beschreibung	@	4	1	1,2	1,3	6,1	1	2
8	5	Beschreibung	@	5	1	1,2	1,3	7,8	1	2
9	6	Beschreibung	@	6	1	1,2	1,3	9,5	1	2
10	7	Beschreibung	@	7	1	1,2	1,3	11,2	1	2
11	8	Beschreibung	@	8	1	1,2	1,3	12,9	1	2
12	9	Beschreibung	@	9	1	1,2	1,3	14,6	1	2
13	10	Beschreibung	@	10	1	1,2	1,3	16,3	1	2
14	11	Beschreibung	@	1	1	1,2	1,3	1,5	4	2
15	12	Beschreibung	@	0	1	1,2	1,3	2,7	4	2
16	13	Beschreibung	@	3	1	1,2	1,3	4,4	4	2
17	14	Beschreibung	@	4	1	1,2	1,3	6,1	4	2
18	15	Beschreibung	@	5	1	1,2	1,3	7,8	4	2
19	16	Beschreibung	@	6	1	1,2	1,3	9,5	4	2
20	17	Beschreibung	@	7	1	1,2	1,3	11,2	4	2
21	18	Beschreibung	@	8	1	1,2	1,3	12,9	4	2
22	19	Beschreibung	@	9	1	1,2	1,3	14,6	4	2
23	20	Beschreibung	@	10	1	1,2	1,3	16,3	4	2
24	21	Beschreibung	@	1	1	1,2	1,3	1,5	7	2
25	22	Beschreibung	@	0	1	1,2	1,3	2,7	7	2
26	23	Beschreibung	@	3	1	1,2	1,3	4,4	7	2
27	24	Beschreibung	@	4	1	1,2	1,3	6,1	7	2
28	25	Beschreibung	@	5	1	1,2	1,3	7,8	7	2
29	26	Beschreibung	@	6	1	1,2	1,3	9,5	7	2
30	27	Beschreibung	@	7	1	1,2	1,3	11,2	7	2
31	28	Beschreibung	@	8	1	1,2	1,3	12,9	7	2
32	29	Beschreibung	@	9	1	1,2	1,3	14,6	7	2
33	30	Beschreibung	@	10	1	1,2	1,3	16,3	7	2
34	31	Beschreibung	@	1	1	1,2	1,3	1,5	10	2
35	32	Beschreibung	@	0	1	1,2	1,3	2,7	10	2
36	33	Beschreibung	@	3	1	1,2	1,3	4,4	10	2
37	34	Beschreibung	@	4	1	1,2	1,3	6,1	10	2
38	35	Beschreibung	@	5	1	1,2	1,3	7,8	10	2
39	36	Beschreibung	@	6	1	1,2	1,3	9,5	10	2
40	37	Beschreibung	@	7	1	1,2	1,3	11,2	10	2
41	38	Beschreibung	@	8	1	1,2	1,3	12,9	10	2
42	39	Beschreibung	@	9	1	1,2	1,3	14,6	10	2
43	40	Beschreibung	@	10	1	1,2	1,3	16,3	10	2

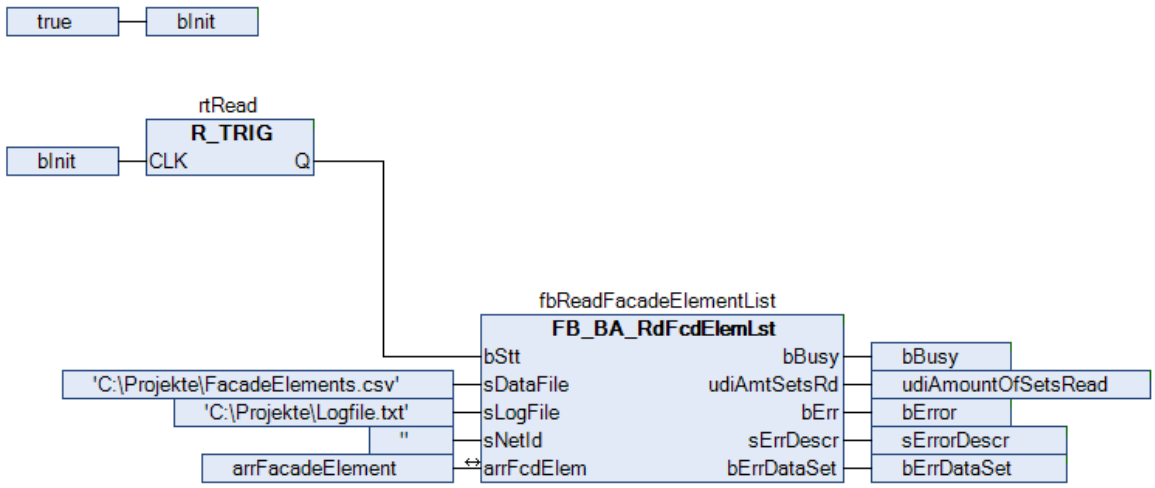
Log-Datei

Bei jedem Start des Lesebausteines wird die Logdatei neu beschrieben und der alte Inhalt gelöscht. Ist die Log-Datei nicht vorhanden, so wird sie zunächst automatisch erstellt. Die Log-Datei enthält dann entweder eine OK Meldung oder eine Auflistung aller aufgetretenen Fehler. Fehler, welche mit dem Öffnen, Beschreiben oder Schließen der Logdatei selbst in Zusammenhang stehen, können nicht mitgeschrieben werden. Daher ist immer auch der Ausgang *sErrDescr* des Lesebausteines zu beachten, der den letzten Fehlercode anzeigt. Da im Ablauf des Lesens die Log-Datei immer als Letztes geschlossen wird, ist ein entsprechender Hinweis im Fehlerfall gewährleistet.

Programmbeispiel

```
PROGRAM ReadFacadeElements
VAR
    bInit                   :   BOOL;
    rtRead                  :   R_TRIG;
    fbReadFacadeElementList :   FB_BA_RdFcdElemLst;
    arrFacadeElement        :   ARRAY [1..uiMaxColumnFcd, 1..uiMaxRowFcd] OF ST_BA_FcdElem;

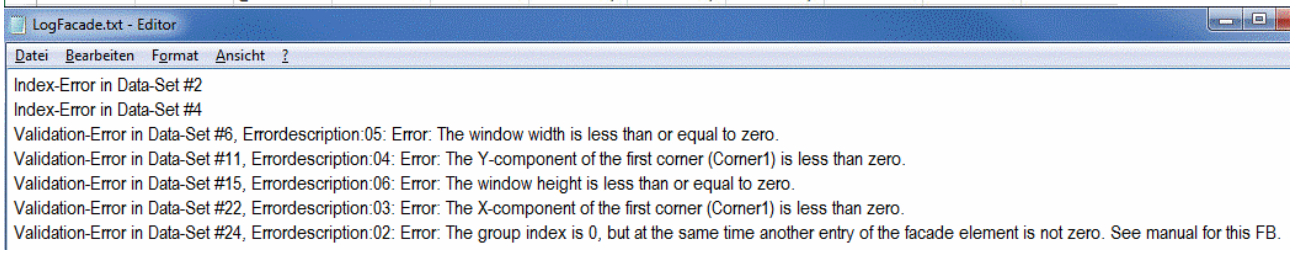
    bBusy                   :   BOOL;
    udiAmountOfSetsRead     :   UDINT;
    bError                  :   BOOL;
    sErrorDescr             :   T_MaxString;
    bErrDataSet             :   BOOL;
END_VAR
```



In diesem Beispiel wird bei PLC-Start die Variable *bInit* zunächst auf TRUE gesetzt. Damit erhält der Eingang *bStt* am Baustein *fbReadFacadeElementList* einmalig eine steigende Flanke, welcher den Lesevorgang auslöst. Gelesen wird die Datei "FacadeElements.csv", welche sich im Ordner "C:\Projekte\" befindet. In demselben Ordner wird dann die Logdatei "Logfile.txt" hinterlegt. Ist diese Logdatei noch nicht vorhanden wird sie erstellt, andernfalls wird der bestehende Inhalt überschrieben. Das Lesen und Schreiben erfolgt auf demselben Rechner, auf dem sich auch die PLC befindet. Das wird durch den Eingang *sNetID* = '' (=lokal) definiert. Alle Daten werden in die im Programm deklarierte Liste *arrFcdElem* geschrieben. Während gelesen und geschrieben wird ist der Ausgang *bBusy* auf TRUE. Der zuletzt auftretende Fehler in Bezug auf das Dateihandling wird an *sErrDescr* angezeigt, *bErr* steht dann auf TRUE. Wird hingegen ein Fehler im Datensatz erkannt, wird dies an *bErrDataSet* angezeigt und in der Log-Datei näher beschrieben. Die Anzahl der gefundenen und gelesenen Datenzeilen wird an *udiAmtSetsRd* zur Kontrolle angezeigt.

In die folgende Excel Liste wurden die markierten Fehler "eingebaut". Dadurch ergibt sich das gezeigte Log-File:

	A	B	C	D	E	F	G	H	I	J	K
1	Nummer	Beschreibung		IndexColumn	IndexRow	Fensterbreit	Fensterhoeh	P1x	P1y	Gruppe	
2				(Achse)	(Stockwerk)	[m]	[m]	[m]	[m]		
3		Text									
4	1	Beschreibun	@	1	1	1,2	1,3	1,5	1,4	2	
5	2	Beschreibun	@	0	1	1,2	1,3	2,7	1	2	
6	3	Beschreibun	@	3	1	1,2	1,3	4,4	1	2	
7	4	Beschreibun	@	4	-1	1,2	1,3	6,1	1	2	
8	5	Beschreibun	@	5	1	1,2	1,3	7,8	1	2	
9	6	Beschreibun	@	6	1	0	1,3	9,5	1	2	
10	7	Beschreibun	@	7	1	1,2	1,3	11,2	1	2	
11	8	Beschreibun	@	8	1	1,2	1,3	12,9	1	2	
12	9	Beschreibun	@	9	1	1,2	1,3	14,6	1	2	
13	10	Beschreibun	@	10	1	1,2	1,3	16,3	1	5	
14	11	Beschreibun	@	1	2	1,2	1,3	1	-1	5	
15	12	Beschreibun	@	2	2	1,2	1,3	2,7	3	5	
16	13	Beschreibun	@	3	2	1,2	1,3	4,4	4	5	
17	14	Beschreibun	@	4	2	1,2	1,3	4,4	4	5	
18	15	Beschreibun	@	5	2	1,2	0	7,8	4	5	
19	16	Beschreibun	@	6	2	1,2	1,3	9,5	4	5	
20	17	Beschreibun	@	7	2	1,2	1,3	11,2	4	5	
21	18	Beschreibun	@	8	2	1,2	1,3	12,9	4	5	
22	19	Beschreibun	@	9	2	1,2	1,3	14,6	4	3	
23	20	Beschreibun	@	10	2	1,2	1,3	16,3	4	3	
24											
25	31	Beschreibun	@	1	3	1,2	1,3	1	7	3	
26	32	Beschreibun	@	2	3	1,2	1,3	-1	6	3	
27	33	Beschreibun	@	3	3	1,2	1,3	4,4	7	3	
28	34	Beschreibun	@	4	3	1,2	1,3	6,1	7	0	
29	35	Beschreibun	@	5	3	1,2	1,3	7,8	7	3	
30	36	Beschreibun	@	6	3	1,2	1,3	9,5	7	3	
31	37	Beschreibu	@	7	3	1,2	1,3	11,2	7	3	
32	38	Beschreibun	@	8	3	1,2	1,3	12,9	7	7	
33	39	Beschreibun	@	9	3	1,2	1,3	14,6	7	7	
34	40	Beschreibun	@	10	3	1,2	1,3	16,3	7	7	



Der erste Fehler ist im Datensatz 2 und ein Index-Fehler, da "0" nicht erlaubt ist.
Der nächste Fehler in Datensatz 6 wurde nach Validierung der Daten mit dem intern genutzten Baustein FB_BA_ShdObjEntry [▶ 599] gefunden und näher mit einer Fehlerbeschreibung versehen. Der dritte und der vierte Fehler trat ebenfalls nach der internen Validierung auf.

Wichtig hierbei ist, dass sich die Datensatz-Nummern (hier 22 und 24) nicht nach den eingetragenen Nummern in der Liste, sondern nach den tatsächlichen laufenden Nummern richten: eingelesen wurden hier lediglich 30 Datensätze.

VAR_INPUT

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

bStt: Eine TRUE-Flanke an diesem Eingang startet den Lesevorgang.

sDataFile: Enthält den Pfad- und Dateinamen der zu öffnenden Daten-Datei. Diese muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein. Wird die Datei mit einem einfachen Text-Editor geöffnet, so müssen die Werte durch Semikola getrennt dargestellt sein. Beispiel für einen Eintrag: *sDataFile:= 'C:\Projekte\FacadeElements.csv'*

sLogFile: dto. Log-File für die auflaufenden Fehler. Diese Datei wird bei jeder neuen Bausteinaktivierung überschrieben, so dass nur aktuelle Fehler enthalten sind.

sNetId: Hier kann ein String mit der AMS Net Id des TwinCAT-Rechners angegeben werden, auf dem die Dateien geschrieben/gelesen werden sollen. Für den lokalen Rechner kann auch ein Leerstring angegeben werden (siehe T_AmsNetId).

> Die Speicherung der Daten kann nur auf dem Steuerungsrechner selbst und den Rechnern erfolgen, die mit dem Steuerungsrechner per ADS verbunden sind. Verweise auf lokale Festplatten dieser Rechner sind möglich, nicht jedoch auf verbundene Netzwerkfestplatten.

VAR_OUTPUT

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

bBusy: Dieser Ausgang steht auf TRUE, solange Elemente aus der Datei ausgelesen werden.

udiAmtSetsRd: Anzahl der gelesenen Datensätze.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn ein Datei-Schreib- bzw. Lesefehler aufgetreten ist.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Dateihandling-Fehler: Öffnen der Logdatei - Es wird die ADS-Fehlernummer angegeben.
02: Dateihandling-Fehler: Öffnen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
03: Dateihandling-Fehler: Lesen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
04: Fehler: Beim Lesen der Datendatei wurde festgestellt, dass die Datei ist zu groß ist (Anzahl Bytes größer als udiMaxDataFileSize)
05: Dateihandling-Fehler: Schreiben in die Logdatei - Es wird die ADS-Fehlernummer angegeben.
06: Dateihandling-Fehler: Schließen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
07: Dateihandling-Fehler: Schreiben in die Logdatei (ok-Meldung, falls keine Fehler erkannt wurden) - Es wird die ADS-Fehlernummer angegeben.
08: Dateihandling-Fehler: Schließen der Logdatei - Es wird die ADS-Fehlernummer angegeben.

bErrDataSet: Dieser Ausgang wird auf TRUE geschaltet, wenn die gelesenen Datensätze fehlerhaft sind. Nähere Einzelheiten werden in die Log-Datei geschrieben.

VAR_IN_OUT

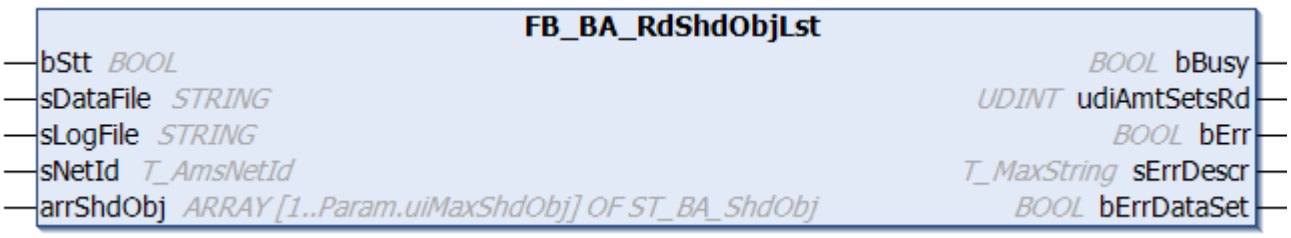
```
arrFcdElem : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: Liste von Fassadenelementen [▶ 572].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.13 FB_BA_RdShdObjLst



Mit Hilfe dieses Bausteines lassen sich Daten für Verschattungsobjekte aus einer vordefinierten Excel-Tabelle im csv-Format in die Liste der Verschattungsobjekte [▶ 572] importieren. Zusätzlich werden die importierten Daten auf Plausibilität überprüft und Fehler in eine Log-Datei geschrieben.

Das folgende Beispiel zeigt die Excel-Tabelle mit den Einträgen der Fensterelemente.
Alle Textfelder sind frei beschreibbar. Wichtig sind die grün markierten Felder, jede Zeile dort kennzeichnet einen Datensatz. Je nachdem ob es sich um den Typ Viereck oder Kugel handelt, haben die Spalten G bis J eine unterschiedliche Bedeutung. Die Spalten K bis M sind bei Kugeln freizulassen. Bezüglich der Vierecks-Koordinaten werden nur die relevanten Daten eingegeben und die restlichen intern berechnet, siehe FB_BA_ShdObjEntry [▶ 599].
Folgende Regeln sind zu beachten:

- Ein Datensatz muss immer mit einem '@' beginnen.
- Die Monatseinträge dürfen nicht 0 und größer 12 sein, alle anderen Kombinationen sind möglich.
  Beispiele:
  Beginn=1, Ende=1: Verschattung im Januar.
  Beginn=1, Ende=5: Verschattung von Anfang Januar bis Ende Mai.
  Beginn=11, Ende=5: Verschattung von Anfang November bis Ende Mai (des folgenden Jahres).
- Fensterbreite und Fensterhöhe müssen größer Null sein
- Die z-Koordinaten P1z und P3z bzw. Mz müssen größer Null sein.
- Der Radius muss größer Null sein.
- Die Gesamtgröße der Tabelle darf systembedingt 65534 Bytes nicht überschreiten.
- Die Tabelle muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein.

Es ist nicht notwendig, alle Verschattungsobjekte, welche pro Fassade möglich sind, zu beschreiben. Nur die, die in der Liste aufgeführt werden, kommen letztendlich zum Tragen.

DE_ShadingObjects.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Nummer	Beschreibung		Typ	Beginn	Ende	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Viereck	(Monat)	(Monat)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Kugel									
4		Text											
5	1	Beschreibung	@	0	1	2	-94,75	0	36,06	11	-70,71	11	68,59
6	2	Beschreibung	@	0	1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Beschreibung	@	0	1	2	62,23	0	0	14,47	62,23	14,47	8
8	4	Beschreibung	@	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Beschreibung	@	0	1	2	46	0	13	14,47	46	14,47	38,89
10	6	Beschreibung	@	0	1	2	0	0	14	9	35	9	14
11	7	Beschreibung	@	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Beschreibung	@	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Beschreibung	@	0	1	2	27,8	0	14	9,8	35	9,8	14
14													
15	10	Beschreibung	@	1	1	2	27	15	40	6			
16	11	Beschreibung	@	1	1	2	38	15	36	6			
17	12	Beschreibung	@	1	1	2	-14	4	4	1,5			
18	13	Beschreibung	@	1	1	2	-6,5	6	6	3,2			
19	14	Beschreibung	@	1	1	2	-7	9	6	1,2			
20	15	Beschreibung	@	1	1	2	-1	6	8	3,2			
21	16	Beschreibung	@	1	1	2	-1	9	8	1,2			

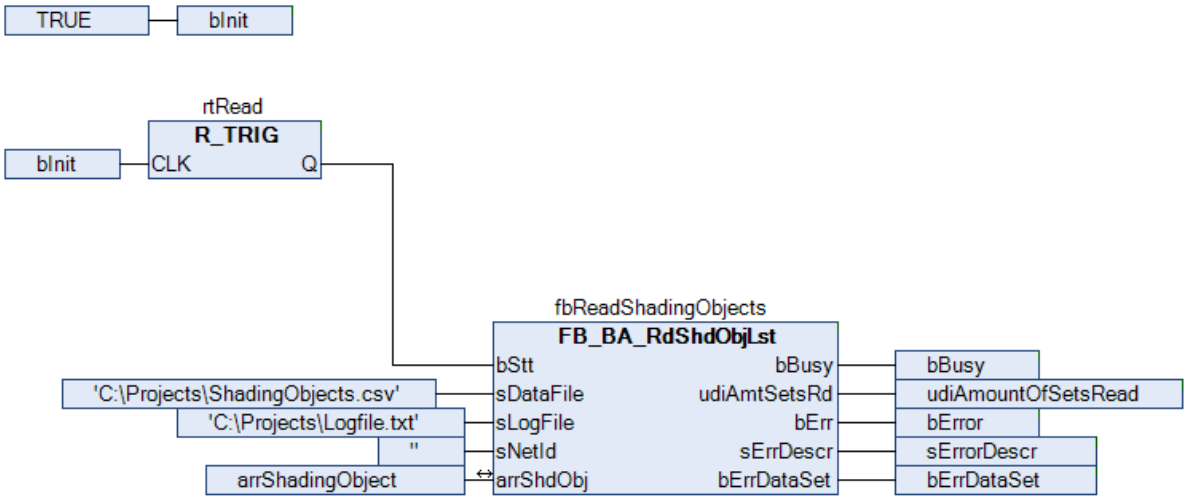
Log-Datei

Bei jedem Start des Lesebausteines wird die Logdatei neu beschrieben und der alte Inhalt gelöscht. Ist die Log-Datei nicht vorhanden, so wird sie zunächst automatisch erstellt. Die Log-Datei enthält dann entweder eine OK Meldung oder eine Auflistung aller aufgetretenen Fehler. Fehler, welche mit dem Öffnen, Beschreiben oder Schließen der Logdatei selbst in Zusammenhang stehen, können nicht mitgeschrieben werden. Daher ist immer auch der Ausgang *sErrDescr* des Lesebausteines zu beachten, der den letzten Fehlercode anzeigt. Da im Ablauf des Lesens die Log-Datei immer als Letztes geschlossen wird, ist ein entsprechender Hinweis im Fehlerfall gewährleistet.

Programmbeispiel

```
PROGRAM ReadShadingObjects
VAR
    bInit                   :   BOOL;
    rtRead                  :   R_TRIG;
    fbReadShadingObjects    :   FB_BA_RdShdObjLst;
    arrShadingObject        :   ARRAY [1..uiMaxShdObj] OF ST_BA_ShdObj;

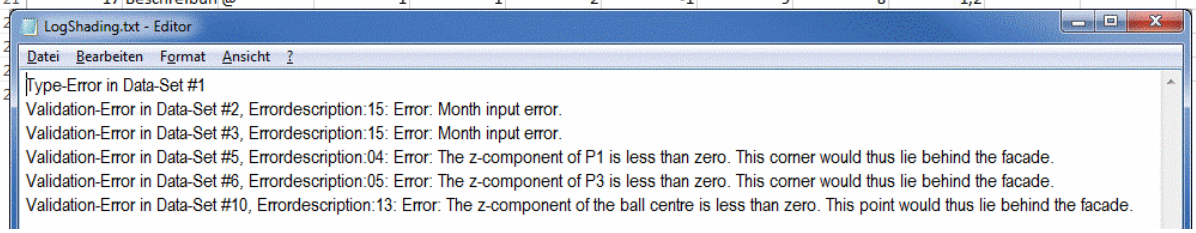
    bBusy                   :   BOOL;
    udiAmountOfSetsRead     :   UDINT;
    bError                  :   BOOL;
    sErrorDescr             :   T_MaxString;
    bErrDataSet             :   BOOL;
END_VAR
```



In diesem Beispiel wird bei PLC-Start die Variable *bInit* zunächst auf TRUE gesetzt. Damit erhält der Eingang *bStt* am Baustein *fbReadShadingObjects* einmalig eine steigende Flanke, welche den Lesevorgang auslöst. Gelesen wird die Datei "ShadingObjects.csv", welche sich im Ordner "C:\Projekte\" befindet. In demselben Ordner wird dann die Logdatei "Logfile.txt" hinterlegt. Ist diese Logdatei noch nicht vorhanden wird sie erstellt, andernfalls wird der bestehende Inhalt überschrieben. Das Lesen und Schreiben erfolgt auf demselben Rechner, auf dem sich auch die PLC befindet. Das wird durch den Eingang *sNetID* = '' (=lokal) definiert. Alle Daten werden in die im Programm deklarierte Liste *arrShdObj* geschrieben. Während gelesen und geschrieben wird ist der Ausgang *bBusy* auf TRUE. Der zuletzt auftretende Fehler in Bezug auf das Dateihandling wird an *sErrDescr* angezeigt, *bErr* steht dann auf TRUE. Wird hingegen ein Fehler im Datensatz erkannt, wird dies an *bErrDataSet* angezeigt und in der Log-Datei näher beschrieben. Die Anzahl der gefundenen und gelesenen Datenzeilen wird an *udiAmtSetsRd* zur Kontrolle angezeigt.

In die folgende Excel Liste wurden die markierten Fehler eingebaut. Dadurch ergibt sich das gezeigte Log-File:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Nummer	Beschreibung		Typ	Beginn	Ende	P1x/Mx	P1y/My	P1z/Mz	P2y/R	P3x	P3y	P3z
2				0 - Viereck	(Monat)	(Monat)	[m]	[m]	[m]	[m]	[m]	[m]	[m]
3				1 - Kugel									
4		Text											
5	1	Beschreibun	@	2	1	2	-94,75	-4	36,06	11	-70,71	11	68,59
6	2	Beschreibun	@	0	-1	2	-23,33	0	9,9	10,5	-3,54	10,5	22,62
7	3	Beschreibun	@	0	1	13	62,23	0	0	14,47	62,23	14,47	8
8	4	Beschreibun	@	0	1	2	46	0	13	14,47	62,23	14,47	8
9	5	Beschreibun	@	0	1	2	46	0	-13	14,47	46	14,47	38,89
10	6	Beschreibun	@	0	1	2	0	0	14	9	35	9	-14
11	7	Beschreibun	@	0	1	2	0	0	14	9,8	16	9,8	14
12	8	Beschreibun	@	0	1	2	23,6	0	14	9,8	25	9,8	14
13	9	Beschreibun	@	0	1	2	27,8	0	14	9,8	35	9,8	14
14													
15	11	Beschreibun	@	1	1	2	27	15	-40	6			
16	12	Beschreibun	@	1	1	2	38	15	36	6			
17	13	Beschreibun	@	1	1	2	-14	4	4	1,5			
18	14	Beschreibun	@	1	1	2	-6,5	6	6	3,2			
19	15	Beschreibun	@	1	1	2	-7	9	6	1,2			
20	16	Beschreibun	@	1	1	2	-1	6	8	3,2			
21	17	Beschreibun	@	1	1	2	-1	9	8	1,2			



Der erste Fehler ist im Datensatz 3 und ein Typ-Fehler, da "2" nicht definiert ist.
Der nächste Fehler in Datensatz 6 wurde nach Validierung der Daten mit dem intern genutzten Baustein FB_BA_ShdObjEntry [▶ 599] gefunden und daher näher mit einer Fehlerbeschreibung versehen. Der dritte Fehler trat ebenfalls nach der internen Validierung auf.

Wichtig hierbei ist, dass sich die Datensatz-Nummer (hier 11) nicht nach der eingetragenen Nummer in der Liste, sondern nach der tatsächlichen laufenden Nummer richtet: eingelesen wurden hier lediglich 16 Datensätze.

VAR_INPUT

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

bStt: Eine TRUE-Flanke an diesem Eingang startet den Lesevorgang.

sDataFile: Enthält den Pfad- und Dateinamen der zu öffnenden Daten-Datei. Diese muss in Excel als Dateityp "CSV (Trennzeichen-getrennt) (*.csv)" abgespeichert worden sein. Wird die Datei mit einem einfachen Text-Editor geöffnet, so müssen die Werte durch Semikola getrennt dargestellt sein. Beispiel für einen Eintrag: *sDataFile:= 'C:\Projects\ShadingObjects.csv'*

sLogFile: dto. Log-File für die auflaufenden Fehler. Diese Datei wird bei jeder neuen Bausteinaktivierung überschrieben, so dass nur aktuelle Fehler enthalten sind.

sNetId: Hier kann ein String mit der AMS Net Id des TwinCAT-Rechners angegeben werden, auf dem die Dateien geschrieben/gelesen werden sollen. Für den lokalen Rechner kann auch ein Leerstring angegeben werden (siehe T_AmsNetId).

Die Speicherung der Daten kann nur auf dem Steuerungsrechner selbst und den Rechnern erfolgen, die mit dem Steuerungsrechner per ADS verbunden sind. Verweise auf lokale Festplatten dieser Rechner sind möglich, nicht jedoch auf verbundene Netzwerkfestplatten.

VAR_OUTPUT

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

bBusy: Dieser Ausgang steht auf TRUE, solange Elemente aus der Datei ausgelesen werden.

udiAmtSetsRd: Anzahl der gelesenen Datensätze.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn ein Datei-Schreib- bzw. Lesefehler aufgetreten ist.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Dateihandling-Fehler: Öffnen der Logdatei - Es wird die ADS-Fehlernummer angegeben.
02: Dateihandling-Fehler: Öffnen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
03: Dateihandling-Fehler: Lesen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
04: Fehler: Beim Lesen der Datendatei wurde festgestellt, dass die Datei ist zu groß ist (Anzahl Bytes größer als udiMaxDataFileSize)
05: Dateihandling-Fehler: Schreiben in die Logdatei - Es wird die ADS-Fehlernummer angegeben.
06: Dateihandling-Fehler: Schließen der Datendatei - Es wird die ADS-Fehlernummer angegeben.
07: Dateihandling-Fehler: Schreiben in die Logdatei (ok-Meldung, falls keine Fehler erkannt wurden) - Es wird die ADS-Fehlernummer angegeben.
08: Dateihandling-Fehler: Schließen der Logdatei - Es wird die ADS-Fehlernummer angegeben.

bErrDataSet: Dieser Ausgang wird auf TRUE geschaltet, wenn die gelesenen Datensätze fehlerhaft sind. Nähere Einzelheiten werden in die Log-Datei geschrieben.

VAR_IN_OUT

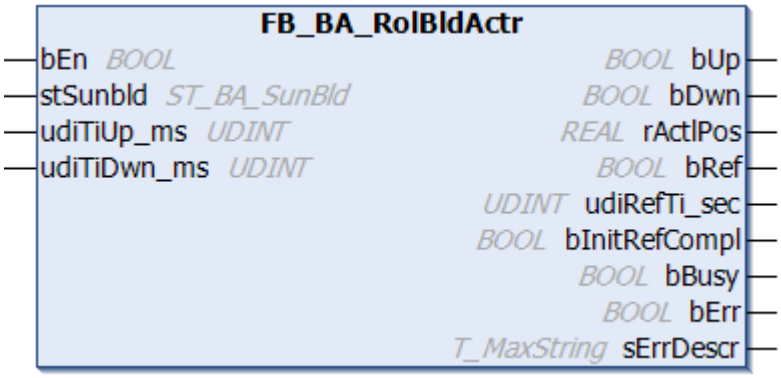
```
arrShdObj: ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;
```

arrShdObj: Liste der Verschattungsobjekte [▶ 572].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.14 FB_BA_RolBldActr



Dieser Baustein dient zur Positionierung eines Rollladens über zwei Ausgänge: Hoch- und Herunterfahren. Über das Positioniertelegramm stSunBld [▶ 702] kann der Rollladen auf eine beliebige Position gesteuert werden. Darüber hinaus beinhaltet das Positioniertelegramm stSunBld [▶ 702] auch Handbefehle, mit denen der Rollladen individuell auf bestimmte Positionen bewegt werden kann. Diese Handbefehle werden von dem Baustein FB_BA_SunBldSwi [▶ 616] angesteuert.

Struktur des Jalousie-Positioniertelegramms stSunBld [▶ 702].

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

Die aktuelle Höhenposition wird dabei nicht durch einen zusätzlichen Encoder eingelesen, sondern intern durch die Laufzeit des Rollladens ermittelt.
Durch die beiden unterschiedlichen Laufzeitparameter *udiTiUp_ms* (Laufzeit Rolllade hoch [ms]) und *udiTiDwn_ms* (Laufzeit Rollladen herunter [ms]) wird den unterschiedlichen Fahrcharakteristiken Rechnung getragen.

Der Baustein steuert den Rollladen grundsätzlich über die Informationen aus dem Positioniertelegramm stSunBld [▶ 702]. Ist der Automatikmodus aktiv (*bManMod*=FALSE), so wird immer die aktuelle Position angefahren, wobei Änderungen sofort berücksichtigt werden. Im Handbetrieb (*bManMod*=TRUE) steuern die Befehle *bManUp* und *bManDwn* den Rollladen.

Referenzieren

Ein sicheres Referenzieren ist gegeben, wenn der Rollladen länger als seine komplette Hochlaufzeit nach oben hin angesteuert wird. Die Position ist dann auf jeden Fall "0". Da eine Rollladenpositionierung ohne Encoder naturgemäß immer fehlerbehaftet ist, ist es wichtig möglichst oft automatisch zu referenzieren: jedes Mal, wenn die Position "0" angefahren werden soll, fährt der Rollladen zunächst ganz normal mit kontinuierlicher Positionsberechnung nach oben. Erreicht sie den errechneten Positionswert 0%, so wird der Ausgang *bUp* noch einmal für die weiterhin gehalten und zwar noch einmal für die komplette Hochlaufzeit + 5s.
Aus Gründen der Flexibilität gibt es nun zwei Möglichkeiten, den Referenziervorgang zu unterbrechen: Bis zum Erreichen der errechneten 0%-Position wird eine Positionsänderung immer noch angenommen und ausgeführt, nach Erreichen dieser 0%-Position kann der Rollladen noch mit dem Handbefehl "herunterfahren" anders bewegt werden. Diese beiden sinnvollen Einschränkungen machen es nötig, dass der Nutzer selbst dafür Sorge trägt, der Rollladen so oft wie möglich sicher referenzieren zu lassen.

Beim Neustart der Anlage referenziert der Baustein in jedem Fall. Die abgeschlossene Initial-Referenzierung wird durch ein TRUE-Signal an Ausgang *bInitRefCmpl* angezeigt. Auch die Initial-Referenzierung kann durch einen Handbefehl "herunterfahren" vorzeitig abgeschlossen werden.

VAR_INPUT

```
bEn           : BOOL;
stSunBld      : ST_BA_Sunblind;
udiTiUp_ms    : UDINT;
udiTiDwn_ms   : UDINT;
```

bEn: Freigabeeingang für den Baustein. Solange dieser Eingang auf TRUE steht, nimmt der Aktorbaustein Befehle entgegen und arbeitet, wie oben beschrieben. Ein FALSE-Signal an diesen Eingang setzt die Steuerausgänge *bUp* und *bDwn* zurück und der Funktionsbaustein verharrt in einem Ruhezustand.

stSunBld: Positioniertelegramm, siehe ST_BA_SunBld [▶ 702].

udiTiUp_ms: Komplette Hochfahrzeit [ms].

udiTiDwn_ms: Komplette Zeit zum Herunterfahren in ms.

```
bUp             : BOOL;
bDwn            : BOOL;
rActlPos        : REAL;
bRef            : BOOL;
udiRefTi_sec    : UDINT;
bInitRefCompl   : BOOL;
bBusy           : BOOL;
bErr            : BOOL;
sErrDescr       : T_MAXSTRING;
```

bUp: Steuerausgang Rollladen hoch.

bDwn: Steuerausgang Rollladen herunter.

rActlPos: Aktuelle Position in Prozent.

bRef: Die Rollladen befindet sich in der Referenzierung, d.h. für die die komplette Hochlaufzeit + 5s wird der Ausgang *bUp* gesetzt. Nur ein Handbefehl "herunter" kann den Rollladen in Gegenrichtung bewegen und diesen Modus beenden.

udiRefTi_sec: Referenzier-Countdown-Anzeige [s].

bInitRefCompl: Initial-Referenziervorgang abgeschlossen.

bBusy: Ein Positionier- oder Referenziervorgang findet statt.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

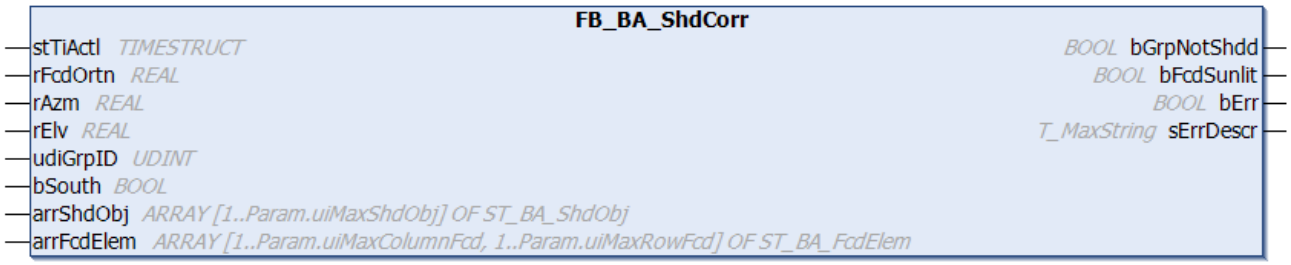
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Die Gesamt-Hoch- oder Herunterfahrzeit (udiTiUp_ms/udiTiDwn_ms) ist Null.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.15 FB_BA_ShdCorr



Der Funktionsbaustein dient zur Verschattungsbeurteilung einer Fenstergruppe auf einer Fassade.

Der Baustein FB_BA_ShdCorr berechnet für eine Gruppe von Fenstern, ob sich diese im Schatten von umliegenden Objekten befindet. Mit dem Ergebnis, welches am Ausgang *bGrpNotShdd* ausgegeben wird, kann beurteilt werden, ob ein Sonnenschutz für diese Fenstergruppe sinnvoll ist.
Dabei greift der Baustein auf zwei zu definierende Listen zu:

- Die Parameter, welche die Verschattungselemente beschreiben, die für die Fassade maßgeblich sind, auf der sich die Fenstergruppe befindet. Diese Liste der Verschattungsobjekte [▶ 572] wird als Eingangsvariable *arrShdObj* an den Baustein angelegt, da die Informationen lediglich gelesen werden.
- Die Daten der Elemente (Fenster) der Fassade, in der sich die zu betrachtende Gruppe befindet. Auf diese Liste der Fassadenelemente [▶ 572] wird über die IN-OUT-Variable *arrFcdElem* zugegriffen, da nicht nur die Koordinaten der Fenster gelesen werden, sondern von dem Baustein FB_BA_ShdCorr in diese Liste auch jeweils die Verschattungsinformation für jede Fensterecke hinterlegt wird. Diese Information kann so in weiteren Teilen des Applikationsprogrammes genutzt werden.

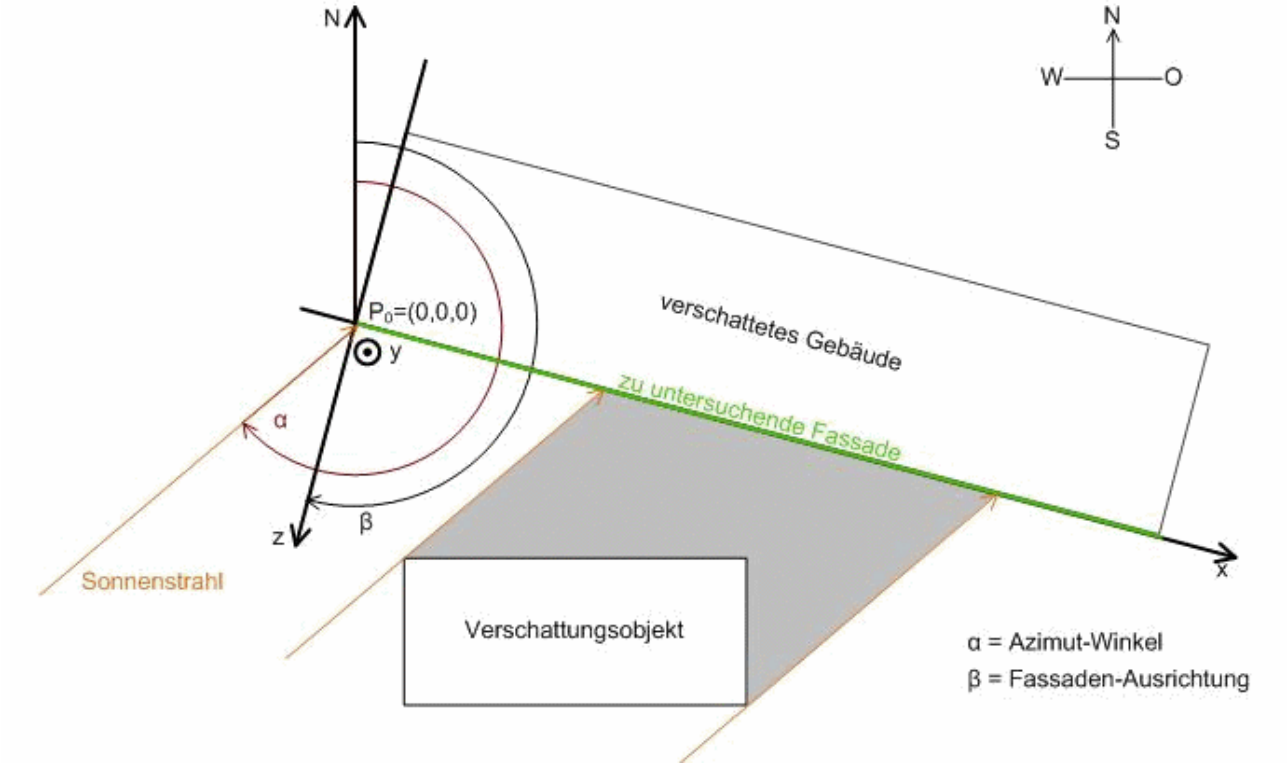
Anhand der Fassadenausrichtung (*rFcdOrtn*), der Sonnenrichtung (*rAzm*) und der Sonnenhöhe (*rElv*) kann dann für jeden Eckpunkt eines Fensters errechnet werden, ob sich dieser im Bereich eines Schattens befindet. Eine Gruppe von Fenstern gilt dann als komplett verschattet, wenn alle Eckpunkte verschattet sind.

Dabei gilt auf der Nordhalbkugel für die Fassadenausrichtung (Blick aus dem Fenster):

Blickrichtung	Fassadenausrichtung
Nord	β=0°
Ost	β=90°
Süd	β=180°
West	β=270°

Der Baustein führt seine Berechnungen nur dann durch, wenn die Sonne tatsächlich auf die Fassade scheint. Wird die in der Einleitung vorgestellte Zeichnung betrachtet, so ist dies gegeben, wenn gilt:

Fassadenausrichtung < Azimutwinkel < Fassadenausrichtung + 180°

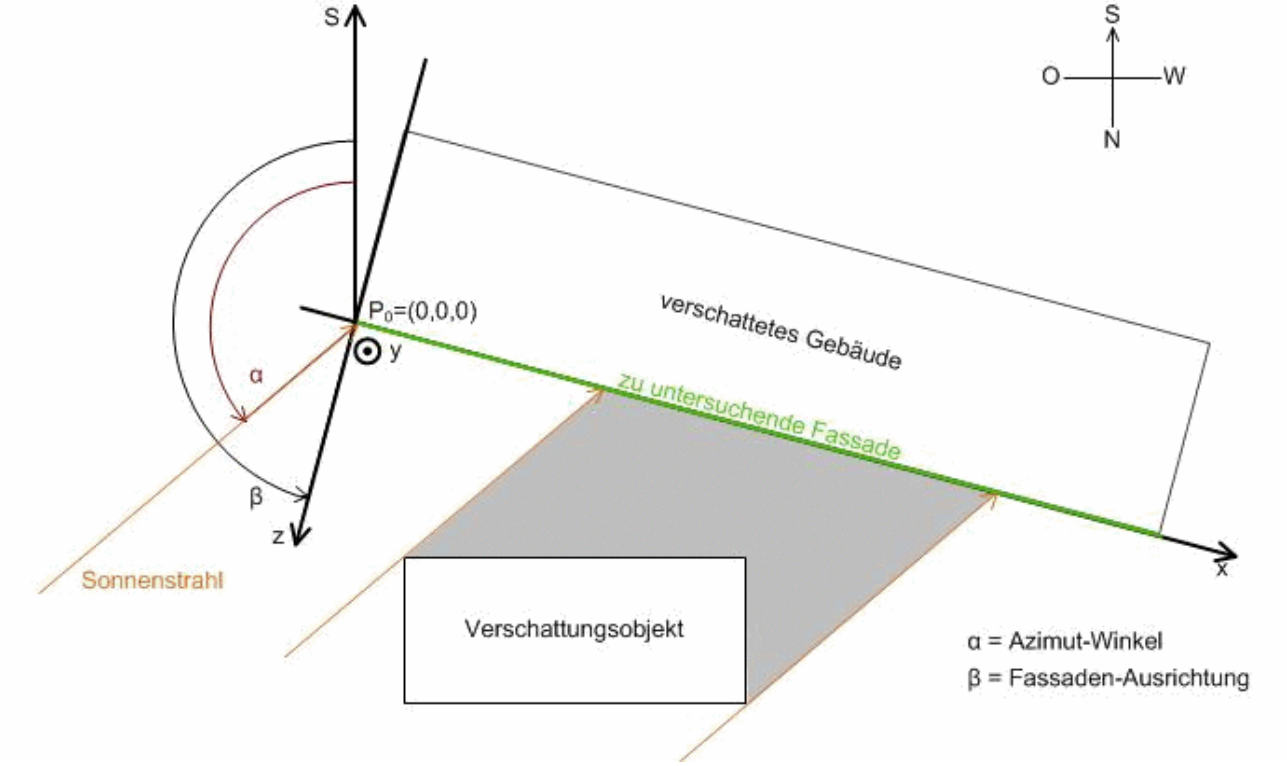


Darüber hinaus ist eine Berechnung auch dann nicht nötig, wenn die Sonne noch nicht aufgegangen ist, die Sonnenhöhe (Elevation) also unter 0° liegt. In beiden Fällen wird der Ausgang *bFcdSunlit* auf FALSE gesetzt.

Für die Südhalbkugel ändern sich die Verhältnisse. Es gilt für die Fassadenausrichtung (Blick aus dem Fenster):

Blickrichtung	Fassadenausrichtung
Süd	β=0°
Ost	β=90°
Nord	β=180°
West	β=270°

Auch ändert sich die interne Berechnung Fassade zu Sonnenstrahl:



Zur Unterscheidung der Verhältnisse - Nord- oder Südhalbkugel - ist der Eingangsparameter *bSouth* auf FALSE zu setzen (Nordhalbkugel) oder auf TRUE (Südhalbkugel)

VAR_INPUT

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

stTiActl: Eingabe der aktuellen Uhrzeit - hier die Ortszeit, da mithilfe dieser Zeit die verschattenden Monate berücksichtigt werden. Bei Verwendung der UTC-Zeit (bzw. GMT) kann, je nach Lage auf der Erde, der Monat mitten am Tag "umspringen" (siehe TIMESTRUCT).

rFcdOrtn: Fassadenausrichtung, siehe Abbildung oben.

rAzm: Sonnenrichtung zum Betrachtungszeitpunkt [°].

rElv: Sonnenhöhe zum Betrachtungszeitpunkt [°].

diGrpId: Betrachtete Fenstergruppe. Die Gruppe 0 ist hierbei für nicht verwendete Fensterelemente reserviert, siehe FB_BA_FcdElemEntry [▶ 576]. Ein 0-Eintrag würde zu einer Fehlerausgabe führen (bErr=TRUE). Der Baustein wird dann nicht weiter abgearbeitet und *bGrpNotShdd* auf FALSE gesetzt.

bSouth: FALSE: Berechnungen beziehen sich auf Verhältnisse der Nordhalbkugel - TRUE: auf Südhalbkugel

arrShdObj: Liste der Verschattungsobjekte [▶ 572].

VAR_OUTPUT

```
bGrpNotShdd : BOOL;
bFcdSunlit  : BOOL;
bErr        : BOOL;
sErrDescr   : T_MAXSTRING
```

bGrpNotShdd: Steht auf TRUE, solange die Fenstergruppe als nicht verschattet errechnet wird.

bFcdSunlit: Dieser Ausgang ist dann auf TRUE gesetzt, wenn die Sonne auf die Fassade scheint. Siehe Beschreibung oben.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Der Index der zu betrachtenden Fenstergruppe udiGrpID ist 0.
02: Fehler: Ein Element der Fassadenliste ist ungültig. Dieses wird in der Fehlerbeschreibung sErrDescr als arrFcdElem[nColumn,nRow] näher angegeben.

VAR_IN_OUT

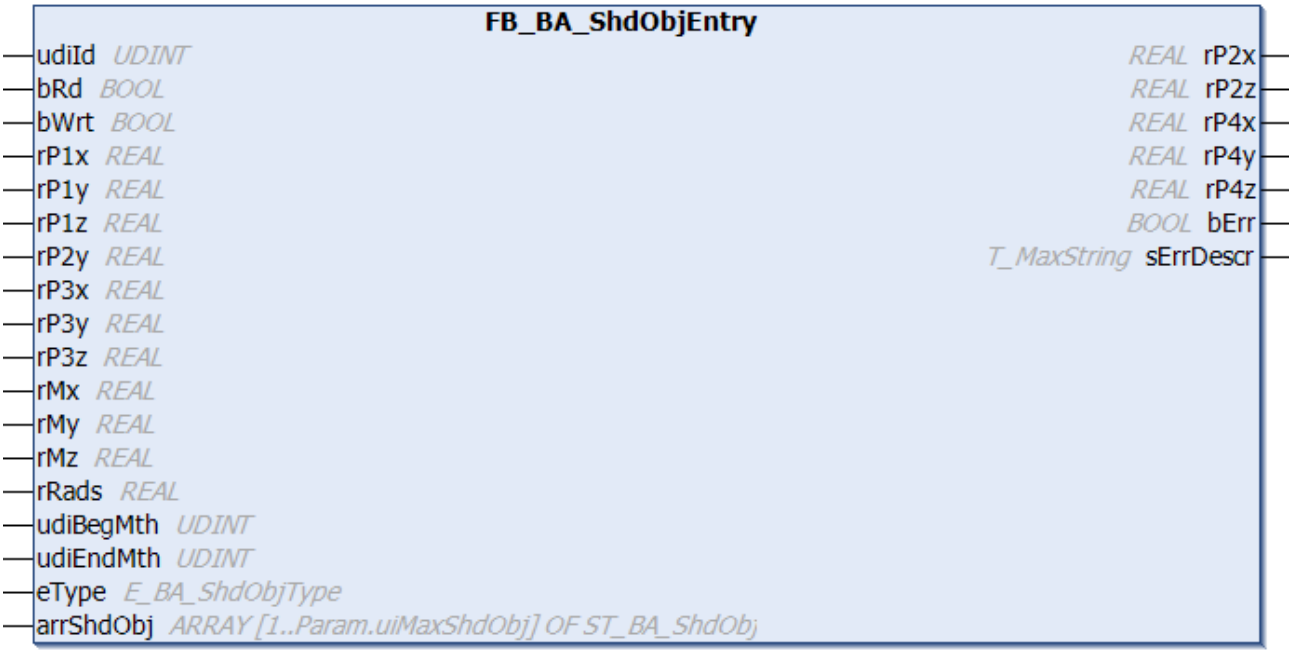
```
arrFcdElem     : ARRAY[1..Param.uiMaxColumnFcd, 1..Param.uiMaxRowFcd] OF ST_BA_FcdElem;
```

arrFcdElem: Liste der Fassadenelemente [▶ 572].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.16 FB_BA_ShdObjEntry



Dieser Baustein dient zur Verwaltung aller Verschattungselemente einer Fassade, welche global in einer Liste von Verschattungselementen [▶ 572] hinterlegt sind. Er soll die Eingabe der Elementinformationen - auch im Hinblick auf die Nutzung einer Visualisierung - erleichtern. Eine schematische Darstellung der Objekte mit Beschreibung der Koordinaten ist unter Verschattungskorrektur: Grundlagen und Definitionen [▶ 558] gegeben.

Die Deklaration der Verschattungselemente erfolgt in den globalen Variablen:

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

Jedes einzelne Element *arrShdObj[1]* bis *arrShdObj [uiMaxShdObj]* trägt die Informationen für jeweils ein Verschattungselement (ST_BA_ShdObj [▶ 701]). Dies sind neben dem gewählten Verschattungstyp (Viereck oder Kugel) die jeweils dazugehörigen Koordinaten. Im Falle eines Vierecks sind es die Eckpunkte *(rP1x*, *rP1y*, *rP1z), (rP2x*, *rP2y*, *rP2z), (rP3x*, *rP3y*, *rP3z)* und *(rP4x*, *rP4y*, *rP4z)* und im Falle einer Kugel der

Mittelpunkt *(rMx*, *rMy*, *rMz)* und der Radius *rRads*. Zusätzlich kann über die Eingänge *udiBegMth* und *udiEndMth* die Phase der Verschattung definiert werden, was bei Objekten, wie z.B. Bäume, die im Winter kein Laub tragen, wichtig ist.

Der Baustein greift dabei über die IN-OUT-Variable *arrShdObj* direkt auf das Feld dieser Informationen zu.

Anmerkung**:** Die Tatsache, dass die Vierecks Koordinaten *rP2x*, *rP2z*, *rP4x*, *rP4y*, und *rP4z* Ausgangswerte sind, ergibt sich daraus, dass sie aus den Eingabeparametern gebildet werden:

rP2x = rP1x; *rP2z = rP1z*; *rP4x = rP3x*; *rP4y = rP1y*; *rP4z = rP3z*;

Das schränkt die Eingabe eines Vierecks so weit ein, dass die seitlichen Kanten senkrecht auf dem Boden stehen (*rP2x = rP1x* und *rP4x = rP3x*), dass das Viereck keine Neigung hat (*rP2z = rP1z* und *rP4z = rP3z*) und nur nach "oben" also in positive y-Richtung eine unterschiedliche Höhe haben kann (*rP4y = rP1y*).

Die Verwendung des Bausteines erfolgt in drei Schritten:

- Auslesen
- Ändern
- Schreiben

Auslesen

Mit dem Eintrag an *udiId* wird das entsprechende Element aus der Liste, *arrShdObj[iId]*, ausgewählt. Eine steigende Flanke an *bRd* liest die Daten aus. Diese Werte werden den Ein- und Ausgangsvariablen des Bausteines zugewiesen. Es handelt sich hierbei um die Eingangswerte *rP1x*, *rP1y*, *rP1z*, *rP2y*, *rP3x*, *rP3y*, *rP3z*, *rMx*, *rMy*, *rMz*, r*Rads* und dem Objekt-Enumerator *eType* und um die Ausgangswerte *rP2x*, *rP2z*, *rP4x*, *rP4y*, und *rP4z.* Wichtig ist hierbei, dass im Schritt des Auslesens die Eingangswerte nicht überschrieben werden. So lassen sich alle Werte zunächst in einer Visualisierung anzeigen.

Ändern

In einem nächsten Programmschritt können dann die aufgeführten Eingabewerte verändert werden. Wenn am Eingang eType [▶ 698] über den Wert *"eObjectTypeTetragon"* die Verwendung eines Vierecks vorgewählt ist, ergeben sich die Ausgangswerte *rP2x, rP2z, rP4x, rP4y*, und *rP4z* aus den eingegebenen Koordinaten des Vierecks, siehe oben.

Die eingegebenen Werte werden ständig auf Plausibilität überprüft. Der Ausgang *bErr* zeigt an, ob die Werte gültig sind (*bErr*=FALSE). Wenn der Wert ungültig ist, wird am Ausgang *sErrDescr* eine entsprechende Fehlermeldung ausgegeben.
Wird ein Viereck definiert, müssen lediglich die Eingänge *rP1x*, *rP1y*, *rP1z*, *rP2y*, *rP3x*, *rP3y* und *rP3z* beschrieben werden, die Eingänge *rMx*, *rMy*, *rMz* und *rRads* brauchen nicht verknüpft zu werden. Im Falle einer Kugeldefinition müssen nur *rMx*, *rMy*, *rMz* und *rRads* beschrieben werden und die Viereckkoordinaten können unverknüpft bleiben

Schreiben

Mit einer positiven Flanke an *bWrt* werden die parametrierten Daten in das Listenelement mit dem Index *udiId* geschrieben, unabhängig davon, ob sie gültige Werte darstellen oder nicht. Daher ist innerhalb der Elementstruktur ST_BA_ShdObj [▶ 701] ein Plausibilitätsbit *bVld* vorhanden, das genau diese Information an den Baustein FB_BA_ShdCorr [▶ 596] weiterreicht und dort Fehlberechnungen vorbeugt.

Diese Vorgehensweise soll nur als Vorschlag angesehen werden. Es ist natürlich auch möglich, den Baustein ganz normal in einem Schritt zu parametrieren und die eingetragenen Werte mit einer steigenden Flanke an *bWrt* in das entsprechende Listenelement zu schreiben.

VAR_INPUT

```
udiId       : UDINT;
bRd         : BOOL;
bWrt        : BOOL;
rP1x        : REAL;
rP1y        : REAL;
rP1z        : REAL;
rP2y        : REAL;
rP3x        : REAL;
```

```
rP3y        : REAL;
rP3z        : REAL;
rMx         : REAL;
rMy         : REAL;
rMz         : REAL;
rRads       : REAL;
udiBegMth   : UDINT;
udiEndMth   : UDINT;
eType       : E_BA_ShdObjType;
```

udiId: Index des gewählten Elementes. Dies bezieht sich auf die Auswahl eines Feldelementes des unter der IN-OUT-Variablen *arrShdObj* angelegten Arrays. Die Variable *udiId* **darf nicht Null sein!** Das ergibt sich aus der Felddefinition, welche mit 1 beginnt. Eine fehlerhafte Eingabe wird jedoch erkannt und als solche an *bErr/sErrDescr* angezeigt.

bRd: Mit einer positiven Flanke an diesem Eingang werden die Informationen des gewählten Elementes, *arrShdObj[udiId]* in den Baustein gelesen und den Eingangsvariablen *rP1x* bis *eType* sowie den Ausgangsvariablen *rP2x* bis *rP4z* zugewiesen. Sind zu diesem Zeitpunkt schon Daten an den Eingängen *rP1x* bis *eType* angelegt, so werden die zuvor ausgelesenen Daten sofort mit diesen überschrieben.

bWrt: Eine positive Flanke schreibt die an den Eingängen *rP1x* bis *eType* angelegten sowie die ermittelten und den Ausgängen *rP2x* bis *rP4z* zugewiesenen Werte in das gewählte Feldelement *arrShdObj[udiId].*

rP1x: X-Koordinate des Punktes 1 des Verschattungselementes (Viereck) [m].

rP1y: Y-Koordinate des Punktes 1 des Verschattungselementes (Viereck) [m].

rP1z: Z-Koordinate des Punktes 1 des Verschattungselementes (Viereck) [m].

rP2y: Y-Koordinate des Punktes 2 des Verschattungselementes (Viereck) [m].

rP3x: X-Koordinate des Punktes 3 des Verschattungselementes (Viereck) [m].

rP3y: Y-Koordinate des Punktes 3 des Verschattungselementes (Viereck) [m].

rP3z: Z-Koordinate des Punktes 3 des Verschattungselementes (Viereck) [m].

rMx: X-Koordinate des Mittelpunktes des Verschattungselementes (Kugel) [m].

rMy: Y-Koordinate des Mittelpunktes des Verschattungselementes (Kugel) [m].

rMz: Z-Koordinate des Mittelpunktes des Verschattungselementes (Kugel) [m].

rRads: Radius des Verschattungselementes (Kugel) [m].

udiBegMth: Anfang der Verschattungsperiode (Monatszahl).

udiEndMth: Ende der Verschattungsperiode (Monatszahl).

eType: Gewählter Elementtyp: Viereck oder Kugel (Siehe E_BA_ShdObjType [▶ 698]).

Bemerkung zur Verschattungsperiode:

Die Monatseinträge dürfen nicht 0 und größer 12 sein, andernfalls sind alle Kombinationen möglich.
Beispiele:
Beginn=1, Ende=1: Verschattung im Januar.
Beginn=1, Ende=5: Verschattung von Anfang Januar bis Ende Mai.
Beginn=11, Ende=5: Verschattung von Anfang November bis Ende Mai (des folgenden Jahres).

VAR_OUTPUT

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

rP2x: Ermittelte X-Koordinate des Punktes 2 des Verschattungselementes (Viereck) [m]. Siehe "Anmerkung [▶ 599]" oben.

rP2z: Ermittelte Z-Koordinate des Punktes 2 des Verschattungselementes (Viereck) [m]. Siehe "Anmerkung [▶ 599]" oben.

rP4x: Ermittelte X-Koordinate des Punktes 4 des Verschattungselementes (Viereck) [m]. Siehe "Anmerkung [▶ 599]" oben.

rP4y: Ermittelte Y-Koordinate des Punktes 4 des Verschattungselementes (Viereck) [m]. Siehe "Anmerkung [▶ 599]" oben.

rP4z: Ermittelte Z-Koordinate des Punktes 4 des Verschattungselementes (Viereck) [m]. Siehe "Anmerkung [▶ 599]" oben.

bErr: Ergebnis der Plausibilitätskontrolle für die eingegebenen Werte. In Bezug auf ein Viereck gilt, dass der Innenwinkel 360° beträgt und die Punkte in einer Ebene liegen und *vor* der betrachteten Fassade liegen. Bei einer Kugel muss der Mittelpunkt ebenfalls vor der Fassade liegen und der Radius muss größer Null sein.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Der Eingang *udiId* liegt außerhalb der zulässigen Grenzen 1..*uiMaxShdObj*.
02 Fehler: Die Summe der Winkel des Vierecks ist nicht 360°. Das bedeutet, dass die Eckpunkte nicht in der Reihenfolge P1, P2, P3 und P4 sondern P1, P3, P2 und P4 sind. Dies ergibt ein über Kreuz geschlagenes Viereck.
03: Fehler: Die Eckpunkte des Vierecks liegen nicht auf derselben Ebene.
04: Fehler: Die z-Komponente von P1 ist kleiner Null. Damit läge dieser Eckpunkt hinter der Fassade.
05: Fehler: Die z-Komponente von P3 ist kleiner Null. Damit läge dieser Eckpunkt hinter der Fassade.
06: Fehler: P1 ist gleich P2. Damit ist das eingetragene Objekt kein Viereck.
07: Fehler: P1 ist gleich P3. Damit ist das eingetragene Objekt kein Viereck.
08: Fehler: P1 ist gleich P4. Damit ist das eingetragene Objekt kein Viereck.
09: Fehler: P2 ist gleich P3. Damit ist das eingetragene Objekt kein Viereck.
10: Fehler: P2 ist gleich P4. Damit ist das eingetragene Objekt kein Viereck.
11: Fehler: P3 ist gleich P4. Damit ist das eingetragene Objekt kein Viereck.
12: Fehler: Der eingetragene Radius ist gleich Null.
13: Fehler: Die z-Komponente des Kugelmittelpunktes ist kleiner Null. Damit läge dieser Punkt hinter der Fassade.
14: Fehler: Fehler Objekttyp *eType* - weder Viereck noch Kugel.
15: Fehler: Fehler Monatseingabe.

VAR_IN_OUT

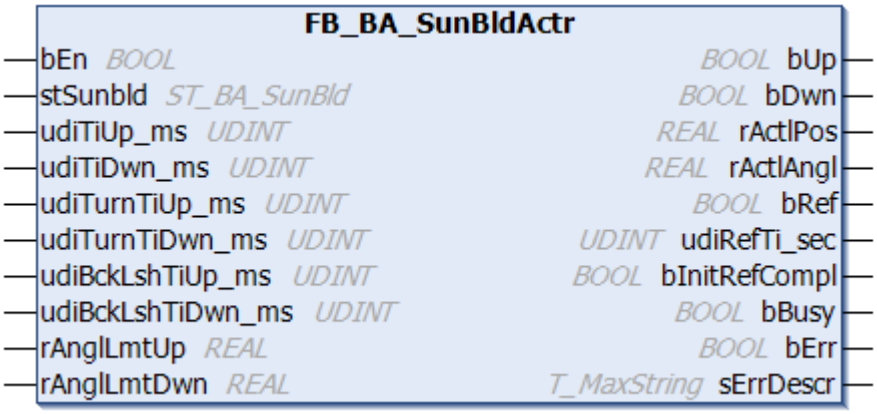
```
arrShdObj : ARRAY[1..Param.uiMaxShdObj] OF ST_BA_ShdObj;
```

arrShdObj: Liste der Verschattungsobjekte [▶ 572].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.17 FB_BA_SunBldActr



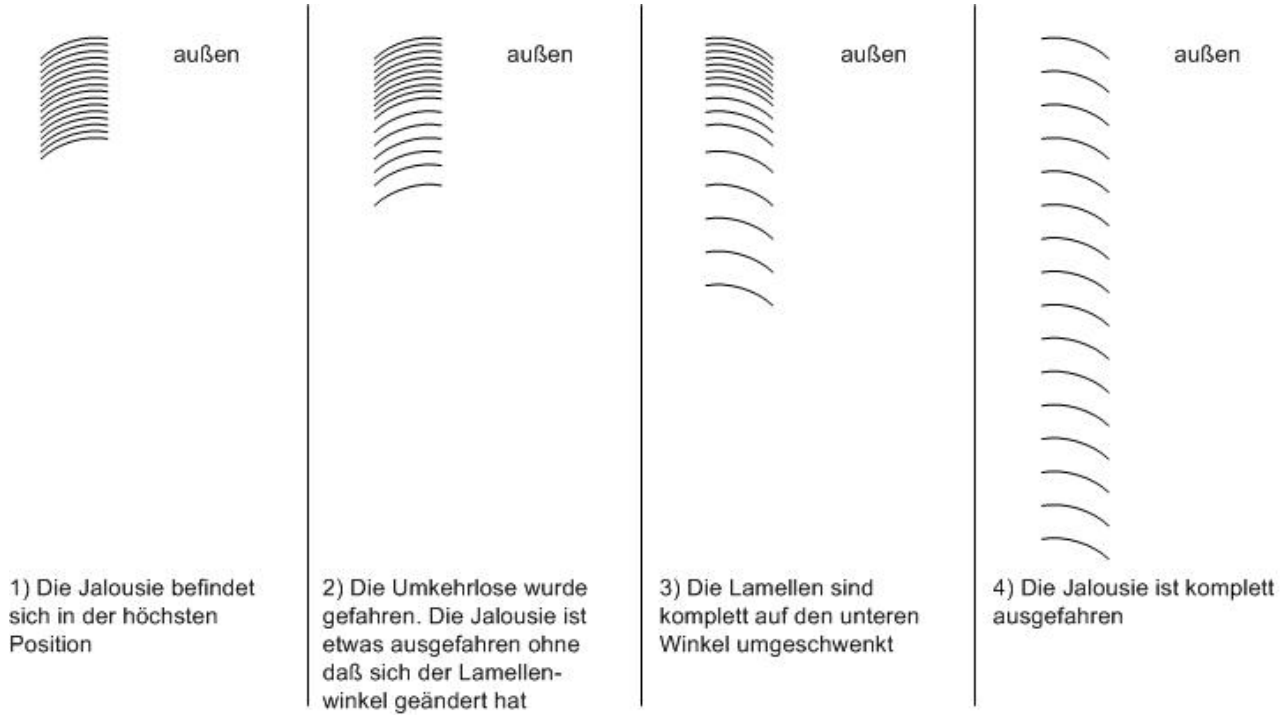
Dieser Baustein dient zur Positionierung einer Lamellen-Jalousie über zwei Ausgänge: hoch- und herunterfahren. Über das Positioniertelegramm stSunBld [▶ 702] kann die Jalousie auf eine beliebige (Höhen-) Position und einen Lamellenwinkel gefahren werden. Darüber hinaus beinhaltet das Positioniertelegramm stSunBld [▶ 702] auch Handbefehle, mit denen die Jalousie individuell auf bestimmte Stellungen bewegt werden kann. Diese Handbefehle werden von dem Baustein FB_BA_SunBldSwi [▶ 616] angesteuert.

Struktur des Jalousie-Positioniertelegramms stSunBld [▶ 702].

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

Die aktuelle Höhenposition und der Lamellenwinkel werden dabei nicht durch einen zusätzlichen Encoder eingelesen, sondern intern durch die Laufzeit der Jalousie ermittelt. Der Berechnung liegt folgendes Fahrprofil zugrunde (von der höchsten und niedrigsten Position der Jalousie aus betrachtet):

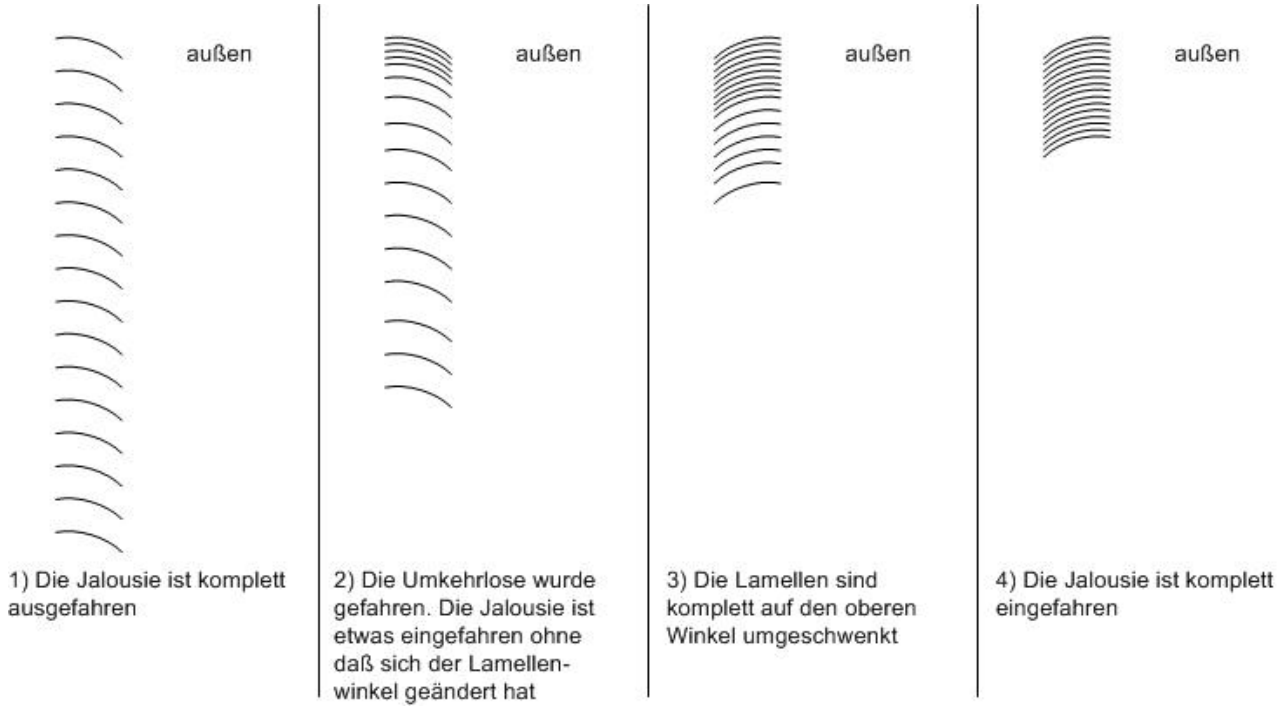
Fahrprofil abwärts:



Nähere Erläuterung zu den Begriffen "Umkehrlose" und "Umschwenken", hier in der Abwärtsbewegung:

Die Jalousie beschreibt ihre Abwärtsbewegung normalerweise mit dem Lamellentiefpunkt nach außen gerichtet, wie in Bild 3).
Befindet sich die Jalousie in einer Ausgangsstellung mit dem Tiefpunkt nach innen (d.h. nach Abschluss einer Aufwärtsbewegung), so vergeht bei einer erneuten Abwärtsfahrt eine gewisse Zeit bis die Lamellen vom "Tiefpunkt innen" bis zu "Tiefpunkt außen" zu schwenken beginnen. Während dieser Zeit ändert sich der Lamellenwinkel nicht, die Jalousie fährt nur herunter (Bild1) und Bild 2)). Diese Zeit ist ein wichtiger Parameter zur Bewegungsberechnung und wird am Baustein unter *udiBckLshTiDwn_ms* [ms] eingetragen. Da an einem x-beliebigen Punkt nach einer beliebig langen Jalousiebewegung nicht belannt ist, ob schon etwas von der Umkehrlosen ausgefahren wurde, ist die Umkehrlose der Abwärtsbewegung bzw. deren Ausfahrzeit am sichersten zu messen, wenn die Jalousie zunächst ganz nach oben gefahren wurde. Ein weiterer wichtiger Parameter ist die Zeitspanne des darauffolgenden Umschwenkens der Lamellen von "Tiefpunkt innen" bis zu "Tiefpunkt außen". Diese Zeit ist als *udiTurnTiDwn_ms* [ms] am Baustein einzutragen.

Fahrprofil aufwärts:



Nähere Erläuterung zu den Begriffen "Umkehrlose" und "Umschwenken", hier in der Aufwärtsbewegung:

Dieser Sachverhalt ist analog zur oben beschriebenen Abwärtsbewegung: Die Jalousie beschreibt ihre Aufwärtsbewegung normalerweise mit dem Lamellentiefpunkt nach innen gerichtet, wie in Bild 3). Befindet sich die Jalousie in einer Ausgangsstellung mit dem Tiefpunkt nach außen (d.h. nach Abschluss einer Abwärtsbewegung), so vergeht bei einer erneuten Aufwärtsfahrt eine gewisse Zeit bis die Lamellen vom "Tiefpunkt außen" bis zu "Tiefpunkt innen" zu schwenken beginnen. Während dieser Zeit ändert sich der Lamellenwinkel nicht, die Jalousie fährt nur herauf (Bild1) und Bild 2)). Auch diese Zeit ist ein wichtiger Parameter zur Bewegungsberechnung und wird am Baustein unter *udiBckLshTiUp_ms* [ms] eingetragen. Da an einem x-beliebigen Punkt nach einer beliebig langen Jalousiebewegung nicht bekannt ist, ob schon etwas von der Umkehrlosen ausgefahren wurde, ist die Umkehrlose der Aufwärtsbewegung bzw. deren Ausfahrzeit am sichersten zu messen, wenn die Jalousie zunächst ganz nach unten gefahren wurde. Ein weiterer wichtiger Parameter ist die Zeitspanne des darauffolgenden Umschwenkens der Lamellen von "Tiefpunkt außen" bis zu "Tiefpunkt innen". Diese Zeit ist als *udiTurnTiUp_ms* [ms] am Baustein einzutragen.

Parametrierung

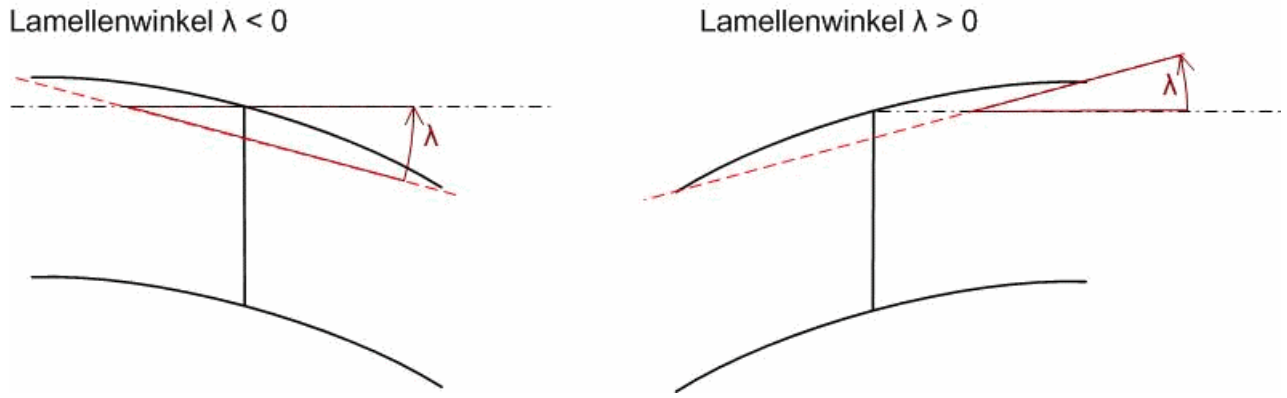
Zur Berechnung der (Höhen-)Position und des Lamellenwinkels sind nun jeweils für die Auf- und Abwärtsbewegung folgende Zeiten zu ermitteln:

- die Verfahrdauer der Umkehrlose (*udiBckLshTiUp_ms* / *udiBckLshTiDwn_ms* [ms])
- die Verfahrdauer des Umschwenkens (*udiTurnTiUp_ms* / *udiTurnTiDwn_ms* [ms])
- die Gesamt-Verfahrdauer (*udiTiUp_ms* / *udiTiDwn_ms* [ms])

Des Weiteren sind zur Berechnung erforderlich:

- der höchste Lamellenwinkel nach einem Umschwenken nach oben (*rAnglLmtUp* [°])
- der niedrigste Lamellenwinkel nach einem Umschwenken nach unten (*rAnglLmtDwn* [°])

Der Lamellenwinkel λ ist dabei durch eine gedachte Gerade durch die Endpunkte der Lamelle zur Horizontalen definiert.



Funktionsweise

Der Baustein steuert die Jalousie grundsätzlich über die Informationen, aus dem Positioniertelegramm stSunBld [▶ 702]. Ist der Automatikmodus aktiv (*bManMod*=FALSE), so wird immer die aktuelle Position und Lamellenwinkel angefahren, wobei Änderungen sofort berücksichtigt werden. Die Positionierung auf die Höhe hat dabei Vorrang: Es wird zunächst die eingegebene Höhe und danach der Lamellenwinkel angefahren. Aus Gründen der Einfachheit bleibt dabei der Positionsfehler durch das Winkel-Verfahren unberücksichtigt. Im Handbetrieb (*bManMod*=TRUE) steuern die Befehle *bManUp* und *bManDwn* die Jalousie.
Bei einem Wechsel vom Hand- in den Automatikmodus wird in jedem Fall ein Automatik-Fahrbefehl ausgelöst.

Referenzieren

Ein sicheres Referenzieren ist gegeben, wenn die Jalousie länger als ihre komplette Hochlaufzeit nach oben hin angesteuert wird. Die Position ist dann auf jeden Fall "0" und der Lamellenwinkel auf seinem Maximum. Da eine Jalousiepositionierung ohne Encoder naturgemäß immer fehlerbehaftet ist, ist es wichtig möglichst oft automatisch zu referenzieren: jedes Mal, wenn die Position "0" angefahren werden soll (der Winkel spielt dabei keine Rolle) fährt die Jalousie zunächst ganz normal mit kontinuierlicher Positionsberechnung nach oben. Erreicht sie den errechneten Positionswert 0 %, so wird der Ausgang *bUp* gehalten und zwar noch einmal für die komplette Hochlaufzeit + 5 s.
Aus Gründen der Flexibilität gibt es nun zwei Möglichkeiten, den Referenziervorgang zu unterbrechen: Bis zum Erreichen der errechneten 0 %-Position wird eine Positionsänderung immer noch angenommen und ausgeführt, nach Erreichen dieser 0 %-Position kann die Jalousie noch mit dem Handbefehl "herunterfahren" anders bewegt werden. Diese beiden sinnvollen Einschränkungen machen es nötig, dass der Nutzer selbst dafür Sorge trägt, die Jalousie so oft wie möglich sicher referenzieren zu lassen.

Beim Neustart der Anlage referenziert der Baustein in jedem Fall. Die abgeschlossene Initial-Referenzierung wird durch ein TRUE-Signal an Ausgang *bInitRefCmpl* angezeigt. Auch die Initial-Referenzierung kann durch einen Handbefehl "herunterfahren" vorzeitig abgeschlossen werden.

Zielgenauigkeit

Da der Baustein die Position der Jalousie alleine über Laufzeiten ermittelt, spielt die Zykluszeit der SPS-Task eine entscheidende Rolle bei der Positioniergenauigkeit. Besteht beispielsweise eine Umschwenkzeit von 1 s für einen Lamellen-Winkelbereich von - 70° bis 10°, so liegt bei einer Zykluszeit von 50 ms die Genauigkeit bei +/- 4°.

VAR_INPUT

```
bEn                : BOOL;
stSunbld           : ST_BA_SunBld;
udiTiUp_ms         : UDINT;
udiTiDwn_ms        : UDINT;
udiTurnTiUp_ms     : UDINT;
udiTurnTiDwn_ms    : UDINT;
udiBckLshTiUp_ms   : UDINT;
udiBckLshTiDwn_ms  : UDINT;
rAnglLmtUp         : REAL;
rAnglLmtDwn        : REAL;
```

bEn: Freigabeeingang für den Baustein. Solange dieser Eingang auf TRUE steht, nimmt der Aktorbaustein Befehle entgegen und arbeitet, wie oben beschrieben. Ein FALSE-Signal an diesen Eingang setzt die Steuerausgänge *bUp* und *bDwn* zurück und der Funktionsbaustein verharrt in einem Ruhezustand.

stSunBld: Positioniertelegramm, (siehe ST_BA_SunBld [▶ 702]).

udiTiUp_ms: Komplette Hochfahrzeit [ms].

udiTiDwn_ms: Komplette Zeit zum Herunterfahren [ms].

udiTurnTiUp_ms: Zeit zum Umschwenken der Lamellen in obere Richtung [ms].

udiTurnTiDwn_ms: Zeit zum Umschwenken der Lamellen in untere Richtung [ms].

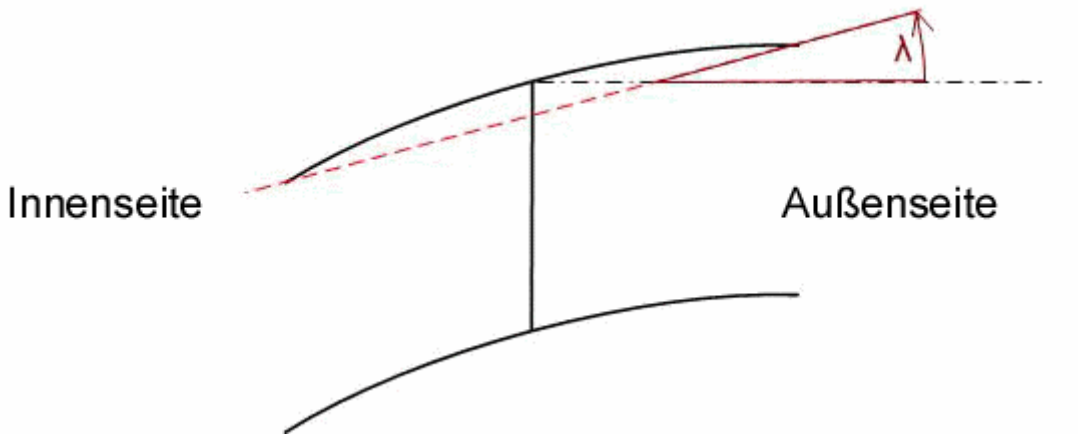
udiBckLshTiUp_ms: Zeit zum Ausfahren der Umkehrlose in obere Richtung [ms]. Diese Eingabe ist intern auf einen Minimalwert von 0 begrenzt.

udiBckLshTiDwn_ms: Zeit zum Ausfahren der Umkehrlose in untere Richtung [ms]. Diese Eingabe ist intern auf einen Minimalwert von 0 begrenzt.

rAnglLmtUp: Höchste Stellung der Lamellen [°].

Diese Stellung ist erreicht, wenn die Jalousie ganz hochgefahren ist.

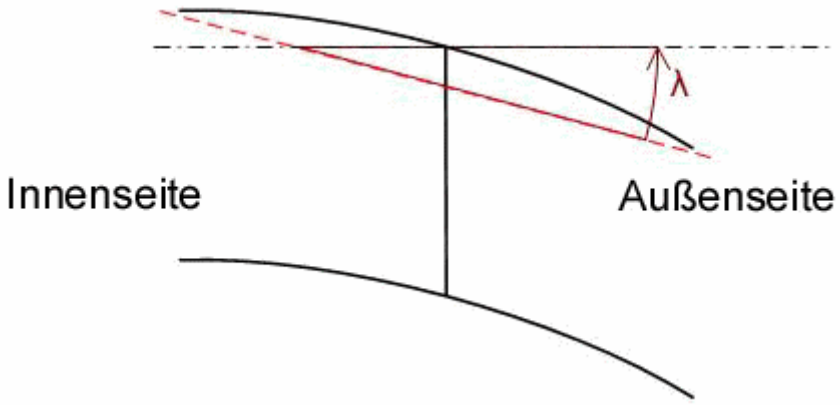
Der Lamellenwinkel λ, so wie er einleitend definiert ist, ist dann typischerweise größer Null.



rAnglLmtDwn: Niedrigste Stellung der Lamellen [°].

Diese Stellung ist erreicht, wenn die Jalousie ganz heruntergefahren ist.

Der Lamellenwinkel λ, so wie er einleitend definiert ist, ist dann typischerweise kleiner Null.



VAR_OUTPUT

```
bUp           : BOOL;
bDwn          : BOOL;
rActlPos      : REAL;
rActlAngl     : REAL;
bRef          : BOOL;
udiRefTi_sec  : UDINT;
bInitRefCompl : BOOL;
bBusy         : BOOL;
bErr          : BOOL;
sErrDesc      : T_MAXSTRING;
```

bUp: Steuerausgang Jalousie hoch.

bDwn: Steuerausgang Jalousie herunter.

rActlPos: Aktuelle Position in Prozent.

rActlAngl: Aktueller Lamellenwinkel [°].

bRef: Die Jalousie befindet sich in der Referenzierung, d.h. für die die komplette Hochlaufzeit + 5s wird der Ausgang *bUp* gesetzt. Nur ein Handbefehl "herunter" kann die Jalousie in Gegenrichtung bewegen und diesen Modus beenden.

udiRefTi_sec: Referenzier-Countdown-Anzeige [s].

bInitRefCompl: Initial-Referenziervorgang abgeschlossen.

bBusy: Ein Positionier- oder Referenziervorgang findet statt.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

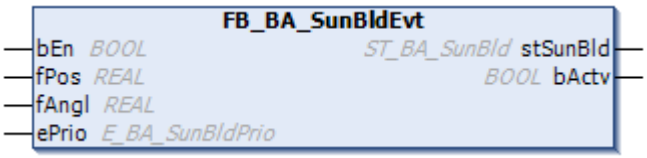
sErrDesc: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Hoch/Herunter-Timer = 0.
02: Fehler: Turning/Wendetimer = 0.
03: Fehler: Lamellenwinkelgrenzen: Die obere Grenze ist kleiner oder gleich der unteren Grenze (*rAnglLmtUp<=rAnglLmtDwn*).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.18 FB_BA_SunBldEvt



Dieser Baustein dient zur Positions- und Winkelvorgabe bei einem beliebigen Ereignis. Sie kann beispielsweise genutzt werden, um eine Parkposition anzufahren oder im Wartungsfall die Jalousie hochfahren zu lassen.

Die Funktion wird über den Eingang *bEn* aktiviert. Ist dies der Fall, so wird der Aktiv-Merker im Positioniertelegramm (*bActv* in *stSunBld*) am Ausgang stSunBld [▶ 702] gesetzt und die an den In-Out-Variablen eingetragenen Werte *rPos* für die Jalousiehöhe [%] und *rAngl* für den Lamellenwinkel [°] in diesem Telegramm weiter gereicht. Ist die Funktion durch Rücksetzen von *bEn* nicht mehr aktiv, so wird der Aktiv-Merker im Positioniertelegramm stSunBld [▶ 702] zurück und die Positionen für Höhe und Winkel auf "0" gesetzt. Mit dem Prioritätenbaustein (z.B. FB_BA_SunBldPrioSwi4 [▶ 612]) kann durch das Rücksetzen eine Funktion niedrigerer Priorität die Steuerung übernehmen.

VAR_INPUT

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

bEn: Ein TRUE-Signal an diesem Eingang aktiviert den Baustein und übergibt die eingetragenen Sollwerte im Positioniertelegramm ST_BA_SunBld [▶ 702] zusammen mit dem Aktivmerker. Ein FALSE-Signal setzt den Aktivmerker wieder zurück, sowie Position und Winkel auf Null.

rPos: Höhenposition der Jalousie [%] im Falle einer Aktivierung.

rAngl: Lamellenwinkel der Jalousie [°] im Falle einer Aktivierung.

VAR_OUTPUT

```
stSunBld    : ST_BA_SunBld;
bActv       : BOOL;
```

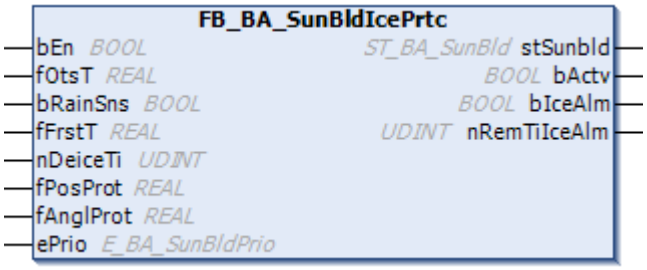
bActv: Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 702] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

stSunBld: Ausgabestruktur der Jalousiestellungen, siehe ST_BA_SunBld [▶ 702]

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.19 FB_BA_SunBldIcePrtc



Der Funktionsbaustein *FB_BA_SunBldIcePrtc* behandelt den richtungsunabhängigen Vereisungsschutz.

Der Witterungsschutz hat bei der Jalousiesteuerung die höchste Priorität (siehe Übersicht [▶ 566]) und soll sicherstellen, dass die Jalousie weder durch Eis, noch durch Wind beschädigt wird.

Eine bevorstehende Vereisung wird dadurch erkannt, dass während einer Niederschlagserkennung an *bRainSns* die gemessene Außentemperatur *rOtsT* unterhalb des Frost-Grenzwertes *rFrstT* liegt. Dieses Ereignis wird intern gespeichert und bleibt dann solange bestehen, bis sichergestellt ist, dass das Eis wieder abgetaut ist. Dazu muss die Außentemperatur den Frost-Grenzwert für die eingetragene Enteisungszeit *udiDeiceTi_sec* [s] überschritten haben. Aus Sicherheitsgründen wird das Vereisungsereignis persistent, also über einen SPS-Ausfall hinweg gespeichert. Fällt die Steuerung also während einer Ver- bzw. Enteisungsperiode aus, so gilt die Jalousie nach Wiederanlauf der Steuerung als neu vereist und der Enteisungszeitmesser startet wieder.

Bei Vereisungsgefahr wird die Jalousie in die Schutzposition gefahren die durch *rPosProt* (Höhenposition in Prozent) und *rAnglProt* (Lamellenwinkel [°]) vorgegeben wird.

VAR_INPUT

```
bEn               : BOOL;
rOtsT             : REAL;
bRainSns          : BOOL;
rFrstT            : REAL;
udiDeiceTi_sec    : UDINT;
rPosProt          : REAL;
rAnglProt         : REAL;
```

bEn: Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_SunBld [▶ 702] werden für Position und Winkel jeweils 0 ausgegeben und *bActv* steht auf FALSE. Das bedeutet, dass über die Prioritätssteuerung eine andere Funktionalität die Jalousieansteuerung übernimmt.

rOtsT: Außentemperatur [°C].

bRainSns: Eingang für einen Niederschlagssensor.

rFrstT: Vereisungstemperatur-Grenzwert [°] Celsius. Dieser Wert darf nicht größer als 0 sein. Anderenfalls wird ein Fehler ausgegeben.

udiDeiceTi_sec: Zeit zum Abtauen der Jalousie nach Vereisung [s]. Danach wird der Vereisungsalarm zurückgesetzt.

rPosProt: Höhenposition der Jalousie [%] im Schutzfall.

rAnglProt: Lamellenwinkel der Jalousie [°] im Schutzfall.

VAR_OUTPUT

```
stSunBld            : ST_BA_SunBld;
bActv               : BOOL;
bIceAlm             : BOOL;
udiRemTiIceAlm_sec  : UDINT;
```

stSunBld: Ausgabestruktur der Jalousiestellungen, siehe ST_BA_SunBld [▶ 702].

bActv: Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 702] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

bIceAlm: Zeigt den Vereisungsalarm an.

udiRemTiIceAlm_sec: Bei aufkommenden Vereisungsfall (*bIceAlm*=TRUE) wird dieser Sekundenzähler auf die Enteisungszeit gesetzt. Sobald die Temperatur über dem eingetragenen Frostpunkt (*rFrstT*) liegt, werden hier die verbleibenden Sekunden bis zur Entwarnung (*bIceAlm*=FALSE) angezeigt. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.



Wenn ein Fehler ansteht, wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.20 FB_BA_SunBldPosDly



Dieser Funktionsbaustein verzögert Positionsänderungen von Automatik-Befehlen.

Wenn durch ein Ereignis, zum Beispiel Wetterschutz, zu viele Jalousieantriebe gleichzeitig gestartet werden, ist es möglich, dass die Summe der hohen Motor-Anzugsströme Sicherungen auslösen. Es ist daher zu empfehlen, die Jalousieantriebe kurz hintereinander zu starten um den Gesamtstrom zu minimieren.

Dieser Baustein gibt Automatikbefehle vom Eingangstelegramm stIn [▶ 702] verzögert an das Ausgangstelegramm stOut [▶ 702] weiter. Er beachtet dazu drei Fälle

1. die Jalousieposition *rPos* hat sich im Automatikmodus verändert (*bManMode*=FALSE im Telegramm *stIn*)

2. der Lamellenwinkel *rAngl* hat sich im Automatikmodus verändert (*bManMode*=FALSE im Telegramm *stIn*)
3. der Hand-Modus wurde gerade verlassen, d.h. der Automatikmodus gerade aktiv (fallende Flanke *bManMode* im Telegramm *stIn*)

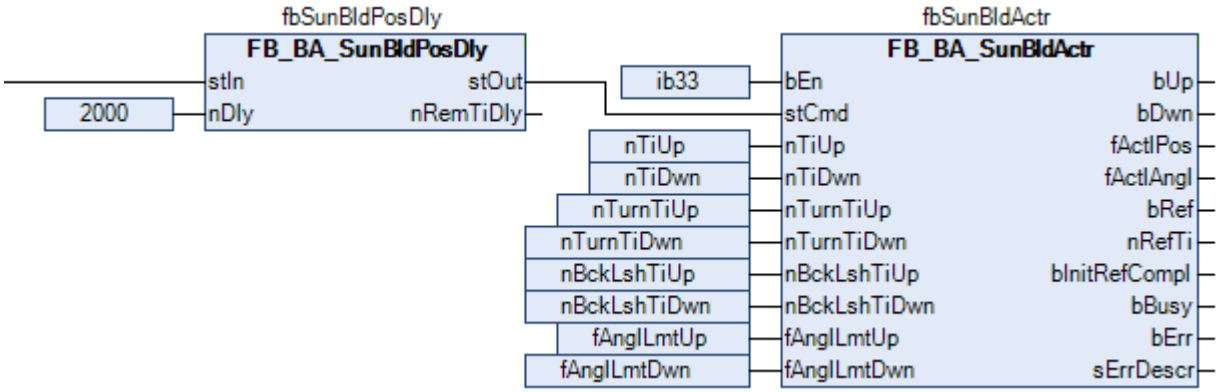
Das Ausgangstelegramm *stOut*, ist immer eine direkte Kopie des Eingangstelegramms *stIn*. In diesen drei Fällen jedoch wird das Ausgangstelegramm *stOut* für die Zeit von *udiDly_ms* [ms] festgesetzt.

Dadurch wird die über den Baustein FB_BA_SunBldActr [▶ 603] angesteuerte Jalousie für den Zeitraum der Verzögerung auf ihrer Position gehalten. Jede weitere Änderung nach den oben genannten Kriterien innerhalb der Verzögerungszeit startet den Zeitgeber neu.

Ein Wechsel auf Hand im Eingangstelegramm (*bManMode* = TRUE) jedoch löscht den Warte-Timer unmittelbar. Das (Hand-)Telegramm wird unverzögert durchgereicht. So werden **nur** Automatiktelegramme verzögert.

Anwendung

Vorzugsweise direkt vor dem Jalousie-Aktorbaustein:



VAR_INPUT

```
stIn        : ST_BA_Sunblind;
udiDly_ms   : UDINT;
```

stIn: Eingangs-Positioniertelegramm, siehe ST_BA_SunBld [▶ 702].

udiDly_ms: Verzögerungszeit des Aktiv-Bits im Positioniertelegramm [ms].

VAR_OUTPUT

```
stOut           : ST_BA_Sunblind;
udiRemTiDly_sec : UDINT;
```

stOut: Ausgangs-Positioniertelegramm, siehe ST_BA_SunBld [▶ 702].

udiRemTiDly_sec: Anzeigeausgang abgelaufene Verzögerungszeit [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.21 FB_BA_SunBldPrioSwi4



Der Funktionsbaustein dient zur Prioritätssteuerung für bis zu 4 Positioniertelegramme (*stSunBld_Prio1* ... *stSunBld_Prio4*) des Typs ST_BA_SunBld [▶ 702].

Struktur des Jalousie-Positioniertelegramms ST_BA_Sunbld [▶ 702].

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

An diesem Baustein lassen sich bis zu 4 Positioniertelegramme verschiedener Steuerbausteine anlegen. Dabei hat das Telegramm an *stSunBld_Prio1* die höchste und das an *stSunBld_Prio4* die niedrigste Priorität. Das aktive Telegramm der höchsten Priorität wird am Ausgang *stSunBld* ausgegeben. "Aktiv" bedeutet, dass innerhalb der Struktur des Positioniertelegramms die Variable *bActv* gesetzt ist.

Dieser Baustein ist so zu programmieren, dass immer eines der angelegten Telegramme aktiv ist. Falls kein Telegramm aktiv ist, wird am Ausgang ein Leertelegramm ausgegeben, d.h. *rPos*=0, *rAngl*=0, *bManUp*=FALSE, *bManDwn*=FALSE, *bManMod*=FALSE, *bActv*=FALSE. Da der Jalousiebaustein FB_BA_SunBldActr [▶ 603] bzw. der Rollladenbaustein FB_BA_RolBldActr [▶ 594] seinerseits nicht auf den Merker *bActv* achtet, würde dieses Telegramm als Fahrbefehl auf Position "0", also vollständig geöffnet, gewertet werden. Das Fehlen eines aktiven Telegramms stellt damit kein Sicherheitsrisiko für die Jalousie dar.

VAR_INPUT

```
stSunBld_Prio1  : ST_BA_SunBld;
stSunBld_Prio2  : ST_BA_SunBld;
stSunBld_Prio3  : ST_BA_SunBld;
stSunBld_Prio4  : ST_BA_SunBld;
```

stSunBld_Prio1..stSunBld_Prio4: Zur Auswahl stehende Positioniertelegramme. Dabei hat *stSunBld_Prio1* die höchste und *stSunBld_Prio4* die niedrigste Priorität.

VAR_OUTPUT

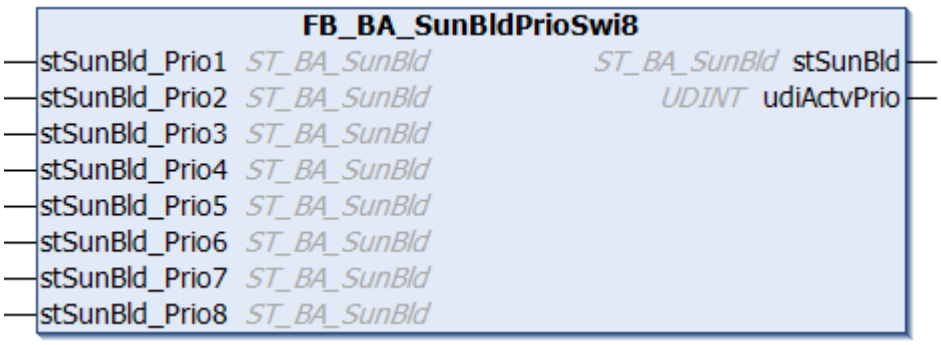
```
stSunBld    : ST_BA_SunBld;
udiActvPrio : UDINT;
```

stSunBld: Resultierendes Positioniertelegramm.
udiActvPrio: Aktives Positioniertelegramm. Ist keines aktiv, so wird "0" ausgegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.22 FB_BA_SunBldPrioSwi8



Der Funktionsbaustein dient zur Prioritätssteuerung für bis zu 8 Positioniertelegramme (*stSunBld_Prio1* ... *stSunBld_Prio8*) des Typs ST_BA_SunBld [▶ 702].

Struktur des Jalousie-Positioniertelegramms ST_BA_Sunbld [▶ 702].

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

An diesem Baustein lassen sich bis zu 8 Positioniertelegramme verschiedener Steuerbausteine anlegen. Dabei hat das Telegramm an *stSunBld_Prio1* die höchste und das an *stSunBld_Prio8* die niedrigste Priorität. Das aktive Telegramm der höchsten Priorität wird am Ausgang *stSunBld* ausgegeben. "Aktiv" bedeutet, dass innerhalb der Struktur des Positioniertelegramms die Variable *bActv* gesetzt ist.

Dieser Baustein ist so zu programmieren, dass immer eines der angelegten Telegramme aktiv ist. Falls ein Telegramm nicht aktiv ist, wird am Ausgang ein Leertelegramm ausgegeben, d.h. *rPos*=0, *rAngl*=0, *bManUp*=FALSE, *bManDwn*=FALSE, *bManMod*=FALSE, *bActv*=FALSE. Da der Jalousiebaustein FB_BA_SunBldActr [▶ 603] bzw. der Rollladenbaustein FB_BA_RolBldActr [▶ 594] seinerseits nicht auf den Merker *bActv* achtet, würde dieses Telegramm als Fahrbefehl auf Position "0", also vollständig geöffnet, gewertet werden. Das Fehlen eines aktiven Telegramms stellt damit kein Sicherheitsrisiko für die Jalousie dar.

VAR_INPUT

```
stSunBld_Prio1  : ST_BA_SunBld;
stSunBld_Prio2  : ST_BA_SunBld;
stSunBld_Prio3  : ST_BA_SunBld;
stSunBld_Prio4  : ST_BA_SunBld;
stSunBld_Prio5  : ST_BA_SunBld;
stSunBld_Prio6  : ST_BA_SunBld;
stSunBld_Prio7  : ST_BA_SunBld;
stSunBld_Prio8  : ST_BA_SunBld;
```

stSunBld_Prio1..stSunBld_Prio8: Zur Auswahl stehende Positioniertelegramme. Dabei hat *stSunBld_Prio1* die höchste und *stSunBld_Prio8* die niedrigste Priorität.

VAR_OUTPUT

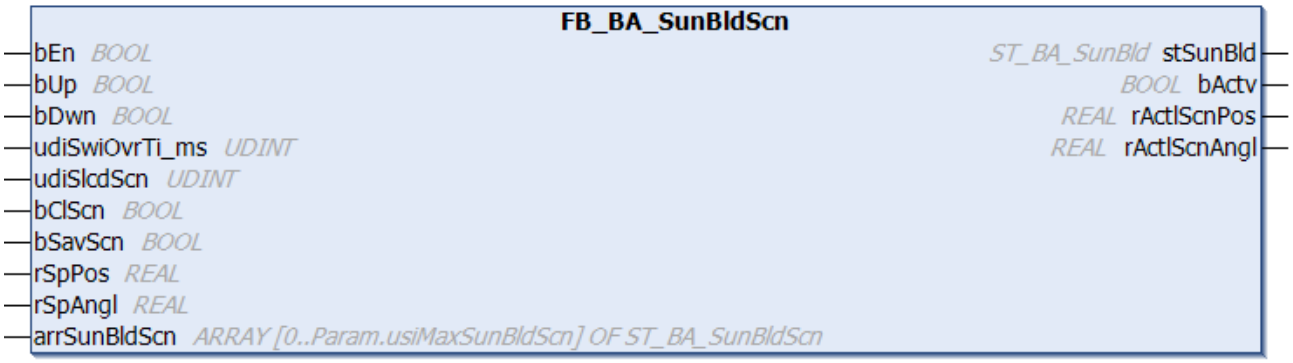
```
stSunBld    : ST_BA_SunBld;
udiActvPrio : UDINT;
```

stSunBld: Resultierendes Positioniertelegramm.
udiActvPrio: Aktives Positioniertelegramm. Ist keines aktiv, so wird "0" ausgegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.23 FB_BA_SunBldScn



Dieser Baustein stellt eine Erweiterung der Handbedienung FB_BA_SunBldSwi [▶ 616] um eine Szenen-Speicher- und Aufruf-Funktionalität dar. Damit lässt sich die Jalousieansteuerung FB_BA_SunBldActr [▶ 603] bzw. die Rollladenansteuerung FB_BA_RolBldActr [▶ 594] sowohl im Handbedienmodus ansteuern, als auch zuvor gespeicherte Positionen (Szenen) direkt anfahren. Es können bis zu 21 Szenen gespeichert werden.

Struktur des Jalousie-Positioniertelegramms ST_BA_Sunbld [▶ 702].

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

Betrieb

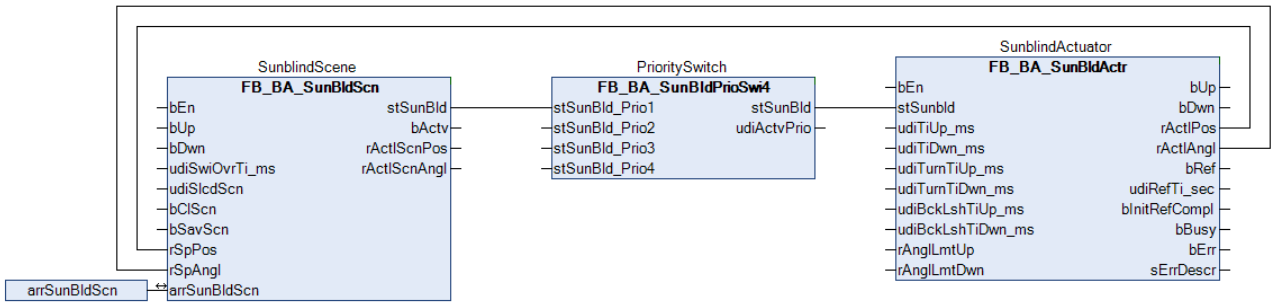
Der Funktionsbaustein steuert über die Befehlseingänge *bUp* und *bDwn* den Jalousiebaustein FB_BA_SunBldActr [▶ 603] bzw. den Rollladenbaustein FB_BA_RolBldActr [▶ 594] im Handbetrieb an, wobei *bUp* Vorrang hat. Die Befehle werden an die jeweiligen Kommandos *bManUp* und *bManDwn* des Positioniertelegramms weitergereicht. Ist ein Befehlseingang länger als die eingetragene Zeit *udiSwiOvrTi_ms* [ms] aktiviert, so geht der entsprechende Steuerbefehl in Selbsthaltung. Ein erneutes Aktivieren eines Befehlseinganges löscht diese Selbsthaltung wieder.
Eine steigende Flanke an *bSavScn* speichert die aktuelle Position und den Lamellenwinkel in die unter *udiSlcdScn* angewählte Szene. Dieser Vorgang ist jederzeit möglich, auch während einer aktiven Positionierung. Mit *bClScn* wird die angewählte Szene aufgerufen, das heißt, die gespeicherten Werte von Position und Winkel angefahren.
Wird der Baustein durch den Eingang *bEn*=TRUE aktiviert, so wird im Positioniertelegramm das Bit *bActv* sofort gesetzt. Damit meldet der Baustein an einem Prioritätsschalter (FB_BA_SunBldPrioSwi4 [▶ 612] bzw. FB_BA_SunBldPrioSwi8 [▶ 613]) seinen Vorrang gegenüber niedrigeren Prioritäten an. Steht nicht der Befehl „Call Scene“ an (*bClScn*=TRUE), so wird auch das Bit *bManMod* im Positioniertelegramm gesetzt, um den angeschossenen Aktor Bausteinen zu melden, dass diese auf Handbefehle reagieren sollen.

Bei Deaktivierung des Bausteines durch *bEn*=FALSE werden beide Bits, *bActv* und *bManMod* wieder auf FALSE gesetzt.

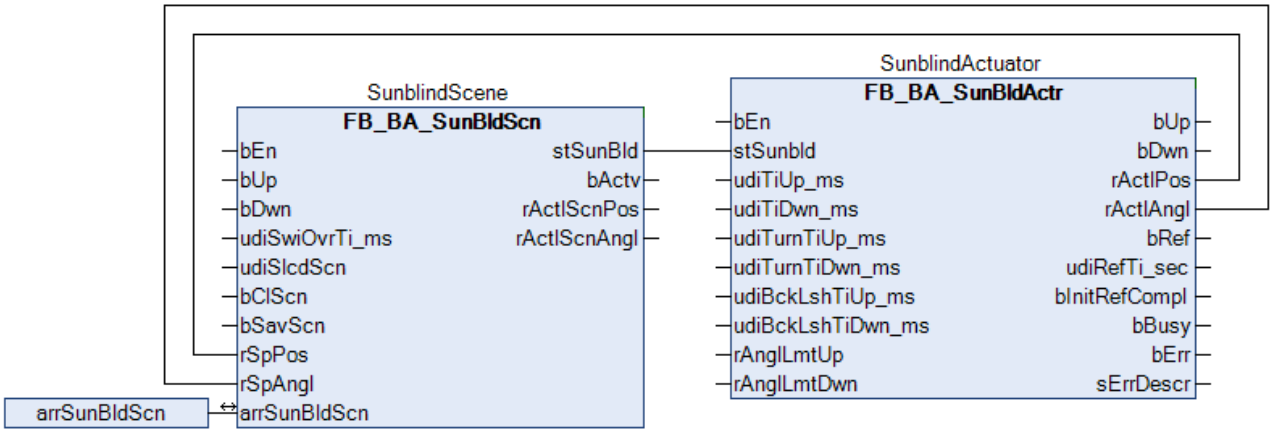
Verknüpfung an Jalousiebaustein

Der Szenenanwahlbaustein kann, wie der "normale" Handbedienbaustein FB_BA_SunBldSwi [▶ 616], entweder über eine voran gestellte Prioritätssteuerung FB_BA_SunBldPrioSwi4 [▶ 612] bzw. FB_BA_SunBldPrioSwi8 [▶ 613] oder aber direkt an den Jalousiebaustein angeschlossen werden. Die Verbindung erfolgt dabei über das Positioniertelegramm ST_BA_Sunbld [▶ 702]. Des Weiteren benötigt der Szenenbaustein die aktuelle Position aus dem Jalousiebaustein der Referenz-Jalousie:

Verwendung einer Prioritätssteuerung:



Direkte Beschaltung:



VAR_INPUT

```
bEn             : BOOL;
bUp             : BOOL;
bDwn            : BOOL;
udiSwiOvrTi_ms  : UDINT;
udiSlcdScn      : UDINT;
bClScn          : BOOL;
bSavScn         : BOOL;
rSpPos          : REAL;
rSpAngl         : REAL;
```

bEn: Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_Sunbld [▶ 702] werden für Position und Winkel jeweils 0 ausgegeben - *bManMod* und *bActv* stehen jeweils auf FALSE. Das bedeutet für eine Beschaltung mit Prioritätssteuerung, dass eine andere Funktionalität die Jalousieansteuerung übernimmt. Eine direkte Beschaltung hingegen lässt die Jalousie direkt auf Position 0, also ganz nach oben fahren, da der Aktorbaustein das Bit *bActv* selbst nicht auswertet.

bUp: Befehlseingang Jalousie hoch.

bDwn: Befehlseingang Jalousie herunter.

udiSwiOvrTi_ms: Zeit [ms] bis bei dauerhaft aktiviertem Befehlseingang der entsprechende Handbefehl im Positioniertelegramm ST_BA_Sunbld [▶ 702] in Selbsthaltung geht. Intern begrenzt auf einen Minimalwert von 0.

udiSlcdScn: Angewählte Szene, welche entweder gespeichert (*bSavScn*) oder aufgerufen (*bClScn*) werden soll. Intern begrenzt auf einen Minimalwert von 0bis *cMaxSunBldScn.*

bClScn: Angewählte Szene aufrufen.

bSavScn: Angewählte Szene speichern.

rSpPos: Sollposition [%], welche in der gewählten Szene gesichert werden soll. Ist mit der Ist-Position des Aktorbausteins FB_BA_SunBldActr [▶ 603] bzw. FB_BA_RolBldActr [▶ 594] der Referenz-Jalousie/Rollladе zu verknüpfen, um dadurch eine vorher manuell angefahrene Position speichern zu können. Intern begrenzt auf Werte von 0 bis 100.

rSpAngl: dto. Lamellenwinkel [°].

VAR_OUTPUT

```
stSunBld      : ST_BA_SunBld;
bActv         : BOOL;
rActlScnPos   : REAL;
rActlScnAngl  : REAL;
```

stSunBld: Positioniertelegramm, siehe ST_BA_SunBld [▶ 702].

bActv: Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 702] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

rActlScnPos: Zeigt die gespeicherte relative Jalousiehöhenposition [%] der aktuell angewählten Szene an.

rActlScnAngl: dto. Lammellenwinkel [°].

Wenn ein Fehler ansteht wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

VAR_IN_OUT

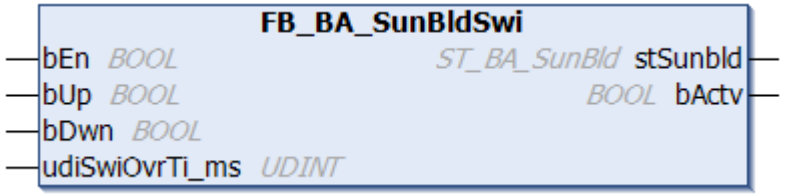
```
arrSunBldScn : ARRAY[0..Param.usiMaxSunBldScn] OF ST_BA_SunBldScn;
```

arrSunBldScn: Tabelle mit den Szeneneinträgen vom Typ ST_BA_SunBldScn [▶ 703].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.24 FB_BA_SunBldSwi



Mit Hilfe dieses Bausteines lassen sich die Jalousieansteuerung FB_BA_SunBldActr [▶ 603] bzw. die Rollladensteuerung FB_BA_RolBldActr [▶ 594] im Handbedienmodus ansteuern. Die Verbindung erfolgt dabei über das Positioniertelegramm ST_BA_Sunbld [▶ 702] entweder direkt oder mit einer zusätzlichen Prioritätssteuerung.

Struktur des Jalousie-Positioniertelegramms ST_BA_Sunbld [▶ 702].

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

Betrieb

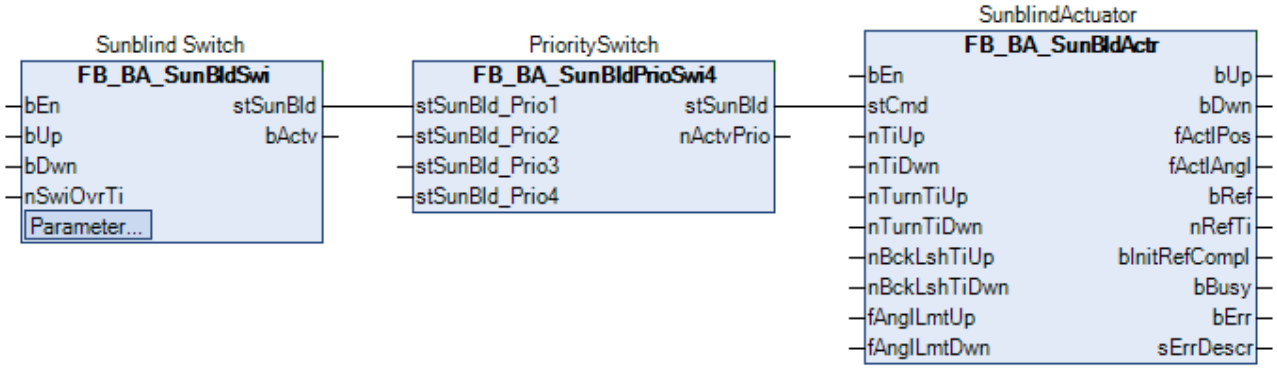
Der Funktionsbaustein steuert über die Befehlseingänge *bUp* und *bDwn* den Jalousiebaustein FB_BA_SunBldActr [▶ 603] bzw. den Rollladenbaustein FB_BA_RolBldActr [▶ 594] im Handbetrieb an, wobei *bUp* Vorrang hat. Die Befehle werden an die jeweiligen Kommandos *bManUp* und *bManDwn* des Positioniertelegramms weitergereicht. Ist ein Befehlseingang länger als die eingetragene Zeit *udiSwiOvrTi_ms* [ms] aktiviert, so geht der entsprechende Steuerbefehl in Selbsthaltung. Ein erneutes Aktivieren eines Befehlseinganges löscht diese Selbsthaltung wieder.
Wird der Baustein durch den Eingang *bEn*=TRUE aktiviert, so wird im Positioniertelegramm das Bit *bActv* sofort gesetzt. Damit meldet der Baustein an einem Prioritätsschalter (FB_BA_SunBldPrioSwi4 [▶ 612] bzw. FB_BA_SunBldPrioSwi8 [▶ 613]) seinen Vorrang gegenüber niedrigen Prioritäten an. Gleichzeitig wird auch das Bit *bManMod* im Positioniertelegramm gesetzt, um den angeschossenen Aktor Bausteinen zu melden, dass diese auf Handbefehle reagieren sollen.

Bei Deaktivierung des Bausteines durch *bEn*=FALSE werden beide Bits, *bActv* und *bManMod* wieder auf FALSE gesetzt.

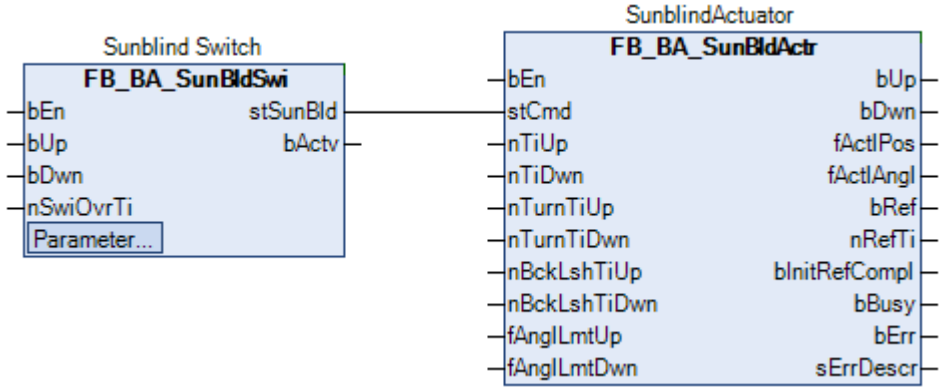
Verknüpfung an Jalousiebaustein

Der Handbedienbaustein kann entweder über eine voran gestellte Prioritätssteuerung FB_BA_SunBldPrioSwi4 [▶ 612] bzw. FB_BA_SunBldPrioSwi8 [▶ 613] oder aber direkt an den Jalousiebaustein angeschlossen werden. Die Verbindung erfolgt dabei über das Positioniertelegramm ST_BA_Sunbld [▶ 702].

Verwendung einer Prioritätssteuerung:



Direkte Beschaltung:



VAR_INPUT

```
bEn             : BOOL;
bUp             : BOOL;
bDwn            : BOOL;
udiSwiOvrTi_ms  : UDINT;
```

bEn: Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_SunBld [▶ 702] werden für Position und Winkel jeweils 0 ausgegeben - *bManMod* und *bActv* stehen jeweils auf FALSE. Das bedeutet für eine Beschaltung mit Prioritätssteuerung, dass eine andere Funktionalität die Jalousieansteuerung übernimmt. Eine direkte Beschaltung hingegen lässt die Jalousie direkt auf Position 0, also ganz nach oben fahren, da der Aktor Baustein das Bit *bActv* selbst nicht auswertet.

bUp: Befehlseingang Jalousie hoch.

bDwn: Befehlseingang Jalousie herunter.

udiSwiOvrTi_ms: Zeit [ms] bis bei dauerhaft aktiviertem Befehlseingang der entsprechende Handbefehl im Positioniertelegramm ST_BA_SunBld [▶ 702] in Selbsthaltung geht. Intern begrenzt auf einen Minimalwert von 0.

VAR_OUTPUT

```
stSunBld    : ST_BA_SunBld;
bActv       : BOOL;
```

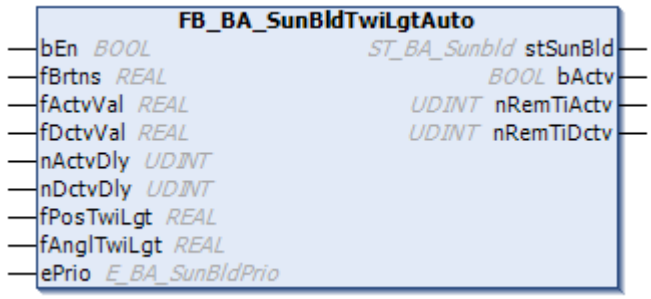
stSunBld: Positioniertelegramm, siehe ST_BA_SunBld [▶ 702].

bActv: Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 702] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.25 FB_BA_SunBldTwiLgtAuto



Dieser Baustein steuert die Jalousie, wenn die Außenhelligkeit einen Grenzwert unterschritten hat.

Die Dämmerungsautomatik arbeitet mit einer Werte- und einer zeitlichen Hysterese: Unterschreitet der Außenhelligkeitswert *rBrtns* [lux] für die Zeit *udiActvDly_sec* [s] den Wert *rActvVal* [lux], so ist der Baustein aktiv und wird die an den Eingangsvariablen angegebenen Jalousiepositionen *rPosTwiLgt* (Höhe [%]) und *rAnglTwiLgt* (Lamellenwinkel [°]) am Ausgang im Positioniertelegramm ST_BA_SunBld [▶ 702] bereitstellen. Überschreitet die Außenhelligkeit hingegen für die Zeit *udiDctvDly_sec* [s] den Wert *rDctvVal* [lux], so ist die Automatik nicht mehr aktiv. Der Aktiv-Merker im Positioniertelegramm ST_BA_SunBld [▶ 702] wird zurück und die Positionen für Höhe und Winkel auf "0" gesetzt. Eine Funktion niedrigerer Priorität kann dann die Steuerung übernehmen.

VAR_INPUT

```
bEn             : BOOL;
rBrtns          : REAL;
rActvVal        : REAL;
rDctvVal        : REAL;
udiActvDly_sec  : UDINT;
udiDctvDly_sec  : UDINT;
rPosTwiLgt      : REAL;
rAnglTwiLgt     : REAL;
```

bEn: Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_SunBld [▶ 702] werden für Position und Winkel jeweils 0 ausgegeben und *bActv* steht auf FALSE. Das bedeutet, dass über die Prioritätssteuerung eine andere Funktionalität die Jalousieansteuerung übernimmt.

rBrtns: Außenhelligkeit [lx].

rActvVal: Aktivierungsgrenzwert [lx]. Der Wert *rActvVal* wird intern begrenzt auf Werte von 0 bis *rDctvVal*.

rDctvVal: Deaktivierungsgrenzwert [lx]. Intern begrenzt auf einen Minimalwert von 0.

udiActvDly_sec: Aktivierungsverzögerung [s]. Intern begrenzt auf einen Minimalwert von 0.

udiDctvDly_sec: Deaktivierungsverzögerung [s]. Intern begrenzt auf einen Minimalwert von 0.

rPosTwiLgt: Höhenposition der Jalousie [%], wenn die Dämmerungsautomatik aktiv ist. Intern begrenzt auf Werte von 0 bis 100.

rAnglTwiLgt: Lamellenwinkel der Jalousie [°], wenn die Dämmerungsautomatik aktiv ist.

VAR_OUTPUT

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

stSunBld: Ausgabestruktur der Jalousiestellungen, siehe ST_BA_SunBld [▶ 702].

bActv: Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 702] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

udiRemTiActv_sec: Zeigt die verbleibende Zeit an nach Unterschreitung des Schaltwertes *rActvVal* bis zur Aktivierung der Automatik [s] an. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.

udiRemTiDctv_sec: Zeigt die verbleibende Zeit an nach Überschreitung des Schaltwertes *rDctvVal* bis zur Abschaltung der Automatik [s] an. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.

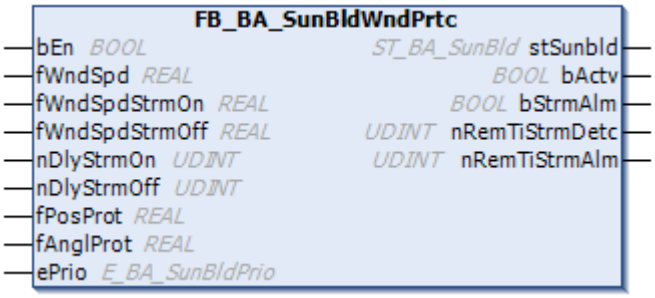


Wenn ein Fehler ansteht wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.26 FB_BA_SunBldWndPrtc



Der Funktionsbaustein *FB_BA_SunBldWndPrtc* behandelt den richtungsabhängigen Windschutz.

Der Witterungsschutz hat bei der Jalousiesteuerung die höchste Priorität (siehe Übersicht [▶ 566]) und soll sicherstellen, dass die Jalousie weder durch Eis, noch durch Wind beschädigt wird.

Liegt die gemessene Windgeschwindigkeit für die Zeit *udiDlyStrmOn_sec* [s] über dem Wert *rWndSpdStrmOn*, so wird davon ausgegangen, dass ein Sturm unmittelbar bevorsteht. Erst wenn die Windgeschwindigkeit den Wert *rWndSpdStrmOff* für die Zeit *udiDlyStrmOff_sec* [s] unterschreitet, gilt der Sturm als abgeflaut und das Fahren der Jalousie als sicher. Aus Sicherheitsgründen wird auch das Sturm-Ereignis persistent gespeichert. Fällt die Steuerung also während eines Sturmes aus, so wird nach Wiederanlauf der Steuerung der Ablauf-Zeitgeber von neuem gestartet.

Bei Windgefahr-Fällen wird die Jalousie in die Schutzposition gefahren die durch *rPosProt* (Höhenposition in Prozent) und *rAnglProt* (Lamellenwinkel [°]) vorgegeben wird.

VAR_INPUT

```
bEn               : BOOL;
rWndSpd           : REAL;
rWndSpdStrmOn     : REAL;
rWndSpdStrmOff    : REAL;
udiDlyStrmOn_sec  : UDINT;
udiDlyStrmOff_sec : UDINT;
rPosProt          : REAL;
rAnglProt         : REAL;
```

bEn: Steht dieser Eingang auf FALSE, so ist der Baustein ohne Funktion. Im Positioniertelegramm ST_BA_Sunbld [▶ 702] werden für Position und Winkel jeweils 0 ausgegeben und *bActv* steht auf FALSE. Das bedeutet, dass über die Prioritätssteuerung eine andere Funktionalität die Jalousieansteuerung übernimmt.

rWndSpd: Windgeschwindigkeit. Die Einheit der Eingabe ist beliebig, jedoch ist es wichtig, dass es keine Werte kleiner als 0 gibt, und die Werte mit zunehmender Geschwindigkeit größer werden.

rWndSpdStrmOn: Windgeschwindigkeits-Grenzwert zur Aktivierung des Sturmalarmes. Dieser Wert darf nicht kleiner als 0 sein und muss oberhalb des Wertes für die Deaktivierung liegen. Anderenfalls wird ein Fehler ausgegeben. Die Einheit der Eingabe muss die gleiche sein wie die des Eingangs *rWndSpd*. Ein Wert größer als dieser Grenzwert löst nach der eingetragenen Zeit *udiDlyStrmOn_sec* den Alarm aus.

rWndSpdStrmOff: Windgeschwindigkeits-Grenzwert zur Deaktivierung des Sturmalarmes. Dieser Wert darf nicht kleiner als 0 sein und muss unterhalb des Wertes für die Aktivierung liegen. Anderenfalls wird ein Fehler ausgegeben. Die Einheit der Eingabe muss die gleiche sein wie die des Einganges *rWndSpd*. Ein Wert kleiner oder gleich diesem Grenzwert setzt nach der eingetragenen Zeit *udiDlyStrmOff_sec* den Alarm zurück.

udiDlyStrmOn_sec: Verzögerungszeit zur Auslösung des Sturmalarmes [s].

udiDlyStrmOff_sec: Verzögerungszeit zum Rücksetzen des Sturmalarmes [s].

rPosProt: Höhenposition der Jalousie [%] im Schutzfall.

rAnglProt: Lamellenwinkel der Jalousie [°] im Schutzfall.

VAR_OUTPUT

```
stSunBld             : ST_BA_SunBld;
bActv                : BOOL;
bStrmAlm             : BOOL;
udiRemTiStrmDetc_sec : UDINT;
udiRemTiStrmAlm_sec  : UDINT;
```

stSunBld: Ausgabestruktur der Jalousiestellungen, siehe ST_BA_SunBld [▶ 702].

bActv: Entspricht dem booleschen Wert *bActv* im Jalousie-Telegramm ST_BA_SunBld [▶ 702] und dient zur reinen Anzeige, ob der Baustein ein aktives Telegramm sendet.

bStrmAlm: Zeigt den Sturmalarm an.

udiRemTiStrmDetc_sec: Im unkritischen Fall zeigt dieser Sekundenzähler konstant die Alarmverzögerungszeit *udiDlyStrmOn_sec* an. Liegt die gemessene Windstärke *rWndSpd* über dem Aktivierungsgrenzwert *rWndSpdStrmOn*, so werden die Sekunden bis zum Alarm heruntergezählt. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.

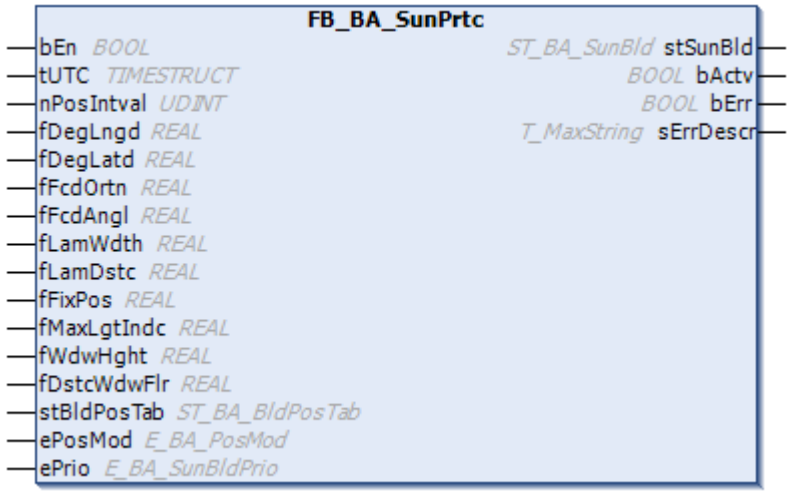
udiRemTiStrmAlm_sec: Sobald der Sturmalarm ausgelöst wird, zeigt dieser Sekundenzähler zunächst konstant die Deaktivierungsverzögerungszeit *udiDlyStrmOff_sec* des Sturmalarmes an. Sinkt die gemessene Windstärke *rWndSpd* unter den Deaktivierungsgrenzwert *rWndSpdStrmOff*, so werden die Sekunden bis zur Entwarnung (*bStrmAlm*=FALSE) heruntergezählt. Solange kein Herunterzählen der Zeit stattfindet, steht dieser Ausgang auf 0.

Wenn ein Fehler ansteht wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.2.3.27 FB_BA_SunPrtc



Der Funktionsbaustein dient zur Blendschutz Steuerung mit Hilfe einer Lamellen-Jalousie.

Der Blendschutz wird durch die Variation des Lamellenwinkels und durch die Position der Jalousiehöhe realisiert.

Der Lamellenwinkel wird in Abhängigkeit des Sonnenstandes so eingestellt, dass direkte Blendung unterbunden, aber trotzdem ein maximaler Einfall natürlichen Lichts möglich ist.

Für das Variieren der Jalousiehöhe stehen drei verschieden Betriebsarten zur Verfügung.

1. Die Jalousie fährt bei aktiven Sonnenschutz in eine fest vorgegebene Höhe. Der Wert für die Höhe wird mit der Variablen *rFixPos* vorgegeben.
2. Die Position der Jalousie wird in Abhängigkeit des Sonnenstandes variiert. Die Position wird in der Tabelle (ST_BA_BldPosTab [▶ 699]) vorgegeben. Siehe auch Beschreibung des FB_BA_BldPosEntry [▶ 572].
3. Die Höhe der Jalousie wird anhand der Fenstergeometrie so berechnet, dass die Sonnenstrahlen bis zu einer vorgegebenen Tiefe in den Raum einfallen. Die Einfalltiefe der Sonnenstrahlen wird mit der Variablen *rMaxLgtIndc* definiert.

Damit das Neupositionieren des Lamellenwinkels nicht zu häufig erfolgt, kann mittels der Variablen *udiPosIntval_min* ein Zeitintervall bestimmt werden, innerhalb dessen keine Anpassung des Lamellenwinkels erfolgt. Um trotzdem eine Blendwirkung auszuschließen, wird der Winkel immer weiter geändert, dass er bis zum Ablauf des Zeitintervalls ausreicht.

Für die Positionierung der Jalousie und der Einstellung des Lamellenwinkels müssen die folgenden Bedingungen erfüllt sein.

- 1. Der Eingang *bEn* muss TRUE sein.
- 2. Die Sonne muss aufgegangen sein. (Elevation > 0)
- 3. Der Baustein ist richtig parametriert (*bErr*=False)

VAR_INPUT

```
bEn                : BOOL;
stUTC              : TIMESTRUCT;
udiPosIntval_min   : UDINT;
rDegLngd           : REAL;
rDegLatd           : REAL;
rFcdOrtn           : REAL;
rFcdAngl           : REAL;
rLamWdth           : REAL;
rLamDstc           : REAL;
rFixPos            : REAL;
rMaxLgtIndc        : REAL;
rWdwHght           : REAL;
rDstcWdwFr         : REAL;
stBldPosTab        : ST_BA_BldPosTab;
ePosMod            : E_BA_PosMod;
```

bEn: Wenn dieser Eingang auf FALSE gesetzt wird, ist die Positionierung nicht aktiv, d.h. in der Positionier-Struktur *stSunBld* vom Typ ST_BA_SunBld [▶ 702] wird das aktiv-Bit (*bActv*) zurückgesetzt und der Baustein selbst verharrt in einem Stillstands-Modus. Ist der Baustein hingegen aktiviert, so ist das aktiv-Bit auf TRUE und der Baustein gibt in der Positionierstruktur zu den entsprechenden Zeiten seine Stellwerte durch (*rPos*, *rAngl*).

stUTC: Eingabe der aktuellen Uhrzeit als koordinierte Weltzeit (UTC, Universal Time Coordinated - im älteren Sprachgebrauch auch GMT, Greenwich-Mean-Time) (siehe TIMESTRUCTTIMESTRUCT). Mit Hilfe des Bausteines FB_BA_GetTime [▶ 631] kann diese Zeit aus einem Zielsystem gelesen werden.



Ein Zeitrücksprung von mehr als 300 s führt, wenn die Jalousie nach den o.a. Kriterien in der Sonne steht und der Blendschutz aktiv ist, zu einer sofortigen Neupositionierung. Diese Funktionalität ist eingefügt worden, um einen nachvollziehbaren Programmablauf zu gewährleisten.

udiPosIntval_min: Positionierintervall in Minuten - Zeitspanne zwischen zwei Ausgaben von Jalousiestellungen. Gültiger Bereich: 1 min...720 min.

rDegLngd: Geographische Länge (Längengrad) [°]. Gültiger Bereich: - 180°...180°.

rDegLatd: Geographische Breite (Breitengrad) [°]. Gültiger Bereich: - 90°...90°.

rFcdOrtn: Fassadenausrichtung [°]:

Dabei gilt auf der Nordhalbkugel für die Fassadenausrichtung (Blick aus dem Fenster):

Blickrichtung	Fassadenausrichtung
Nord	β=0°
Ost	β=90°
Süd	β=180°
West	β=270°

Für die Südhalbkugel gilt hingegen:

Blickrichtung	Fassadenausrichtung
Süd	β=0°
Ost	β=90°
Nord	β=180°
West	β=270°

rFcdAngl: Fassadenneigung [°]. Siehe Fassadenneigung [▶ 570].

rLamWdth: Breite der Lamellen in mm, siehe Skizze [▶ 567].

rLamDstc: Lamellenabstand in mm, siehe Skizze [▶ 567].

rFixPos: Fixe (konstante) Jalousiehöhe [0..100%]. Gültig, wenn *ePosMod* = *ePosModFix* (siehe Enumerator E_BA_PosMod [▶ 698]).

rMaxLgtIndc: Maximal gewünschter Lichteinfall in mm gemessen ab Außenseite der Wand (siehe Höhenverstellung [▶ 569]). Mit Hilfe der Parameter *rWdwHght* und *rDstcWdwFlr* wird in Abhängigkeit des Sonnenstandes errechnet, wie hoch die Jalousie stehen muss, damit der Lichteinfall den Wert *rMaxLgtIndc* nicht überschreitet. Gültig, wenn *ePosMod* = *ePosModeMaxIncidence* (siehe Enumerator E_BA_PosMod [▶ 698]).

rWdwHght: Fensterhöhe in mm zur Errechnung des der Jalousiehöhe wenn der Modus des "Maximal gewünschter Lichteinfalls" gewählt ist.

rDstcWdwFlr: Abstand Boden - Fenstersims in mm zur Errechnung des der Jalousiehöhe wenn der Modus des "Maximal gewünschter Lichteinfalls" gewählt ist.

stBldPosTab: Tabelle von 6 Stützpunkten, davon 4 parametrierbar, aus denen dann durch lineare Interpolation eine Jalousieposition in Abhängigkeit des Sonnenstandes gegeben wird. Gültig, wenn *ePosMod* = *ePosModFix* (siehe Enumerator E_BA_PosMod [▶ 698]). Weitere Beschreibung siehe FB_BA_BldPosEntry [▶ 572].

ePosMod: Auswahl des Positioniermodus, siehe Enumerator E_BA_PosMod [▶ 698].

VAR_OUTPUT

```
stSunBld     : ST_BA_SunBld;
bActv        : BOOL;
bErr         : BOOL;
sErrorDescr  : T_MAXSTRING;
```

stSunBld: Ausgabestruktur der Jalousiestellungen, siehe ST_BA_SunBld [▶ 702]

bActv: Der Baustein ist im aktiv-Zustand, das heißt, es liegt kein Fehler an, der Baustein ist freigegeben und der Sonnenstand befindet sich im eingetragenen Fassadenbereich (die Fassade wird beschienen).

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Die Dauer des Positionierintervalls ist kleiner oder gleich Null oder überschreitet 720 min.
02: Fehler: Der eingetragene Längengrad ist nicht im gültigen Bereich von -180°..180°.
03: Fehler: Der eingetragene Breitengrad ist nicht im gültigen Bereich von -90°..90°.
04: Fehler: Der eingetragene Wert für die Fassadenneigung *rFcdAngl* ist außerhalb des gültigen Bereiches von -90°..90°.
05: Fehler: Der Wert für den Lamellenabstand (*rLamDstc*) ist größer oder gleich dem Wert für die Lamellenbreite (*rLamWdth*). Dies stellt keine "gültige" Jalousie dar, da die Lamellen nicht vollständig schließen könnten. Mathematisch gesehen würde dieses zu Fehlern führen.
06: Fehler: Der eingetragene Wert für die Lamellenbreite *rLamWdth* ist gleich Null.
07: Fehler: Der eingetragene Wert für den Lamellenabstand *rLamDstc* ist gleich Null.
08: Fehler: Der eingetragene Wert für die fixe Jalousiehöhe (*rFixPos*) ist größer als 100 oder kleiner als 0. Gleichzeitig ist Positionierung "fixe Jalousiehöhe" angewählt - *ePosMod*=ePosModFix.
09: Fehler: Das Bit "Werte gültig" (*bVld*) in der Positioniertabelle *stBldPosTab* ist nicht gesetzt - ungültige Werte: siehe FB_BA_BldPosEntry. Gleichzeitig ist Positionierung "Tabelle" angewählt - *ePosMod*=ePosModTab.
10: Fehler: Der eingetragene Wert für den maximal gewünschten Lichteinfall *rMaxLgtIndc* ist kleiner oder gleich Null. Gleichzeitig ist Positionierung "maximaler Lichteinfall" angewählt - *ePosMod*=ePosModMaxIndc.
11: Fehler: Der eingetragene Wert für die Fensterhöhe *rWdwHght* ist kleiner oder gleich Null. Gleichzeitig ist Positionierung "maximaler Lichteinfall" angewählt - *ePosMod*=ePosModMaxIndc.
12: Fehler: Der eingetragene Abstand Fensterunterkante zu Boden *rDstcWdwFlr* ist kleiner Null. Gleichzeitig ist Positionierung "maximaler Lichteinfall" angewählt - *ePosMod*=ePosModMaxIndc.
13: Fehler: nicht gültiger Positioniermodus am Eingang *ePosMod* eingetragen.

Wenn ein Fehler ansteht wird diese Automatik deaktiviert und Position und Winkel auf 0 gesetzt. Das bedeutet, dass bei Verwendung einer Prioritätssteuerung automatisch eine andere Funktion niedrigerer Priorität (siehe Übersicht) die Steuerung der Jalousie übernimmt. Bei direkter Beschaltung hingegen wird die Jalousie auf Position/Winkel 0 fahren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.3 Warmwasserbereitung

Funktionsbausteine

Name	Beschreibung
FB_BA_DHW2P [▶ 624]	Steuerung der Ladung eines Warmwasserspeichers mittels eines Zweipunkt-Reglers.
FB_BA_LglPrev [▶ 626]	Funktionsbaustein zur Desinfektion des Brauchwassers und zum Abtöten von Legionellen.

6.1.2.3.2.1.3.1 FB_BA_DHW2P

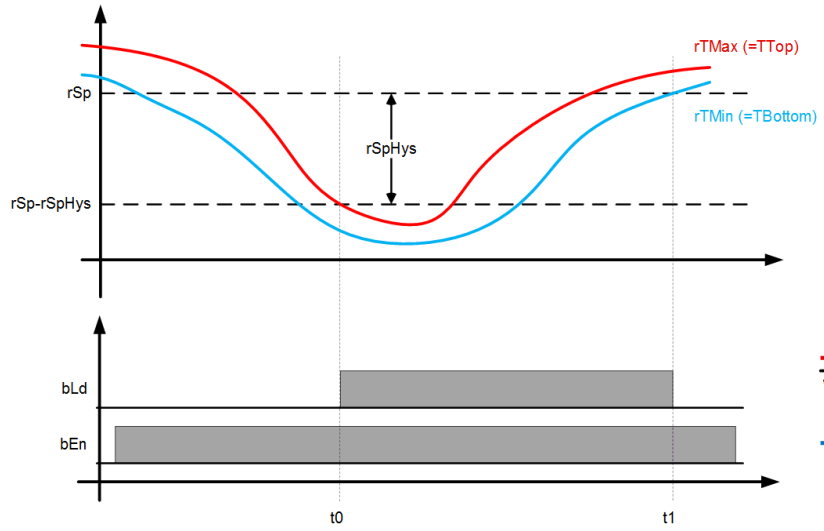


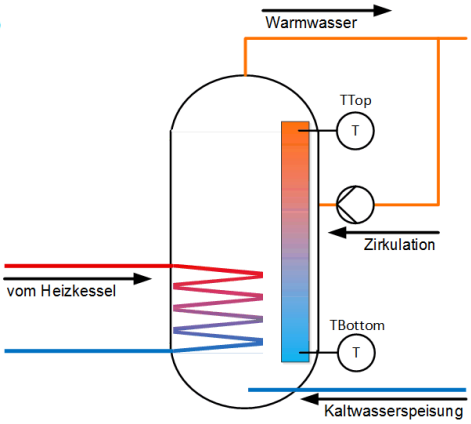
Der Funktionsbaustein steuert die Ladung eines Warmwasserspeichers mittels eines Zweipunkt-Reglers. An dem Eingang *bEn* wird die Speicherladung frei gegeben. Bei aktiver Speicherladung ist der Ausgang *bLd* TRUE. Mit der Variablen *rSp* wird dem Funktionsbaustein der Sollwert für die Brauchwassertemperatur übergeben. An dem Eingang *rTMin* wird eine Minimalauswahl, am Eingang *rTMax* eine Maximalauswahl aller Temperaturfühler des Warmwasserspeichers angeschlossen.
Bedingt durch die Temperaturschichtung im Warmwasserspeicher ist der oberste Fühler in allgemeinen der mit der höchsten Temperatur und der untere derjenige mit der niedrigsten.

Die Speicherladung kann mittels der Variablen *bKepFul* in zwei Arten erfolgen:

bKepFul = FALSE

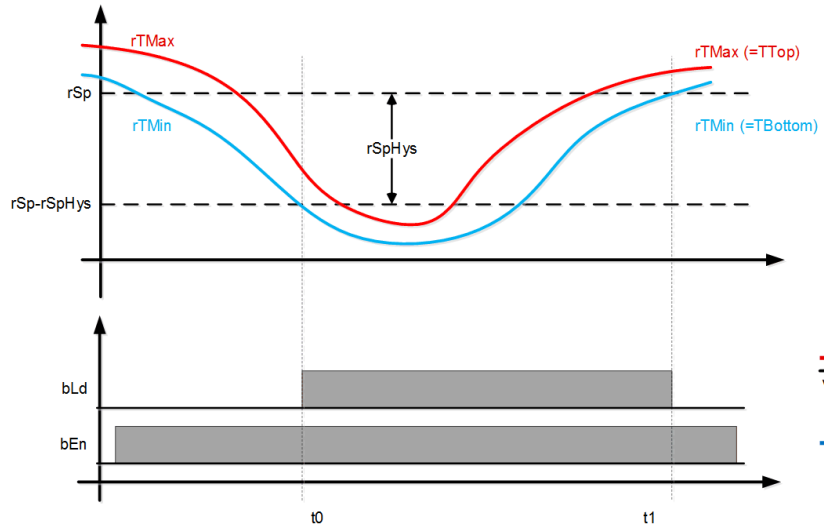
Die Ladung wird angefordert, wenn *rTMax* unter dem Wert von *rSp-rSpHys* liegt. Die Ladungsanforderung wird deaktiviert wenn *rTMin* über dem Sollwert von *rSp* liegt.
Dadurch, dass im Allgemeinen der oberste Fühler die höchste Temperatur misst, erfolgt eine Einschaltung der Speicherladung erst bei entladenem Warmwasserspeicher.

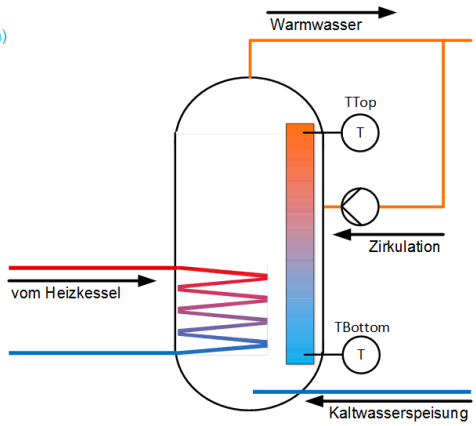




bKepFul = TRUE

Die Ladung wird angefordert, wenn *rTMin* unter dem Wert von *rSp-rSpHys* liegt. Die Ladungsanforderung wird deaktiviert wenn *rTMin* wieder über dem Sollwert ist.
Durch die Minimalauswahl aller Speichertemperaturen, wird der kälteste Punkt des Speichers für die Regelung verwendet. Eine Nachladung erfolgt sobald der Speicher nicht mehr ganz gefüllt ist.





VAR_INPUT

```
bEn       : BOOL;
rSp       : REAL;
rSpHys    : REAL;
```

```
rTMax     : REAL;
rTMin     : REAL;
bKepFul   : BOOL;
```

bEn: Freigabe Boiler Ladung.

rSp: Temperatursollwert des Brauchwassers [°C].

rSpHys: Hysterese, empfohlen 1°K bis 5°K.

rTMax: Maximalauswahl aller Speichertemperaturfühler [°C].

rTMin: Minimalauswahl aller Speichertemperaturfühler [°C].

bKepFul: Steuertemperaturanwahl:

FALSE = mit *rTMax* wird wird *bLd* gefordert, mit *rTMin* ausgeschaltet

TRUE = *rTMin* allein steuert das Ein-Ausschalten von *bLd*

VAR_OUTPUT

```
bLd      : BOOL;
rSpOut   : REAL;
```

bLd: Freigabe des Ladebetriebes.

rSpOut: Sollwertweitergabe an Ladeschaltung:

- *rSpOut* = *rSp* (Eingang) wenn der Baustein aktiviert ist
- *rSpOut* = 0 wenn der Baustein nicht aktiviert ist

Voraussetzungen

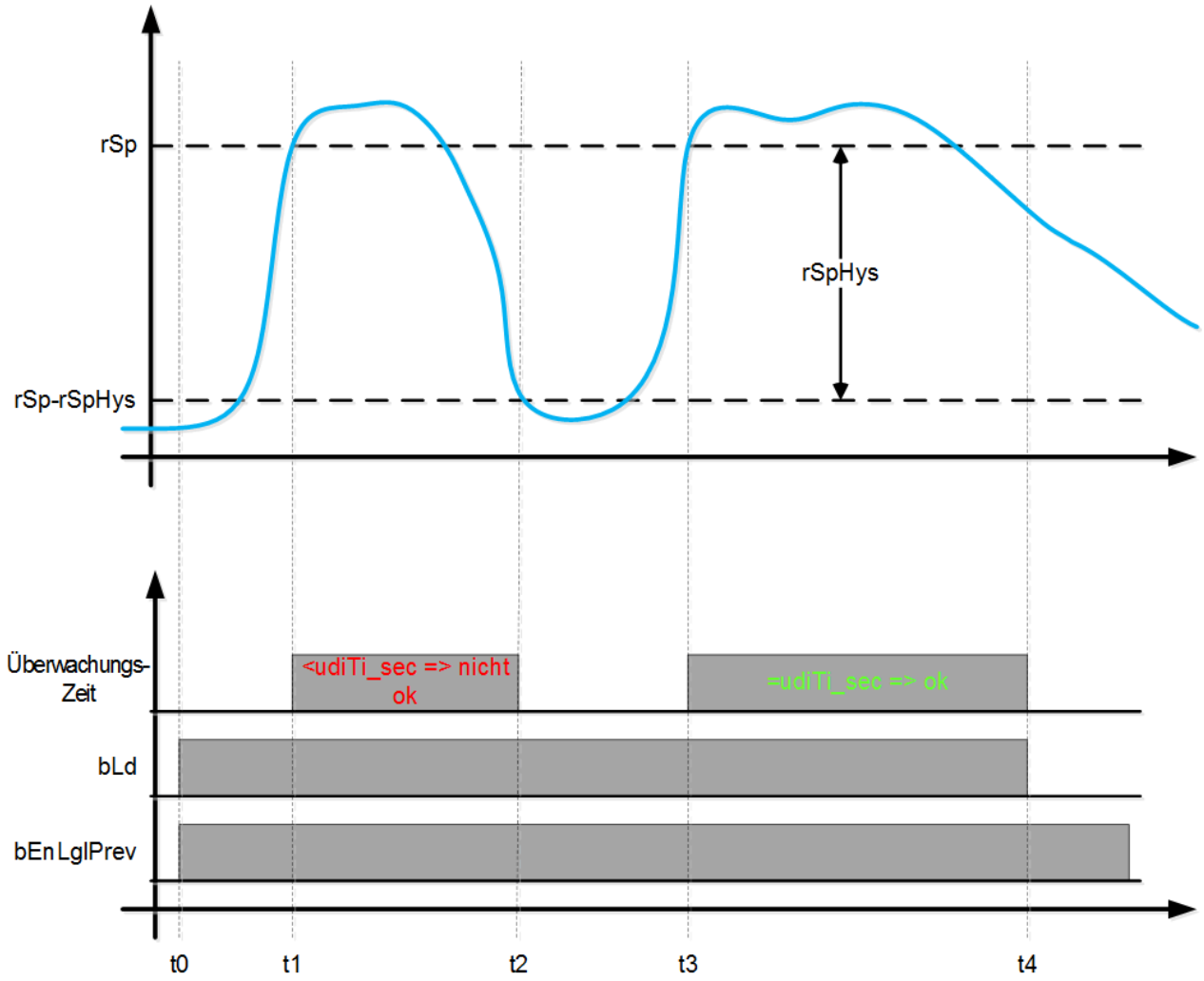
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.3.2 FB_BA_LglPrev



Der Funktionsbaustein dient zur Desinfektion des Brauchwassers und zum Abtöten von Legionellen. Die Freigabe des Desinfektionsbetriebes erfolgt an dem Eingang *bEnLglPrev* durch die Anschaltung eines Zeitschaltprogrammes. Zu empfehlen ist die Durchführung der Desinfektion mindestens einmal in der Woche (nachts). Die Temperatur sollte mindestens 70°C betragen. Das Einschaltintervall an *bEnLglPrev* muss dafür ausreichend lang gewählt werden. Der Ausgang *bLd* aktiviert dabei die Speicherladung.
Bei Warmwasserspeichern mit mehreren Temperaturfühlern muss an *rTMin* eine Min-Auswahl aller Fühler angeschlossen werden.
Überschreitet *rTMin* den Wert von *rSp,* wird ein Überwachungstimer mit einer Zeit von *udiTi_sec* [s] gestartet. Bleibt die minimale Speichertemperatur *rTMin* bis zum Ablauf des Timers oberhalb von *rSp -rSpHys* ist eine ausreichende Aufheizung des Speichers erfolgt. Bei einer vorhandenen Zirkulation muss der Ausgang *bLd* mit der Freigabe der Zirkulationspumpe verknüpft sein, damit auch die Wasserleitung innerhalb des Brauchwassersystems desinfiziert wird. Ist die Temperatur jedoch während des Desinfektionsprozesses unter *rSp -rSpHys* gefallen, muss der Desinfektionsprozess neu gestartet werden bis die Zeit von *udiTi_sec* einmal komplett abgelaufen ist. Bei einer erfolgreichen Desinfektion wird der Ausgang *bLd* zurückgesetzt.

Wurde während der Bausteinaktivierung (*bEnLglPrev)* kein vollständiger Desinfektionsprozess abgeschlossen, wird dieses mit dem Ausgang *bAlm* signalisiert. Der Ausgang muss mit *bRst* zurückgesetzt werden.



Erläuterung zur Grafik:

t0 Starten des Legionellenprogramms und Schalten des Ausgangs *bLd*. Aufheizen des Warmwasserspeichers.

t1 Der Speicher hat die Temperatur *rSp* erreicht. Der Timer für die Aufheizzeit wird gestartet.

t2 Die minimale Speichertemperatur ist unterhalb von *rSp -rSpHys* gefallen. Der Timer für die Aufheizzeit wird wieder zurückgesetzt.

t3 Die Temperatur steigt erneut über *rSp* und der Aufheiztimer wird wieder gestartet.

t4 Die Minimale Speichertemperatur war über den Zeitraum von *udiTi_sec* hinweg oberhalb der Grenze *rSp -rSpHys*, die Desinfektion war erfolgreich. *bLd* wird zurückgesetzt und der Warmwasserspeicher geht wieder in den Normalbetrieb.

VAR_INPUT

```
bEnLglPrev   : BOOL;
rTMin        : REAL;
rSp          : REAL;
rSpHys       : REAL;
udiTi_sec    : UDINT;
bRst         : BOOL;
```

bEnLglPrev: Freigabe des Desinfektionsbetriebes über ein Zeitschaltprogamm.

rTMin: Minimale Speichertemperatur [°C]. Minimalauswahl von Temperaturfühler oben und Temperaturfühler unten.

rSp: Sollwert Desinfektion [°C].

rSpHys: Temperaturspreizung [°K] untere Grenze - wird immer absolut gerechnet.

udiTi_sec: Überwachungszeitraum [s].

bRst: Rücksetzen des Legionellen Alarms;

VAR_OUTPUT

```
bLd     : BOOL;
rSpOut  : REAL;
udiRTi  : UDINT;
udiSta  : UDINT;
```

bLd: Anti-Legionellenbetrieb aktiv.

rSpOut: Sollwertweitergabe an Ladeschaltung:

- rSp (Eingang) wenn der Baustein aktiviert ist
- 0 wenn der Baustein nicht aktiviert ist

udiRTi: Countdown Desinfektions-Betriebstimer.

udiSta: Status Desinfektionsprogramm:

1. Der Desinfektionsbetrieb war erfolgreich.
2. Die Desinfektion ist erfolgreich abgeschlossen. Zum Abschluss und für eine neue Aktivierung der Legionellenvorsorge muss *bEnLglPrev* FALSE sein.
3. Der Desinfektionsbetrieb ist aktiv.
4. Desinfektion war nicht erfolgreich. Alarm steht an.
5. Desinfektion war nicht erfolgreich, der Alarm wurde quittiert.
6. Bedeutet ein Neustart der Steuerung bzw. es wurde noch kein Legionellenbetrieb angefordert.

bAlm: Temperatursollwert wurde nicht durchgängig über den Zeitraum von *udiTi_sec* hinweg erreicht, so dass eine ausreichende Desinfektion nicht gewährleistet ist.

Voraussetzungen

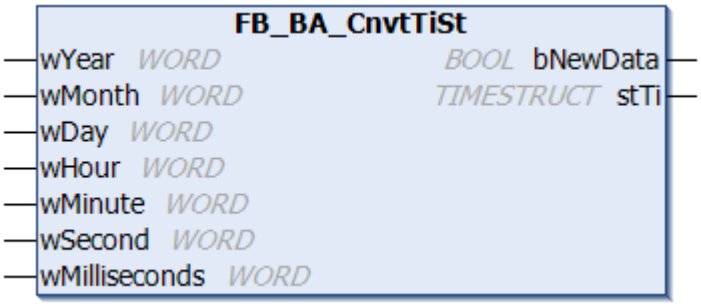
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.4 System

Funktionsbausteine

Name	Beschreibung
FB_BA_CnvtTiSt [▶ 629]	Umwandlung Jahr, Monat, Tag, Stunde, Minute und Sekunde in Zeitstruktur
FB_BA_ExtTiSt [▶ 630]	Umwandlung Zeitstruktur in Jahr, Monat, Tag, Stunde, Minute und Sekunde
FB_BA_GetTime [▶ 631]	Interne Uhr mit Zeitinformationen - synchronisierbar mit Systemzeit
FB_BA_SetTime [▶ 633]	Setzen der Systemzeit
FB_BA_WrtPersistDat [▶ 634]	Schreibt persistente Daten

6.1.2.3.2.1.4.1 FB_BA_CnvtTiSt



Mit dem Funktionsbaustein *FB_BA_CnvtTiSt* können die verschiedenen Bestandteile einer Zeitstruktur zu dieser selbst zusammen gefasst werden.

> Der Baustein verfügt über keine Überprüfung fehlerhafter Eingaben, wie z. B. eine Stundeneingabe von 99. Diese Überprüfung erfolgt sinnvollerweise in den angeschlossenen Bausteinen, die ihrerseits die Zeitstruktur ohnehin überprüfen müssen. Die zulässigen Grenzen werden hier jedoch bei der Variablenerläuterung aufgezeigt.

VAR_INPUT

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

wYear: Das Jahr (1970..2106).

wMonth: Der Monat (1..12).

wDay: Der Tag im Monat (1..31).

wHour: Die Stunde (0..23).

wMinute: Die Minuten (0..59).

wSecond s: Die Sekunden (0..59).

wMillisecond: Die Millisekunden (0..999).

VAR_OUTPUT

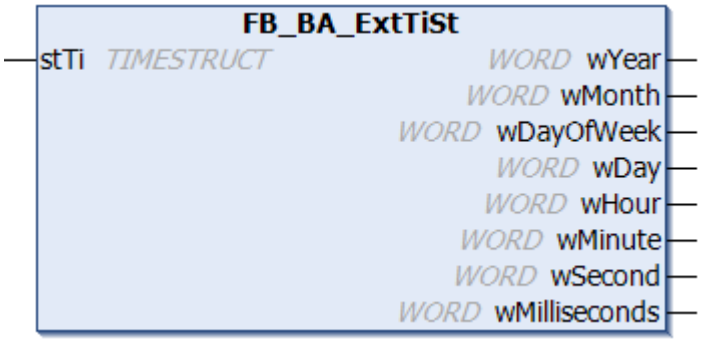
```
stTi      : TIMESTRUCT;
```

stTi: Ausgabe Zeitstruktur (siehe TIMESTRUCT)

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.4.2 FB_BA_ExtTiSt



Der Funktionsbaustein *FB_BA_ExtTiSt* löst eine Zeitstruktur in die verschiedenen Bestandteile auf, um sie so etwa für Zeitbedingungen nutzbar zu machen.

VAR_INPUT

```
stTi      : TIMESTRUCT;
```

stTi: Eingabe Zeitstruktur (siehe TIMESTRUCT)

VAR_OUTPUT

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

wYear: Das Jahr (1970..2106).

wMonth: Der Monat (1..12).

wDayOfWeek: Der Wochentag (0(So)..0(Sa)).

wDay: Der Tag im Monat (1..31).

wHour: Die Stunde (0..23).

wMinute: Die Minuten (0..59).

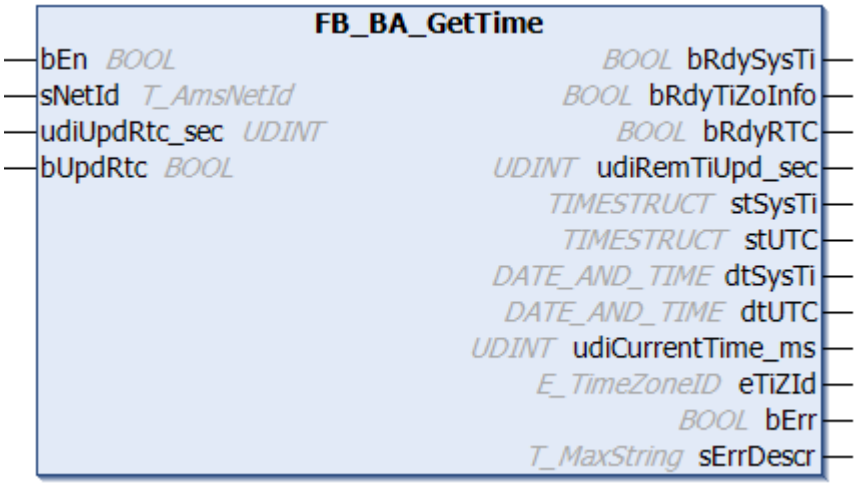
wSecond: Die Sekunden (0..59).

wMilliseconds: Die Millisekunden (0..999).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.4.3 FB_BA_GetTime



Mit diesem Funktionsbaustein kann eine interne Uhr (Real Time Clock RTC) in der TwinCAT SPS realisiert werden. Die RTC-Uhr wird mit der Freigabe des Funktionsbausteines über *bEn* mit der aktuellen NT-Systemzeit initialisiert. Es wird ein Systemtakt der CPU benutzt um die aktuelle RTC-Zeit zu berechnen. Der Funktionsbaustein muss in jedem Zyklus der SPS einmal aufgerufen werden, damit die aktuelle Zeit berechnet werden kann. Intern wird in dem Funktionsbaustein eine Instanz der Funktionsbausteine NT_GetTime, FB_GetTimeZoneInformation und RTC_EX2 aufgerufen. Die Ausgabe der Zeit erfolgt an den Ausgängen *stSysTi* für die gelesene Systemzeit und *stUtc* für die koordinierte Weltzeit (Coordinated Universal Time). Diese wird intern aus der Systemzeit und der Zeitzone ermittelt. Ist die Systemzeit und/oder die Zeitzone fehlerhaft eingegeben worden, so wird auch die UTC-Zeit nicht richtig sein.

Die Systemzeit wird über den einzustellenden Timer *udiUpdRTC_sec* [s] zyklisch ausgelesen und damit die interne RTC-Uhr synchronisiert. In demselben Zyklus werden auch die Zeitinformationen (Zeitzone, Zeitverschiebung zu UTC und Sommer-/ Winterzeit) ausgelesen. Der Ausgang *udiRemTiUpd_sec* zeigt die verbleibenden Sekunden bis zum nächsten Lesezyklus. Die ausgegebenen Zeitstrukturen *stSysTi* und *stUtc* können mit Hilfe des Bausteines FB_BA_ExtTiSt [▶ 630] weiter in die Bestandteile - Tag, Monat, Stunde, Minute etc. - aufgelöst werden.

Hinweise zum Lese-/Wartezyklus

Während des Lesezyklus fallen die Ausgänge *bRdySysTi* und *bRdyTiZoInfo* auf FALSE, der Enumerator *eTiZId* zeigt 0 = *eTimeZoneID_Unknown*. Wurde erfolgreich gelesen, so gehen die Ausgänge wieder auf TRUE bzw. zeigen die jeweilige Information Sommer- oder Winterzeit an, sofern sie verfügbar ist. War das Lesen nicht erfolgreich - es wird intern 5s lang auf eine Antwort gewartet - so bleiben die Ausgänge auf FALSE bzw. 0 stehen und ein erneuter Wartezyklus bis zum nächsten Lesezyklus wird gestartet. Im Fehlerfall wird die interne RTC-Uhr zwar nicht synchronisiert und kann immer noch eine richtige Zeit anzeigen, die Zeitinformationen jedoch können fehlerhaft sein und damit auch die UTC-Zeit. Fehler während des Lesezyklus schlagen sich in jedem Fall in der Anzeige an *bErr* und *sErrDescr* nieder. Der Countdown-Ausgang *udiRemTiUpd_sec* wird erst dann neu gestartet, wenn der Wartezyklus beginnt.

VAR_INPUT

```
bEn             : BOOL;
sNetId          : T_AmsNetId;;
udiUpdRtc_sec   : UDINT;
bUpdRtc         : BOOL;
```

bEn: Freigabe des Bausteins. Ist *bEn* = TRUE, so wird die RTC-Uhr mit der NT-Systemzeit initialisiert.

sNetId: Hier kann die AmsNetId (siehe T_AmsNetId) des TwinCAT-Rechners angegeben werden dessen NT-Systemzeit als Zeitbasis gelesen werden soll. Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

udiUpdRtc_sec: Zeitangabe [s], mit der die RTC-Uhr mit der NT-Systemzeit regelmäßig synchronisiert wird. Diese Zeitangabe Dieser Wert wird intern auf ein Minimum von 5s begrenzt, um die Abarbeitung der internen Bausteine zu gewährleisten.

bUpdRtc: Parallel zu der Zeitangabe *udiUpdRtc_sec* kann die RTC-Uhr über eine positive Flanke an diesem Eingang synchronisiert werden.

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: Die Systemzeit wurde erfolgreich aus dem Zielsystem gelesen.

bRdyTiZoInfo: Die zusätzlichen Zeitinformationen (Zeitzone, Zeitverschiebung zu UTC und Sommer-/ Winterzeit) wurden erfolgreich gelesen.

bRdyRTC: Wurde der Funktionsbaustein mindestens einmal initialisiert, so wird dieser Ausgang gesetzt. Ist dieser Ausgang gesetzt, dann sind die Werte für das Datum, Uhrzeit und Millisekunden an den Ausgängen gültig.

udiRemTiUpd_sec: Countdown zur nächsten Synchronisation bzw. Aktualisierung der Zeitinformationen.

stSysTi: Systemzeit des ausgelesenen Zielsystems (siehe TIMESTRUCT). Die Zeitstruktur kann mit Hilfe des Bausteines FB_BA_ExtTiSt [▶ 630] weiter in die Bestandteile - Tag, Monat, Stunde, Minute etc. - aufgelöst werden.



Ist der Baustein nicht aktiviert (*bEn*=FALSE), so zeigen der Ausgang *stSysTi* in seinen Teilelementen (Tag Monat, etc.) jeweils 0 an.

stUTC: Koordinierte Weltzeit (siehe TIMESTRUCT). Diese wird intern aus der Systemzeit und den gelesenen Zeitinformationen des Zielsystems ermittelt. Die Zeitstruktur kann mit Hilfe des Bausteines FB_BA_ExtTiSt [▶ 630] weiter in die Bestandteile - Tag, Monat, Stunde, Minute etc. - aufgelöst werden.



Ist der Baustein nicht aktiviert (*bEn*=FALSE), so zeigen der Ausgang *stUTC* in seinen Teilelementen (Tag Monat, etc.) jeweils 0 an.

dtSysTi / dtUTC: Wie *stSysTi* / *stUTC*, jedoch im DATE-AND-TIME-Format: Jahr-Monat-Tag-Stunden-Minuten-Sekunden.



Ist der Baustein nicht aktiviert (*bEn*=FALSE), so zeigen die Ausgänge *dtSysTi* und *dtUTC* jeweils DT#1970-01-01-00:00 an, da dies die Untergrenze ist und es den Nullen in der Strukturdarstellung von *stSysTi* / *stUTC* entspricht.

udiCurrentTime_ms: Aktuelle Tageszeit in ms.

eTiZId: Enumerator zur Sommer-/ Winterzeitinformation (siehe E_TimeZoneID).

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Warnung: ADS-Fehler beim Lesen der Zeit (*FB NT_GetTime*). Es wird die ADS-Fehlernummer angegeben.
02: Warnung: ADS-Fehler beim Lesen der Zeitzoneninformation (*FB_GetTimeZoneInformation*). Es wird die ADS-Fehlernummer angegeben.

Voraussetzungen

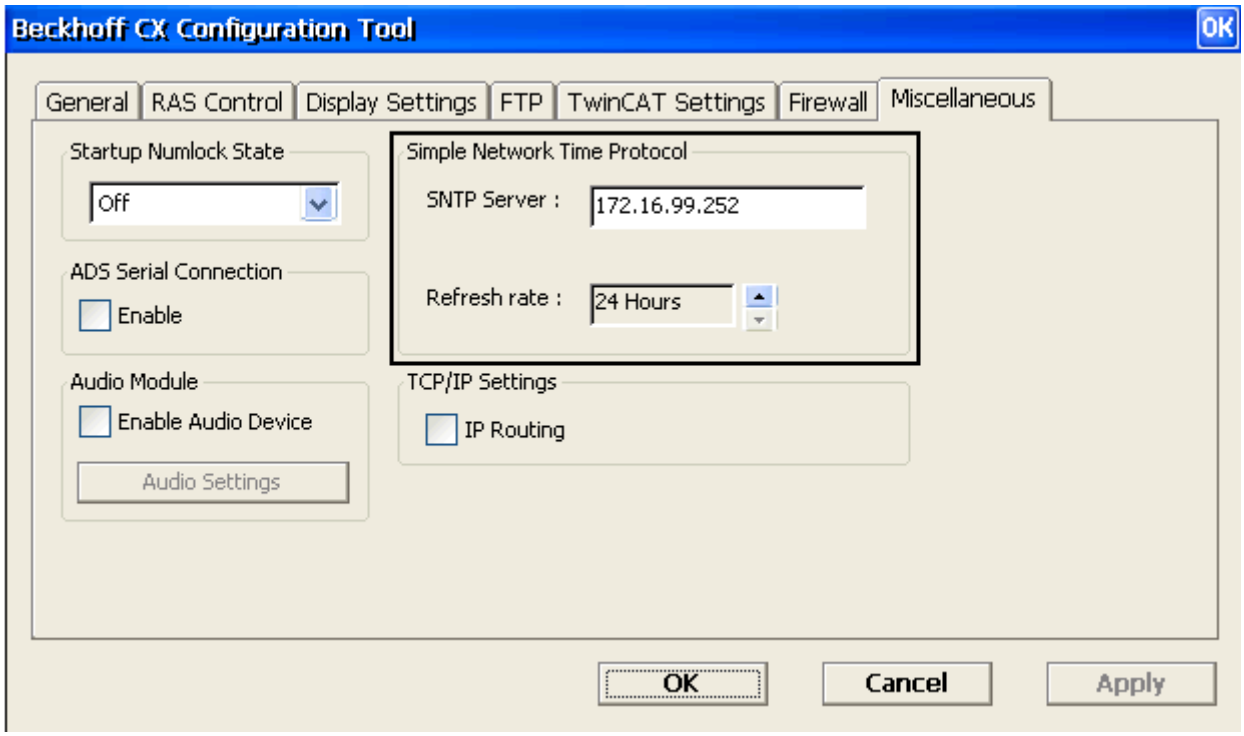
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.4.4 FB_BA_SetTime



Mit dem Funktionsbaustein *FB_BA_SetTime* kann die lokale NT-Systemzeit und das Datum eines TwinCAT-Systems gesetzt werden (die lokale NT-Systemzeit wird in der Taskleiste eingeblendet). Die Systemzeit wird über die Struktur *stSysTi* vorgegeben.
Intern wird in dem Funktionsbaustein eine Instanz des Funktionsbausteins NT_SetLocalTime aus der TcUtilities-Bibliothek aufgerufen.

Die lokale NT-Systemzeit können Sie auch mit der Hilfe des SNTP-Protokolls mit einer Referenzzeit synchronisieren. Mehr Informationen dazu finden Sie im Beckhoff Information System unter: Beckhoff Information System > Embedded-PC > Betriebssysteme > CE > SNTP: Simple Network Time Protocol



VAR_INPUT

```
bSet          : BOOL;
sNetId        : T_AmsNetId;
stSysTi       : TIMESTRUCT;
udiTiOut_sec  : UDINT;
```

bSet: Aktivierung des Funktionsbausteins mit einer steigenden Flanke.

sNetId: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden dessen lokale NT-Systemzeit gesetzt werden soll. Für den lokalen Rechner kann auch ein Leerstring *sNetId* := ''; angegeben werden (siehe T_AmsNetId).

stSysTi: Struktur mit der neuen lokalen NT-Systemzeit (siehe TIMESTRUCT). Ist die Zeit als Struktur nicht vorhanden, so empfiehlt sich die Verwendung des Bausteines FB_BA_CnvtTiSt [▶ 629], welcher die Teilvariablen von Datum und Uhrzeit in einer Struktur zusammen führt.

udiTiOut_sec: Gibt die Timeout-Zeit [s] an, die bei der Ausführung nicht überschritten werden darf.

VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
sErrorDescr     : T_MAXSTRING;
```

bBusy: Bei der Aktivierung des Funktionsbausteins über eine steigende Flanke an *bSet* wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn entweder die zu übertragene Systemzeit fehlerhaft ist, oder aber ein ADS-Fehler in der Übertragung stattfindet.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Fehler Bereichsüberschreitung Jahr
02: Fehler: Fehler Bereichsüberschreitung Monat
03: Fehler: Fehler Bereichsüberschreitung Monatstag
04: Fehler: Fehler Bereichsüberschreitung Stunde
05: Fehler: Fehler Bereichsüberschreitung Minute
06: Fehler: Fehler Bereichsüberschreitung Sekunde
07: Fehler: Fehler Bereichsüberschreitung Millisekunde
08: Warnung: Ein ADS-Fehler ist während des Setzens der Zeit aufgetreten (FB NT_SetLocalTime). Es wird die ADS-Fehlernummer angegeben.

Grenzen der Zeitvorgabe

Die angelegte Zeitstruktur *stSysTi* wird bausteinintern auf Grenzen geprüft (siehe TIMESTRUCT)

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.4.5 FB_BA_WrtPersistDat



Der Baustein *FB_BA_WrtPersistDat* speichert bei Aktivierung zunächst die persistenten Daten in die Port_xxx.bootdata-Datei. Dabei muss nicht explizit angegeben werden, auf welchem Port bzw. Laufzeitsystem sich die PLC befindet – dieses wird intern ermittelt. Sind die Daten geschrieben worden, so wird der Inhalt der Datei Port_xxx.bootdata in die Sicherungsdatei Port_xxx.bootdata-old kopiert. Damit sind beide Dateien immer gleich gehalten und im Fall, dass die Original-Datei mit den persistenten Daten nicht lesbar sein sollte, enthält die Sicherungskopie, welche dann gelesen wird, dieselben Daten.



Dazu muss in jedem Fall das Häkchen bei „Clear Invalid Persistent Data“ entfernt sein (siehe Beschreibung persistentes Datenhandling unter TwinCAT 3 [▶ 635])

Der Baustein lässt sich über zwei Arten starten:

Über eine positive Flanke an dem Eingang *bStt*, sofern der Baustein sich nicht in der eingestellten Aufstartphase befindet.

Initial nach Ablauf der Aufstartphase nach einem Reset bzw. TwinCAT-Restart. Die Dauer wird an *udiInitSttDly_sec* in Sekunden eingestellt. Ist dort eine “0” eingetragen, so ist die Dauer der Aufstartphase 0 und ein initiales Ausführen des Bausteines wird übergangen.

Während der Aufstartphase werden keine Befehle an *bStt* entgegengenommen.

Treten Fehler beim Lesen, Schreiben, Öffnen oder Schließen der Dateien auf, so wird dies mit einer entsprechenden Fehlermeldung an *bErr/sErrDescr* angezeigt. Nach einer intern fest eingestellten Wartezeit von 2 Sekunden versucht der Baustein selbsttätig den Befehl (Lesen, Schreiben, Öffnen oder Schließen) erneut auszuführen.

Es empfiehlt sich daher die Fehlerausgänge im Auge zu behalten bzw. diese auszuwerten.

Des Weiteren ist darauf zu achten, ob beim TwinCAT-Neustart bzw. nach einem Reset die Backup-Datei der persistenten Daten geladen wurde. Das deutet nämlich darauf hin, dass die Originaldatei nicht lesbar und die Speicherkarte des Controllers defekt ist. Dieses kann für jedes Laufzeitsystem mit der boolschen Zuweisung von *TwinCAT_SystemInfoVarList._AppInfo.OldBootData* (siehe PlcAppSystemInfo) abgefragt werden.

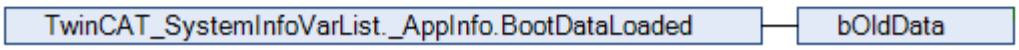
Beispiel in ST:

```
PROGRAM Example_ST
VAR
    bOldData    :   BOOL;
END_VAR
```

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

Beispiel in CFC:

```
PROGRAM Example_CFC
VAR
    bOldData    :   BOOL;
END_VAR
```



HINWEIS

Datei-Handle-Konflikt

Es ist darauf zu achten, dass nur dieser Baustein und auch nur eine Instanz von ihm auf die persistenten Daten zugreifen. Öffnen mehrere Bausteine eine Datei und schließen sie nicht wieder, kann es zu unvorhergesehenen Datei-Handle-Konflikten kommen, welche nicht abgefangen werden können. Die persistenten Daten werden dann nicht mehr in der xxx.bootdata-Datei aktualisiert.

Beschreibung persistentes Datenhandling unter TwinCAT 3

TwinCAT speichert bei jedem geordneten Herunterfahren, das heißt beim Wechsel vom Run- in den Config- oder Stop-Modus, für jedes Laufzeitsystem (Runtime) die persistenten Daten in eine Datei.

Der Name der Datei besteht aus der ADS-Portbezeichnung des Laufzeitsystems mit der Dateiendung *.bootdata*, z. B.: *Port_851.bootdata* und wird im TwinCAT-Verzeichnis unter *TwinCAT\3.1\Boot\PLC* abgespeichert.

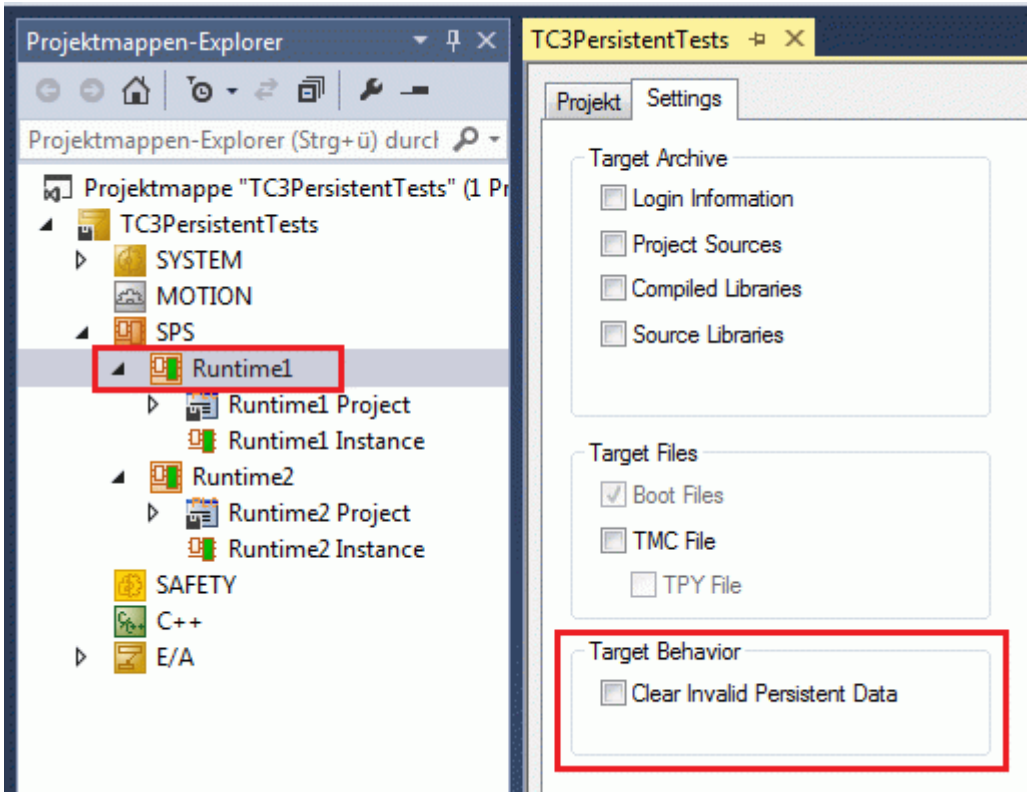
Bei einem erneuten Start des Systems, d.h. bei einem Wechsel in den Run-Modus wird diese Datei gelesen und danach als Port_xxx.bootdata-old gespeichert.

Existiert die Datei Port_xxx.bootdata-old bereits, so wird diese überschrieben.

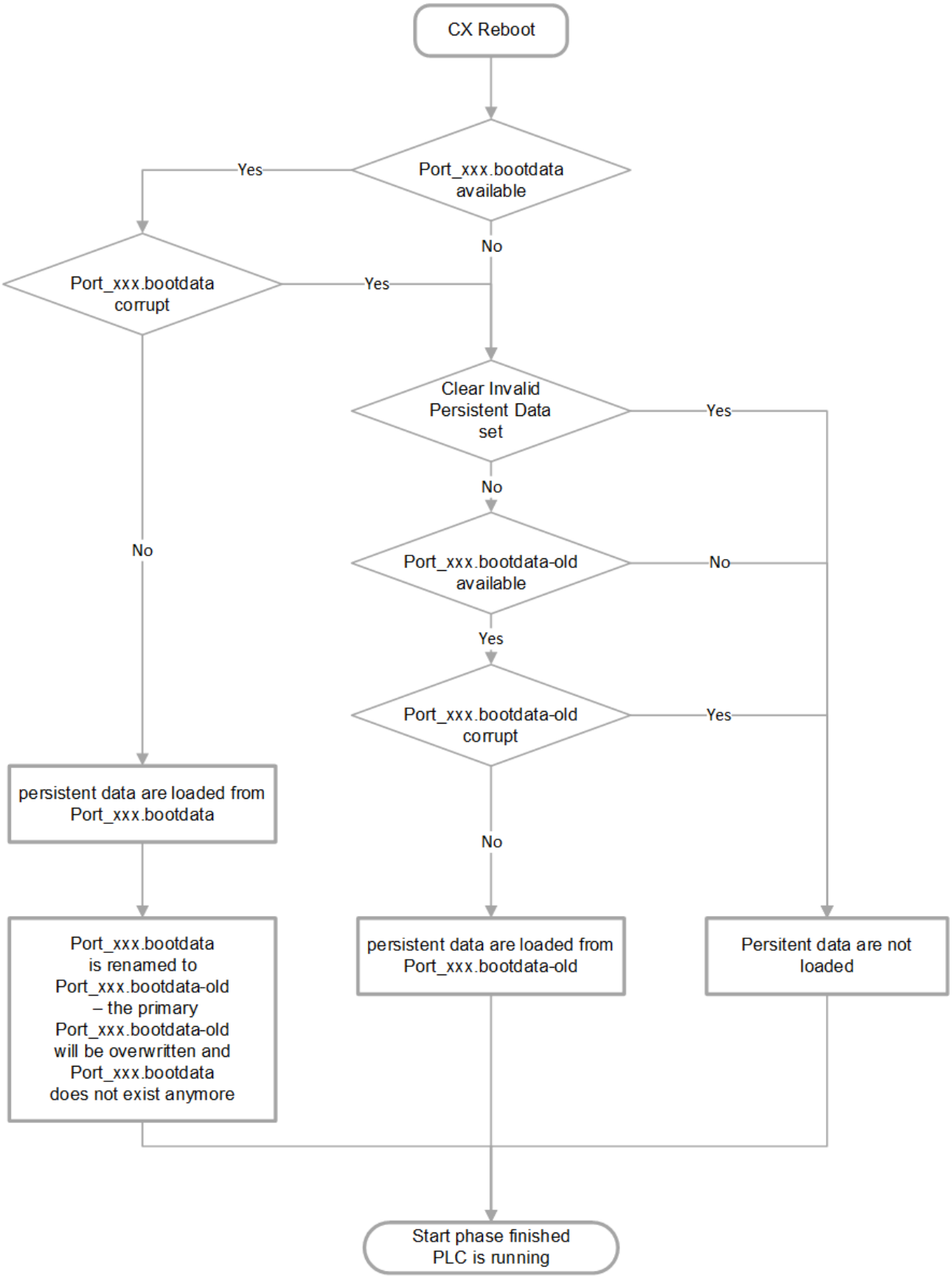
Die ursprüngliche Datei Port_xxx.bootdata besteht nun nicht mehr. Sie wird erst wieder bei einem Wechsel in den Stop-Modus automatisch oder durch den Baustein *FB_WritePersistentData* aus der TC2_Utilities-Bibliothek neu angelegt.

Dieses Verhalten gilt jeweils für jedes Laufzeitsystem – jedes hat seine „eigenen“ Dateien mit persistenten Daten.

Ist nun bei einem Neustart des TwinCAT-Systems die Datei defekt, so greift dann das System automatisch auf die Sicherungsdatei Port_xxx.bootdata-old zurück. Dieses Verhalten gilt aber nur, wenn unter den Laufzeit-Einstellungen das Häkchen **Clear Invalid Persistent Data** abgewählt ist. Ist es angewählt und die Originaldatei ist defekt, so werden keine Daten gelesen.



Der andere Fall, unter denen auf Sicherungsdatei Port_xxx.bootdata-old zurückgegriffen wird ist, wenn der Controller spannungslos gemacht wird. Auch in diesem Fall werden die gerade gültigen persistenten Daten nicht in Port_xxx.bootdata abgespeichert. Dem System stehen dann bei einem erneuten Start nur die alten Daten zur Verfügung, es sei denn, vor dem Abschalten ist durch den Baustein FB_WritePersistentData eine aktuellere Datei erzeugt worden.



VAR_INPUT

```
bStt                : BOOL;
udiInitSttDly_sec   : UDINT;
```

bStt: Eine steigende Flanke an diesem Eingang startet den Baustein, sofern er sich nicht in der Aufstartphase befindet.

udiInitSttDly_sec: Aufstartphase nach einem Reset bzw. TwinCAT-Restart. Die Dauer wird in Sekunden eingestellt. Nach Ablauf der Aufstartphase wird der Baustein einmal automatisch gestartet. Während der Aufstartphase werden keine Befehle für *bStt* entgegengenommen. Ist eine „0“ an *udiInitSttDly_sec* eingestellt, so wird die Aufstartphase übersprungen. Dieser Eingang ist mit 10s vorparametriert.

VAR_OUTPUT

```
bBusy                   : BOOL;
udiRemTiInitSttDly_sec  : UDINT;
bErr                    : BOOL;
sErrDescr               : T_MaxString;
```

bBusy: Der Baustein befindet sich in seiner Ausführung.

udiRemTiInitSttDly_sec: Countdown der eingestellten Aufstartphase.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Die Nummer des von der SPS ausgegebenen ADS-Ports ist „0“
02: Warnung: Fehler beim Schreiben der Schreiben der Persistenten Daten über den internen Baustein *FB_WritePersistentData*. Zusätzlich dessen Fehlernummer.
03: Warnung: Fehler beim Öffnen der Backup-Datei (xxx.bootdata-old) über den internen Baustein *FB_FileOpen*. Zusätzlich dessen Fehlernummer.
04: Warnung: Fehler beim Lesen der Original-Datei (xxx.bootdata) über den internen Baustein *FB_FileRead*. Zusätzlich dessen Fehlernummer
05: Warnung: Fehler beim Schreiben in die Backup-Datei (xxx.bootdata-old) über den internen Baustein *FB_FileWrite*. Zusätzlich dessen Fehlernummer.
06: Warnung: Fehler beim Schließen der Original-Datei (xxx.bootdata) über den internen Baustein *FB_FileClose*. Zusätzlich dessen Fehlernummer.
07: Warnung: Fehler beim Schließen der Backup-Datei (xxx.bootdata-old) über den internen Baustein *FB_FileClose*. Zusätzlich dessen Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

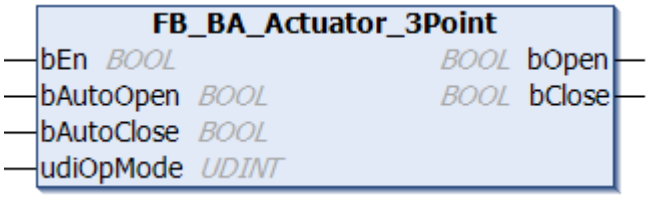
6.1.2.3.2.1.5 Universal

6.1.2.3.2.1.5.1 Aktoren

Funktionsbausteine

Name	Beschreibung
FB_BA_Actuator_3Point [▶ 639]	Ansteuerung von Dreipunktklappen oder -ventilen
FB_BA_Anlg3Pnt [▶ 640]	Analogwert zu Dreipunkt-Wandler
FB_BA_AntBlkg [▶ 641]	Blockierschutz von Pumpen oder Stellantrieben
FB_BA_Motor1St [▶ 642]	Steuerung von einstufigen Antrieben
FB_BA_Motor2St [▶ 643]	Steuerung von zweistufigen Antrieben
FB_BA_PWM [▶ 644]	Puls-Weiten-Modulations-Baustein

6.1.2.3.2.1.5.1.1 FB_BA_Actuator_3Point



Der Funktionsbaustein dient der Steuerung einer 3-Punkt-Aktuators z. B. einer 3-Punktklappe oder eines 3-Punkt-Ventils.

Am Ausgang *bOpen* wird der Befehl zum Auffahren des Aktors angeschlossen.

Am Ausgang *bClose* wird der Befehl zum Zufahren des Aktors angeschlossen.

Im Automatikbetrieb (*udiOpMode*=0) werden die Stellbefehle von *bCmdOpen* und *bCmdClose* direkt an die Ausgänge *bOpen* und *bClose* weitergeleitet.

Mit Hilfe des Eingangs *udiOpMode* wird die Betriebsart des 3-Punkt-Aktors bestimmt:

- 0 = Automatik
- 1 = Stop (bOpen = bClose = FALSE)
- 2 = Zu
- 3 = Auf

VAR_INPUT

```
bEn         : BOOL;
bAutoOpen   : BOOL;
bAutoClose  : BOOL;
udiOpMode   : UDINT
```

bEn: Allgemeine Freigabe des Funktionsbausteins.

bAutoOpen: Befehl zum Auffahren des Aktors.

bAutoClose: Befehl zum Zufahren des Aktors.

udiOpMode: Auswahl der Betriebsart (0 = Automatik, 1 = Stop (bOpen = bClose = FALSE), 2 = Zu, 3 = Auf)

VAR_OUTPUT

```
bOpen   : BOOL;
bClose  : BOOL;
```

bOpen: Steuerausgang öffnen.

bClose: Steuerausgang schließen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.1.2 FB_BA_Anlg3Pnt



Der Funktionsbaustein ist für die Ansteuerung von Dreipunktstellantrieben zur Betätigung von Ventilen oder Klappen vorgesehen.

Ein stetiges Stellsignal für die Positionierung eines Stellantriebs wird in die binären Befehle für das Auf- und Zufahren umgewandelt.

Ist die Abweichung zwischen dem Positionssollwert *rIn* und dem errechneten Positionsistwert *rPos* des Stellorgans größer als der eingestellte Schwellwert *rHys/2*, dann beginnt der Funktionsbaustein abhängig von dem Betrag der Regelabweichung durch Schalten der Ausgänge *bOpn* oder *bCls* die Position zu korrigieren:

	bOpn	bCls
rIn - rPos > rHys/2	TRUE	FALSE
rIn - rPos < - rHys/2	FALSE	TRUE

Erreicht der Baustein durch einen entsprechenden Eingangswert *rIn* eine Endlage *rOut*=0 oder *rOut*=100, so bleibt der entsprechende Schaltausgang dauerhaft gesetzt, um diese Endlage auch am Ventil bzw. der Klappe sicher zu erreichen:

	bOpn	bCls
rOut = 0	FALSE	dauerhaft TRUE
rOut = 100	dauerhaft TRUE	FALSE

Eine eventuelle Abschaltung des Dauersignals muss durch äußere Programmierung des Anwenders erfolgen.

Die Eingabe *rIn* wird intern automatisch auf den Bereich von 0..100% begrenzt.

Dies gilt ebenfalls für die Eingaben *rHys* und *rRefVal*. Die Verfahrzeiten *udiTiCls_ms* sowie *udiTiOpn_ms* sind beide nach unten hin auf 10 (Millisekunden) begrenzt.

Eine steigende Flanke an *bRef* löst einen Referenzierbefehl (Setzen der errechneten Ist Position auf *rRefVal*) aus.

Falls der Antrieb über Endlagenschalter verfügt, können diese auch direkt mittels Digitaleingang erfasst und zur Referenzierung an *bRef* verwendet werden.

VAR_INPUT

```
rIn          : REAL;
rHys         : REAL;
udiTiCls_ms  : UDINT;
udiTiOpn_ms  : UDINT;
bRef         : BOOL;
rRefVal      : REAL;
bCloseInit   : BOOL;
```

rIn: Sollwert für die Position des Stellantriebs [0 - 100%]. Intern begrenzt auf Werte von 0 bis 100.

rHys: Hysterese für die Position des Stellantriebs [0 - 100%]. Intern begrenzt auf Werte von 0 bis 100.

udiTiCls_ms: Fahrzeit des Stellantriebs von auf nach zu [ms]. Intern begrenzt auf Werte von 0 bis 100.

udiTiOpn_ms: Fahrzeit des Stellantriebs von zu nach auf [ms]. Intern begrenzt auf Werte von 0 bis 100.

bRef: Flanke referenziert den internen Positionsspeicher des Antriebs auf Wert von *rRefVal* [0 - 100%].

rRefVal: Wert für die Referenzierung des Stellantrieb mit *bRef* [0 - 100%]. Intern begrenzt auf Werte von 0 bis 100.

bCloseInit: Wenn dieser Eingang TRUE ist, dann ist der Ausgang *bCls* für die Zeit udiTiOpn_ms TRUE

VAR_OUTPUT

```
bCls    : BOOL;
bOpn    : BOOL;
rPos    : REAL;
```

bCls: Ausgang für das Zufahren des Stellantriebs.

bOpn: Ausgang für das Auffahren des Stellantriebs.

rPos: aktuelle errechnete Position des Stellantriebs [0 - 100%].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.1.3 FB_BA_AntBlkg



Dieser Funktionsbaustein verhindert das Blockieren von Pumpen oder Stellantrieben nach längeren Zeiten ohne Bewegung durch die Ausgabe eines Einschaltimpulses.
Die maximale Dauer des Stillstands bis zur Ausgabe eines Impulses wird durch den Wert von *udiTiOffMin_sec* bestimmt. Zur Erfassung der Stillstandzeit muss der Eingang *bFdb* mit der Betriebsrückmeldung des Aggregates verbunden werden. Die Länge des Impulses wird mit *udiTiImplLngt_sec* parametriert. Falls die Antiblockierschutz-Impulse nicht in Abhängigkeit der Stillstandszeiten, sondern zyklisch von einem Zeitschaltplan ausgehen sollen, ist hierfür der Eingang *bExe* zu verwenden. Bei einer steigenden Flanke an *bExe* wird unmittelbar ein Impuls an *bQ* ausgegeben. Eine Impulsausgabe erfolgt generell nur wenn der Funktionsbaustein an *bEn* frei geschaltet ist.

VAR_INPUT

```
bEn               : BOOL;
bFdb              : BOOL;
bExe              : BOOL;
udiTiOffMin_sec   : UDINT;
udiTiImplLngt_sec : UDINT;
```

bEn: Freigabe des Funktionsbausteins.

bFdb: Eingang zum Anschluss des Feedbacksignals eines Motors oder Ventils.

bExe: Steigende Flanke forciert die Ausgabe eines Impulses.

udiTiOffMin_sec: Mindestausschaltzeit [s]: nach dem Ablauf der Zeit von *udiTiOffMin_sec* ohne Bewegung des Aggregates erfolgt die Ausgabe eines Impulses.

udiTiImplLngt_sec: Länge des Antiblockierschutz-Impulses [s] an *bQ*.

VAR_OUTPUT

```
bQ                : BOOL;
udiRTiOffMin_sec  : UDINT;
udiRTiImplLngt_sec : UDINT;
```

bQ: Ausgang zur Ausgabe des Impulses.

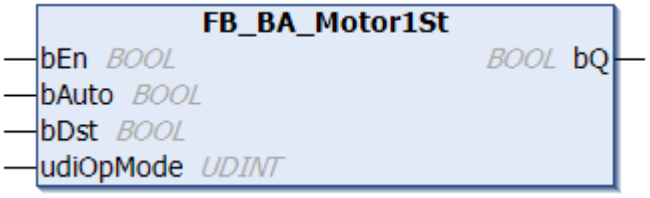
udiRTiOffMin_sec: Verbleibende Zeit [s] bis zur Ausgabe des nächsten Impulses bei andauerndem Stillstand.

udiTiImplLngt_sec: Verbleibende Restzeit [s] des Impulses an *bQ*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.1.4 FB_BA_Motor1St



Funktionsbaustein zur Steuerung eines einfachen einstufigen Motors.

Der Eingang *bEn* dient der allgemeinen Freigabe des Motors.

Über den Eingang *udiOpMode* wird die Betriebsart des Motors eingestellt:

- 0 = Automatik
- 1 = Hand Aus
- 2 = Hand Ein

Im Automatikbetrieb (*udiOpMode* = 0) lässt sich der Motor über den Eingang *bAuto* bedienen (*bAuto* = *bQ* =TRUE).

An *bDst* wird die Sammlung aller möglichen Störungen eines Motors angeschlossen.

VAR_INPUT

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

bEn: Freigabe Motor.

bAuto: Anforderung des Aktors im Automatikbetrieb (*udiOpMode*= 0).

bDst: Eingang zur Sammlung der möglichen Motorstörungen.

udiOpMode: Auswahl der Betriebsart (0 = Automatik, 1 = Hand Aus, 2 = Hand Ein).

VAR_OUTPUT

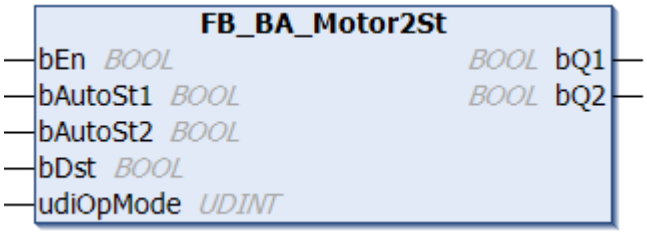
```
bQ  : BOOL;
```

bQ: Steuerausgang.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.1.5 FB_BA_Motor2St



Funktionsbaustein zur Steuerung eines einfachen zweistufigen Motors.

Der Eingang *bEn* dient der allgemeinen Freigabe des Motors.

Über den Eingang *udiOpMode* wird die Betriebsart des Motors eingestellt:

- 0 = Automatik
- 1 = Hand Aus
- 2 = Hand Stufe 1
- 3 = Hand Stufe 2

Im Automatikbetrieb (*udiOpMode*= 0) lässt sich die gewünschte Stufe über die Eingänge *bAutoSt1*(Stufe 1) und *bAutoSt2* (Stufe 2) einstellen.

An *bDst* wird die Sammlung aller möglichen Störungen eines Motors angeschlossen.

VAR_INPUT

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

bEn: Freigabe Motor.

bAutoSt1: Anforderung des Aktors auf Stufe 1 im Automatikbetrieb (*udiOpMode*= 0).

bAutoSt2: Anforderung des Aktors auf Stufe 2 im Automatikbetrieb (*udiOpMode*= 0).

bDst: Eingang zur Sammlung der möglichen Motorstörungen.

udiOpMode: Auswahl der Betriebsart (0 = Automatik, 1 = Hand Aus, 2 = Hand Stufe 1, 3 = Hand Stufe 2).

VAR_OUTPUT

```
bQ1  : BOOL;
bQ2  : BOOL;
```

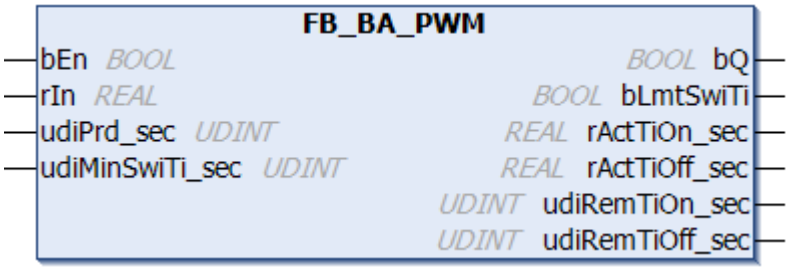
bQ1: Steuerausgang Stufe 1.

bQ2: Steuerausgang Stufe 2.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.1.6 FB_BA_PWM



Der Funktionsbaustein errechnet aus einem analogen Eingangssignal *rIn* (0..100%, **intern fest begrenzt**) und dem der Periodendauer *udiPrd_sec* [s] eine Ein- und eine Ausschaltzeit *rActTiOn_sec* und *rActTiOff_sec* [s].

Dabei entspricht:

- 100% am Eingang einer Einschaltzeit *rActTiOn_sec* von der gesamten Periodendauer *udiPrd_sec* und einer Ausschaltzeit *rActTiOff_sec* von 0s
- 0% am Eingang einer Einschaltzeit *rActTiOn_sec* von 0s und einer Ausschaltzeit *rActTiOff_sec* von der gesamten Periodendauer *udiPrd_sec*.

Darüber hinaus gibt es die Möglichkeit über *udiMinSwiTi_sec* [s] die Schaltdauer nach unten hin zu begrenzen, um Schäden an Antrieben durch zu kurze Stellimpulse zu vermeiden. Dieses Verhalten gilt jedoch nur für 0>*rIn*>100!

Bei *rIn*=0 bzw. 100 bleibt der Ausgang *bQ* jedoch gelöscht, bzw. gesetzt, Nach Ablauf der Periodenzeit wird erneut das aktuelle Eingangssignal bewertet. Steht es immer noch auf 0 bzw. 100 erfolgt kein Zustandswechsel von *bQ*.

Schaltverhalten

1. Ein FALSE-Signal am Eingang *bEn* setzt den Baustein außer Funktion und *bQ* auf FALSE. Es werden lediglich die Ein-und Ausschaltzeiten kontinuierlich berechnet und an den Ausgängen *rActTiOn_sec*/ *rActTiOff_sec* [s] angezeigt.
2. Eine steigende Flanke am Eingang *bEn* aktiviert den Baustein: Er springt zunächst in einen Entscheidungs-Schritt. Je nach vorherigem Zustand vom Schaltausgang *bQ* wird nun in den umschaltenden Schritt gesprungen. Steht der Eingang *rIn* jedoch auf 0, so erfolgt ein unmittelbarer Sprung in den Aus-Schritt (*bQ*=FALSE) bzw. bei *rIn*=100 in den EIN-Schritt (*bQ*=TRUE), ohne auf den vorherigen Zustand von *bQ* zu achten. Die Mindest-Schaltzeit ist für diese beiden Fälle deaktiviert.
3. In dem jeweils aktiven Schritt (EIN oder AUS) läuft nun ein Countdown-Timer mit dem aktuell errechneten Startwert ab, der sich aus dem Puls-Pause-Verhältnis ergibt. Der Ein- bzw. Aus-Schritt wird mit der errechneten Zeit durchlaufen, egal, ob sich währenddessen das Puls-Pause-Verhältnis ändert. Der jeweilige Countdown wird an den Ausgängen *udiRemTiOn_sec*/*udiRemTiOff_sec* in vollen Sekunden angezeigt.
4. Nach Ablauf des Ein- bzw. des Aus-Schrittes erfolgt ein Rücksprung in den Entscheidungsschritt (Punkt 2).

VAR_INPUT

```
bEn             : BOOL;
rIn             : REAL;
udiPrd_sec      : UDINT;
udiMinSwiTi_sec : UDINT;
```

bEn: Aktivierung der Puls-Weiten-Modulation.

rIn: Eingangssignal, intern auf 0..100% fest begrenzt.

udiPrd_sec: Periodenzeit [s]. Intern begrenzt auf einen Minimalwert von 0.

udiMinSwiTi_sec: Minimale Einschaltzeit [s], zur Vermeidung zu kurzer Pulse. Intern begrenzt auf Werte von 0 bis *udiPrd_sec*.

VAR_OUTPUT

```
bQ              : BOOL;
bLmtSwiTi       : BOOL;
rActTiOn_sec    : REAL;
rActTiOff_sec   : REAL;
udiRemTiOn_sec  : UDINT;
udiRemTiOff_sec : UDINT;
```

bQ: PWM-Ausgang.

bLmtSwiTi: Informationsausgang, dass das Eingangssignal so niedrig ist, dass mit der minimalen Einschaltzeit begrenzt wird.

rActTiOn_sec: Informationsausgang: errechnete Einschaltzeit.

rActTiOff_sec: Informationsausgang: errechnete Ausschaltzeit.

udiRemTiOn_sec: Countdown Einschalttimer.

udiRemTiOff_sec: Countdown Ausschalttimer.

Voraussetzungen

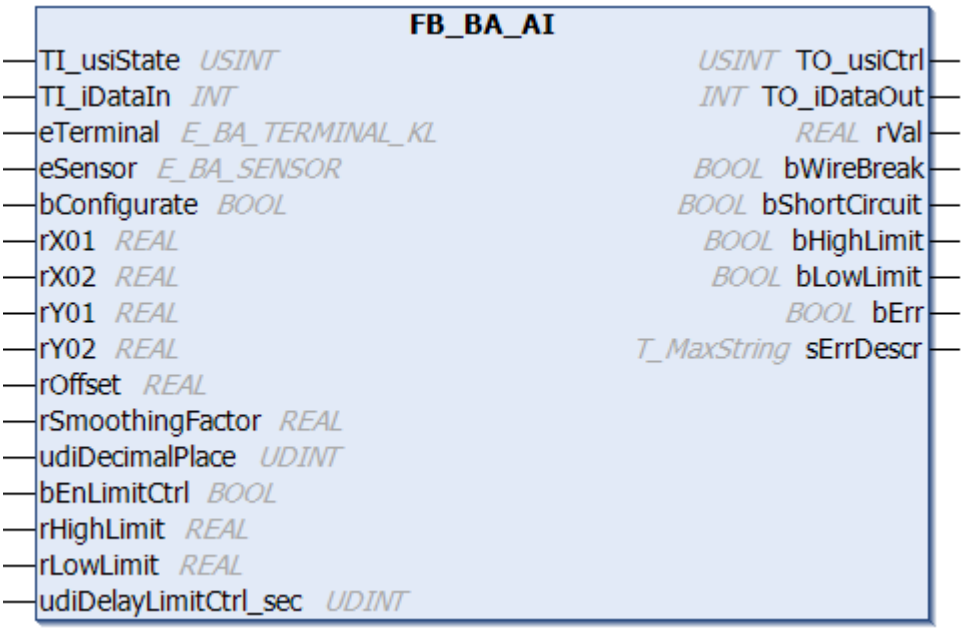
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.2 Analoge Ein-/Ausgänge

Funktionsbausteine

Name	Beschreibung
FB_BA_AI [▶ 645]	Erfassung analoger Eingangssignale
FB_BA_AO [▶ 648]	Ansteuerung von stetigen Stellorganen mit integrierter Skalierungsfunktion
FB_BA_KL32xxConfig	Parametrierung des angeschlossenen Sensortyps an einem Eingangskanal aus der SPS heraus

6.1.2.3.2.1.5.2.1 FB_BA_AI



Der Funktionsbaustein dient der Messdatenverarbeitung und der Klemmenkonfiguration aller branchenüblichen K-Bus- Analogeingangsklemmen.

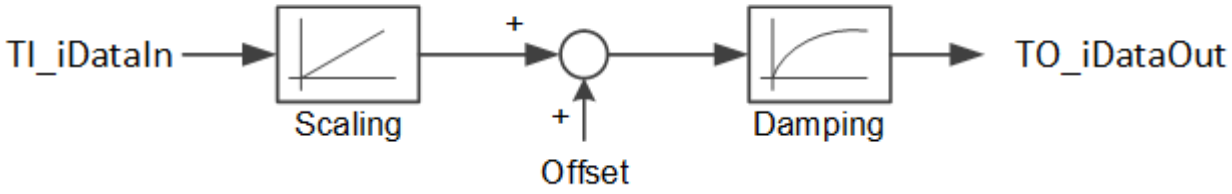
Klemmenkonfiguration

Der erste Schritt bei der Verwendung dieses Funktionsbausteins, ist die Auswahl des entsprechenden Klemmentyps eTerminal. Eine korrekte Funktion des Bausteins ist nur gewährleistet, wenn die mit der Variablen *eTerminal*ausgewählte Klemme, mit der tatsächlich eingesetzten und verknüpften Klemme übereinstimmt.
Bei den Klemmen für die Widerstandstemperaturmessung vom Typ KL3208_0010 und KL320x_0000, wird mit der Enumeration eSensor zusätzlich der an dem Klemmeneingang verwendete Temperatursensor ausgewählt.
Eine steigende Flanke an *bConfigurate* schreibt die Klemmeneinstellungen mittels der Variablen *TO_usiCtrl* und *TO_iDataOut* in die Klemme herunter.

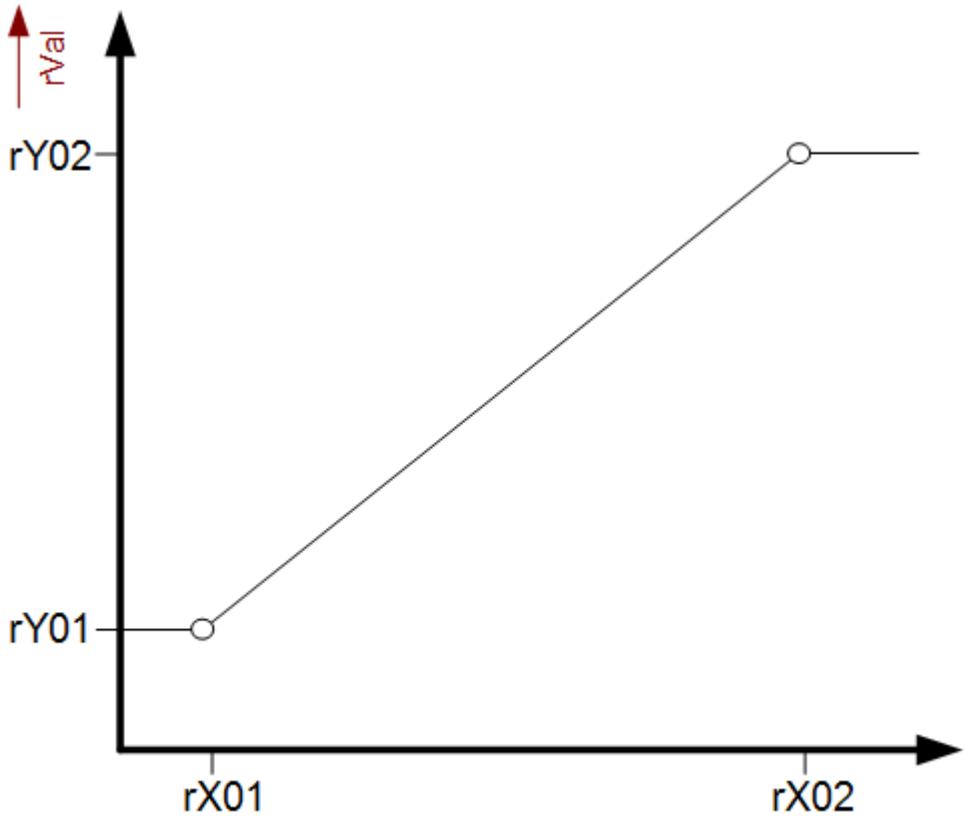
Messwertskalierung

Bei den Analogeingangsklemmen der Typen:
KL300x, KL306x, KL3132_0000, KL3162_0000, KL3172_0000, KL3172_0500, KL3172_1000, KL3182_0000, KL3404, KL3464, KL3408, KL3468, wird der Rohwert *TI_iDataIn* von der Klemme durch den internen Funktionsbaustein FB_BA_Chrct02 [▶ 683] skaliert. (Sieke Skizze). Die Parametrierung der Skalierungfunktion erfolgt mittels der Parameter X(01/02) und Y(01/02). Ein Signal-Offset, für eine eventuell erforderliche Messwertkorrektur, kann mit der Variablen *rOffset* parametriert werden. Durch *rSmoothingFactor* wird das skalierte Messsignal einschließlich dem Offset bedämpft.

Signalfluss:

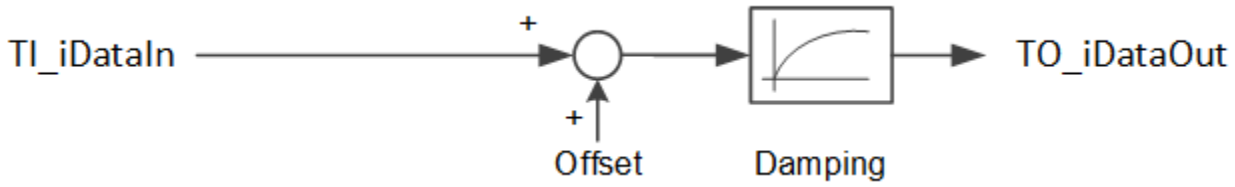


Skalierung:



Bei den Temperaturmessklemmen der Typen KL3208_0010, KL320x_0000 erfolgt keine Skalierung des Meßsignals innerhalb des Funktionsbausteins, sondern direkt in der Klemme. Eine Bedämpfung des eventuell zu stark schwankenden Messsignals, erfolgt mit dem Faktor *rSmoothingFactor*. Wird ein Fehler an der Klemme, wie beispielsweise *bShortCircuit* oder *bWireBrake* erkannt, dann wird bis zur Beseitigung des Fehlers am Ausgang des Funktionsbausteins der zuletzt gültige Wert ausgegeben.

Signalfluss:



Messbereichsüberwachung

Mit den Variablen *rHighLimit* und *rLowLimit* wird der Messwert *rVal* am Ausgang des Funktionsbausteins auf die Einhaltung eines gültigen Bereiches überwacht. Ist der Messwert *rVal* oberhalb von *rHighLimit* oder unterhalb von *rLowLimit*, dann wird dieses an den Ausgängen *bHighLimit* bzw *bLowLimit* angezeigt. Die Messbereichsüberwachung ist deaktiviert, wenn die Eingangsvariable *bEnLimitCtrl* FALSE ist, oder wenn mittels des Klemmenstatus *TI_usiState*, ein Fehler (z. B. Kurzschluss) erkannt wird.
Das Ansprechen der Messbereichsüberwachung kann durch die Variable *udiDelayLimitCtrl_sec* ansprechverzögert werden.

VAR_INPUT

```
TI_usiState           : USINT;
TI_iDataIn            : INT;
eTerminal             : E_BA_TERMINAL_KL;
eSensor               : E_BA_SENSOR;
bConfigurate          : BOOL;
rX01                  : REAL;
rX02                  : REAL;
rY01                  : REAL;
rY02                  : REAL;
rOffset               : REAL;
rSmoothingFactor      : REAL;
udiDecimalPlace       : UDINT(1..6);
bEnLimitCtrl          : BOOL;
rHighLimit            : REAL;
rLowLimit             : REAL;
udiDelayLimitCtrl_sec : UDINT;
```

TI_usiState: Verknüpfung mit dem entsprechenden Status Byte der Busklemme im E/A Bereich des Programms.

TI_iDataIn: Verknüpfung mit den entsprechenden Rohdaten (Data In) der Busklemme im E/A Bereich des Programms (0..32767).

eTerminal: Auswahl der entsprechenden Busklemme (siehe E_BA_Terminal_KL).

eSensor: Auswahl des Sensortyps (siehe E_BA_Sensor).

bConfigurate: Eine steigende Flanke startet die Konfiguration der Busklemme.

rX01: X-Wert für den Interpolationspunkt P1.

rX02: X-Wert für den Interpolationspunkt P2.

rY01: Y-Wert für den Interpolationspunkt P1.

rY02: Y-Wert für den Interpolationspunkt P2.

rOffset: Offset.

rSmoothingFactor: Dämpfungsfaktor. Intern begrenzt von 1 bis 10000.

udiDecimalPlace: Angabe der Dezimalstellen für den Wert *rVal*. Voreingestellt auf 1.

bEnLimitCtrl: Freigabe Grenzwertüberwachung.

rHighLimit: Oberer Grenzwert.

rLowLimit: Unterer Grenzwert.

udiDelayLimitCtrl_sec: Zeitverzögerung bis zur Aktivierung der Grenzwertüberwachung.

VAR_OUTPUT

```
TO_usiCtrl      : USINT;
TO_iDataOut     : INT;
rVal            : REAL;
bWireBreak      : BOOL;
bShortCircuit   : BOOL;
bHighLimit      : BOOL;
bLowLimit       : BOOL;
bErr            : BOOL;
sErrDescr       : T_MAXSTRING;
```

TO_usiCtrl: Verknüpfung mit dem entsprechenden Control Byte der Busklemme im E/A Bereich des Programms.

TO_iDataOut: Verknüpfung mit den entsprechenden Rohdaten (Data Out) der Busklemme im E/A Bereich des Programms.

rVal: Skalierter Ausgangswert.

bWireBreak: Drahtbruch am Sensor.

bShortCircuit: Kurzschluss am Sensor.

bHighLimit: Oberer Grenzwert überschritten.

bLowLimit: Unterer Grenzwert unterschritten.

bErr: Der Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: Falscher Skalierungsparameter *rX01*/*rX02*/*rY01*/*rY02*
02: Fehler: Klemmenkonfiguration überprüfen KL32xx *eTerminal*/*eSensor*/*TI_usiState*/*TI_iDataIn*/*TO_usiCtrl*/*IO_iDataOut*

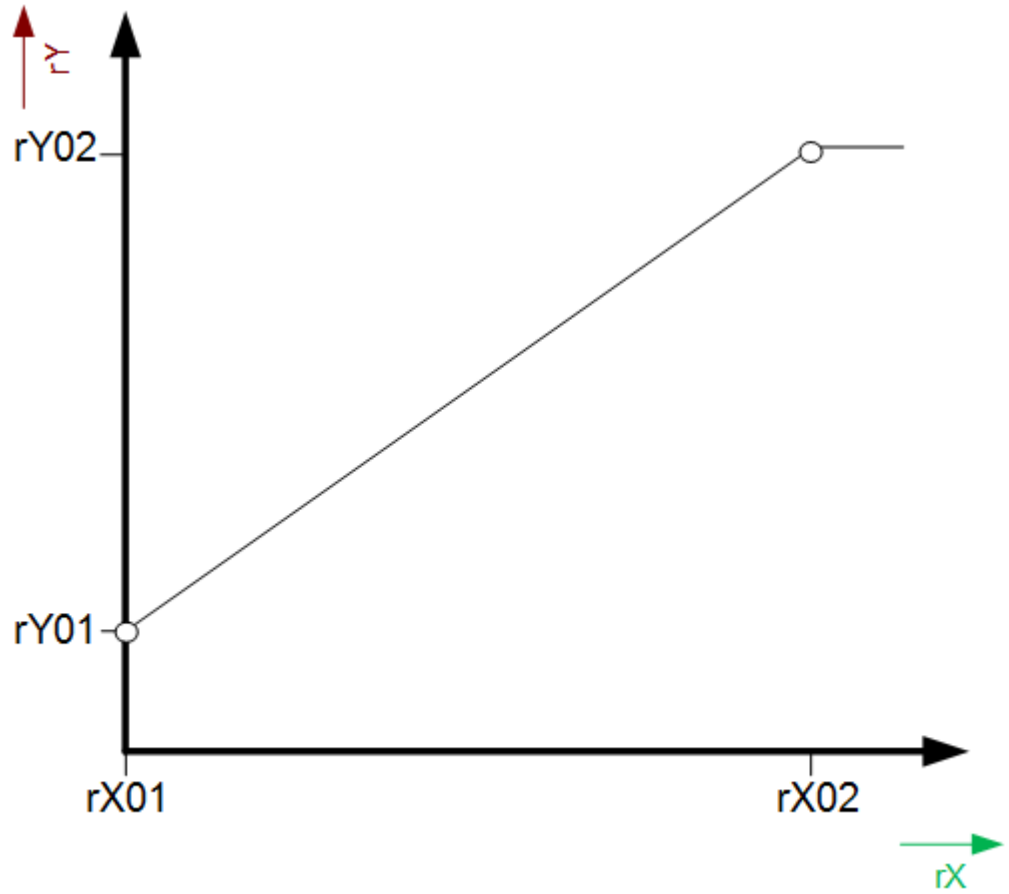
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.2.2 FB_BA_AO



Der Funktionsbaustein *FB_BA_AO* dient der Ausgabe und Skalierung eines analogen Ausgangs. Der Eingangswert an *rX* wird mittels einer Geradengleichung in einen Ausgabewert von 0 bis 32767 bzw. 0-10 Volt oder 4 – 20 mA umgewandelt.



Über den Eingang *udiOpMod* wird die Betriebsart eingestellt:

- 0 = Automatik
- 1 = Hand

Im Automatik Betrieb (*udiOpMod* = *0*) wird der Eingangswert *rX* weitergegeben.

Im Hand Betrieb (*udiOpMod* = 1) wird der Eingangswert *rXManual* weitergegeben.

VAR_INPUT

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

rX: Eingangswert vom Prozess im Automatikbetrieb.

rX01: X-Wert für den Interpolationspunkt P1.

rX02: X- Wert für den Interpolationspunkt P2.

rY01: Y- Wert für den Interpolationspunkt P1.

rY02: Y- Wert für den Interpolationspunkt P2.

udiOpMod: Auswahl der Betriebsart (0 = Automatik, 1 = Hand).

rXManual: Eingangswert für den Handbetrieb.

VAR_OUTPUT

```
rY     : REAL;
TO_iY  : INT;
```

rY: Ausgangssignal als Fließkomma-Zahl.

TO_iY: Ausgangssignal als Integer Wert zur Verknüpfung mit dem Ausgangswert der Busklemme im E/A Bereich des Programms.

Voraussetzungen

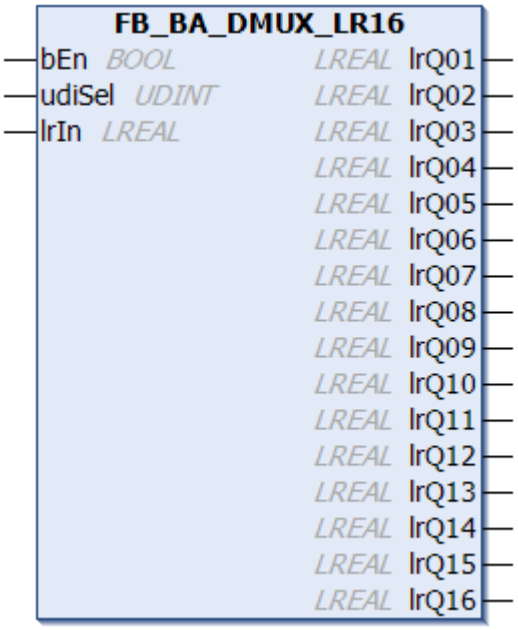
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3 Generelle Steuerfunktionen

Funktionsbausteine

Name	Beschreibung
FB_BA_DMUX_XX [▶ 650]	Demultiplexer-Bausteine
FB_BA_MMUX_XX [▶ 653]	Funktionsbausteine schalten in Abhängigkeit eines Selektors und der entsprechenden Eingangs-Selektorbedingung einen Eingangswert auf den Ausgang
FB_BA_MUX_XX [▶ 656]	Multiplexer-Bausteine
FB_BA_PrioSwi_XX [▶ 658]	Prioritätenschalter
FB_BA_Blink [▶ 660]	Einfacher Oszillator-Baustein
FB_BA_FIFO04 [▶ 661]	Folgesteuerung von bis zu vier Aggregaten
FB_BA_FIFO08 [▶ 663]	Folgesteuerung von bis zu acht Aggregaten
FB_BA_StepCtrl08 [▶ 665]	Schrittkettenbaustein, 8 Schritte
FB_BA_StepCtrl12 [▶ 668]	Schrittkettenbaustein, 12 Schritte
FB_BA_FIFO04_XX [▶ 671]	
FB_BA_FIFO08_XX [▶ 672]	

6.1.2.3.2.1.5.3.1 FB_BA_DMUX_XX



Die Demultiplexer-Bausteine existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Ausgangsgrößen (4, 8, 12 und 16), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_DMUX_LR16 beschrieben.

Der Funktionsbaustein gibt im aktivierten Zustand (*bEn*=TRUE) den Wert am Eingang *lrIn* an demjenigen Ausgang *lrQ01..lrQ16* aus dessen Nummer am Eingang *udiSel* eingetragen ist. Alle anderen Ausgänge werden auf 0 gesetzt (bei den booleschen Demultiplexern auf FALSE).

Beispiel:

Eingänge	Ausgänge
bEn = TRUE	lrQ01 = 0.0
udiSel = 5	lrQ02 = 0.0
lrIn = 32.5	lrQ03 = 0.0
	lrQ04 = 0.0
	lrQ05 = 32.5
	lrQ06 = 0.0
	lrQ07 = 0.0
	lrQ08 = 0.0
	lrQ09 = 0.0
	lrQ10 = 0.0
	lrQ11 = 0.0
	lrQ12 = 0.0
	lrQ13 = 0.0
	lrQ14 = 0.0
	lrQ15 = 0.0
	lrQ16 = 0.0

Ist der eingetragene Wert an *udiSel* größer als die Anzahl der Ausgänge, so wird am "höchsten" Ausgang der Wert von *lrIn* ausgegeben:

Eingänge	Ausgänge
bEn = TRUE	lrQ01 = 0.0
udiSel = 25	lrQ02 = 0.0
rIn = 32.5	lrQ03 = 0.0
	lrQ04 = 0.0
	lrQ05 = 0.0
	lrQ06 = 0.0
	lrQ07 = 0.0
	lrQ08 = 0.0
	lrQ09 = 0.0
	lrQ10 = 0.0
	lrQ11 = 0.0
	lrQ12 = 0.0
	lrQ13 = 0.0
	lrQ14 = 0.0
	lrQ15 = 0.0
	lrQ16 = 32.5

Bei *bEn*=FALSE wird an allen Ausgängen 0.0 bzw. FALSE bei den booleschen Demultiplexern ausgegeben.

VAR_INPUT

```
bEn     : BOOL;
udiSel  : UDINT;
rIn     : LREAL;
```

bEn: Aktivierung der Bausteinfunktion.

udiSel: Nummer des Ausgangs *lrQ00...lrQ16*, der den Wert des Eingangs *lrIn* annehmen soll.

lrIn: Wert, der zur Ausgabe kommen soll.

VAR_OUTPUT

```
lrQ00 : LREAL;
lrQ01 : LREAL;
lrQ02 : LREAL;
lrQ03 : LREAL;
lrQ04 : LREAL;
lrQ05 : LREAL;
lrQ06 : LREAL;
lrQ07 : LREAL;
lrQ08 : LREAL;
lrQ09 : LREAL;
lrQ10 : LREAL;
lrQ11 : LREAL;
lrQ12 : LREAL;
lrQ13 : LREAL;
lrQ14 : LREAL;
lrQ15 : LREAL;
lrQ16 : LREAL;
```

lrQ00...lrQ16: Wertausgänge.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.2 FB_BA_MMUX_XX



Der Funktionsbaustein schaltet in Abhängigkeit eines Selektors und der entsprechenden Eingangs-Selektorbedingung einen Eingangswert auf den Ausgang.

Die Multiplexer-Bausteine existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Eingangsgrößen (4, 8, 12, 16 und 24), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_MMUX_R16 beschrieben.

Der Funktionsbaustein schaltet im aktivierten Zustand (*bEn*=TRUE) in Abhängigkeit eines Selektors *udiSel* und der entsprechenden Eingangs-Selektorbedingung *udiEnxx* einen von den Eingangswerten *rValxx* auf den Ausgang *rVal*.
Wenn mehrere Eingangs-Selektorbedingungen *udiEn01*...*udiEn16* gleich sind und der Selektor *udiSel* einer Bedingung entspricht, dann wird der Eingangwert *rVal01*... *rVal16* der tiefsten aktiven Selektorbedingung auf den Ausgang *rVal* geschaltet. *udiEn01* ist die tiefste, *udiEn16* die höchste Selektorbedingung.

Die Ausgangsvariable *bQ* zeigt an, dass der Selektor *udiSel* mit einer Eingangs-Selektorbedingung *udiEnxx* übereinstimmt.

Die Ausgangsvariable *udiActvPrio* zeigt die aktive Selektorbedingung an.

Ist keine Selektorbedingung aktiv, so wird *rReplVal* an *rVal* ausgegeben. *bQ* ist dann FALSE und *udiActvPrio* zeigt eine 255 an.

Beispiel:

Eingänge		**Ausgang**	
Variable	**Wert**	**Variable**	**Wert**
bEn	TRUE	bQ	TRUE
udiSel	5	rVal	1,123
udiEn01	4	udiActvPrio	7
rVal01	123		
udiEn02			
rVal02			
udiEn03	3		
rVal03	321		
udiEn04			
rVal04			
udiEn05	8		
rVal05	345		
udiEn06			
rVal06			
udiEn07	5		
rVal07	1,123		
udiEn08			
rVal08			
udiEn09	5		
rVal09	5,4321		
udiEn10			
rVal10			
udiEn11			
rVal11			
udiEn12			
rVal12			
udiEn13			
rVal13			
udiEn14			
rVal14			
udiEn15			
rVal15			
udiEn16			
rVal16			
rReplVal			

Wenn keine aktive Priorität ansteht, dann wird an dem Ausgang *udiActvPrio* der Wert der globalen Konstanten Const.udiNoActvPrio [▶ 705] ausgegeben.

VAR_INPUT

```
bEn         : BOOL;
udiSel      : UDINT;
udiEn01     : UDINT := Const.udiNoActvPrio;
rVal01      : REAL;
udiEn02     : UDINT := Const.udiNoActvPrio;
```

```
rVal02     : REAL;
udiEn03    : UDINT := Const.udiNoActvPrio;
rVal03     : REAL;
udiEn04    : UDINT := Const.udiNoActvPrio;
rVal04     : REAL;
udiEn05    : UDINT := Const.udiNoActvPrio;
rVal05     : REAL;
udiEn06    : UDINT := Const.udiNoActvPrio;
rVal06     : REAL;
udiEn07    : UDINT := Const.udiNoActvPrio;
rVal07     : REAL;
udiEn08    : UDINT := Const.udiNoActvPrio;
rVal08     : REAL;
udiEn09    : UDINT := Const.udiNoActvPrio;
rVal09     : REAL;
udiEn10    : UDINT := Const.udiNoActvPrio;
rVal10     : REAL;
udiEn11    : UDINT := Const.udiNoActvPrio;
rVal11     : REAL;
udiEn12    : UDINT := Const.udiNoActvPrio;
rVal12     : REAL;
udiEn13    : UDINT := Const.udiNoActvPrio;
rVal13     : REAL;
udiEn14    : UDINT := Const.udiNoActvPrio;
rVal14     : REAL;
udiEn15    : UDINT := Const.udiNoActvPrio;
rVal15     : REAL;
udiEn16    : UDINT := Const.udiNoActvPrio;
rVal16     : REAL;
rReplVal   : REAL;
```

bEn: Aktivierung der Bausteinfunktion.

udiSel: Selektor. Intern begrenzt auf Werte von 0 bis 4294967294.

udiEn01..udiEn16: Eingangs-Selektorbedingung.
Die Eingangsvariablen sind auf den Wert 255 vorinitialisiert.

rVal01...rVal16: Eingangswerte, aus denen gewählt werden soll.

rReplVal: Ersatzwert, wenn keine Eingangs-Selektorbedingung aktiv ist

VAR_OUTPUT

```
bQ           : BOOL;
rVal         : REAL;
udiActvPrio  : UDINT;
```

bQ: Ist TRUE, wenn der Selektor *udiSel* mit einer Eingangs-Selektorbedingung *udiEnxx* übereinstimmt.

rVal: Wert der gewählten Eingangs-Selektorbedingung.

udiActvPrio: Zeigt an welche Eingangs-Selektorbedingung aktiv ist. Wenn keine aktive Priorität ansteht, dann wird an dem Ausgang *udiActvPrio* der Wert der globalen Konstanten Const.udiNoActvPrio [▶ 705] ausgegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.3 FB_BA_MUX_XX



Die Multiplexer-Bausteine existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Eingangsgrößen (4, 8, 12 und 16), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_MUX_LR16 beschrieben.

Der Funktionsbaustein gibt im aktivierten Zustand (*bEn*=TRUE) denjenigen Eingangswert *lrIn01..lrIn16* am Ausgang *lrQ* aus, dessen Nummer am Eingang *udiSel* eingetragen ist.
Beispiel:

Eingänge	Ausgang
bEn = TRUE	lrQ = 16.5
udiSel = 5	
lrIn01 = 15.9	
lrIn02 = 32.5	
lrIn03 = 17.4	
lrIn04 = 5.84	
lrIn05 = 9.56	
lrIn06 = 16.5	
lrIn07 = 32.781	
lrIn08 = 25.4	
lrIn09 = 44.5	
lrIn10 = 66.1	
lrIn11 = 45.5	
lrIn12 = 83.3	
lrIn13 = 54.56	
lrIn14 = 33.8	
lrIn15 = 98.5	
lrIn16 = 71.3	

Ist der eingetragene Wert an *udiSel* größer als die Anzahl der Eingänge, so wird der "höchstrangige" Eingang an *lrQ* ausgegeben:

Eingänge	Ausgang
bEn = TRUE	lrQ = 2.3
udiSel = 25	
lrIn01 = 15.9	
lrIn02 = 32.5	
lrIn03 = 17.4	
lrIn04 = 5.84	
lrIn05 = 9.56	
lrIn06 = 16.5	
lrIn07 = 32.781	
lrIn08 = 25.4	
lrIn09 = 44.5	
lrIn10 = 66.1	
lrIn11 = 45.5	
lrIn12 = 83.3	
lrIn13 = 54.56	
lrIn14 = 33.8	
lrIn15 = 98.5	
lrIn16 = 71.3	

Bei *bEn*=FALSE wird am Ausgang *lrQ* 0.0 bzw. FALSE bei den booleschen Multiplexern ausgegeben.

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: Aktivierung der Bausteinfunktion.

udiSel: Nummer des Eingangs, dessen Wert am Ausgang *lrQ* ausgegeben werden soll.

lr00...lr16: Eingangswerte, aus denen gewählt werden soll.

VAR_OUTPUT

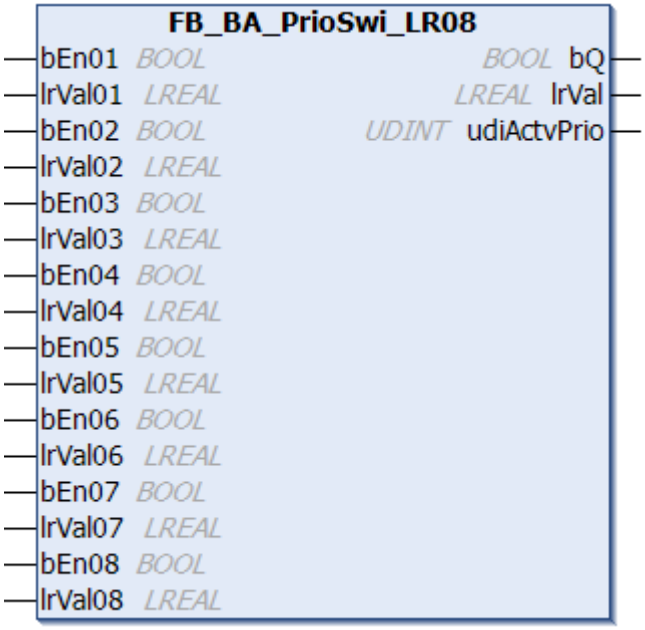
```
lrQ   : LREAL;
```

lrQ: Wert des gewählten Eingangs.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.4 FB_BA_PrioSwi_XX



Die Prioritätenschalter existieren für verschiedene Variablentypen (BOOL, INT, LREAL, REAL, USINT, UINT, UDINT und DINT) und in verschiedenen Ausgangsgrößen (4, 8, 12 und 16 bzw. 24), haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_PrioSwi_LR08 beschrieben.

Zur Selektion verschiedener Werte stehen Prioritätenschalter zur Verfügung. Zum Ausgang *lrVal* wird derjenige Wert mit der höchsten Priorität durchgeschaltet dessen Eingang *bEnxx* TRUE ist.

Beispiel:

Eingänge			Ausgänge		
bEn01	FALSE		bQ	TRUE	
lrVal01		32.5	lrVal		5.84
bEn02	FALSE		udiActvPrio		3
lrVal02		17.4			
bEn03	TRUE				
lrVal03		5.84			
bEn04	TRUE				
lrVal04		9.56			
bEn05	FALSE				
lrVal05		16.5			
bEn06	TRUE				
lrVal06		32.781			
bEn07	FALSE				
lrVal07		25.4			
bEn08	TRUE				
lrVal08		44.5			

Ist keine der Prioritäten freigegeben, so fällt der Ausgang *bQ* auf FALSE. An den Ausgängen *lrVal* und *udiActvPrio* wird jeweils 0 ausgegeben. Bei einem booleschen Prioritätenschalter wird an dem Ausgang *bVal* dann FALSE ausgegeben.

Eingänge			Ausgänge		
bEn01	FALSE		bQ	FALSE	
lrVal01		32.5	lrVal		0.0
bEn02	FALSE		udiActvPrio		0
lrVal02		17.4			
bEn03	FALSE				
lrVal03		5.84			
bEn04	FALSE				
lrVal04		9.56			
bEn05	FALSE				
lrVal05		16.5			
bEn06	FALSE				
lrVal06		32.781			
bEn07	FALSE				
lrVal07		25.4			
bEn08	FALSE				
lrVal08		44.5			

Wenn keine aktive Priorität ansteht, dann wird am Ausgang *udiActvPrio* der Wert der globalen Konstanten *ConstudiNoActvPrio* ausgegeben.

VAR_INPUT

```
bEn01   : BOOL;
lrVal01 : LREAL;
bEn02   : BOOL;
lrVal02 : LREAL;
bEn03   : BOOL;
lrVal03 : LREAL;
bEn04   : BOOL;
lrVal04 : LREAL;
bEn05   : BOOL;
lrVal05 : LREAL;
bEn06   : BOOL;
lrVal06 : LREAL;
```

```
bEn07   : BOOL;
lrVal07 : LREAL;
bEn08   : BOOL;
lrVal08 : LREAL;
```

bEn01...bEn08: Freigabe des Prioritätswertes.

lrVal01...lrVal08: Prioritätswert.

VAR_OUTPUT

```
bQ          : BOOL;
lrVal       : LREAL;
udiActvPrio : UDINT;
```

bQ: Ausgabe, ob überhaupt eine Priorität freigegeben ist.

lrVal: Ausbabe des Wertes der aktuellen (höchsten) Priorität, die freigegeben ist.

udiActvPrio: aktuelle (höchste) Priorität, die freigegeben ist.

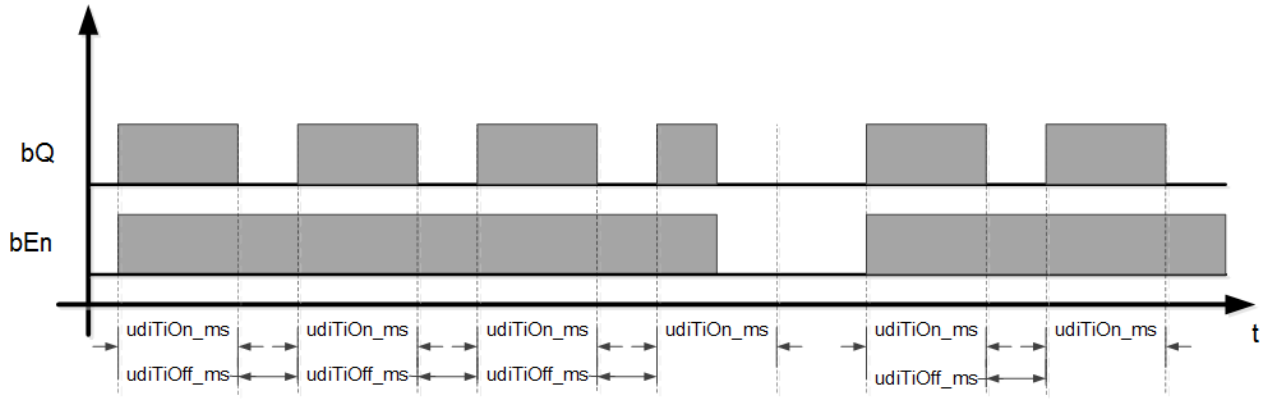
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.5 FB_BA_Blink



Dieser Baustein ist ein Oszillator mit einstellbarer Puls- und Pausenzeit, *udiTiOn_ms* und *udiTiOff_ms* [ms]. Er wird mit einem TRUE-Signal an *bEn* freigegeben und beginnt mit der Puls-Phase.



udiTiNextSwi_sec ist ein Countdown [s] zum nächsten Wechsel von *bQ*.

VAR_INPUT

```
bEn          : BOOL;
udiTiOn_ms   : UDINT;
udiTiOff_ms  : UDINT;
```

bEn: Bausteinfreigabe.

udiTiOn_ms: Puls-Zeit [ms].

udiTiOff_ms: Pausenzeit [ms].

VAR_OUTPUT

```
bQ               : BOOL;
udiTiNextSwi_sec : UDINT;
```

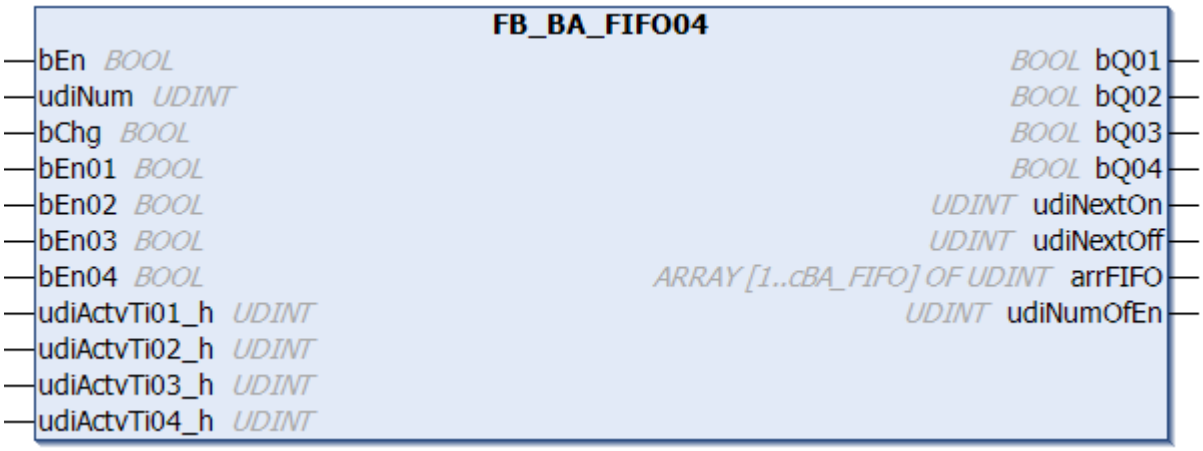
bQ: Oszillatorausgang.

udiTiNextSwi_sec: Countdown bis zum nächsten Wechsel von *bQ* [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.6 FB_BA_FIFO04



Der Funktionsbaustein *FB_BA_FIFO04* ermöglicht eine Folgesteuerung von bis zu vier Aggregaten mit automatischem Wechsel der Einschaltreihenfolge nach Betriebsstunden.
Der Funktionsbaustein steht in zwei Ausführungen zur Verfügung: für eine Sequenz von vier und von acht [▶ 663] Aggregaten.
In der Abfolge haben die Aggregate mit den wenigen Betriebsstunden Vorrang vor den mit vielen Betriebsstunden.

Eine steigende Flanke an *bChg* erzwingt einen Folgewechsel. Die Aggregate mit den geringsten Betriebsstunden werden in der FIFO nach vorne gesetzt und somit vorrangig eingeschaltet.
In der Folge werden nur Aggregate eingetragen, welche an den Eingängen *bEn01*..*bEn04* frei gegeben sind.
udiNum gibt die Anzahl der angeforderten Aggregate an.
An den Eingängen *udiActvTi01_h* bis *udiActvTi04_h* werden die Betriebsstunden der Aggregate eingetragen.
Werden diese Eingänge alle konstant auf Null gesetzt, erfolgt der Folgewechsel nur in Abhängigkeit von *bChg* zyklisch gesteuert.
Dabei fällt immer das erste Aggregat aus dem FIFO heraus, die anderen werden aufgeschoben, und das erste Aggregat hinten in dem FIFO wieder angehängt. Somit rotieren die Aggregate in ihrer Folge.
Werden am Eingang *udiNum* mehr Aggregate angefordert als an den Eingängen *bEn01* bis *bEn04* zur Verfügung stehen, wird dieses an *bErr* mit einem TRUE signalisiert.

Fehlerbehandlung

Werden am Eingang *udiNum* mehr Aggregate angefordert als an den Eingängen *bEn01* bis *bEn04* zur Verfügung stehen, wird dieses an *bErr* mit einem TRUE signalisiert.

VAR_INPUT

```
bEn           : BOOL;
udiNum        : UDINT;
bChg          : BOOL;
bEn01         : BOOL;
bEn02         : BOOL;
bEn03         : BOOL;
bEn04         : BOOL;
udiActvTi01_h : UDINT;
udiActvTi02_h : UDINT;
udiActvTi03_h : UDINT;
udiActvTi04_h : UDINT;
```

bEn: Freigabe des Bausteins.

udiNum: Anzahl der Aggregate.

bChg: Folgewechsel erzwingen.

bEn01...bEn04: Freigabe Aggregat 1...Freigabe Aggregat 4.

udiActvTi01_h...udiActvTi04_h: Betriebsstunden Aggregat 1...Betriebsstunden Aggregat 4.

VAR_OUTPUT

```
bQ01         : BOOL;
bQ02         : BOOL;
bQ03         : BOOL;
bQ04         : BOOL;
udiNextOn    : UDINT;
udiNextOff   : UDINT;
arrFIFO      : ARRAY [1..4] OF UDINT;
udiNumOfEn   : UDINT;
bErr         : BOOL;
sErrDesc     : STRING;
```

bQ01...bQ04: Schaltet Aggregat 1..4.

udiNextOn: Nummer des Aggregates, welches als nächstes eingeschaltet wird.

udiNextOff: Nummer des Aggregates, welches als nächstes eingeschaltet wird.

arrFIFO: FIFO-Speicher als Feld.

udiNumOfEn: Anzahl Teilnehmer in Abhängigkeit der einzelnen Freigaben.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

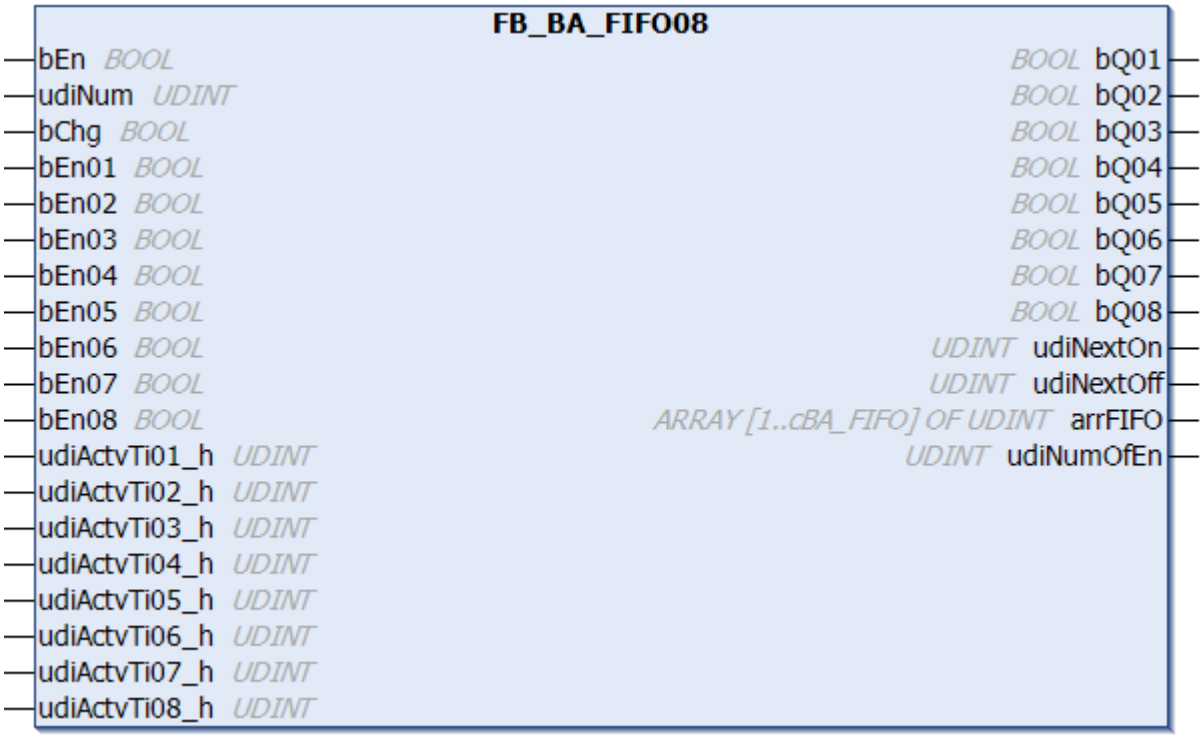
sErrDesc: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Warnung: Am Eingang udiNum sind mehr als 4 Teilnehmer eingetragen. Diese Anzahl wird auf die Zahl begrenzt, die an den Eingängen bEn01..bEn04 freigegeben sind.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.7 FB_BA_FIFO08



Der Funktionsbaustein *FB_BA_FIFO08* ermöglicht eine Folgesteuerung von bis zu acht Aggregaten mit automatischem Wechsel der Einschaltreihenfolge nach Betriebsstunden.
Der Funktionsbaustein steht in zwei Ausführungen zur Verfügung: für eine Sequenz von vier [▶ 661] und von acht Aggregaten.

In der Abfolge haben die Aggregate mit den wenigen Betriebsstunden Vorrang vor den mit vielen Betriebsstunden.

Eine steigende Flanke an *bChg* erzwingt einen Folgewechsel. Die Aggregate mit den geringsten Betriebsstunden werden in der FIFO nach vorne gesetzt und somit vorrangig eingeschaltet.
In der Folge werden nur Aggregate eingetragen, welche an den Eingängen *bEn01..bEn08* frei gegeben sind. *udiNum* gibt die Anzahl der angeforderten Aggregate an.
An den Eingängen *udiActvTi01_h* bis *udiActvTi08_h* werden die Betriebsstunden der Aggregate eingetragen. Werden diese Eingänge alle konstant auf Null gesetzt, erfolgt der Folgewechsel nur in Abhängigkeit von *bChg* zyklisch gesteuert.
Dabei fällt immer das erste Aggregat aus dem FIFO heraus, die anderen werden aufgeschoben, und das erste Aggregat hinten in dem FIFO wieder angehängt. Somit rotieren die Aggregate in ihrer Folge.
Werden am Eingang *udiNum* mehr Aggregate angefordert als an den Eingängen *bEn01* bis *bEn08* zur Verfügung stehen, wird dieses an *bErr* mit einem TRUE signalisiert.

Fehlerbehandlung

Werden am Eingang *udiNum* mehr Aggregate angefordert als an den Eingängen *bEn01* bis *bEn08* zur Verfügung stehen, wird dieses an *bErr* mit einem TRUE signalisiert.

VAR_INPUT

```
bEn           : BOOL;
udiNum        : UDINT;
bChg          : BOOL;
bEn01         : BOOL;
bEn02         : BOOL;
bEn03         : BOOL;
bEn04         : BOOL;
bEn05         : BOOL;
bEn06         : BOOL;
bEn07         : BOOL;
bEn08         : BOOL;
udiActvTi01_h : UDINT;
```

```
udiActvTi02_h : UDINT;
udiActvTi03_h : UDINT;
udiActvTi04_h : UDINT;
udiActvTi05_h : UDINT;
udiActvTi06_h : UDINT;
udiActvTi07_h : UDINT;
udiActvTi08_h : UDINT;
```

bEn: Freigabe des Bausteins.

udiNum: Anzahl der Aggregate.

bChg: Folgewechsel erzwingen.

bEn01...bEn08: Freigabe Aggregat 1...Freigabe Aggregat 8.

udiActvTi01_h...udiActvTi08_h: Betriebsstunden Aggregat 1...Betriebsstunden Aggregat 8.

VAR_OUTPUT

```
bQ01        : BOOL;
bQ02        : BOOL;
bQ03        : BOOL;
bQ04        : BOOL;
bQ05        : BOOL;
bQ06        : BOOL;
bQ07        : BOOL;
bQ08        : BOOL;
udiNextOn   : UDINT;
udiNextOff  : UDINT;
arrFIFO     : ARRAY [1..8] OF UDINT;
udiNumOfEn  : UDINT;
bErr        : BOOL;
sErrDescr   : STRING;
```

bQ01...bQ08: Schaltet Aggregat 1..8.

udiNextOn: Nummer des Aggregates, welches als nächstes eingeschaltet wird.

udiNextOff: Nummer des Aggregates, welches als nächstes eingeschaltet wird.

arrFIFO: FIFO-Speicher als Feld.

udiNumOfEn: Anzahl Teilnehmer in Abhängigkeit der einzelnen Freigaben.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

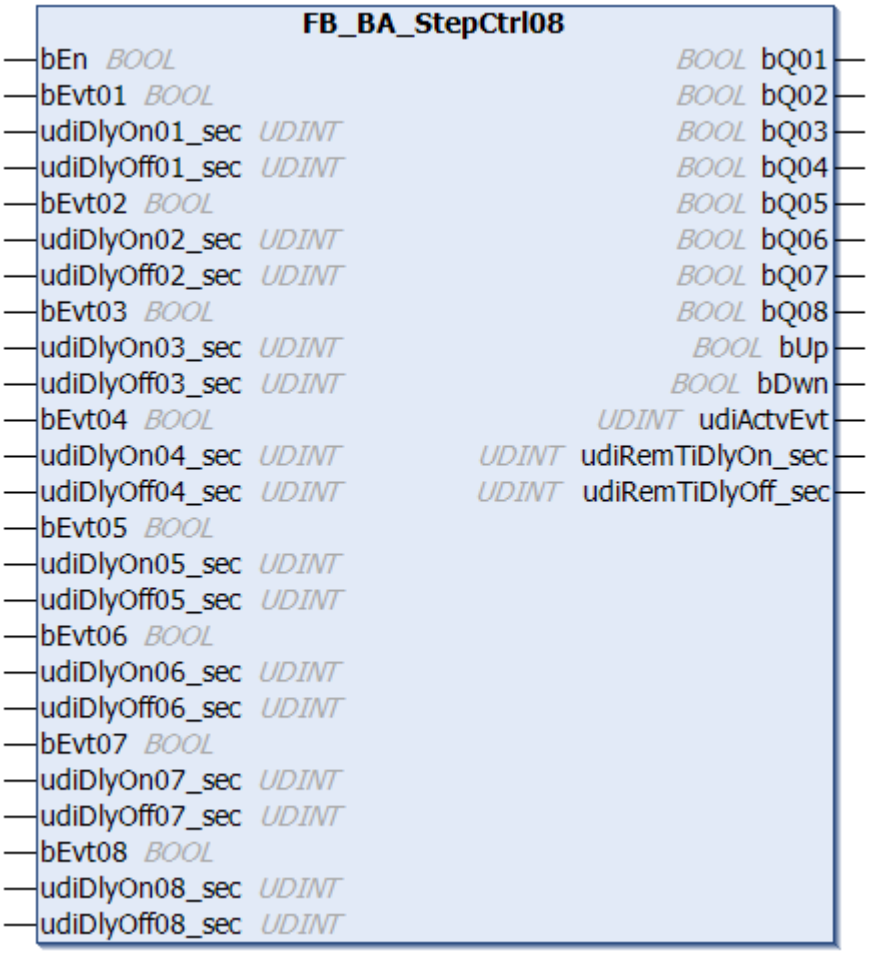
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Warnung: Am Eingang udiNum sind mehr als 8 Teilnehmer eingetragen. Diese Anzahl wird auf die Zahl begrenzt, die an den Eingängen bEn01..bEn08 freigegeben sind.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.8 FB_BA_StepCtrl08

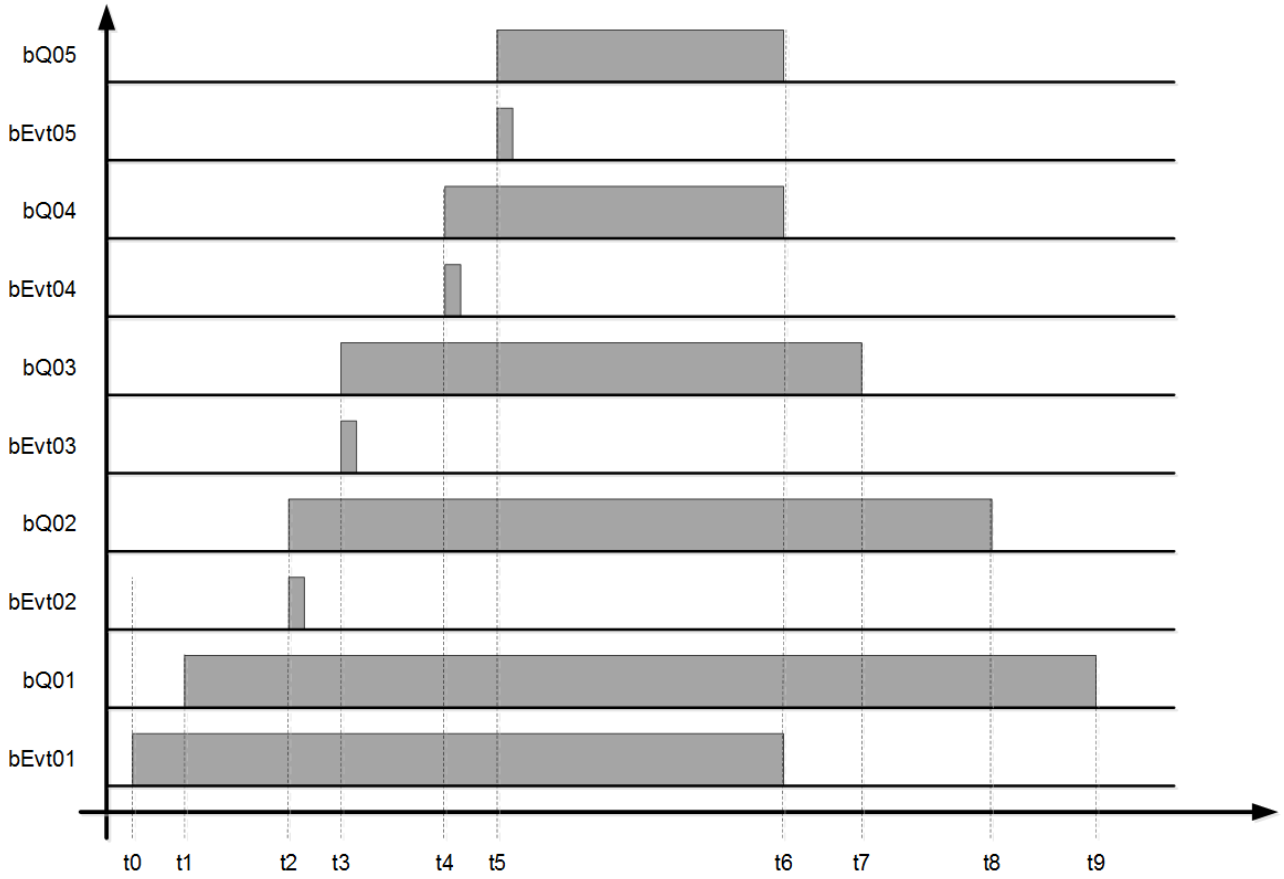


Der Funktionsbaustein dient zur Ausgabe sequenziell folgender Steuerungsbefehle. Eine typische Anwendung dieses Bausteins ist der Startvorgang einer raumlufttechnischen Anlage. *bEn* ist für die generelle Freigabe des Funktionsbausteins. Bei *bEn* = FALSE werden alle Ausgänge von *bQ01* bis *bQ08* auf FALSE gesetzt. Das Starten der Steuersequenz erfolgt am Eingang *bEvt01*. Nach dem Ablauf des Timers *udiDlyOn01_sec,* siehe Parameter, wird der zugehörige Ausgang *bQ01* gesetzt. Weitere Stufen schalten nach einer steigenden Flanke an den Eingängen *bEvt02* bis *bEvt08* jeweils Zeitverzögert über die Timer *udiDlyOn02_sec* bis *udiDlyOn08_sec* zu. Wird *bEvt01* bei hochgelaufener Steuerkette FALSE dann schaltet die Steuersequenz in umgekehrter Reihenfolge zurück. Das Abschalten der Ausgänge wird durch die Timer *udiDlyOff01_sec* bis *udiDlyOff08_sec*, siehe Parameter, verzögert.

Die Ausgänge *bUp* und *bDwn* zeigen an ob sich die Steuerkette im aufsteigenden oder fallenden Zustand befindet. Die Variable *udiActvEvt* zeigt an in welchen Schritt sich die Steuerkette aktuell befindet. Dabei bedeutet "0", dass die Schrittkette nicht aktiv ist.

Der Ausgang *udiRemTiDlyOn_sec* zeigt beim Hochschalten der Steuerkette die verbleibende Zeit bis zum nächsten Schritt an. Der Ausgang *udiRemTiDlyOff_sec* zeigt beim Runterschalten der Steuerkette die verbleibende Zeit bis zum Umschalten in den nächst niedrigeren Schritt an.

Beispiel



- t0 Einschalten der Schrittkette
- t1 Einschalten Schritt 1 *udiDlyOn01_sec* = t1 - t0
- t2 Ereignis Freigabe Schritt 2, Einschalten Schritt 2, *udiDlyOn02_sec* = 0
- t3 Ereignis Freigabe Schritt 3, Einschalten Schritt 3, *udiDlyOn03_sec* = 0
- t4 Ereignis Freigabe Schritt 4, Einschalten Schritt 4, *udiDlyOn04_sec* = 0
- t5 Ereignis Freigabe Schritt 5, Einschalten Schritt 5, *udiDlyOn05_sec* = 0
- t6 Abschalten der Schrittkette, Abschalten Schritt 5, Abschalten Schritt 4; *udiDlyOff05_sec* = 0, *udiDlyOff04_sec* = 0
- t7 Abschalten Schritt 3, *udiDlyOff03_sec* = t7 -t6
- t8 Abschalten Schritt 2, *udiDlyOff02_sec* = t8 -t7
- t9 Abschalten Schritt 1, *udiDlyOff01_sec* = t9 -t8

VAR_INPUT

```
bEn             : BOOL;
bEvt01          : BOOL;
udiDlyOn01_sec  : UDINT;
udiDlyOff01_sec : UDINT;
bEvt02          : BOOL;
udiDlyOn02_sec  : UDINT;
udiDlyOff02_sec : UDINT;
bEvt03          : BOOL;
udiDlyOn03_sec  : UDINT;
udiDlyOff03_sec : UDINT;
bEvt04          : BOOL;
udiDlyOn04_sec  : UDINT;
udiDlyOff04_sec : UDINT;
bEvt05          : BOOL;
udiDlyOn05_sec  : UDINT;
udiDlyOff05_sec : UDINT;
bEvt06          : BOOL;
udiDlyOn06_sec  : UDINT;
udiDlyOff06_sec : UDINT;
bEvt07          : BOOL;
```

```
udiDlyOn07_sec  : UDINT;
udiDlyOff07_sec : UDINT;
bEvt08          : BOOL;
udiDlyOn08_sec  : UDINT;
udiDlyOff08_sec : UDINT;
```

bEn: Freigabe des Funktionsbausteins.

bEvt01..08: Einschaltbefehl für Schritt 1 bis 8.

udiDlyOn01..08_sec: Einschaltverzögerung für Ausgang *bQ01 .. 08* [s].

udiDlyOff01..08_sec: Ausschaltverzögerung für Ausgang *bQ01 .. 08* [s].

VAR_OUTPUT

```
bQ01               : BOOL;
bQ02               : BOOL;
bQ03               : BOOL;
bQ04               : BOOL;
bQ05               : BOOL;
bQ06               : BOOL;
bQ07               : BOOL;
bQ08               : BOOL;
bUp                : BOOL;
bDwn               : BOOL;
udiActvEvt         : UDINT;
udiRemTiDlyOn_sec  : UDINT;
udiRemTiDlyOff_sec : UDINT;
```

bQ01..08: Schritt 1 bis 8 Ein.

bUp: Steuerkette ist im aufsteigenden Zustand.

bDwn: Steuerkette ist im fallenden Zustand.

udiActvEvt: Aktiver Schritt, Anzeige 0 .. 8, wobei "0" für eine nicht aktive Schrittkette steht.

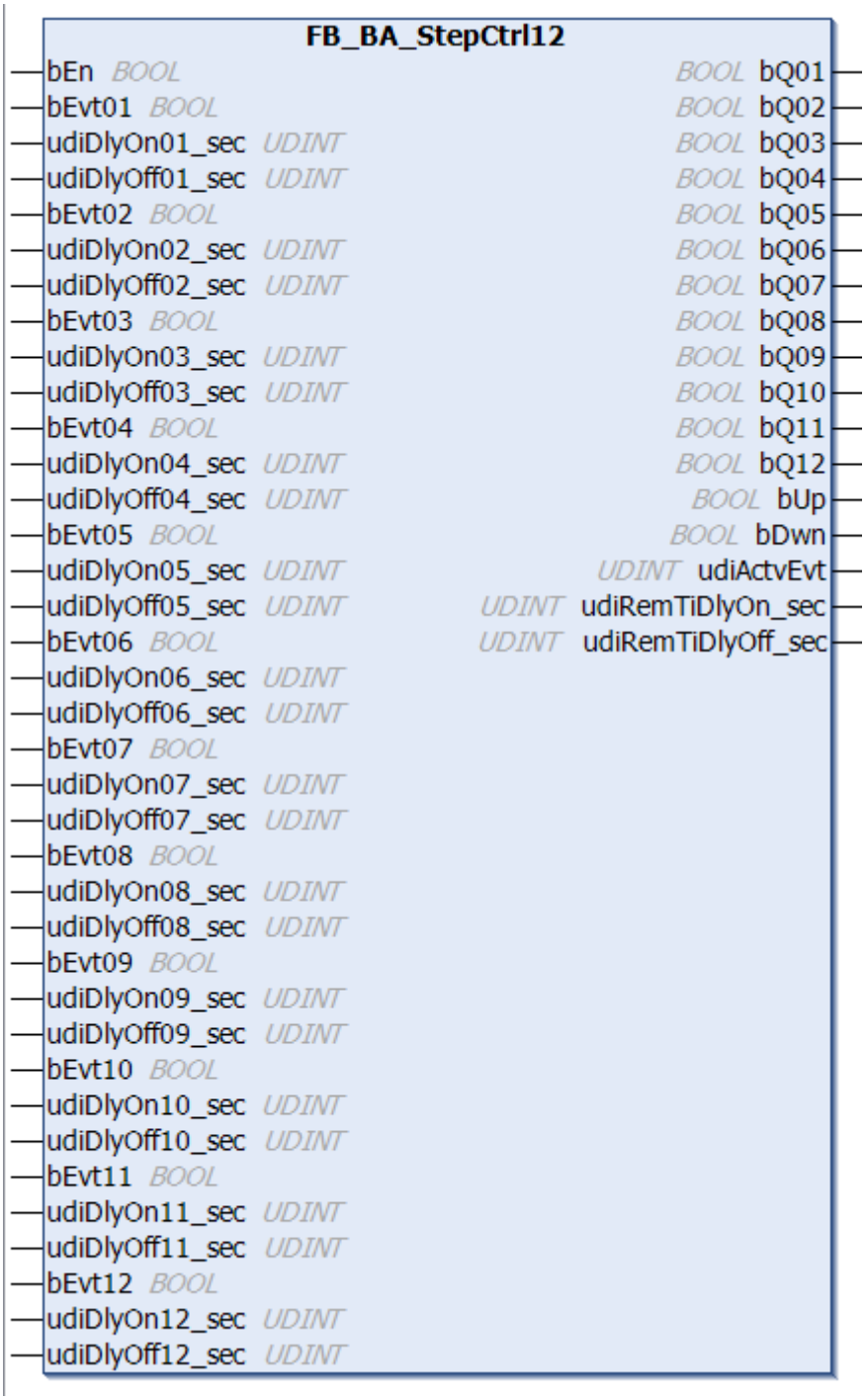
udiRTiDlyOn: Restzeit bis zum Hochschalten in den nächsten Schritt [s].

udiRTiDlyOff: Restzeit bis zum Runterschalten in den vorherigen Schritt [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.9 FB_BA_StepCtrl12

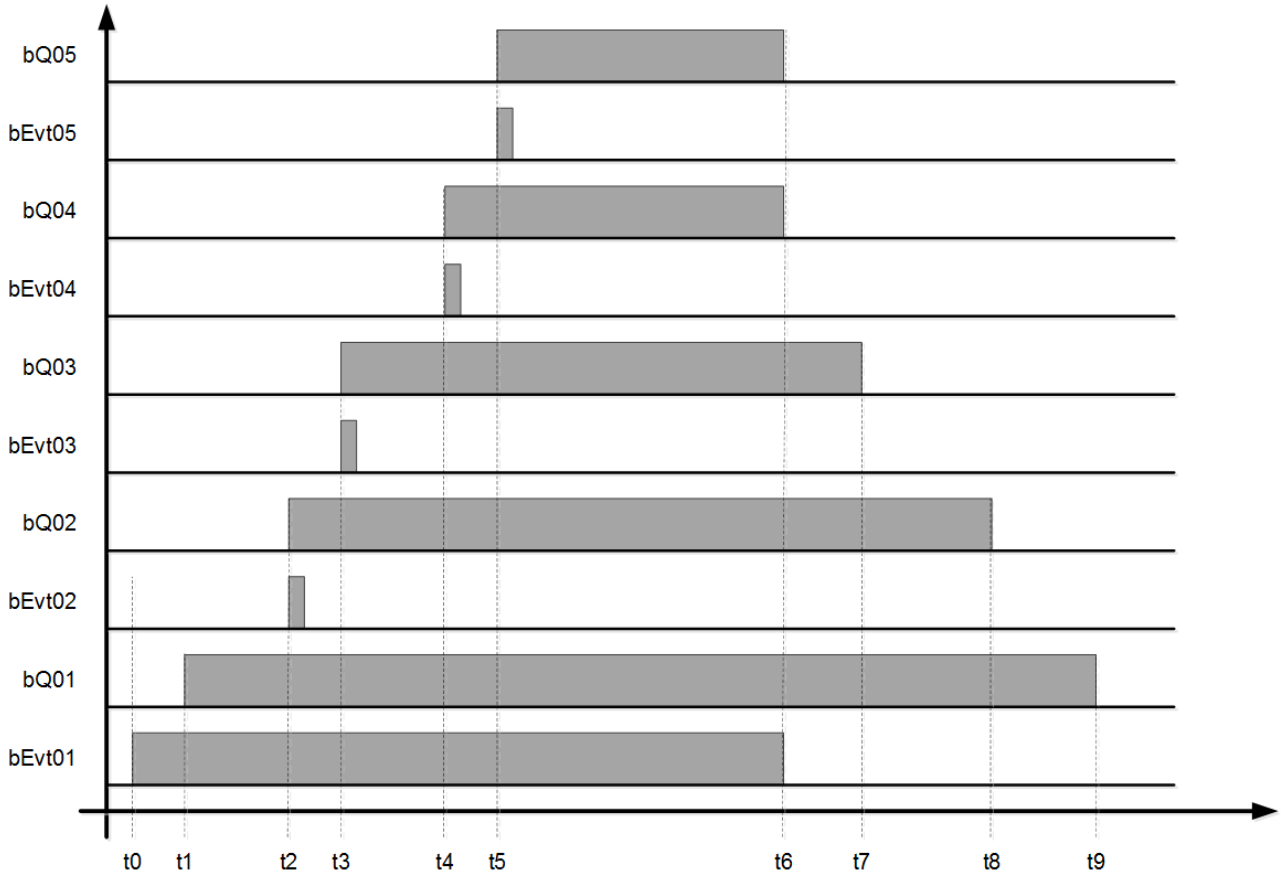


Der Funktionsbaustein dient zur Ausgabe sequenziell folgender Steuerungsbefehle. Eine typische Anwendung dieses Bausteins ist der Startvorgang einer raumlufttechnischen Anlage. *bEn* ist für die generelle Freigabe des Funktionsbausteins. Bei *bEn* = FALSE werden alle Ausgänge von *bQ01* bis *bQ12* auf FALSE gesetzt. Das Starten der Steuersequenz erfolgt am Eingang *bEvt01*. Nach dem Ablauf des Timers *udiDlyOn01_sec,* siehe Parameter, wird der zugehörige Ausgang *bQ01* gesetzt. Weitere Stufen schalten nach einer steigenden Flanke an den Eingängen *bEvt02* bis *bEvt12* jeweils zeitverzögert über die Timer *udiDlyOn02_sec* bis *udiDlyOn12_sec* zu. Wird *bEvt01* bei hochgelaufener Steuerkette FALSE dann schaltet die Steuersequenz in umgekehrter Reihenfolge zurück. Das Abschalten der Ausgänge wird durch die Timer *udiDlyOff01_sec* bis *udiDlyOff12_sec*, siehe Parameter, verzögert.

Die Ausgänge *bUp* und *bDwn* zeigen an ob sich die Steuerkette im aufsteigenden oder fallenden Zustand befindet. Die Variable *udiActvEvt* zeigt an in welchen Schritt sich die Steuerkette aktuell befindet. Dabei bedeutet "0", dass die Schrittkette nicht aktiv ist.

Der Ausgang *udiRemTiDlyOn_sec* zeigt beim Hochschalten der Steuerkette die verbleibende Zeit bis zum nächsten Schritt an. Der Ausgang *udiRemTiDlyOff_sec* zeigt beim Runterschalten der Steuerkette die verbleibende Zeit bis zum Umschalten in den nächst niedrigeren Schritt an.

Beispiel



- t0 Einschalten der Schrittkette
- t1 Einschalten Schritt 1 *udiDlyOn01_sec* = t1 - t0
- t2 Ereignis Freigabe Schritt 2, Einschalten Schritt 2, *udiDlyOn02_sec* = 0
- t3 Ereignis Freigabe Schritt 3, Einschalten Schritt 3, *udiDlyOn03_sec* = 0
- t4 Ereignis Freigabe Schritt 4, Einschalten Schritt 4, *udiDlyOn04_sec* = 0
- t5 Ereignis Freigabe Schritt 5, Einschalten Schritt 5, *udiDlyOn05_sec* = 0
- t6 Abschalten der Schrittkette, Abschalten Schritt 5, Abschalten Schritt 4; *udiDlyOff05_sec* = 0, *udiDlyOff04_sec* = 0
- t7 Abschalten Schritt 3, *udiDlyOff03_sec* = t7 -t6
- t8 Abschalten Schritt 2, *udiDlyOff02_sec* = t8 -t7
- t9 Abschalten Schritt 1, *udiDlyOff01_sec* = t9 -t8

VAR_INPUT

```
bEn               : BOOL;
bEvt01            : BOOL;
udiDlyOn01_sec    : UDINT;
udiDlyOff01_sec   : UDINT;
bEvt02            : BOOL;
udiDlyOn02_sec    : UDINT;
udiDlyOff02_sec   : UDINT;
bEvt03            : BOOL;
udiDlyOn03_sec    : UDINT;
udiDlyOff03_sec   : UDINT;
bEvt04            : BOOL;
udiDlyOn04_sec    : UDINT;
udiDlyOff04_sec   : UDINT;
bEvt05            : BOOL;
```

```
udiDlyOn05_sec   : UDINT;
udiDlyOff05_sec  : UDINT;
bEvt06           : BOOL;
udiDlyOn06_sec   : UDINT;
udiDlyOff06_sec  : UDINT;
bEvt07           : BOOL;
udiDlyOn07_sec   : UDINT;
udiDlyOff07_sec  : UDINT;
bEvt08           : BOOL;
udiDlyOn08_sec   : UDINT;
udiDlyOff08_sec  : UDINT;
bEvt09           : BOOL;
udiDlyOn09_sec   : UDINT;
udiDlyOff09_sec  : UDINT;
bEvt10           : BOOL;
udiDlyOn10_sec   : UDINT;
udiDlyOff10_sec  : UDINT;
bEvt11           : BOOL;
udiDlyOn11_sec   : UDINT;
udiDlyOff11_sec  : UDINT;
bEvt12           : BOOL;
udiDlyOn12_sec   : UDINT;
udiDlyOff12_sec  : UDINT;
```

bEn: Freigabe des Funktionsbausteins.

bEvt01..12: Einschaltbefehl für Schritt 1 bis 12.

udiDlyOn01..12_sec: Einschaltverzögerung für Ausgang *bQ01 .. 12* [s].

udiDlyOff01..12_sec: Ausschaltverzögerung für Ausgang *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: Schritt 1 bis 12 Ein.

bUp: Steuerkette ist im aufsteigenden Zustand.

bDwn: Steuerkette ist im fallenden Zustand.

udiActvEvt: Aktiver Schritt, Anzeige 0 .. 12, wobei "0" für eine nicht aktive Schrittkette steht.

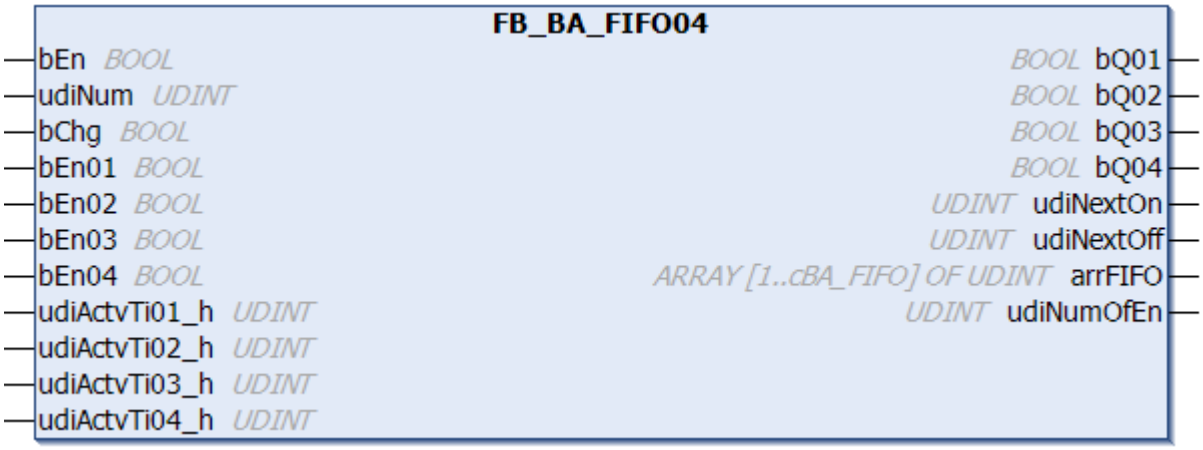
udiRTiDlyOn: Restzeit bis zum Hochschalten in den nächsten Schritt [s].

udiRTiDlyOff: Restzeit bis zum Runterschalten in den vorherigen Schritt [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.10 FB_BA_FIFO04_XX



Dieser Funktionsbaustein dient zum Auswerten des FiFo-Speichers aus dem FB_BA_FIFO04 [▶ 661]. Die Eingänge werden nach FIFO-Tabelle auf die entsprechenden Ausgänge des verwendeten Bausteins *FB_BA_FIFO04_BOOL* oder *FB_BA_FIFO04_REAL* verknüpft.

Beispiel:

Im Beispiel steht in dem Array: 4,3,1,2,0,0,0,0. Als Ergebnis wird im *FB_BA_FIFO04_REAL* folgendes ausgegeben:

rIn01 auf den Ausgang *rVal04*

rIn02 auf den Ausgang *rVal03*

rIn03 auf den Ausgang *rVal01*

rIn04 auf den Ausgang *rVal02*

VAR_INPUT

```
arrFIFO          : Array [1..4] OF UDINT;
rIn01 - rIn04    : REAL;
```

arrFIFO: Enthält die Zuordnungstabelle mit maximal acht Werten. Der erste Wert gibt an wohin der erste Eingang kopiert wird, der zweite Wert gibt an wohin der zweite Eingang kopiert wird, etc. Bei "0" wird keine Zuordnung vorgenommen.

rIn01 – rIn04: Sollwerte die verknüpft werden sollen.

VAR_OUTPUT

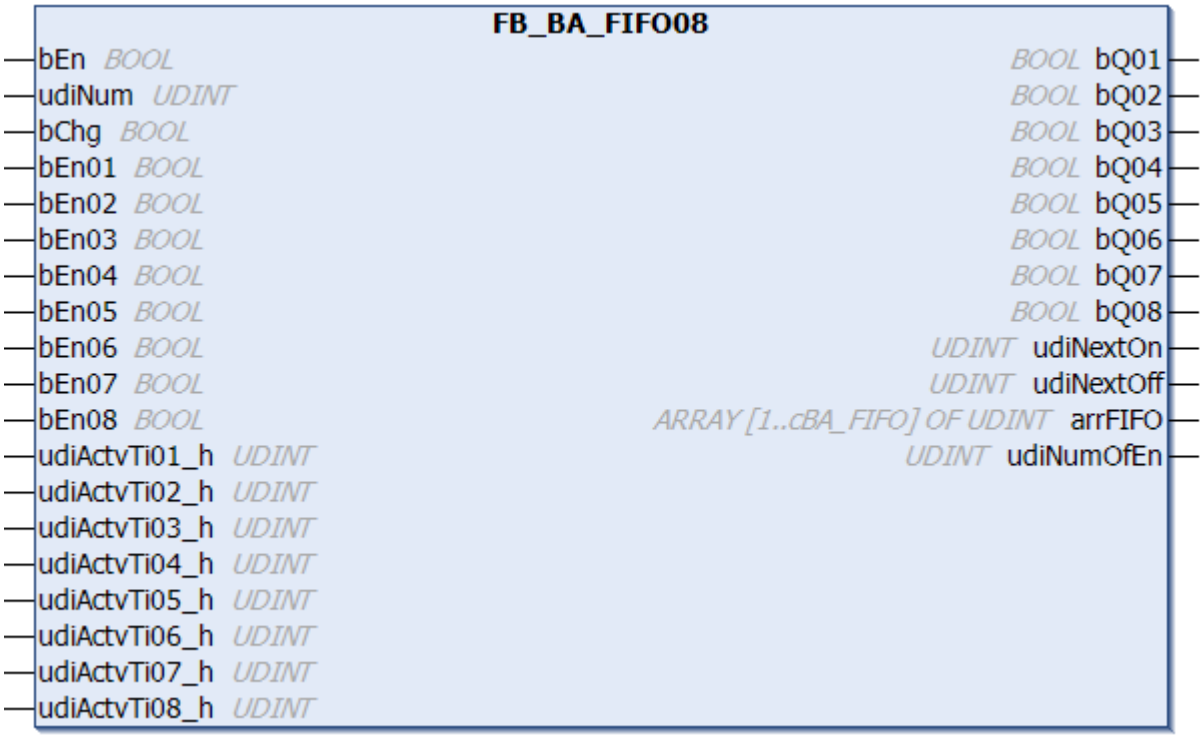
```
rVal01 - rVal04   : REAL;
```

rVal01 – rVal04: Aktor Sollwert, laut FIFO-Tabelle verknüpfter Eingangswert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.3.11 FB_BA_FIFO08_XX



Dieser Funktionsbaustein dient zum Auswerten des FiFo-Speichers aus dem FB_BA_FIFO08 [▶ 663]. Die Eingänge werden nach FIFO-Tabelle auf die entsprechenden Ausgänge des verwendeten Bausteins *FB_BA_FIFO08_BOOL* oder *FB_BA_FIFO08_REAL* verknüpft.

Beispiel:

Im Beispiel steht in dem Array: 4,3,1,2,0,0,0,0. Als Ergebnis wird im *FB_BA_FIFO08_REAL* nun

rIn01 auf den Ausgang *rVal04*

rIn02 auf den Ausgang *rVal03*

rIn03 auf den Ausgang *rVal01*

rIn04 auf den Ausgang *rVal02*

ausgegeben.

VAR_INPUT

```
arrFIFO          : Array [1..8] OF UDINT;
rIn01 - rIn08    : REAL;
```

arrFIFO: Enthält die Zuordnungstabelle mit maximal acht Werten. Der erste Wert gibt an wohin der erste Eingang kopiert wird, der zweite Wert gibt an wohin der zweite Eingang kopiert wird, etc. Bei "0" wird keine Zuordnung vorgenommen.

rIn01 – rIn08: Sollwerte die verknüpft werden sollen.

VAR_OUTPUT

```
rVal01 - rVal08   : REAL;
```

rVal01 – rVal08: Aktor Sollwert, laut FIFO-Tabelle verknüpfter Eingangswert.

Voraussetzungen

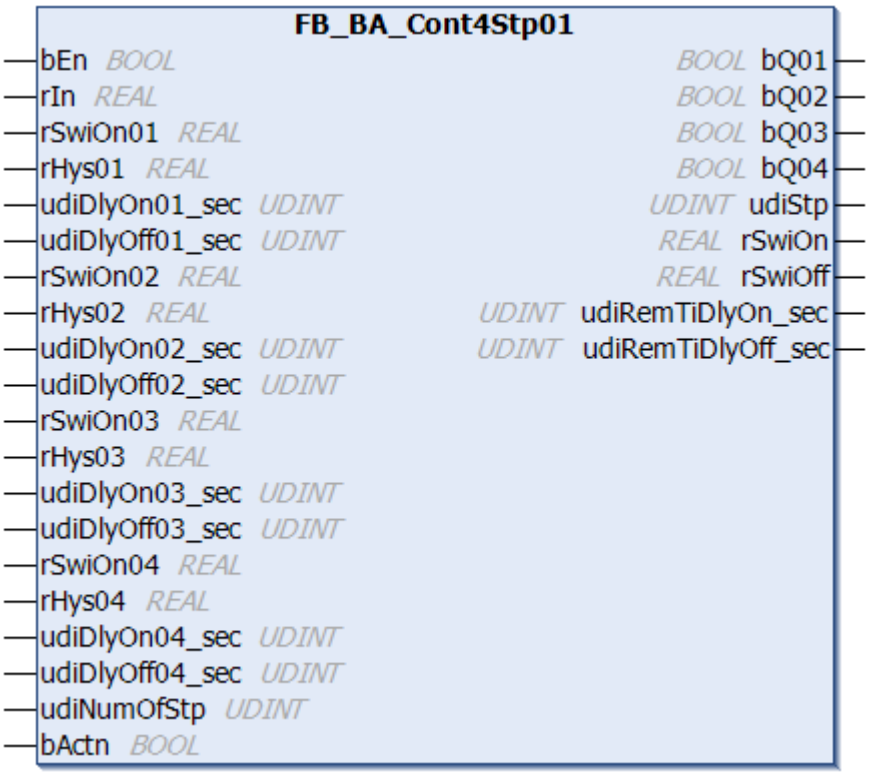
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.4 Hysterese, 2-Punkt-Regelung

Funktionsbausteine

Name	Beschreibung
FB_BA_Cont4Stp01 [▶ 673]	Stufenschalter mit vier Stufen
FB_BA_Swi2P [▶ 678]	Zweipunktschalter
FB_BA_SwiHys2P [▶ 680]	Zweipunktschalter mit einem Schaltpunkt

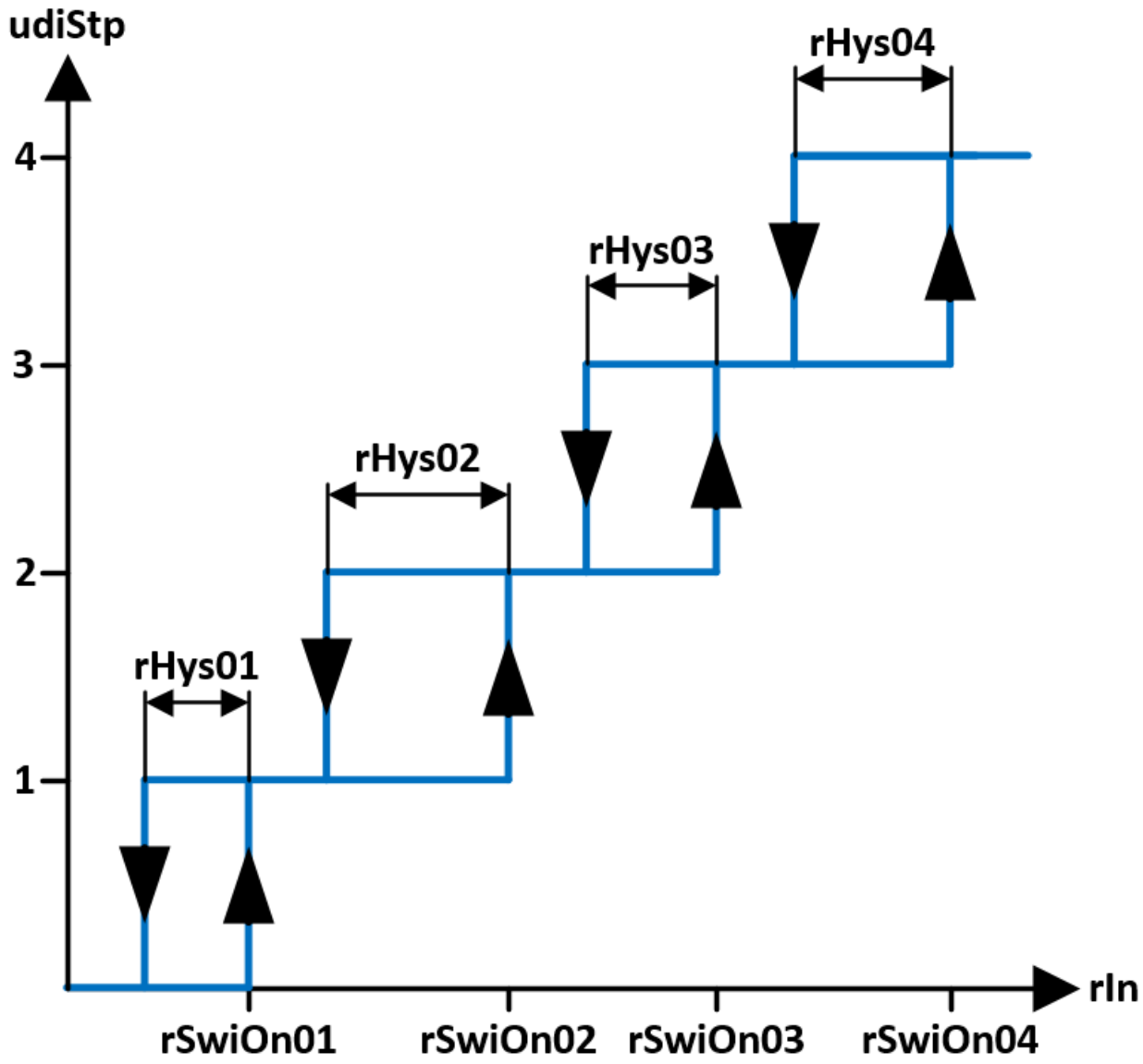
6.1.2.3.2.1.5.4.1 FB_BA_Cont4Stp01



Der Funktionsbaustein ermittelt abhängig vom Eingangssignal die resultierenden Schaltstufen eines mehrstufigen Aggregats.
Es sind vier Einschaltschwellen und vier Hysteresen parametrierbar.

Diagramm 01

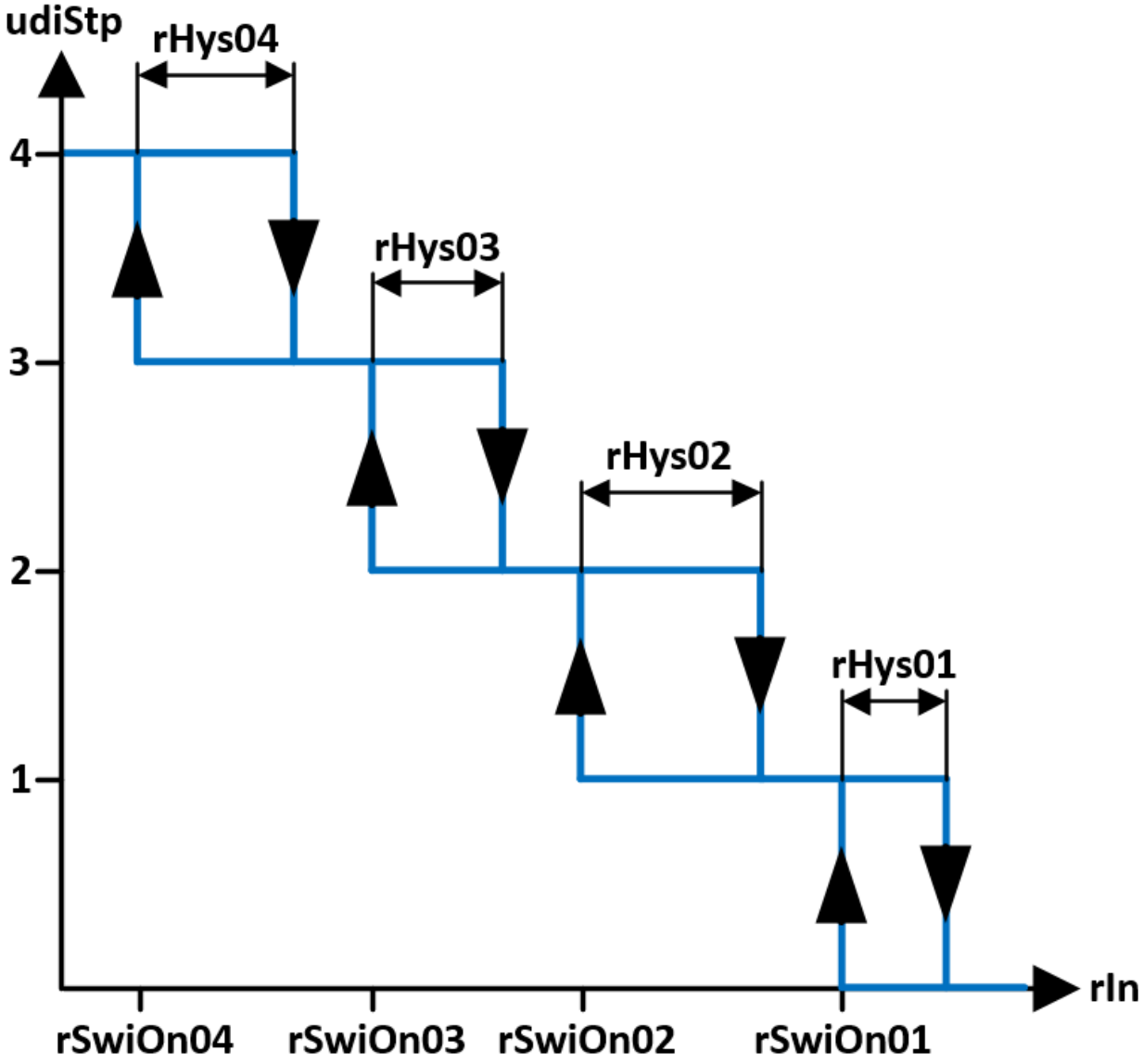
Wirksinn Parameter *bActn* = FALSE = Reverse = Heizen



udiStp	udi-NumOfStp	rSwiOn	rSwiOff	udiRem-TiDly-On_sec	udiRem-TiD-lyOff_sec	bQ01	bQ02	bQ03	bQ04
0	0	rSwiOn01	rSwiOn01 - rHys01	udiDlyOn01_sec	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	rSwiOn02	rSwiOn01 - rHys01	udiDlyOn02_sec	udiDlyOff01_sec	TRUE	FALSE	FALSE	FALSE
2	>= 2	rSwiOn03	rSwiOn02 - rHys02	udiDlyOn03_sec	udiDlyOff02_sec	TRUE	TRUE	FALSE	FALSE
3	>= 3	rSwiOn04	rSwiOn03 - rHys03	udiDlyOn04_sec	udiDlyOff03_sec	TRUE	TRUE	TRUE	FALSE
4	>= 4	rSwiOn04	rSwiOn04 - rHys04	0	udiDlyOff04_sec	TRUE	TRUE	TRUE	TRUE

Diagramm 02

Wirksinn Parameter *bActn* =TRUE = Direct = Kühlen



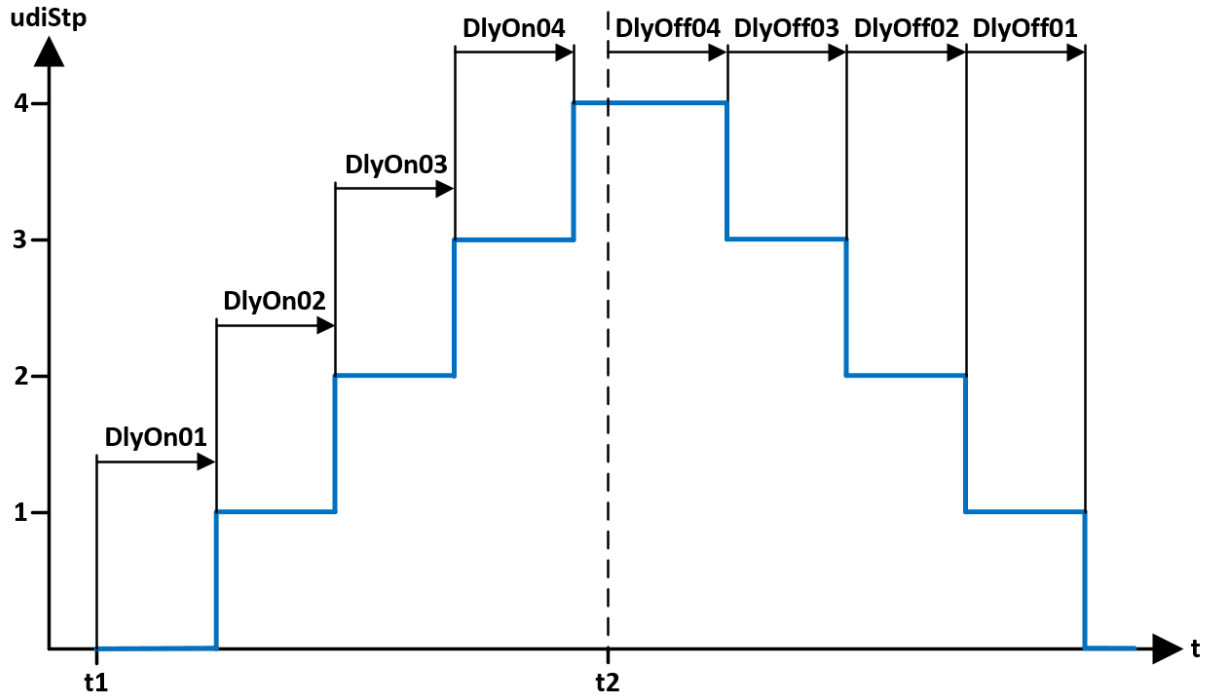
udiStp	udi-NumOfStp	rSwiOn	rSwiOff	udiRem-TiDly-On_sec	udiRem-TiD-lyOff_sec	bQ01	bQ02	bQ03	bQ04
0	0	rSwiOn01	rSwiOn01 + rHys01	udiDlyOn 01_sec	0	FALSE	FALSE	FALSE	FALSE
1	>= 1	rSwiOn02	rSwiOn01 + rHys01	udiDlyOn 02_sec	udiDlyOff 01_sec	TRUE	FALSE	FALSE	FALSE
2	>= 2	rSwiOn03	rSwiOn02 + rHys02	udiDlyOn 03_sec	udiDlyOff 02_sec	TRUE	TRUE	FALSE	FALSE
3	>= 3	rSwiOn04	rSwiOn03 + rHys03	udiDlyOn 04_sec	udiDlyOff 03_sec	TRUE	TRUE	TRUE	FALSE
4	4	rSwiOn04	rSwiOn04 + rHys04	0	udiDlyOff 04_sec	TRUE	TRUE	TRUE	TRUE

Diagramm 03

Zeitverhalten der Ein- und Ausschaltverzögerungen

Zum Zeitpunkt t1 springt *rIn* von *rSwiOn01* auf *rSwiOn04*

Zum Zeitpunkt t2 springt *rIn* von *rSwiOn04*auf *rSwiOn01 – rHys01*



VAR_INPUT

```
bEn              : BOOL;
rIn              : REAL;
rSwiOn01         : REAL
rHys01           : REAL;
udiDlyOn01_sec   : UDINT;
udiDlyOff01_sec  : UDINT;
rSwiOn02         : REAL;
rHys02           : REAL;
udiDlyOn02_sec   : UDINT;
udiDlyOff02_sec  : UDINT;
rSwiOn03         : REAL;
rHys03           : REAL;
udiDlyOn03_sec   : UDINT;
udiDlyOff03_sec  : UDINT;
rSwiOn04         : REAL;
rHys04           : REAL;
udiDlyOn04_sec   : UDINT;
udiDlyOff04_sec  : UDINT;
udiNumOfStp      : UDINT;
bActn            : BOOL;
```

bEn: Generelle Freigabe des Funktionsbausteins. Ist *bEn* = FALSE, so sind sämtliche Ausgänge auf 0 gesetzt.

rIn: Eingangswert von dem der Schaltzustand abgeleitet wird.

rSwiOn01: Einschaltpunkt Stufe 01

rHys01: Absolutwert Hysterese Stufe 01

udiDlyOn01_sec: Einschaltverzögerung Stufe 01

udiDlyOff01_sec: Ausschaltverzögerung Stufe 01

rSwiOn02: Einschaltpunkt Stufe 02

rHys02: Absolutwert Hysterese Stufe 02

udiDlyOn02_sec: Einschaltverzögerung Stufe 02

udiDlyOff02_sec: Ausschaltverzögerung Stufe 02

rSwiOn03: Einschaltpunkt Stufe 03

rHys03: Absolutwert Hysterese Stufe 03

udiDlyOn03_sec: Einschaltverzögerung Stufe 03

udiDlyOff03_sec: Ausschaltverzögerung Stufe 03

rSwiOn04: Einschaltpunkt Stufe 04

rHys04: Absolutwert Hysterese Stufe 04

udiDlyOn04_sec: Einschaltverzögerung Stufe 04

udiDlyOff04_sec: Ausschaltverzögerung Stufe 04

udiNumOfStp: Eingabe der Anzahl der Stufen, die benötigt werden.
Die Eingabe ist begrenzt von 0 bis 4

bActn: Eingangsvariable mit der der Wirksinn des Stufenschalters bestimmt wird.
TRUE = Direct = Kühlen; FALSE = Reverse = Heizen

VAR_OUTPUT

```
bQ01               : BOOL;
bQ02               : BOOL;
bQ03               : BOOL;
bQ04               : BOOL;
udiStp             : UDINT;
rSwiOn             : REAL;
rSwiOff            : REAL;
udiRemTiDlyOn_sec  : UDINT;
udiRemTiDlyOff_sec : UDINT;
```

bQ01: Anzeige des Status Stufe 01
TRUE = EIN; FALSE = AUS
udiStp >= 1

bQ02: Anzeige des Status Stufe 02
TRUE = EIN; FALSE = AUS
udiStp >= 2

bQ03: Anzeige des Status Stufe 03
TRUE = EIN; FALSE = AUS
udiStp >= 3

bQ04: Anzeige des Status Stufe 04
TRUE = EIN; FALSE = AUS
udiStp >= 4

udiStp: Anzeige in welcher Stufe sich der Stufenschalter befindet

rSwiOn: Anzeige des nächsten Einschaltpunktes

rSwiOff: Anzeige des nächsten Ausschaltpunktes

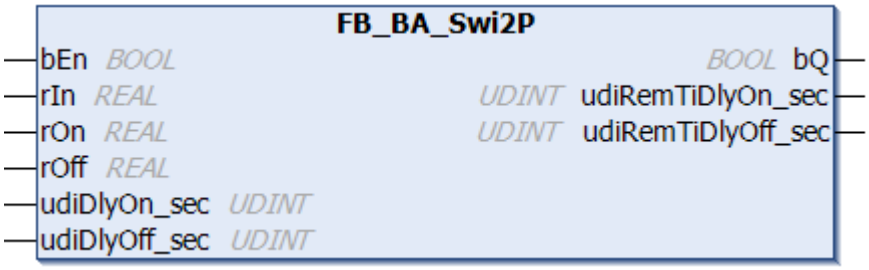
udiRemTiDlyOn_sec: Ist der Einschaltpunkt für das Weiterschalten auf die nächste Stufe erfüllt, dann wird hier der Ablauf der Einschaltverzögerungszeit angezeigt.

udiRemTiDlyOff_sec: Ist der Ausschaltpunkt für das Herunterschalten auf die nächste Stufe erfüllt, dann wird hier der Ablauf der Ausschaltverzögerungszeit angezeigt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

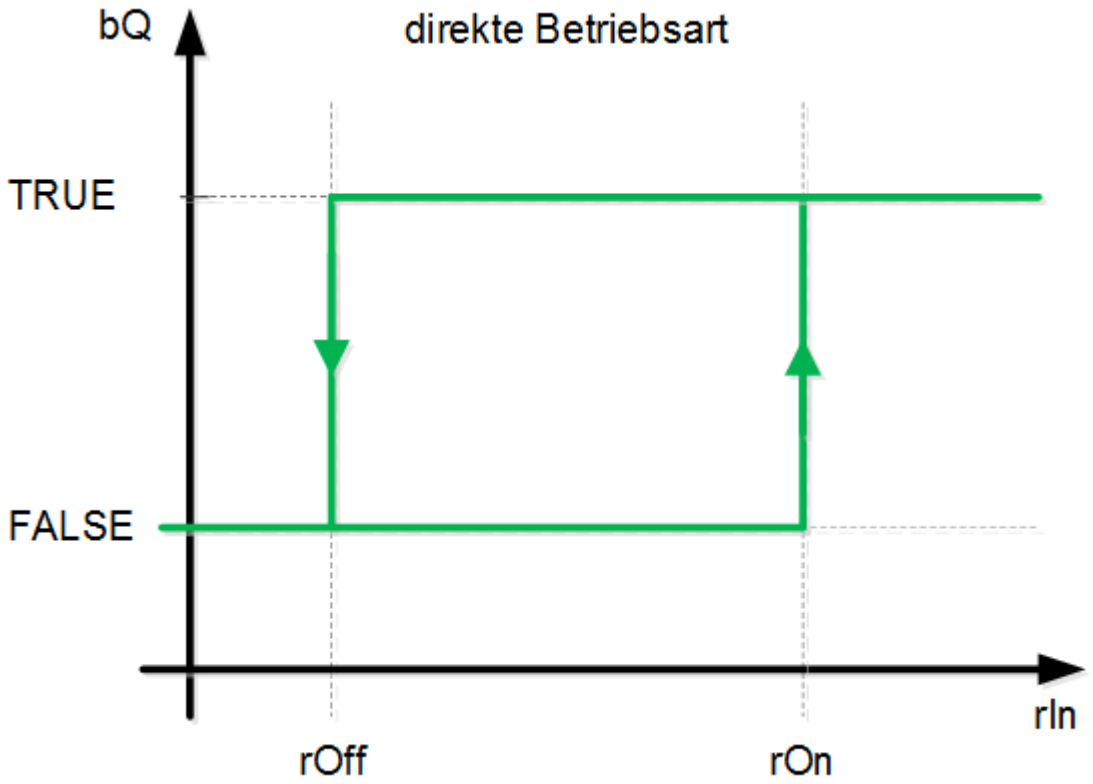
6.1.2.3.2.1.5.4.2 FB_BA_Swi2P



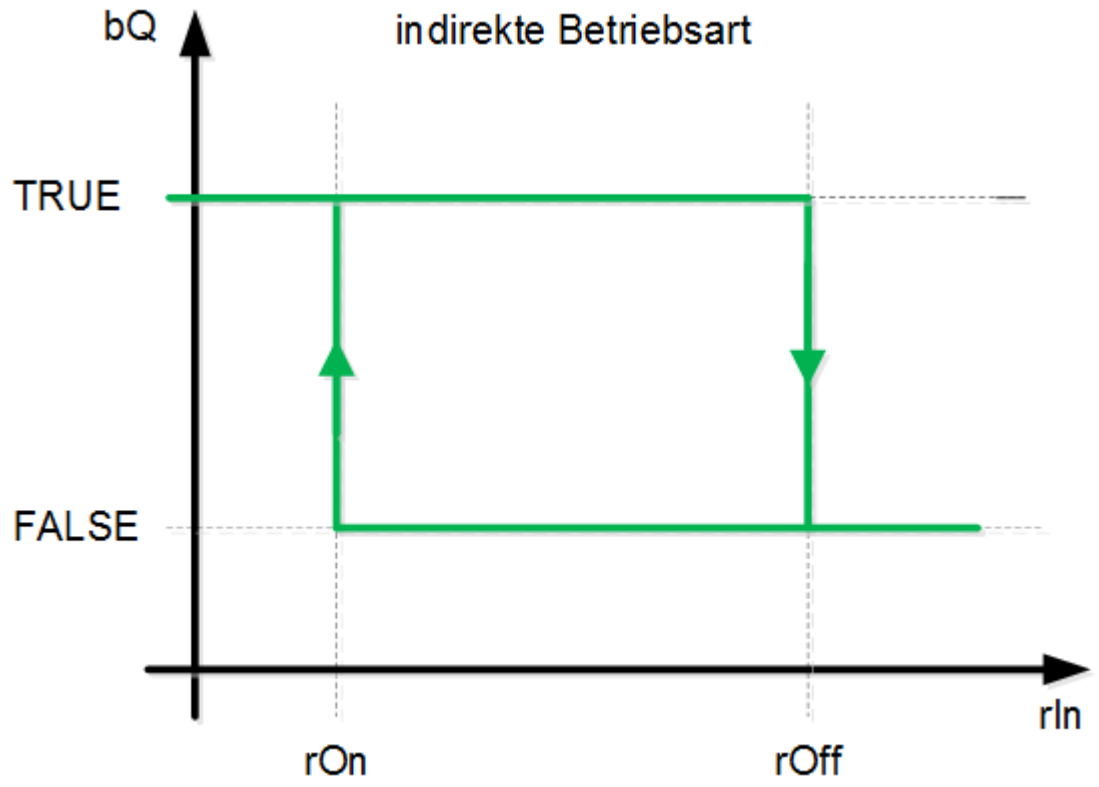
Der Funktionsbaustein *FB_BA_Swi2P* ist ein Zweipunktschalter mit einem Ein- und einem Ausschaltpunkt.

Eine generelle Freigabe des Funktionsbausteins erfolgt am Eingang *bEn*. Solange *bEn* FALSE ist, ist der Ausgang *bQ* FALSE. Der Wirksinn des Bausteins hängt von der relativen Lage des Einschaltpunktes und des Ausschaltpunktes ab.

Ist der Einschaltpunkt größer als der Ausschaltpunkt dann ist der Wirksinn direkt bzw. gleichläufig (Kühlbetrieb).



Ist der Ausschaltpunkt größer als der Einschaltpunkt dann ist der Wirksinn indirekt bzw. gegenläufig (Heizbetrieb).



VAR_INPUT

```
bEn           : BOOL;
rIn           : REAL;
rOn           : REAL;
rOff          : REAL;
udiDlyOn_sec  : UDINT;
udiDlyOff_sec : UDINT;
```

bEn: generelle Freigabe des Funktionsbausteins.

rIn: Eingangswert.

rOn: Einschaltpunkt.

rOff: Ausschaltpunkt.

udiDlyOn_sec: Einschaltverzögerung.

udiDlyOff_sec: Ausschaltverzögerung.

VAR_OUTPUT

```
bQ                : BOOL;
udiRemTiDlyOn_sec  : UDINT;
udiRemTiDlyOff_sec : UDINT;
```

bQ: Steuerausgang.

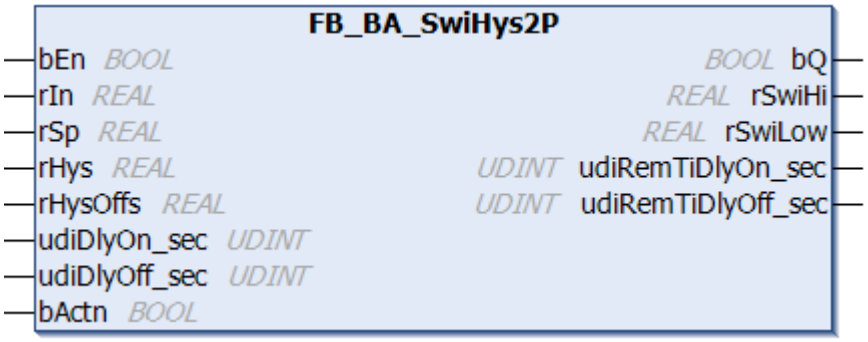
udiRemTiDlyOn_sec: Verbleibende Zeit der Einschaltverzögerung.

udiRemTiDlyOff_sec: Verbleibende Zeit der Ausschaltverzögerung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.4.3 FB_BA_SwiHys2P



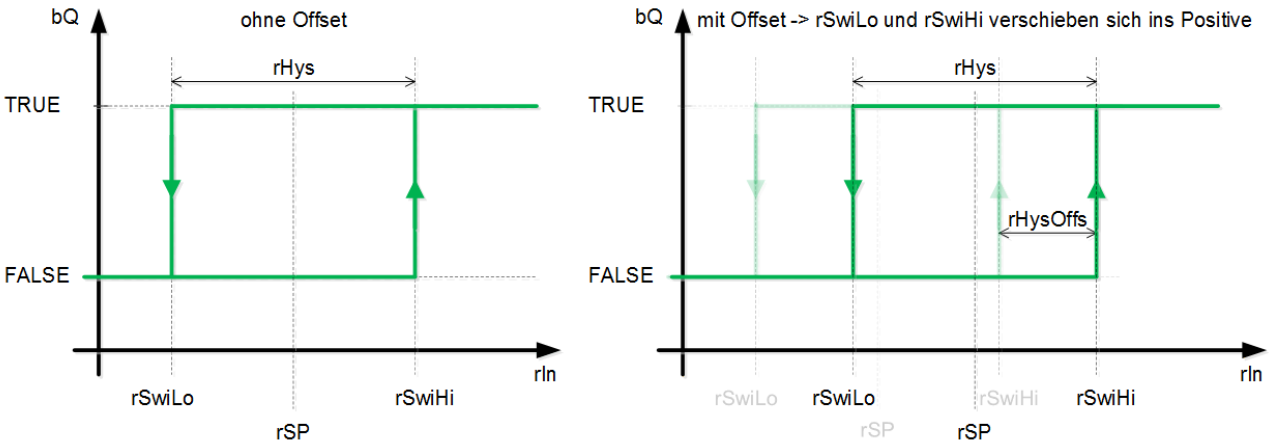
Der Funktionsbaustein *FB_BA_SwiHys2P* ist ein Zweipunktschalter mit einstellbarer Hysterese und Hysterese Offset.

Eine generelle Freigabe des Funktionsbausteins erfolgt am Eingang *bEn*. Bei gesperrten Funktionsbaustein ist der Ausgang *bQ* FALSE. Am Eingang *rSp* wird der Sollwert für den Zweipunktschalter angeschlossen. Der Wirksinn des Bausteins hängt von der Eingangsvariablen *bActn* ab.

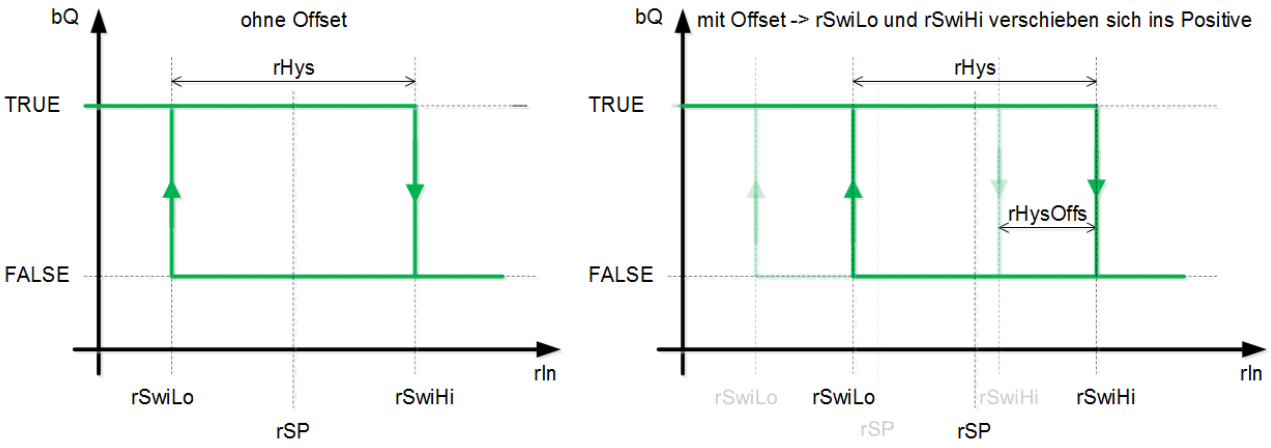
Die aktiven Schaltpunkte ergeben sich aus dem Sollwert, der Hysterese und dem Offset für die Hysterese. Diese werden an den Ausgängen *rSwiHi* und *rSwiLo* ausgegeben.

- Der obere Schaltpunkt ergibt sich aus *rSp + rHys/2 + rHysOffs*.
- Der untere Schaltpunkt ergibt sich aus *rSp - rHys/2 + rHysOffs*.

Ist *bActn* TRUE ergibt sich ein direkter bzw. gleichläufiger Wirksinn (Kühlbetrieb).



Ist *bActn* FALSE ergibt sich ein indirekter bzw. gegenläufiger Wirksinn (Heizbetrieb).



VAR_INPUT

```
bEn           : BOOL;
rIn           : REAL;
rSp           : REAL;
rHys          : REAL;
rHysOffs      : REAL;
udiDlyOn_sec  : UDINT;
udiDlyOff_sec : UDINT;
bActn         : BOOL;
```

bEn: generelle Freigabe des Funktionsbausteins.

rIn: Eingangswert.

rSp: Sollwerteingang.

rHys: Hysterese.

rHysOffs: Offset für die Hysterese.

udiDlyOn_sec: Einschaltverzögerung

udiDlyOff_sec: Ausschaltverzögerung

bActn: Wirksinn.

VAR_OUTPUT

```
bQ                : BOOL;
rSwiHi            : REAL;
rSwiLo            : REAL;
udiRemTiDlyOn_sec  : UDINT;
udiRemTiDlyOff_sec : UDINT;
```

bQ: Ausgang.

rSwiHi: oberer Schaltpunkt.

rSwiLo: unterer Schaltpunkt.

udiRemTiDlyOn_sec: Verbleibende Zeit bis zum Einschalten.

udiRemTiDlyOff_sec: Verbleibende Zeit bis zum Ausschalten.

Voraussetzungen

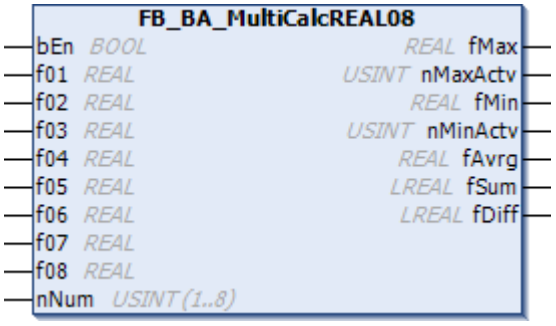
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.5 Mathematische Funktionen

Funktionsbausteine

Name	Beschreibung
FB_BA_MultiCalc_XX [▶ 682]	Multi-Kalkulations-Bausteine
FB_BA_Chrct02 [▶ 683]	Linearinterpolation für 2 Stützstellen
FB_BA_Chrct04 [▶ 684]	Linearinterpolation für 4 Stützstellen
FB_BA_Chrct07 [▶ 686]	Linearinterpolation für 7 Stützstellen
FB_BA_Chrct32 [▶ 688]	Linearinterpolation für 32 Stützstellen
FB_BA_TiAvrg [▶ 689]	Zeitlich arithmetischer Mittelwert

6.1.2.3.2.1.5.5.1 FB_BA_MultiCalc_XX



Die Multi-Kalkulations-Bausteine existieren für die Variablentypen LREAL und REAL, haben jedoch alle die gleiche Funktionalität.
Exemplarisch wird hier der Baustein FB_BA_R08 beschrieben.

Der Funktionsbaustein ermittelt im aktivierten Zustand (*bEn*=TRUE) aus den 8 Eingangswerten *r01...r08* folgendes:

- den Maximalwert von allen Eingängen *rMax*
- den Eingang, an dem dieser Maximalwert anliegt *udiMinActv*
- den Minimalwert von allen Eingängen *rMin*
- den Eingang, an dem dieser Minimalwert anliegt *udiMinActv*
- den Mittelwert aller Eingänge *rAvrg*
- die Summe aller Eingänge *rSum*
- die Differenz zwischen dem Maximal- und Minimalwert *rDiff*

Sollen nicht alle Eingänge zur Berechnung kommen, so lässt sich die Anzahl durch einen Eintrag an *udiNum* einschränken: mit *udiNum*=6 beispielsweise werden die Berechnungen nur für die Eingänge *r01...r06* durchgeführt.
Ein Eintrag größer als 8 wird automatisch auf 8 limitiert, ein Eintrag kleiner als 1 automatisch auf 1.

Beispiel:

Eingänge	Ausgang
bEn = TRUE	rMax = 32
r01 = 32	udiMaxActv = 1
r02 = 17	rMin = 5
r03 = 5	udiMinActv = 3
r04 = 9	rAvrg = 18.5
r05 = 16	rSum = 111
r06 = 32	rDiff = 27
r07 = 25	
r08 = 44	
udiNum = 6	

Bei *bEn*=FALSE wird an allen Ausgängen 0 ausgegeben.

VAR_INPUT

```
bEn      : BOOL;
r01      : REAL;
r02      : REAL;
r03      : REAL;
r04      : REAL;
r05      : REAL;
r06      : REAL;
r07      : REAL;
r08      : REAL;
udiNum   : UDINT;
```

bEn: Aktivierung der Bausteinfunktion.

r01...r08: Eingangswerte, aus denen berechnet werden soll.

udiNum: Anzahl der Eingangswerte, mit denen gerechnet wird.

VAR_OUTPUT

```
rMax         : REAL;
udiMaxActv   : UDINT;
rMin         : REAL;
udiMinActv   : UDINT;
rAvrg        : REAL;
rSum         : REAL;
rDiff        : REAL;
```

rMax: Maximalwert von allen Eingängen.

udiMaxActv: Eingang, an dem der Maximalwert anliegt.

rMin: Minimalwert von allen Eingängen.

udiMinActv: Eingang, an dem der Minimalwert.

rAvrg: Mittelwert aller Eingänge.

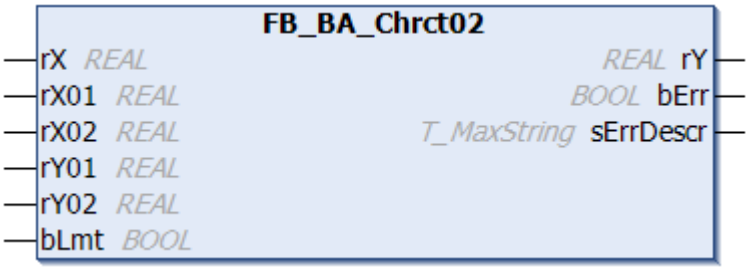
rSum: Summe aller Eingänge.

rDiff: Differenz zwischen dem Maximal- und Minimalwert.

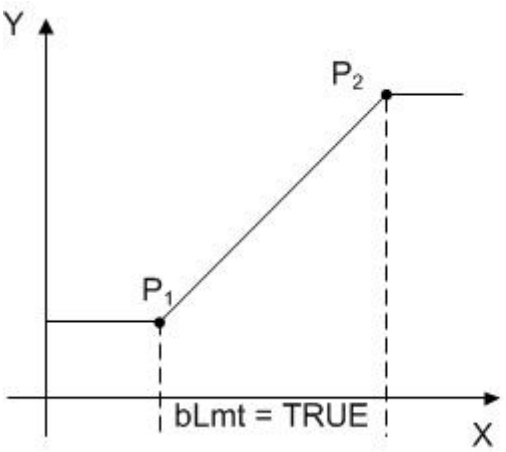
Voraussetzungen

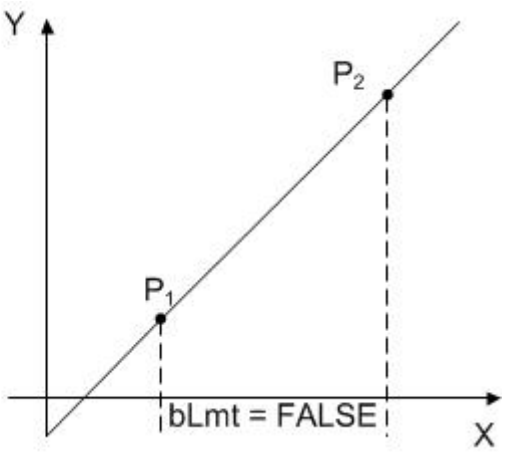
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.5.2 FB_BA_Chrct02



Der Baustein FB_BA_Chrct02 stellt eine Linear-Interpolation mit 2 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*rX1/rY1*] und [*rX2/rY2*] bestimmt. Wenn die Eingangsvariable *bLmt* TRUE ist, wird *rY* von *rY01* und *rY02* begrenzt. Bei *bLmt* gleich FALSE erfolgt keine Begrenzung von *rY*.





Fehlerbehandlung

Die Eingabewerte für *rX[n+1]* müssen immer mindestens um den Betrag von 0.0000001 größer als von *rX[n]* sein.
Im Fehlerfall zeigt die Variable *sErrDescr* an, dass an einem Punkt der Kennlinie die Werte nicht monoton steigend sind.

VAR_INPUT

```
rX    : REAL;
rX01  : REAL;
rX02  : REAL;
rY01  : REAL;
rY02  : REAL;
bLmt  : BOOL;
```

rX: Eingangswert der Kennlinie.

rX01: X-Wert für Stützpunkt P1.

rX02: X-Wert für Stützpunkt P2.

rY01: Y-Wert für Stützpunkt P1.

rY02: Y-Wert für Stützpunkt P2.

bLmt: Begrenzung des Ausgangswertes *rY*.

VAR_OUTPUT

```
rY         : REAL;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
```

rY: Errechneter Ausgangswert der Kennlinie.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

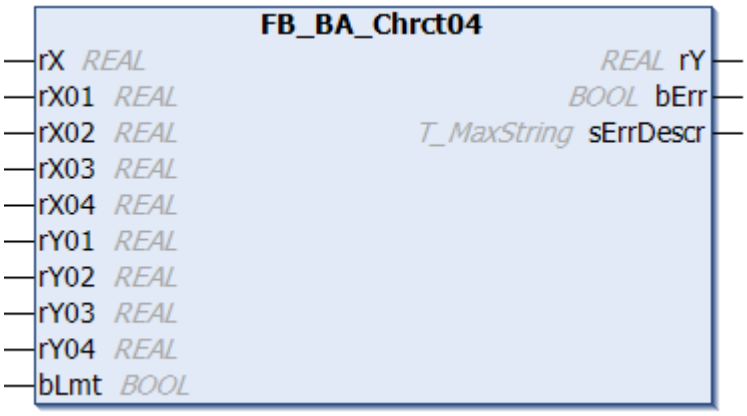
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: rX01 darf nicht gleich rX02 sein.

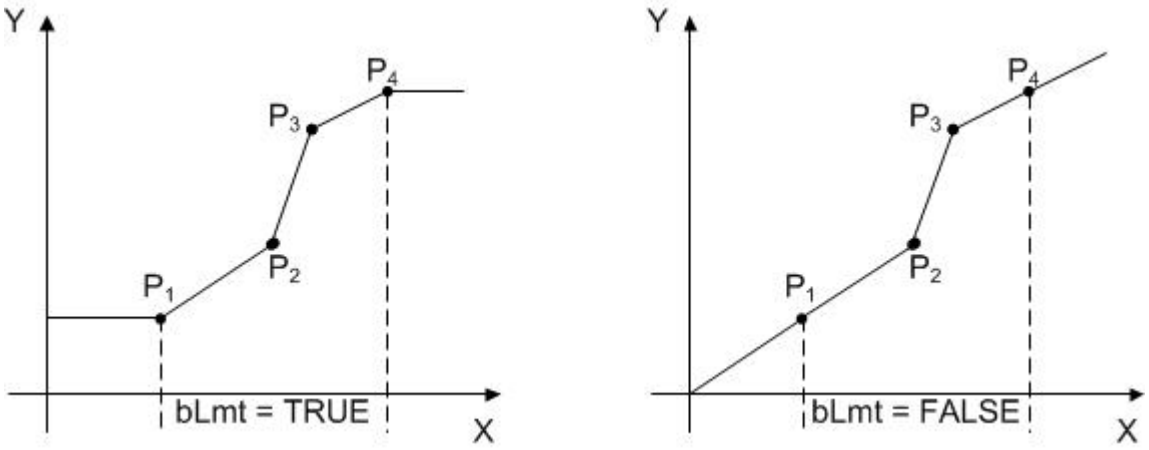
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.5.3 FB_BA_Chrct04



Der Baustein FB_BA_Chrct04 stellt eine Linear-Interpolation mit 4 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*rX1/rY1*] bis [*rX4/rY4*] bestimmt. Wenn die Eingangsvariable *bLmt* TRUE ist, wird *rY* von *rY01* und *rY04* begrenzt. Bei *bLmt* gleich FALSE erfolgt keine Begrenzung von *rY*.



Fehlerbehandlung

Die Eingabewerte für *rX[n+1]* müssen immer mindestens um den Betrag von 0.0000001 größer als von *rX[n]* sein.
Im Fehlerfall zeigt die Variable *sErrDescr* an, dass an einem Punkt der Kennlinie die Werte nicht monoton steigend sind.

VAR_INPUT

```
rX     : REAL;
rX01   : REAL;
rX02   : REAL;
rX03   : REAL;
rX04   : REAL;
rY01   : REAL;
rY02   : REAL;
rY03   : REAL;
rY04   : REAL;
bLmt   : BOOL;
```

rX: Eingangswert der Kennlinie.

rX01: X-Wert für Stützpunkt P1.

rX02: X-Wert für Stützpunkt P2.

rX03: X-Wert für Stützpunkt P3.

rX04: X-Wert für Stützpunkt P4.

rY01: Y-Wert für Stützpunkt P1.

rY02: Y-Wert für Stützpunkt P2.

rY03: Y-Wert für Stützpunkt P3.

rY04: Y-Wert für Stützpunkt P4.

bLmt: Begrenzung des Ausgangswertes *rY*.

VAR_OUTPUT

```
rY         : REAL;
bErr       : BOOL;
sErrDescr  : T_MAXSTRING;
```

rY: Errechneter Ausgangswert der Kennlinie.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

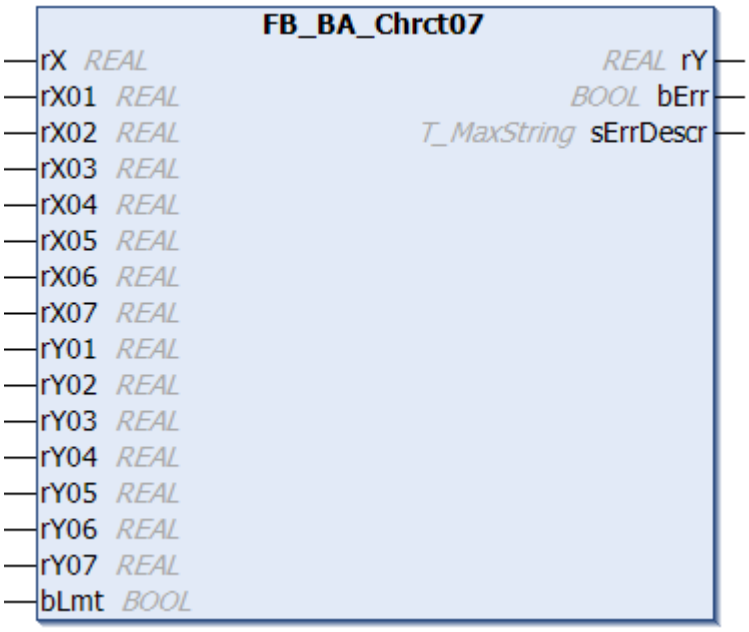
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: am angegebenen Element. Die Reihenfolge muss immer rX01 > rX02 > rXn oder rX01 < rX02 < rXn sein.

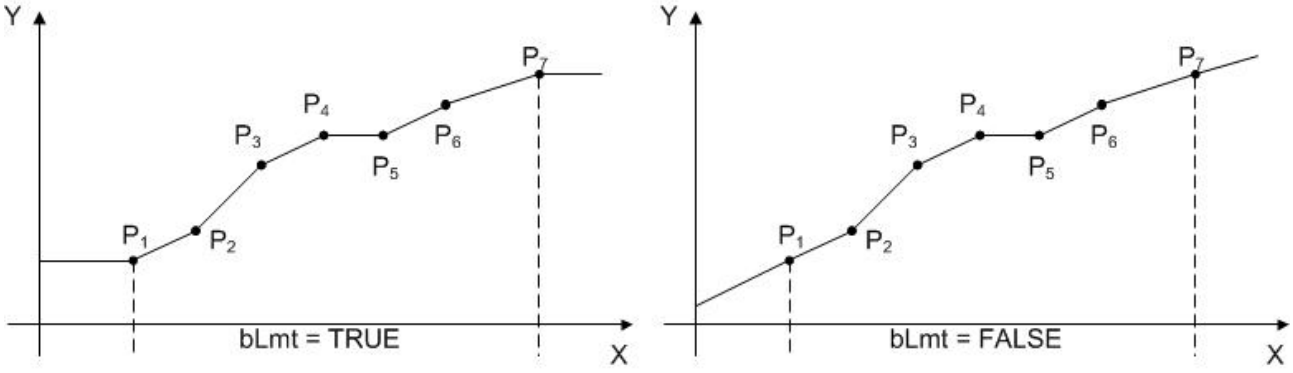
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.5.4 FB_BA_Chrct07



Der Baustein FB_BA_Chrct07 stellt eine Linear-Interpolation mit 7 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*rX1/rY1*] bis [*rX7/rY7*] bestimmt. Wenn die Eingangsvariable *bLmt* TRUE ist, wird *rY* von *rY01* und *rY07* begrenzt. Bei *bLmt* gleich FALSE erfolgt keine Begrenzung von *rY*.



Fehlerbehandlung

Die Eingabewerte für *rX[n+1]* müssen immer mindestens um den Betrag von 0.0000001 größer als von *rX[n]* sein.
Im Fehlerfall zeigt die Variable *sErrDescr* an, dass an einem Punkt der Kennlinie die Werte nicht monoton steigend sind.

VAR_INPUT

```
rX   : REAL;
rX01 : REAL;
rX02 : REAL;
rX03 : REAL;
rX04 : REAL;
rX05 : REAL;
rX06 : REAL;
rX07 : REAL;
rY01 : REAL;
rY02 : REAL;
rY03 : REAL;
rY04 : REAL;
rY05 : REAL;
rY06 : REAL;
rY07 : REAL;
bLmt : BOOL;
```

rX: Eingangswert der Kennlinie.

rX01: X-Wert für Stützpunkt P1.

rX02: X-Wert für Stützpunkt P2.

rX03: X-Wert für Stützpunkt P3.

rX04: X-Wert für Stützpunkt P4.

rX05: X-Wert für Stützpunkt P5.

rX06: X-Wert für Stützpunkt P6.

rX07: X-Wert für Stützpunkt P7.

rY01: Y-Wert für Stützpunkt P1.

rY02: Y-Wert für Stützpunkt P2.

rY03: Y-Wert für Stützpunkt P3.

rY04: Y-Wert für Stützpunkt P4.

rY05: Y-Wert für Stützpunkt P5.

rY06: Y-Wert für Stützpunkt P6.

rY07: Y-Wert für Stützpunkt P7.

bLmt: Begrenzung des Ausgangswertes *rY*.

VAR_OUTPUT

```
rY        : REAL;
bErr      : BOOL;
sErrDescr : T_MAXSTRING;
```

rY: Errechneter Ausgangswert der Kennlinie.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

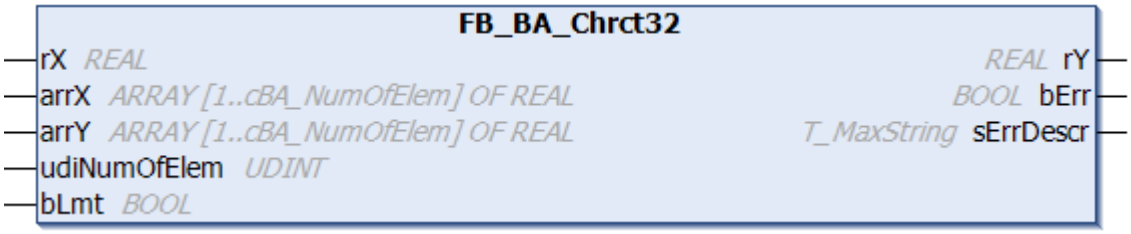
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: am angegebenen Element. Die Reihenfolge muss immer rX01 > rX02 > rXn oder rX01 < rX02 < rXn sein.

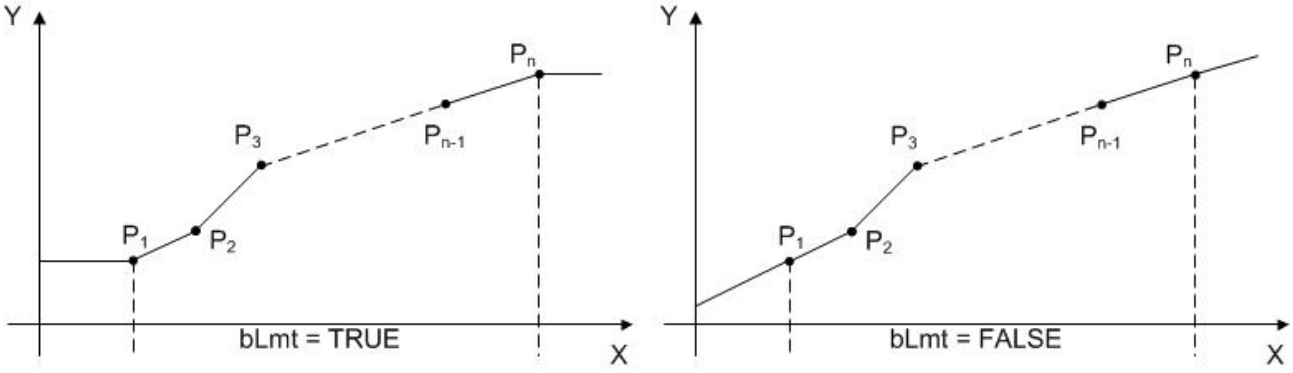
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.5.5 FB_BA_Chrct32



Der Baustein FB_BA_Chrct32 stellt eine Linear-Interpolation mit bis zu 32 Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Stützstellen werden hier, anders als bei den "kleineren" Interpolationsbausteinen FB_BA_Chrct02 [▶ 683], FB_BA_Chrct04 [▶ 684] und FB_BA_Chrct07 [▶ 686] der Übersicht halber über Feldvariablen [*arrX[1]/arrY[1]* bis [*arrX[n]/arrY[n]*] bestimmt. Wenn die Eingangsvariable *bLmt* TRUE ist, wird *rY* von *arrY[1]* und *arrY[n]* begrenzt. Bei *bLmt* gleich FALSE erfolgt keine Begrenzung von *rY*.



Fehlerbehandlung

Die Eingabewerte für *rX[n+1]* müssen immer mindestens um den Betrag von 0.0000001 größer als von *rX[n]* sein.
Im Fehlerfall zeigt die Variable *sErrDescr* an, dass an einem Punkt der Kennlinie die Werte nicht monoton steigend sind.

Der Parameter für die Anzahl der Stützstellen, *diNumOfElem*, muss im Bereich 2..32 liegen.

VAR_INPUT

```
rX          : REAL;
arrX        : ARRAY [1..FB_BA_Chrct32.cBA_NumOfElem] OF REAL;
arrY        : ARRAY [1..FB_BA_Chrct32.cBA_NumOfElem] OF REAL;
diNumOfElem : DINT(2..32);
bLmt        : BOOL;
```

rX: Eingangswert der Kennlinie

arrX: Feld mit den X-Werten für die Stützpunkte.

arrY: Feld mit den Y-Werten für die Stützpunkte.

diNumOfElem: Anzahl der Stützstellen. Intern begrenzt auf Werte von 2 bis 32.

bLmt: Begrenzung des Ausgangswertes *rY.*

VAR_OUTPUT

```
rY        : REAL;
bErr      : BOOL;
sErrDescr : T_MAXSTRING;
```

rY: Errechneter Ausgangswert der Kennlinie.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn die eingetragenen Parameter fehlerhaft sind.

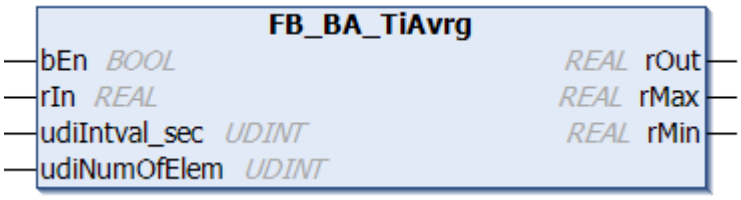
sErrDescr: Enthält die Fehlerbeschreibung.

Fehlerbeschreibung
01: Fehler: am angegebenen Element. Die Reihenfolge muss immer rX01 > rX02 > rXn oder rX01 < rX02 < rXn sein.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.5.6 FB_BA_TiAvrg



Der Funktionsbaustein *FB_BA_TiAvrg* errechnet den zeitlich arithmetischen Mittelwert eines aufgenommenen Analogwertes. Dazu werden zeitlich diskret Werte in einen FIFO-Speicher hineingeschrieben. *udiIntval_sec* gibt das Zeitintervall [s] an, in dem die Werte aufgenommen und in den FIFO geschrieben werden. Das Schreiben der Werte erfolgt, wenn der Eingang *bEn* = TRUE ist. Mit der Variablen *udiNumOfElem* wird die größer des FIFO-Speichers bestimmt. Diese ist auf 1..512 begrenzt.
Der Funktionsbaustein kann zum Beispiel für die Berechnung einer mittleren Außentemperatur im stündlichen Tagesmittel verwendet werden. In dem Fall wäre *udiNumOfElem* = 24 und *udiIntval_sec* = 3600 Sekunden. *bEn* ist die allgemeine Freigabe des Funktionsbausteins. Bei *bEn* = FALSE wird der FIFO-Speicher innerhalb des Funktionsbaustein komplett gelöscht und es werden keine Daten aufgezeichnet.

Beispiel:

udiNumOfElem = 5

	1. Zyklus		2. Zyklus		3. Zyklus		4. Zyklus	
	rIn	rOut	rIn	rOut	rIn	rOut	rIn	rOut
t0	2	2/1 = 2	6	(4+6+7+7+6)/5 = 6	1	(7+6+5+4+1)/5 = 4.6	3	rIn = 3
t1	4	(2+4)/2 = 3	5	(6+7+7+6+5)/5 = 6.25	2	(6+5+4+1+2)/5 = 3.6	1.5	rIn = 1.5
t2	6	(2+4+6)/3 = 4	4	(7+7+6+5+4)/5 = 5.8	4	(5+4+1+2+4)/5 = 3.2		
t3	7	(2+4+6+7)/4 = 4.75	2	(7+7+6+5+4)/5 =5.8	5	(4+1+2+4+5)/5 = 3.2		
t4	7	(2+4+6+7+7)/5 = 5.2	1	(7+7+6+5+4)/5 =5.8	4	(1+2+4+5+4)/5 = 3,2		

VAR_INPUT

```
bEn           : BOOL;
rIn           : REAL;
udiIntval_sec : UDINT;
udiNumOfElem  : UDINT;
```

bEn: Freigabe des Bausteins.

rIn: Eingangswert zur Mittelwertbildung.

udiIntVal_SEC: Zeitintervall [s] für das Schreiben neuer Werte in den FIFO. Intern begrenzt auf einen Wert von 1 bis 2147483.

udiNumOfElem: Größe des FIFO-Wertespeichers. Eine Änderung setzt die vorangegangene Mittelwertbildung zurück. Intern begrenzt auf einen Wert von 1 bis 512.

VAR_OUTPUT

```
rOut        : REAL;
rMax        : REAL;
rMin        : REAL;
```

rOut: Berechneter Mittelwert.

rMax: größter Wert im FIFO-Speicher.

rMin: kleinster Wert im FIFO-Speicher.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.6 Monitoring Funktionen

Funktionsbausteine

Name	Beschreibung
FB_BA_FdbCtrlBinary [▶ 691]	Rückmeldungsüberwachung eines Aktors mittels digitaler Rückmeldung.
FB_BA_FixedLimitCtrl [▶ 690]	Grenzwertüberwachung eines festen Wertes.
FB_BA_SlidingLimitCtrl [▶ 692]	Gleitende Grenzwertüberwachung.

6.1.2.3.2.1.5.6.1 FB_BA_FixedLimitCtrl



Funktionsbaustein zur Überwachung eines festen Grenzwertes.

Der Eingang *bEn* dient der allgemeinen Freigabe des Bausteins.

Um den Wert zu überwachenden Wert *rIn* herum wird ein Toleranzbereich definiert.

Der Toleranzbereich ergibt sich aus einem oberen Grenzwert *rInHighLimit* und einem unteren Grenzwert *rInLowLimit.*

Übersteigt der Wert *rIn* den oberen Grenzwert des Toleranzbereiches, dann wird zur Warnung der Ausgang *bHighLimit* TRUE. Eine Ansprechverzögerung des Ausgangs *bHighLimit* ist mit dem Timer *udiDelay_sec* zu parametrieren.

Unterschreitet der Wert *rIn* den unteren Grenzwert des Toleranzbereiches, dann wird zur Warnung der Ausgang *bLowLimit* TRUE. Eine Ansprechverzögerung des Ausgangs *bLowLimit* ist mit dem Timer *udiDelay_sec* zu parametrieren.

VAR_INPUT

```
bEn           : BOOL;
rHighLimit    : REAL := 32;
rLowLimit     : REAL := 16;
rIn           : REAL;
udiDelay_sec  : UDINT;
```

bEn: Freigabe Baustein.

rHighLimit: Vorgabe oberer Grenzwert, voreingestellt auf 32.

rLowLimit: Vorgabe unterer Grenzwert, voreingestellt auf 16.

rIn: Zu überwachender Eingangswert.

udiDelay_sec: Ansprechverzögerung [s] des Ausgangs. Intern begrenzt auf Werte von 0 bis Const.udiTiSec [▶ 705].

VAR_OUTPUT

```
bHighLimit          : BOOL;
bLowLimit           : BOOL;
udiRemTiDelay_sec   : UDINT;
```

bHighLimit: Oberer Grenzwert erreicht.

bLowLimit: Unterer Grenzwert erreicht.

udiRemTiDelay_sec: Verbleibende Zeit nach Überschreitung eines Grenzwertes bis entweder der Ausgang *bHighLimit* oder *bLowLimit* ansprechen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.6.2 FB_BA_FdbCtrlBinary



Der Funktionsbaustein dient der Feedback Überwachung eines Aktors mittels digitaler Rückmeldung. Anwendungsbeispiele des Funktionsbausteins sind zum Beispiel eine Betriebsrückmeldeüberwachung, eine Prozessrückmeldeüberwachung oder die Laufüberwachung eines Antriebs mittels Endlagenschalter.

Der Eingang *bEn* dient der allgemeinen Freigabe des Bausteins. Wenn *bEn* FALSE ist wird der Meldeausgang *bQ* immer FALSE sein.

An dem Eingang *bActuator* wird der Schaltaktorausgang des zu überwachenden Aggregates angeschlossen. Der Eingang *bSwitch* dient dem Anschluss des Feedback-Signals (z.B. Differenzdruckschalter, Strömungswächter oder Endlagenschalter).

Mittels des Timers *udiFdbDelay_sec* [s] wird eine Ansprechverzögerung der Feedback-Kontrolle nach dem Start des Aggregates eingestellt.

Der zweite Timer *udiInterruptionDelay_sec* [s] dient einer Ansprechverzögerung der Feedback-Kontrolle nach dem Erreichen des Endzustands.

VAR_INPUT

```
bEn                     : BOOL;
bActuator               : BOOL;
bSwitch                 : BOOL;
udiFdbDelay_sec         : UDINT;
udiInterruptionDelay_sec : UDINT;
```

bEn: Freigabe Baustein.

bActuator: Rückmeldung des Schaltausgangs.

bSwitch: Feedback-Signal vom Prozess.

udiFdbDelay_sec: Ansprechverzögerung [s] der Überwachungsfunktion beim Start des Aktors. Intern begrenzt auf Werte von 0 bis Const.udiTiSec [▶ 705].

udiInterruptionDelay_sec: Ansprechverzögerung [s] der Überwachungsfunktion bei bereits erfolgreich gestarteten Aktor (z.B. Druckschwankungen bei einer Laufüberwachung eines Ventilators). Intern begrenzt auf Werte von 0 bis Const.udiTiSec [▶ 705].

VAR_OUTPUT

```
bQ                       : BOOL;
udiRemTiFdbDelay         : UDINT;
udiRemTiInterruptionDelay : UDINT;
```

bQ: Ausgabe einer Fehlermeldung falls das Feedback-Signal nicht in der parametrierten Zeit von *udiFdbDelay_sec* ansteht, oder das Feedbacksignal länger als nach *udiInterruptionDelay_sec* unterbrochen wurde.

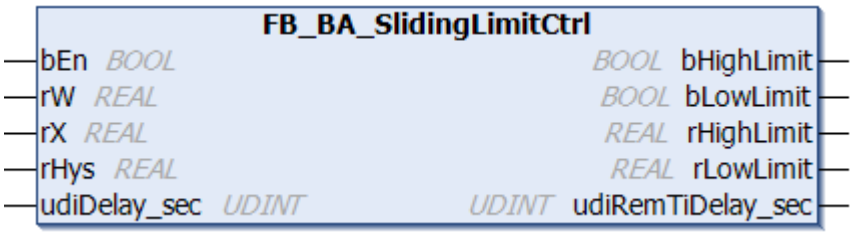
udiRemTiFdbDelay: Verbleibende Zeit [s] bis der Ausgang *bErrOpn* gesetzt wird.

udiRemTiInterruptionDelay: Verbleibende Zeit [s] bis der Ausgang *bErrSwi* gesetzt wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.6.3 FB_BA_SlidingLimitCtrl

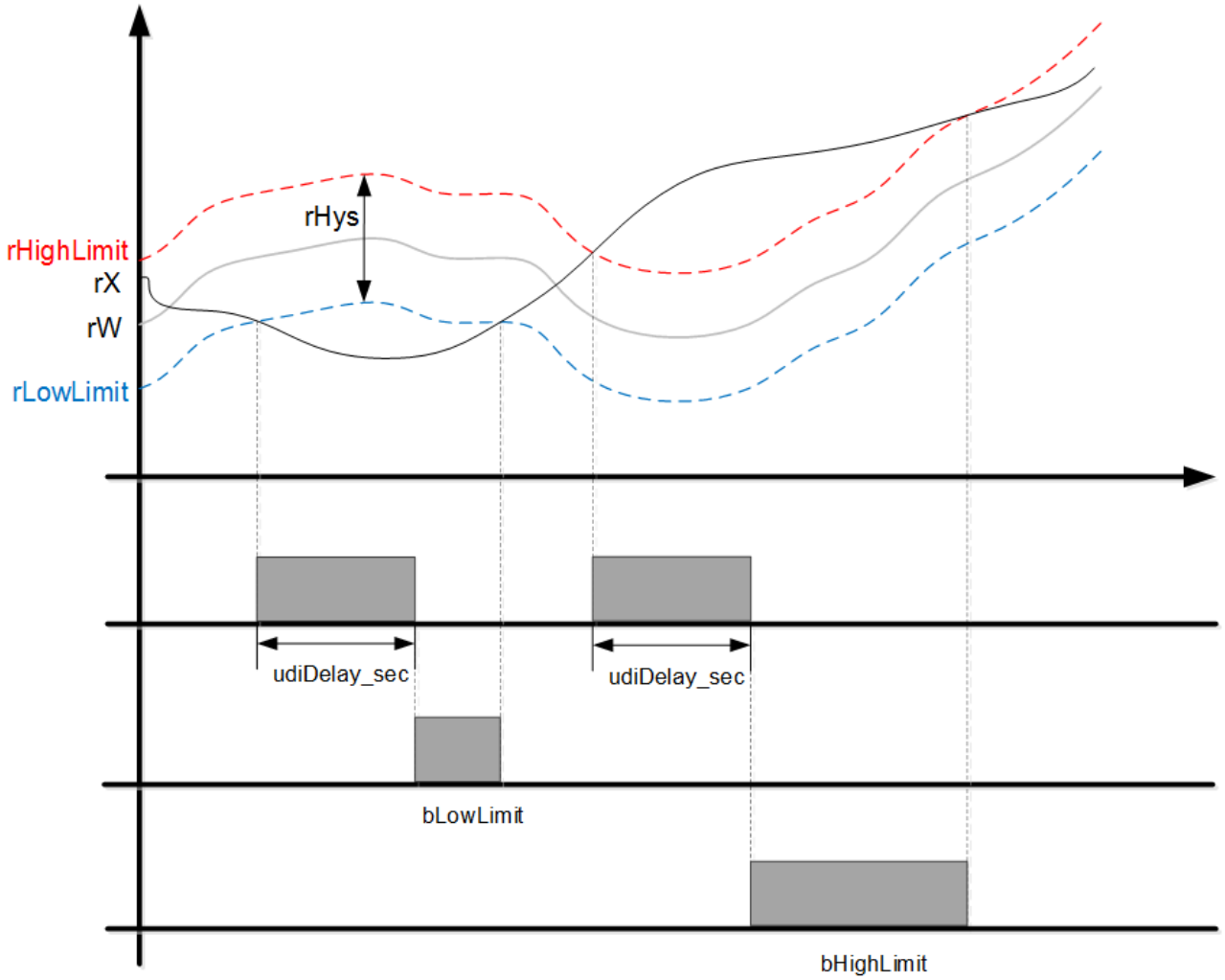


Funktionsbaustein zur Überwachung eines gleitenden Sollwertes.

Der Eingang *bEn* dient der allgemeinen Freigabe des Bausteins.

Um die Funktion einer Regelung zu überprüfen, wird der Istwert mit dem Sollwert der Regelstrecke verglichen.

Ist die Abweichung von Soll- und Istwert innerhalb des Toleranzbereiches *rHys*, dann ist die Regelung in Ordnung. Weicht der Istwert über einen längeren Zeitraum um einen Betrag außerhalb dieses Toleranzbereiches von dem Sollwert ab, dann wird der Timer **udiDelay_sec** gestartet. Nach dem Ablauf des Timers, bei bleibender Regelabweichung, wird entweder der Ausgang *bLowLimit* oder *bHighLimit* TRUE. des Funktionsbausteins eine Meldung ausgegeben.



VAR_INPUT

```
bEn          : BOOL;
rW           : REAL;
rX           : REAL;
rHys         : REAL;
udiDelay_sec : UDINT;
```

bEn: Freigabe Baustein.

rW: Sollwert.

rX: Istwert.

rHys: Hysterese.

udiDelay_sec: Ansprechverzögerung [s] des Ausgangs. Intern begrenzt auf Werte von 0 bis Const.udiTiSec [▶ 705].

VAR_OUTPUT

```
bHighLimit         : BOOL;
bLowLimit          : BOOL;
rHighLimit         : REAL;
rLowLimit          : REAL;
udiRemTiDelay_sec  : UDINT;
```

bHighLimit: Oberer Grenzwert erreicht.

bLowLimit: Unterer Grenzwert erreicht.

rHighLimit: Ausgabe des oberen Grenzwerts.

rLowLimit: Ausgabe des unteren Grenzwerts.

udiRemTiDelay_sec: Verbleibende Zeit nach Überschreitung eines Grenzwertes bis entweder der Ausgang *bHighLimit* oder *bLowLimit* ansprechen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.7 Rampen, Filter,Regler

Funktionsbausteine

Name	Beschreibung
FB_BA_FltrPT1 [▶ 694]	Filter erster Ordnung
FB_BA_RampLmt [▶ 695]	Rampenbegrenzung
FB_BA_SeqCtrl	Sequenzregler (Funktionsbaustein in Tc3_BA_Common)
FB_BA_SeqLink	Sequenzlinker (Funktionsbaustein in Tc3_BA_Common)
FB_BA_PIDCtrl	PID-Regler (Funktionsbaustein in Tc3_BA_Common)

6.1.2.3.2.1.5.7.1 FB_BA_FltrPT1



Filter erster Ordnung.



Beim ersten Aufruf des Bausteines (Systemstart) wird der Ausgang *rOut* einmalig automatisch gleich dem Eingang *rIn* gesetzt.

VAR_INPUT

```
rIn              : REAL;
udiDampConst_sec : UDINT;
bSetActl         : BOOL;
```

rIn: Eingangssignal

udiDampConst_sec: Filterzeitkonstante [s]. Intern begrenzt auf Werte von 0 bis 86400.

bSetActl: Eine steigende Flanke an diesem Eingang setzt den Ausgangswert *rOut* auf den Eingangswert *rIn*.

VAR_OUTPUT

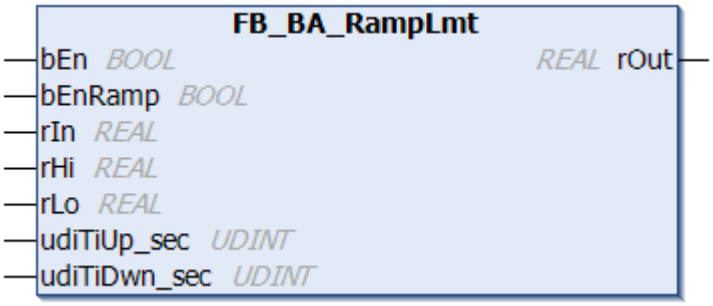
```
rOut    : REAL;
```

rOut: gefiltertes Ausgangssignal.

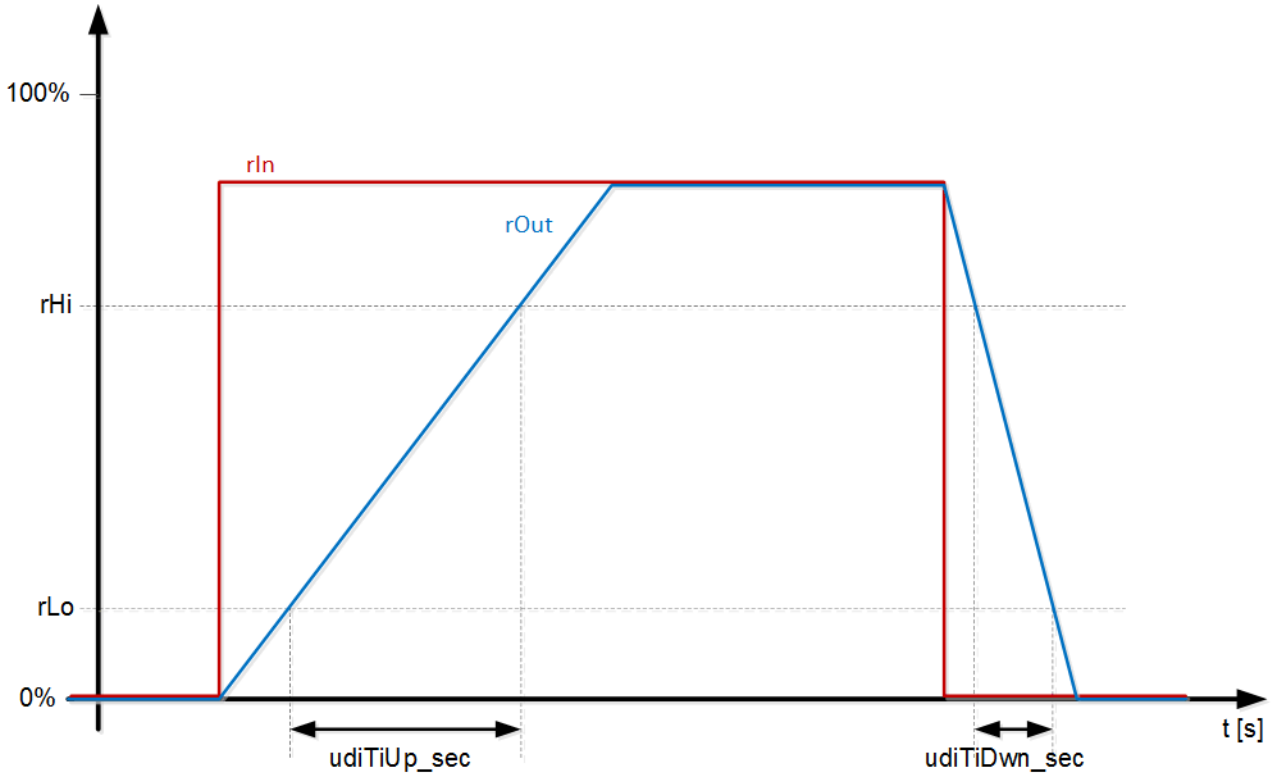
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.7.2 FB_BA_RampLmt



Der Funktionsbaustein limitiert die Anstiegs- bzw. Abfallgeschwindigkeit eines Eingangssignals.
Beim Anstieg von *rIn* wird der Ausgang *rOut* auf die Steigung von (*rHi-rLo*)/*udiTiUp_sec* begrenzt.
Beim Abfallen von *rIn* wird der Ausgang *rOut* auf die Steigung von (*rHi-rLo*)/*udiTiDwn_sec* begrenzt.



VAR_INPUT

```
bEn             : BOOL;
bEnRamp         : BOOL;
rIn             : REAL;
rHi             : REAL;
rLo             : REAL;
udiTiUp_sec     : UDINT;
udiTiDwn_sec    : UDINT;
```

bEn: Freigabe Baustein, wenn FALSE, dann ist *rOut* = 0.0.

bEnRamp: Freigabe Rampenbegrenzung, wenn FALSE, dann ist *rOut* = *rIn*.

rIn: Eingangswert der Rampenfunktion

rHi: Obere Stützstelle zur Berechnung der Rampen.

rLo: Untere Stützstelle zur Berechnung der Rampen. *rHi* muss größer als *rLo* sein, ansonsten wird ein Fehler ausgegeben!

udiTiUp_sec: Anstiegszeit [s].

udiTiDwn_sec: Abfallzeit [s]

VAR_OUTPUT

```
rOut     : REAL;
```

rOut: durch die Rampen steigungs-begrenztes Ausgangssignal

Voraussetzungen

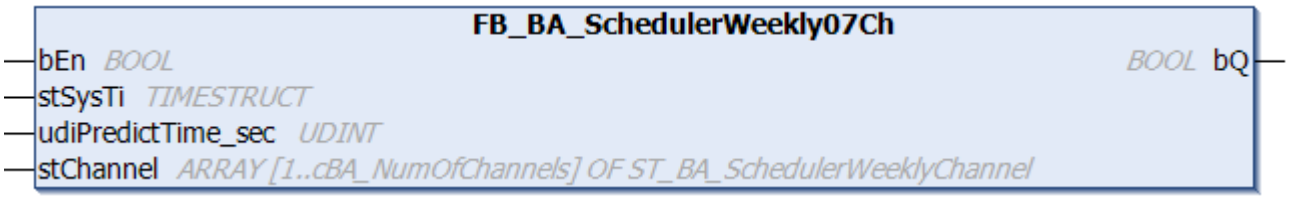
Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.8 Kalender

Funktionsbausteine

Name	Beschreibung
FB_BA_SchedulerWeeklyXXCh [▶ 696]	Wochenzeitschaltplan
FB_BA_CalenderXXCh [▶ 697]	Jahreszeitschaltplan

6.1.2.3.2.1.5.8.1 FB_BA_SchedulerWeeklyXXCh



Wochenzeitschaltuhr mit 1, 7 oder 28 Zeitschaltkanälen.
Exemplarisch wird hier der Baustein *FB_BA_SchedulerWeekly07Ch* beschrieben.

Der Funktionsbaustein dient der Eingabe von insgesamt bis zu 7 Einschaltzeiträumen.
Jedem Einschaltzeitraum kann eine Einschaltzeit [hh:mm:ss] und eine Ausschaltzeit [hh:mm:ss] zugewiesen werden.
Mit den Variablen *bMonday* bis *bSunday* kann ausgewählt werden, an welchen Wochentagen der Einschaltzeitraum aktiv sein soll.
Ein Einschaltzeitraum ist nur aktiv wenn die Variable *bEn* des Kanals auf TRUE gesetzt ist.
Für unregelmäßige aber wiederkehrende Ereignisse, kann die Variable *bResetAfterOn* auf TRUE gesetzt werden. Damit wird die Freigabe des Kanals *bEn* nach dem Ablauf des Ereignisses automatisch auf FALSE zurückgesetzt werden.
Zur Erleichterung der Dateneingabe setzt eine steigende Flanke an *bAllActive, bEn* und alle Wochentage (bMonday bis bSunday) auf TRUE.

Der Baustein arbeitet nur dann, wenn ein TRUE-Signal an *bEn* anliegt.

Für eine bedarfsabhängige Einschaltoptimierung, kann das Einschalten des Ausgangs *bQ* um die Zeit von der Variablen *udiPredictTime_sec* vorverlegt werden.



Die Ein- und Ausschaltpunkte eines Kanals müssen in ein und demselben Jahr sein. Der Ausschaltpunkt darf nicht vor dem Einschaltpunkt liegen. Sonst wird der Ausschaltpunkt automatisch korrigiert und gleich gesetzt mit dem Einschaltpunkt.
Ist der Einschaltpunkt gleich dem Ausschaltpunkt, so bleibt der Kanal aus.

VAR_INPUT

```
bEn                 : BOOL;
stSysTi             : TIMESTRUCT;
udiPredictTime_sec  : UDINT;
```

bEn: Generelle Baustein Freigabe.

stSysTi: Struktur mit der lokalen NT-Systemzeit (siehe TIMESTRUCT).

udiPredictTime_sec: Vorausberechnete Einschaltzeit. Intern begrenzt auf Werte zwischen 0 und 43200.

VAR_OUTPUT

```
bQ   : BOOL;
```

bQ: Schaltausgang

VAR_IN_OUT

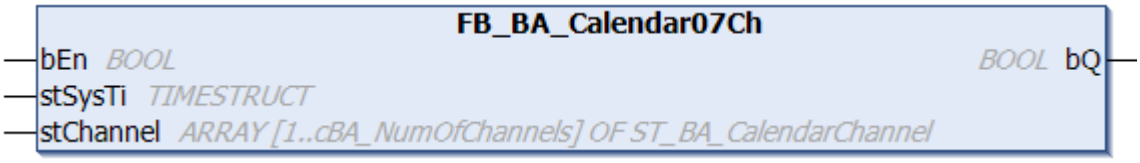
```
arrChannel: ARRAY [1..cBA_NumOfChannels] OF ST_BA_SchedulerWeeklyChannel;
```

arrChannel: Wochenzeitschaltplan, bei dem einkanaligen Baustein heißt die Variable stChannel (siehe ST_BA_SchedulerWeeklyChannel [▶ 704]). Intern begrenzt auf die jeweilige Anzahl der möglichen Kanäle über die Variable *cBA_NumOfChannels*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.1.5.8.2 FB_BA_CalenderXXCh



Jahreszeitschaltplan mit 1, 7 oder 28 Kanälen.

Exemplarisch wird hier der Baustein *FB_BA_Calender07Ch* beschrieben.

Dieser Funktionsbaustein dient der Eingabe von Zeiträumen z.B. Schulferien oder Betriebsferien.
Mit der Eingangsvariable *bEnable* wird der Baustein aktiviert.
Der Eingang *stSsyTi* wird mit der aktuellen Systemzeit verknüpft.
Wenn die Zeitschaltbedingung erfüllt ist, ist der Ausgang *bQ* gesetzt.
Ein Zeitraum wird innerhalb des Kalenders durch ein Einschaltdatum [Tag, Monat, Stunde, Minute] und ein Ausschaltdatum [Tag, Monat, Stunde, Minute] beschrieben.

Ein Einschaltzeitraum ist nur aktiv wenn die Variable *bEn* des Kanals auf TRUE gesetzt ist.
Für unregelmäßige, aber wiederkehrende Zeiträume kann die Variable *bResetAfterOn* auf TRUE gesetzt werden. Damit wird die Freigabe *bEn* nach Ablauf der Zeit automatisch auf FALSE zurückgesetzt.



Die Ein- und Ausschaltpunkte eines Kanals müssen in ein und demselben Jahr sein. Der Ausschaltpunkt darf nicht vor dem Einschaltpunkt liegen. Sonst wird der Ausschaltpunkt automatisch korrigiert und gleich gesetzt mit dem Einschaltpunkt.
Ist der Einschaltpunkt gleich dem Ausschaltpunkt, so bleibt der Kanal aus.

VAR_INPUT

```
bEn                 : BOOL;
stSysTi             : TIMESTRUCT;
```

bEn: Generelle Baustein Freigabe.

stSysTi: Struktur mit der lokalen NT-Systemzeit (siehe TIMESTRUCT).

VAR_IN_OUT

```
arrChannel: ARRAY [1..7] OF ST_BA_CalendarChannel;
```

arrChannel: Jahreszeitschaltplan, bei dem einkanaligen Baustein heißt die Variable stChannel (siehe ST_BA_CalenderChannel [▶ 703]). Intern begrenzt auf die jeweilige Anzahl der möglichen Kanäle über die Variable *cBA_NumOfChannels*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2 DUTs

6.1.2.3.2.2.1 Enums

Enumerationen

Name	Beschreibung
E_BA_PosMod [▶ 698]	Enumerator zur Definition des Positioniermodus.
E_BA_ShdObjType [▶ 698]	Enumerator zur Auswahl des Verschattungsobjekt-Typs.
E_BA_Sensor	Enumerator zur Auswahl eines Sensortyps zur Erfassung von Analogwerten.
E_BA_Terminal_KL	Enumerator zur Auswahl der entsprechenden Busklemme.

6.1.2.3.2.2.1.1 E_BA_PosMod

Enumerator zur Definition des Positioniermodus.

```
TYPE E_BA_PosMod :
(
  PosModFix:= 0,
  PosModTab,
  PosModMaxIndc
);
END_TYPE
```

PosModFix: Die Jalousiehöhe nimmt einen festen Wert ein, welcher am Baustein FB_BA_SunPrtc [▶ 621] über den Wert *lrFixPos* eingestellt wird [%].

PosModTab: Die Höhenpositionierung erfolgt mit Hilfe einer Tabelle von 6 Stützpunkten, davon 4 parametrierbar. Aus diesen Punkten wird dann durch lineare Interpolation eine Jalousieposition in Abhängigkeit des Sonnenstandes errechnet. Weitere Beschreibung siehe FB_BA_BldPosEntry [▶ 572].

PosModMaxIndc: Die Positionierung erfolgt unter Angabe des maximal gewünschten Lichteinfalles.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.1.2 E_BA_ShdObjType

Enumerator zur Auswahl des Verschattungsobjekt-Typs.

```
TYPE E_BA_ShdObjType :
(
  ObjTypeTetragon := 0,
  ObjTypeGlobe
);
END_TYPE
```

ObjTypeTetragon: Objekttyp ist ein Viereck.

ObjTypeGlobe: Objekttyp ist eine Kugel.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2 Structures

Strukturen

Name	Beschreibung
ST_BA_BldPosTab [▶ 699]	Struktur der Stützpunkteinträge für die Höhenverstellung der Jalousie.
ST_BA_Cnr [▶ 699]	Information über Fenster-Eckpunkte.
ST_BA_Fcd [▶ 700]	Fassadenspezifische Daten zur Aktivierung der Automatikfunktionen.
ST_BA_FcdElem [▶ 700]	Listeneintrag eines Fassadenelementes (Fenster).
ST_BA_ShdObj [▶ 701]	Listeneintrag eines Verschattungsobjektes
ST_BA_SpRmT [▶ 701]	Raumtemperatur Sollwerte.
ST_BA_SunBld [▶ 702]	Struktur des Jalousie-Positioniertelegramms.
ST_BA_SunBldScn [▶ 703]	Tabelleneintrag einer Jalousie-Szene.
ST_BA_CalenderChannel [▶ 703]	Eingabe von Kalendereinträgen.
ST_BA_SchedulerWeeklyChannel [▶ 704]	Eingabe von Zeitschalteinträgen.

6.1.2.3.2.2.2.1 ST_BA_BldPosTab

Struktur der Stützpunkteinträge für die Höhenverstellung der Jalousie.

```
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: Die 6 Stützstellen welche übergeben werden, wobei die Array-Elemente 0 und 5 die oben erwähnten automatisch generierten Rand-Elemente darstellen.

bVld: Gültigkeitsflag für den Baustein FB_BA_SunPrtc [▶ 621]. Wird vom Baustein FB_BA_BldPosEntry [▶ 572] auf TRUE gesetzt, wenn die eingegebenen Daten den beschriebenen Gültigkeitskriterien entsprechen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.2 ST_BA_Cnr

Information über Fenster-Eckpunkte.

```
TYPE ST_BA_Cnr :
STRUCT
     rX       : REAL;
     rY       : REAL;
     bShdd    : BOOL;
END_STRUCT
END_TYPE
```

rX: X-Koordinate des Fensters (auf der Fassade).

rY: Y-Koordinate des Fensters (auf der Fassade).

bShdd: Information, ob dieser Eckpunkt verschattet ist: *bShdd*=TRUE: Eckpunkt ist verschattet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.3 ST_BA_Fcd

Fassadenspezifische Daten an Raum-Ebene zur Aktivierung der Automatikfunktionen.

```
TYPE ST_BA_Fcd:
STRUCT
  rSunPrtcAngl     : REAL;
  rSunPrtcPos      : REAL;
  rFcdThAutoPos    : REAL;
  rFcdThAutoAngl   : REAL;
  bFcdThAutoEn     : BOOL;
  bThAutoEn        : BOOL;
  bTwiLgtAutoEn    : BOOL;
  bSunPrtcEn       : BOOL;
END_STRUCT
END_TYPE
```

rSunPrtcAngl: Sonnenschutz: Aktuell errechnete Position [%] für die Jalousien.

rSunPrtcPos: Sonnenschutz: Aktuell errechneter Lamellenwinkel [°] für die Jalousien.

rFcdThAutoPos: Thermoautomatik für gesamte Fassade: Aktuell gültige Position [%] für die Jalousien (Heiz- oder Kühlposition).

rFcdThAutoAngl: Thermoautomatik für gesamte Fassade: Aktuell gültiger Lamellenwinkel [°] für die Jalousien (Heiz- oder Kühlposition).

bFcdThAutoEn: Thermoautomatik für gesamte Fassade freigegeben.

bThAutoEn: Thermoautomatik freigegeben.

bTwiLgtAutoEn: Dämmerungsautomatik freigegeben.

bSunPrtcEn: Sonnenschutzautomatik freigegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.4 ST_BA_FcdElem

Listeneintrag eines Fassadenelementes (Fenster).

```
TYPE ST_BA_FcdElem:
STRUCT
  rWdwWdth  : REAL;
  rWdwHght  : REAL;
  stCnr     : ARRAY [1..4] OF ST_BA_Cnr;
  diGrp     : DINT;
  bVld      : BOOL;
END_STRUCT
END_TYPE
```

rWdwWdth: Breite des Fensters.

rWdwHght: Höhe des Fensters.

stCnr: Koordinaten der Fenster-Eckpunkte und Information, ob dieser Eckpunkt verschattet ist, siehe ST_BA_Cnr [▶ 699].

bVld: Plausibilität der eingetragenen Daten: *bVld*=TRUE: Daten sind plausibel.

diGrp: Gruppenzugehörigkeit des Elementes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.5 ST_BA_ShdObj

Listeneintrag eines Verschattungsobjektes.

```
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: Eckkoordinaten. Nur von Bedeutung, wenn das Element ein Viereck ist.

rMx .. rMz: Mittelpunktkoordinaten. Nur von Bedeutung, wenn das Element eine Kugel ist.

rRads: Radius der Kugel. Nur von Bedeutung, wenn das Element eine Kugel ist.

diBegMth: Anfang der Verschattungsperiode (Monatszahl).

diEndMth: Ende der Verschattungsperiode (Monatszahl).

eType: Objekttyp, siehe E_BA_ShdObjType [▶ 698].

bVld: Plausibilität der Daten: *bVld*=TRUE: Daten sind plausibel.

Bemerkung zur Verschattungsperiode:

Die Monatseinträge dürfen nicht 0 und größer 12 sein, andernfalls sind alle Kombinationen möglich.

Beispiele:

Beginn=1, Ende=1: Verschattung im Januar.
Beginn=1, Ende=5: Verschattung von Anfang Januar bis Ende Mai.
Beginn=11, Ende=5: Verschattung von Anfang November bis Ende Mai (des folgenden Jahres).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.6 ST_BA_SpRmT

Raum-Temperatursollwerte.

```
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
```

Die Werte in der Struktur sind mit den Voreinstellwerten deklariert.

Die Variablen haben folgende Bedeutung:

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.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.7 ST_BA_SunBld

Struktur des Jalousie-Positioniertelegramms.

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

rPos: Übergebene Jalousiehöhe [%].

rAngl: Übergebene Lamellenstellung [°].

bManUp: Handbefehl: Jalousie hoch.

bManDwn: Handbefehl: Jalousie herunter.

bManMod: TRUE: Der Handbedienmodus ist aktiv. FALSE: Der Automatikmodus ist aktiv.

bActv: Der Absender des Telegramms ist aktiv. Dieses Bit wird nur von der Prioritätssteuerung FB_BA_SunBldPrioSwi4 [▶ 612] bzw. FB_BA_SunBldPrioSwi8 [▶ 613] ausgewertet. Die Sonnenschutzaktoren FB_BA_SunBldActr [▶ 603] und FB_BA_RolBldActr [▶ 594] beachteten es nicht.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.8 ST_BA_SunBldScn

Tabelleneintrag einer Jalousie-Szene.

```
TYPE ST_BA_SunBldScn:
STRUCT
  rPos     : REAL;
  rAngl    : REAL;
END_STRUCT
END_TYPE
```

rPos: Jalousiehöhe [%].

rAngl: Lamellenstellung [°].

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.9 ST_BA_CalenderChannel

Struktur zur Eingabe von Kalendereinträgen.

```
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: Einschaltpunkt Tag.

udiOn_Month: Einschaltpunkt Monat.

udiOn_hh: Einschaltpunkt Stunde.

udiOn_mm: Einschaltpunkt Minute.

udiOff_Day: Ausschaltpunkt Tag.

udiOff_Month: Ausschaltpunkt Monat.

udiOff_hh: Ausschaltpunkt Stunde.

udiOff_mm: Ausschaltpunkt Minute.

bEn: TRUE -> Freigabe des Kanals, FALSE -> bQ = FALSE

bResetAfterOn: Einmaliges und nicht wiederkehrendes Einschalten.

bQ: Ausgabe Kanalzustand.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.2.2.10 ST_BA_SchedulerWeeklyChannel

Struktur zur Eingabe von Zeitschalteinträgen.

```
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: Einschaltpunkt Stunde.

udiOn_mm: Einschaltpunkt Minute.

udiOn_ss: Einschaltpunkt Sekunde.

udiOff_hh: Ausschaltpunkt Stunde.

udiOff_mm: Ausschaltpunkt Minute.

udiOff_ss: Ausschaltpunkt Sekunde.

bAllActive: Aktivierung der Zeitschaltbedingung für alle Wochentage.

bEn: TRUE -> Freigabe des Kanals, FALSE -> bQ = FALSE.

bMonday: Einschaltpunkt Montag.

bTuesday: Einschaltpunkt Dienstag.

bWednesday: Einschaltpunkt Mittwoch.

bThursday: Einschaltpunkt Donnerstag.

bFriday: Einschaltpunkt Freitag.

bSaturday: Einschaltpunkt Samstag.

bSunday: Einschaltpunkt Sonntag.

bResetAfterOn: Einmaliges und nicht wiederkehrendes Einschalten.

bQ: Ausgabe Kanalzustand.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.3 GVLs

6.1.2.3.2.3.1 Konstanten

Globale Konstanten

```
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: Vergleichswert zur Vermeidung eine Division durch Null.

udiNoActvPrio: Der Wert der Konstanten zeigt an, dass keine Priorität aktiv ist.

udiTiSec: Konstante zur Vorgabe einer Zeit in Sekunden.

wSUNDAY: Konstantwert für Sonntag.

wMONDAY: Konstantwert für Montag.

wTUESDAY: Konstantwert für Dienstag.

wWEDNESDAY: Konstantwert für Mittwoch.

wTHURSDAY: Konstantwert für Donnerstag.

wFRIDAY: Konstantwert für Freitag.

wSATURDAY: Konstantwert für Samstag.

TimeValue24h_ms: Zeitwert für 24 Stunden in Millisekunden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.2.3.2.3.2 Parameter

Globale Parameter

```
VAR_GLOBAL CONSTANT
  usiMaxSunBldScn     : USINT := 20;
  uiMaxRowFcd         : UINT  := 10;
  uiMaxColumnFcd      : UINT  := 20;
  uiMaxShdObj         : UINT  := 20;
  udiMaxDataFileSize  : UDINT  := 100000;
END_VAR
```

usiMaxSunBldScn: Maximale Anzahl an Szenen, die durch den FB_BA_SunBldScn [▶ 614] abgearbeitet werden.

uiMaxRowFcd: Maximale Anzahl an Etagen, die durch die Verschattungskorrektur angesprochen werden (horizontale Anordnung der Fenster).

uiMaxColumnFcd: Maximale Anzahl an Achsen, die durch die Verschattungskorrektur angesprochen werden (vertikale Anordnung der Fenster).

uiMaxShdObj: Maximale Anzahl an Verschattungsobjekten, die Schatten auf die Fassade werfen.

udiMaxDataFileSize: Maximale Dateigröße für die Excel Liste, in Byte, die durch die Bausteine FB_BA_RdFcdElemLst [▶ 586] und FB_BA_RdShdObjLst [▶ 590] gelesen wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT ab v3.1.4024.7	Tc3_BA ab v1.1.6.0

6.1.3 SPS-Project-Templates

6.1.3.1 Standard-SPS-BA-Template

Das *Standard-SPS-BA-Template* ist eine SPS-Vorlage für ein Standard TF8040 Projekt.

Mit dem *Standard-SPS-BA-Template* werden alle notwendigen Bibliotheken und Projekteinstellungen für einen einfachen Start mit TF8040 geladen. Die Vorlage eignet sich sehr gut für den Projekteinstieg mit TF8040.

Aufbau

Die *Standard-Projekt-Vorlage* enthält alle notwendigen Deklarationen, FB-Aufrufe und Bibliotheken für die erste Inbetriebnahme eines TF8040 Controllers.

Vorlage für ein *Building-Automation-SPS-Projekt* mit den folgenden grundlegenden Inhalten:

Einstellungen

- **Task**
  - SPS-Zykluszeit: 45 ms

Referenzen

- **Tc3_BA2**
- **Tc3_BA2_Common**
- **Tc3_BACnetRev14**

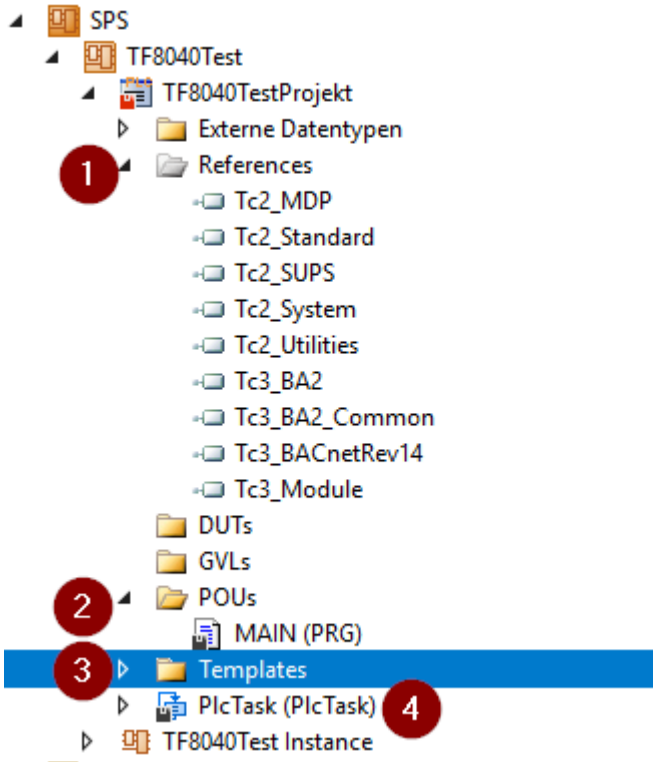
Programme

- **MAIN**
  - FB_BA_DPAD
    Enthält vorkonfigurierte Ebenen für kleine Projekte.
  - Projektstruktur [▶ 39]
    Enthält Deklarationen und FB-Aufrufe für:
    - Schaltschrank
    - Gerät

Templates

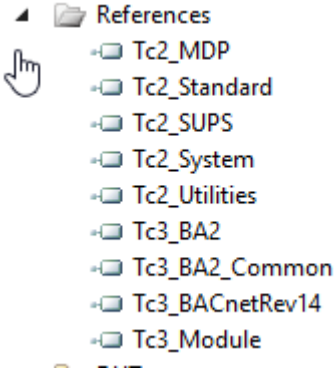
Enthält alle notwendigen Templates, um die Projektvorlage fehlerfrei kompilieren zu können.

Aufbau im Solutiontree



- **Referenzen**

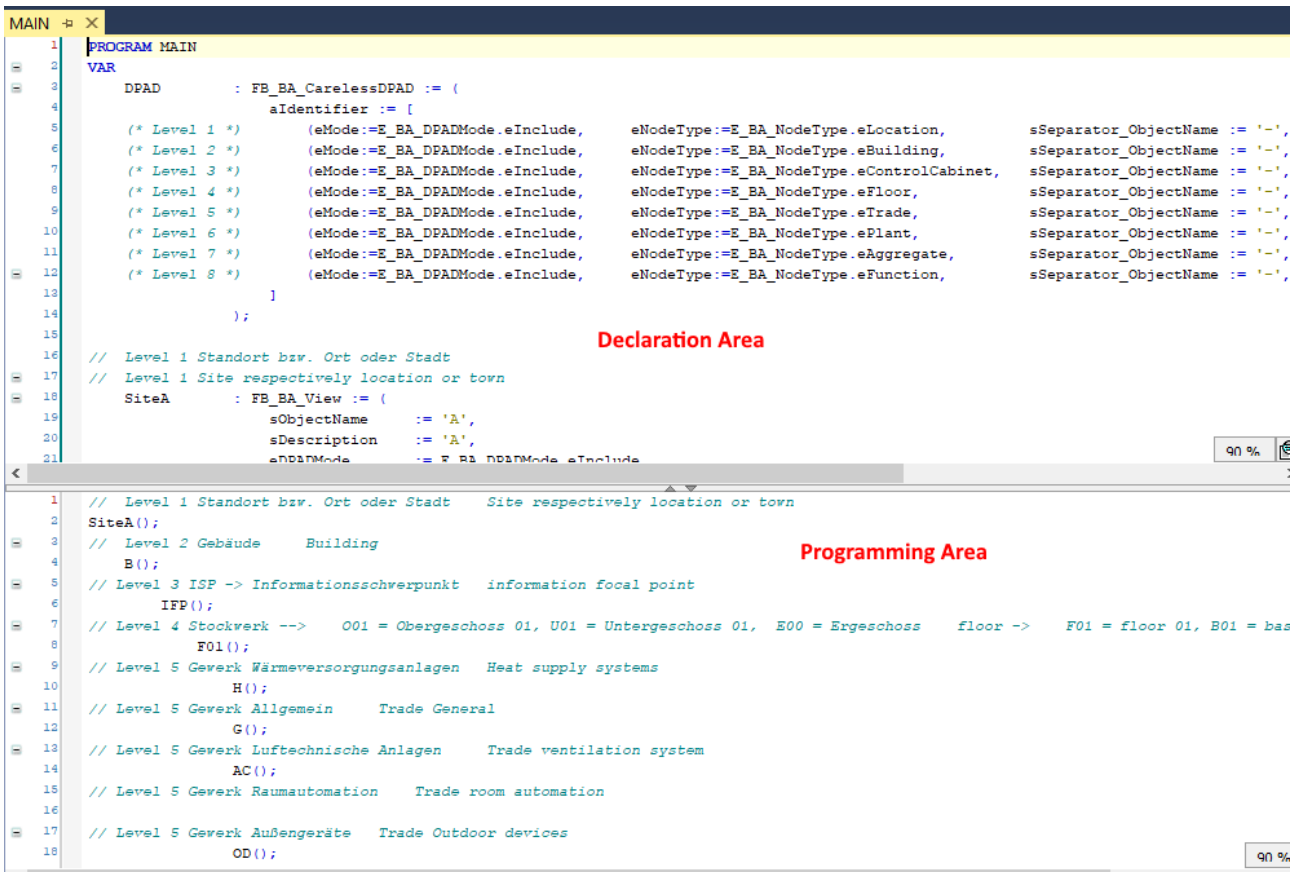
Hier werden alle notwendigen Bibliotheken für ein TF8040 Projekt geladen.



- **MAIN POU**

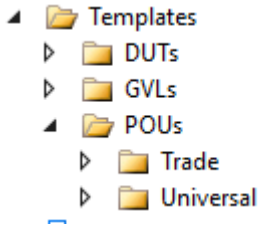
Im MAIN POU wird die Standard-Projektstruktur aufgerufen.

Diese Vorlage liefert eine Projektstruktur als Applikationsbeispiel mit acht Ebenen nach einem Beispiel aus der VDI 3814.



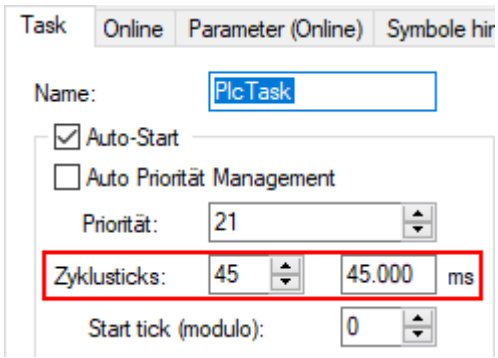
- **Templates**

Alle für den ersten Start notwendigen Templates befinden sich in diesem Ordner. Diese ermöglichen eine fehlerfreie Übersetzung der PLC-Projektvorlage. Weitere Templates zur Realisierung projektspezifischer Applikationen entnehmen Sie dem Template Repository [▶ 1366].



- **SPS-Task**

Die Zykluszeit des SPS-Task ist auf 45 ms eingestellt und sollte aus Gründen der Performance nicht kleiner eingestellt werden.



Projektspezifische Programmierung

- **Kontrolle**

Sobald das Projekt fehlerfrei kompiliert werden kann, sind die Vorbereitungen abgeschlossen.

- **Templates importieren**

Die projektspezifische Programmierung beginnt mit der Integration von Templates aus dem Template Repository [▶ 1366].

Beginnen Sie jetzt mit der projektspezifischen Programmierung.

6.1.4 Templates

Mit den Templates stellt Beckhoff den Anwendern von TwinCAT3-Building-Automation einen großen Systembaukasten zur Verfügung. Die Verwendung von Templates erleichtert die Projektierung und erhöht die Qualität der Projektausführung.

Übersicht Gewerke

Kategorie	Name	Beschreibung
Gewerketemplates	Gewerk [▶ 721]	Air Conditioning, Automationssteuerung, Gebäudeautomation, Raumautomation
Universaltemplates	Universal [▶ 999]	Aggregat, Klappen, Motor, Pumpe, Sensor, Steuerung, Rauchmelder, Ventil, Wetterstation

Jedes Template gibt es in 2 verschiedenen Ausstattungsvarianten bezüglich der Verbindung zur IO-Prozessebene, siehe IO-Mapping. Diese Dokumentation bezieht sich auf die Variante RAW.

6.1.4.1 DUTs

6.1.4.1.1 Enumerations

6.1.4.1.1.1 Building

6.1.4.1.1.1.1 E_BA_BuildingMode

Die Enumeration beschreibt den aktuellen Gebäudezustand in Abhängigkeit des Tagesablaufs.

```
TYPE E_BA_BuildingMode :
(
    eDefault        := 1,
    eNightWatch     := 2,
    eCleaning       := 3
) UDINT;
END_TYPE
```

Name	Beschreibung
eDefault	Standard (Tag-) Betrieb.
eNightWatch	Nachtwächterrundgang
eCleaning	Gebäudereinigung

6.1.4.1.1.2 Plant

6.1.4.1.1.2.1 AC

6.1.4.1.1.2.1.1 E_BA_AC_SeqNumber_H

Sequenznummernregler bei dem die Reihenfolge der Sequenznummern berücksichtigt werden.

```
TYPE E_BA_AC_SeqNumber_H :
(
  eHumidifier       := 1,
  eDehumidifier     := 2,
```

```
  eOff              := 3
) UDINT;
END_TYPE
```

Name	Beschreibung
eHumidifier	Sequenznummer Befeuchter.
eDehumidifier	Sequenznummer Entfeuchter.
eOff	Kein Sequenzregler aktiv.

6.1.4.1.1.2.1.2 E_BA_AC_SeqNumber_T

Sequenznummernregler bei dem die Reihenfolge der Sequenznummern berücksichtigt werden.

```
TYPE E_BA_AC_SeqNumber_T :
(
eReHeater           := 1,
ePreHeater          := 2,
eMixedAir           := 3,
eEnergyRecovery     := 4,
eCooler             := 5,
eOff                := 6
) UDINT;
END_TYPE
```

Name	Beschreibung
eReHeater	Sequenznummer Nacherhitzer.
ePreHeater	Sequenznummer Vorerhitzer.
eMixedAir	Sequenznummer Mischluft.
eEnergyRecovery	Sequenznummer Energierückgewinnung.
eCooler	Sequenznummer Kühler.
eOff	Kein Sequenzregler aktiv.

6.1.4.1.1.2.1.3 E_BA_AC_OpMod02

```
TYPE E_BA_AC_OpMod01 :
(
  Invalid                       := 0,
  eOff                          := 1,
  eStep1                        := 2,
  eStep2                        := 3,
  eStep3                        := 4,
  eEmergency                    := 5,
  eFrost                        := 6,
  eSmokeExtractionProgram       := 7,
  eSmokeExtractionSupplyAir     := 8,
  eSmokeExtractionExhaustAir    := 9,
  eFire                         := 10,
  eNightCooling                 := 11,
  eCoolDownProtection           := 12,
  eOverHeatingProtection        := 13,
  eAlarm                        := 14,
  eForcedVentilation            := 15,
  eCentralSwitchOff             := 16
) UDINT;
END_TYPE
```

Name	Beschreibung
eOff	Anlagenschritt Aus.
eStep1	Stufe 1
eStep2	Stufe 2
eStep3	Stufe 3
eEmergency	Notfall
eFrost	Frost
eSmokeExtractionProgramm	Entrauchung Programm
eSmokeExtractionSupplyAir	Entrauchung Zuluft
eSmokeExtractionExhaustAir	Entrauchung Fortluft
eFire	Feueralarm
eNightCooling	Nachtkühlung
eCoolDownProtection	Stützbetrieb,Auskühlschutz
eOverHeatingProtection	Überhitzungsschutz
eAlarm	Störung
eForcedVentilation	Zwangsbelüftung
eCentralSwitchOff	Zentralabschaltung

6.1.4.1.1.2.1.4 E_BA_AC_PlantStep01

Anlagenschritte im Startprogramm einer Klimaanlage

```
TYPE E_BA_AC_PlantStep01 :
(
  eOff                                := 1,
  eErc                                := 2,
  ePreRinse                           := 3,
  eDamperOuA                          := 4,
  eFanSupplyAir                       := 5,
  eDamperExhA                         := 6,
  eFanExtractAir                      := 7,
  eCooler                             := 8,
  eReheater                           := 9,
  eMixedAir                           := 10,
  eEnablingTemperatureControl         := 11,
  eEnablingHumidityControl            := 12,
  eOn                                 := 13
) UDINT;
END_TYPE
```

Name	Beschreibung
eOff	Anlagenschritt Aus.
eErc	Anlagenschritt Energierückgewinnung.
ePreRinse	Anlagenschritt Vorspülen.
eDamperOuA	Anlagenschritt Fortluftklappe.
eFanSupplyAir	Anlagenschritt Zuluftventilator.
eDamperExhA	Anlagenschritt Außenluftklappe.
eFanExtractAir	Anlagenschritt Abluftventilator.
eCooler	Anlagenschritt Kühler.
eReHeater	Anlagenschritt Nacherhitzer.
eMixedAir	Analgenschritt Mischluft.
eEnablingTemperatureControl	Anlagenschritt Freigabe Temperaturregelung.
eEnablingHumidityControl	Anlagenschritt Freigabe Feuchteregelung.
eOn	Ein

6.1.4.1.1.2.1.5 E_BA_AC_SelSpErc

```
TYPE E_BA_AC_SelSpErc :
(
  eAction           := 1, //E_BA_Action.eDirect = SpCol, E_BA_Action.eReverse = SpHtg,
  eAverageHtgCol    := 2
) UDINT;
END_TYPE
```

6.1.4.1.1.2.1.6 E_BA_AC_OpMod01

```
TYPE E_BA_AC_OpMod01 :
(
  Invalid                       := 0,
  eOff                          := 1,
  eOn                           := 2,
  eEmergency                    := 3,
  eFrost                        := 4,
  eSmokeExtractionProgram       := 5,
  eSmokeExtractionSupplyAir     := 6,
  eSmokeExtractionExhaustAir    := 7,
  eFire                         := 8,
  eNightCooling                 := 9,
  eCoolDownProtection           := 10,
  eOverHeatingProtection        := 11,
  eAlarm                        := 12,
  eForcedVentilation            := 13,
  eCentralSwitchOff             := 14
) UDINT;
END_TYPE
```

Name	Beschreibung
eOff	Anlagenschritt Aus.
eOn	Ein
eEmergency	Notfall
eFrost	Frost
eSmokeExtractionProgramm	Entrauchung Programm
eSmokeExtractionSupplyAir	Entrauchung Zuluft
eSmokeExtractionExhaustAir	Entrauchung Fortluft
eFire	Feueralarm
eNightCooling	Nachtkühlung
eCoolDownProtection	Stützbetrieb,Auskühlschutz
eOverHeatingProtection	Überhitzungsschutz
eAlarm	Störung
eForcedVentilation	Zwangsbelüftung
eCentralSwitchOff	Zentralabschaltung

6.1.4.1.1.3 Universal

6.1.4.1.1.3.1 E_BA_Conversion_kFactor

Die Enumeration zeigt die Einheit des Konversationsfaktors an.

```
{attribute 'qualified_only'}
TYPE E_BA_Conversion_kFactor :
(
  e_Pa              := 0,
  e_hPa             := 1,
  e_mbar            := 2,
  e_mm_HG           := 3,
  e_in_HG           := 4,
  e_mm_WS           := 5,
  e_psi             := 6,
  e_inches_H2O      := 7
) UDINT;
END_TYPE
```

6.1.4.1.1.3.2 E_BA_OnOff

Die Enumeration zeigt die Betriebsarten Ein und Aus an.

```
TYPE E_BA_OnOff :
(
  eOff      := 1,
  eOn       := 2
) UDINT;
END_TYPE
```

Name	Beschreibung
eOff	Anlagenschritt Aus.
eOn	Ein

6.1.4.1.1.3.3 E_BA_EnergyLvlEx

Die Enumeration dient der Darstellung der vier Gebäudeenergieniveaus.

Eine reduzierte Variante ohne das Niveau "*Precomfort*" wird durch E_BA_EnergyLvl [▶ 713] abgebildet.

```
TYPE E_BA_EnergyLvlEx :
(
  eProtection    := 1,
  eEconomy       := 2,
  ePreComfort    := 3,
  eComfort       := 4
) UDINT;
END_TYPE
```

Name	Beschreibung
eProtection	Energieniveau „Protection“.
eEconomy	Energieniveau „Economy“.
ePreComfort	Energieniveau „Precomfort“.
eComfort	Energieniveau „Comfort“.

6.1.4.1.1.3.4 E_BA_EnergyLvl

Die Enumeration dient der Darstellung des Gebäudeenergieniveaus.

Diese reduzierte Enumeration enthält im Gegensatz zu E_BA_EnergyLvlEx [▶ 713] nicht den Zustand "*Precomfort*".

```
TYPE E_BA_EnergyLvl :
(
  eProtection    := 1,
  eEconomy       := 2,
  eComfort       := 3
) UDINT;
END_TYPE
```

Name	Beschreibung
eProtection	Energieniveau *Protection*.
eEconomy	Energieniveau *Economy*.
eComfort	Energieniveau *Comfort*.

6.1.4.1.1.3.5 E_BA_EnergyLvlA

Die Enumeration dient der Darstellung des Gebäudeenergieniveaus. Zusätzlich ist der Zustand "Automatik" aufgeführt, der die Betrachtung eines Automatikbetriebes zulässt.

Diese reduzierte Enumeration enthält im Gegensatz zu E_BA_EnergyLvlExA [▶ 714] nicht den Zustand "Precomfort".

```
TYPE E_BA_EnergyLvlA :
(
  eAutomatic     := 1,
  eProtection    := 2,
  eEconomy       := 3,
  eComfort       := 4
) UDINT;
END_TYPE
```

Name	Beschreibung
eAutomatic	Automatik
eProtection	Energieniveau *Protection*.
eEconomy	Energieniveau *Economy*.
eComfort	Energieniveau *Comfort*.

6.1.4.1.1.3.6 E_BA_EnergyLvlExA

Die Enumeration dient der Darstellung der vier Gebäudeenergieniveaus. Zusätzlich ist der Zustand "Automatik" aufgeführt, der die Betrachtung eines Automatikbetriebes zulässt.

Eine reduzierte Variante ohne das Niveau "*Precomfort*" wird durch E_BA_EnergyLvlA [▶ 713] abgebildet.

```
TYPE E_BA_EnergyLvlExA :
(
  eAutomatic     := 1,
  eProtection    := 2,
  eEconomy       := 3,
  ePreComfort    := 4,
  eComfort       := 5
) UDINT;
END_TYPE
```

Name	Beschreibung
eAutomatic	Automatik
eProtection	Energieniveau *Protection*.
eEconomy	Energieniveau *Economy*.
ePreComfort	Energieniveau *Precomfort*.
eComfort	Energieniveau *Comfort*.

6.1.4.1.2 Types

6.1.4.1.2.1 Actuator

6.1.4.1.2.1.1 ST_BA_SunblindActuatorFeedback

Diese Struktur beinhaltet Rückmeldeinformationen eines Jalousieaktors zu einer Raum- (Zonen-) Anwenderfunktion.

Syntax

```
TYPE ST_BA_SunblindActuatorFeedback:
STRUCT
  bReferencing    : BOOL;
  bErr            : BOOL;
  ePrio           : BYTE;
  fPosition       : REAL;
  fAngle          : REAL;
  fAnglLmtUp      : REAL;
  fAnglLmtDwn     : REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bReferencing	BOOL	Die Jalousiefunktion bzw. der angesteuerte Aktor referenziert gerade.
bErr	BOOL	Die Jalousiefunktion bzw. der angesteuerte Aktor ist im Fehlerzustand.
ePrio	BYTE	Aktuelle Priorität des Telegrammes, welches den Jalousieaktor ansteuert.
fPosition	REAL	Aktuelle Position des Antriebs.
fAngle	REAL	Aktuell angefahrener Winkel der Lamellen.
fAnglLmtUp	REAL	Oberer Grenzwert [▶ 981] der Lamellenstellung zur weiteren Verwendung in Zusammenhang mit der HMI.
fAnglLmtDwn	REAL	Unterer Grenzwert [▶ 981] der Lamellenstellung zur weiteren Verwendung in Zusammenhang mit der HMI.

6.1.4.1.2.1.2 ST_BA_LightActuatorFeedback

Diese Struktur beinhaltet Rückmeldeinformationen eines Lichtaktors zu einer Raum- (Zonen-) Anwenderfunktion.

Syntax

```
TYPE ST_BA_LightActuatorFeedback:
STRUCT
  bInitializing        : BOOL;
  bErr                 : BOOL;
  ePrio                : BYTE;
  fLightValue          : REAL;
  fLightTemperature    : REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bInitializing	BOOL	Die Lichtfunktion bzw. der angesteuerte Aktor wird gerade initialisiert.
bErr	BOOL	Die Lichtfunktion bzw. der angesteuerte Aktor ist im Fehlerzustand.
ePrio	BYTE	Aktuelle Priorität des Telegrammes, welches den Lichtaktor ansteuert.
fLightValue	REAL	Aktueller Lichtwert des Aktors.
fLightTemperature	REAL	Aktuelle Farbtemperatur des Aktors.

6.1.4.1.2.2 Building

6.1.4.1.2.2.1 ST_BA_BuildingLighting

Diese Struktur dient als Sendetelegramm, um gebäudespezifische Freigaben und Informationen, die die Lichtsteuerung betreffen, auf andere Controller zu übermitteln.

Syntax

```
TYPE ST_BA_BuildingLighting:
STRUCT
  stLighting      : ST_BA_Lighting;
  bGlobalReset    : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Resultierendes Beleuchtungstelegramm des Gebäudes bezogen auf die höchste Priorität. Dieses kann folgende Kriterien Feuer oder Einbruch beinhalten (siehe FB_BA_BuildingLighting [▶ 933]).
bGlobalReset	BOOL	Globales Rücksetzen der Beleuchtungsfunktionen.

6.1.4.1.2.2.2 ST_BA_BuildingAlarms

Diese Struktur dient als Sendetelegramm, um gebäudespezifische Alarme auf andere Controller zu übermitteln.

Syntax

```
TYPE ST_BA_BuildingAlarms :
STRUCT
  bBurglary             : BOOL;
  bFireAlert            : BOOL;
  bCentralOff           : BOOL;
  bForcedVentilation    : BOOL;
  bSmokeExtraction      : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bBurglary	BOOL	Einbruchalarm
bFireAlert	BOOL	Feueralarm
bCentralOff	BOOL	Zentraler Aus - Befehl
bForcedVentilation	BOOL	Zwangsbelüftung
bSmokeExtraction	BOOL	Entrauchung

6.1.4.1.2.2.3 ST_BA_BuildingSunblind

Diese Struktur dient als Sendetelegramm, um gebäudespezifische Alarme, Freigaben und Informationen, die die Jalousieansteuerung betrifft, auf andere Controller zu übermitteln.

Syntax

```
TYPE ST_BA_BuildingSunblind :
STRUCT
  stSunBld                      : ST_BA_SunBld;
  bGlobalThAuto_Release         : BOOL;
  bGlobalTwiLgtAuto_Release     : BOOL;
  bGlobalResetManMode           : BOOL;
  nSunPrtc_PositionInterval     : UDINT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Resultierendes Positioniertelegramm aus den gebäudeweiten Alarmen: Feuer, Einbruch oder Vereisung.
bGlobalThAuto_Release	BOOL	Globales Freigabekriterium für die Thermoautomatik.
bGlobalTwiLgtAuto_Release	BOOL	Globales Freigabekriterium für die Dämmerungsautomatik.
bGlobalResetManMode	BOOL	Globales Rücksetzen der Handfunktionen.
nSunPrtc_PositionInterval	UDINT	Positionierintervall der Lamellennachführung bei Nutzung der Sonnenschutzautomatik.

6.1.4.1.2.3 Facade

6.1.4.1.2.3.1 ST_BA_Facade

Diese Struktur dient als Sendetelegramm, zur Übermittlung von fassadenspezifischen Daten auf andere Controller.

Syntax

```
TYPE ST_BA_Facade :
STRUCT
  stSunBld               : ST_BA_SunBld;
  fSunPrtc_Position      : REAL;
  fSunPrtc_Angle         : REAL;
  bThAuto_Release        : BOOL;
  bTwiLgtAuto_Release    : BOOL;
  bSunPrtc_State         : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Resultierendes Jalousietelegramm der Fassade, bezogen auf die höchste Priorität. Dieses kann folgende Kriterien beinhalten: • Einbruch, Feuer oder Eis • Sturmschutz • Wartung • Fassaden-Thermoautomatik • Fassaden-Dämmerungsautomatik • Fassaden-Parkposition
fSunPrtc_Position	REAL	Aktuelle Sonnenschutzposition [%].
fSunPrtc_Angle	REAL	Aktueller Lamellenwinkel [°].
bThAuto_Release	BOOL	Freigabe der Thermoautomatik für die Zonen- / Gruppenansteuerung.
bTwiLgtAuto_Release	BOOL	Freigabe der Dämmerungsautomatik für die Zonen- / Gruppenansteuerung.
bSunPrtc_State	BOOL	Der Sonnenschutz wird aktiviert, wenn er für eine gewisse Einschaltzeit einen bestimmten Einschaltwert überschreitet. Umgekehrt wird er ausgeschaltet, wenn er über eine gewisse Ausschaltzeit einen bestimmten Ausschaltwert unterschreitet.

6.1.4.1.2.4 General

6.1.4.1.2.4.1 ST_BA_GeneralSettings

Diese Struktur beinhaltet allgemeine Witterungsparameter.

Syntax

```
TYPE ST_BA_GeneralSettings :
STRUCT
  fFrostProtectionSetpoint    : REAL;
  fTWth                       : REAL;
  fTWthLowLimit               : REAL;
  bTWthLowLimit               : BOOL;
  fTWthDamped                 : REAL;
  bTWthDampedLowLimit         : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
fFrostProtectionSetpoint	REAL	Frostschutzsollwert z. B. für Heizkreise im Schutzmodus.
fTWth	REAL	Aktueller Wert der Außentemperatur.
fTWthLowLimit	REAL	Unterer Grenzwert der Außentemperatur. Unterhalb dieses Wertes werden alle Frostschutzfunktionen in HVAC-Anlagen aktiviert.
bTWthLowLimit	BOOL	Die Variable ist TRUE, wenn die Außentemperatur unter dem Wert von *fTWthLowCrit* liegt.
fTWthDamped	REAL	Aktueller Wert der gedämpften Außentemperatur.
bTWthDampedLowLimit	BOOL	Die Variable ist TRUE, wenn die gedämpfte Außentemperatur unter dem Wert von *fTWthLowCrit* liegt.

6.1.4.1.2.5 WeatherStation

6.1.4.1.2.5.1 ST_BA_WeatherStation

Diese Struktur dient als Standard-Sendetelegramm für die Daten einer Wetterstation.

Syntax

```
TYPE ST_BA_BuildingAlarms :
STRUCT
  bDisturb                : BOOL;
  fLatitude               : REAL;
  fLongitude              : REAL;
  fSunAzimuth             : REAL;
  fSunElevation           : REAL;
  bRain                   : BOOL;
  stDateTime              : TIMESTRUCT;
  fOutsideTemperature     : REAL;
  fDewPointTemperature    : REAL;
  fPressureAbs            : REAL;
  fPressureRel            : REAL;
  fHumidityAbs            : REAL;
  fHumidityRel            : REAL;
  fBrightnessNorth        : REAL;
  fBrightnessEast         : REAL;
  fBrightnessSouth        : REAL;
  fBrightnessWest         : REAL;
  fDawn                   : REAL;
  fGlobalRadiation        : REAL;
  fWindDirection          : REAL;
  fWindSpeed              : REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bDisturb	BOOL	An der Wetterstation liegt eine Störung vor.
fLatitude	REAL	geographische Breite des Aufstellungsortes [°]
fLongitude	REAL	geographische Länge des Aufstellungsortes [°]
fSunAzimuth	REAL	aktueller Sonnenstand [°]
fSunElevation	REAL	aktuelle Sonnenhöhe [°]
bRain	BOOL	Regensensor
stDateTime	TIMESTRUCT	Datum / Uhrzeit
fOutsideTemperature	REAL	Temperatur [°C]
fDewPointTemperature	REAL	Taupunkttemperatur [°C]
fPressureAbs	REAL	absoluter Luftdruck [hPa]
fPressureRel	REAL	relativer Luftdruck [hPa]
fHumidityAbs	REAL	absolute Feuchte [g/m³]
fHumidityRel	REAL	relative Feuchte [%rF]
fBrightnessNorth	REAL	Helligkeitssensor Norden [kLux]
fBrightnessEast	REAL	Helligkeitssensor Osten [kLux]
fBrightnessSouth	REAL	Helligkeitssensor Süden [kLux]
fBrightnessWest	REAL	Helligkeitssensor Westen [kLux]
fDawn	REAL	Dämmerung [Lux]
fGlobalRadiation	REAL	Globalstrahlung [W/m²]
fWindDirection	REAL	Windrichtung [°]
fWindSpeed	REAL	Windgeschwindigkeit [m/s]

6.1.4.1.2.5.2 ST_BA_WSC11Data

Diese Struktur dient zur Übertragung der Daten einer Thies WSC11 Wetterstation.

Syntax

```
TYPE ST_BA_WSC11Data :
STRUCT
  stStatus                : ST_BA_WSC11Status;
  fCaseTemperature        : REAL;
  fLatitude               : REAL;
  fLongitude              : REAL;
  fSunAzimuth             : REAL;
  fSunElevation           : REAL;
  bRain                   : BOOL;
  stDateTime              : TIMESTRUCT;
  sTimeFormat             : STRING(7);
  fOutsideTemperature     : REAL;
  fDewPointTemperature    : REAL;
  fPressureAbs            : REAL;
  fPressureRel            : REAL;
  fHumidityAbs            : REAL;
  fHumidityRel            : REAL;
  fBrightnessNorth        : REAL;
  fBrightnessEast         : REAL;
  fBrightnessSouth        : REAL;
  fBrightnessWest         : REAL;
  fDawn                   : REAL;
  fGlobalRadiation        : REAL;
  fWindDirection          : REAL;
  fWindSpeed              : REAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
stStatus	ST_BA_WSC11Status [▶ 720]	Statusvariable der WSC11
fCaseTemperature	REAL	Gehäuseinnentemperatur der WSC11 [°C]
fLatitude	REAL	geographische Breite des Aufstellungsortes [°]
fLongitude	REAL	geographische Länge des Aufstellungsortes [°]
fSunAzimuth	REAL	aktueller Sonnenstand [°]
fSunElevation	REAL	aktuelle Sonnenhöhe [°]
bRain	BOOL	Regensensor
stDateTime	TIMESTRUCT	Datum / Uhrzeit
sTimeFormat	STRING(7)	Angabe des Zeitformats
fOutsideTemperature	REAL	Temperatur [°C]
fDewPointTemperature	REAL	Taupunkttemperatur [°C]
fPressureAbs	REAL	absoluter Luftdruck [hPa]
fPressureRel	REAL	relativer Luftdruck [hPa]
fHumidityAbs	REAL	absolute Feuchte [g/m³]
fHumidityRel	REAL	relative Feuchte [%rF]
fBrightnessNorth	REAL	Helligkeitssensor Norden [kLux]
fBrightnessEast	REAL	Helligkeitssensor Osten [kLux]
fBrightnessSouth	REAL	Helligkeitssensor Süden [kLux]
fBrightnessWest	REAL	Helligkeitssensor Westen [kLux]
fDawn	REAL	Dämmerung [Lux]
fGlobalRadiation	REAL	Globalstrahlung [W/m²]
fWindDirection	REAL	Windrichtung [°]
fWindSpeed	REAL	Windgeschwindigkeit [m/s]

6.1.4.1.2.5.3 ST_BA_WSC11Status

Diese Struktur dient zur Übertragung der Status einer Thies WSC11.

Syntax

```
TYPE ST_BA_WSC11Status :
STRUCT
  bDewProtection                        : BOOL;
  bSensorDryingPeriod                   : BOOL;
  bInvalidRMC_Telegramm                 : BOOL;
  bInvalidGPS_Time                      : BOOL;
  bInvalidDataAD_Converter              : BOOL;
  bInvalidDataPressureSensor            : BOOL;
  bInvalidDataBrightnessSensorNorth     : BOOL;
  bInvalidDataBrightnessSensorEast      : BOOL;
  bInvalidDataBrightnessSensorSouth     : BOOL;
  bInvalidDataBrightnessSensorWest      : BOOL;
  bInvalidDataTwilightSensor            : BOOL;
  bInvalidDataSolarRadiationSensor      : BOOL;
  bInvalidDataOutsideTemperatureSensor  : BOOL;
  bInvalidDataRainSensor                : BOOL;
  bInvalidDataWindSpeedSensor           : BOOL;
  bInvalidDataWindDirectionSensor       : BOOL;
  bInvalidDataHumiditySensor            : BOOL;
  bLatestRestartByWatchdogReset         : BOOL;
  bInvalidEEPROM_Parameters             : BOOL;
  bDefaultEEPROM_Parameters             : BOOL;
  bLatestRestartWithNewFirmware         : BOOL;
END_STRUCT
END_TYPE
```

Die folgenden Beschreibungen beziehen sich auf ein TRUE am jeweiligen Parameter.

Name	Typ	Beschreibung
bDewProtection	BOOL	Niederschlagssensor: Betauungsschutz aktiv.
bSensorDryingPeriod	BOOL	Niederschlagssensor: Trocknungsphase der Sensoroberfläche.
bInvalidRMC_Telegramm	BOOL	GPS-Daten: Kein gültiges RMC Telegramm empfangen.
bInvalidGPS_Time	BOOL	RTC Daten vom GPS-Empfänger: Zeit vom GPS-Empfänger sind ungültig.
bInvalidDataAD_Converter	BOOL	Werte vom Analog-Digital-Umsetzer sind ungültig.
bInvalidDataPressureSensor	BOOL	Luftdruck: Messwert vom Drucksensor ist ungültig.
bInvalidDataBrightnessSensorNorth	BOOL	Messwert vom Helligkeitssensor Nord ist ungültig.
bInvalidDataBrightnessSensorEast	BOOL	Messwert vom Helligkeitssensor Ost ist ungültig.
bInvalidDataBrightnessSensorSouth	BOOL	Messwert vom Helligkeitssensor Süd ist ungültig.
bInvalidDataBrightnessSensorWest	BOOL	Messwert vom Helligkeitssensor West ist ungültig.
bInvalidDataTwilightSensor	BOOL	Messwert der Dämmerung ist ungültig.
bInvalidDataSolarRadiationSensor	BOOL	Messwert vom Globalstrahlungssensor ist ungültig.
bInvalidDataOutsideTemperatureSensor	BOOL	Messwert vom Lufttemperatursensor ist ungültig.
bInvalidDataRainSensor	BOOL	Messwert vom Niederschlagssensor ist ungültig.
bInvalidDataWindSpeedSensor	BOOL	Messwert vom Windgeschwindigkeitssensor ist ungültig.
bInvalidDataWindDirectionSensor	BOOL	Messwert vom Windrichtungssensor ist ungültig.
bInvalidDataHumiditySensor	BOOL	Messwerte vom Feuchtesensor ungültig (relative Feuchte, absolute Feuchte, Taupunkttemperatur).
bLatestRestartByWatchdogReset	BOOL	Letzter Neustart durch Watchdog-Reset.
bInvalidEEPROM_Parameters	BOOL	Interne EEPROM-Parameter sind ungültig.
bDefaultEEPROM_Parameters	BOOL	Interne EEPROM-Parameter enthalten die Standardwerte.
bLatestRestartWithNewFirmware	BOOL	Letzter Neustart erfolgte mit neuer Firmware.

6.1.4.2 POUs

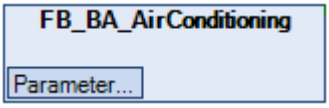
In diesem Kapitel sind die im Template Repositoriy enthaltenen Funktionsbausteine beschrieben.

6.1.4.2.1 Trade

Templates aus den einzelnen Gewerken.

6.1.4.2.1.1 AirConditioning

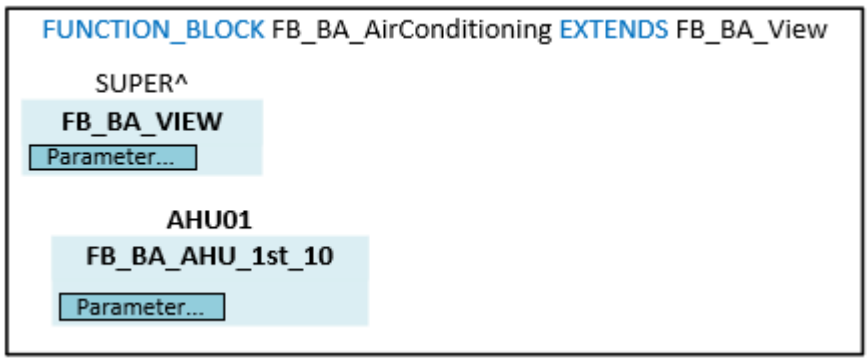
6.1.4.2.1.1.1 FB_BA_AirConditioning



Der Funktionsbaustein dient als Aufruf-Template von Anlagen für das Gewerk *Lufttechnische Anlagen*.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AirConditioning EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AHU01      : FB_BA_AHU_1st_10;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
AHU01	FB_BA_AHU_1st_10 [▶ 823]	Das Anlagen-Template ist eine raumlufttechnische Lüftungsanlage und besteht im Wesentlichen aus den folgenden Teilen: • Anlagensteuerung mit den verschiedenen Betriebsarten und Sollwerten • Temperaturregelung von Sequenzelementen • Nachtkühlung • Druckgeregelte Zu- und Abluftventilatoren • Thermische Luftbehandlung durch Lufterhitzer, Kühler und Energierückgewinnung mittels Plattenwärmetauscher • Außen- und Abluftklappe mit einem Federrücklaufantrieb und Endlagenkontrolle • Außen- und Abluftfilter mit analoger Differenzdruck-Überwachung

Voraussetzungen

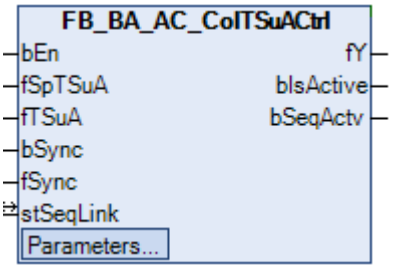
Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2 Aggregates

6.1.4.2.1.1.2.1 Cooler

6.1.4.2.1.1.2.1.1 Temperature

6.1.4.2.1.1.2.1.1.1 FB_BA_AC_ColTSuACtrl



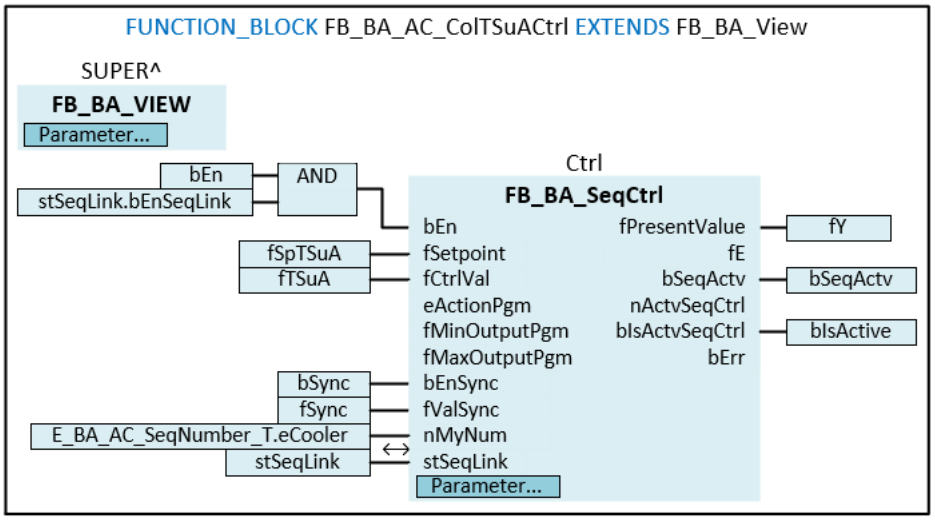
Das Template repräsentiert die Zulufttemperatur Regelung eines Kühlers mit dem Sequenzregler *Ctrl*.

Die Freigabe des Sequenzreglers erfolgt über die Eingangsvariable *bEn*.

Anhand der Daten- und Befehlsstruktur *stSeqLink* erhalten die Sequenzregler die Freigabe der Sequenzregelung. Dieses wird durch die Variable *bSeqActv* angezeigt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ColTSuACtrl EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
  fSpTSuA            : REAL;
  fTSuA              : REAL;
  bSync              : BOOL;
  fSync              : REAL;
END_VAR
VAR_OUTPUT
  fY                 : REAL;
  bIsActive          : BOOL;
  bSeqActv           : BOOL;
END_VAR
VAR_IN_OUT
```

```
  stSeqLink           : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  Ctrl                : FB_BA_SeqCtrl;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe für den Frequenzregler *Ctrl*.
fSpTSuA	REAL	Sollwert der Zulufttemperatur.
fTSuA	REAL	Messwert der Zulufttemperatur.
bSync	BOOL	Impuls für die Synchronisation des Sequenzreglers *Ctrl*.
fSync	REAL	Wert für die Synchronisation des Sequenzreglers *Ctrl*.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Ausgabe der Stellgröße
bIsActive	BOOL	Der Sequenzregler ist der Aktive in der Sequenzregelung.
bSeqActv	BOOL	Der Sequenzregler ist in der Regelsequenz eingebunden.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.

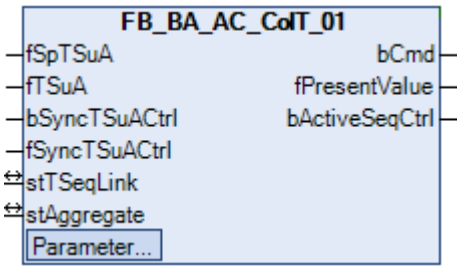
Eingänge CONSTANT

Name	Typ	Beschreibung
Ctrl	FB_BA_SeqCtrl [▶ 172]	Der Zulufttemperatur Sequenzregler *Ctrl* ist das Kernstück dieses Templates. Er ist zuständig für die Zuluft Temperaturregelung des Kühlers. Außerdem ist der Sequenzregler Teil der Zulufttemperatur-Sequenzregelung einer raumlufttechnischen Anlage, siehe Beispiel FB_BA_AC_SeqT [▶ 791]. Der Datenaustausch innerhalb dieser Sequenzregelung geschieht über die Daten- und Befehlsstruktur *stSeqLink*. Durch die globale Variable E_BA_AC_SeqNumber_T.eCooler [▶ 710] erhält der Sequenzregler seine Ordnungsnummer innerhalb der Temperatur Sequenzregelung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.1.1.2 FB_BA_AC_ColT_01



Das Template repräsentiert die Steuerung und Regelung eines Kaltwasser-Luftkühlers ohne Entfeuchteregelung.

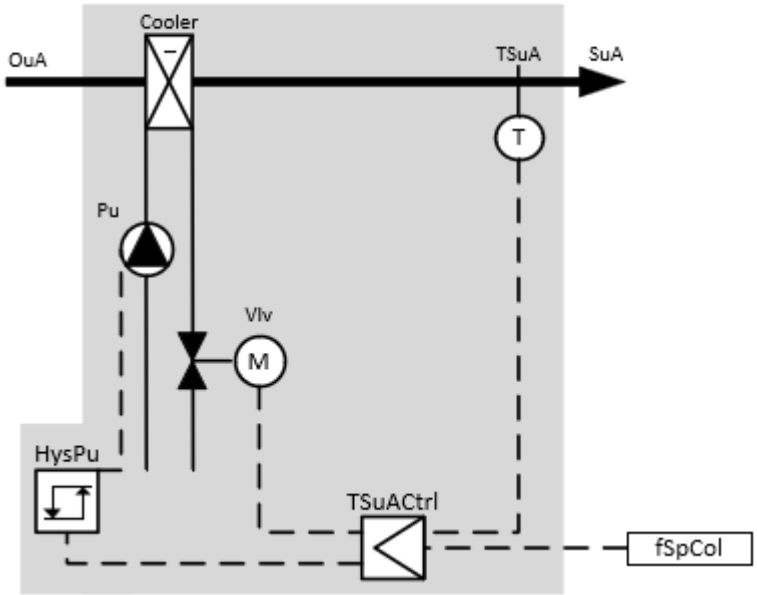
Die wesentlichen Aufgaben des Templates sind:

- Regelung der Zulufttemperatur, siehe *TSuACtrl*
- Freigabe der Kühlerpumpe, siehe *Pu*
- Ansteuerung des Kühlerventils, siehe *Vlv*
- Sammeln und Auswerten der sicherheitsrelevanten Störungen durch das *PlantLock*
- Teil der Schrittkettensteuerung einer raumlufttechnischen Anlage zu sein, siehe *Aggregate*

Prinzipschema 01

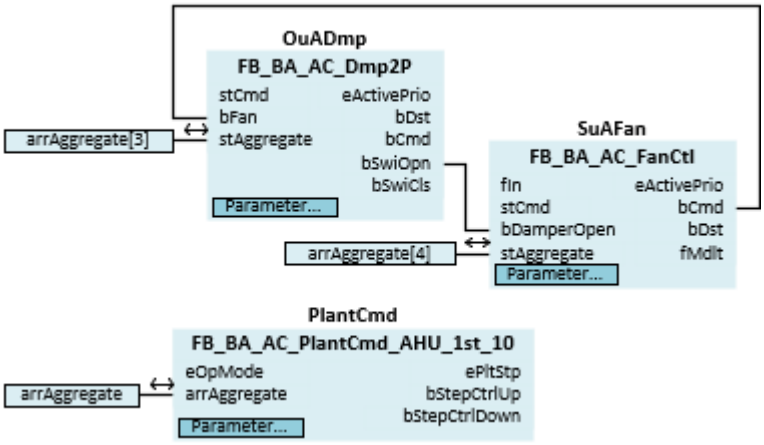
Das Schema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.





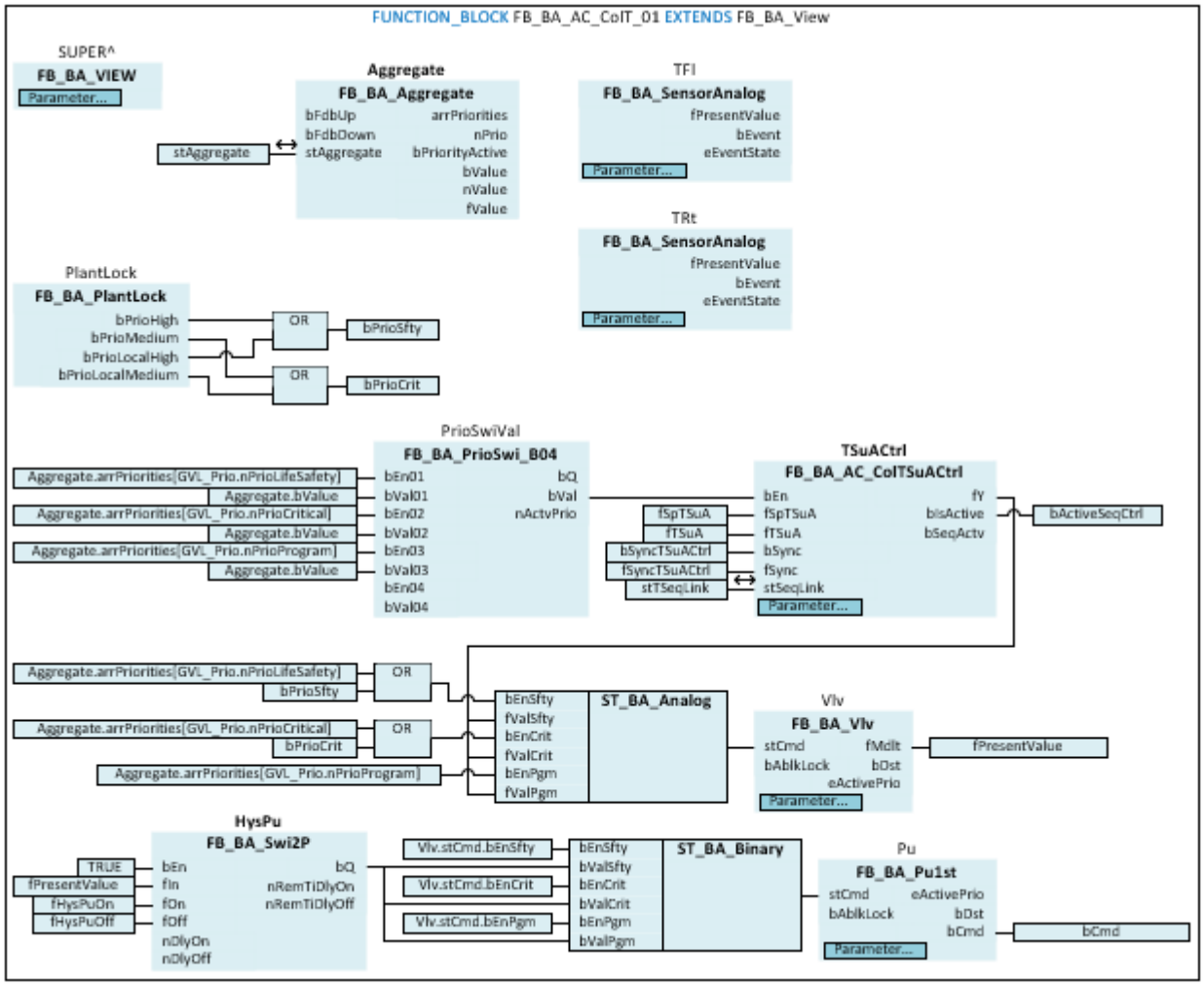
Prinzipschema 02

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ColT_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA           : REAL;
  fTSuA             : REAL;
  bSyncTSuACtrl     : BOOL;
  fSyncTSuACtrl     : REAL;
END_VAR
VAR_OUTPUT
```

```
  bCmd              : BOOL;
  fPresentValue     : REAL;
  bActiveSeqCtrl    : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink        : ST_BA_SeqLink;
  stAggregate       : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOn          : REAL := 5.0;
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOff         : REAL := 1.0;
END_VAR
VAR_INPUT CONSTANT
  TFl               : FB_BA_SensorAnalog;
  TRt               : FB_BA_SensorAnalog;
  TSuACtrl          : FB_BA_AC_ColTSuACtrl;
  Vlv               : FB_BA_Vlv;
  Pu                : FB_BA_Pu1st;
  PlantLock         : FB_BA_PlantLock;
  Aggregate         : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty         : BOOL;
  bPrioCrit         : BOOL;
  PrioSwiVal        : FB_BA_PrioSwi_B04;
  HysPu             : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fSpTSuA	REAL	Sollwert der Zulufttemperatur
fTSuA	REAL	Messwert der Zulufttemperatur
bSyncTSuACtrl	BOOL	Eingang für die Synchronisation des Zuluft Sequenzreglers im Funktionsbaustein *TSuACtrl*.
fSyncTSuACtrl	REAL	Synchronisationswert für den Zuluft Sequenzregler im Funktionsbaustein *TSuACtrl*.

Ausgänge

Name	Typ	Beschreibung
bCmd	BOOL	Aktueller Schaltzustand der einstufigen Pumpe
fPresentValue	REAL	Aktueller Wert des Ventils des Kühlers
bActiveSeqCtrl	BOOL	Zeigt an, dass der Sequenzregler des Kühlers der aktive in der Sequenzregelung ist.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fHysPuOn	REAL	Oberer Schaltpunkt der Hysterese, um die Pumpe einzuschalten.
fHysPuOff	REAL	Unterer Schaltpunkt der Hysterese, um die Pumpe auszuschalten.

Eingänge CONSTANT

Name	Typ	Beschreibung
TFl	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert den Vorlauftemperaturfühler.
TRt	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert den Rücklauftemperaturfühler.
TSuACtrl	FB_BA_AC_ColTSuACtrl [▶ 723]	Der Funktionsbaustein repräsentiert die Zulufttemperatur-Regelung des Kühlers und ist ein Teil der Temperatur-Sequenzregelung einer raumlufttechnischen Anlage. Das Stellsignal wird an das Ventil *Vlv* weitergeleitet.
Vlv	FB_BA_Vlv [▶ 1111]	Der Funktionsbaustein repräsentiert das Ventil.
Pu	FB_BA_Pu1st [▶ 1093]	Der Funktionsbaustein repräsentiert die Kühlerpumpe.
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen des Templates die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen verursachen gezielte Schalthandlungen durch die Variablen *bPrioSfty* und *bPrioCrit* im Template. Die Parametrierung der Lockpriorität der ereignisfähigen Objekte ist im *FB_init* dieses Templates zu finden.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

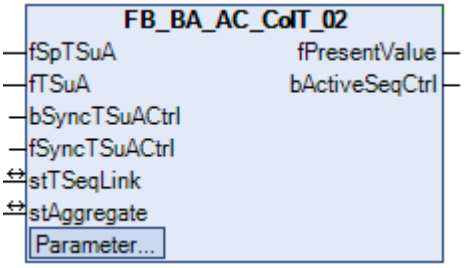
Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Sicherheit“ der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Kritisch“ der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter *PrioSwiVal* ermittelt anhand des Aggregatebausteins *Aggregate* und der globalen Variablenliste Priority [▶ 1138] die Schaltbedingungen und Freigaben für die Zulufttemperatur-Regelung *TSuACtrl*.
HysPu	FB_BA_Swi2P	Der Zweipunktschalter *HysPu* schaltet die Pumpe *Pu* in Abhängigkeit der Ventilstellung *fPresentValue* und der Schaltpunkte der Hysterese *fHysPuOn*/*fHysPuOff* ein und aus.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.1.1.3 FB_BA_AC_ColT_02



Das Template repräsentiert die Steuerung und Regelung eines Kaltwasser-Luftkühlers ohne Entfeuchteregelung.

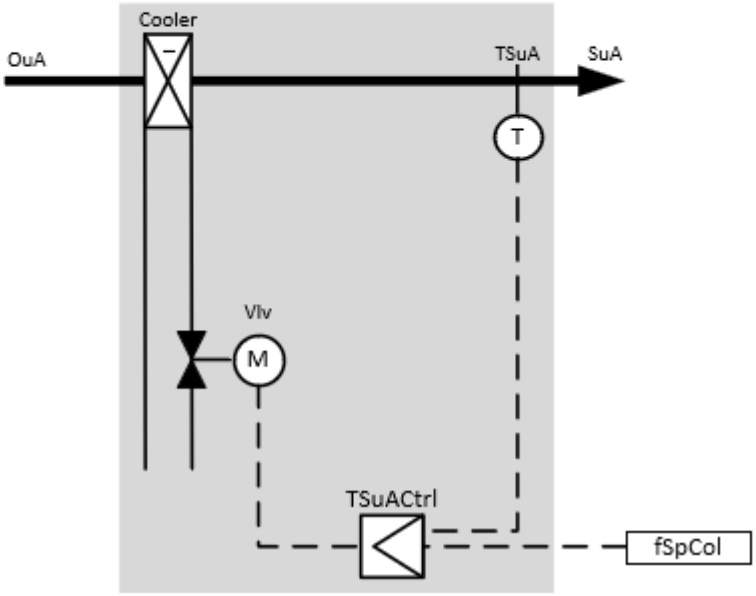
Die wesentlichen Aufgaben des Templates sind:

- Regelung der Zulufttemperatur, siehe *TSuACtrl*
- Ansteuerung des Kühlerventils, siehe *Vlv*
- Sammeln und Auswerten der sicherheitsrelevanten Störungen durch das *PlantLock*
- Teil der Schrittkettensteuerung einer raumlufttechnischen Anlage zu sein, siehe *Aggregate*

Prinzipschema 01

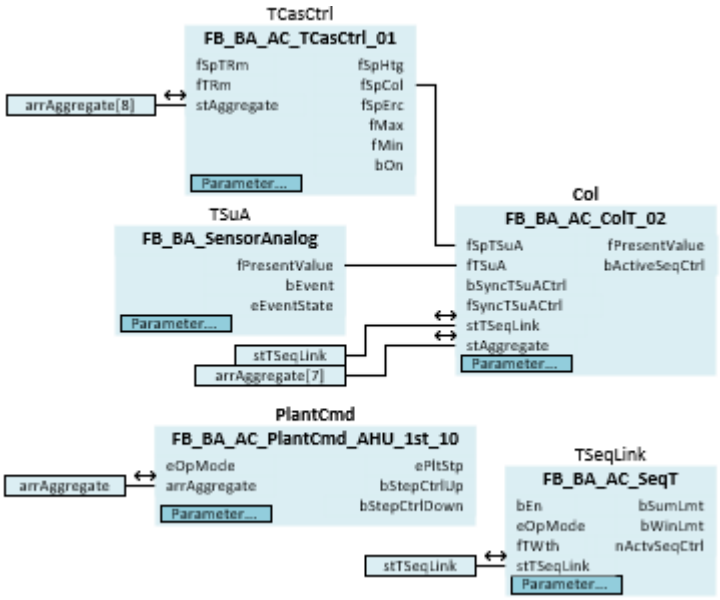
Das Schema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.





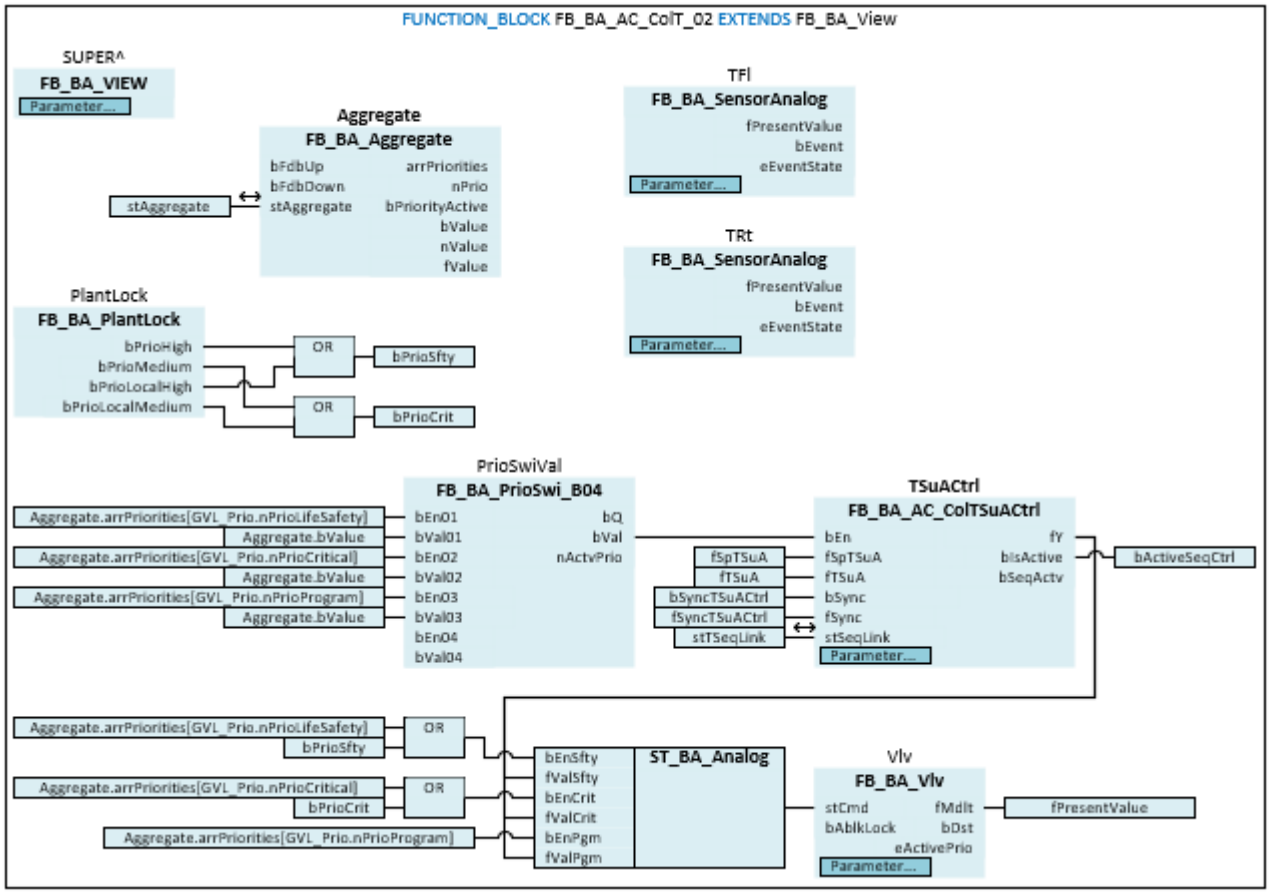
Prinzipschema 02

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ColT_02 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA          : REAL;
  fTSuA            : REAL;
  bSyncTSuACtrl    : BOOL;
  fSyncTSuACtrl    : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue    : REAL;
  bActiveSeqCtrl   : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink       : ST_BA_SeqLink;
  stAggregate      : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  TFl              : FB_BA_SensorAnalog;
  TRt              : FB_BA_SensorAnalog;
  TSuACtrl         : FB_BA_AC_ColTSuACtrl;
  Vlv              : FB_BA_Vlv;
  PlantLock        : FB_BA_PlantLock;
  Aggregate        : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty        : BOOL;
  bPrioCrit        : BOOL;
  PrioSwiVal       : FB_BA_PrioSwi_B04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fSpTSuA	REAL	Sollwert der Zulufttemperatur
fTSuA	REAL	Messwert der Zulufttemperatur
bSyncTSuACtrl	BOOL	Eingang für die Synchronisation des Zuluft Sequenzreglers im Funktionsbaustein *TSuACtrl*.
fSyncTSuACtrl	REAL	Synchronisationswert für den Zuluft Sequenzregler im Funktionsbaustein *TSuACtrl*.

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller Wert des Ventils des Kühlers.
bActiveSeqCtrl	BOOL	Zeigt an, dass der Sequenzregler des Kühlers der aktive in der Sequenzregelung ist.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

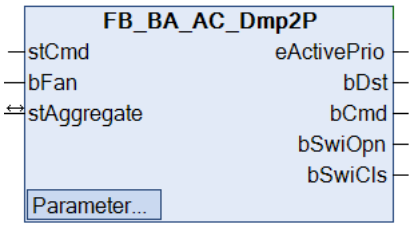
Name	Typ	Beschreibung
TFl	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert den Vorlauftemperaturfühler.
TRt	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert den Rücklauftemperaturfühler.
TSuACtrl	FB_BA_AC_ColTSuACtrl [▶ 723]	Der Funktionsbaustein repräsentiert die Zulufttemperatur-Regelung des Kühlers und ist ein Teil der Temperatur-Sequenzregelung einer raumlufttechnischen Anlage. Das Stellsignal wird an das Ventil *Vlv* weitergeleitet.
Vlv	FB_BA_Vlv [▶ 1111]	Der Funktionsbaustein repräsentiert das Ventil.
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen des Templates die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen verursachen gezielte Schalthandlungen durch die Variablen *bPrioSfty* und *bPrioCrit* im Template. Die Parametrierung der Lockpriorität der ereignisfähigen Objekte ist im *FB_init* dieses Templates zu finden.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Sicherheit“ der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Kritisch“ der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter *PrioSwiVal* ermittelt anhand des Aggregatebausteins *Aggregate* und der globalen Variablenliste Priority [▶ 1138] die Schaltbedingungen und Freigaben für die Zulufttemperatur-Regelung *TSuACtrl*.

6.1.4.2.1.1.2.2 Damper

6.1.4.2.1.1.2.2.1 FB_BA_AC_Dmp2P



Das Aggregate-Template repräsentiert die Ansteuerung einer Zweipunktklappe mit integrierter Überwachung beider Endlagen.

Typische Anwendungen des Templates sind die Außen- und Fortluftklappe einer raumlufttechnischen Anlage.

Ein wesentlicher Bestandteil der Klappensteuerung ist das Template

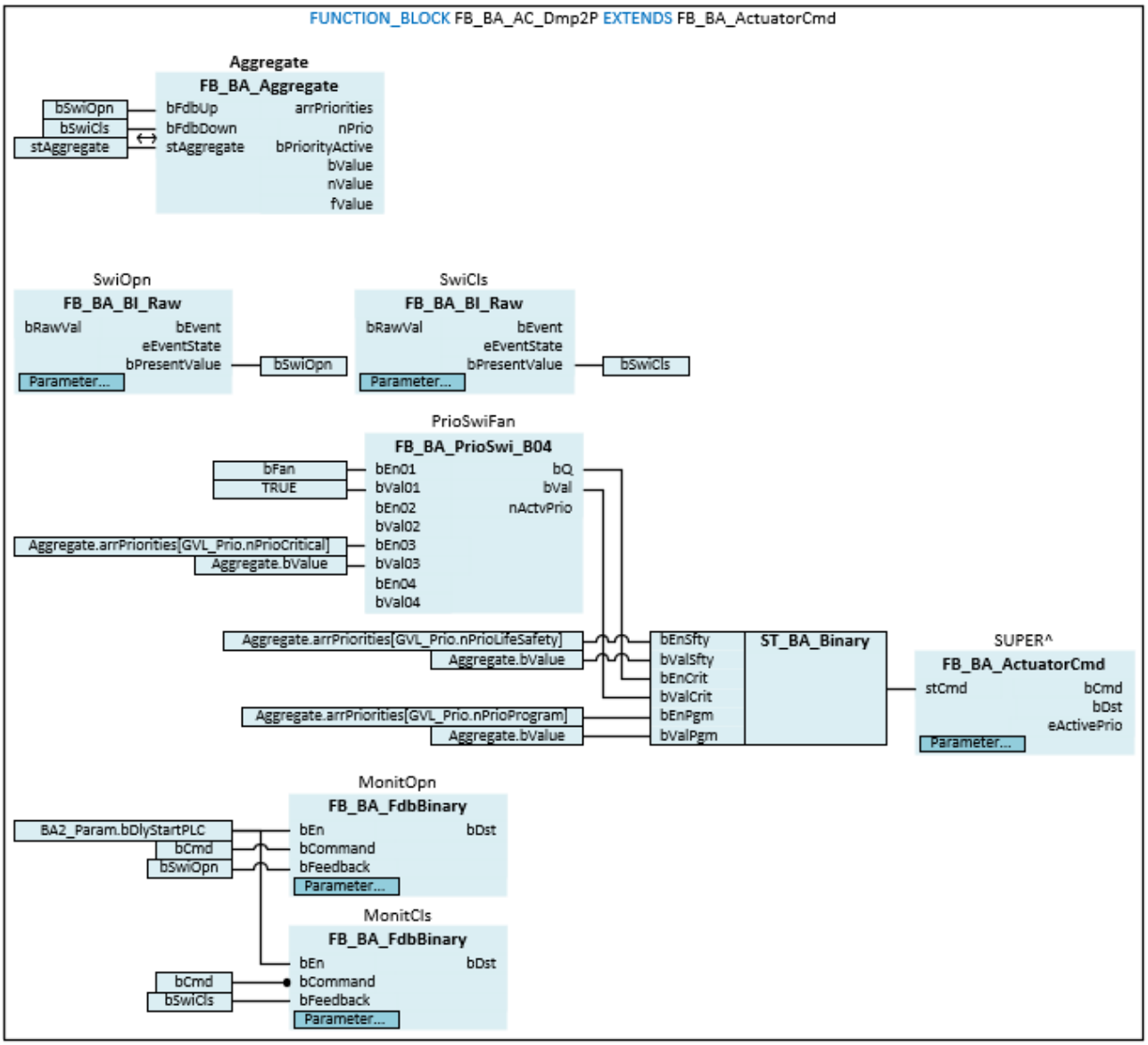
FB_BA_ActuatorCmd [▶ 1004]

Die Einbindung der Klappe in das Anlagensteuerprogramm der Lüftungsanlage erfolgt mit dem Funktionsbaustein *Aggregate*.

Jedes Template gibt es in 2 verschiedenen Ausstattungsvarianten bezüglich der Verbindung zur IO-Prozessebene, siehe IO-Mapping. Diese Dokumentation bezieht sich auf die Variante RAW.

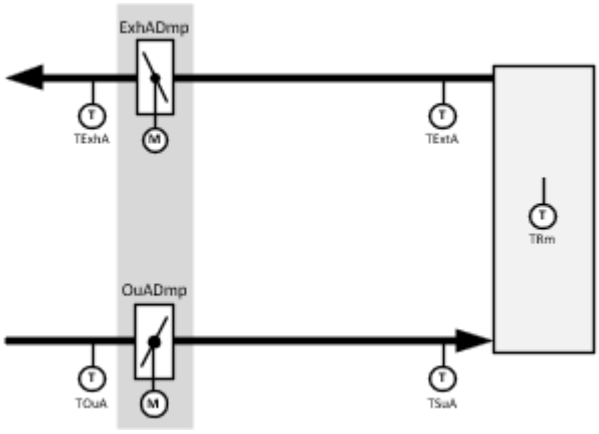
Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



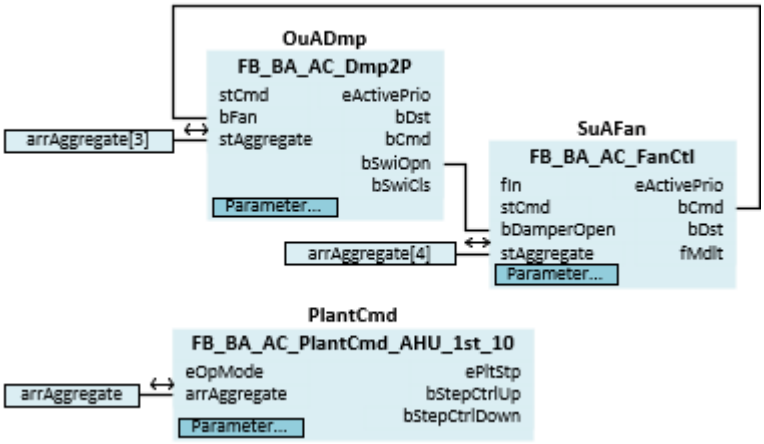
Prinzipschema 01

Das Schema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.



Prinzipschema 02

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Syntax

```
FUNCTION_BLOCK FB_BA_AC_Dmp2P EXTENDS FB_BA_ActuatorCmd
VAR_INPUT
  bFan                    : BOOL;
END_VAR
VAR_OUTPUT
  bSwiOpn                 : BOOL;
  bSwiCls                 : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate             : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  SwiOpn                  : FB_BA_BI_Raw;
  SwiCls                  : FB_BA_BI_Raw;
  MonitOpen               : FB_BA_FdbBinary;
  MonitClose              : FB_BA_FdbBinary;
  Aggregate               : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiFan              : FB_BA_PrioSwi_B04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bFan	BOOL	Das Eingangssignal „Ventilator Ein“ dient innerhalb einer raumlufttechnischen Anlage dem Anlagenschutz der angeschlossenen Klappe. Solange das Eingangssignal „Ventilator Ein“ TRUE ist, wird innerhalb des Templates die Priorität „Kritisch“ am Prioritätenschalter *PrioSwiFan* aktiviert und die Klappe eingeschaltet.

Ausgänge

Name	Typ	Beschreibung
bSwiOpn	BOOL	Endlage „Offen“ der Klappe ist erreicht.
bSwiCls	BOOL	Endlage „Zu“ der Klappe ist erreicht.

Ein- / Ausgänge

Name	Typ	Beschreibung
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung Aggregate übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
SwiOpn	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung des Endlagenschalters Offen.
SwiCls	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung des Endlagenschalters Zu.
MonitOpn	FB_BA_FdbCtrlBinary [▶ 477]	Das Template überwacht die Endlage „Auf“ der Klappe.
MonitCls	FB_BA_FdbCtrlBinary [▶ 477]	Das Template überwacht die Endlage „Zu“ der Klappe.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

Variablen

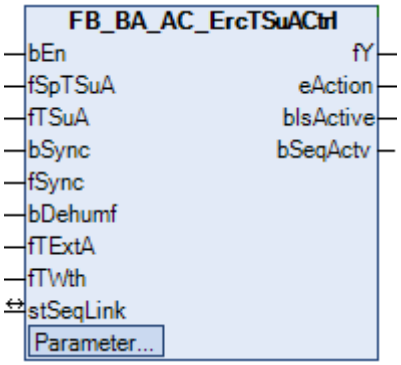
Name	Typ	Beschreibung
PrioSwiFan	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter *PrioSwiFan* ermittelt anhand des Aggregatebausteins *Aggregate* und der Eingangsvariable *bFan* die Freigabebedingungen und Einschaltwerte für die Priorität kritisch der Klappe.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.2.3 ERC

6.1.4.2.1.1.2.3.1 FB_BA_AC_ErcTSuACtrl



Das Template repräsentiert die Zulufttemperatur Regelung einer Energierückgewinnung mit dem Sequenzregler *Ctrl*.

Die Freigabe des Sequenzreglers erfolgt über die Eingangsvariable *bEn*. Bei einem indirekten Wirksinn der Energierückgewinnung (Heizbetrieb) und gleichzeitigem Entfeuchtebetrieb *bDehumf* der raumlufttechnischen Anlage wird die Energierückgewinnung gesperrt.

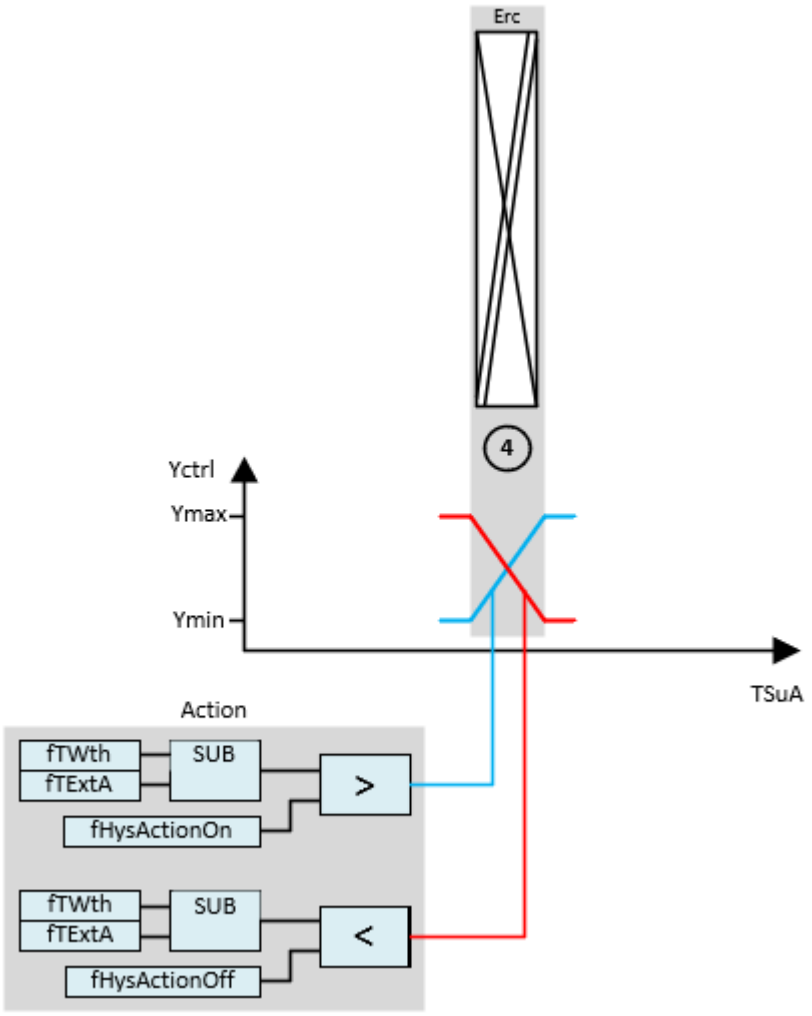
Anhand der Daten- und Befehlsstruktur *stSeqLink* erhalten die Sequenzregler die Freigabe der Sequenzregelung. Dieses wird durch die Variable *bSeqActv* angezeigt.

Wirksinn

Die Auswahl des Wirksinns des Sequenzreglers *Ctrl* erfolgt anhand eines Vergleichs von der Außentemperatur mit der Ablufttemperatur.

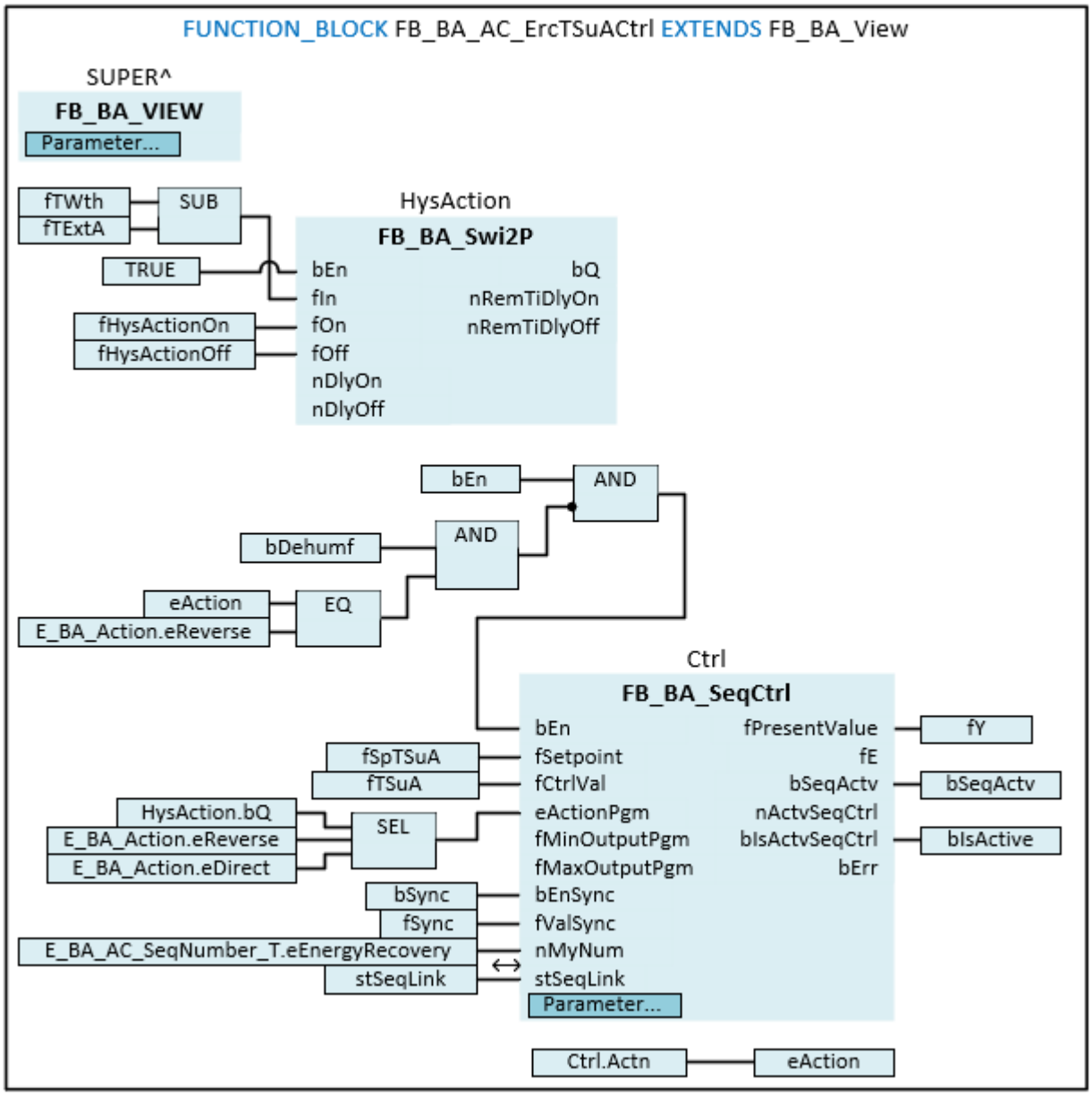
Ist die Außentemperatur kleiner als die Ablufttemperatur, dann ist der Wirksinn des Sequenzreglers *Ctrl* indirekt (Heizbetrieb), siehe E_BA_Action.

Ist die Außentemperatur größer als die Ablufttemperatur, dann ist der Wirksinn des Sequenzreglers *Ctrl* direkt (Kühlbetrieb), siehe E_BA_Action.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcTSuACtrl EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
  fSpTSuA            : REAL;
  fTSuA              : REAL;
  bSync              : BOOL;
  fSync              : REAL;
  bDehumf            : BOOL;
  fTExtA             : REAL;
  fTWth              : REAL;
END_VAR
VAR_OUTPUT
  fY                 : REAL;
  eAction            : E_BA_Action;
  bIsActive          : BOOL;
  bSeqActv           : BOOL;
END_VAR
VAR_IN_OUT
  stSeqLink          : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysActionOn       : REAL := 0.25;
  {attribute 'parameterCategory' := 'Behaviour'}
```

```
  fHysActionOff       : REAL := -0.25;
END_VAR
VAR_INPUT CONSTANT
  Ctrl                : FB_BA_SeqCtrl;
END_VAR
VAR
  HysAction           : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe für den Frequenzregler *Ctrl*.
fSpTSuA	REAL	Sollwert der Zulufttemperatur.
fTSuA	REAL	Messwert der Zulufttemperatur.
bSync	BOOL	Impuls für die Synchronisation des Sequenzreglers *Ctrl*.
fSync	REAL	Wert für die Synchronisation des Sequenzreglers *Ctrl*.
bDehumf	BOOL	Zeigt mit einem TRUE an, dass die raumlufttechnische Anlage im Entfeuchtebetrieb ist.
fTExtA	REAL	Messwert Ablufttemperatur
fTWth	REAL	Messwert Witterungstemperatur

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Ausgabe der Stellgröße
eAction	E_BA_Action	Die Ausgabe des Wirksinns des Zuluftreglers *Ctrl* wird innerhalb einer raumlufttechnischen Anlage für die Sollwertstrategie benötigt.
bIsActive	BOOL	Der Sequenzregler ist der Aktive in der Sequenzregelung.
bSeqActv	BOOL	Der Sequenzregler ist in der Regelsequenz eingebunden.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fHysActionOn	REAL	Oberer Schaltpunkt der Hysterese *HysAction,* um den Wirksinn des Zuluftreglers *Ctrl* zu bestimmen.
fHysActionOff	REAL	Unterer Schaltpunkt der Hysterese *HysAction,* um den Wirksinn des Zuluftreglers *Ctrl* zu bestimmen.

Eingänge CONSTANT

Name	Typ	Beschreibung
Ctrl	FB_BA_SeqCtrl [▶ 172]	Der Zulufttemperatur Sequenzregler *Ctrl* ist das Kernstück dieses Templates. Er ist zuständig für die Zuluft Temperaturregelung des Kühlers. Außerdem ist der Sequenzregler Teil der Zulufttemperatur-Sequenzregelung einer raumlufttechnischen Anlage, siehe Beispiel FB_BA_AC_SeqT [▶ 791]. Der Datenaustausch innerhalb dieser Sequenzregelung geschieht über die Daten- und Befehlsstruktur *stSeqLink*. Durch die globale Variable E_BA_AC_SeqNumber_T.eEnergyRecovery [▶ 710] erhält der Sequenzregler seine Ordnungsnummer innerhalb der Temperatur Sequenzregelung.

Variablen

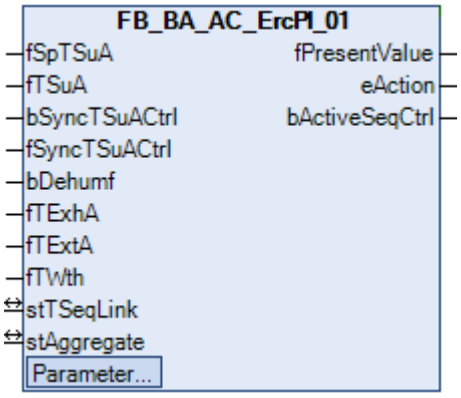
Name	Typ	Beschreibung
HysAction	FB_BA_Swi2P	Der Zweipunktschalter bestimmt den Wirksinn des Sequenzreglers in Abhängigkeit der Außentemperatur *fTWth* und der Ablufttemperatur *fTExtA* und der Schaltpunkte der Hysterese *fHysActionOn*/*fHysActionOff*. Ist die Subtraktion von der Außentemperatur *fTWth* und der Ablufttemperatur *fTExtA* > *fHysActionOn*, dann ist der Ausgang des Funktionsbausteins *HysAction* TRUE. Der Wirksinn des Sequenzreglers *Ctrl* ist damit direkt und die Energierückgewinnung befindet sich im Kühlbetrieb. Ist die Subtraktion von der Außentemperatur *fTWth* und der Ablufttemperatur *fTExtA* < *fHysActionOff*, dann ist der Ausgang des Funktionsbausteins *HysAction* FALSE. Der Wirksinn des Sequenzreglers *Ctrl* ist damit indirekt und die Energierückgewinnung befindet sich im Heizbetrieb.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.3.2 Plate

6.1.4.2.1.1.2.3.2.1 FB_BA_AC_ErcPl_01



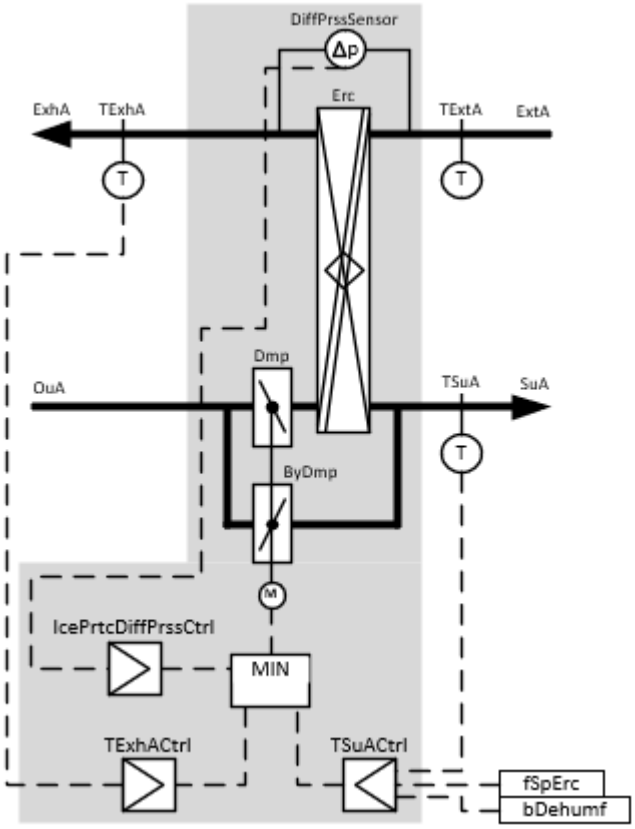
Das Template repräsentiert die Steuerung und Regelung einer Energierückgewinnung mit Plattenwärmetauscher.

Die wesentlichen Aufgaben des Templates sind:

- Regelung der Zulufttemperatur
- Minimalbegrenzung der Fortlufttemperatur
- Vereisungsschutz des Wärmetauschers mittels eines Differenzdruckfühlers über dem Wärmetauscher
- Ansteuerung eines Bypass-Klappensystems
- Sammeln und Auswerten der sicherheitsrelevanten Störungen durch das *PlantLock*
- Teil der Schrittkettensteuerung einer raumlufttechnischen Anlage zu sein, siehe *Aggregate*

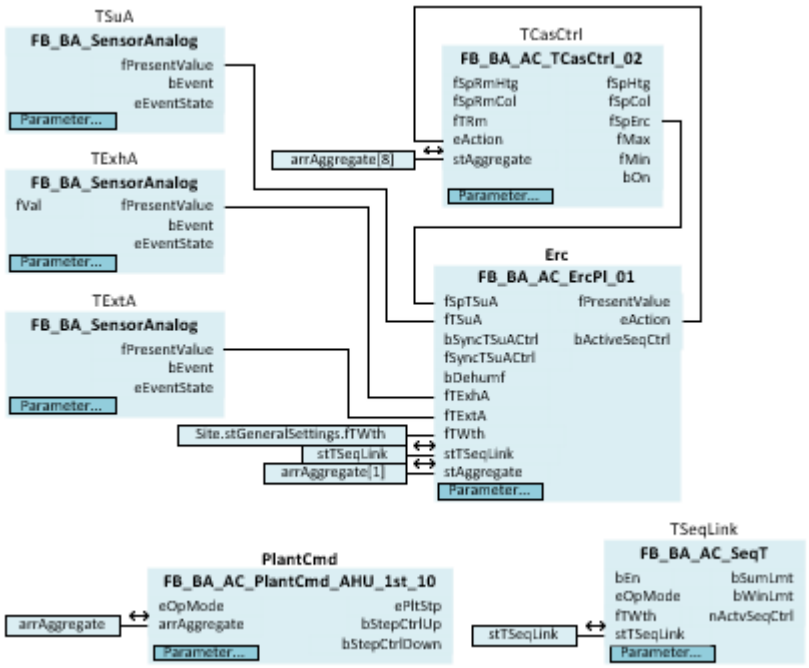
Der Fortluftminimalbegrenzer *TExhACtrl* und das Frostschutzprogramm *IcePrtcDiffPrssCtrl* begrenzen die Stellgröße des Temperatur-Sequenzreglers *TSuACtrl* der Energierückgewinnung über eine Minimum-Auswahl, um einer Vereisung des Wärmetauschers vorzubeugen.

Prinzipschema 01



Das Schema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.

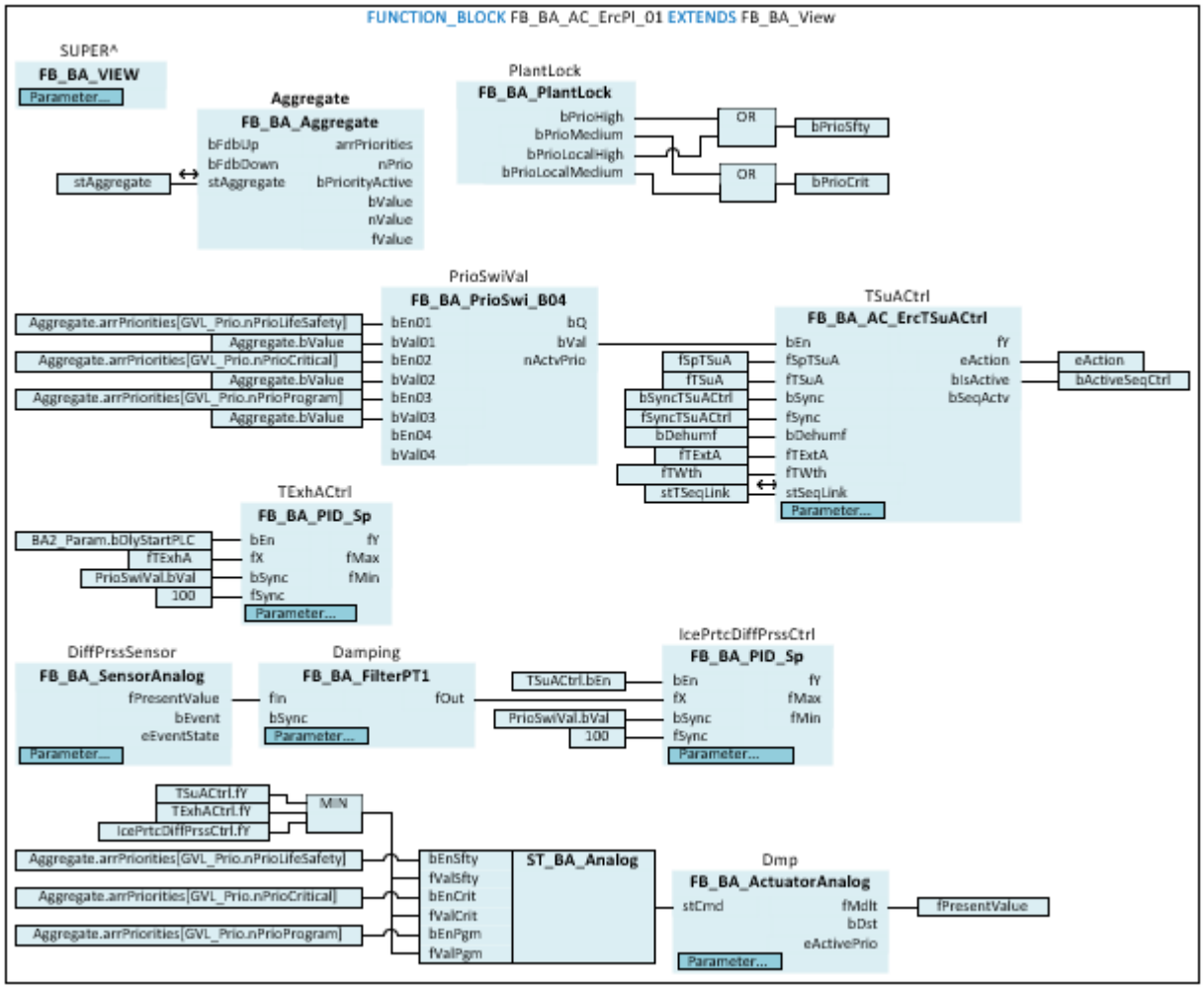
Prinzipschema 02



Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcPl_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA                 : REAL;
  fTSuA                   : REAL;
  bSyncTSuACtrl           : BOOL;
  fSyncTSuACtrl           : REAL;
  bDehumf                 : BOOL;
  fTExhA                  : REAL;
  fTExtA                  : REAL;
  fTWth                   : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue           : REAL;
  eAction                 : E_BA_Action;
  bActiveSeqCtrl          : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink              : ST_BA_SeqLink;
  stAggregate             : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  TSuACtrl                : FB_BA_AC_PreHtrTSuACtrl;
  TExhACtrl               : FB_BA_PID_Sp;
  Dmp                     : FB_BA_ActuatorAnalog;
  DiffPrssSensor          : FB_BA_SensorAnalog;
  Damping                 : FB_BA_FilterPT1;
  IcePrtcDiffPrssCtrl     : FB_BA_PID_Sp;
  PlantLock               : FB_BA_PlantLock;
  Aggregate               : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty               : BOOL;
```

```
  bPrioCrit               : BOOL;
  PrioSwiVal              : FB_BA_PrioSwi_B04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fSpTSuA	REAL	Sollwert der Zulufttemperatur
fTSuA	REAL	Messwert der Zulufttemperatur
bSyncTSuACtrl	BOOL	Impuls für die Synchronisation des Sequenzreglers *TSuACtrl*.
fSyncTSuACtrl	REAL	Wert für die Synchronisation des Zuluft- Sequenzreglers *TSuACtrl*.
bDehumf	BOOL	Eingang Entfeuchtebetrieb aktiv. Dieser Zustand hat Auswirkungen auf die Sequenzsteuerung im Template Zulufttemperatur Regelung *TSuACtrl*.
fTExhA	REAL	Messwert Fortlufttemperatur
fTExtA	REAL	Messwert Ablufttemperatur
fTWth	REAL	Messwert Witterungstemperatur

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller Wert des Bypass-Klappensystems
eAction	E_BA_Action	Die Ausgabe des Wirksinns des Zuluftreglers *TSuACtrl* wird innerhalb einer raumlufttechnischen Anlage für die Sollwertstrategie benötigt.
bActiveSeqCtrl	BOOL	Der Sequenzregler *TSuACtrl* ist der Aktive in der Sequenzregelung.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 759]	Der Funktionsbaustein repräsentiert die Zulufttemperatur Regelung der Energierückgewinnung und ist ein Teil der Temperatur Sequenzregelung einer raumlufttechnischen Anlage. Das Stellsignal wird über eine Minimum-Auswahl an die stetige Ansteuerung der Klappe *Dmp* weitergeleitet.
TExhACtrl	FB_BA_PID_Sp [▶ 1053]	Der Funktionsbaustein repräsentiert die Fortlufttemperaturregelung für den Vereisungsschutz des Plattenwärmetauschers.
Dmp	FB_BA_ActuatorAnalog [▶ 1000]	Der Funktionsbaustein dient der Ansteuerung der stetigen Klappe des Plattenwärmetauschers.
DiffPrssSensor	FB_BA_SensorAnalog [▶ 1106]	Das Analog-Eingangs-Objekt repräsentiert einen Differenzdruckfühler über dem Wärmetauscher. Dieser ist die Regelgröße für die Differenzdruckregelung *IcePrtcDiffPrssCtrl*.
Damping	FB_BA_FilterPT1 [▶ 1021]	Der Funktionsbaustein dient der Dämpfung des Differenzdruckfühler *DiffPrssSensor*.
IcePrtcDiffPrssCtrl	FB_BA_PID_Sp [▶ 1053]	Der Funktionsbaustein repräsentiert die Differenzdruckregelung für den Vereisungsschutz des Wärmetauschers.
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen des Templates die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen verursachen gezielte Schalthandlungen durch die Variablen *bPrioSfty* und *bPrioCrit* im Template. Die Parametrierung der Lockpriorität der ereignisfähigen Objekte ist im *FB_init* dieses Templates zu finden.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

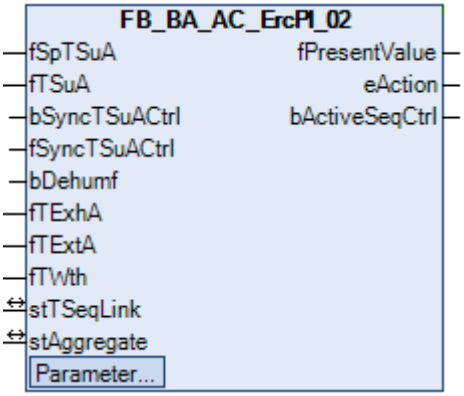
Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität Sicherheit der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität Kritisch der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter *PrioSwiVal* ermittelt anhand des Aggregatebausteins *Aggregate* und der Globalen Variablenliste Priority [▶ 1138] die Schaltbedingungen und Freigaben für die Zulufttemperatur Regelung *TSuACtrl* und der Differenzdruckregelung für den Vereisungsschutz *IcePrtcDiffPrssCtrl*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.3.2.2 FB_BA_AC_ErcPl_02



Das Template repräsentiert die Steuerung und Regelung einer Energierückgewinnung mit einem Plattenwärmetauscher.

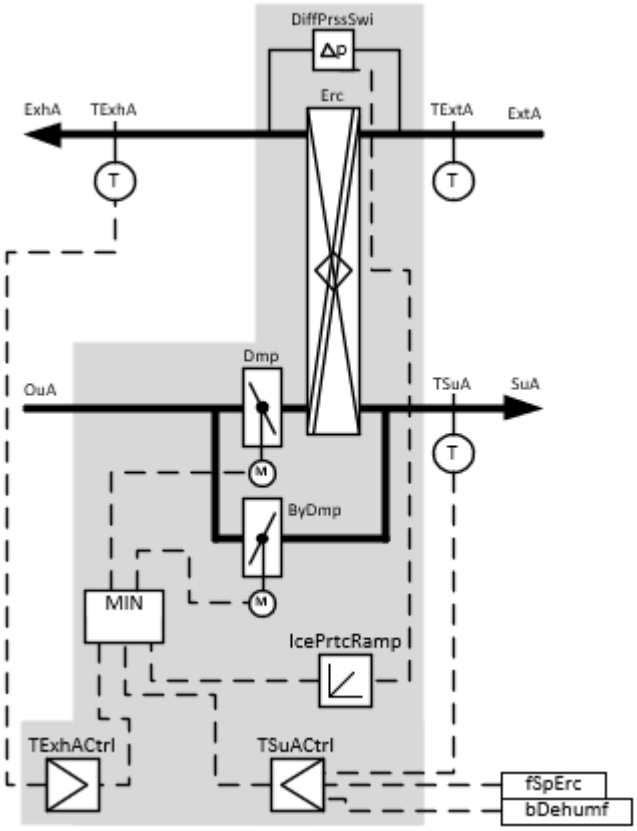
Die wesentlichen Aufgaben des Templates sind:

- Regelung der Zulufttemperatur
- Minimalbegrenzung der Fortlufttemperatur
- Vereisungsschutz des Wärmetauschers mittels eines Differenzdruckwächters über dem Wärmetauscher
- Ansteuerung einer Klappe
- Ansteuerung einer Bypass-Klappe
- Sammeln und Auswerten der sicherheitsrelevanten Störungen durch das *PlantLock*
- Teil der Schrittkettensteuerung einer raumlufttechnischen Anlage zu sein, siehe *Aggregate*

Der Fortluftminimalbegrenzer *TExhACtrl* und das Frostschutzprogramm *IcePrtcRamp* begrenzen die Stellgröße des Temperatur-Sequenzreglers *TSuACtrl* der Energierückgewinnung über eine Minimum-Auswahl, um einer Vereisung des Wärmetauschers vorzubeugen.

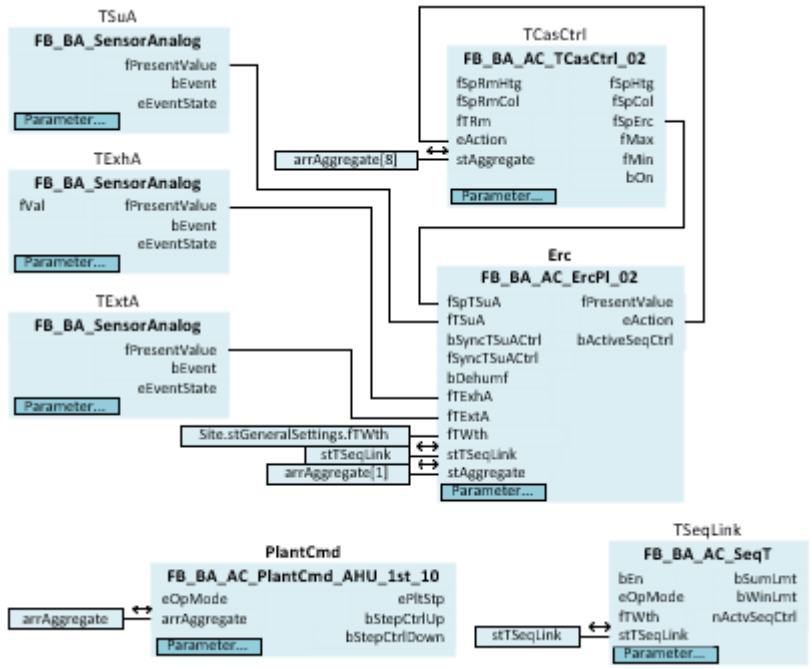
Prinzipschema 01

Das Schema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.



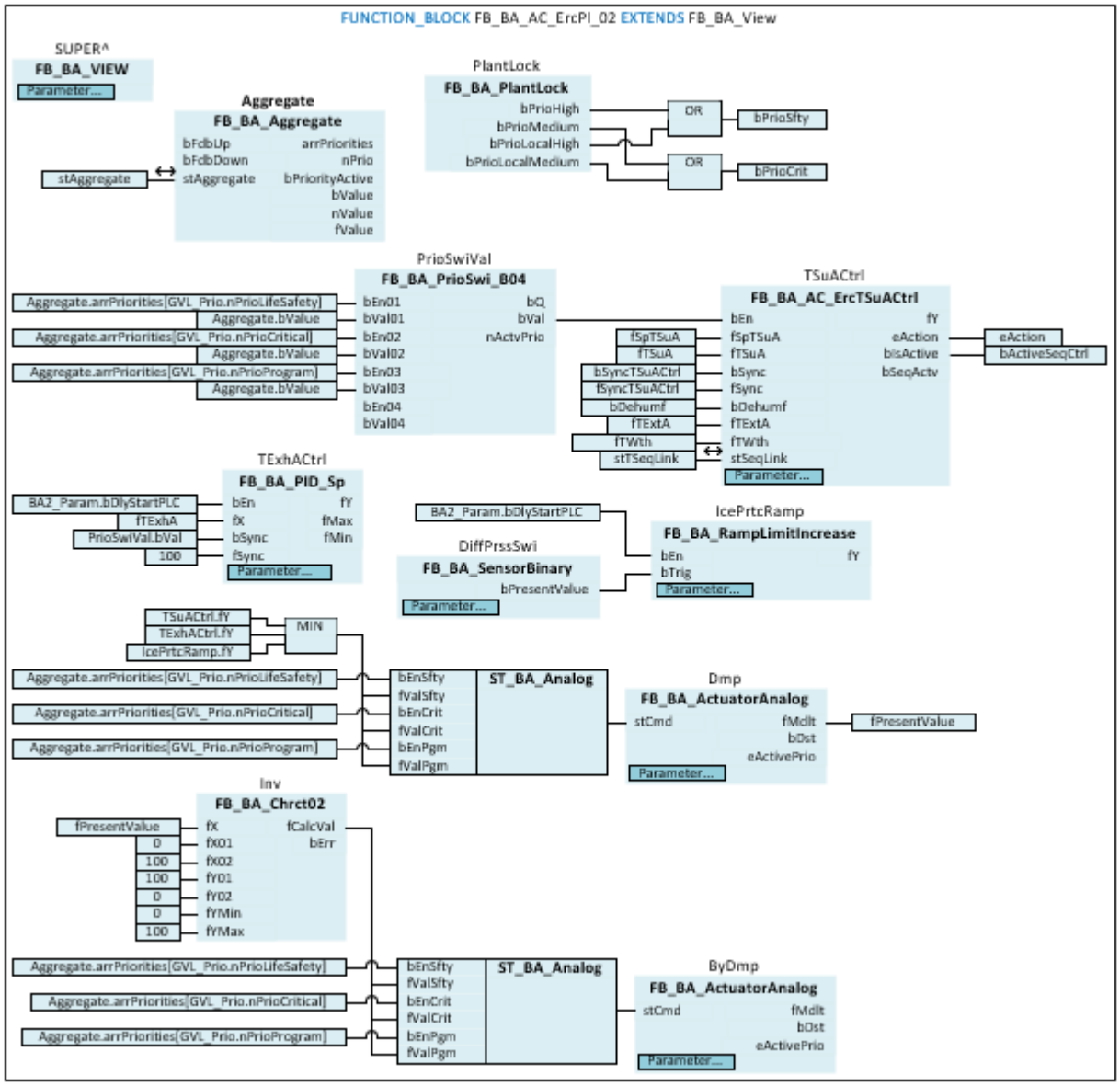
Prinzipschema 02

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcPl_02 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA           : REAL;
  fTSuA             : REAL;
  bSyncTSuACtrl     : BOOL;
  fSyncTSuACtrl     : REAL;
  bDehumf           : BOOL;
  fTExhA            : REAL;
  fTExtA            : REAL;
  fTWth             : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue     : REAL;
  eAction           : E_BA_Action;
  bActiveSeqCtrl    : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink        : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  TSuACtrl          : FB_BA_AC_PreHtrTSuACtrl;
  TExhACtrl         : FB_BA_PID_Sp;
  DiffPrssSwi       : FB_BA_SensorBinary;
  IcePrtcRamp       : FB_BA_PID_Sp;
  Dmp               : FB_BA_ActuatorAnalog;
  ByDmp             : FB_BA_ActuatorAnalog;
```

```
  PlantLock         : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty         : BOOL;
  bPrioCrit         : BOOL;
  PrioSwiVal        : FB_BA_PrioSwi_B04;
  Inv               : FB_BA_Chrct02;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fSpTSuA	REAL	Sollwert der Zulufttemperatur
fTSuA	REAL	Messwert der Zulufttemperatur
bSyncTSuACtrl	BOOL	Impuls für die Synchronisation des Sequenzreglers *TSuACtrl*.
fSyncTSuACtrl	REAL	Wert für die Synchronisation des Zuluft- Sequenzreglers *TSuACtrl*.
bDehumf	BOOL	Eingang Entfeuchtebetrieb aktiv. Dieser Zustand hat Auswirkungen auf die Sequenzsteuerung im Template Zulufttemperatur Regelung *TSuACtrl*.
fTExhA	REAL	Messwert Fortlufttemperatur
fTExtA	REAL	Messwert Ablufttemperatur
fTWth	REAL	Messwert Witterungstemperatur

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller Wert des Bypass-Klappensystems
eAction	E_BA_Action	Die Ausgabe des Wirksinns des Zuluftreglers *TSuACtrl* wird innerhalb einer raumlufttechnischen Anlage für die Sollwertstrategie benötigt.
bActiveSeqCtrl	BOOL	Der Sequenzregler *TSuACtrl* ist der Aktive in der Sequenzregelung.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 759]	Der Funktionsbaustein repräsentiert die Zulufttemperatur Regelung der Energierückgewinnung und ist ein Teil der Temperatur Sequenzregelung einer raumlufttechnischen Anlage. Das Stellsignal wird über eine Minimum-Auswahl an die stetige Ansteuerung der Klappe *Dmp* weitergeleitet.
TExhACtrl	FB_BA_PID_Sp [▶ 1053]	Der Funktionsbaustein repräsentiert die Fortlufttemperaturregelung für den Vereisungsschutz des Plattenwärmetauschers.
DiffPrssSwi	FB_BA_SensorBinary [▶ 1107]	Das Binär-Eingangs-Objekt repräsentiert einen Differenzdruckwächter über dem Wärmetauscher. Dieser wird für den Vereisungsschutz eingesetzt und aktiviert die Rampenfunktion *IcePrtcRamp*.
IcePrtcRamp	FB_BA_RampLimitIncrease [▶ 1058]	Der Funktionsbaustein repräsentiert den Vereisungsschutz des Plattenwärmetauschers anhand einer steigenden Rampenfunktion. Mittels des Differenzdruckwächters *DiffPrssSwi* wird diese Funktion aktiviert.
Dmp	FB_BA_ActuatorAnalog [▶ 1000]	Der Funktionsbaustein dient der Ansteuerung der stetigen Klappe des Plattenwärmetauschers.
ByDmp	FB_BA_ActuatorAnalog [▶ 1000]	Der Funktionsbaustein dient der Ansteuerung der stetigen Bypass Klappe des Plattenwärmetauschers.
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen des Templates die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen verursachen gezielte Schalthandlungen durch die Variablen *bPrioSfty* und *bPrioCrit* im Template. Die Parametrierung der Lockpriorität der ereignisfähigen Objekte ist im *FB_init* dieses Templates zu finden.

Variablen

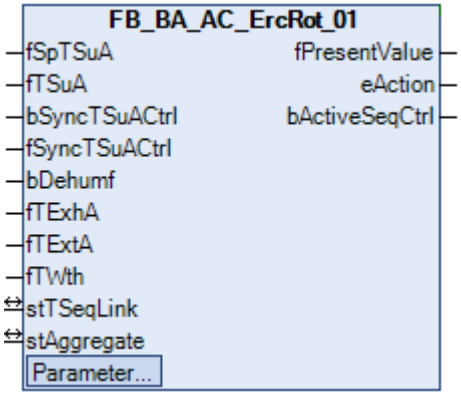
Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität Sicherheit der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität Kritisch der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter *PrioSwiVal* ermittelt anhand des Aggregatebausteins *Aggregate* und der Globalen Variablenliste Priority [▶ 1138] die Schaltbedingungen und Freigaben für die Zulufttemperatur Regelung *TSuACtrl* und der Differenzdruckregelung für den Vereisungsschutz *IcePrtcDiffPrssCtrl*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.3.3 Rotation

6.1.4.2.1.1.2.3.3.1 FB_BA_AC_ErcRot_01



Das Template repräsentiert die Steuerung und Regelung einer Energierückgewinnung mit Rotationswärmetauscher.

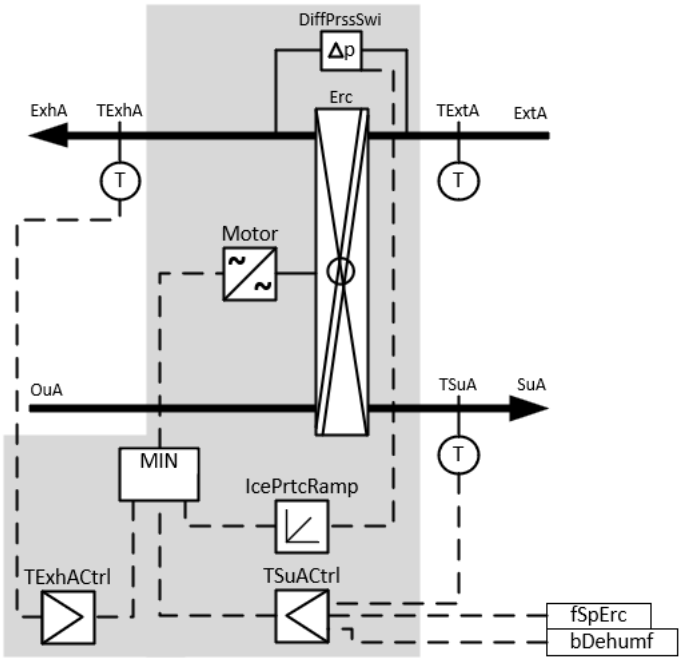
Die wesentlichen Aufgaben des Templates sind:

- Regelung der Zulufttemperatur
- Minimalbegrenzung der Fortlufttemperatur
- Vereisungsschutz des Wärmetauschers mittels Differenzdruckwächter
- Ansteuerung Motor des Rotationswärmetauschers
- Sammeln und Auswerten der sicherheitsrelevanten Störungen durch das *PlantLock*
- Teil der Schrittkettensteuerung einer raumlufttechnischen Anlage zu sein, siehe *Aggregate*

Der Fortluftminimalbegrenzer *TExhACtrl* und das Frostschutzprogramm *IcePrtcRamp* begrenzen die Stellgröße des Temperatur-Sequenzreglers *TSuACtrl* der Energierückgewinnung über eine Minimum-Auswahl, um einer Vereisung des Wärmetauschers vorzubeugen.

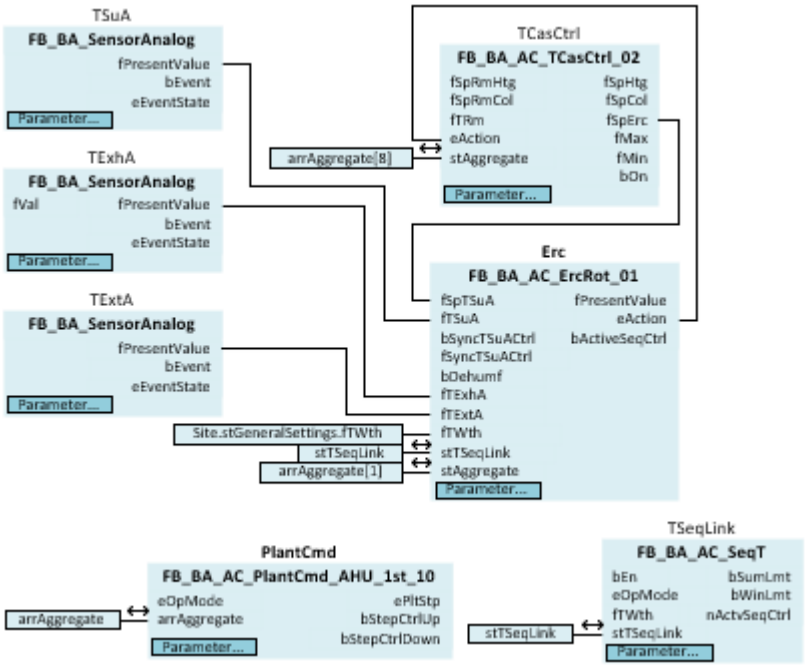
Prinzipschema 01

Das Schema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.



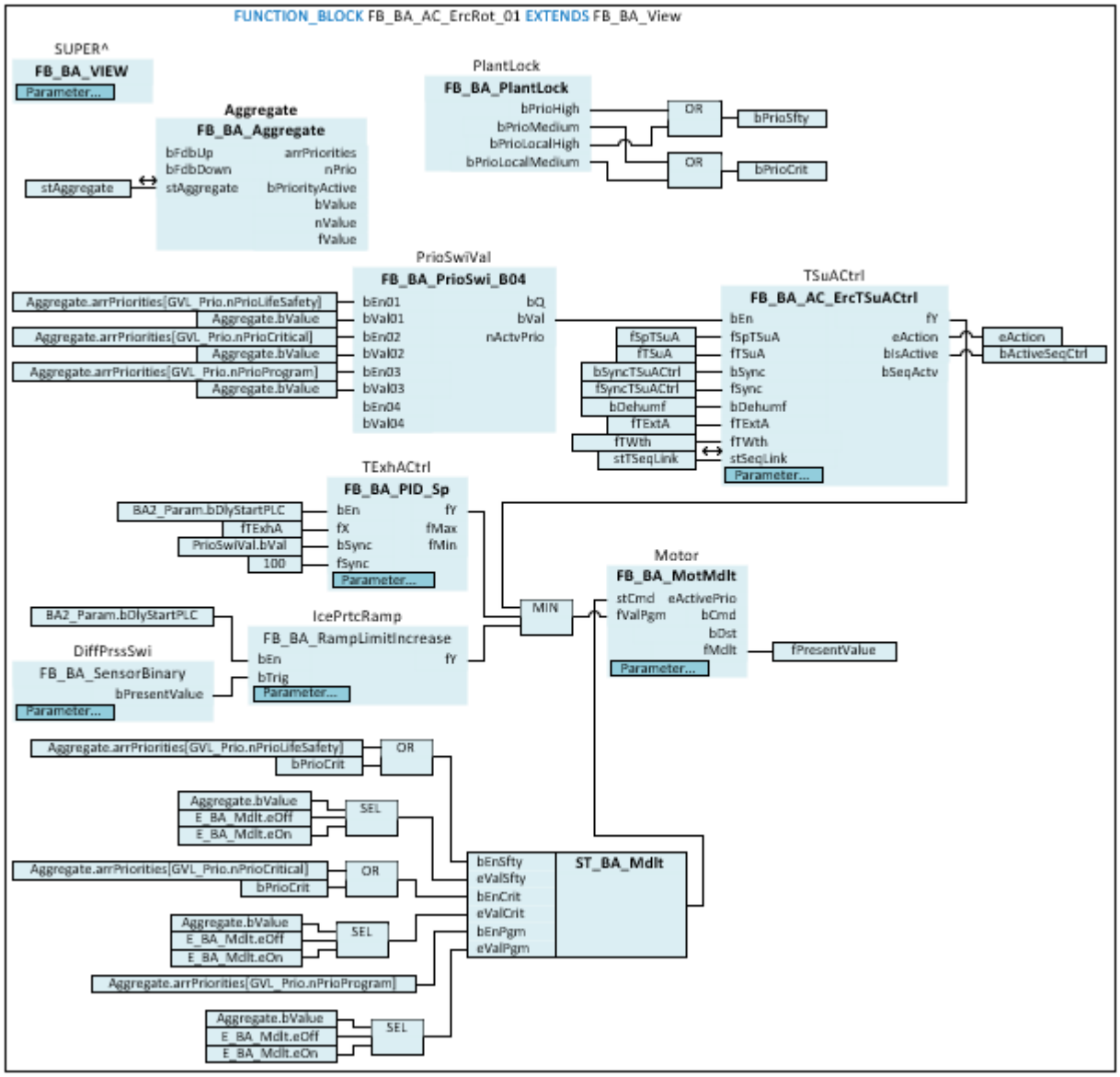
Prinzipschema 02

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_ErcRot_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA           : REAL;
  fTSuA             : REAL;
  bSyncTSuACtrl     : BOOL;
  fSyncTSuACtrl     : REAL;
  bDehumf           : BOOL;
  fTExhA            : REAL;
  fTExtA            : REAL;
  fTWth             : REAL;
END_VAR
VAR_OUTPUT
  fPresentValue     : REAL;
  eAction           : E_BA_Action;
  bActiveSeqCtrl    : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink        : ST_BA_SeqLink;
  stAggregate       : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  TSuACtrl          : FB_BA_AC_PreHtrTSuACtrl;
  TExhACtrl         : FB_BA_PID_Sp;
  DiffPrssSwi       : FB_BA_SensorBinary;
  IcePrtcRamp       : FB_BA_RampLimitIncrease;
  Motor             : FB_BA_MotMdlt;
  PlantLock         : FB_BA_PlantLock;
```

```
  Aggregate          : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty          : BOOL;
  bPrioCrit          : BOOL;
  PrioSwiVal         : FB_BA_PrioSwi_B04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fSpTSuA	REAL	Sollwert der Zulufttemperatur
fTSuA	REAL	Messwert der Zulufttemperatur
bSyncTSuACtrl	BOOL	Impuls für die Synchronisation des Sequenzreglers *TSuACtrl*.
fSyncTSuACtrl	REAL	Wert für die Synchronisation des Zuluft- Sequenzreglers *TSuACtrl*.
bDehumf	BOOL	Eingang Entfeuchtebetrieb aktiv. Dieser Zustand hat Auswirkungen auf die Sequenzsteuerung im Template Zulufttemperatur Regelung *TSuACtrl*.
fTExhA	REAL	Messwert Fortlufttemperatur
fTExtA	REAL	Messwert Ablufttemperatur
fTWth	REAL	Messwert Witterungstemperatur

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller Wert des Bypass-Klappensystems
eAction	E_BA_Action	Die Ausgabe des Wirksinns des Zuluftreglers *TSuACtrl* wird innerhalb einer raumlufttechnischen Anlage für die Sollwertstrategie benötigt.
bActiveSeqCtrl	BOOL	Der Sequenzregler *TSuACtrl* ist der Aktive in der Sequenzregelung.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 759]	Der Funktionsbaustein repräsentiert die Zulufttemperatur Regelung der Energierückgewinnung und ist ein Teil der Temperatur Sequenzregelung einer raumlufttechnischen Anlage. Das Stellsignal wird über eine Minimum-Auswahl an die stetige Ansteuerung der Klappe *Dmp* weitergeleitet.
TExhACtrl	FB_BA_PID_Sp [▶ 1053]	Der Funktionsbaustein repräsentiert die Fortlufttemperaturregelung für den Vereisungsschutz des Plattenwärmetauschers.
DiffPrssSwi	FB_BA_SensorBinary [▶ 1107]	Das Binär-Eingangs-Objekt repräsentiert einen Differenzdruckwächter über dem Wärmetauscher. Dieser wird für den Vereisungsschutz eingesetzt und aktiviert die Rampenfunktion *IcePrtcRamp*.
IcePrtcRamp	FB_BA_RampLimitIncrease [▶ 1058]	Der Funktionsbaustein repräsentiert den Vereisungsschutz des Rotationswärmetauschers anhand einer steigenden Rampenfunktion. Mittels des Differenzdruckwächters *DiffPrssSwi* wird diese Funktion aktiviert.
Motor	FB_BA_MotMdlt [▶ 1089]	Der Funktionsbaustein dient der Ansteuerung eines Frequenzumformers. Dieser steuert dann den Motor des Rotationswärmetauschers.
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen des Templates die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen verursachen gezielte Schalthandlungen durch die Variablen *bPrioSfty* und *bPrioCrit* im Template. Die Parametrierung der Lockpriorität der ereignisfähigen Objekte ist im *FB_init* dieses Templates zu finden.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

Variablen

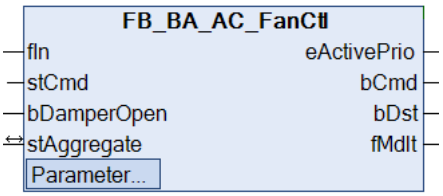
Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität Sicherheit der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität Kritisch der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter *PrioSwiVal* ermittelt anhand des Aggregatebausteins *Aggregate* und der Globalen Variablenliste Priority [▶ 1138] die Schaltbedingungen und Freigaben für die Zulufttemperatur Regelung *TSuACtrl* und der Differenzdruckregelung für den Vereisungsschutz *IcePrtcDiffPrssCtrl*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.4 Fan

6.1.4.2.1.1.2.4.1 FB_BA_AC_FanCtl



Das Aggregate-Template repräsentiert die Steuerung und Regelung eines druckgeregelten Ventilators. Typische Anwendungen dieses Templates sind die Zu- und Abluftventilatoren einer raumlufttechnischen Anlage mit variablen Volumenstrom.

Das Basistemplate FB_BA_MotCtl [▶ 1085] ist um zwei binäre Eingänge zur Erfassung einer zusätzlichen Störung und eines Wartungsschalters erweitert.

Mittels des Eingangs *bDamperOpen* wird das Starten des Ventilators bei einer geschlossenen Klappe verhindert.

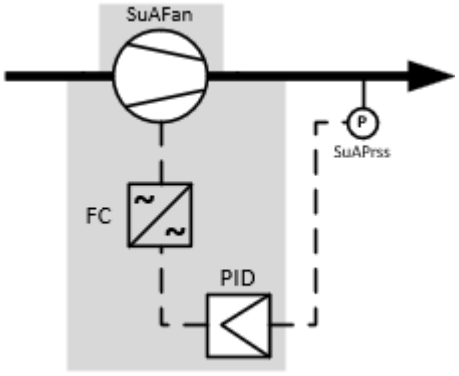
Mit dem Funktionsbaustein *Aggregate* und der Ein/Ausgangs-Struktur stAggregate [▶ 283] wird der druckgeregelte Ventilator in das Steuerungsprogramm der raumlufttechnischen Anlage integriert.

Jedes Template gibt es in 2 verschiedenen Ausstattungsvarianten bezüglich der Verbindung zur IO-Prozessebene, siehe IO-Mapping. Diese Dokumentation bezieht sich auf die Variante RAW.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

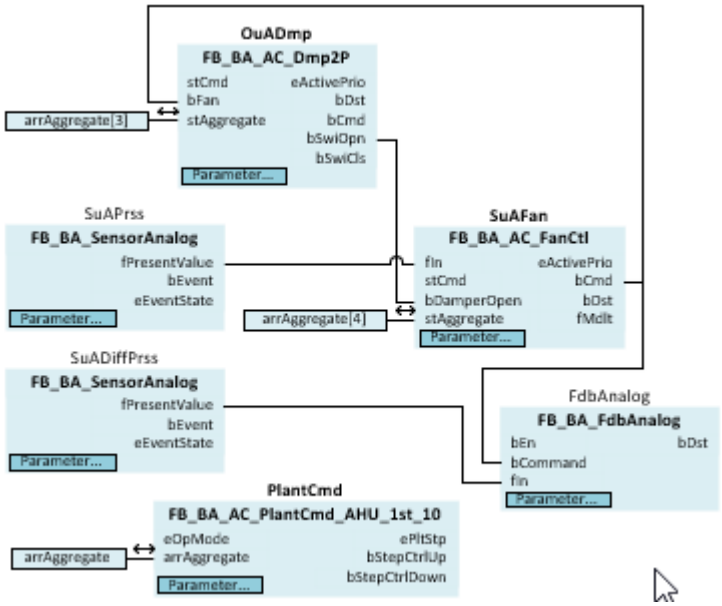
Prinzipschema 01

Das Schema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.

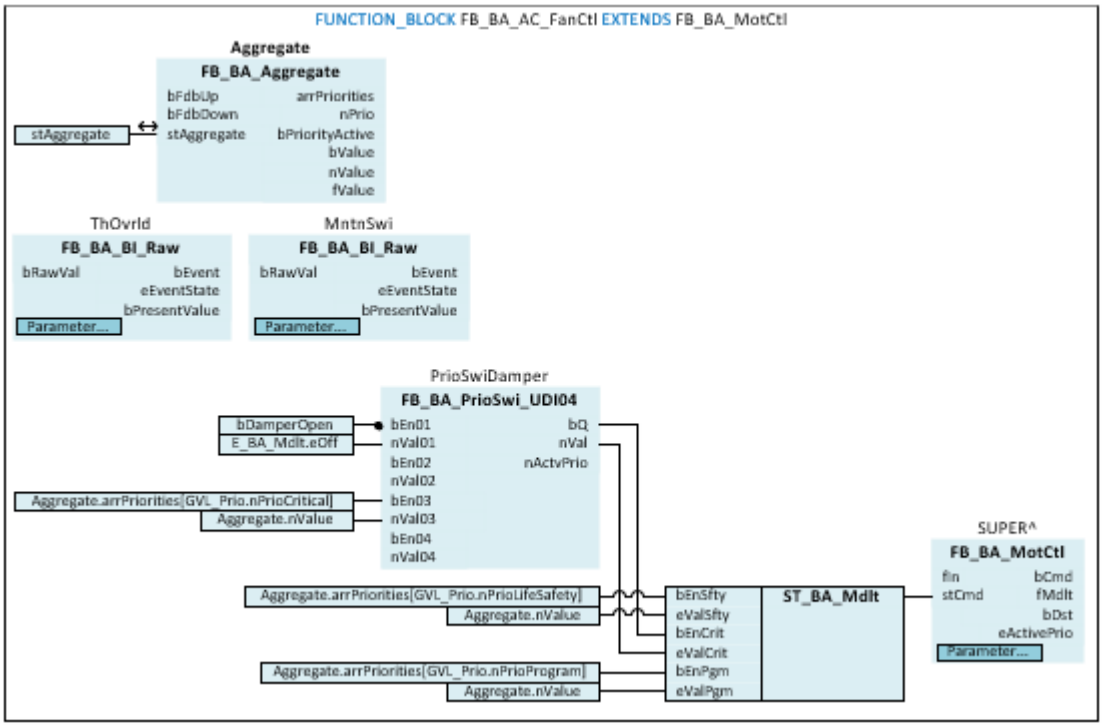


Prinzipschema 02

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_FanCtl EXTENDS FB_BA_MotCtl
VAR_INPUT
  bDamperOpen       : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate       : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  ThOvrld           : FB_BA_BI_Raw;
  MntnSwi           : FB_BA_BI_Raw;
  Aggregate         : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiDamper     : FB_BA_PrioSwi_UDI04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bDamperOpen	BOOL	Das Eingangssignal „Klappe Auf“ dient innerhalb einer raumlufttechnischen Anlage dem Anlagenschutz der angeschlossen Klappe. Solange das Eingangssignal „Klappe Auf“ nicht TRUE ist, wird innerhalb des Templates die Priorität „Kritisch“ am Prioritätenschalter *PrioSwiDamper* aktiviert und der Ventilator ausgeschaltet.

Ein- / Ausgänge

Name	Typ	Beschreibung
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung Aggregate übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
ThOvrld	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Störung "Thermische Überlastung".
MntnSwi	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung eines Wartungsschalters.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

Variablen

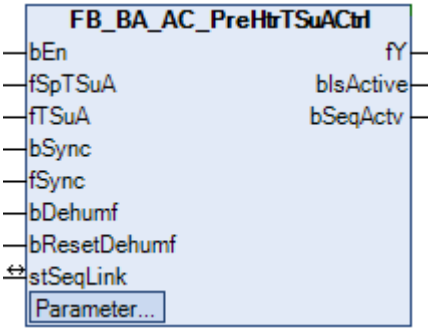
Name	Typ	Beschreibung
PrioSwiDamper	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter *PrioSwiDamper* ermittelt anhand des Aggregatebausteins *Aggregate* und der Eingangsvariable *bDamperOpen* die Freigabebedingungen und Einschaltwerte für die Priorität „Kritisch“ des Ventilators.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.2.5 PreHeater

6.1.4.2.1.1.2.5.1 FB_BA_AC_PreHtrTSuACtrl



Das Template repräsentiert die Zulufttemperaturregelung eines Vorerhitzers in einer Lüftungsanlage. Der Zuluftregler *Ctrl* ist das Kernstück des Templates. Er ist in eine Zulufttemperaturregelsequenz eingebunden.

Die Freigabe des Sequenzreglers erfolgt anhand der Eingangsvariablen *bEn* und der Variable *stSeqLink.bEnSeqLink* der Daten- und Befehlsstruktur *stSeqLink*.

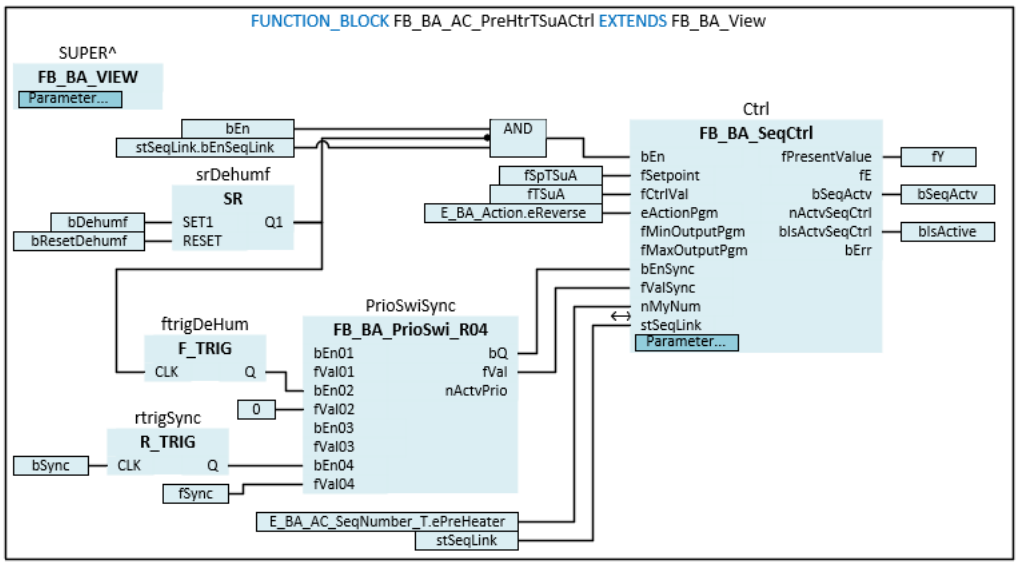
Im Entfeuchtebetrieb *bDehumf* wird der Sequenzregler *Ctrl* gesperrt und damit aus der Sequenzregelung herausgenommen.

Über die Daten- und Befehlsstruktur *stSeqLink* erhalten die Sequenzregler die Freigabe der Sequenzregelung. Dieses wird durch die Variable *bSeqActv* angezeigt.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  fSpTSuA               : REAL;
  fTSuA                 : REAL;
  bSync                 : BOOL;
  fSync                 : REAL;
  bDehumf               : BOOL;
  bResetDehumf          : BOOL;
END_VAR
VAR_OUTPUT
```

```
  fY                    : REAL;
  bIsActive             : BOOL;
  bSeqActv              : BOOL;
END_VAR
VAR_IN_OUT
  stSeqLink             : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  Ctrl                  : FB_BA_SeqCtrl;
END_VAR
VAR
  srDehumf              : SR;
  ftrigDeHum            : F_TRIG;
  rtrigSync             : R_TRIG;
  PrioSwiSync           : FB_BA_PrioSwi_R04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Sequenzregler *Ctrl*.
fSpTSuA	REAL	Sollwert der Zulufttemperatur.
fTSuA	REAL	Messwert der Zulufttemperatur.
bSync	BOOL	Impuls für die Synchronisation des Sequenzreglers *Ctrl*.
fSync	REAL	Wert für die Synchronisation des Sequenzreglers *Ctrl*.
bDehumf	BOOL	Eingang, um den Entfeuchtebetrieb am RS-Flipflop *srDehumf* zu setzen.
bResetDehumf	BOOL	Eingang, um den Entfeuchtebetrieb am RS-Flipflop *srDehumf* zurückzusetzen.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Stellgrößenausgabe Regelventil.
bIsActive	BOOL	Der Sequenzregler ist der Aktive in der Sequenzregelung.
bSeqActv	BOOL	Der Sequenzregler ist in der Regelsequenz eingebunden.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur *stSeqLink* sind das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung TSeqLink [▶ 791] einer raumlufttechnischen Anlage.

Eingänge CONSTANT

Name	Typ	Beschreibung
Ctrl	FB_BA_SeqCtrl	Der Zulufttemperatur Sequenzregler *Ctrl* ist das Kernstück dieses Templates. Er ist zuständig für die Zuluft-Temperaturregelung des Vorerhitzers. Außerdem ist der Sequenzregler Teil der Zulufttemperatur Sequenzregelung einer raumlufttechnischen Anlage. Der Datenaustausch innerhalb dieser Sequenzregelung geschieht über die Daten- und Befehlsstruktur *stSeqLink*. Durch die globale Variable E_BA_AC_SeqNumber_T.ePreHeater [▶ 710] erhält der Sequenzregler seine Ordnungsnummer innerhalb der Temperatur Sequenzregelung.

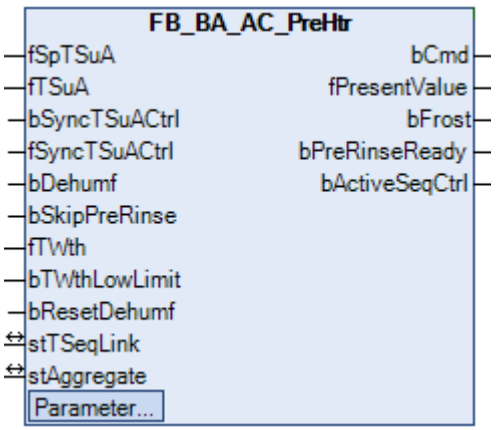
Variablen

Name	Typ	Beschreibung
srDehumf	SR	Durch das Setzen des RS-Flipflop *srDehumf* wird angezeigt, dass der Entfeuchtebetrieb innerhalb der raumlufttechnischen Anlage aktiv ist. Dem Sequenzregler *Ctrl* wird die Freigabe genommen. Das Rücksetzen des RS-Flipflop *srDehumf* erfolgt durch den Eingang *bResetDehumf*.
ftrigDeHum	F_TRIG	Durch eine fallende Flanke am Eingang *CLK* des Funktionsbausteins *ftrigDeHum* wird der Sequenzregler *Ctrl* auf den Wert 0 synchronisiert. Ausgelöst wird die fallende Flanke durch das Rücksetzen des Entfeuchtebtriebs am RS-Flipflop *srDehumf*.
rtrigSync	R_TRIG	Bei einer steigenden Flanke am Eingang *CLK* des Funktionsbausteins *rtrigSync* wird der Sequenzregler *Ctrl* auf den Wert von *fSync* synchronisiert.
PrioSwiSync	FB_BA_PrioSwi_R04 [▶ 443]	Der Prioritätenschalter priorisiert die Synchronisation des Sequenzreglers *Ctrl*

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.5.2 FB_BA_AC_PreHtr



Das Template repräsentiert die Steuerung und Regelung eines Warmwasserlufterhitzers.

Die wesentlichen Aufgaben des Templates sind:

- Regelung der Zulufttemperatur (*TSuACtrl*).
- Regelung der Rücklauftemperatur (*TRtCtrl*).
- Frostüberwachung luftseitig mit Frostschutzthermostat (*FrostThermostat*).
- Frostüberwachung luftseitig mit stetigen Frostschutzfühler (*TFrost*).
- Freigabe der Erhitzerpumpe (*Pu*).
- Ansteuerung des Erhitzerventils (*Vlv*).
- Sammeln und Auswerten der sicherheitsrelevanten Störungen durch das *PlantLock*
- Teil der Schrittkettensteuerung einer raumlufttechnischen Anlage zu sein, siehe *Aggregate*

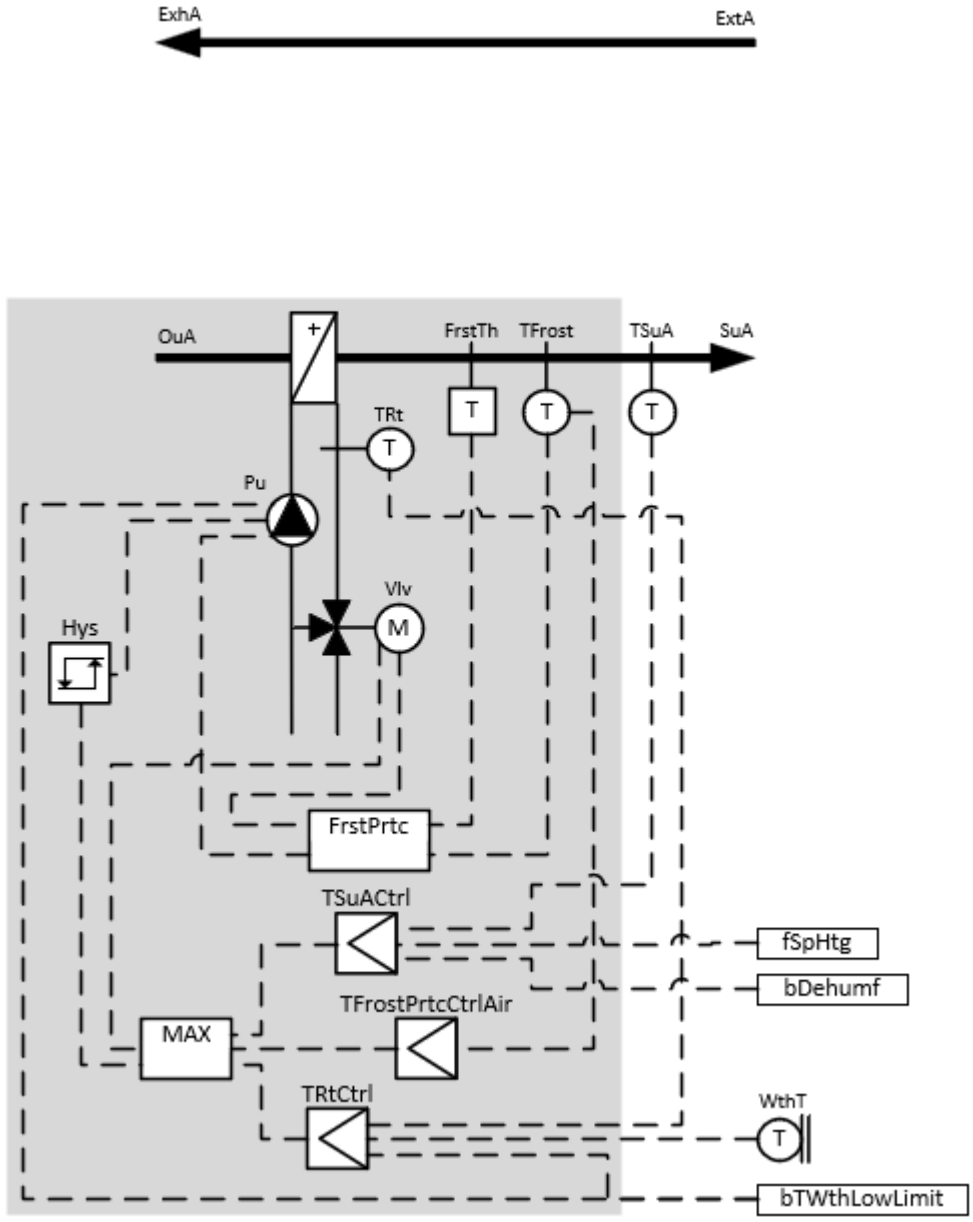
Eine Maximum-Auswahl der Stellsignale vom Temperatur-Sequenzregler *TSuACtrl* und der Frostschutzfunktionalitäten *TFrostPrtcCtrlAir* / *TRtCtrl* beugen einer Vereisung des Warmwasserlufterhitzers vor.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

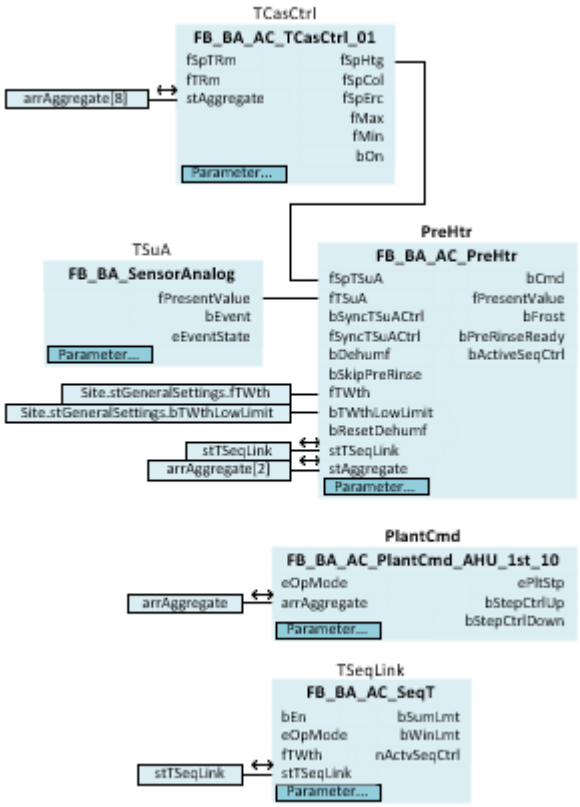
Prinzipschema 01

Das Prinzipschema zeigt die vorgesehene Anwendung des Templates mit den beteiligten Anlagenelementen.

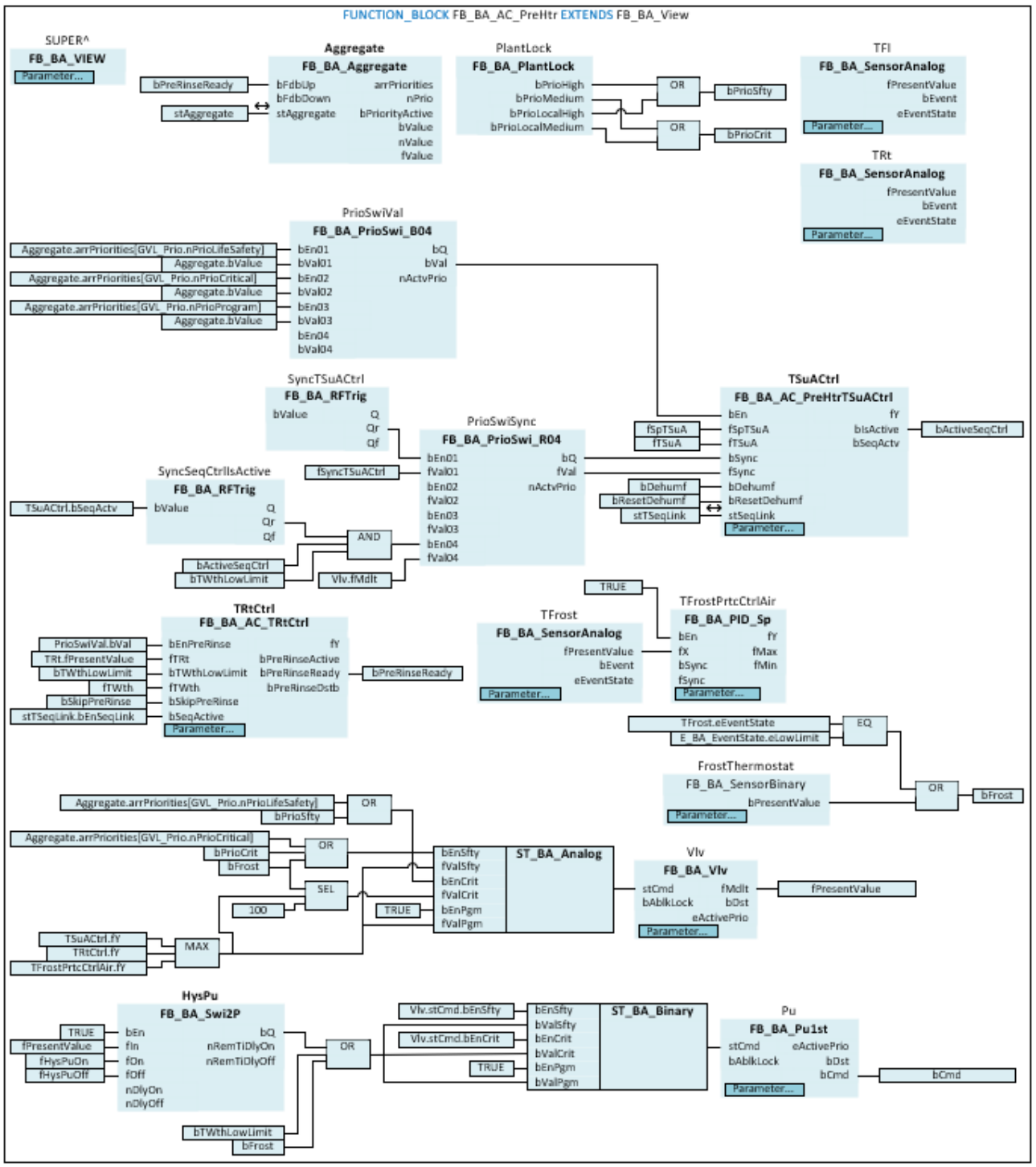


Prinzipschema 02

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.


TCasCtrl
FB_BA_AC_TCasCtrl_01
fSpTRm
fTRm
stAggregate
fSpHtg
fSpCol
fSpErr
fMax
fMin
bOn
Parameter...
arrAggregate[8]
TSuA
FB_BA_SensorAnalog
fPresentValue
bEvent
eEventState
Parameter...
PreHtr
FB_BA_AC_PreHtr
fSpTSuA
fTSuA
bSyncTSuACtrl
fSyncTSuACtrl
bDehumf
bSkipPreRinse
fTWth
bTWthLowLimit
bResetDehumf
stTSeqLink
stAggregate
bCmd
fPresentValue
bFrost
bPreRinseReady
bActiveSeqCtrl
Parameter...
Site.stGeneralSettings.fTWth
Site.stGeneralSettings.bTWthLowLimit
stTSeqLink
arrAggregate[2]
PlantCmd
FB_BA_AC_PlantCmd_AHU_1st_10
eOpMode
arrAggregate
ePltStp
bStepCtrlUp
bStepCtrlDown
Parameter...
arrAggregate
TSeqLink
FB_BA_AC_SeqT
bEn
eOpMode
fTWth
stTSeqLink
bSumLmt
bWinLmt
nActvSeqCtrl
Parameter...
stTSeqLink

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_PreHtr EXTENDS FB_BA_View
VAR_INPUT
  fSpTSuA                 : REAL;
  fTSuA                   : REAL;
  bSyncTSuACtrl           : BOOL;
  fSyncTSuACtrl           : REAL;
  bDehumf                 : BOOL;
  bSkipPreRinse           : BOOL;
  fTWth                   : REAL;
  bTWthLowLimit           : BOOL;
  bResetDehumf            : BOOL;
END_VAR
VAR_OUTPUT
  bCmd                    : BOOL;
  fPresentValue           : REAL;
```

```
  bFrost                  : BOOL;
  bPreRinseReady          : BOOL;
  bActiveSeqCtrl          : BOOL;
END_VAR
VAR_IN_OUT
  stTSeqLink              : ST_BA_SeqLink;
  stAggregate             : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOn                : REAL := 5.0;
  {attribute 'parameterCategory' := 'Behaviour'}
  fHysPuOff               : REAL := 1.0;
END_VAR
VAR_INPUT CONSTANT
  TFl                     : FB_BA_SensorAnalog;
  TRt                     : FB_BA_SensorAnalog;
  TSuACtrl                : FB_BA_AC_PreHtrTSuACtrl;
  TRtCtrl                 : FB_BA_AC_TRtCtrl;
  TFrost                  : FB_BA_SensorAnalog;
  TFrostPrtcCtrlAir       : FB_BA_PID_Sp;
  FrostThermostat         : FB_BA_SensorBinary;
  Vlv                     : FB_BA_Vlv;
  Pu                      : FB_BA_Pulst;
  PlantLock               : FB_BA_PlantLock;
  Aggregate               : FB_BA_Aggregate;
END_VAR
VAR
  bPrioSfty               : BOOL;
  bPrioCrit               : BOOL;
  PrioSwiVal              : FB_BA_PrioSwi_B04;

  HysPu                   : FB_BA_Swi2P;
  SyncTSuACtrl            : FB_BA_RFTrig;
  SyncSeqCtrlIsActive     : FB_BA_RFTrig;
  PrioSwiSync             : FB_BA_PrioSwi_R04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fSpTSuA	REAL	Sollwert der Zulufttemperatur
fTSuA	REAL	Messwert der Zulufttemperatur
bSyncTSuACtrl	BOOL	Eingang für die Synchronisation des Zuluft Sequenzreglers im Funktionsbaustein *TSuACtrl*.
fSyncTSuACtrl	REAL	Synchronisationswert für den Zuluft Sequenzregler im Funktionsbaustein *TSuACtrl*.
bDehumf	BOOL	Eingang Entfeuchtebetrieb aktiv. Dieser Zustand hat Auswirkungen auf die Sequenzsteuerung im Template Zulufttemperatur Regelung *TSuACtrl*.
bSkipPreRinse	BOOL	Eingang, um den Vorspülprozess im Funktionsbaustein *TRtCtrl* zu überspringen.
fTWth	REAL	Messwert Witterungstemperatur
bTWthLowLimit	BOOL	Die Variable zeigt an, dass der untere Grenzwert der Außentemperatur unterschritten wurde. Folgende Aktionen werden beim Unterschreiten eines kritischen Außentemperaturwertes ausgelöst: • Einschaltung Erhitzerpumpe, Pumpenzwangslauf • Freigabe der Rücklauftemperaturregelung im Funktionsbaustein *TRtCtrl* • Während des Anlagenstarts ist der Vorspülbetrieb im Funktionsbaustein *TRtCtrl* aktiv
bResetDehumf	BOOL	Eingang, um den Entfeuchtebetrieb im Template *TSuACtrl* zurückzusetzen.

Ausgänge

Name	Typ	Beschreibung
bCmd	BOOL	Aktueller Schaltzustand der einstufigen Pumpe.
fPresentValue	REAL	Aktueller Stellwert des Erhitzerventils.
bFrost	BOOL	Anzeige Frostschutz aktiv. Ausgelöst wird die Meldung entweder durch das Frostschutzthermostat *FrostThermostat* oder durch Unterschreiten des unteren Grenzwertes des stetigen Frostschutzfühlers *TFrost*. Das Signal Frostschutz aktiv löst im Template FB_BA_AC_OpMod1St_Prio [▶ 776] die Anlagenbetriebsart Frostschutz aus.
bPreRinseReady	BOOL	Anzeige Vorspülprozess hat Solltemperatur erreicht. Dieses Signal dient als Weiterschaltbedingung für die Schrittkettensteuerung einer Lüftungsanlage.
bActiveSeqCtrl	BOOL	Zeigt an, dass der Sequenzregler *TSuACtrl* der Aktive in der Sequenzregelung ist.

Ein- / Ausgänge

Name	Typ	Beschreibung
stSeqLink	ST_BA_SeqLink [▶ 125]	Die Daten- und Befehlsstruktur ist das Bindeglied zwischen dem Zulufttemperatur- Sequenzregler *Ctrl* und der Zulufttemperatur Sequenzregelung *TSeqLink* einer raumlufttechnischen Anlage.
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregatestruktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregatestruktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fHysPuOn	REAL	Oberer Schaltpunkt der Hysterese, um die Pumpe einzuschalten.
fHysPuOff	REAL	Unterer Schaltpunkt der Hysterese, um die Pumpe auszuschalten.

Eingänge CONSTANT

Name	Typ	Beschreibung
TFl	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert den Vorlauftemperaturfühler.
TRt	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert den Rücklauftemperaturfühler.
TSuACtrl	FB_BA_AC_PreHtrTSuACtrl [▶ 759]	Der Funktionsbaustein repräsentiert die Zulufttemperatur Regelung eines Vorerhitzer und ist ein Teil der Temperatur Sequenzregelung einer raumlufttechnischen Anlage. Das Stellsignal wird über eine Maximum-Auswahl an das Erhitzerventil *Vlv* weitergeleitet.
TRtCtrl	FB_BA_AC_TRtCtrl [▶ 769]	Der Funktionsbaustein repräsentiert die Steuerung und Regelung der Rücklauftemperatur eines Warmwasserlufterhitzers. Das Stellsignal wird über eine Maximum-Auswahl an das Erhitzerventil *Vlv* weitergeleitet.
TFrost	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert einen Frostschutzfühler luftseitig.
TFrostPrtcCtrlAir	FB_BA_PID_Sp [▶ 1053]	Der Funktionsbaustein repräsentiert die stetige Frostschutzüberwachung luftseitig des Vorerhitzers. Das Stellsignal wird über eine Maximum-Auswahl an das Erhitzerventil *Vlv* weitergeleitet.
FrostThermostat	FB_BA_SensorBinary [▶ 1107]	Der Funktionsbaustein repräsentiert einen Frostschutzthermostat.
Vlv	FB_BA_Vlv [▶ 1111]	Der Funktionsbaustein repräsentiert das Erhitzerventil.
Pu	FB_BA_Pu1st [▶ 1093]	Der Funktionsbaustein repräsentiert die Erhitzerpumpe.
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen des Templates die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen verursachen gezielte Schalthandlungen durch die Variablen *bPrioSfty* und *bPrioCrit* im Template. Im Folgenden sind die ereignisfähigen Objekte des Templates aufgelistet, die Störungen mit *bPrioCrit* auslösen. *TRt.MV, TFrost.MV, FrostThermostat.Input, Pu.Dst* Die Parametrierung der Lockpriorität der ereignisfähigen Objekte ist im FB_init dieses Templates zu finden.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

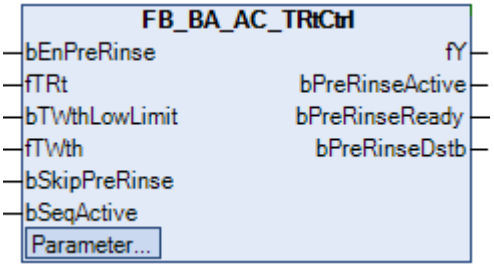
Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Sicherheit“, der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Kritisch“, der ereignisfähigen Objekte der Projektstruktur und verursacht bei Auslösung einer relevanten Störung gezielte Schalthandlungen im Template.
PrioSwiVal	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter ermittelt anhand der Kommandostruktur *stCmd* die Freigabebedingungen für die Zulufttemperatur Regelung *TSuACtrl* und für den Vorspülprozess der Rücklauftemperatur Regelung *TRtCtrl*.
HysPu	FB_BA_Swi2P	Der Zweipunktschalter schaltet die Erhitzerpumpe in Abhängigkeit der Ventilstellung *fPresentValue* und der Schaltpunkt der Hysterese *fHysPuOn*/*fHysPuOff* ein und aus.
SyncTSuACtrl	FB_BA_RFTrig	Mit einer steigenden Flanke am Eingang *bValue* des Funktionsbausteins wird die Zulufttemperatur-Regelung *TSuACtrl* auf den Wert von *fSyncTSuACtrl* synchronisiert.
SyncSeqCtrlIsActive	FB_BA_RFTrig	Die Variable *TSuACtrl.bSeqActv* zeigt an, dass die Sequenzregelung aktiv ist. Diese steigende Flanke löst am Eingang *bValue* des Funktionsbausteins *SyncSeqCtrlIsActive* einen Impuls aus. Ist der Sequenzregler des Vorerhitzers der Aktive in der Sequenzregelung *bActiveSeqCtrl* und der untere Grenzwert der Außentemperatur *bTWthLowLimit* ist aktiv, so wird mit dem Impuls von *SyncSeqCtrlIsActive* eine Synchronisation des Zulufttemperatur-Reglers *TSuACtrl* auf den Wert der Ventilstellung *Vlv.fMdlt* ausgelöst.
PrioSwiSync	FB_BA_PrioSwi_R04 [▶ 443]	Der Prioritätenschalter priorisiert die Synchronisation der Zulufttemperatur Regelung *TSuACtrl*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.2.5.3 FB_BA_AC_TRtCtrl



Das Template dient zur Steuerung und Regelung der Rücklauftemperatur eines Warmwasserlufterhitzers.

Um Frostschäden zu vermeiden, wird das Heizregister der Lüftungsanlage bei niedrigen Außentemperaturen zunächst mit warmem Wasser vorerwärmt.

Dazu wird in dem Template *FB_BA_AC_TRtCtrl* zunächst das FlipFlop *rsPreRinseActive* gesetzt. Der Rücklauftemperaturregler *Ctrl* erhält über den Selektor den Vorspülsollwert aus der Sollwertkennlinie *PreRinseSp*. Um ein Überhitzen des Heizregisters während des Vorerwärmens zu vermeiden, wird der

Vorspülsollwert durch die Kennlinie *PreRinseSp* außentemperaturabhängig variiert. Beim Erreichen der gewünschten Rücklauftemperatur schaltet das Hysterese Modul *PreRinseHys* ein. Im Anschluss daran wird das FlipFlop *rsPreRinseReady* gesetzt.

Dem Anlagenstartprogramm FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 799] wird mittels des Ausgangs *bPreRinseReady* mitgeteilt, dass es mit dem nächsten Schritt z. B. dem Öffnen der Außenluftklappe fortfahren kann.

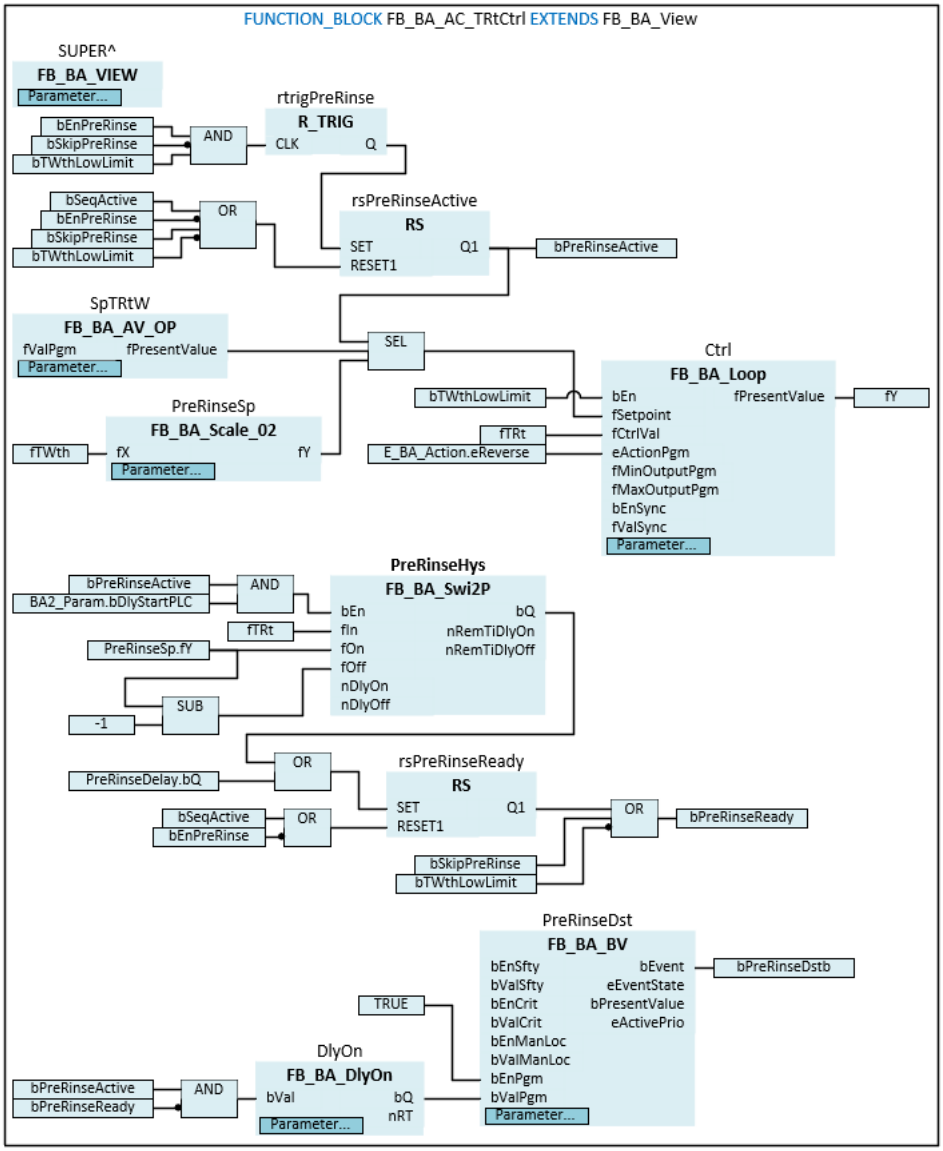
Das FlipFlop rsPreRinseActive bleibt nach wie vor gesetzt, sodass am Rücklauftemperaturregler weiterhin der Vorspülsollwert ansteht.

Wenn das Anlagenstartprogramm in seinen folgenden Schritten, die Ventilatoren eingeschaltet und die Zulufttemperaturregelung frei gegeben hat, wird das FlipFlop *rsPreRinseActive* mittels des Eingangs *bSeqActive* zurückgesetzt. Der Vorspülprozess ist damit abgeschlossen. Für den dauerhaften Frostschutzbetrieb des Heizregisters während das Anlagenbetriebs und Anlagenstillstands, erhält der Rücklauftemperaturregler den reduzierten Sollwert des AV-Objektes *SpTRtW*. Der Ausgang *bQ* des Funktionsbaustein *PreRinseDelay* wird TRUE, wenn der Vorspülprozess gestartet ist und nach Ablauf einer Verzögerungszeit der Vorspülsollwert nicht erreicht wird. Die Störung wird mittels des BV-Objektes *PreRinseDst* gemeldet.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEnPreRinse               : BOOL;
  fTRt                      : REAL;
  bTWthLowLimit             : BOOL;
  fTWth                     : REAL;
  bSkipPreRinse             : BOOL;
  bSeqActive                : BOOL;
END_VAR
VAR_OUTPUT
  fY                        : REAL;
  bPreRinseActive           : BOOL;
  bPreRinseReady            : BOOL;
  bPreRinseDstb             : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SpTRtW                    : FB_BA_AV_Op;
  PreRinseDelay             : FB_BA_DlyOn;
  PreRinseDst               : FB_BA_BV;
  PreRinseSp                : FB_BA_Scale_02;
  Ctrl                      : FB_BA_Loop;
END_VAR
VAR
  rtrigPreRinse             : R_TRIG;
  rsPreRinseReady           : RS;
  rsPreRinseActive          : RS;
  PreRinseHys               : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEnPreRinse	BOOL	Freigabe des Vorspülprozesses.
fTRt	REAL	Messwert der Rücklauftemperatur des Warmwasserlufterhitzers.
bTWthLowLimit	BOOL	Die Variable zeigt an, dass der untere Grenzwert der Außentemperatur unterschritten wurde. Dieser Zustand dient als Freigabe für den Rücklauftemperaturregler *Ctrl*.
fTWth	REAL	Aktueller Wert der Außentemperatur.
bSkipPreRinse	BOOL	Eingang zum Überspringen des Vorspülprozesses.
bSeqActive	BOOL	Mit diesem Eingang wird der Regelung mitgeteilt, dass die Zuluftemperaturregelung der Lüftung in Betrieb gegangen, bzw. der Zulufttemperatursequenzregler freigegeben wurde.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Stellgrößenausgabe des Rücklauftemperaturreglers *Ctrl*.
bPreRinseActive	BOOL	Anzeige, Vorspülprozess aktiv.
bPreRinseReady	BOOL	Anzeige, Vorspülprozess hat Solltemperatur erreicht. Dieses Signal dient als Weiterschaltbedingung für die Schrittkettensteuerung einer Lüftungsanlage, siehe Beispiel FB_BA_AC_PlantCmd_AHU_1st_10.
bPreRinseDstb	BOOL	Anzeige, Vorspülprozess fehlerhaft.

Eingänge CONSTANT

Name	Typ	Beschreibung
SpTRtW	FB_BA_AV_Op [▶ 207]	Das Analog-Wert-Objekt dient der Eingabe des Rücklauftemperatursollwertes bei Frostgefahr bzw. niedrigen Außentemperaturen.
PreRinseDelay	FB_BA_DlyOn [▶ 1073]	Das Template repräsentiert eine Einschaltverzögerung und löst die Meldung „Vorspülprozess fehlerhaft“ an dem Objekt *PreRinseDst* aus, wenn das Ergebnis der Vorspül-Hysterese *PreRinseHys* nicht erreicht wird.
PreRinseDst	FB_BA_BV [▶ 218]	Das binäre Objekt zeigt die Meldung „Vorspülprozess fehlerhaft“ an.
PreRinseSp	FB_BA_Scale_02 [▶ 1060]	Template zur Berechnung des Vorspülsollwertes in Abhängigkeit der Außentemperatur *fTWth* für die Rücklauftemperaturregelung während des Anlagenstarts.
Ctrl	FB_BA_Loop [▶ 225]	PID-Regler für die Regelung der Rücklauftemperatur des Heizregisters.

Variablen

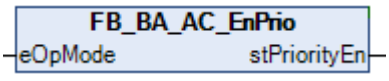
Name	Typ	Beschreibung
rtrigPreRinse	R_TRIG	rtrigPreRinse aktiviert durch eine steigende Flanke am Eingang CLK das Vorspülen am RS-Flipflop rsPreRinseActive.
rsPreRinseReady	RS	Durch das Setzen des RS-Flipflop wird angezeigt, dass der Vorspülprozess seine Solltemperatur erreicht hat.
rsPreRinseActive	RS	Durch das Setzen des RS-Flipflop wird der Vorspülprozess aktiviert.
PreRinseHys	FB_BA_Swi2P	Das Ergebnis der Vorspül-Hysterese *PreRinseHys* zeigt an, dass der Vorspülprozess seine Solltemperatur erreicht hat.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.3 General

6.1.4.2.1.1.3.1 FB_BA_AC_EnPrio



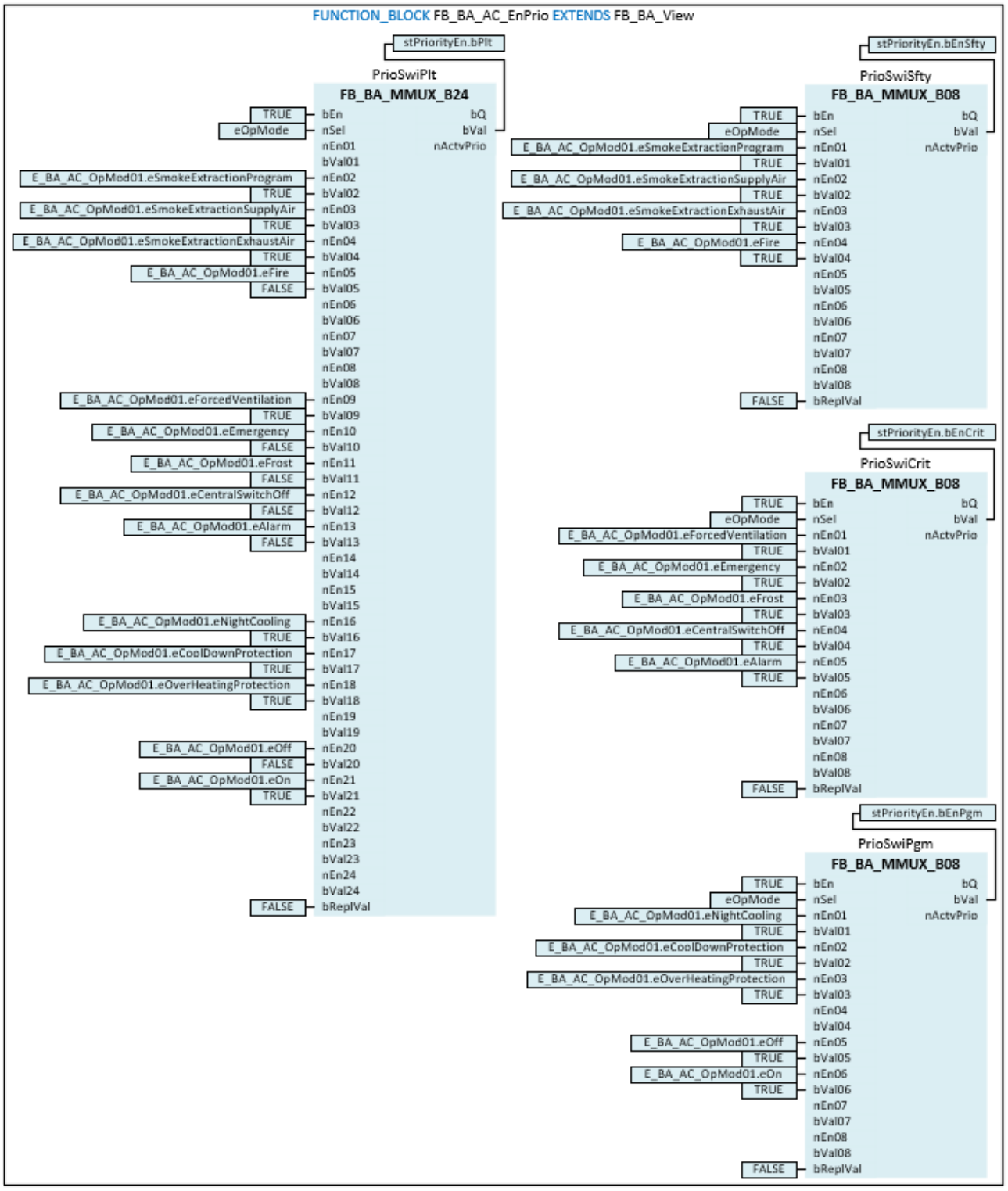
Das Template repräsentiert die Anlagenfreigabe und die Freigabe der Prioritäten Safty, Critcial und Program einer raumlufttechnischen Anlage.

Die Multiplexer *PrioSwiPlt*, *PrioSwiSfty*, *PrioSwiCrit* und *PrioSwiPgm* definieren anhand der Anlagenbetriebsart *eOpMode* die Anlagenfreigabe für die Ansteuerung der Schrittkettensteuerung einer Anlage und die Freigabe der Prioritäten „Safety“, „Critcial“ und „Program“, siehe Kommandostruktur *stPriorityEn*.

eOpmode	Wert	Betriebsart	Anlagen-status	stPriority-En.bpLT	stPriority-En.bEnSfty	stPriority-En.bEnCrit	stPriority-En.bPgm
E_BA_AC_OpMod01.eOff	1	Aus	Ausschaltung der Anlage	FALSE	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eOn	2	Ein	Einschaltung der Anlage	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eEmergency	3	Notfall	Ausschaltung der Anlage	FALSE	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eFrost	4	Frost	Ausschaltung der Anlage	FALSE	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eSmokeExtractionProgram	5	Entrauchung Programm	Einschaltung der Anlage	**TRUE**	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir	6	Entrauchung Zuluft	Einschaltung der Anlage	**TRUE**	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir	7	Entrauchung Fortluft	Einschaltung der Anlage	**TRUE**	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eFire	8	Feuer	Ausschaltung der Anlage	FALSE	**TRUE**	FALSE	FALSE
E_BA_AC_OpMod01.eNightCooling	9	Nachtkühlung	Einschaltung der Anlage	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eCoolDownProtection	10	Stützbetrieb, Auskühlschutz	Einschaltung der Anlage	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eOverHeatingProtection	11	Überhitzungsschutz	Einschaltung der Anlage	**TRUE**	FALSE	FALSE	**TRUE**
E_BA_AC_OpMod01.eAlarm	12	Störung	Ausschaltung der Anlage	FALSE	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eForcedVentilation	13	Zwangsbelüftung	Einschaltung der Anlage	**TRUE**	FALSE	**TRUE**	FALSE
E_BA_AC_OpMod01.eCentralSwitchOff	14	Zentralabschaltung	Ausschaltung der Anlage	FALSE	FALSE	**TRUE**	FALSE

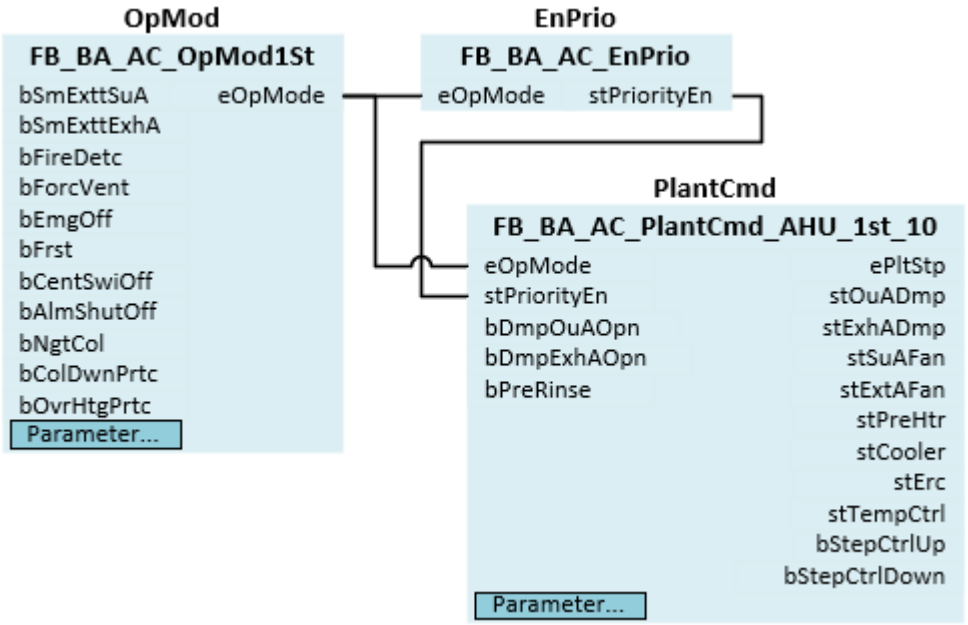
Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Prinzipschema

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Syntax

```
FUNCTION_BLOCK FB_BA_AC_EnPrio
VAR_INPUT
  eOpMode           : E_BA_AC_OpMod01;
END_VAR
VAR_OUTPUT
  stPriorityEn      : ST_BA_PriorityEn;
END_VAR
VAR
  PrioSwiPlt        : FB_BA_MMUX_B24;
  PrioSwiSfty       : FB_BA_MMUX_B08;
  PrioSwiCrit       : FB_BA_MMUX_B08;
  PrioSwiPgm        : FB_BA_MMUX_B08;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eOpMode	E_BA_AC_OpMod01 [▶ 712]	Eingang der aktuellen Betriebsart *eOpMode*.

Ausgänge

Name	Typ	Beschreibung
stPriorityEn	ST_BA_PriorityEn [▶ 283]	Ausgabe der aktuellen Kommandostruktur *stPriorityEn*. Diese beinhaltet die Anlagenfreigabe für die Schrittkettensteuerung einer raumlufttechnischen Anlage und die dazu gehörigen Freigaben der Prioritäten „Safety“, „Critcial“ und „Program“.

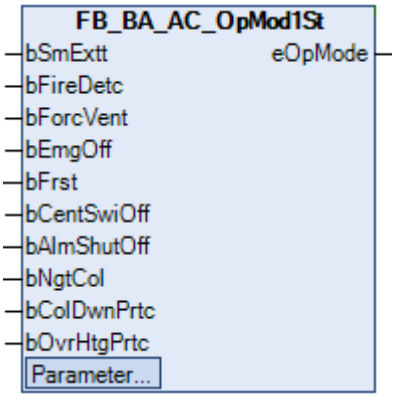
Variablen

Name	Typ	Beschreibung
PrioSwiPlt	FB_BA_MMUX_B24 [▶ 438]	Der Multiplexer definiert anhand der Anlagenbetriebsart *eOpMode* die Anlagenfreigabe für die Ansteuerung der Aggregate einer Anlage.
PrioSwiSfty	FB_BA_MMUX_B08 [▶ 438]	Der Multiplexer definiert anhand der Anlagenbetriebsart *eOpMode* die Freigabe der Priorität „Safety“ für die Ansteuerung der Aggregate einer Anlage.
PrioSwiCrit	FB_BA_MMUX_B08 [▶ 438]	Der Multiplexer definiert anhand der Anlagenbetriebsart *eOpMode* die Freigabe der Priorität „Critical“ für die Ansteuerung der Aggregate einer Anlage.
PrioSwiPgm	FB_BA_MMUX_B08 [▶ 438]	Der Multiplexer definiert anhand der Anlagenbetriebsart *eOpMode* die Freigabe der Priorität „Program“ für die Ansteuerung der Aggregate einer Anlage.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.3.2 FB_BA_AC_OpMod1St

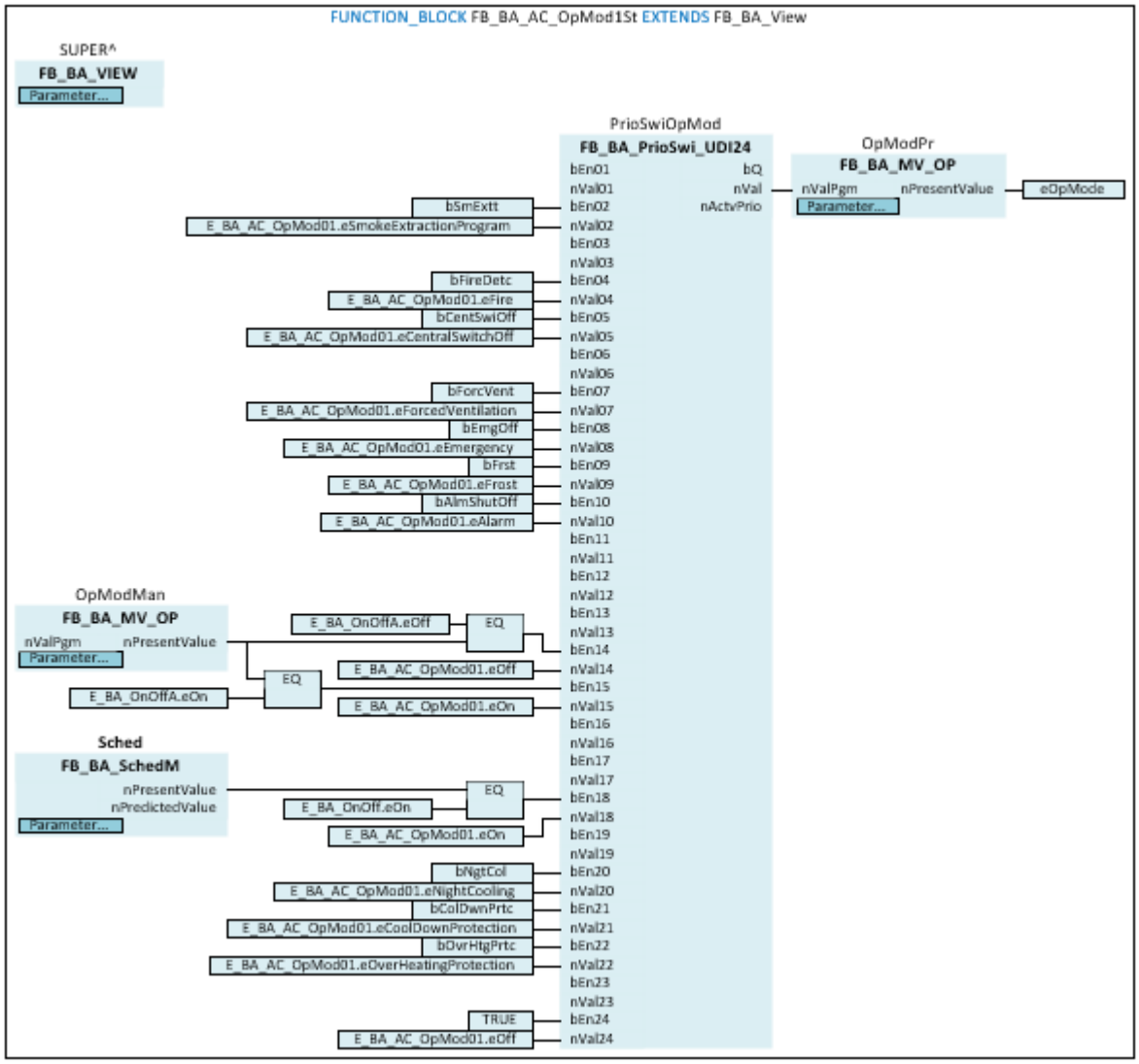


Das Template repräsentiert die Betriebsart einer raumlufttechnischen Anlage.

Am Prioritätenschalter *PrioSwiOpMod* werden verschiedene Ereignisse oder Befehle wie zum Beispiel Brandalarm, Anforderung vom Zeitschaltplan oder die Anforderung vom Anlagenwahlschalter priorisiert und eine resultierende Betriebsart bzw. ein resultierender Anlagenstatus auf die Variable eOpMode [▶ 712] geschrieben.

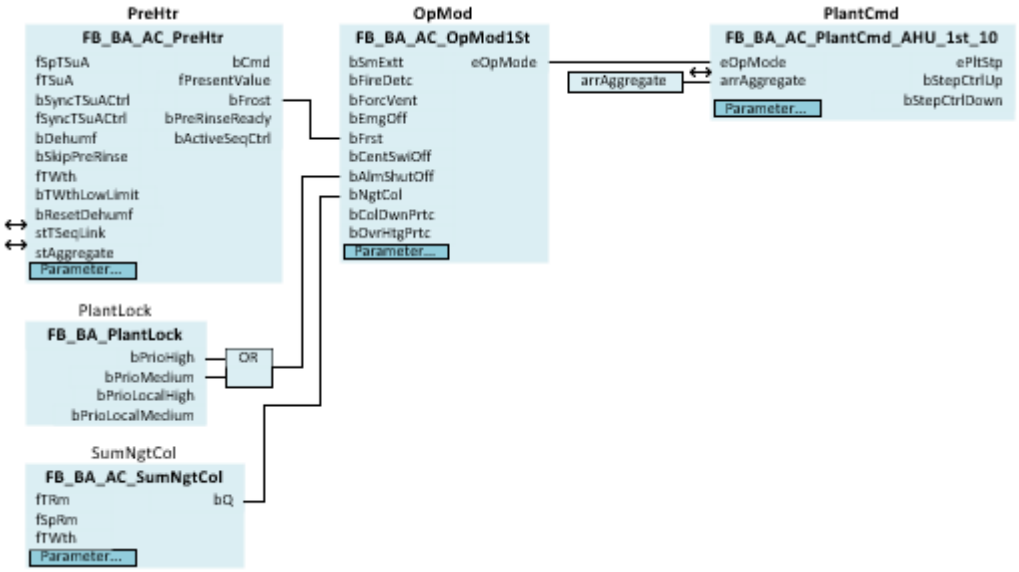
Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Prinzipschema

Das Schema zeigt die Einbindung des Templates innerhalb einer Anlage.



Syntax

```
FUNCTION_BLOCK FB_BA_AC_OpMod1St EXTENDS FB_BA_View
VAR_INPUT
  bSmExtt                   : BOOL;
  bFireDetc                 : BOOL;
  bForcVent                 : BOOL;
  bEmgOff                   : BOOL;
  bFrst                     : BOOL;
  bCentSwiOff               : BOOL;
  bAlmShutOff               : BOOL;
  bNgtCol                   : BOOL;
  bColDwnPrtc               : BOOL;
  bOvrHtgPrtc               : BOOL;
END_VAR
VAR_OUTPUT
  eOpMode                   : E_BA_AC_OpMod01;
END_VAR
VAR_INPUT CONSTANT
  OpModMan                  : FB_BA_MV_Op;
  Sched                     : FB_BA_SchedM;
  OpModPr                   : FB_BA_MV_Op;
END_VAR
VAR
  PrioSwiOpMod              : FB_BA_PrioSwi_UDI24;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bSmExtt	BOOL	Entrauchung der Anlage angefordert
bFireDetc	BOOL	Meldung Feueralarm von Brandmeldezentrale
bForceVent	BOOL	Anforderung Zwangsbelüftung
bEmgOff	BOOL	Not-Aus
bFrst	BOOL	Frostschutzprogramm aktiv
bCentSwiOff	BOOL	Zentralabschaltung
bAlmShutOff	BOOL	Sammelstörmeldung – Anlage abschalten
bNgtCol	BOOL	Anforderung vom Programm Sommernachtkühlung (siehe FB_BA_AC_SumNgtCol [▶ 796])
bColDwnPrtc	BOOL	Anforderung Programm Auskühlschutz
bOvrHtgPrtc	BOOL	Anforderung Programm Überhitzungsschutz

Ausgänge

Name	Typ	Beschreibung
eOpMode	E_BA_AC_OpMod01 [▶ 712]	Ausgabe der aktuellen Betriebsart eOpMode

Eingänge CONSTANT

Name	Typ	Beschreibung
OpModMan	FB_BA_MV_Op [▶ 244]	Das Multistate-Value-Objekt repräsentiert einen Betriebsartenschalter mit den Betriebsarten Auto, Hand-Aus und Hand-Ein.
Sched	FB_BA_SchedM [▶ 230]	Zeitschaltobjekt (Automatik) für das Gebäudeenergieniveau „BuildingEnergyLevel“.
OpModPr	FB_BA_MV_Op [▶ 244]	Das Multistate-Value-Objekt zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.

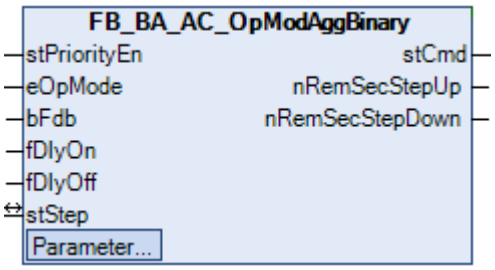
Variablen

Name	Typ	Beschreibung
PrioSwiOpMod	FB_BA_PrioSwi_UDI24 [▶ 443]	Der Prioritätenschalter ermittelt die aktuelle Betriebsart *eOpMode* aus den anstehenden Ereignissen oder Befehlen der Anlage.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.3.3 FB_BA_AC_OpModAggBinary



Das Template dient als Bindeglied von Schritten innerhalb der Schrittkettensteuerung einer raumlufttechnischen Anlage.

Der Funktionsbaustein *Step* ist das Kernstück des Templates und repräsentiert den binären Empfangsbaustein einer Schrittkettensteuerung.

Über die Kommandostruktur *stCmd* erhält das angeschlossene Aggregat seine Befehle.

Der Datenaustausch zum Steuerbaustein der Schrittkettensteuerung (*StepCtrlAgg*) geschieht über die Daten- und Kommandostruktur *stStep*.

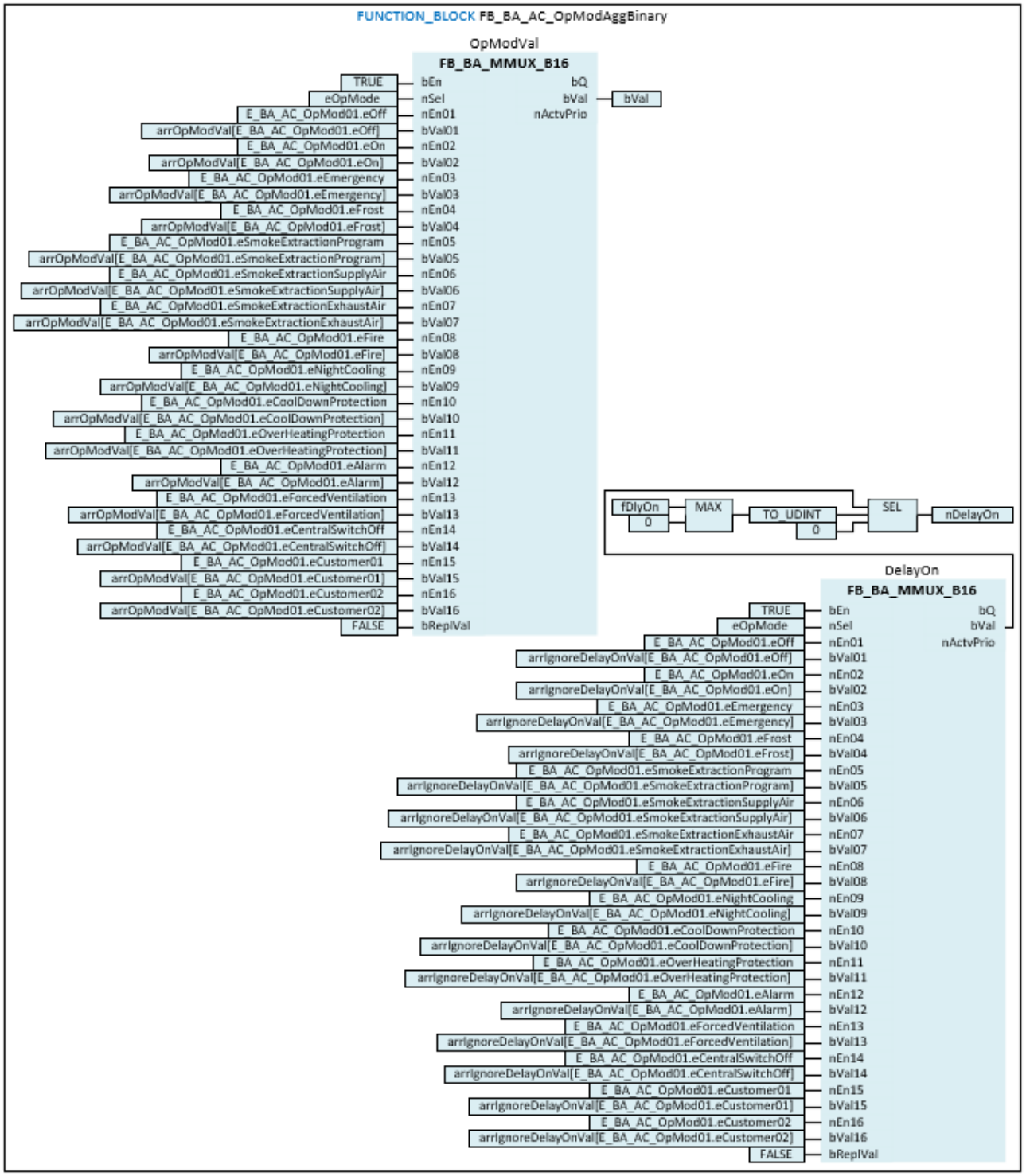
Die vier Multiplexer *OpModVal*, *DelayOn*, *DelayOff* und *IgnoreFdb* definieren anhand der Anlagenbetriebsart *eOpMode* die Ein- und Ausschaltbedingungen des Anlagenschrittes. An jedem dieser Multiplexer liegt ein zu parametrierendes Array für den Ausgabewert der Multiplexer an (*arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal*).

Die resultierenden Ergebnisse der Multiplexer *bVal*, *_bFdb*, *nDelayOn* und *nDelayOff* werden an den Empfangsbaustein der Schrittkettensteuerung *Step* übertragen, siehe Blockschaltbild Teil 02.

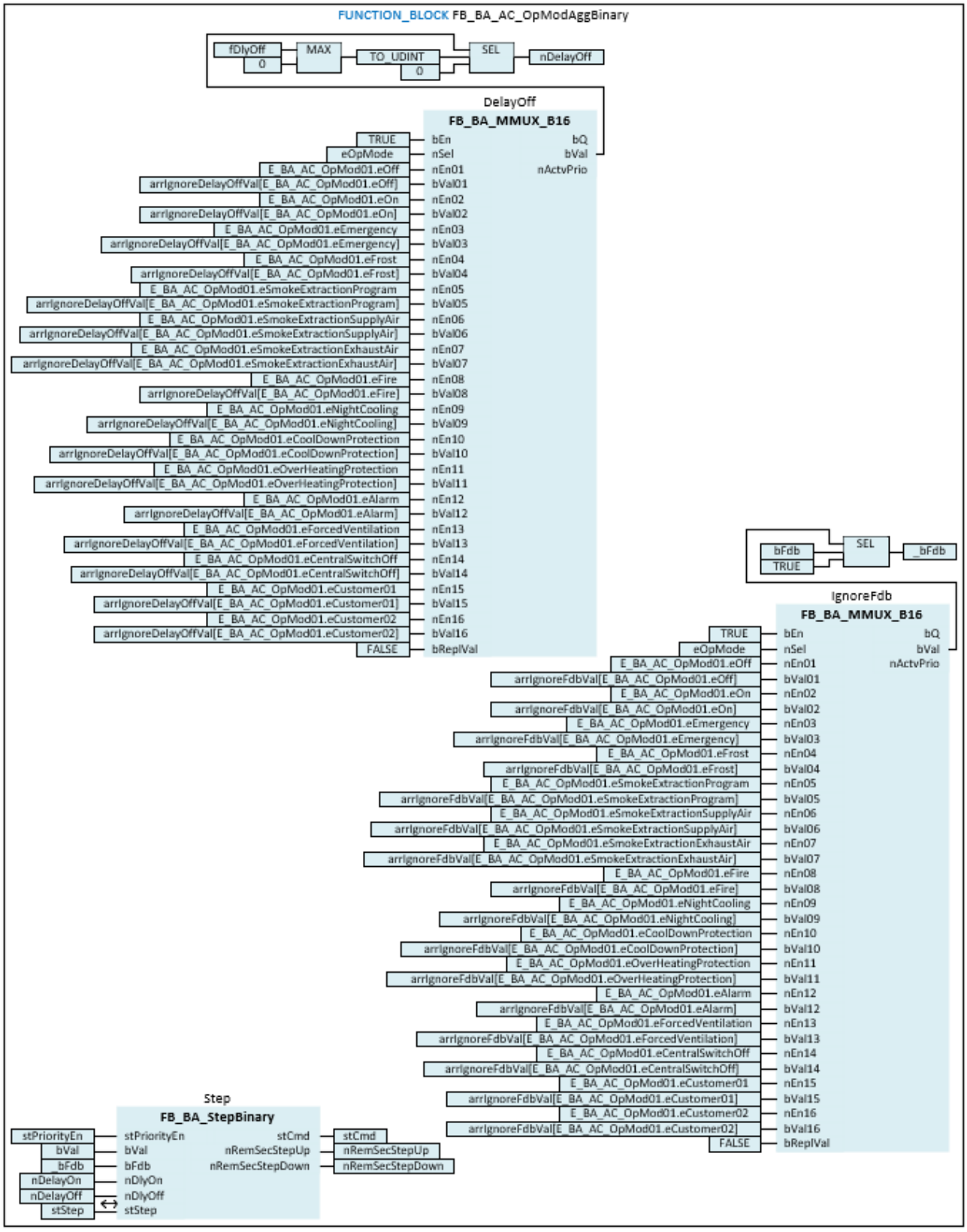
Die Parametrierung der Arrays sind in einer raumlufttechnischen Anlage in den Methoden unterhalb des Beispiel-Templates FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 799] zu finden, siehe Methoden.

Blockschaltbild

Teil 01


FUNCTION_BLOCK FB_BA_AC_OpModAggBinary
OpModVal
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrOpModVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrOpModVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrOpModVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrOpModVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrOpModVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrOpModVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrOpModVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrOpModVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrOpModVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrOpModVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrOpModVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrOpModVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrOpModVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio
bVal
fDlyOn
0
MAX
TO_UDINT
0
SEL
nDelayOn
DelayOn
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio

Teil 02



Syntax

```
FUNCTION_BLOCK FB_BA_AC_OpModAggBinary
VAR_INPUT
  stPriorityEn              : ST_BA_PriorityEn;
  eOpMode                   : E_BA_AC_OpMod01;
  bFdb                      : BOOL;
  fDlyOn                    : REAL;
  fDlyOff                   : REAL;
END_VAR
VAR_OUTPUT
```

```
  stCmd                    : ST_BA_Binary;
  nRemSecStepUp            : UDINT;
  nRemSecStepDown          : UDINT;
END_VAR
VAR_IN_OUT
  stStep                   : ST_BA_Step;
END_VAR
VAR_INPUT CONSTANT
  OpModVal                 : FB_BA_MMUX_B16;
  DelayOn                  : FB_BA_MMUX_B16;
  DelayOff                 : FB_BA_MMUX_B16;
  IgnoreFdb                : FB_BA_MMUX_B16;
  Step                     : FB_BA_StepBinary;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bInit                    : BOOL := TRUE;
  arrOpModVal              : ARRAY [1..16] OF BOOL;
  arrIgnoreDelayOnVal      : ARRAY [1..16] OF BOOL;
  arrIgnoreDelayOffVal     : ARRAY [1..16] OF BOOL;
  arrIgnoreFdbVal          : ARRAY [1..16] OF BOOL;
END_VAR
VAR
  bVal                     : BOOL;
  _bFdb                    : BOOL;
  nDelayOn                 : UDINT;
  nDelayOff                : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stPriorityEn	ST_BA_PriorityEn [▶ 283]	Eingang der aktuellen Kommandostruktur stPriorityEn. Diese beinhaltet die Anlagenfreigabe für die Schrittkettensteuerung einer raumlufttechnischen Anlage und die dazu gehörigen Freigaben der Prioritäten „Safety“, „Critcial“ und „Program“.
eOpMode	E_BA_AC_OpMod01 [▶ 712]	Eingang der aktuellen Betriebsart.
bFdb	BOOL	Rückmeldung des angeschlossenen Aggregats der Schrittkettensteuerung. Die Rückmeldung wird zur Weiterschaltung in den nächsten Schritt benötigt. *bFdb* wird nur berücksichtigt, wenn das Aggregat das Aktive in der Schrittkettensteuerung ist. Durch Parametrierung der einzelnen Elemente des Array *arrIgnoreFdbVal* mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Rückmeldung des Aggregates ignoriert werden.
fDlyOn	REAL	Zeitangabe der Einschaltverzögerung [s]. Die Zeitangabe wird zur Weiterschaltung in den nächsten Schritt benötigt. Durch Parametrierung der einzelnen Elemente des Array *arrIgnoreDelayOnVal* mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Einschaltverzögerung des Schrittes ignoriert werden.
fDlyOff	REAL	Zeitangabe der Ausschaltverzögerung [s]. Die Zeitangabe wird zur Ausschaltung des aktiven Schrittes benötigt. Durch Parametrierung der einzelnen Elemente des Array *arrIgnoreDelayOffVal* mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Ausschaltverzögerung des Schrittes ignoriert werden.

Ausgänge

Name	Typ	Beschreibung
stCmd	ST_BA_Binary [▶ 282]	Die Kommandostruktur *stCmd* übermittelt dem angeschlossenen Aggregat die Freigaben und Schaltwerte der Prioritäten.
nRemSecStepUp	UDINT	Countdown zur Einschaltung des nächsten Schrittes [s].
nRemSecStepDown	UDINT	Countdown zur Ausschaltung des aktiven Schrittes [s].

Ein- / Ausgänge

Name	Typ	Beschreibung
stStep	ST_BA_Step [▶ 282]	Daten- und Befehlsstruktur zwischen dem Schrittkettenbaustein *FB_BA_StepBinary* und dem Steuerbaustein der Schrittkette FB_BA_StepCtrlAgg16 [▶ 482].

Eingänge CONSTANT

Name	Typ	Beschreibung
OpModVal	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer ermittelt den Einschaltwert des angebundenen Aggregates. Das resultierende Ergebnis wird an den Ausgang *bVal* übermittelt.
DelayOn	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer und das angebundene Netzwerk von Funktionen ermitteln die Weiterschaltverzögerung in den nächsten Schritt. Das resultierende Ergebnis wird an den Ausgang *nDelayOn* übermittelt.
DelayOff	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer und das angebundene Netzwerk von Funktionen ermitteln die Ausschaltverzögerung des aktuellen Schrittes. Das resultierende Ergebnis wird an den Ausgang *nDelayOff* übermittelt.
IgnoreFdb	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer und das angebundene Netzwerk von Funktionen ermitteln die Rückmeldung des Aggregates. Das resultierende Ergebnis wird an den Ausgang *_bFdb* übermittelt.
Step	FB_BA_StepBinary [▶ 484]	Der Funktionsbaustein repräsentiert den binären Empfangsbaustein einer Schrittkettensteuerung und steuert über die Kommandostruktur *stCmd* das angeschlossene Aggregat an. Der Datenaustausch zum Steuerbaustein der Schrittkettensteuerung (FB_BA_StepCtrlAgg16 [▶ 482]) geschieht über die Daten- und Kommandostruktur *stStep*.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bInit	BOOL	Die Variable dient der Initialisierung der Arrays *arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal* in den Methoden unterhalb des Templates FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 799]. Nach dem Starten der Steuerung wird die Variable nach einem SPS-Zyklus auf FALSE gesetzt und dieser Zustand persistent gespeichert.
arrOpModVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays kann für die unterschiedlichen Betriebsarten *eOpMode* der Einschaltwert des angebundenen Aggregates am Schrittkettenbaustein definiert werden.
arrIgnoreDelayOnVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Weiterschaltverzögerung in den nächsten Schritt innerhalb der Schrittkettensteuerung ignoriert werden.
arrIgnoreDelayOffVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Ausschaltverzögerung des aktuellen Schrittes innerhalb der Schrittkettensteuerung ignoriert werden.
arrIgnoreFdbVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Rückmeldung des Aggregates *bFdb* ignoriert werden.

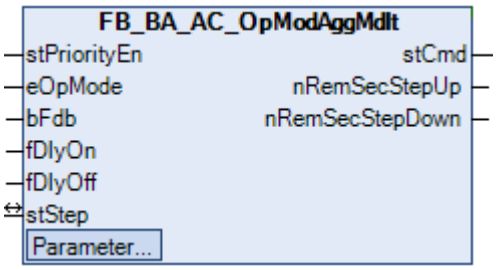
Variablen

Name	Typ	Beschreibung
bVal	BOOL	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *OpModVal* und des Arrays *arrOpModVal*. *bVal* definiert den Einschaltwert am Schrittkettenbaustein *Step*.
_bFdb	BOOL	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *IgnoreFdb* und des Arrays *arrIgnoreFdbVal*. *_bFdb* definiert die Weiterschaltbedingung in den nächsten Schritt am Schrittkettenbaustein *Step*.
nDelayOn	UDINT	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *DelayOn* und des Arrays *arrIgnoreDelayOnVal*. *nDelayOn* definiert die Weiterschaltverzögerung in den nächsten Schritt am Schrittkettenbaustein *Step*.
nDelayOff	UDINT	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *DelayOff* und des Arrays *arrIgnoreDelayOffVal*. *nDelayOff* definiert die Ausschaltverzögerung des aktuellen Schrittes am Schrittkettenbaustein *Step*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.3.4 FB_BA_AC_OpModAggMdlt



Das Template dient als Bindeglied zwischen den Schritten innerhalb der Schrittkettensteuerung einer raumlufttechnischen Anlage.

Der Funktionsbaustein *Step* ist das Kernstück des Templates und repräsentiert den binären Empfangsbaustein einer Schrittkettensteuerung.

Über die Kommandostruktur *stCmd* erhält das angeschlossene Aggregat seine Befehle.

Der Datenaustausch zum Steuerbaustein der Schrittkettensteuerung (*StepCtrlAgg*) geschieht über die Daten- und Kommandostruktur *stStep*.

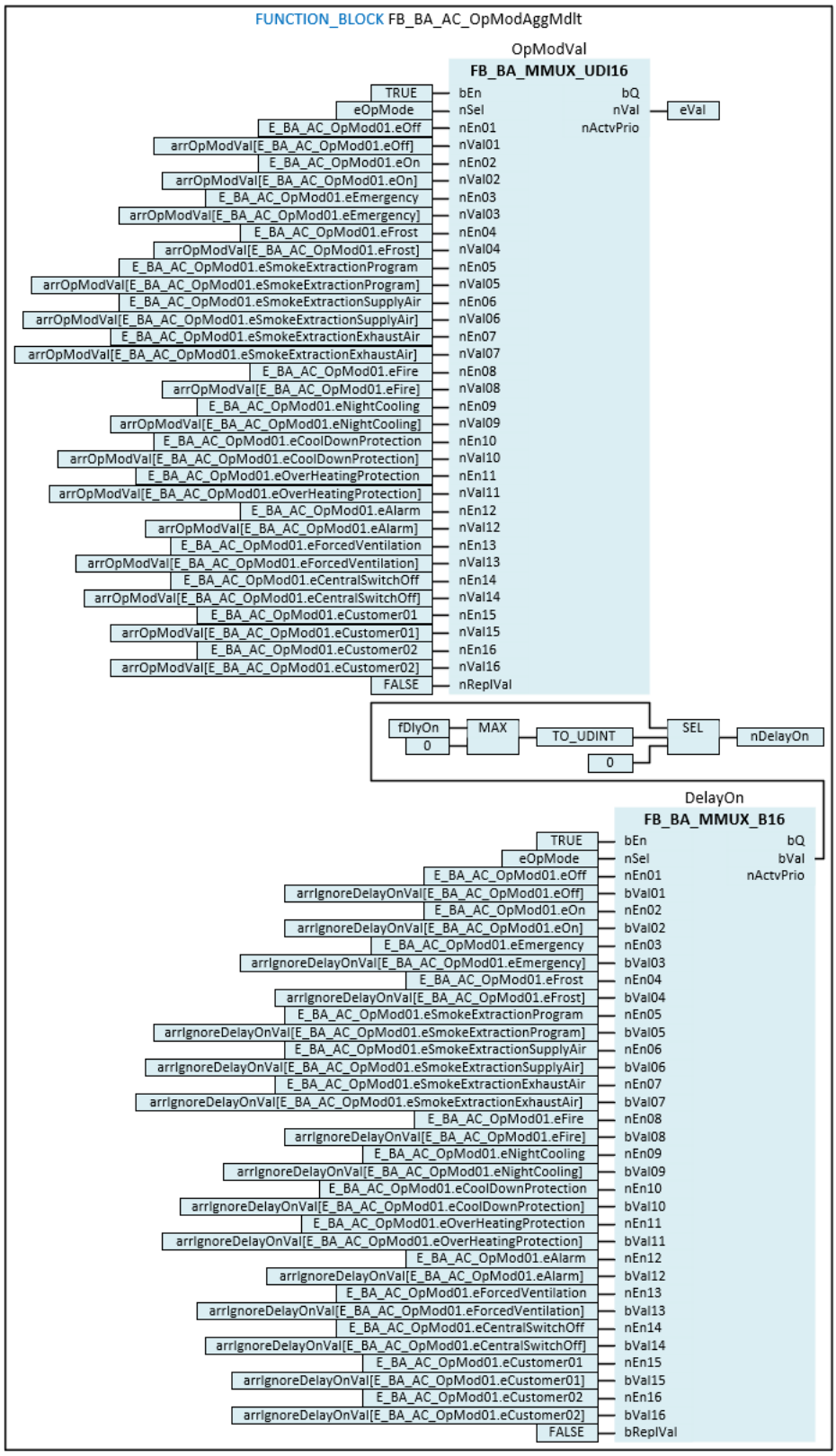
Die 4 Multiplexer *OpModVal*, *DelayOn*, *DelayOff* und *IgnoreFdb* definieren anhand der Anlagenbetriebsart *eOpMode* die Ein- und Ausschaltbedingungen des Anlagenschrittes. An jedem dieser Multiplexer liegt ein zu parametrierendes Array für den Ausgabewert der Multiplexer an (*arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal*).

Die resultierenden Ergebnisse der Multiplexer *eVal*, *_bFdb*, *nDelayOn* und *nDelayOff* werden an den Empfangsbaustein der Schrittkettensteuerung *Step* übertragen, siehe Blockschaltbild Teil 02.

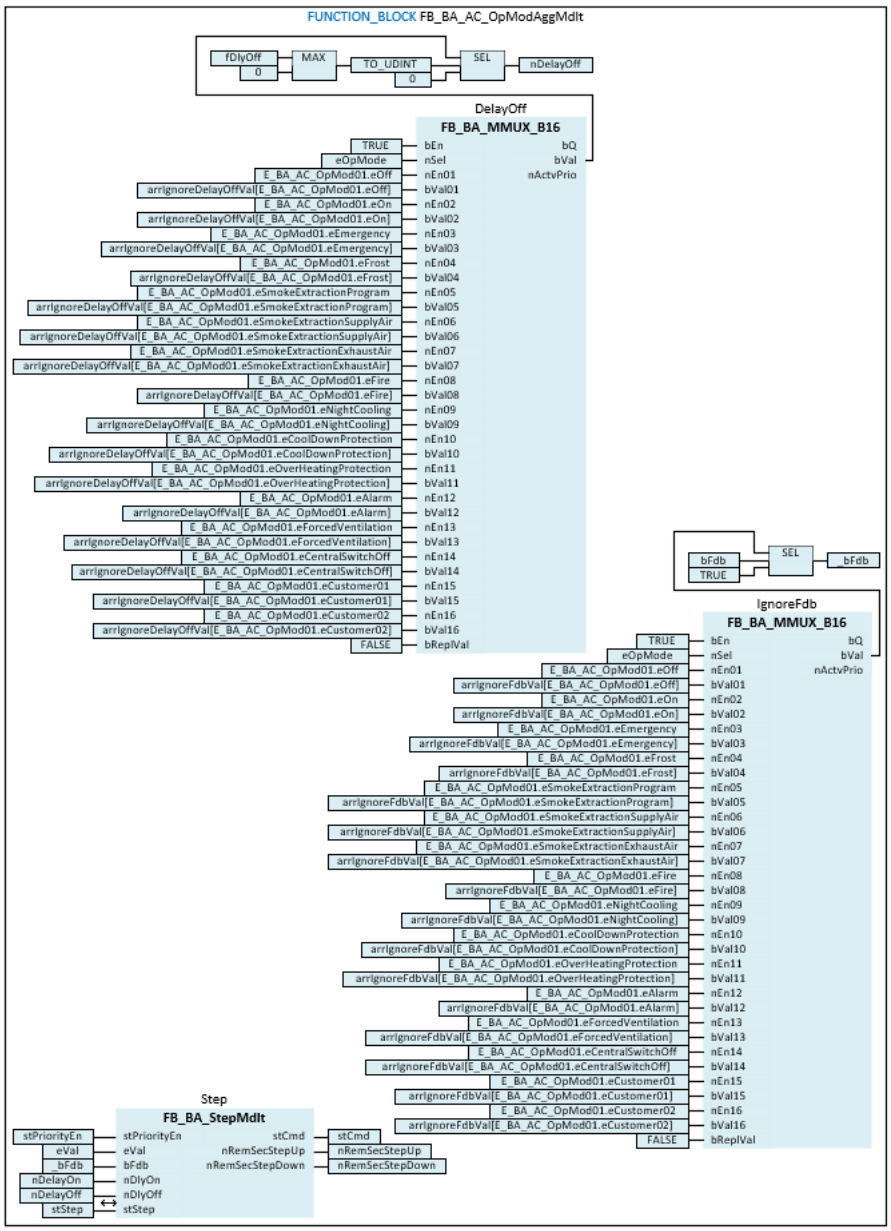
Die Parametrierung der Arrays sind in einer raumlufttechnischen Anlage in den Methoden unterhalb des Beispiel-Templates FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 799] zu finden, siehe Methoden.

Blockschaltbild

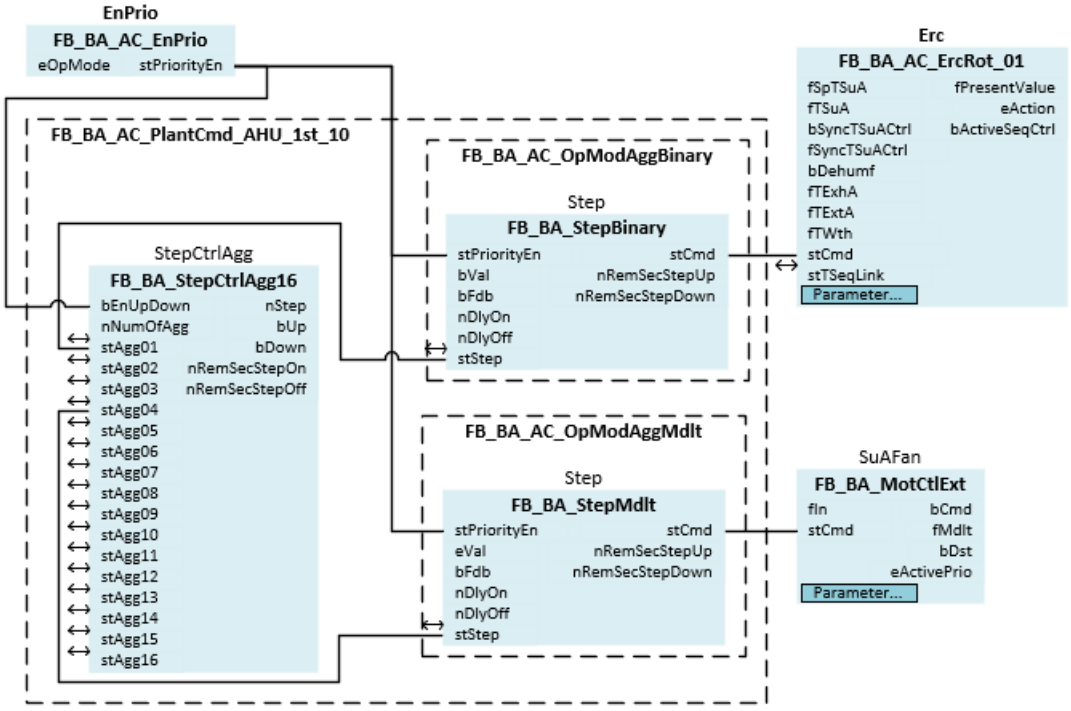
Teil 01


FUNCTION_BLOCK FB_BA_AC_OpModAggMdlt
OpModVal
FB_BA_MMUX_UDI16
TRUE bEn
eOpMode nSel
E_BA_AC_OpMod01.eOff nEn01
arrOpModVal[E_BA_AC_OpMod01.eOff] nVal01
E_BA_AC_OpMod01.eOn nEn02
arrOpModVal[E_BA_AC_OpMod01.eOn] nVal02
E_BA_AC_OpMod01.eEmergency nEn03
arrOpModVal[E_BA_AC_OpMod01.eEmergency] nVal03
E_BA_AC_OpMod01.eFrost nEn04
arrOpModVal[E_BA_AC_OpMod01.eFrost] nVal04
E_BA_AC_OpMod01.eSmokeExtractionProgram nEn05
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionProgram] nVal05
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir nEn06
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir] nVal06
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir nEn07
arrOpModVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir] nVal07
E_BA_AC_OpMod01.eFire nEn08
arrOpModVal[E_BA_AC_OpMod01.eFire] nVal08
E_BA_AC_OpMod01.eNightCooling nEn09
arrOpModVal[E_BA_AC_OpMod01.eNightCooling] nVal09
E_BA_AC_OpMod01.eCoolDownProtection nEn10
arrOpModVal[E_BA_AC_OpMod01.eCoolDownProtection] nVal10
E_BA_AC_OpMod01.eOverHeatingProtection nEn11
arrOpModVal[E_BA_AC_OpMod01.eOverHeatingProtection] nVal11
E_BA_AC_OpMod01.eAlarm nEn12
arrOpModVal[E_BA_AC_OpMod01.eAlarm] nVal12
E_BA_AC_OpMod01.eForcedVentilation nEn13
arrOpModVal[E_BA_AC_OpMod01.eForcedVentilation] nVal13
E_BA_AC_OpMod01.eCentralSwitchOff nEn14
arrOpModVal[E_BA_AC_OpMod01.eCentralSwitchOff] nVal14
E_BA_AC_OpMod01.eCustomer01 nEn15
arrOpModVal[E_BA_AC_OpMod01.eCustomer01] nVal15
E_BA_AC_OpMod01.eCustomer02 nEn16
arrOpModVal[E_BA_AC_OpMod01.eCustomer02] nVal16
FALSE nReplVal
bQ
nVal
nActvPrio
eVal
fDlyOn
0
MAX
TO_UDINT
SEL
nDelayOn
0
DelayOn
FB_BA_MMUX_B16
TRUE bEn
eOpMode nSel
E_BA_AC_OpMod01.eOff nEn01
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOff] bVal01
E_BA_AC_OpMod01.eOn nEn02
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOn] bVal02
E_BA_AC_OpMod01.eEmergency nEn03
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eEmergency] bVal03
E_BA_AC_OpMod01.eFrost nEn04
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFrost] bVal04
E_BA_AC_OpMod01.eSmokeExtractionProgram nEn05
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionProgram] bVal05
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir nEn06
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir] bVal06
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir nEn07
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir] bVal07
E_BA_AC_OpMod01.eFire nEn08
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eFire] bVal08
E_BA_AC_OpMod01.eNightCooling nEn09
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eNightCooling] bVal09
E_BA_AC_OpMod01.eCoolDownProtection nEn10
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCoolDownProtection] bVal10
E_BA_AC_OpMod01.eOverHeatingProtection nEn11
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eOverHeatingProtection] bVal11
E_BA_AC_OpMod01.eAlarm nEn12
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eAlarm] bVal12
E_BA_AC_OpMod01.eForcedVentilation nEn13
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eForcedVentilation] bVal13
E_BA_AC_OpMod01.eCentralSwitchOff nEn14
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCentralSwitchOff] bVal14
E_BA_AC_OpMod01.eCustomer01 nEn15
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer01] bVal15
E_BA_AC_OpMod01.eCustomer02 nEn16
arrIgnoreDelayOnVal[E_BA_AC_OpMod01.eCustomer02] bVal16
FALSE bReplVal
bQ
bVal
nActvPrio

Teil 02


FUNCTION_BLOCK FB_BA_AC_OpModAggMdlt
fDlyOff
0
MAX
TO_UDINT
0
SEL
nDelayOff
DelayOff
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrIgnoreDelayOffVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio
bFdb
TRUE
SEL
_bFdb
IgnoreFdb
FB_BA_MMUX_B16
TRUE
eOpMode
E_BA_AC_OpMod01.eOff
arrIgnoreFdbVal[E_BA_AC_OpMod01.eOff]
E_BA_AC_OpMod01.eOn
arrIgnoreFdbVal[E_BA_AC_OpMod01.eOn]
E_BA_AC_OpMod01.eEmergency
arrIgnoreFdbVal[E_BA_AC_OpMod01.eEmergency]
E_BA_AC_OpMod01.eFrost
arrIgnoreFdbVal[E_BA_AC_OpMod01.eFrost]
E_BA_AC_OpMod01.eSmokeExtractionProgram
arrIgnoreFdbVal[E_BA_AC_OpMod01.eSmokeExtractionProgram]
E_BA_AC_OpMod01.eSmokeExtractionSupplyAir
arrIgnoreFdbVal[E_BA_AC_OpMod01.eSmokeExtractionSupplyAir]
E_BA_AC_OpMod01.eSmokeExtractionExhaustAir
arrIgnoreFdbVal[E_BA_AC_OpMod01.eSmokeExtractionExhaustAir]
E_BA_AC_OpMod01.eFire
arrIgnoreFdbVal[E_BA_AC_OpMod01.eFire]
E_BA_AC_OpMod01.eNightCooling
arrIgnoreFdbVal[E_BA_AC_OpMod01.eNightCooling]
E_BA_AC_OpMod01.eCoolDownProtection
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCoolDownProtection]
E_BA_AC_OpMod01.eOverHeatingProtection
arrIgnoreFdbVal[E_BA_AC_OpMod01.eOverHeatingProtection]
E_BA_AC_OpMod01.eAlarm
arrIgnoreFdbVal[E_BA_AC_OpMod01.eAlarm]
E_BA_AC_OpMod01.eForcedVentilation
arrIgnoreFdbVal[E_BA_AC_OpMod01.eForcedVentilation]
E_BA_AC_OpMod01.eCentralSwitchOff
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCentralSwitchOff]
E_BA_AC_OpMod01.eCustomer01
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCustomer01]
E_BA_AC_OpMod01.eCustomer02
arrIgnoreFdbVal[E_BA_AC_OpMod01.eCustomer02]
FALSE
bEn
nSel
nEn01
bVal01
nEn02
bVal02
nEn03
bVal03
nEn04
bVal04
nEn05
bVal05
nEn06
bVal06
nEn07
bVal07
nEn08
bVal08
nEn09
bVal09
nEn10
bVal10
nEn11
bVal11
nEn12
bVal12
nEn13
bVal13
nEn14
bVal14
nEn15
bVal15
nEn16
bVal16
bReplVal
bQ
bVal
nActvPrio
Step
FB_BA_StepMdlt
stPriorityEn
eVal
_bFdb
nDelayOn
nDelayOff
stStep
stPriorityEn
eVal
bFdb
nDlyOn
nDlyOff
stStep
stCmd
nRemSecStepUp
nRemSecStepDown
stCmd
nRemSecStepUp
nRemSecStepDown

Prinzipschema



Syntax

```
FUNCTION_BLOCK FB_BA_AC_OpModAggMdlt
VAR_INPUT
  stPriorityEn              : ST_BA_PriorityEn;
  eOpMode                   : E_BA_AC_OpMod01;
  bFdb                      : BOOL;
  fDlyOn                    : REAL;
  fDlyOff                   : REAL;
END_VAR
VAR_OUTPUT
  stCmd                     : ST_BA_Mdlt;
  nRemSecStepUp             : UDINT;
  nRemSecStepDown           : UDINT;
END_VAR
VAR_IN_OUT
  stStep                    : ST_BA_Step;
END_VAR
VAR_INPUT CONSTANT
  OpModVal                  : FB_BA_MMUX_UDI16;
  DelayOn                   : FB_BA_MMUX_B16;
  DelayOff                  : FB_BA_MMUX_B16;
  IgnoreFdb                 : FB_BA_MMUX_B16;
  Step                      : FB_BA_StepMdlt;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bInit                     : BOOL := TRUE;
  arrOpModVal               : ARRAY [1..16] OF E_BA_Mdlt;
  arrIgnoreDelayOnVal       : ARRAY [1..16] OF BOOL;
  arrIgnoreDelayOffVal      : ARRAY [1..16] OF BOOL;
  arrIgnoreFdbVal           : ARRAY [1..16] OF BOOL;
END_VAR
VAR
  eVal                      : E_BA_Mdlt;
  _bFdb                     : BOOL;
  nDelayOn                  : UDINT;
  nDelayOff                 : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stPriorityEn	ST_BA_PriorityEn [▶ 283]	Eingang der aktuellen Kommandostruktur stPriorityEn. Diese beinhaltet die Anlagenfreigabe für die Schrittkettensteuerung einer raumlufttechnischen Anlage und die dazu gehörigen Freigaben der Prioritäten „Safety“, „Critcial“ und „Program“.
eOpMode	E_BA_AC_OpMod01 [▶ 712]	Eingang der aktuellen Betriebsart.
bFdb	BOOL	Rückmeldung des angeschlossenen Aggregats der Schrittkettensteuerung. Die Rückmeldung wird zur Weiterschaltung in den nächsten Schritt benötigt. *bFdb* wird nur berücksichtigt, wenn das Aggregat das Aktive in der Schrittkettensteuerung ist. Durch Parametrierung der einzelnen Elemente des Array *arrIgnoreFdbVal* mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Rückmeldung des Aggregates ignoriert werden.
fDlyOn	REAL	Zeitangabe der Einschaltverzögerung [s]. Die Zeitangabe wird zur Weiterschaltung in den nächsten Schritt benötigt. Durch Parametrierung der einzelnen Elemente des Array *arrIgnoreDelayOnVal* mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Einschaltverzögerung des Schrittes ignoriert werden.
fDlyOff	REAL	Zeitangabe der Ausschaltverzögerung [s]. Die Zeitangabe wird zur Ausschaltung des aktiven Schrittes benötigt. Durch Parametrierung der einzelnen Elemente des Array *arrIgnoreDelayOffVal* mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Ausschaltverzögerung des Schrittes ignoriert werden.

Ausgänge

Name	Typ	Beschreibung
stCmd	ST_BA_Binary [▶ 282]	Die Kommandostruktur *stCmd* übermittelt dem angeschlossenen Aggregat die Freigaben und Schaltwerte der Prioritäten.
nRemSecStepUp	UDINT	Countdown zur Einschaltung des nächsten Schrittes [s].
nRemSecStepDown	UDINT	Countdown zur Ausschaltung des aktiven Schrittes [s].

Ein- / Ausgänge

Name	Typ	Beschreibung
stStep	ST_BA_Step [▶ 282]	Daten- und Befehlsstruktur zwischen dem Schrittkettenbaustein *FB_BA_StepBinary* und dem Steuerbaustein der Schrittkette FB_BA_StepCtrlAgg16 [▶ 482].

Eingänge CONSTANT

Name	Typ	Beschreibung
OpModVal	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer ermittelt den Einschaltwert des angebundenen Aggregates. Das resultierende Ergebnis wird an den Ausgang *bVal* übermittelt.
DelayOn	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer und das angebundene Netzwerk von Funktionen ermitteln die Weiterschaltverzögerung in den nächsten Schritt. Das resultierende Ergebnis wird an den Ausgang *nDelayOn* übermittelt.
DelayOff	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer und das angebundene Netzwerk von Funktionen ermitteln die Ausschaltverzögerung des aktuellen Schrittes. Das resultierende Ergebnis wird an den Ausgang *nDelayOff* übermittelt.
IgnoreFdb	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer und das angebundene Netzwerk von Funktionen ermitteln die Rückmeldung des Aggregates. Das resultierende Ergebnis wird an den Ausgang *_bFdb* übermittelt.
Step	FB_BA_StepBinary [▶ 484]	Der Funktionsbaustein repräsentiert den binären Empfangsbaustein einer Schrittkettensteuerung und steuert über die Kommandostruktur *stCmd* das angeschlossene Aggregat an. Der Datenaustausch zum Steuerbaustein der Schrittkettensteuerung (FB_BA_StepCtrlAgg16 [▶ 482]) geschieht über die Daten- und Kommandostruktur *stStep*.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bInit	BOOL	Die Variable dient der Initialisierung der Arrays *arrOpModVal*, *arrIgnoreDelayOnVal*, *arrIgnoreDelayOffVal*, *arrIgnoreFdbVal* in den Methoden unterhalb des Templates FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 799]. Nach dem Starten der Steuerung wird die Variable nach einem SPS-Zyklus auf FALSE gesetzt und dieser Zustand persistent gespeichert.
arrOpModVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays kann für die unterschiedlichen Betriebsarten *eOpMode* der Einschaltwert des angebundenen Aggregates am Schrittkettenbaustein definiert werden.
arrIgnoreDelayOnVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Weiterschaltverzögerung in den nächsten Schritt innerhalb der Schrittkettensteuerung ignoriert werden.
arrIgnoreDelayOffVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Ausschaltverzögerung des aktuellen Schrittes innerhalb der Schrittkettensteuerung ignoriert werden.
arrIgnoreFdbVal	ARRAY [1..16] OF BOOL	Durch Parametrierung der einzelnen Elemente des Arrays mit einem TRUE kann für die unterschiedlichen Betriebsarten *eOpMode* die Rückmeldung des Aggregates *bFdb* ignoriert werden.

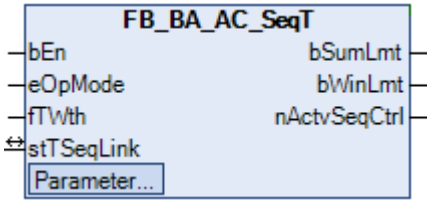
Variablen

Name	Typ	Beschreibung
bVal	BOOL	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *OpModVal* und des Arrays *arrOpModVal*. *bVal* definiert den Einschaltwert am Schrittkettenbaustein *Step*.
_bFdb	BOOL	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *IgnoreFdb* und des Arrays *arrIgnoreFdbVal*. *_bFdb* definiert die Weiterschaltbedingung in den nächsten Schritt am Schrittkettenbaustein *Step*.
nDelayOn	UDINT	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *DelayOn* und des Arrays *arrIgnoreDelayOnVal*. *nDelayOn* definiert die Weiterschaltverzögerung in den nächsten Schritt am Schrittkettenbaustein *Step*.
nDelayOff	UDINT	Die Variable ist das resultierende Ergebnis der Betriebsart *eOpMode*, des Multiplexers *DelayOff* und des Arrays *arrIgnoreDelayOffVal*. *nDelayOff* definiert die Ausschaltverzögerung des aktuellen Schrittes am Schrittkettenbaustein *Step*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.3.5 FB_BA_AC_SeqT



Das Template repräsentiert das Starten und Steuern der Zulufttemperatur Sequenzregelung einer raumlufttechnischen (RLT) Anlage.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Startverhalten der Sequenz

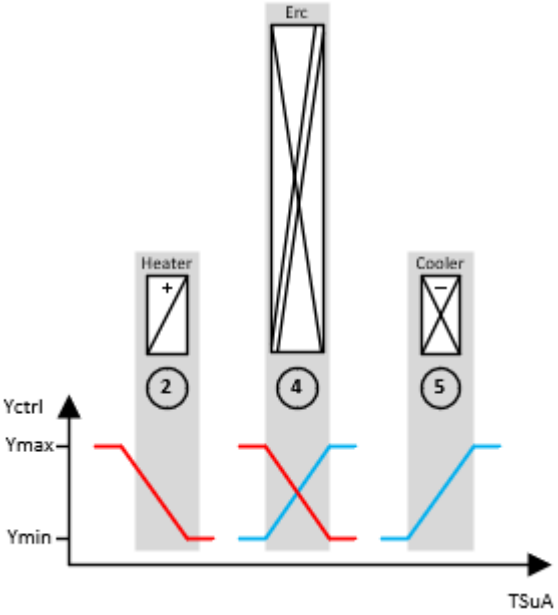
Beim Anfahren der RLT-Anlage wird bestimmt, ob sie mit der Heiz-, Kühl- oder WRG-Sequenz gestartet wird. Die Auswahl der Startsequenz erfolgt in Abhängigkeit der Anlagenbetriebsart *eOpMode* und der Außentemperatur *fTWth*. Das Ergebnis des Netzwerkes an dem Eingang *nNumStartCtrl* des Funktionsbausteins *SeqLink* bestimmt den Startregler der Regelsequenz.

Ist der an *nNumStartCtrl* angegebene Sequenzregler nicht betriebsbereit, so wird mit dem Parameter-Enum eNoStartCtrlFound des Funktionsbausteins *SeqLink* die Vorgehensweise definiert, um einen neuen Startregler zu finden. Somit ist gewährleistet, dass die Sequenz immer aufstartet.

Steuerung der Sequenzreglung

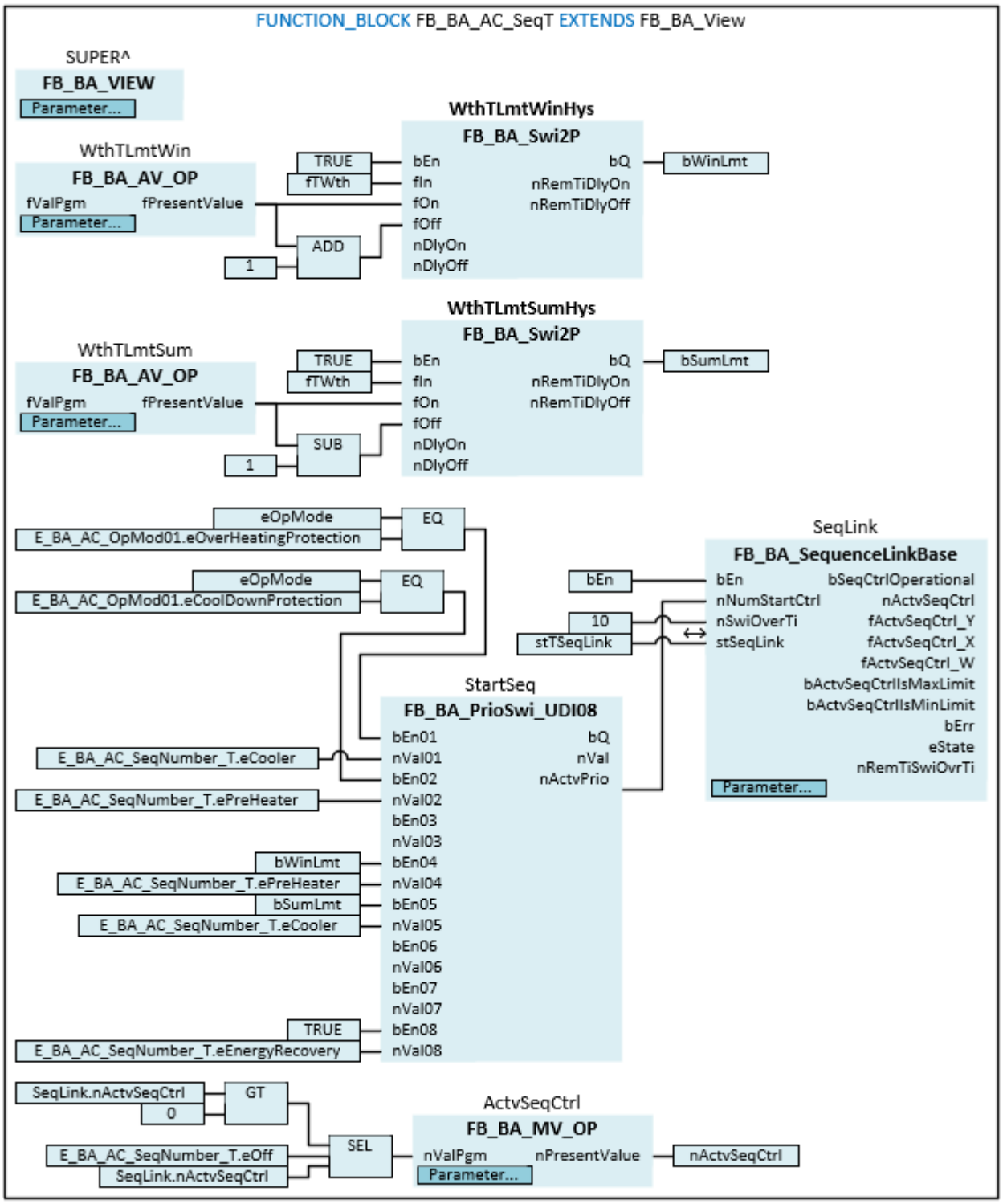
Nur ein Element der Sequenz kann regelnd sein. Erreicht ein regelnder Sequenzregler seinen Maximal- oder Minimalwert (Ymax,Ymin), so wird je nach Regler-Wirksinn auf den nächsten Regler in der Rangfolge umgeschaltet.

Das Umschalten auf einen anderen Sequenzregler kann durch die Zeitangabe *nSwiOverTi* des Funktionsbausteins *SeqLink* verzögert werden.

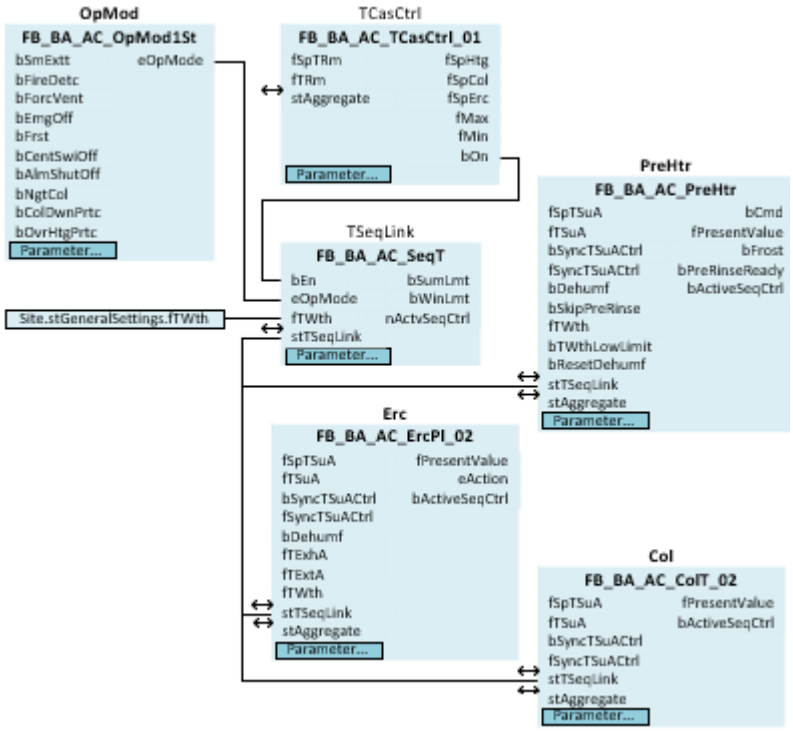


Die Reihenfolge der Temperatur Regelsequenzen ist durch die global definierte Aufzählungsvariable E_BA_SeqNumber_T [▶ 710] vorgegeben.

Blockschaltbild



Prinzipschema Anlage



Syntax

```
FUNCTION_BLOCK FB_BA_AC_SeqT EXTENDS FB_BA_View
VAR_INPUT
  bEn                 : BOOL;
  eOpMode             : E_BA_AC_OpMod01;
  fTWth               : REAL;
END_VAR
VAR_OUTPUT
  bSumLmt             : BOOL;
  bWinLmt             : BOOL;
  nActvSeqCtrl        : UDINT;
END_VAR
VAR_IN_OUT
  stTSeqLink          : ST_BA_SeqLink;
END_VAR
VAR_INPUT CONSTANT
  WthTLmtSum          : FB_BA_AV_Op;
  WthTLmtWin          : FB_BA_AV_Op;
  SeqLink             : FB_BA_SequenceLinkBase;
  ActvSeqCtrl         : FB_BA_MV_Op;
END_VAR
VAR
  WthTLmtWinHys       : FB_BA_Swi2P;
  WthTLmtSumHys       : FB_BA_Swi2P;
  StartSeq            : FB_BA_PrioSwi_UDI08;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Aktivierung der Zulufttemperatur-Sequenzregelung.
eOpMode	E_BA_AC_OpMod01 [▶ 712]	Die Anlagenbetriebsart dient der Auswahl des Startsequenzreglers.
fTWth	REAL	Eingang für die Witterungstemperatur.

Ausgänge

Name	Typ	Beschreibung
bSumLmt	BOOL	Die Ausgangsvariable zeigt an, dass die Lüftungsanlage mit der Kühlsequenz startet.
bWinLmt	BOOL	Die Ausgangsvariable zeigt an, dass die Lüftungsanlage mit der Heizsequenz startet.
nActvSeqCtrl	UDINT	Die Ausgangsvariable zeigt die Nummer des aktiven Sequenzreglers an.

Ein- / Ausgänge

Name	Typ	Beschreibung
stTSeqLink	ST_BA_SeqLink [▶ 125]	Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern FB_BA_SeqCtrl der Temperaturregelsequenz (Vorerhitzer, Energierückgewinnung, Kühler) und dem Steuerbaustein *SeqLink*.

Eingänge CONSTANT

Name	Typ	Beschreibung
WthTLmtSum	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Grenzwertes ab welcher Außentemperatur die RLT-Anlage in der Heizsequenz startet.
WthTLmtWin	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Grenzwertes ab welcher Außentemperatur die RLT-Anlage in der Kühlsequenz startet
SeqLink	FB_BA_SequenceLinkBase [▶ 175]	Der Funktionsbaustein *SeqLink* ist das Kernstück des Templates *FB_BA_AC_SeqT*. Der Sequenzlinker ist über die Datenstruktur stTSeqLink [▶ 125] mit den Zuluftreglern der Sequenz verbunden. Er ist das zentrale Steuerorgan und für das Umschalten zwischen den Sequenzreglern und das Starten der Regelsequenz verantwortlich.
ActvSeqCtrl	FB_BA_MV_Op [▶ 244]	Das MV-Objekt zeigt den aktuell aktiven Sequenzregler an.

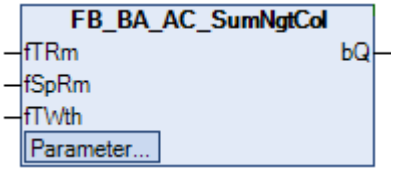
Variablen

Name	Typ	Beschreibung
WthTLmtWinHys	FB_BA_Swi2P	Zweipunktschalter, welcher anhand der Witterungstemperatur *fTWth* und des Grenzwertes Witterungstemperatur Winter *WthTLmtWin* das binäre Schaltsignal für *bWinLmt* erzeugt.
WthTLmtSumHys	FB_BA_Swi2P	Zweipunktschalter, welcher anhand der Witterungstemperatur *fTWth* und des Grenzwertes Witterungstemperatur Sommer *WthTLmtSum* das binäre Schaltsignal für *bWinLmt* erzeugt.
StartSeq	FB_BA_PrioSwi_UDI08 [▶ 443]	Der Prioritätenschalter dient zur Auswahl der Startsequenz. **Prio 1: bEN01** Bei der Betriebsart Überhitzungsschutz *E_BA_AC_OpMod01.eOverHeatingProtection* wird grundsätzlich mit der Sequenz Kühlen (*E_BA_AC_SeqNumber.eCooler*) gestartet. **Prio 2: bEN02** Bei der Betriebsart Auskühlschutz *E_BA_AC_OpMod01.eCoolDownProtection* wird grundsätzlich mit der Sequenz Heizen (*E_BA_AC_SeqNumber.ePreHeater*) gestartet. **Prio 4: bEN04** Der vorgelagerte Funktionsbaustein *WthTLmtWinHys* prüft, ob die tatsächliche Außentemperatur unterhalb des kritischen Wertes von AV-Objekt *WthTLmtSum* liegt. Ist dieses der Fall dann startet die RLT-Anlage mit dem Sequenzregler des Vorerhitzers (*E_BA_AC_SeqNumber.ePreHeater*). **Prio 5: bEN05** Der vorgelagerte Funktionsbaustein *WthTLmtSumHys* prüft, ob die tatsächliche Außentemperatur oberhalb des Wertes von AV-Objekt *WthTLmtSum* liegt. Ist dieses der Fall dann startet die RLT-Anlage mit dem Sequenzregler des Kühlers(*E_BA_AC_SeqNumber.eCooler*). **Prio 8: bEN08** In den Übergangszeiten zwischen Winter und Sommer sind die Eingänge von *StartSeq.bEn04* und *StartSeq.bEn05* witterungsbedingt FALSE. In diesem Fall gilt am Prioritätenschalter die Priorität 8. Damit wird die RLT-Anlage in der Sequenz der Energierückgewinnung (*E_BA_AC_SeqNumber.eEnergyRecovery*) gestartet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

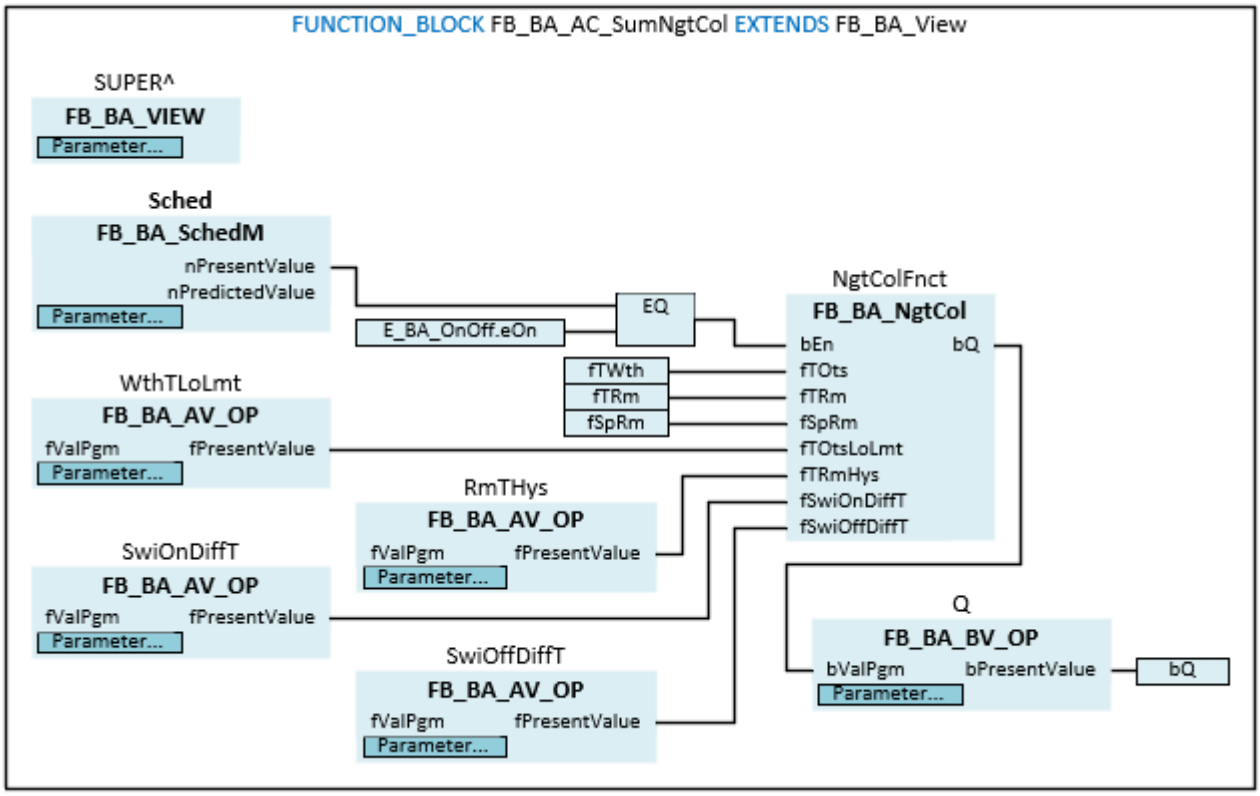
6.1.4.2.1.1.3.6 FB_BA_AC_SumNgtCol



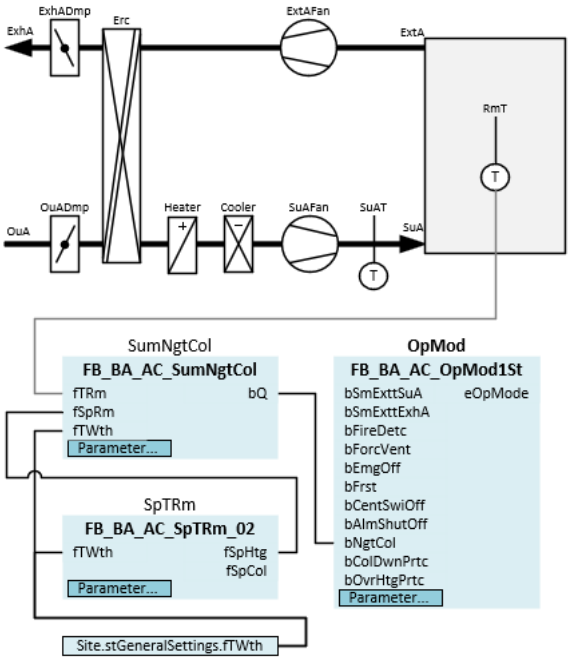
Das Template repräsentiert eine Nachtkühlfunktion für Lüftungsanlagen. Es wird verwendet, um nachts mit kühler Außenluft die tags zuvor aufgeheizten Räume herunterzukühlen. Die Funktion der Sommernachtkühlung dient zur Verbesserung der Luftqualität und zur Einsparung elektrischer Energie. In den ersten Stunden des nächsten Sommertages wird elektrische Energie zur Kälteerzeugung eingespart.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Prinzipschema



Syntax

```
FUNCTION_BLOCK FB_BA_AC_SumNgtCol EXTENDS FB_BA_View
VAR_INPUT
  fTRm                : REAL;
  fSpRm               : REAL;
  fTWth               : REAL;
END_VAR
VAR_OUTPUT
  bQ                  : BOOL;
END_VAR
VAR_INPUT CONSTANT
  WthTLoLmt           : FB_BA_AV_Op;
  RmTHys              : FB_BA_AV_Op;
  SwiOnDiffT          : FB_BA_AV_Op;
  SwiOffDiffT         : FB_BA_AV_Op;
  Q                   : FB_BA_BV_Op;
  Sched               : FB_BA_SchedM;
END_VAR
VAR  NgtColFnct       : FB_BA_NgtCol;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fTRm	REAL	Eingang Messwert Raumtemperatur.
fSpRm	REAL	Eingang Sollwert Raumtemperatur.
fTWth	REAL	Eingang für die Witterungstemperatur.

Ausgänge

Name	Typ	Beschreibung
bQ	BOOL	Sommernachtkühlung eingeschaltet.

Eingänge CONSTANT

Name	Typ	Beschreibung
WthTLoLmt	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für die untere Außentemperatur. Der Wert soll ein zu starkes Auskühlen verhindern.
RmTHys	FB_BA_AV_Op [▶ 207]	Eingabe der Raumtemperatur-Hysterese.
SwiOnDiffT	FB_BA_AV_Op [▶ 207]	Einschaltwert Temperaturdifferenz zwischen der Raumtemperatur und der Außentemperatur [K].
SwiOffDiffT	FB_BA_AV_Op [▶ 207]	Ausschaltwert Temperaturdifferenz zwischen der Raumtemperatur und der Außentemperatur [K].
Q	FB_BA_BV_Op [▶ 221]	Das BV-Objekt signalisiert, dass die Nachtkühlung eingeschaltet ist.
Sched	FB_BA_SchedM [▶ 230]	Mit dem Schedule-Objekt werden die Zeiträume definiert, in denen die Nachtkühlung freigegeben ist.

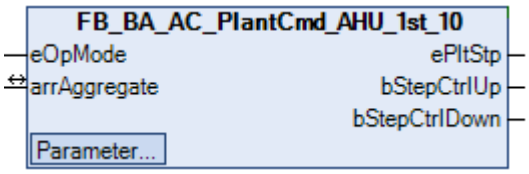
Variablen

Name	Typ	Beschreibung
NgtColFct	FB_BA_NgtCol [▶ 294]	Der Funktionsbaustein ist das Kernstück des Templates und beinhaltet den eigentlichen Steuerungsprozess des Nachtkühlprogramms.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.3.7 FB_BA_AC_PlantCmd_AHU_1st_10



Das Template repräsentiert die Anlagenkommandierung einer raumlufttechnischen Anlage ohne Be- und Entfeuchtung.

Der Funktionsbaustein *PlantControl* ist das Kernstück der Anlagenkommandierung. Er dient als übergeordnete Anlagensteuereinheit der angeschlossenen Aggregate.

Entsprechend der vorgegebenen Anlagenbetriebsart *eOpMode* und den Angaben in der Steuermatrix der Anlagensteuereinheit *PlantControl* werden für jedes Aggregat die entsprechenden Ein- respektive Ausschaltbefehle und dessen Priorität ermittelt.

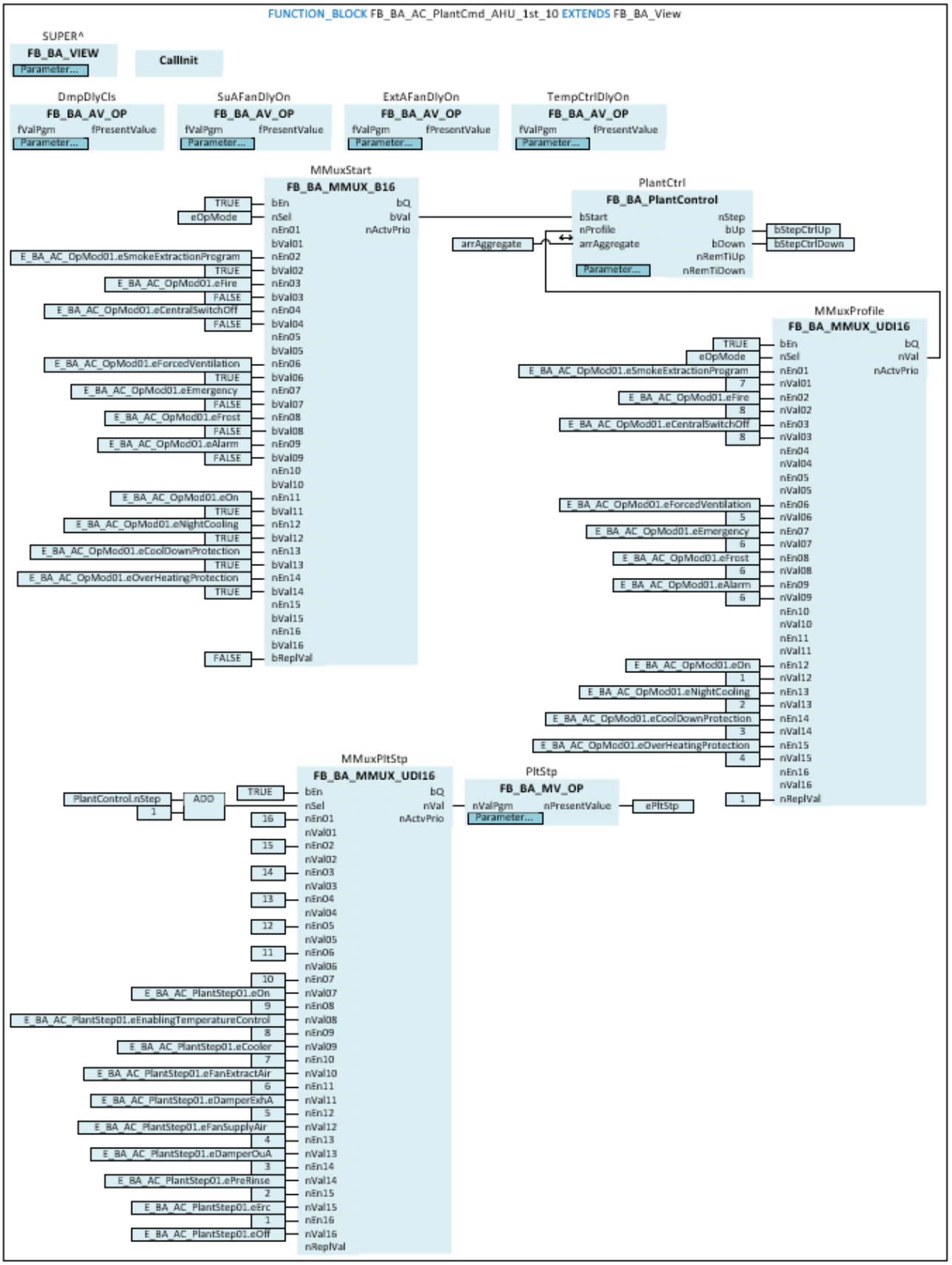
Die bidirektionale Kommunikation zu den Aggregaten der raumlufttechnischen Anlage geschieht über das Array *arrAggregate*, siehe Prinzipschema Anlage.

Die Parametrierung der Steuermatrix geschieht in den Methoden unterhalb dieses Templates.

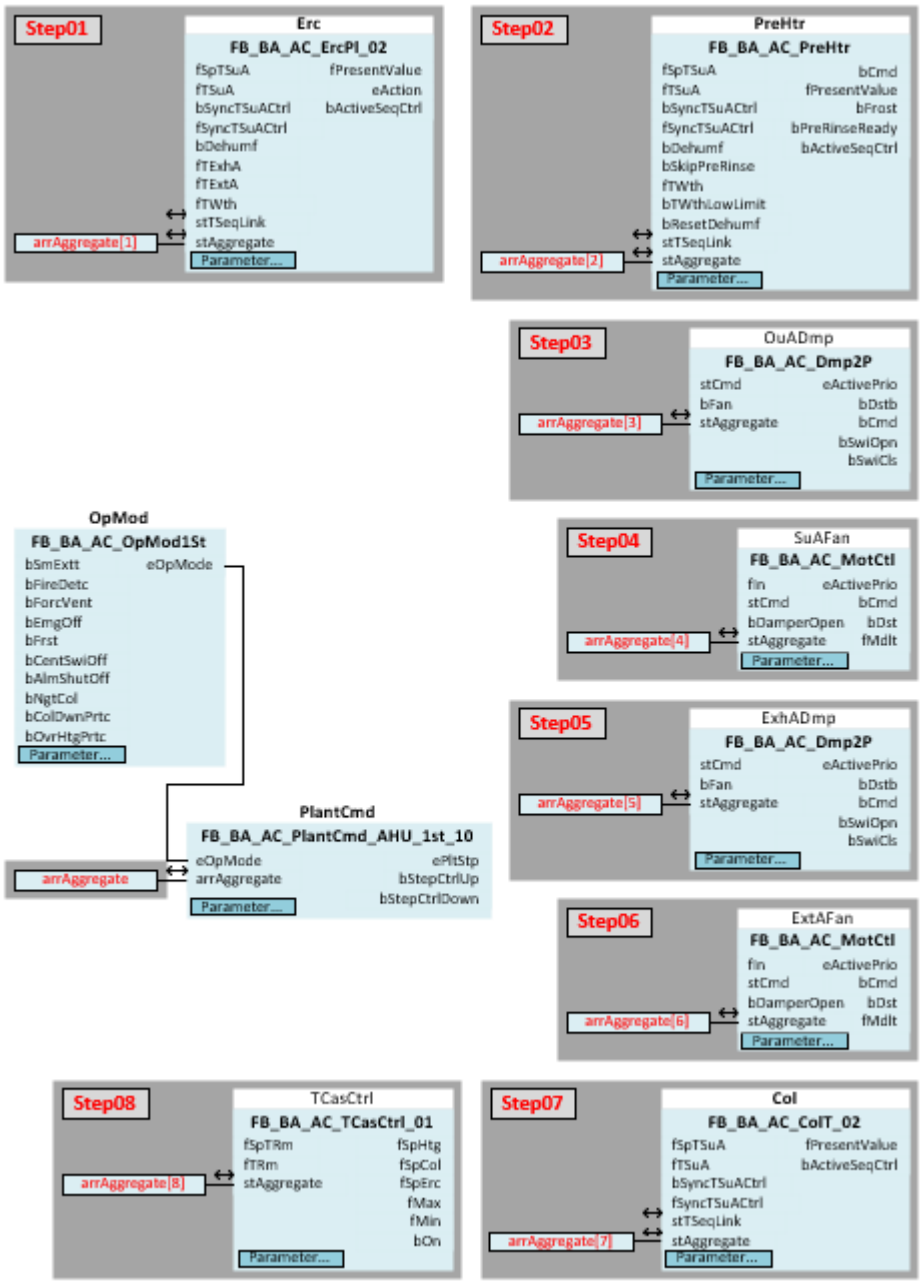


Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Prinzipschema Anlage



Syntax

```
FUNCTION_BLOCK FB_BA_AC_PlantCmd_AHU_1st_10 EXTENDS FB_BA_View
VAR_INPUT
  eOpMode                 : E_BA_AC_OpMod01;
END_VAR
VAR_OUTPUT
  ePltStp                 : E_BA_AC_PlantStep01;
  bStepCtrlUp             : BOOL;
  bStepCtrlDown           : BOOL;
END_VAR
VAR_IN_OUT
  arrAggregate            : ARRAY [1..MAX(1,BA_Param.nMaxAggregate)] OF ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  DmpDlyCls               : FB_BA_AV_Op;
  SuAFanDlyOn             : FB_BA_AV_Op;
  ExtAFanDlyOn            : FB_BA_AV_Op;
  TempCtrlDlyOn           : FB_BA_AV_Op;
  PltStp                  : FB_BA_MV_Op;
  PlantControl            : FB_BA_PlantControl;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bInit                   : BOOL := TRUE;
```

```
END_VAR
VAR
  MMuxStart             : FB_BA_MMUX_B16;
  MMuxProfile           : FB_BA_MMUX_UDI16;
  MMuxPltStp            : FB_BA_MMUX_UDI16;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eOpMode	E_BA_AC_OpMod01 [▶ 712]	Eingang der aktuellen Betriebsart *eOpMode*, siehe Template FB_BA_AC_OpMod1St [▶ 776].

Ausgänge

Name	Typ	Beschreibung
ePltStp	E_BA_AC_PlantStep01 [▶ 711]	Ausgabe des aktuellen Anlagenschrittes der raumlufttechnischen Anlage.
bStepCtrlUp	BOOL	Zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung im aufsteigenden Zustand befindet.
bStepCtrlDown	BOOL	zeigt an, dass sich die Reihenfolge der Schrittkettensteuerung im absteigenden Zustand befindet.

Ein- / Ausgänge

Name	Typ	Beschreibung
arrAggregate	ARRAY [1..MAX(1,BA_Param.nMaxAggregate [▶ 289])] OF ST_BA_Aggregate [▶ 283]	Das Array beinhaltet für jedes Aggregat eine bidirektionale Daten- und Kommandostruktur. Beim Hoch- respektive Zurückschalten der Schrittkettensteuerung wird der für jedes Aggregat ermittelte Schaltbefehl über die bidirektionale Kommunikationsstruktur *arrAggregate* übergeben. Eine Rückmeldung der Aggregate für das Hoch- oder Runterschalten der Schrittkettensteuerung wird über die Kommunikationsstruktur an die Anlagensteuereinheit *PlantControl* übertragen.

Eingänge CONSTANT

Name	Typ	Beschreibung
DmpDlyCls	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes, um das Schließen der Klappen beim Herunterfahren der RLT-Anlage zu verzögern. Damit wird die Nachlaufzeit der Ventilatoren beim Abschalten der Anlage berücksichtigt.
SuAFanDlyOn	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes für die Einschaltverzögerung des Zuluftventilators beim Einschalten der RLT-Anlage.
ExtAFanDlyOn	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes für die Einschaltverzögerung des Abluftventilators beim Einschalten der RLT-Anlage.
TempCtrlDlyOn	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes für die zeitverzögerte Freigabe der Temperaturregelung.
PltStp	FB_BA_MV_Op [▶ 244]	Das MV-Objekt zeigt den aktuellen Schritt der Schrittkettensteuerung der Anlage an.
PlantControl	FB_BA_PlantControl [▶ 491]	Der Funktionsbaustein repräsentiert die übergeordnete Anlagensteuereinheit einer Schrittkettensteuerung von Aggregaten. Der Baustein stellt sicher, dass die einzelnen Aggregate einer Anlage in einer bestimmten Reihenfolge sequenziell hintereinander ein- respektive ausgeschaltet werden.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bInit	BOOL	Die Variable dient dem Aufruf der Initialisierungsmethoden für die Steuermatrix in der Methode CallInit [▶ 804]. Nach dem Starten der Steuerung wird die Variable nach einem SPS-Zyklus auf FALSE gesetzt und dieser Zustand persistent gespeichert.

Variablen

Name	Typ	Beschreibung
MMuxStart	FB_BA_MMUX_B16 [▶ 438]	Der Multiplexer definiert anhand der Anlagenbetriebsart *eOpMode* ob die raumlufttechnische Anlage gestartet oder gestoppt wird.
MMuxProfile	FB_BA_MMUX_UDI16 [▶ 438]	Der Multiplexer definiert anhand der Anlagenbetriebsart *eOpMode* das Steuerungsprofil der Anlagensteuerung *PlantControl*.
MMuxPltStp	FB_BA_MMUX_UDI16 [▶ 438]	Der Multiplexer wertet die Schritte der Schrittkettensteuerung aus und überträgt diese an das MV-Objekt *PltStp* zur Anzeige in der Managementebene.

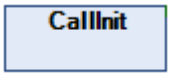
Methoden

Name	Definitionsort	Beschreibung
CallInit [▶ 804]	Lokal	Aufruf der Initialisierungsmethoden zur Parametrierung der Steuermatrix der Anlagensteuereinheit.
ProfileDescription [▶ 804]	Lokal	Dient der Initialisierung des Parameterarray *arrProfileDescription* der Anlagensteuereinheit.
ProfileParameter0x [▶ 805]	Lokal	Initialisierung eines Teils des mehrdimensionalen Profil Parameter Array *arrProfileParameter* der Steuereinheit.
StepDescription [▶ 805]	Lokal	Dient der Initialisierung des Parameterarray *arrStepDescription* der Anlagensteuereinheit.

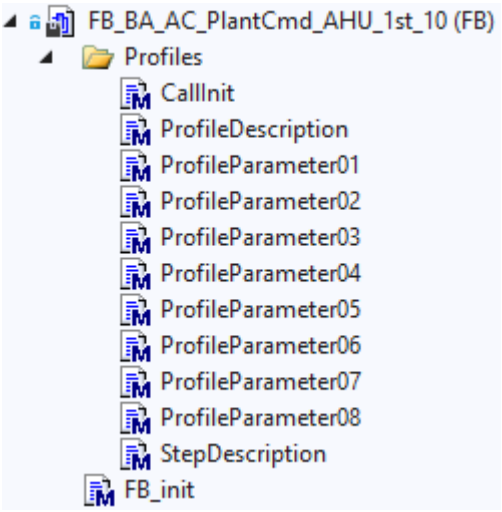
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.3.7.1 CallInit



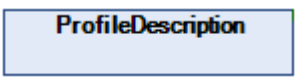
Die Methode dient dem Aufruf der Initialisierungsmethoden zur Parametrierung der Steuermatrix der Anlagensteuereinheit *PlantControl*.



Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.3.7.2 ProfileDescription

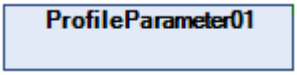


Die Methode dient der Initialisierung des Parameterarray *arrProfileDescription* der Anlagensteuereinheit *PlantControl*. Jedes Element des Arrays steht für ein Profil. Dementsprechend werden in dieser Methode die Namen der Profile vergeben und die dazu gehörige Priorität *nPrio* festgelegt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.3.7.3 ProfileParameter0x



Die Methode dient der Initialisierung eines Teils des mehrdimensionalen Profil Parameter Array *arrProfileParameter* der Steuereinheit PlantControl [▶ 491].

Diese Beschreibung steht stellvertretend für alle acht Methoden dieser Art.

Das erste Element des Arrays der Methode ProfileParameter01 steht für das Anlagenprofil „Auto“.

Das erste Element des Arrays der Methode ProfileParameter02 steht für das Anlagenprofil „NightCooling“.

Das erste Element des Arrays der Methode ProfileParameter03 steht für das Anlagenprofil „CoolDownProtection“.

Das erste Element des Arrays der Methode ProfileParameter04 steht für das Anlagenprofil „OverHeatingProtection“.

Das erste Element des Arrays der Methode ProfileParameter05 steht für das Anlagenprofil „ForcedVentilation“.

Das erste Element des Arrays der Methode ProfileParameter06 steht für das Anlagenprofil „CriticalOff“.

Das erste Element des Arrays der Methode ProfileParameter07 steht für das Anlagenprofil „SmokeExtractionProgram“.

Das erste Element des Arrays der Methode ProfileParameter08 steht für das Anlagenprofil „LifeSafetyOff“.

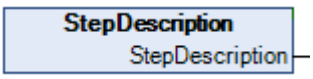
Das zweite Element steht jeweils für die Schritte innerhalb der Schrittkettensteuerung.

Parametriert werden die Schaltwerte für die Aggregate (*bValue*, *nValue*, *fValue*), die Verzögerungszeiten für die Schrittkettensteuerung (*fDelayUp*, *fDelayDown*) und die Freigaben der Verwendung der Prozessrückmeldung der Aggregate in dem Schritt (*bEnFdbUp*, *bEnFdbDown*).

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.3.7.4 StepDescription



Die Methode dient der Initialisierung des Parameterarray *arrStepDescription* der Anlagensteuereinheit *PlantControl*.

In dem Parameterarray wird der Klartext der Anlagenschritte festgelegt. Diese Angabe dient rein der Anzeige.

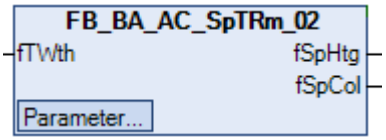
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.4 Setpoint

6.1.4.2.1.1.4.1 Temperature

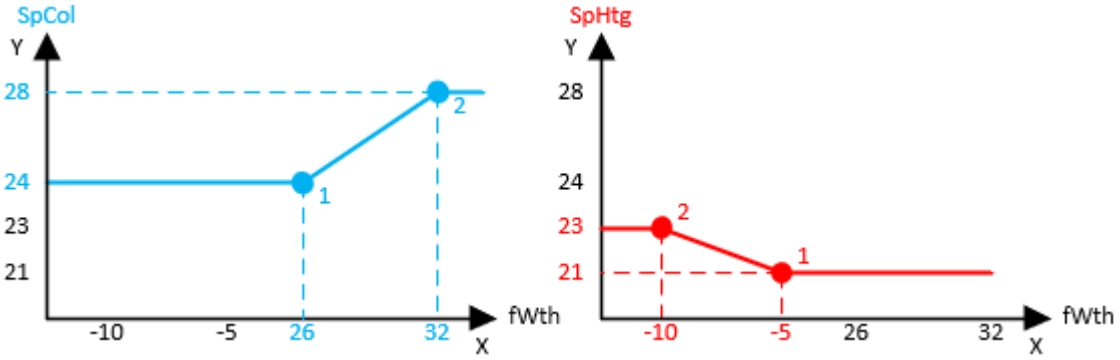
6.1.4.2.1.1.4.1.1 FB_BA_AC_SpTRm_02



Das Template ist ein Sollwertprogramm für eine raumlufttechnische Anlage.

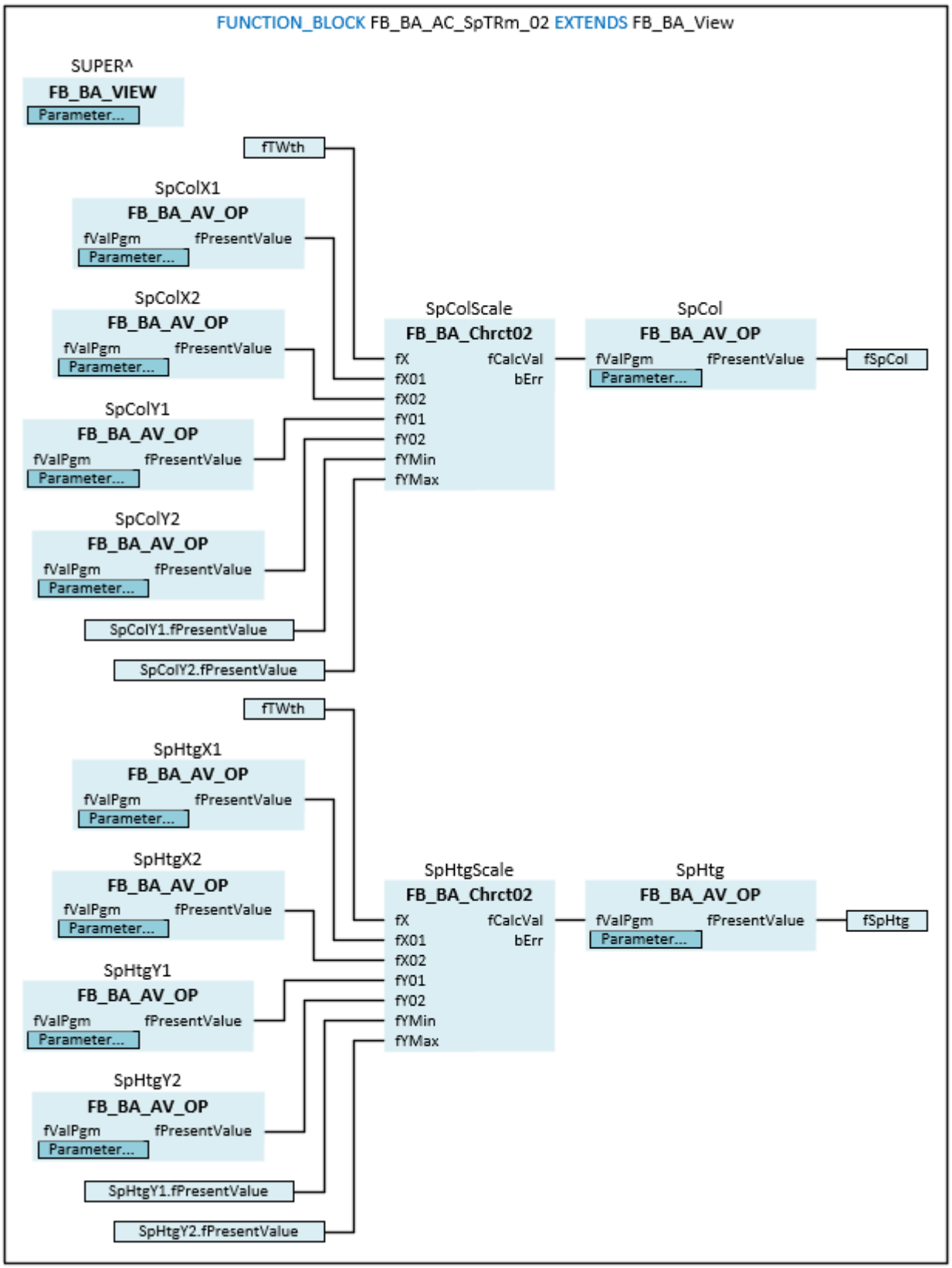
Es dient als Sollwertprogramm für eine Abluft- / Zuluftkaskade mit jeweils einem Raumtemperatursollwert für den Heiz- und Kühlbetrieb, inklusive Sommer- und Winterkompensation.

Es gibt zwei Raumtemperaturbasissollwerte (*SpHtgY1*, *SpColY1*) mit einer energieneutralen Zone zwischen dem unteren (Heizbetrieb) und oberen (Kühlbetrieb) Sollwert. Eine außentemperaturabhängige Verschiebung der Raumtemperaturbasissollwerte ist durch zwei Kurvenfunktionen realisiert (Sommer-Winterkompensation).

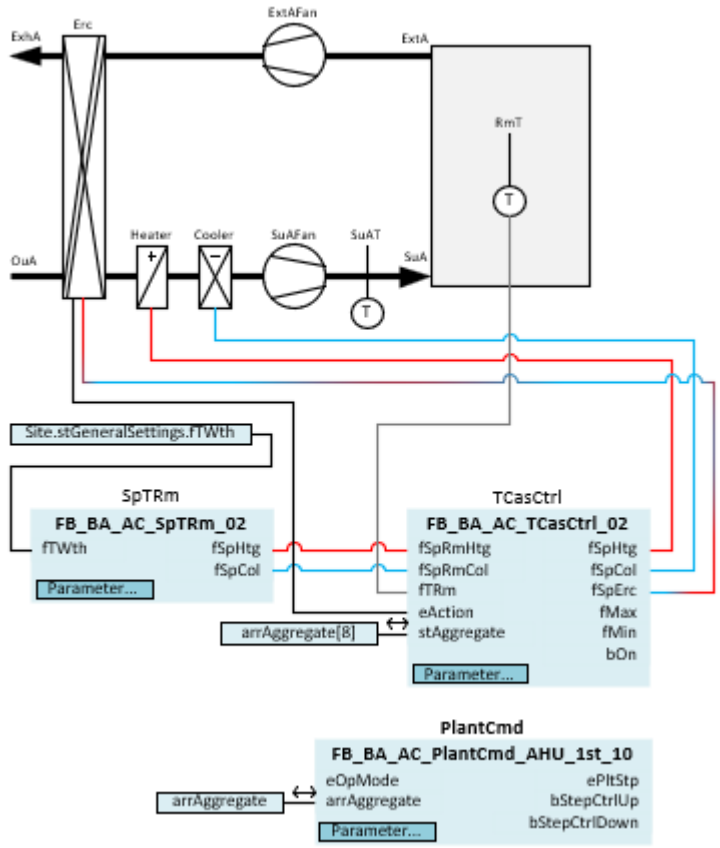


Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

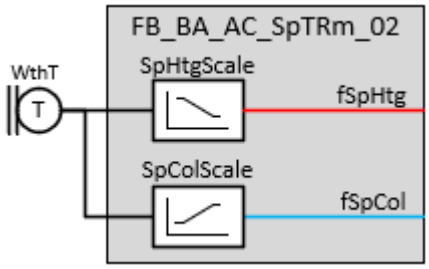
Blockschaltbild



Prinzipschema01



Prinzipschema02



Syntax

```
FUNCTION_BLOCK FB_BA_AC_SpTRm_02 EXTENDS FB_BA_View
VAR_INPUT
  fTWth          : REAL;
END_VAR
VAR_OUTPUT
  fSpHtg         : REAL;
  fSpCol         : REAL;
END_VAR
VAR_INPUT CONSTANT
  SpColX1        : FB_BA_AV_Op;
  SpColX2        : FB_BA_AV_Op;
  SpColY1        : FB_BA_AV_Op;
  SpColY2        : FB_BA_AV_Op;

  SpHtgX1        : FB_BA_AV_Op;
  SpHtgX2        : FB_BA_AV_Op;
  SpHtgY1        : FB_BA_AV_Op;
  SpHtgY2        : FB_BA_AV_Op;

  SpHtg          : FB_BA_AV_Op;
  SpCol          : FB_BA_AV_Op;
END_VAR
VAR
```

```
  SpColScale    : FB_BA_Chrct02;
  SpHtgScale    : FB_BA_Chrct02;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
fTWth	REAL	Aktueller Wert der Außentemperatur.

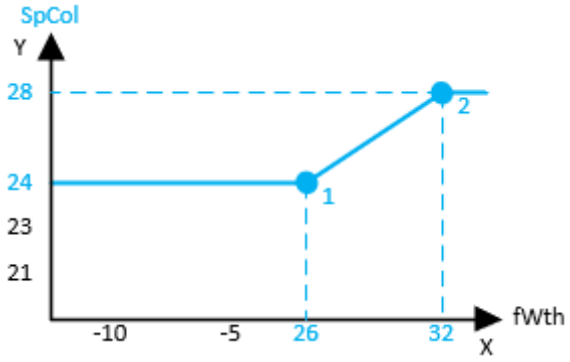
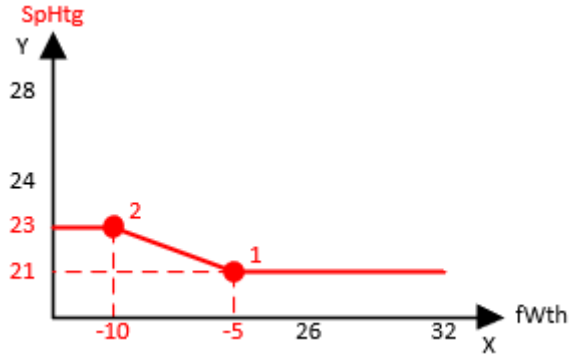
Ausgänge

Name	Typ	Beschreibung
fSpHtg	REAL	Errechneter Sollwert der Zulufttemperatur für den Erhitzer.
fSpCol	REAL	Errechneter Sollwert der Zulufttemperatur für den Kühler.

Eingänge CONSTANT

Name	Typ	Beschreibung
SpColX1	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X1 der Sommerkompensation für den Sollwert Kühlen.
SpColX2	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X2 der Sommerkompensation für den Sollwert Kühlen.
SpColY1	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt Y1 der Sommerkompensation für den Sollwert Kühlen. *SpColY1* ist der Basis-Sollwert für das Kühlen.
SpColY2	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt Y2 der Sommerkompensation für den Sollwert Kühlen.
SpHtgX1	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X1 der Winterkompensation für den Sollwert Heizen.
SpHtgX2	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X2 der Winterkompensation für den Sollwert Heizen.
SpHtgY1	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt Y1 der Winterkompensation für den Sollwert Heizen. *SpHtgY1* ist der Basis-Sollwert für das Heizen.
SpHtgY2	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt Y2 der Winterkompensation für den Sollwert Heizen.
SpHtg	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Raumtemperatur-Sollwertes Heizen.
SpCol	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Raumtemperatur-Sollwertes Kühlen.

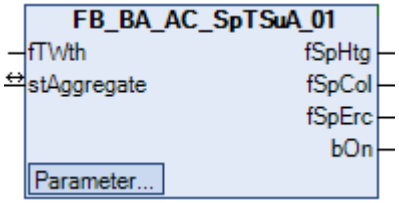
Variablen

Name	Typ	Beschreibung
SpColScale	FB_BA_Chrct02 [▶ 466]	Der Funktionsbaustein berechnet die Sollwertkennlinie Kühlen (Sommerkompensation) für die aktuelle Raumtemperatur in Abhängigkeit der Außentemperatur. 
SpHtgScale	FB_BA_Chrct02 [▶ 466]	Der Funktionsbaustein berechnet die Sollwertkennlinie Heizen (Winterkompensation) für die aktuelle Raumtemperatur in Abhängigkeit der Außentemperatur. 

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.1.4.1.2 FB_BA_AC_SpTSuA_01

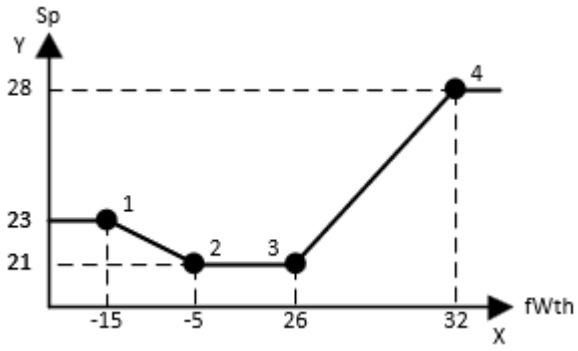


Das Template generiert den Zulufttemperatur-Sollwert einer raumlufttechnischen Anlage inklusive Sommer- und Winterkompensation.

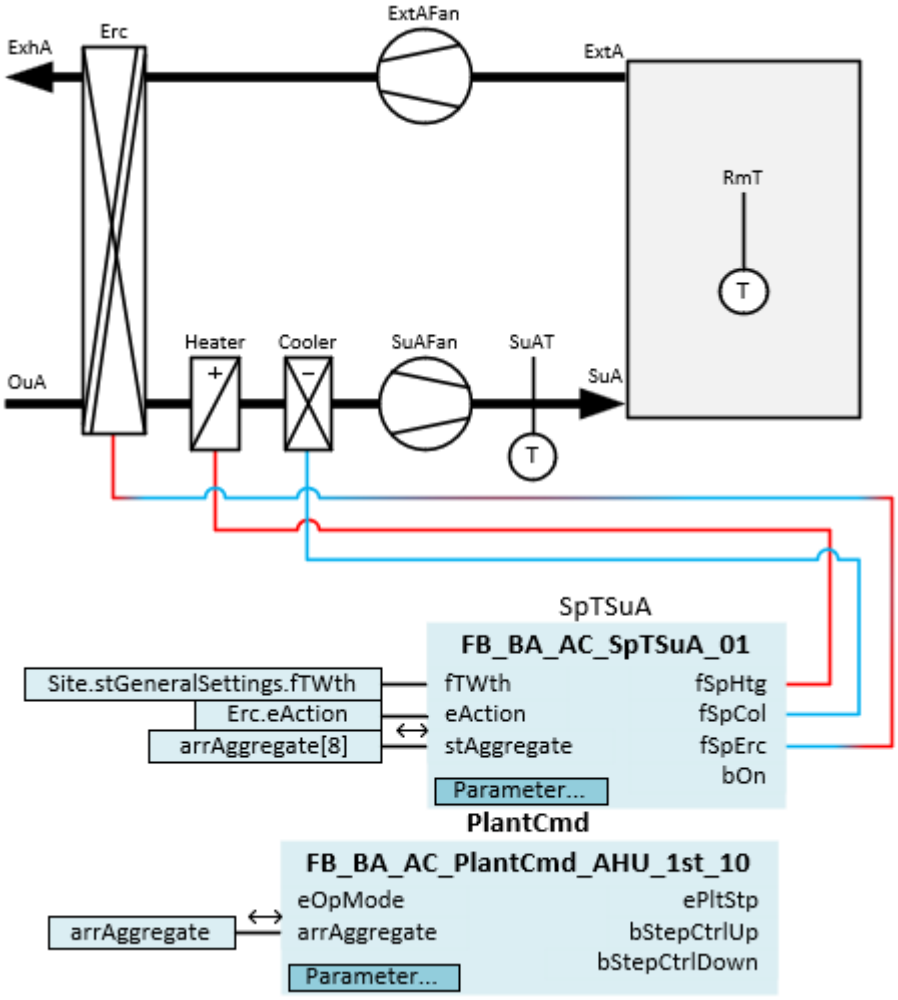
Die Bestimmung des Zulufttemperatur-Sollwertes erfolgt mit dem Funktionsbaustein Scale [▶ 469] in Abhängigkeit der Außentemperatur.

Die Freigabe der Zulufttemperatur-Sequenzregelung erfolgt mit der Variablen *bOn* durch die Auswertung des Empfangsbausteins der Schrittkettensteuerung *Aggregate*.

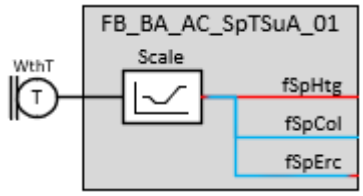
Der an die Variablen *fSpHtg*, *fSpCol* und *fSpErc* verknüpfte Sollwert, wird an die Zuluftregler der Aggregate Erhitzer, Kühler und Energierückgewinnung weitergegeben.



Prinzipschema 01

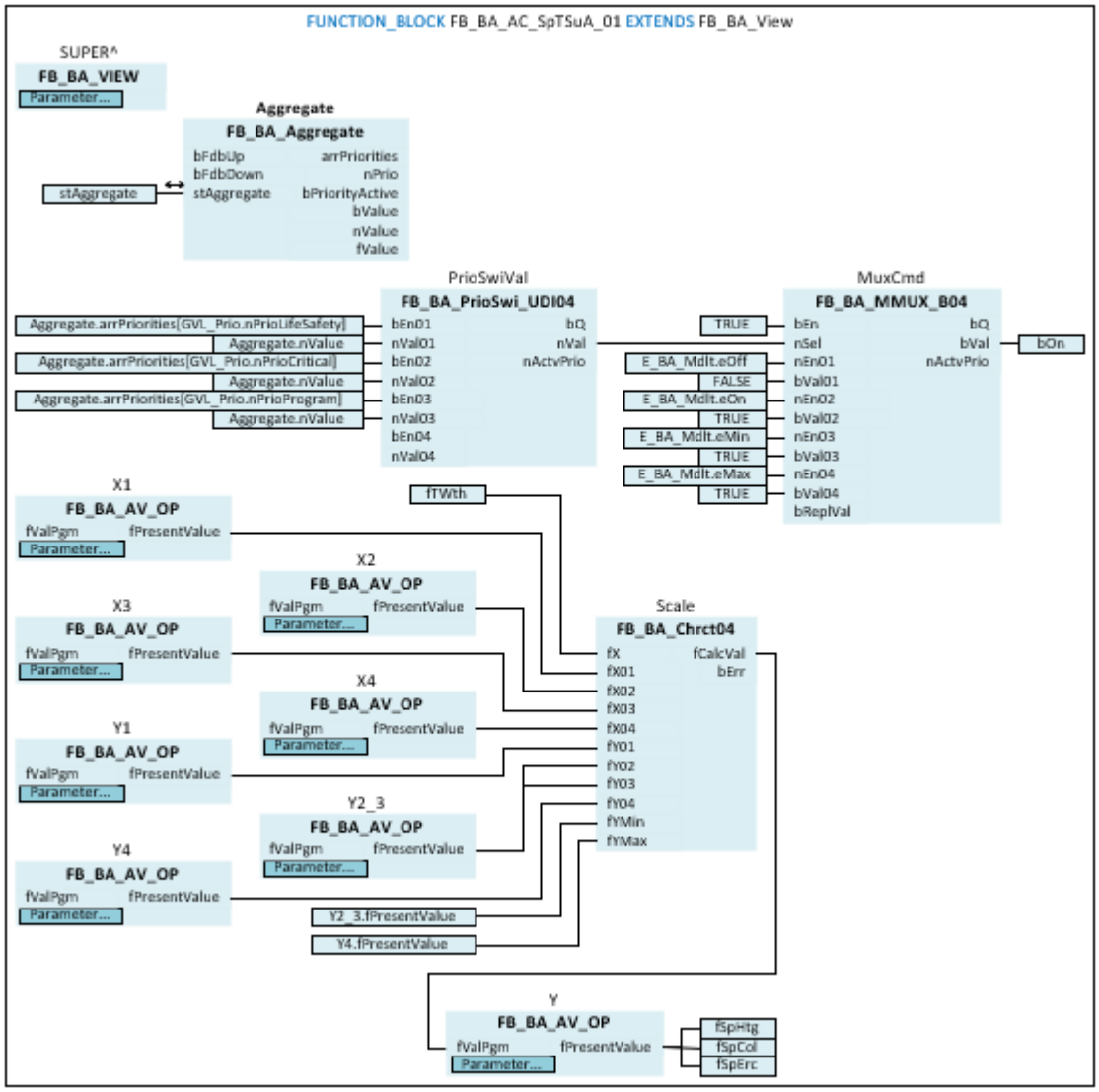


Prinzipschema 02



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_SpTSuA_01 EXTENDS FB_BA_View
VAR_INPUT
  fTWth          : REAL;
END_VAR
VAR_OUTPUT
  fSpHtg         : REAL;
  fSpCol         : REAL;
  fSpErc         : REAL;
  bOn            : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate    : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  X1             : FB_BA_AV_Op;
  X2             : FB_BA_AV_Op;
  X3             : FB_BA_AV_Op;
  X4             : FB_BA_AV_Op;
  Y1             : FB_BA_AV_Op;
  Y2_3           : FB_BA_AV_Op;
  Y4             : FB_BA_AV_Op;
  Sp             : FB_BA_AV_Op;
  Aggregate      : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiVal     : FB_BA_PrioSwi_UDI04;
  MuxCmd         : FB_BA_MMUX_B04;
  Scale          : FB_BA_Chrct04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fTWth	REAL	Aktueller Wert der Außentemperatur.

Ausgänge

Name	Typ	Beschreibung
fSpHtg	REAL	Errechneter Sollwert der Zulufttemperatur für den Erhitzer.
fSpCol	REAL	Errechneter Sollwert der Zulufttemperatur für den Kühler.
fSpErc	REAL	Errechneter Sollwert der Zulufttemperatur für die Energierückgewinnung.
bOn	BOOL	Der Ausgang zeigt das resultierende Ergebnis des Multiplexer *MuxCmd* an. Dieses wird innerhalb der Lüftungsanlage weiterverarbeitet und dient der Freigabe der Zulufttemperatur Sequenzregelung.

Ein- / Ausgänge

Name	Typ	Beschreibung
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregate Struktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregate Struktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
X1	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X1.
X2	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X2.
X3	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X3.
X4	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt X4.
Y1	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt Y1.
Y2_3	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt Y2/Y3.
Y4	FB_BA_AV_Op [▶ 207]	Eingabe des Wertes für den Stützpunkt Y4.
Sp	FB_BA_AV_Op [▶ 207]	Ausgabe des errechneten, einfachen Zulufttemperatur-Sollwertes. Dieser wird über die Ausgänge *fSpHtg*, *fSpCol* und *fSpErc* ausgegeben.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

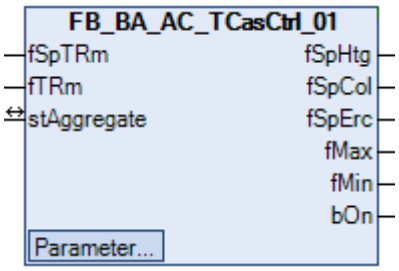
Variablen

Name	Typ	Beschreibung
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter PrioSwiVal [▶ 443] ermittelt anhand der Kommandostruktur *stCmd* den Modulationsbefehl für die Multiplexer *MuxCmd* und *MuxMdlt*.
MuxCmd	FB_BA_MMUX_B04 [▶ 438]	Der Multiplexer MuxCmd [▶ 438] ermittelt den aktuellen Schaltwert aus der Kommandierung des Prioritätenschalters *PrioSwiVal*. Das resultierende Ergebnis wird an das Binär-Ausgangs-Objekt *bOn* übermittelt.
Scale	FB_BA_Chrct04 [▶ 469]	Der Funktionsbaustein berechnet den Zulufttemperatur-Sollwert in Abhängigkeit der Außentemperatur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.4.1.3 FB_BA_AC_TCasCtrl_01

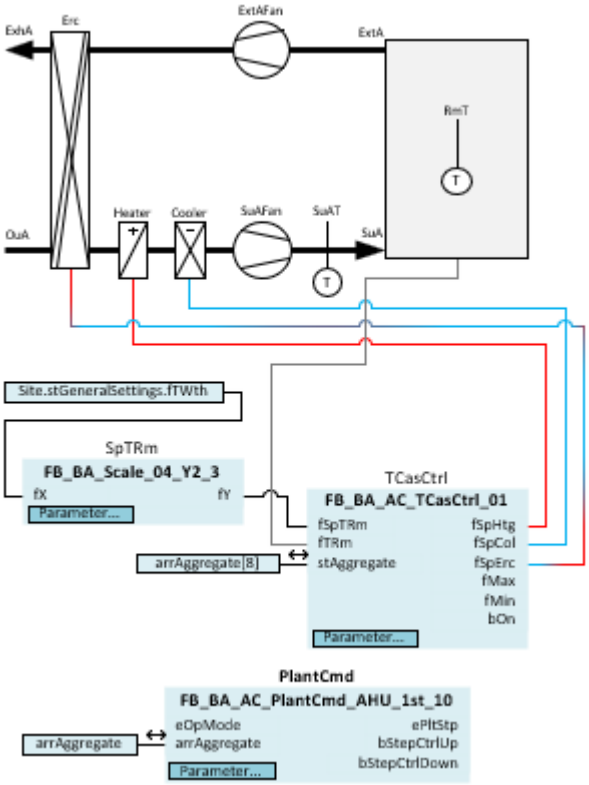


Das Template dient der Raumtemperaturregelung mittels Raum-Zuluft-Kaskade.

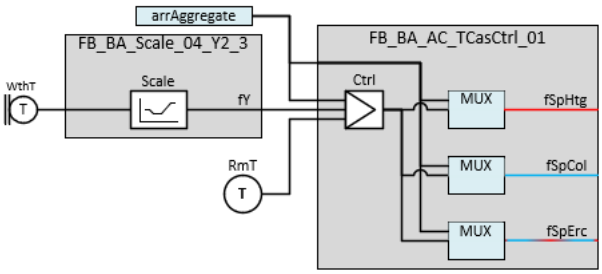
Es besteht aus einem Führungsregler zur Berechnung des Zulufttemperatur-Sollwertes für die Aggregate Erhitzer, Kühler und Energierückgewinnung.

Die Freigabe der Zulufttemperatur Sequenzregelung erfolgt mit der Variablen *bOn* durch die Auswertung des Empfangsbausteins der Schrittkettensteuerung *Aggregate*.

Prinzipschema01

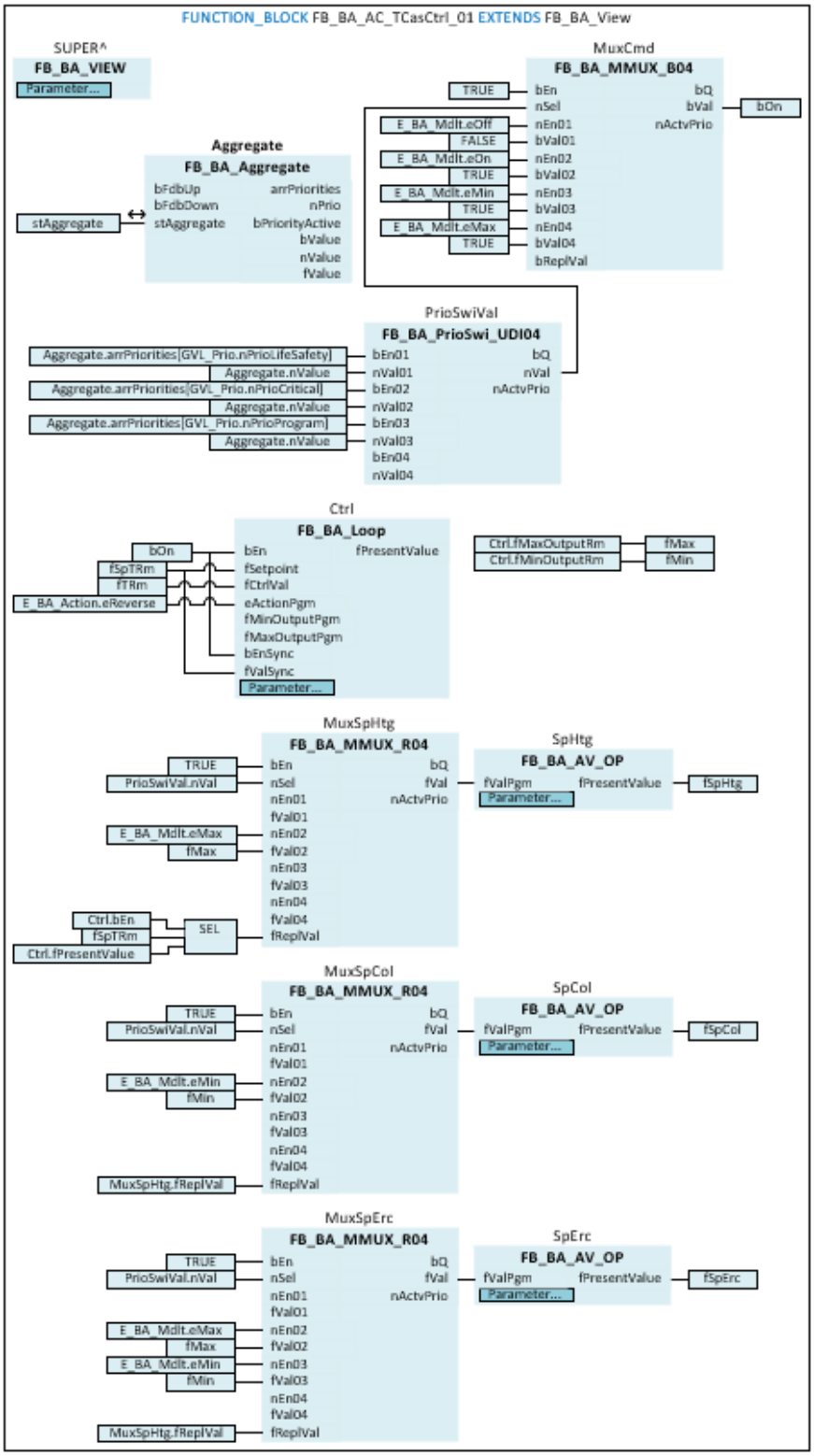


Prinzipschema02



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_TCasCtrl_01 EXTENDS FB_BA_View
VAR_INPUT
  fSpTRm         : REAL;
  fTRm           : REAL;
END_VAR
VAR_OUTPUT
  fSpHtg         : REAL;
  fSpCol         : REAL;
  fSpErc         : REAL;
  fMax           : REAL;
  fMin           : REAL;
```

```
  bOn           : BOOL;
END_VAR
VAR_IN_OUT
  stAggregate   : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  SpHtg         : FB_BA_AV_Op;
  SpCol         : FB_BA_AV_Op;
  SpErc         : FB_BA_AV_Op;
  Ctrl          : FB_BA_Loop;
  Aggregate     : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiVal    : FB_BA_PrioSwi_UDI04;
  MuxCmd        : FB_BA_MMUX_B04;
  MuxSpHtg      : FB_BA_MMUX_R04;
  MuxSpCol      : FB_BA_MMUX_R04;
  MuxSpErc      : FB_BA_MMUX_R04;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
fSpTRm	REAL	Eingangsvariable für den Sollwert Raumtemperatur.
fTRm	REAL	Eingangsvariable, an der die Raumtemperatur angelegt wird. Die Raumtemperatur ist die Regelgröße des PID-Führungsreglers *Ctrl*. Ist keine Raumtemperatur vorhanden, so kann die Ablufttemperatur einer Lüftungsanlage als Regelgröße genommen werden.

Ausgänge

Name	Typ	Beschreibung
fSpHtg	REAL	Errechneter Sollwert der Zulufttemperatur für den Erhitzer.
fSpCol	REAL	Errechneter Sollwert der Zulufttemperatur für den Kühler.
fSpErc	REAL	Errechneter Sollwert der Zulufttemperatur für die Energierückgewinnung.
fMax	REAL	Oberer Wert der Regler-Ausgangsbegrenzung.
fMin	REAL	Unterer Wert der Regler-Ausgangsbegrenzung.
bOn	BOOL	Der Ausgang zeigt das resultierende Ergebnis des Multiplexer *MuxCmd* an. Dieses wird innerhalb der Lüftungsanlage weiterverarbeitet und dient der Freigabe der Zulufttemperatur Sequenzregelung.

Ein- / Ausgänge

Name	Typ	Beschreibung
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregate Struktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregate Struktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
SpHtg	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Raumtemperatur-Sollwertes Heizen.
SpCol	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Raumtemperatur-Sollwertes Kühlen.
SpErc	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Zulufttemperatur-Sollwertes Energierückgewinnung.
Ctrl	FB_BA_Loop [▶ 225]	Das LOOP-Objekt dient als Führungsregler für eine Raumtemperaturregelung mittels Raum-Zuluft-Kaskade. Es liefert den Zulufttemperatur-Sollwert.
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

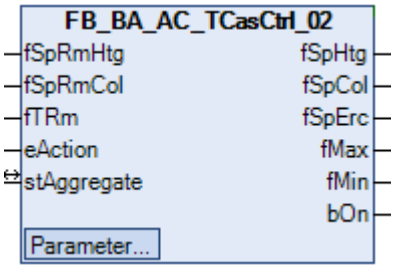
Variablen

Name	Typ	Beschreibung
PrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter ermittelt anhand der Kommandostruktur *stCmd* den Modulationsbefehl für den Multiplexer *MuxCmd*.
MuxCmd	FB_BA_MMUX_B04 [▶ 438]	Der Multiplexer ermittelt den aktuellen Schaltbefehl aus dem Modulationsbefehl des Prioritätenschalters *PrioSwiVal*. Das resultierende Ergebnis wird an den Ausgang *bOn* übermittelt. Der Multiplexer definiert die Freigabebedingungen des PID-Führungsreglers *Ctrl*.
MuxSpHtg	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer definiert den Sollwert Heizen *fSpHtg* in Abhängigkeit des Prioritätenschalters *PrioSwiVal*.
MuxSpCol	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer definiert den Sollwert Kühlen *fSpCol* in Abhängigkeit des Prioritätenschalters *PrioSwiVal*.
MuxSpErc	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer definiert den Sollwert Energierückgewinnung *fSpErc* in Abhängigkeit des Prioritätenschalters *PrioSwiVal*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.4.1.4 FB_BA_AC_TCasCtrl_02



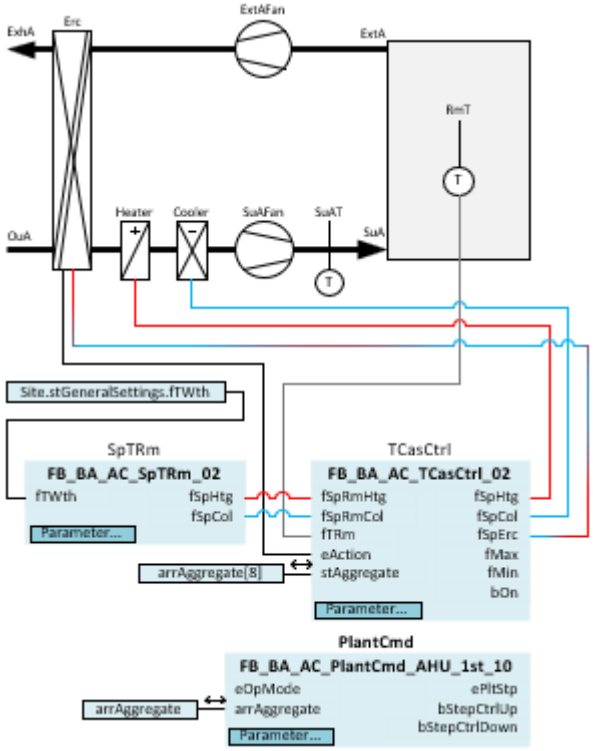
Das Template dient der Raumtemperaturregelung mittels Raum-Zuluft-Kaskade.

Es besteht aus zwei Führungsreglern zur Berechnung der Zuluft Temperatursollwerte für die Aggregate Erhitzer und Kühler. Jeder der beiden Führungsregler erhält einen separaten Sollwert. Der Sollwert des Führungsreglers für die Funktion des Heizens liegt unterhalb dem des Sollwertes des Führungsreglers für das Kühlen. Zwischen den beiden Sollwerten ergibt sich eine energieneutrale Zone.

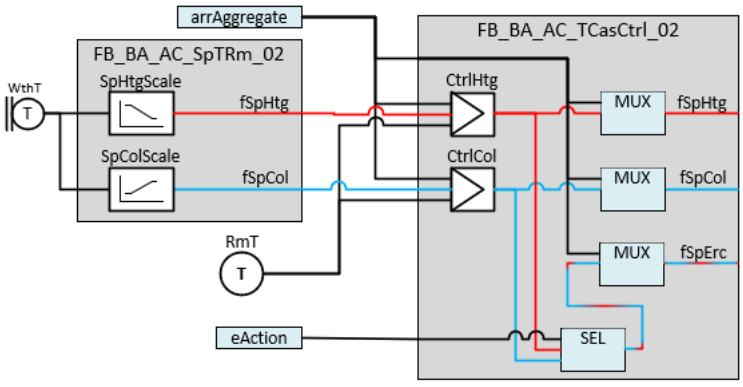
Zusätzlich ermittelt es den Zulufttemperatur Sollwert für die Energierückgewinnung abhängig von deren Wirksinn.

Die Freigabe der Zulufttemperatur Sequenzregelung erfolgt mit der Variablen *bOn* durch die Auswertung des Empfangsbausteins der Schrittkettensteuerung *Aggregate*.

Prinzipschema01

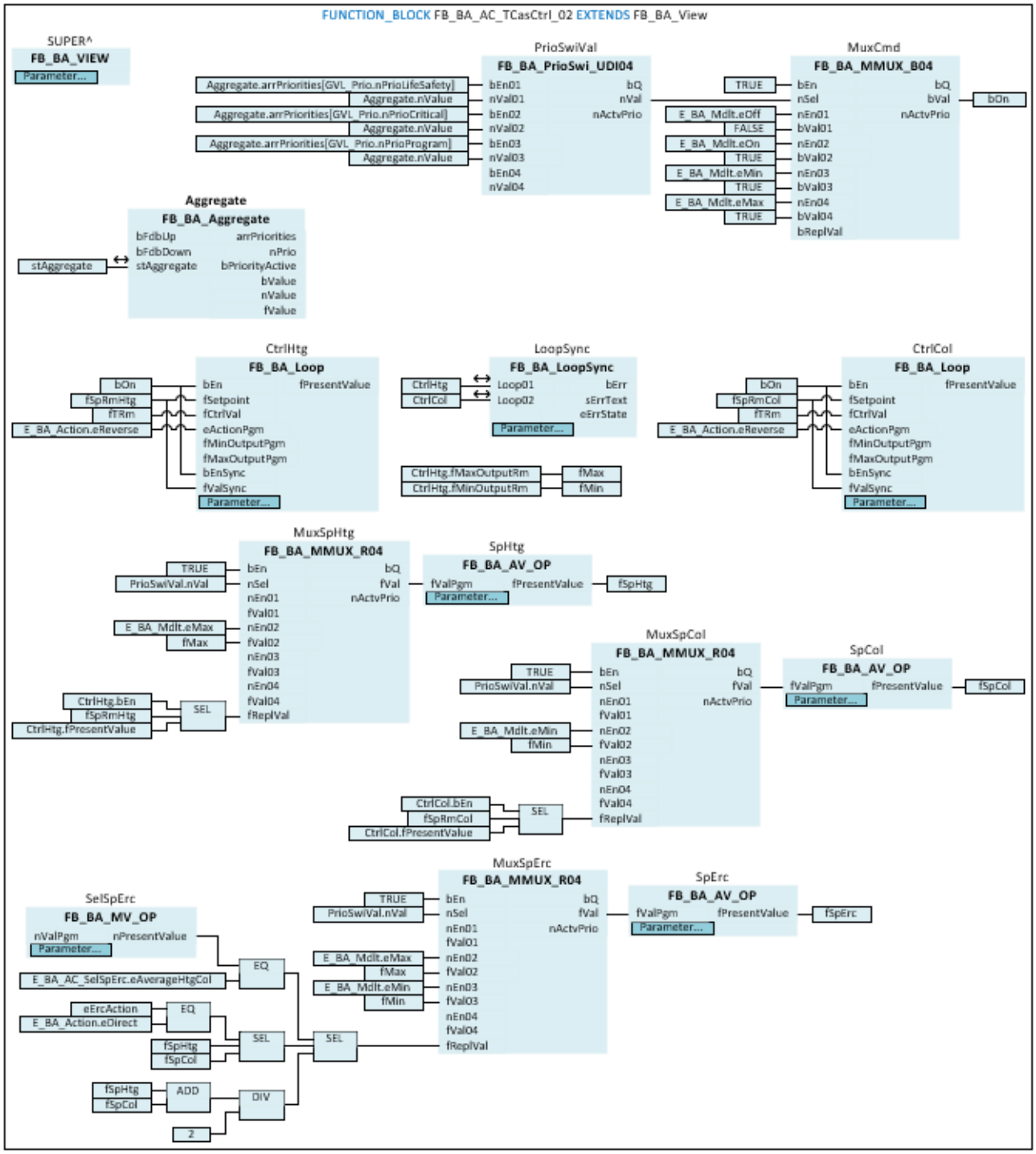


Prinzipschema02



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AC_TCasCtrl_02 EXTENDS FB_BA_View
VAR_INPUT
  fSpRmHtg       : REAL;
  fSpRmCol       : REAL;
  fTRm           : REAL;
  eAction        : E_BA_Action := E_BA_Action.eDirect;
END_VAR
VAR_OUTPUT
  fSpHtg         : REAL;
  fSpCol         : REAL;
  fSpErc         : REAL;
  fMax           : REAL;
  fMin           : REAL;
  bOn            : BOOL;
END_VAR
VAR_IN_OUT
```

```
  stAggregate    : ST_BA_Aggregate;
END_VAR
VAR_INPUT CONSTANT
  SpHtg          : FB_BA_AV_Op;
  SpCol          : FB_BA_AV_Op;
  SpErc          : FB_BA_AV_Op;
  SelSpErc       : FB_BA_MV_Op;
  CtrlHtg        : FB_BA_Loop;
  CtrlCol        : FB_BA_Loop;
  LoopSync       : FB_BA_LoopSync;
  Aggregate      : FB_BA_Aggregate;
END_VAR
VAR
  PrioSwiVal     : FB_BA_PrioSwi_UDI04;
  MuxCmd         : FB_BA_MMUX_B04;
  MuxSpHtg       : FB_BA_MMUX_R04;
  MuxSpCol       : FB_BA_MMUX_R04;
  MuxSpErc       : FB_BA_MMUX_R04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fSpRmHtg	REAL	Eingangsvariable für den Sollwert Raumtemperatur Heizen [▶ 806].
fSpRmCol	REAL	Eingangsvariable für den Sollwert Raumtemperatur Kühlen. [▶ 806]
fTRm	REAL	Eingangsvariable, an der die Raumtemperatur angelegt wird. Die Raumtemperatur ist die Regelgröße des PID-Führungsreglers *Ctrl*. Ist keine Raumtemperatur vorhanden, so kann die Ablufttemperatur einer Lüftungsanlage als Regelgröße genommen werden.
eAction	E_BA_Action	Eingangsvariable, an der der Wirksinn der Energierückgewinnung angelegt wird. In Abhängigkeit des Wirksinns wird der Sollwert für die Energierückgewinnung bestimmt.

Ausgänge

Name	Typ	Beschreibung
fSpHtg	REAL	Errechneter Sollwert der Zulufttemperatur für den Erhitzer.
fSpCol	REAL	Errechneter Sollwert der Zulufttemperatur für den Kühler.
fSpErc	REAL	Errechneter Sollwert der Zulufttemperatur für die Energierückgewinnung.
fMax	REAL	Oberer Wert der Regler-Ausgangsbegrenzung.
fMin	REAL	Unterer Wert der Regler-Ausgangsbegrenzung.
bOn	BOOL	Der Ausgang zeigt das resultierende Ergebnis des Multiplexer *MuxCmd* an. Dieses wird innerhalb der Lüftungsanlage weiterverarbeitet und dient der Freigabe der Zulufttemperatur Sequenzregelung.

Ein- / Ausgänge

Name	Typ	Beschreibung
stAggregate	ST_BA_Aggregate [▶ 283]	Bidirektionale Aggregate Struktur über welche die Freigaben und Schaltwerte an den Auswertebaustein der Schrittkettensteuerung *Aggregate* übermittelt werden. Die Aggregate Struktur überträgt die aufgenommenen Rückmeldungen hoch und runter an die zentrale Anlagensteuerung.

Eingänge CONSTANT

Name	Typ	Beschreibung
SpHtg	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Raumtemperatur-Sollwertes Heizen.
SpCol	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Raumtemperatur-Sollwertes Kühlen.
SpErc	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige des errechneten Zulufttemperatur-Sollwertes Energierückgewinnung.
SelSpErc	FB_BA_MV_Op [▶ 244]	Das Multistate-Werte-Objekt dient der Auswahl der Strategie Sollwert Energierückgewinnung. E_BA_AC_SelSpErc.eAction [▶ 712]: = 1: Wirksinnabhängig, definiert durch den Eingang *eErcAction*. E_BA_AC_SelSpErc.eAverageHtgCol [▶ 712]:= 2: Mittelwert aus den Eingangsvariablen *fSpHtg* und *fSpCol*.
CtrlHtg	FB_BA_Loop [▶ 225]	Das LOOP-Objekt dient als Führungsregler für eine Raumtemperaturregelung mittels Raum-Zuluft-Kaskade. Es liefert den Zulufttemperatur-Sollwert für das Heizen.
CtrlCol	FB_BA_Loop [▶ 225]	Das LOOP-Objekt dient als Führungsregler für eine Raumtemperaturregelung mittels Raum-Zuluft-Kaskade. Es liefert den Zulufttemperatur-Sollwert für das Kühlen.
LoopSync	FB_BA_LoopSync [▶ 169]	Der Funktionsbaustein führt eine Parameter-Synchronisation der beiden Führungsregler *CtrlHtg/CtrlCol* durch
Aggregate	FB_BA_Aggregate [▶ 489]	Der Aggregatebaustein repräsentiert den Empfangsbaustein einer Schrittkettensteuerung. Innerhalb des Funktionsbausteins wird die Aggregatestruktur *stAggregate* ausgewertet und anhand der Ein- und Ausgänge in das Template integriert.

Variablen

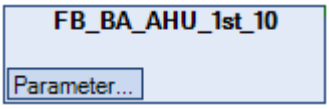
Name	Typ	Beschreibung
PrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter ermittelt anhand der Kommandostruktur *stCmd* den Modulationsbefehl für den Multiplexer *MuxCmd*.
MuxCmd	FB_BA_MMUX_B04 [▶ 438]	Der Multiplexer ermittelt den aktuellen Schaltbefehl aus dem Modulationsbefehl des Prioritätenschalters *PrioSwiVal*. Das resultierende Ergebnis wird an den Ausgang *bOn* übermittelt. Der Multiplexer definiert die Freigabebedingungen des PID-Führungsreglers *Ctrl*.
MuxSpHtg	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer definiert den Sollwert Heizen *fSpHtg* in Abhängigkeit des Prioritätenschalters *PrioSwiVal*.
MuxSpCol	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer definiert den Sollwert Kühlen *fSpCol* in Abhängigkeit des Prioritätenschalters *PrioSwiVal*.
MuxSpErc	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer definiert den Sollwert Energierückgewinnung *fSpErc* in Abhängigkeit des Prioritätenschalters *PrioSwiVal*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.1.5 Plants

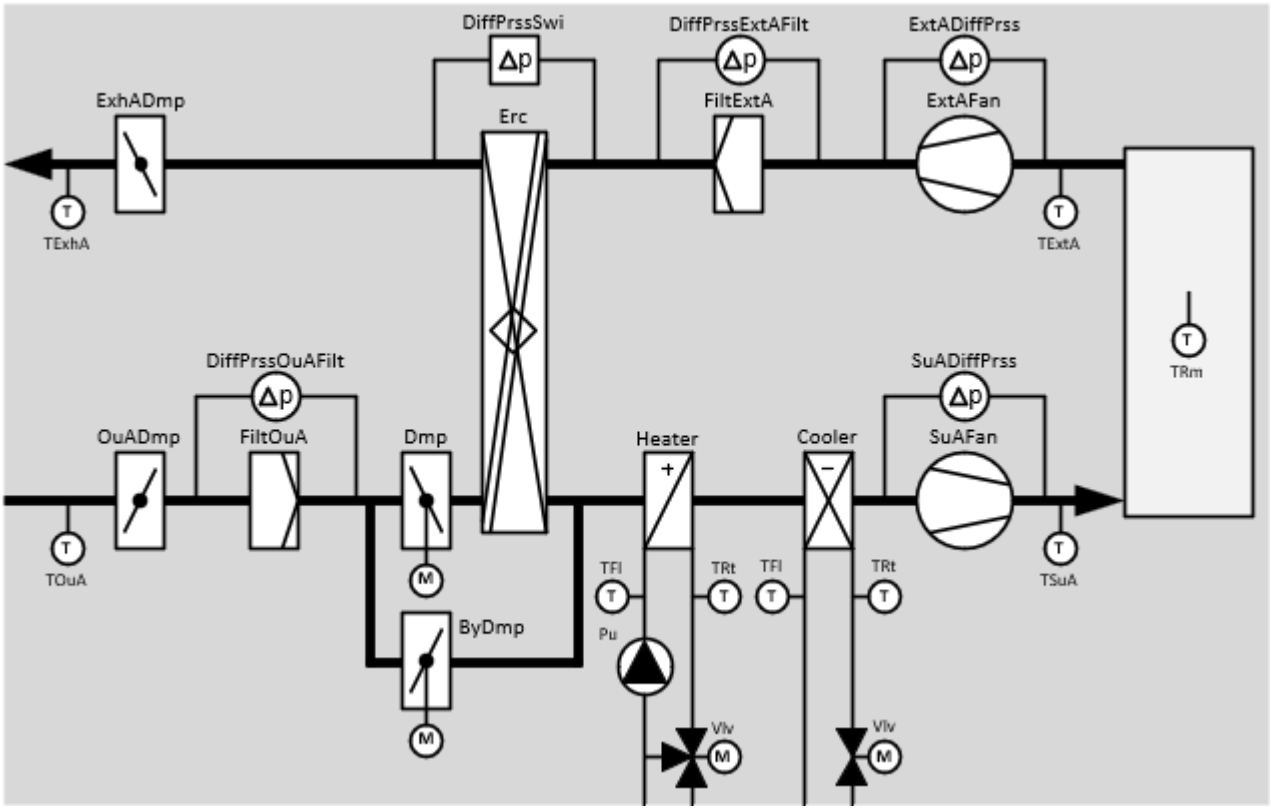
6.1.4.2.1.1.5.1 FB_BA_AHU_1st_10



Template einer raumlufttechnischen Anlage.

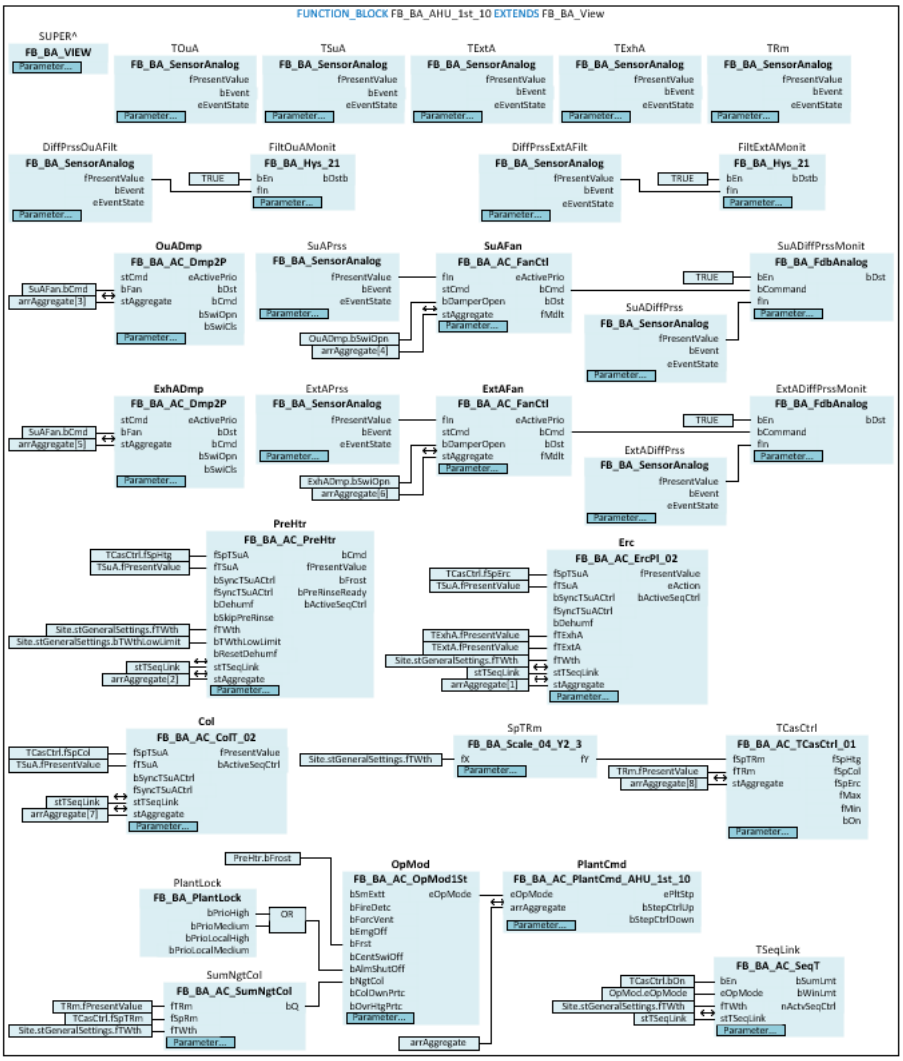
Die wesentlichen Bestandteile des Templates sind:

- Anlagensteuerungs- und Startprogramm
- Zulufttemperatur-Sequenzregelung
- Raumtemperatur-Kaskadenregelung
- Sommernachtkühlung
- Druckregelung des Zu- und Abluftventilators
- Thermische Luftbehandlung durch Lufterhitzer, Kühler und Energierückgewinnung mittels Plattenwärmetauscher
- Außen- und Fortluftklappe mit einem Federrücklaufantrieb und Endlagenkontrolle
- Außen- und Abluftfilter mit analoger Differenzdrucküberwachung

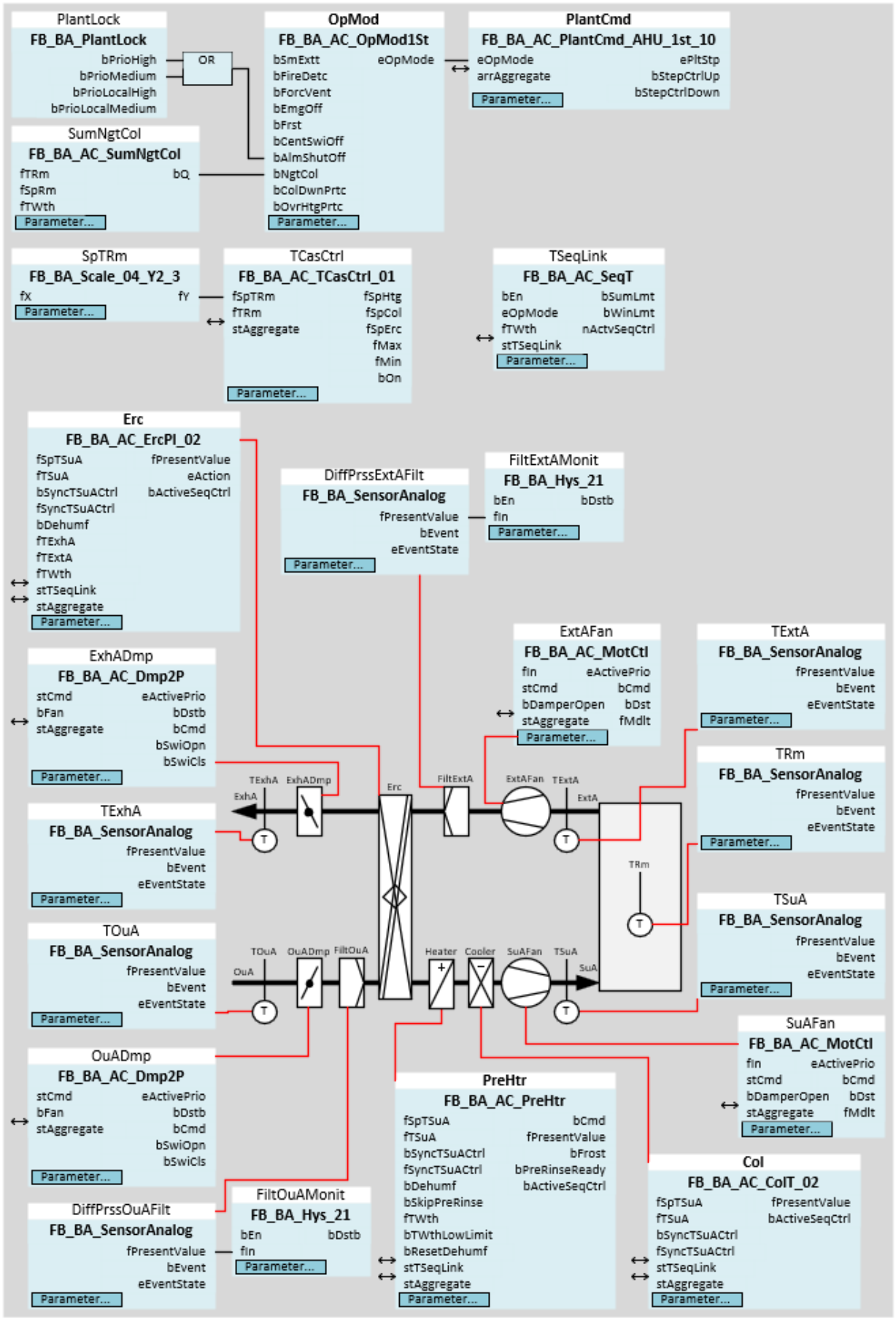


Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



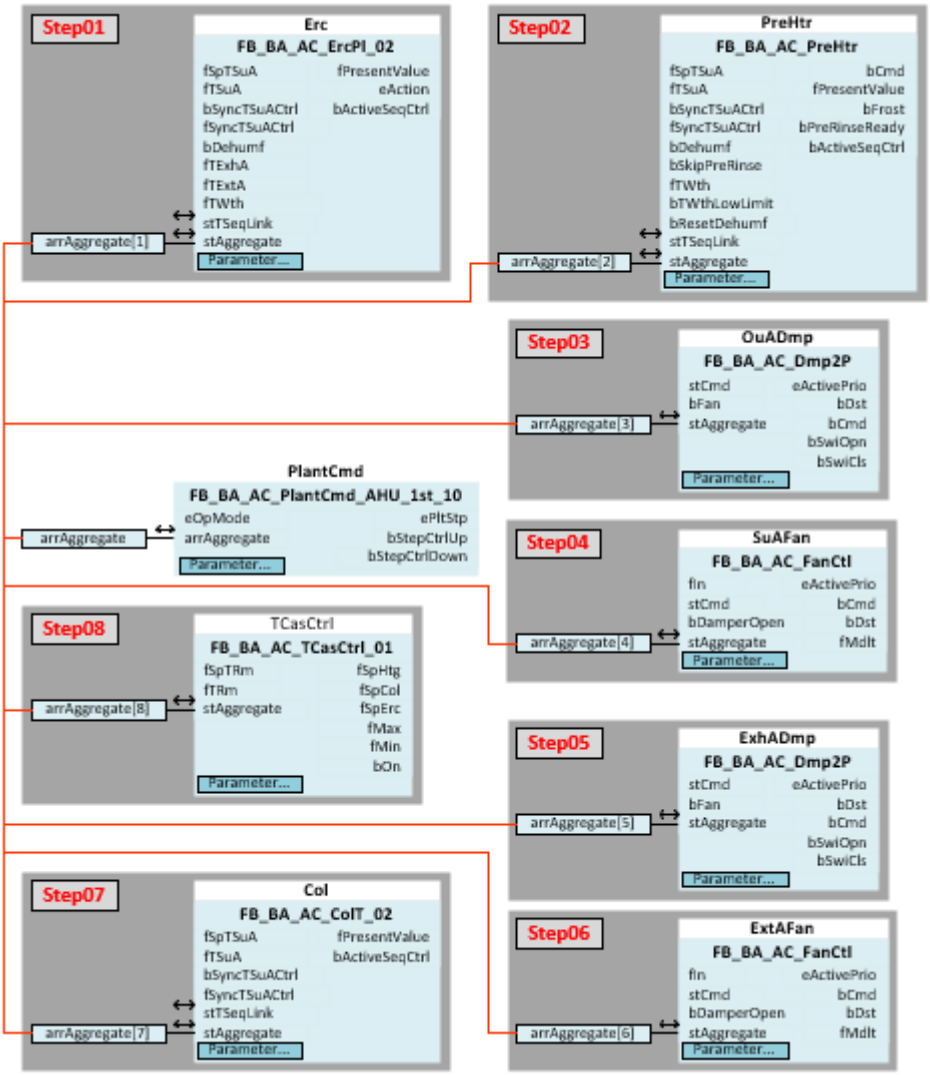
Anlagenschema



Einschaltreihenfolge der Aggregate der Anlage

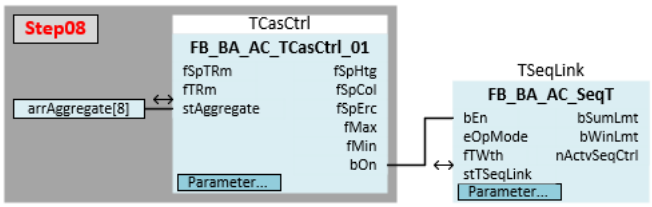
Innerhalb der Anlagenkommandierung *PlantCmd* steuert eine Schrittkettensteuerung sequenziell hintereinander das Ein- respektive Ausschalten der Aggregate. Dabei werden beim Weiterschalten von Aggregat zu Aggregat vorgegebene Verzögerungszeiten und Rückmeldungen der Aggregate berücksichtigt.

Die Reihenfolge der Aggregate wird durch die Ordnungszahl der einzelnen Elemente des bidirektionalen Array *arrAggregate* festgelegt.



Schritt	Stufen des Anlagenstarts	Aggregat	Funktionsbausteine
00	E_BA_AC_PlantStep01.eOff	Anlage Aus	
01	E_BA_AC_PlantStep01.eErc	Energierückgewinnung	*Erc*
02	E_BA_AC_PlantStep01.ePreRinse	Vorspülen des Warmwasserlufterhitzer	*PreHtr*
03	E_BA_AC_PlantStep01.eDamperOuA	Außenluftklappe	*OuADmp*
04	E_BA_AC_PlantStep01.eFanSupplyAir	Zuluftventilator	*SuAFan*
05	E_BA_AC_PlantStep01.eDamperExhA	Fortluftklappe	*ExhADmp*
06	E_BA_AC_PlantStep01.eFanExtractAir	Abluftventilator	*ExtAFan*
07	E_BA_AC_PlantStep01.eCooler	Kaltwasser-Luftkühler	*Col*
08	E_BA_AC_PlantStep01.eOn	Raum-Zuluft-Kaskade, Anlage Ein	*TCasCtrl*

Ist die Anlage in Schritt 08 *E_BA_AC_PlantStep01.eOn* angekommen, so wird mit Einschaltung der Kaskadenregelung *TCasCtrl* die Zulufttemperatur Sequenzregelung *TSeqLink* freigegeben.



Anfahren der Anlage bei kalter Witterung

Beim Einschalten der Anlage bei kalter Witterung ist in der Schrittkettensteuerung der Anlagenkommandierung *PlantCmd* der Anlagenschritt Vorspülen des Vorerhitzers *E_BA_AC_PlantStep01.ePreRinse* vorgesehen. In diesem Schritt wird die Regelung der Rücklauftemperatur des Warmwasserlufterhitzers aktiviert. Dieser Vorgang soll das Einfrieren des Warmwasserlufterhitzer während des Anlagenstarts verhindern.

Störabschaltungen

Die Anlage schaltet ab bei:

- Brandalarm
- Frostalarm

Bei Frostalarm verbleibt auch bei abgeschalteter Anlage der Warmwasserlufterhitzer *PreHtr* in Stellung Frostschutz!

- Störungen der Ventilatoren (Rückmeldeüberwachung, thermoelektrische Überlast, Wartungsschalter)
- Fehlstellungen der Außen- und Fortluftklappe.
- Filterverstopfungen

Die folgende Tabelle listet die ereignisfähigen Objekte des Templates auf, welche relevante Störungen am *PlantLock* auslösen können. Die Parametrierung dieser Ereignisse ist im *FB_init* zu finden.

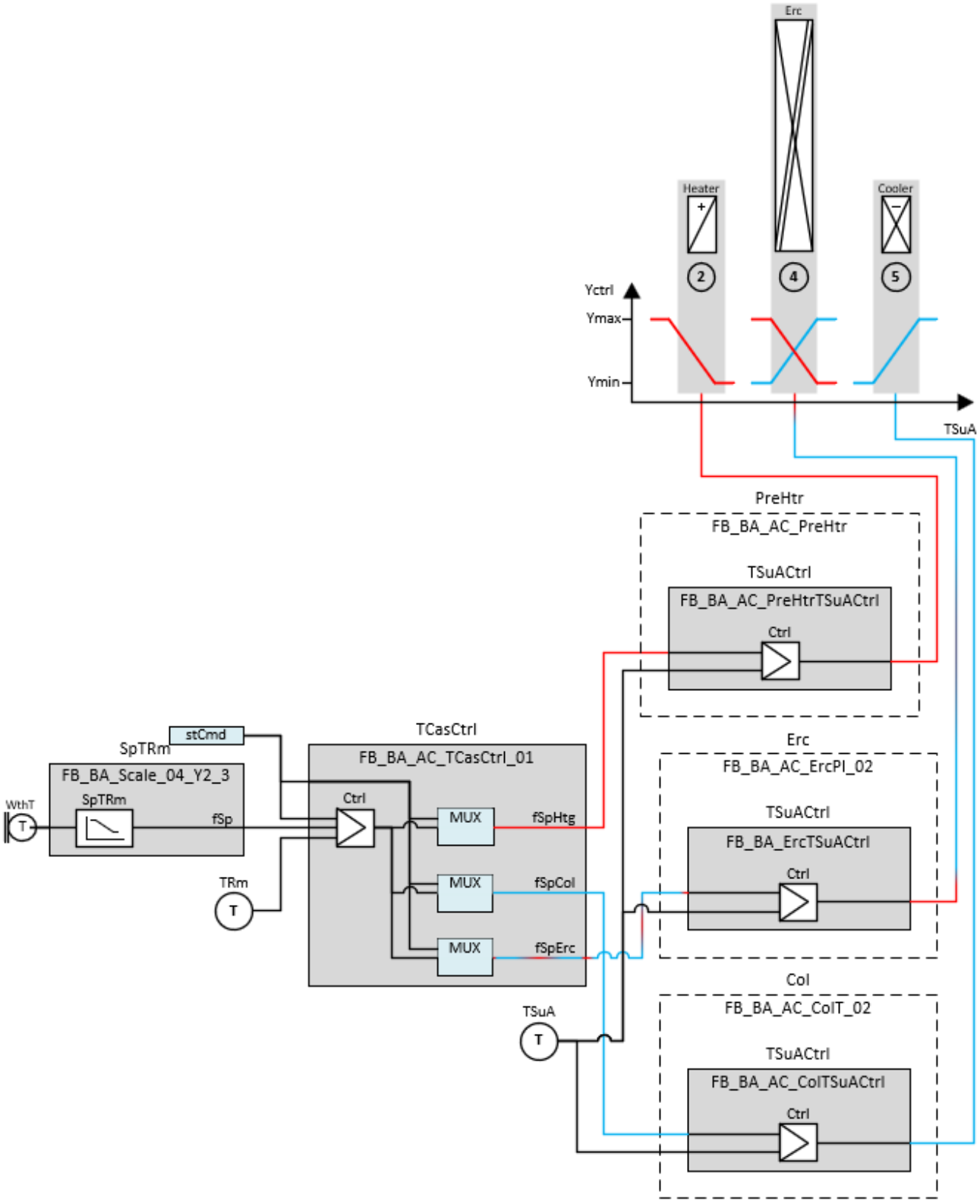
Funktionsblock	eEnPlantLock	Funktionsblock	bPrioHigh	bPrioMedium
TSuA.MV	E_BA_LockPriority.eMedium	PlantLock		x
FiltOuAMonit.DstMainAlarm	E_BA_LockPriority.eMedium			x
OuADmp.MonitOpen.Dst	E_BA_LockPriority.eMedium			x
SuAFan.Dst	E_BA_LockPriority.eMedium			x
SuAFan.ThOvrld	E_BA_LockPriority.eMedium			x
SuAFan.MntnSwi	E_BA_LockPriority.eMedium			x
SuADiffPrssMonit.Dst	E_BA_LockPriority.eMedium			x
ExhADmp.MonitOpen.Dst	E_BA_LockPriority.eMedium			x
ExtAFan.Dst	E_BA_LockPriority.eMedium			x
ExtAFan.ThOvrld	E_BA_LockPriority.eMedium			x
ExtAFan.MntnSwi	E_BA_LockPriority.eMedium			x
ExtADiffPrssMonit.Dst	E_BA_LockPriority.eMedium			x
PreHtr.TRt.MV	E_BA_LockPriority.eMedium			x
PreHtr.TFrost.MV	E_BA_LockPriority.eMedium			x
PreHtr.FrostThermostat.Input	E_BA_LockPriority.eMedium			x
PreHtr.Pu.Dst	E_BA_LockPriority.eMedium			x

Die Parametrierung der Lockpriorität der ereignisfähigen Objekte ist im *FB_init* dieses Templates zu finden.

Temperaturregelung

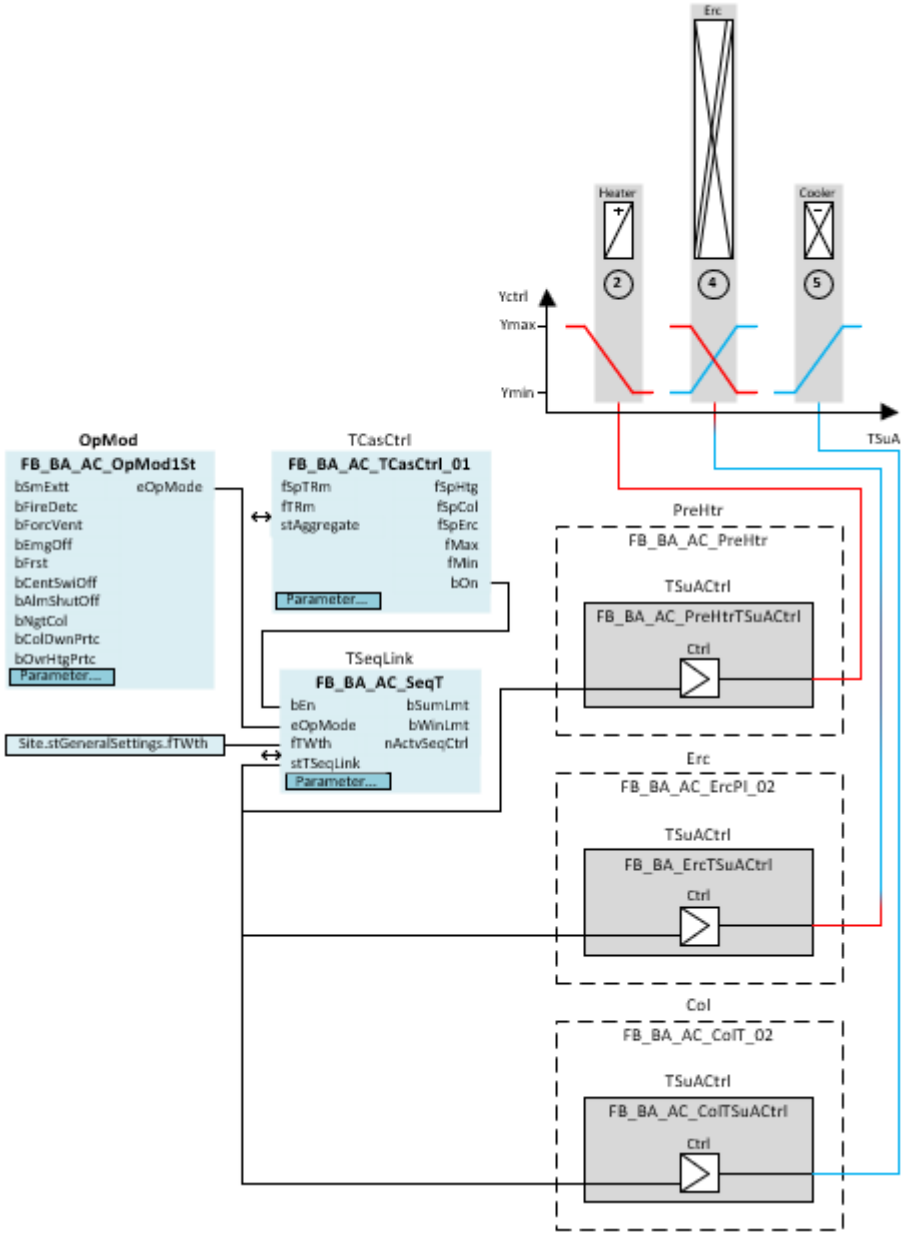
In diesem Anlagentemplate wird eine einfache Raumluft-Zuluft-Kaskade als Regelstrategie realisiert.

In den Templates *SpTRm* / *TCasCtrl* werden die Sollwerte für die Anlage erzeugt und an die Sequenzregler innerhalb der Templates *PreHtr*, *Col* und *Erc* weitergeleitet.



Sequenzregelung

Die Zahl der Sequenzregler in der Regelsequenz variiert und hängt ab von der Zahl der Aggregate in der Lüftungsanlage. Links in der Regelsequenz werden die PID-Sequenzregler angeordnet, welche zur Erhöhung der Zulufttemperatur beitragen. Rechts befinden sich die Regler der Aggregate, welche die Zulufttemperatur herunterkühlen.
Es können mehre Heiz- oder auch mehrere Kühlaggregate in die Regelsequenz eingebunden werden.
In der Sequenzregelung ist immer nur ein PID-Sequenzregler aktiv. Alle anderen verharren bei dem Wert ihrer maximalen oder minimalen Stellgröße.



Mit Erhalt der Freigabe *bOn* aus der Kaskadenregelung *TCasCtrl* wird die Sequenzregelung *TSeqLink* freigegeben.
Mit welchem Startsequenzregler die Anlage anläuft, wird in Abhängigkeit der Witterungstemperatur Site.stGeneralSettings.fTWth [▶ 1135] oder der Betriebsart *OpMod* festgelegt.

Über die Datenstruktur *stTSeqLink* bekommen die Sequenzregler der Aggregate Lufterhitzer *PreHtr*, Luftkühler *Col* und Energierückgewinnung *Erc* die Anlagenfreigabe und die Nummer des Startreglers übertragen.
Die Lüftungsanlage in dem Sequenzschema ist mit einem Lufterhitzer, einer Energierückgewinnung und einem Luftkühler ausgerüstet.
Zur stetigen Regelung der Zulufttemperatur *TSuA* bei sehr niedrigen Außentemperaturen im Winter bis hin zu hohen Außentemperaturen im Sommer werden alle Aggregate in eine Sequenzregelung eingebunden.

Links in der Sequenz wird mit dem Stellbereich des Luftherhitzers *PreHtr* begonnen. Bei niedrigen Außentemperaturen wird die Leistung des Luftererhitzers so lange erhöht, bis die Zulufttemperatur *TSuA* ihren Sollwert erreicht hat.

Steigt die Außentemperatur *TOuA* an, dann reduziert der Sequenzregler *TSuACtrl* des Lufterhitzers *PreHtr* seine Stellgröße so lange bis er an die untere Grenze seines Stellbereiches angekommen ist.

Wird der Sollwert der Zulufttemperatur *TCasCtrl* nach wie vor nicht erreicht, dann schaltet der Sequenzregler *TSuACtrl* des Lufterhitzers PreHtr weiter an den Sequenzregler *TSuACtrl* der Energierückgewinnung *Erc*. Das Verhalten der Energierückgewinnung *Erc* hängt vom Verhältnis der Witterungstemperatur Site.stGeneralSettings.fTWth [▶ 1135] und der Ablufttemperatur *TExtA* ab.

Ist die Witterungstemperatur Site.stGeneralSettings.fTWth [▶ 1135] kleiner als die Ablufttemperatur *TExtA*, dann gewinnt Energierückgewinnung Wärmeenergie aus der Abluft zurück und überträgt diese auf die Außenluft. In der Regelsequenz ist dann die rote Kennlinie aktiviert.

Der Wirksinn des Sequenzreglers *TSuACtrl* ist damit indirekt (Heizbetrieb).

Im Sommer ist die Ablufttemperatur häufig niedriger als die Witterungstemperatur. Der Energiefluss der Energierückgewinnung *Erc* ist dann invertiert. Die Wärme der Außenluft überträgt sich auf die Fortluft *TExhA* und die Zulufttemperatur wird dadurch verringert. In diesem Betriebsfall folgt die Sequenz der blauen Kennlinie. Der Wirksinn des Sequenzreglers *TSuACtrl* ist damit direkt (Kühlbetrieb).

Bei hohen Außentemperaturen, wenn die Rückgewinnungsleistung der *Erc* nicht ausreicht, um die Außenluft *OuA* weit genug herunterzukühlen, schaltet der Sequenzregler *TSuACtrl* der Energierückgewinnung *Erc* weiter an den Sequenzregler *TSuACtrl* des Kühlers *Col*. Dieser erhöht die Leistung des Kühlers so lange bis der Sollwert der Zulufttemperatur erreicht wird.

Filterüberwachung

Der Außen- und der Abluftfilter werden über die Funktionsbausteine *FiltOuAMonit* und *FiltExtAMonit* mit den analogen Differenzdruckfühlern *DiffPrssOuAFilt* / *DiffPrssExtAFilt* überwacht. Mit Erhalt der Freigabe *bOn* aus der Kaskadenregelung *TCasCtrl* werden die Filterüberwachungen freigegeben.

Nachtkühlung

Über den Funktionsbaustein **SumNgtCol** kann die Nachtkühlung gesteuert werden.

Syntax

```
FUNCTION_BLOCK FB_BA_AHU_1st_10 EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  TOuA                    : FB_BA_SensorAnalog;
  TExhA                   : FB_BA_SensorAnalog;
  TSuA                    : FB_BA_SensorAnalog;
  TExtA                   : FB_BA_SensorAnalog;
  TRm                     : FB_BA_SensorAnalog;

  DiffPrssOuAFilt         : FB_BA_SensorAnalog;
  FiltOuAMonit            : FB_BA_Hys_21;
  DiffPrssExtAFilt        : FB_BA_SensorAnalog;
  FiltExtAMonit           : FB_BA_Hys_21;

  OuADmp                  : FB_BA_AC_Dmp2P;
  SuAPrss                 : FB_BA_SensorAnalog;
  SuADiffPrss             : FB_BA_SensorAnalog;
  SuAFan                  : FB_BA_AC_FanCtl;
  SuADiffPrssMonit        : FB_BA_FdbAnalog;

  ExhADmp                 : FB_BA_AC_Dmp2P;
  ExtAPrss                : FB_BA_SensorAnalog;
  ExtADiffPrss            : FB_BA_SensorAnalog;
  ExtAFan                 : FB_BA_AC_FanCtl;
  ExtADiffPrssMonit       : FB_BA_FdbAnalog;

  PreHtr                  : FB_BA_AC_PreHtr;
  Erc                     : FB_BA_AC_ErcPl_02;
  Col                     : FB_BA_AC_ColT_02;

  SpTRm                   : FB_BA_Scale_04_Y2_3;
  TCasCtrl                : FB_BA_AC_TCasCtrl_01;
```

```
  SumNgtCol               : FB_BA_AC_SumNgtCol;
  OpMod                   : FB_BA_AC_OpMod1St;
  EnPrio                  : FB_BA_AC_EnPrio;

  PlantLock               : FB_BA_PlantLock;

  PlantCmd                : FB_BA_AC_PlantCmd_AHU_1st_10;

  TSeqLink                : FB_BA_AC_SeqT;
  stTSeqLink              : ST_BA_SeqLink;

  arrAggregate            : ARRAY [1..BA_Param.nMaxAggregate] OF ST_BA_Aggregate;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
TOuA	FB_BA_SensorAnalog [▶ 1106]	Repräsentiert die Außenlufttemperatur
TExhA	FB_BA_SensorAnalog [▶ 1106]	Repräsentiert die Fortlufttemperatur
TSuA	FB_BA_SensorAnalog [▶ 1106]	Repräsentiert die Zulufttemperatur
TExtA	FB_BA_SensorAnalog [▶ 1106]	Repräsentiert die Ablufttemperatur
TRm	FB_BA_SensorAnalog [▶ 1106]	Repräsentiert die Raumtemperatur
DiffPrssOuAFilt	FB_BA_SensorAnalog [▶ 1106]	Analoger Differenzdruck über den Außenluftfilter
FiltOuAMonit	FB_BA_Hys_21 [▶ 1028]	Analoge Überwachung des Außenluftfilters. Die Filterüberwachung verwendet den analogen Differenzdruck *DiffPrssOuAFilt* und kann sowohl einen Voralarm wie auch einen Hauptalarm auslösen.
DiffPrssExtAFilt	FB_BA_SensorAnalog [▶ 1106]	Analoger Differenzdruck über den Abluftfilter
FiltExtAMonit	FB_BA_Hys_21 [▶ 1028]	Analoge Überwachung des Abluftfilters. Die Filterüberwachung verwendet den analogen Differenzdruck *FiltExtAMonit* und kann sowohl einen Voralarm wie auch einen Hauptalarm auslösen.
OuADmp	FB_BA_AC_Dmp2P [▶ 732]	Ansteuerung der Außenluftklappe mit einem Federrücklaufantrieb und Endlagenkontrolle.
SuAPrss	FB_BA_SensorAnalog [▶ 1106]	Zuluftdruck
SuADiffPrss	FB_BA_SensorAnalog [▶ 1106]	Differenzdruck über dem Zuluftventilator
SuAFan	FB_BA_AC_FanCtl [▶ 756]	Ansteuerung und Regelung des drehzahlgesteuerten Zuluftventilators.
SuADiffPrssMonit	FB_BA_FdbAnalog [▶ 1034]	Analoge Differenzdrucküberwachung über dem Zuluftventilator. Die Differenzdrucküberwachung verwendet den analogen Differenzdruckfühler *SuADiffPrss* und kann einen anlagenabschaltenden Alarm auslösen.
ExhADmp	FB_BA_AC_Dmp2P [▶ 732]	Ansteuerung der Fortluftklappe mit einem Federrücklaufantrieb und Endlagenkontrolle.
ExtAPrss	FB_BA_SensorAnalog [▶ 1106]	Abluftdruck. Der Abluftdruck ist die Regelgröße des Druckreglers vom Abluftventilator *ExtAFan*.
ExtADiffPrss	FB_BA_SensorAnalog [▶ 1106]	Differenzdruck über dem Abluftventilator
ExtAFan	FB_BA_AC_FanCtl [▶ 756]	Ansteuerung und Regelung des drehzahlgesteuerten Abluftventilators.
ExtADiffPrssMonit	FB_BA_FdbAnalog [▶ 1034]	Analoge Differenzdrucküberwachung über dem Abluftventilator. Die Differenzdrucküberwachung verwendet den analogen Differenzdruckfühler *ExtADiffPrss* und kann einen anlagenabschaltenden Alarm auslösen.

Name	Typ	Beschreibung
PreHtr	FB_BA_AC_PreHtr [▶ 761]	Ansteuerung und Regelung eines Warmwasserlufterhitzers. Der Funktionsbaustein wird als Aggregat angesehen und ist aufgrund seines implementierten Sequenzreglers Teil der Temperatur- Sequenzregelung dieser Anlage.
Erc	FB_BA_AC_ErcPl_02 [▶ 746]	Ansteuerung und Regelung eines Plattenwärmetauschers für die Energierückgewinnung einer Lüftungsanlage. Der Funktionsbaustein wird als Aggregat angesehen und ist aufgrund seines implementierten Sequenzreglers Teil der Temperatur- Sequenzregelung dieser Anlage.
Col	FB_BA_AC_ColT_02 [▶ 729]	Ansteuerung und Regelung eines temperaturgesteuerten Kaltwasser-Luftkühlers. Der Funktionsbaustein wird als Aggregat angesehen und ist aufgrund seines implementierten Sequenzreglers Teil der Temperatur- Sequenzregelung dieser Anlage.
SpTRm	FB_BA_Scale_04_Y2_3 [▶ 1066]	Sollwertprogramm für eine Zulufttemperatur-Regelung mit einem Zulufttemperatur Sollwert inklusive Sommer- und Winterkompensation über eine Kennlinie.
TCasCtrl	FB_BA_AC_TCasCtrl_01 [▶ 814]	Raumtemperaturregelung mittels Raum-Zuluft-Kaskade. Es besteht aus einem Führungsregler zur Sollwertberechnung für das Heizen, Kühlen und der Energie-Rückgewinnung.
SumNgtCol	FB_BA_AC_SumNgtCol [▶ 796]	Steuerung der Nachtkühlung
OpMod	FB_BA_AC_OpMod1St [▶ 776]	Der Funktionsbaustein dient dem Einschalten der Anlage. Er repräsentiert die Auswahl der Anlagenbetriebsart, ein Zeitschaltprogramm und einen Anlagenwahlschalter.
EnPrio	FB_BA_AC_EnPrio [▶ 772]	Definiert anhand der Anlagenbetriebsart *OpMod* die Anlagenfreigabe für die Ansteuerung der Schrittkettensteuerung einer Anlage und die Freigabe der Prioritäten „Safety“, „Critcial“ und „Program“. Diese Freigaben werden an die Anlagenkommandierung *PlantCmd* übertragen.
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen der Anlage die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen lösen die Betriebsart Störung *E_BA_AC_OpMod01.eAlarm* im Betriebsartenprogramm *OpMod* aus. Dies hat zur Folge, dass die raumlufttechnische Anlage heruntergefahren wird. Eine Auflistung der anlagenabschaltenden Störungen ist unter Störabschaltungen zu finden.
PlantCmd	FB_BA_AC_PlantCmd_AHU_1st_10 [▶ 799]	Der Funktionsbaustein repräsentiert die Anlagenkommandierung der Aggregate und Funktionen dieser raumlufttechnischen Anlage. Die integrierte Schrittkettensteuerung legt fest in welcher Reihenfolge die einzelnen Aggregate der Anlage sequenziell hintereinander ein- respektive ausgeschaltet werden.
TSeqLink	FB_BA_AC_SeqT [▶ 791]	Repräsentiert das Starten und Steuern der Zulufttemperatur Sequenzregelung.
stTSeqLink	ST_BA_SeqLink [▶ 125]	Daten- und Befehlsstruktur zwischen den einzelnen Sequenzreglern der Temperaturregelsequenz *PreHtr*, *Col*, *Erc* und der Zulufttemperatur Sequenzregelung *TSeqLink*.

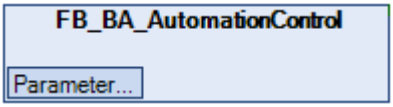
Name	Typ	Beschreibung
arrAggregate	ST_BA_Aggregate [▶ 283]	Das Array beinhaltet für jedes Aggregat eine bidirektionale Daten- und Kommandostruktur. Beim Hoch- respektive Zurückschalten der Schrittkettensteuerung wird, der für jedes Aggregat ermittelte Schaltbefehl über die bidirektionale Kommunikationsstruktur *arrAggregate* übergeben. Eine Rückmeldung der Aggregate für das Hoch- oder Runterschalten der Schrittkettensteuerung wird über die Kommunikationsstruktur an die Anlagensteuereinheit übertragen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.2 AutomationControl

6.1.4.2.1.2.1 FB_BA_AutomationControl



Das Template beinhaltet für eine Automationsstation relevante Basisfunktionen und Projektinformationen.

Im Sub-Template FB_BA_Device [▶ 843] werden Informationen zum Projekt und Diagnosemöglichkeiten angeboten. Eine Parametrierung des BACnet Device und allgemeiner BACnet-Parameter findet statt.

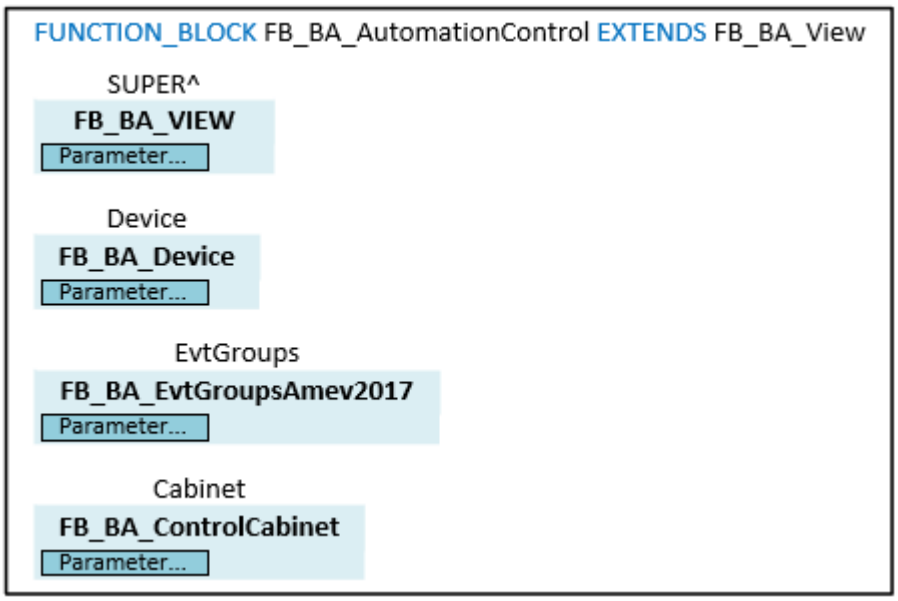
Unterhalb von *FB_BA_Device* gibt es zwei Möglichkeiten, um die BACnet-Systemzeit oder die lokale NT-Systemzeit des TwinCAT-Systems zu integrieren.

Das Sub-Template FB_BA_EvtGroupsAMEV2017 [▶ 857] stellt für das Absetzen der objektintegrierten Meldungen (intrinsic reporting) Ereignisklassen (Meldeklassen) bereit.

Das Sub-Template FB_BA_ControlCabinet [▶ 842] dient der Sammlung und Anzeige von Schaltschrankmeldungen. Eine Auswertung der Alarme der Automationsstation findet statt und ein zentrales Quittieren kann ausgelöst werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AutomationControl EXTENDS FB_BA_View
VAR_INPUT CONSTANT
     Device             : FB_BA_Device;
     EvtGroups          : FB_BA_EvtGroupsAMEV2017;
     Cabinet            : FB_BA_ControlCabinet;
END_VAR
```

VAR_INPUT CONSTANT

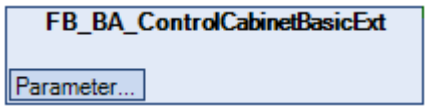
Name	Typ	Beschreibung
Device	FB_BA_Device [▶ 843]	Im Template werden Informationen zum Projekt und Diagnosemöglichkeiten angeboten. Eine Parametrierung des BACnet Device und allgemeiner BACnet Parameter findet statt. Unterhalb von FB_BA_Device gibt es zwei Möglichkeiten, um die BACnet-Systemzeit oder die lokale NT-Systemzeit des TwinCAT-Systems zu integrieren.
EvtGroups	FB_BA_EvtGroupsAMEV2017 [▶ 857]	Das Template stellt für das Absetzen der objektintegrierten Meldungen (intrinsic reporting) Ereignisklassen (Meldeklassen) bereit.
Cabinet	FB_BA_ControlCabinet [▶ 842]	Das Template dient der Sammlung und Anzeige von Schaltschrankmeldungen. Eine Auswertung der Alarme der Automationsstation findet statt und ein zentrales Quittieren kann ausgelöst werden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.2 ControlCabinet

6.1.4.2.1.2.2.1 FB_BA_ControlCabinetBasicExt



Das Template dient der Ansteuerung einer Sammelstörmeldelampe und dem Quittieren bzw. Rücksetzen von Alarmen mittels eines Tasters.

Der im Template verwendete Funktionsbaustein FB_BA_EventObserver [▶ 137] dient dem Quittieren, dem Rücksetzen und der Ausgabe aller Alarme in der SPS.

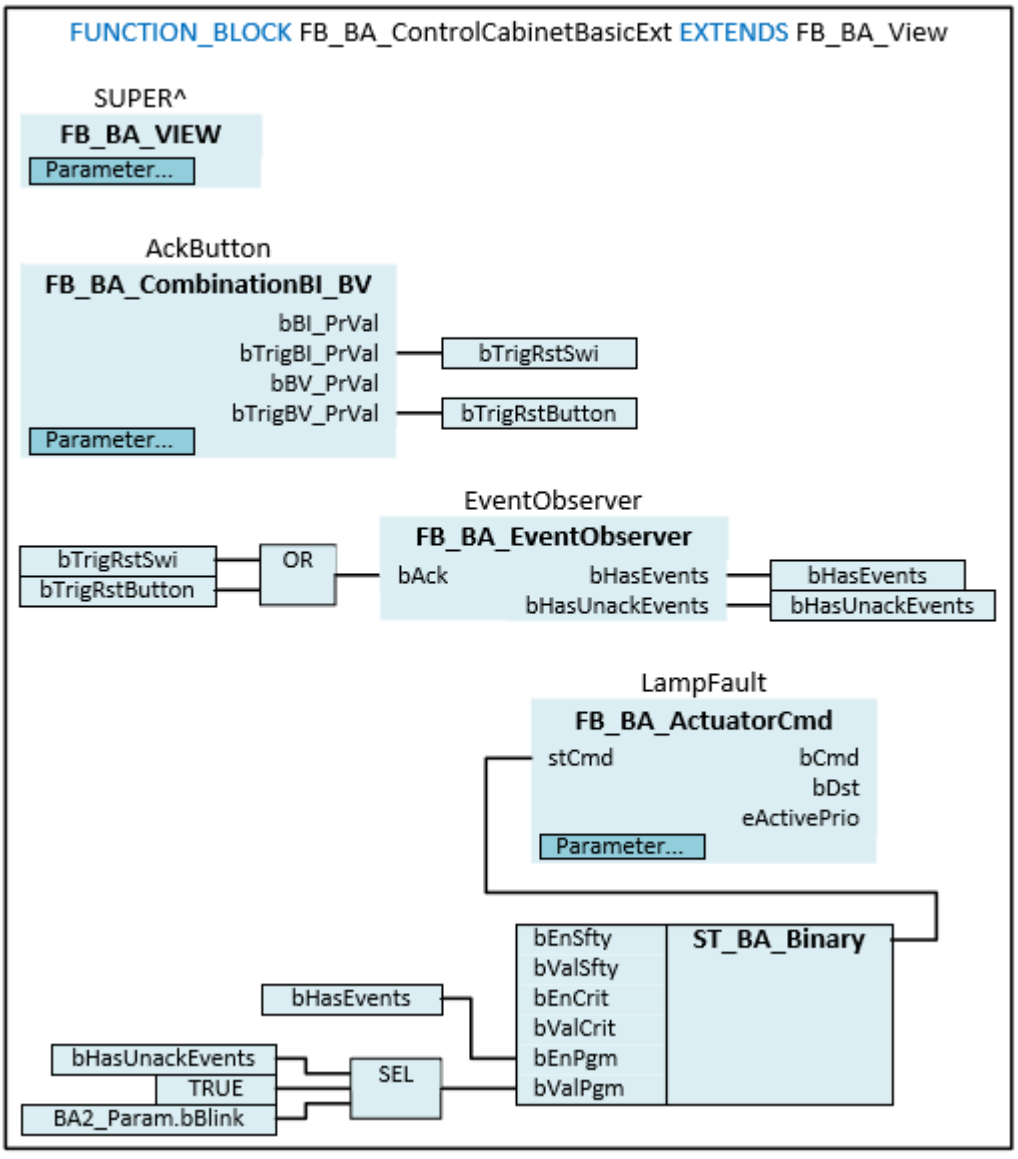
Der resultierende Sammelarm wird in dem Template LampFault [▶ 1004] zur Ansteuerung einer Sammelstörmeldelampe genutzt. Bei einem neuen, nicht quittierten Alarm bzw. Event wird die Meldelampe blinkend angesteuert.

Falls nur quittierte Alarme anstehen, ist die Lampe dauernd an (Neuwertmeldung).

Für das Quittieren und Rücksetzen eines Alarms muss der Quittiertaster zweimal betätigt werden. Sollte das Quittieren und Rücksetzen mit einem Tastendruck erfolgen, dann muss in der GVL BA_Param der Wert von *eEvtMgmt_AckMode.eSingle* auf *eEvtMgmt_AckMode.eEntire* geändert werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ControlCabinetBasicExt EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  LampFault          : FB_BA_ActuatorCmd;
  AckButton          : FB_BA_CombinationBI_BV;
  EventObserver      : FB_BA_EventObserver;
END_VAR
VAR
  bHasEvent          : BOOL;
  bHasUnackEvent     : BOOL;
  bTrigRstSwi        : BOOL;
  bTrigRstButton     : BOOL;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
LampFault	FB_BA_ActuatorCmd [▶ 1004]	Das Template dient der Ansteuerung einer Sammelstörmeldelampe.
AckButton	FB_BA_CombinationBI_BV [▶ 1103]	Das Template beinhaltet einen binären Eingang zum Anschluss des Quittier-Tasters sowie ein binäres Value Objekt zur Fernauslösung einer Quittierung von der Managementebene aus.
EventObserver	FB_BA_EventObserver [▶ 137]	Der Funktionsbaustein EventObserver realisiert die Auswertung von sämtlichen Alarmen / Ereignissen der Projektstruktur (Grundgerüst) und deren Quittierung. Die Verbindung zur Projektstruktur geschieht über die Initialisierung des Properties *Parent* des Template FB_BA_EventObserver. In diesem Template wird an dem Property eine Zuweisung auf das Basisobjekt FB_BA_TopView der Projektstruktur/Grundgerüst realisiert.

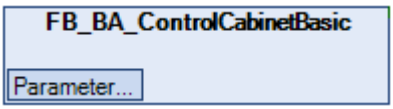
VAR

Name	Typ	Beschreibung
bHasEvent	BOOL	Diese Variable zeigt an, dass ein Alarm im Projekt vorhanden ist.
bHasUnackEvent	BOOL	Diese Variable zeigt an, dass nicht quittierte Alarme in dem Projekt vorhanden sind.
bTrigRstSwi	BOOL	Anzeige des Quittiersignals vom BV-Objekt *Input* (siehe FB_BA_CombinationBI_BV).
bTrigRstButton	BOOL	Anzeige des Quittiersignals vom BV-Objekt *Value* (siehe FB_BA_CombinationBI_BV).

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.2.2 FB_BA_ControlCabinetBasic



Das Template dient der Ansteuerung einer Sammelstörmeldelampe, dem Quittieren bzw. Rücksetzen von Alarmen mittels eines Tasters und dem Rücksetzen von Hardware im Schaltschrank durch die Ausgabe eines Wischimpulses.

Der im Template verwendete Funktionsbaustein FB_BA_EventObserver [▶ 137] dient dem Quittieren, dem Rücksetzen und der Ausgabe aller Alarme in der SPS.

Der resultierende Sammelarm wird in dem Template LampFault [▶ 1004] zur Ansteuerung einer Sammelstörmeldelampe genutzt. Bei einem neuen, nicht quittierten Alarm bzw. Event wird die Meldelampe blinkend angesteuert.

Falls nur quittierte Alarme anstehen, ist die Lampe dauernd an (Neuwertmeldung).

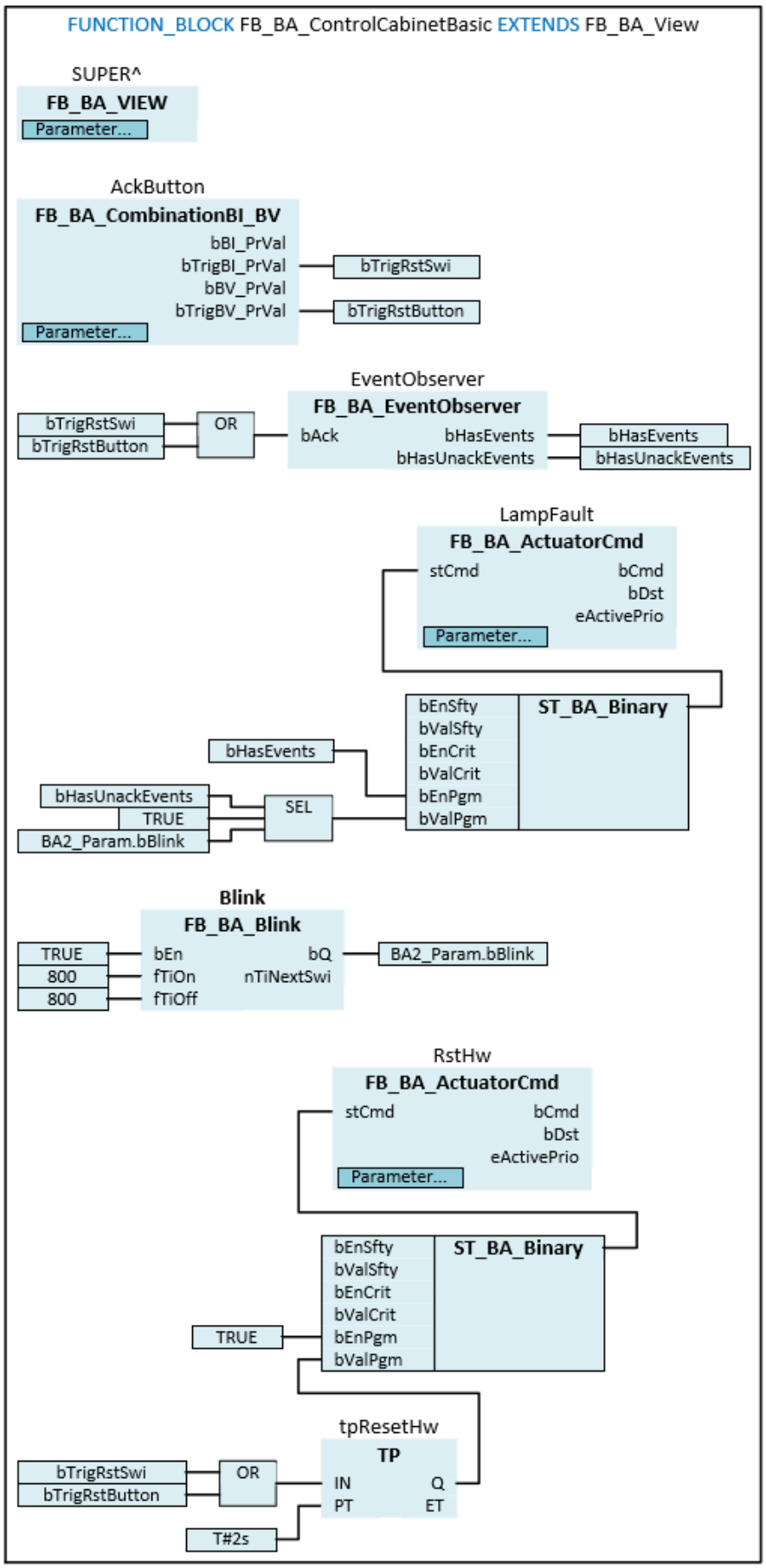
Ein Wischimpuls für das Rücksetzen von Hardware, wird über den Baustein RstHw [▶ 1004] ausgegeben. Dieser ist durch das Zeitglied *tpResetHw* auf zwei Sekunden verlängert. So werden Relaisschaltungen innerhalb des Schaltschrankes sicher durch einen ausreichend langen Impuls quittiert.

Für das Quittieren und Rücksetzen eines Alarms muss der Quittiertaster zweimal betätigt werden. Sollte das Quittieren und Rücksetzen mit einem Tastendruck erfolgen, dann muss in der GVL BA_Param der Wert von *eEvtMgmt_AckMode.eSingle* auf *eEvtMgmt_AckMode.eEntire* geändert werden.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ControlCabinetBasic EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  LampFault          : FB_BA_ActuatorCmd;
  AckButton          : FB_BA_CombinationBI_BV;
  RstHw              : FB_BA_ActuatorCmd;
  EventObserver      : FB_BA_EventObserver;
END_VAR
VAR
  bHasEvent          : BOOL;
  bHasUnackEvent     : BOOL;
  Blink              : FB_BA_Blink;
  tpResetHw          : TP;
  bTrigRstSwi        : BOOL;
  bTrigRstButton     : BOOL;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
LampFault	FB_BA_ActuatorCmd [▶ 1004]	Das Template dient der Ansteuerung einer Sammelstörmeldelampe.
AckButton	FB_BA_CombinationBI_BV [▶ 1103]	Das Template beinhaltet einen binären Eingang zum Anschluss des Quittier-Tasters sowie ein binäres Value Objekt zur Fernauslösung einer Quittierung von der Managementebene aus.
RstHw	FB_BA_ActuatorCmd [▶ 1004]	Das Template kann zum Hereinwischen von Relaisschaltungen (z.B. Frostschutzrelais) verwendet werden.
EventObserver	FB_BA_EventObserver [▶ 137]	Der Funktionsbaustein EventObserver realisiert die Auswertung von sämtlichen Alarmen / Ereignissen der Projektstruktur (Grundgerüst) und deren Quittierung. Die Verbindung zur Projektstruktur geschieht über die Initialisierung des Properties *Parent* des Template FB_BA_EventObserver. In diesem Template wird an dem Property eine Zuweisung auf das Basisobjekt FB_BA_TopView der Projektstruktur/Grundgerüst realisiert.

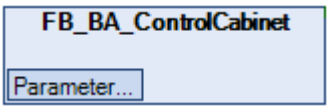
VAR

Name	Typ	Beschreibung
bHasEvent	BOOL	Diese Variable zeigt an, dass ein Alarm im Projekt vorhanden ist.
bHasUnackEvent	BOOL	Diese Variable zeigt an, dass nicht quittierte Alarme in dem Projekt vorhanden sind.
Blink	FB_BA_Blink [▶ 445]	Der Funktionsbaustein erzeugt einen Blinkimpuls. Dieser Blinkimpuls wird auf die globale Variable *BA2_Param.bBlink* geschrieben.
tpResetHw	TP	Das Zeitglied verlängert den Quittierimpuls für das Hereinwischen von Relaisschaltungen (z.B. Frostschutzrelais).
bTrigRstSwi	BOOL	Anzeige des Quittiersignals vom BV-Objekt *Input* (siehe FB_BA_CombinationBI_BV).
bTrigRstButton	BOOL	Anzeige des Quittiersignals vom BV-Objekt *Value* (siehe FB_BA_CombinationBI_BV).

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.2.3 FB_BA_ControlCabinet



Das Template dient zur Sammlung und Anzeige von Schaltschrankmeldungen wie z. B. Sicherungen, Überstromschutzeinrichtungen, Netzwiederkehr Relais, usw. Es besteht im Wesentlichen aus drei BI-Objekten zur Anzeige von Sicherungen.

Das Template dient als Vorlage und soll auf die jeweiligen Gegebenheiten angepasst werden.

Die Basisklasse FB_BA_ControlCabinetBasic [▶ 838] greift auf das Grundgerüst zu. Es wertet die Alarme der Automationsstation aus und ein zentrales Quittieren kann ausgelöst werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Vererbunghierarchie

FB_BA_Base

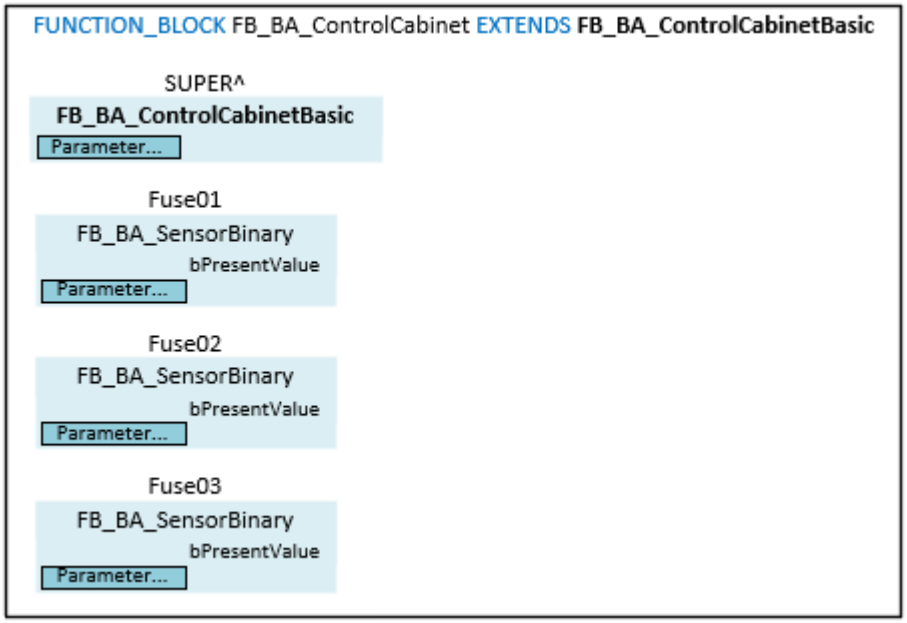
FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_View [▶ 246]

FB_BA_ControlCabinetBasic [▶ 838]

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ControlCabinet EXTENDS FB_BA_ControlCabinetBasic
VAR_INPUT CONSTANT
  Fuse01    : FB_BA_SensorBinary;
  Fuse02    : FB_BA_SensorBinary;
  Fuse03    : FB_BA_SensorBinary;
END_VAR
```

VAR_INPUT CONSTANT

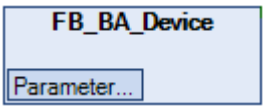
Name	Typ	Beschreibung
Fuse01	FB_BA_SensorBinary [▶ 1107]	Das Template beinhaltet einen binären Eingang zum Anschluss einer Sicherungsmeldung.
Fuse02	FB_BA_SensorBinary [▶ 1107]	Das Template beinhaltet einen binären Eingang zum Anschluss einer Sicherungsmeldung.
Fuse03	FB_BA_SensorBinary [▶ 1107]	Das Template beinhaltet einen binären Eingang zum Anschluss einer Sicherungsmeldung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.3 Device

6.1.4.2.1.2.3.1 FB_BA_Device



Das Template beinhaltet verschiedene Basisfunktionen.

Der Funktionsbaustein FB_BA_ProjectEx liefert Informationen zum Projekt und bietet Informationen und Diagnosemöglichkeiten an. Er löst das persistente Speichern von Daten aus. Eine Teil-Parametrierung des BACnet Device findet über den Funktionsbaustein FB_BA_ProjectEx statt.

Das Template FB_BA_TrendLogging beinhaltet diverse Trendlog-Objekte.

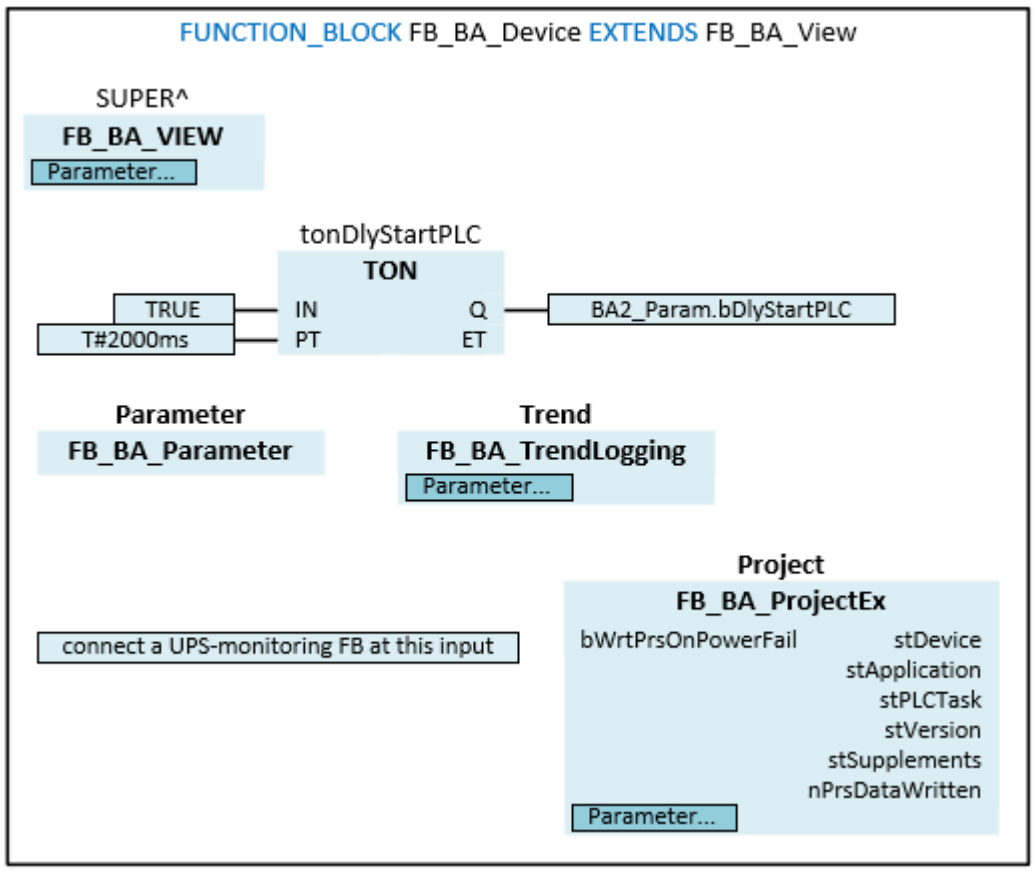
Über den Funktionsbaustein FB_BA_Parameter [▶ 846] werden globale Parameterlisten der Bibliothek Tc3_XBA [▶ 101] parametriert.

Innerhalb des Ordners TimeSettings gibt es zwei Möglichkeiten, um die BACnet-Systemzeit oder die lokale NT-Systemzeit des TwinCAT-Systems zu integrieren.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Settings EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  Trend             : FB_BA_TrendLogging;
  Project           : FB_BA_ProjectEx;
  Parameter         : FB_BA_Parameter;
END_VAR
VAR
  tonDlyStartPLC    : TON;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
Trend	FB_BA_TrendLogging [▶ 1074]	Aufruf der Trendlog-Objekte.
Project	FB_BA_ProjectEx [▶ 854]	Der Funktionsbaustein liefert Informationen zum Projekt und bietet Informationen und Diagnosemöglichkeiten an. Zudem speichert er die persistenten Daten. Er besitzt den Eingang *bWrtPrsOnPowerFail*, der bei einer Flanke die persistenten Daten abspeichert und zur Verknüpfung eines USV-Bausteins gedacht ist. Weiterhin dient er der Parametrierung des Objektnamens und der Objektbeschreibung des BACnet Device Objektes (Server).
Parameter	FB_BA_Parameter [▶ 846]	Innerhalb des Funktionsbausteins werden globale Parameterlisten der Bibliotheken Tc3_XBA [▶ 101], Tc3_BACnetRev14, Tc3_BA2_Common und Tc3_BA2 [▶ 272] parametriert.

Variablen

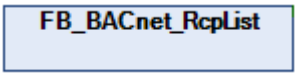
Name	Typ	Beschreibung
tonDlyStartPLC	TON	Das Zeitglied setzt nach einem Neustart der SPS verzögert die globale Variable bDlyStartPLC [▶ 1137].

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.3.2 BACnetSettings

6.1.4.2.1.2.3.2.1 FB_BACnet_RcpList



Das Beispiel zeigt die Parametrierung der Empfängerliste der Meldungs-/Ereignisklassen-Objekte an. Dieses Beispiel muss auf das jeweilige Projekt angepasst werden.

Syntax

```
FUNCTION_BLOCK FB_BACnet_RcpList
VAR
  bWrite                    : BOOL ;

// Recipient list to write (to all event classes):
  aRecipientList : T_BACnet_RecipientList := [(
       nProcessId           := 10000,
       stValidDays          := F_BACnet_ValidDays(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE),
       stFromTime           := F_BA_ToSTTime(T#0H),
       stToTime             := F_BA_ToSTTime(T#23H59M59S),
       bIssueConfirmed      := FALSE,
       stEventTransitions   := F_BACnet_EventTransitionBits(TRUE, TRUE, TRUE),
       stRecipient          := F_BACnet_DeviceRecipient(nDeviceInstance:=1445709)
      ),(
       nProcessId           := 30100,
       stValidDays          := F_BACnet_ValidDays(TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, FALSE),
       stFromTime           := F_BA_ToSTTime(T#0H),
       stToTime             := F_BA_ToSTTime(T#23H59M59S),
       bIssueConfirmed      := TRUE,
       stEventTransitions   := F_BACnet_EventTransitionBits(TRUE, TRUE, TRUE),
       stRecipient          := F_BACnet_EthernetRecipient(nIPAddress1:=192,168,10,200, nPort:=47808
, nNetworkNr:=444)
      ),(
       nProcessId           := 40100,
       stValidDays          := F_BACnet_ValidDays(TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, FALSE),
       stFromTime           := F_BA_ToSTTime(T#0H),
       stToTime             := F_BA_ToSTTime(T#23H59M59S),
       bIssueConfirmed      := TRUE,
       stEventTransitions   := F_BACnet_EventTransitionBits(TRUE, TRUE, TRUE),
       stRecipient          := F_BACnet_EthernetRecipient(nIPAddress1:=192,168,15,200, nPort:=47808
, nNetworkNr:=555)
      )];
END_VAR
VAR
  bResult                   : BOOL;
  _iObj                     : I_BA_Object;
  _fbBACnetObj              : POINTER TO FB_BACnet_BaseObject;
  _hRes                     : HRESULT;
END_VAR
```

Implementierungsteil

```
IF (XBA_Globals.Top.ProjectState = E_BA_ProjectState.eFirstOpCycle) THEN
  bWrite := TRUE;
END_IF

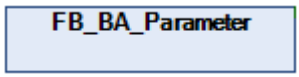

IF (bWrite) THEN
```

```
  bWrite := FALSE;
  // Iterate over all event class objects:
  _iObj := 0;
  WHILE (F_BA_IterateObjectIndex(_iObj, E_BA_ObjectType.eEventClass)) DO
    // Receive internal BACnet object from BA object:
    IF (NOT _iObj.GetBACnetObject(fbObject=>_fbBACnetObj)) THEN
      _iObj.LogMsg.Show(ADSLOG_MSGTYPE_ERROR, 'RL09', 'Failed to receive internal BACnet object!', T
RUE);
    ELSE
      // Write recipient list:
      _hRes := _fbBACnetObj^.WritePropertyRecipientList(aRecipientList);
      IF (FAILED(_hRes)) THEN
        ; // Do some error handling (An error message has already been logged here).
      END_IF
    END_IF
  END_WHILE
END_IF
```

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.3.2.2 FB_BA_Parameter



Der Funktionsbaustein beinhaltet im Deklarationsteil der Methode *FB_init* den Funktionsbaustein *FB_BA_Param*. Dieser dient der Parametrierung der globalen Variablenlisten *XBA_BACnetParam* und XBA Param [▶ 128] der Bibliothek Tc3_XBA [▶ 101]. Im Implementierungsteil der Methode *FB_init* sind dazu diverse Parametrierungen aufgeführt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Ein Aufruf des *FB_BA_Param* ist aufgrund der internen Attribute *{attribute 'no_explicit_call' := 'No need to call this FB.'}* und *{attribute 'TcIgnorePersistent'}* nicht nötig, siehe FB_BA_Param [▶ 163]!

Außerdem ist im Template ein auskommentiertes Beispiel vorhanden, welches die Parametrierung der Empfängerliste der Meldungs-/Ereignisklassen-Objekte beinhaltet. Dieses Beispiel muss auf das jeweilige Projekt angepasst werden.

Syntax

```
FUNCTION_BLOCK FB_BA_Parameter
VAR
//    Recipient    : FB_BACnet_RcpList;
END_VAR
```

Implementierungsteil

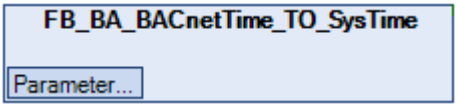
```
// Recipient list to write (to all event classes):
// Recipient();
```

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.3.3 TimeSettings

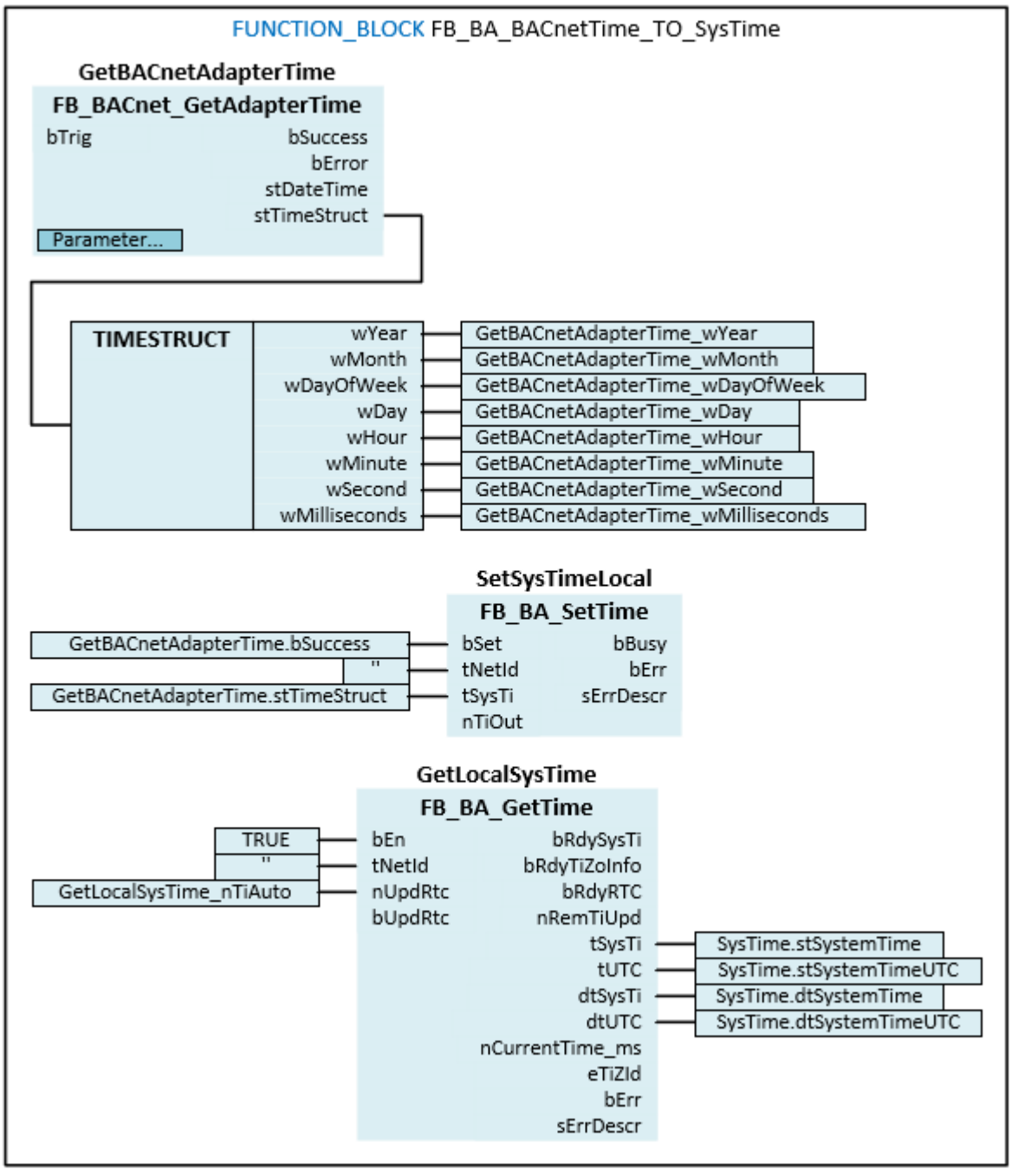
6.1.4.2.1.2.3.3.1 FB_BA_BACnetTime_TO_SysTime



Das Template liest in regelmäßigen Zeitabständen die BACnet-Systemzeit aus und schreibt diese auf die lokale NT-Systemzeit des TwinCAT-Systems.

Die NT-Systemzeit wird in regelmäßigen Zeitabständen ausgelesen und die Zeitinformationen in der SPS auf lokale und globale Variablen [▶ 1136] abgebildet.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_BACnetTime_TO_SysTime
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterUnit':= 's'}
  GetLocalSysTime_nTiAuto                : UDINT := 60;
END_VAR
VAR_INPUT CONSTANT
  SetSysTimeLocal                        : FB_BA_SetTime;
  GetLocalSysTime                        : FB_BA_GetTime;
  GetBACnetAdapterTime                   : FB_BACnet_GetAdapterTime;
END_VAR
VAR
  GetBACnetAdapterTime_wYear             : WORD;
  GetBACnetAdapterTime_wMonth            : WORD;
  GetBACnetAdapterTime_wDayOfWeek        : WORD;
  GetBACnetAdapterTime_wDay              : WORD;
  GetBACnetAdapterTime_wHour             : WORD;
  GetBACnetAdapterTime_wMinute           : WORD;
  GetBACnetAdapterTime_wSecond           : WORD;
  GetBACnetAdapterTime_wMilliseconds     : WORD;
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
GetLocalSysTime_nTiAuto	UDINT	Anhand dieser Zeitangabe wird die lokale NT-Systemzeit des TwinCAT-Systems regelmäßig ermittelt.

Eingänge CONSTANT

Name	Typ	Beschreibung
SetSysTimeLocal	FB_BA_SetTime	Die lokal, ermittelte BACnet-Systemzeit (GetBACnetAdapterTime) wird mit dem Funktionsbaustein SetSysTimeLocal auf die lokale NT-Systemzeit des TwinCAT-Systems geschrieben.
GetLocalSysTime	FB_BA_GetTime	Der Funktionsbaustein GetLocalSysTime ermittelt die lokale NT-Systemzeit des TwinCAT-Systems. Die ermittelte Zeit wird an die globale Variablenlisten SysTime [▶ 1136] übertragen.
GetBACnetAdapterTime	FB_BACnet_GetAdapterTime	Der Funktionsbaustein GetBACnetAdapterTime ermittelt die lokale BACnet-Systemzeit. Wichtige Parameter sind unten erläutert.

VAR

Name	Typ	Beschreibung
GetBACnetAdapterTime_wYear	WORD	Jahresangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wMonth	WORD	Monatsangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wDayOfWeek	WORD	Angabe des Wochentages der BACnet-Systemzeit.
GetBACnetAdapterTime_wDay	WORD	Angabe des Tages im Monat der BACnet-Systemzeit.
GetBACnetAdapterTime_wHour	WORD	Stundenangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wMinute	WORD	Minutenangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wSecond	WORD	Sekundenangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wMilliseconds	WORD	Millisekundenangabe der BACnet-Systemzeit.

Parameter of GetBACnetAdapterTime

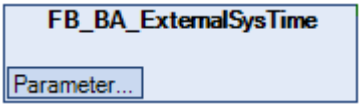
Diese Parameter dienen zur Konfiguration der Zeitsynchronisation.

Name	Typ	Beschreibung
GetBACnetAdapterTime.bTrig	BOOL	Bei einer steigenden Flanke an diesem Eingang wird die lokale BACnet-Systemzeit eines TwinCAT-Systems ermittelt.
GetBACnetAdapterTime.nTiAuto	UDINT	Anhand dieser Zeitangabe aus dem Parameter Menü wird die lokale BACnet-Systemzeit eines TwinCAT-Systems regelmäßig automatisch ermittelt. Die Zeitangabe muss größer gleich 500ms sein. Der Default Wert beträgt 1800s.
GetBACnetAdapterTime.fbAdapter	fbAdapter	Der Funktionsbaustein *fbAdapter* repräsentiert ein BACnet Device. Dieses ist vorinitialisiert auf das lokale BACnet Device des TwinCAT Systems unter Geräte.
GetBACnetAdapterTime.bSuccess	BOOL	Die Systemzeit wurde erfolgreich aus dem Zielsystem gelesen. Die Variable ist für einen Zyklus TRUE.
GetBACnetAdapterTime.stDateTime	ST_BA_DateTime	Aktuelle BACnet-Systemzeit eines TwinCAT-Systems.
GetBACnetAdapterTime.stTimeStruct	TIMESTRUCT	Aktuelle BACnet-Systemzeit eines TwinCAT-Systems.
GetBACnetAdapterTime.bError	BOOL	Fehler beim Auslesen der BACnet-Systemzeit.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.3.3.2 FB_BA_ExternalSysTime

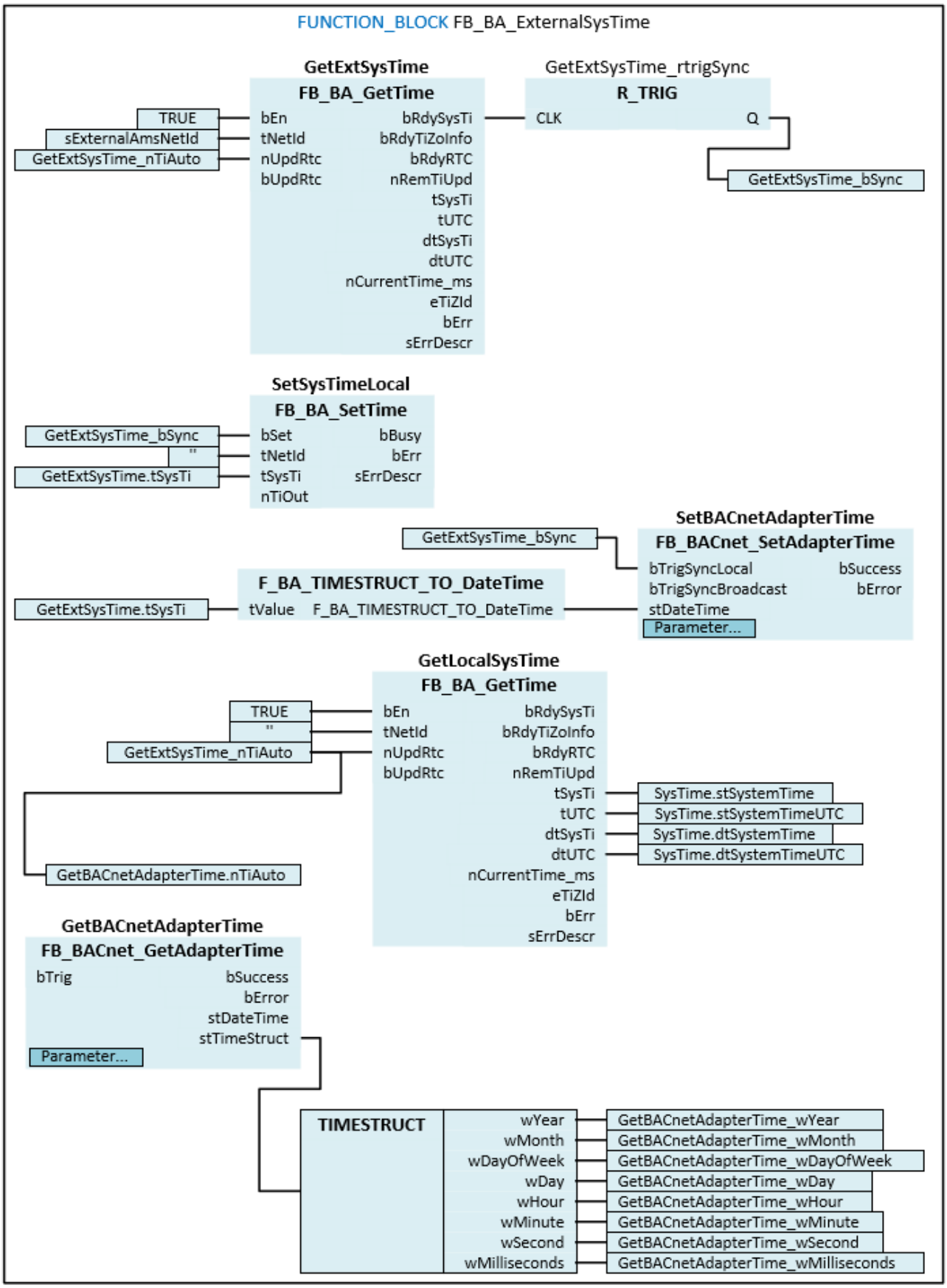


Das Template liest in regelmäßigen Zeitabständen die NT-Systemzeit eines externen TwinCAT System aus.

Wenn das Auslesen erfolgreich war, dann wird die externe Systemzeit auf die lokale NT-Systemzeit des TwinCAT Systems und die lokale BACnet-Systemzeit des TwinCAT Systems geschrieben.

In regelmäßigen Zeitabständen werden die beiden lokalen Systemzeiten ausgelesen und in der SPS auf lokale und globale Variablen abgebildet.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ExternalSysTime
VAR_INPUT CONSTANT PERSISTET
  sExternalAmsNetId                    : T_AmsNetId := '';
```

```
  {attribute 'parameterUnit':= 's'}
  GetExtSysTime_nTiAuto                  : UDINT := 600;
  {attribute 'parameterUnit':= 's'}
  GetLocalSysTime_nTiAuto                : UDINT := 60;
END_VAR
VAR_INPUT CONSTANT
  GetExtSysTime                          : FB_BA_GetTime;

  SetSysTimeLocal                        : FB_BA_SetTime;
  SetBACnetAdapterTime                   : FB_BACnet_SetAdapterTime;

  GetLocalSysTime                        : FB_BA_GetTime;
  GetBACnetAdapterTime                   : FB_BACnet_GetAdapterTime;
END_VAR
VAR
  GetExtSysTime_bSync                    : BOOL;
  GetExtSysTime_rtrigSync                : R_TRIG;

  GetBACnetAdapterTime_wYear             : WORD;
  GetBACnetAdapterTime_wMonth            : WORD;
  GetBACnetAdapterTime_wDayOfWeek        : WORD;
  GetBACnetAdapterTime_wDay              : WORD;
  GetBACnetAdapterTime_wHour             : WORD;
  GetBACnetAdapterTime_wMinute           : WORD;
  GetBACnetAdapterTime_wSecond           : WORD;
  GetBACnetAdapterTime_wMilliseconds     : WORD;
END_VAR
```

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
sExternalAmsNetId	T_AmsNetId	Hier wird die AmsNetId des TwinCAT-Rechners angegeben dessen NT-Systemzeit ausgelesen werden soll.
GetBACnetAdapterTime_nTiAuto	UDINT	Anhand dieser Zeitangabe wird die lokale BACnet-Systemzeit des TwinCAT Systems regelmäßig ermittelt.
GetLocalSysTime_nTiAuto	UDINT	Anhand dieser Zeitangabe wird die lokale NT-Systemzeit des TwinCAT Systems regelmäßig ermittelt.

Eingänge CONSTANT

Name	Typ	Beschreibung
GetExtSysTime	FB_BA_GetTime	Der Funktionsbaustein GetExtSysTime ermittelt die externe NT-Systemzeit eines TwinCAT Systems.
SetSysTimeLocal	FB_BA_SetTime	Die lokal, ermittelte BACnet-Systemzeit (GetBACnetAdapterTime) wird mit dem Funktionsbaustein SetSysTimeLocal auf die lokale NT-Systemzeit des TwinCAT-Systems geschrieben.
SetBACnetAdapterTime	FB_BACnet_SetAdapterTime	Der Funktionsbaustein SetBACnetAdapterTime kann sowohl die lokale BACnet-Systemzeit eines TwinCAT-Systems (*SetBACnetAdapterTime.bTrigSyncLocal*) wie auch durch Auslösen eines Broadcast-Befehls (*SetBACnetAdapterTime.bTrigSyncBroadcast*) sämtliche BACnet-Systemzeiten eines Netzes setzen. Wichtige Parameter sind unten erlätert.
GetLocalSysTime	FB_BA_GetTime	Der Funktionsbaustein GetLocalSysTime ermittelt die lokale NT-Systemzeit des TwinCAT-Systems. Die ermittelte Zeit wird an die globale Variablenlisten SysTime [▶ 1136] übertragen.
GetBACnetAdapterTime	FB_BACnet_GetAdapterTime	Der Funktionsbaustein "GetBACnetAdapterTime" ermittelt die lokale BACnet-Systemzeit. Wichtige Parameter sind unten erläutert.

VAR

Name	Typ	Beschreibung
GetExtSysTime_bSync	BOOL	Die Variable löst das Schreiben der externen Systemzeit auf die lokale NT-Systemzeit des TwinCAT Systems und die lokale BACnet-Systemzeit des TwinCAT Systems aus.
GetExtSysTime_rtrigSync	R_TRIG	Der Funktionsbaustein erzeugt nach dem erfolgreichen Auslesen der externen Systemzeit GetExtSysTime eine steigende Flanke und gibt diese an *GetExtSysTime_bSync* weiter.
GetBACnetAdapterTime_wYear	WORD	Jahresangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wMonth	WORD	Monatsangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wDayOfWeek	WORD	Angabe des Wochentages der BACnet-Systemzeit.
GetBACnetAdapterTime_wDay	WORD	Angabe des Tages im Monat der BACnet-Systemzeit.
GetBACnetAdapterTime_wHour	WORD	Stundenangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wMinute	WORD	Minutenangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wSecond	WORD	Sekundenangabe der BACnet-Systemzeit.
GetBACnetAdapterTime_wMilliseconds	WORD	Millisekundenangabe der BACnet-Systemzeit.

Parameter of SetBACnetAdapterTime

Diese Parameter dienen zur Konfiguration der Zeitsynchronisation.

Name	Typ	Beschreibung
SetBACnetAdapterTime.bTrigSyncLocal	BOOL	Bei einer steigenden Flanke an diesem Eingang wird der Inhalt der Struktur *SetBACnetAdapterTime.stDateTime* auf die lokale BACnet-Systemzeit des TwinCAT-Systems geschrieben.
SetBACnetAdapterTime.bTrigSyncBroadcast	BOOL	Bei einer steigenden Flanke an diesem Eingang wird der Inhalt der Struktur *SetBACnetAdapterTime.stDateTime* auf sämtliche BACnet-Systemzeiten eines Netzes wie auch die lokale BACnet-Systemzeit des TwinCAT-Systems geschrieben.
SetBACnetAdapterTime.stDateTime	ST_BA_DateTime	Zeitangabe, die auf die lokale BACnet-Systemzeit geschrieben wird.
SetBACnetAdapterTime.fbAdapter	fbAdapter	Der Funktionsbaustein *fbAdapter* repräsentiert ein BACnet Device. Dieses ist vorinitialisiert auf das lokale BACnet Device des TwinCAT Systems unter Geräte.
SetBACnetAdapterTime.bSuccess	BOOL	Die Systemzeit wurde erfolgreich aus dem Zielsystem gelesen. Die Variable ist für einen Zyklus TRUE.
SetBACnetAdapterTime.bError	BOOL	Fehler beim Auslesen der BACnet-Systemzeit.

Parameter of GetBACnetAdapterTime

Diese Parameter dienen zur Konfiguration der Zeitsynchronisation.

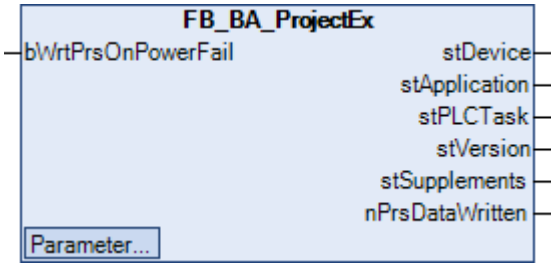
Name	Typ	Beschreibung
GetBACnetAdapterTime.bTrig	BOOL	Bei einer steigenden Flanke an diesem Eingang wird die lokale BACnet-Systemzeit eines TwinCAT-Systems ermittelt.
GetBACnetAdapterTime.nTiAuto	UDINT	Anhand dieser Zeitangabe aus dem Parameter Menü wird die lokale BACnet-Systemzeit eines TwinCAT-Systems regelmäßig automatisch ermittelt. Die Zeitangabe muss größer gleich 500ms sein. Der Default Wert beträgt 1800s.
GetBACnetAdapterTime.fbAdapter	fbAdapter	Der Funktionsbaustein *fbAdapter* repräsentiert ein BACnet Device. Dieses ist vorinitialisiert auf das lokale BACnet Device des TwinCAT Systems unter Geräte.
GetBACnetAdapterTime.bSuccess	BOOL	Die Systemzeit wurde erfolgreich aus dem Zielsystem gelesen. Die Variable ist für einen Zyklus TRUE.
GetBACnetAdapterTime.stDateTime	ST_BA_DateTime	Aktuelle BACnet-Systemzeit eines TwinCAT-Systems.
GetBACnetAdapterTime.stTimeStruct	TIMESTRUCT	Aktuelle BACnet-Systemzeit eines TwinCAT-Systems.
GetBACnetAdapterTime.bError	BOOL	Fehler beim Auslesen der BACnet-Systemzeit.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.2.3.4 Projekt

6.1.4.2.1.2.3.4.1 FB_BA_ProjectEx



Dieses Template leitet sich von dem Basisbaustein FB_BA_Project [▶ 232] ab.

Es stellt zu Informationszwecken die Projekt-Versionsnummer zur Verfügung und sorgt für das Speichern der persistenten Daten.

Das Schreiben der persistenten Daten kennt 2 Modi:

- **Normalzustand der Anlage:**

Die Inbetriebnahme ist abgeschlossen und die Anlage läuft.

In einem festen Intervall *tPrsData_DefaultPeriod*, voreingestellt auf 30 Minuten, werden die persistenten Daten geschrieben. Damit wird die Speicherkarte geschont. Ein Parametrieren von *tPrsData_DefaultPeriod* unter 10 Minuten setzt diesen Wert sicherheitshalber auf 30 Minuten zurück.

- **Inbetriebnahmemodus**:

Durch das automatische Erkennen eines online-change wird in den Inbetriebnahmemodus gewechselt und die persistenten Daten werden nun in kürzeren Abständen gespeichert. Es gilt hier das Intervall *tPrsData_ComissioningPeriod*, welches auf 10 Minuten voreingestellt ist. Ein Parametrieren von *tPrsData_ComissioningPeriod* unter 30 Sekunden setzt diesen Wert sicherheitshalber auf 5 Minuten zurück.

Wird nach Ablauf einer fest eingestellten Zeit *DETECT_COMISSIONING_DURATION* von 60 Minuten kein online-change mehr festgestellt, so fällt die Routine in den Normalbetrieb zurück.

Syntax

```
FUNCTION_BLOCK FB_BA_ProjectEx EXTENDS FB_BA_Project
VAR_INPUT CONSTANT PERSISTENT
  tPrsData_DefaultPeriod         : TIME := T#30M;
  tPrsData_ComissioningPeriod    : TIME := T#10M;
END_VAR
VAR_INPUT CONSTANT
  bPrsData_ManualWriteData       : BOOL;
END_VAR
VAR_INPUT
  bWrtPrsOnPowerFail             : BOOL;
END_VAR
VAR_OUTPUT
  nPrsDataWritten                : UINT;
END_VAR
```

Eingänge CONSTANT PERSITENT

Name	Typ	Beschreibung
tPrsData_DefaultPeriod	TIME	Speicherintervall im Normalzustand.
tPrsData_ComissioningPeriod	TIME	Speicherintervall im Inbetriebnahmezustand.

Eingänge CONSTANT

Name	Typ	Beschreibung
bPrsData_ManualWriteData	BOOL	Zu Inbetriebnahme Zwecken kann diese Variable auf TRUE gesetzt werden. Dadurch wird einmalig ein Speichern der persistenten Variablen ausgelöst.

Eingänge

Name	Typ	Beschreibung
bWrtPrsOnPowerFail	BOOL	Eingangsvariable zur Verknüpfung mit einem USV-Baustein

Ausgänge

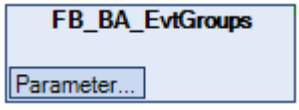
Name	Typ	Beschreibung
nPrsDataWritten	UINT	Zählerausgang für die erfolgten Schreibzyklen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.62	TF8040 | TwinCAT Building Automation ab V5.9.0.0

6.1.4.2.1.2.4 EventClasses

6.1.4.2.1.2.4.1 FB_BA_EvtGroups



Das Template stellt eine Gruppe von Meldeklassen zur Verfügung.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Meldungsklassenmatrix

Ereigniskategorie	Bedeutung	Priorität	Mel-dungs-klasse	NC-Objekt EC-Objekt	Beispiel
Gefahrenmeldung (Life Safety)	Gefahr für Leben	00 - 29	EC_ID.NC10	LifeSafety.EC	Brandalarm, Überfall
Gefahrenmeldung (Property Safety)	Sicherheitsmeldung	30 - 59	EC_ID.NC20	SafetyMsg.EC	Einbruch, unberechtigter Zutritt
Alarmmeldung	Meldung signalisiert Anlagenausfall oder erfordert sofortigen Eingriff.	60 - 89	EC_ID.NC30	AlarmMsg.EC	Sicherheitstemperaturbegrenzer (STB), Sicherheitsdruckbegrenzer (SDB), Übertemperatur der Warmwasserbereitung (WWB), Sicherheitsventile, Hauptpumpen, Keilriemenwächter, Frequenzumformer, Kälteanlagen, Spannungsausfall usw.
Störungsmeldung	Meldung weist auf einen anormalen Betriebszustand hin.	90 - 119	EC_ID.NC40	FaultMsg.EC	Temperaturwächter (TW), Druckwächter (DW), Temperaturüberwachung von Wärmetauscher (WT) und WWB, Motorschutz, Aufzug-Sammelstörmeldung, Netzdrücke usw.
Wartungsmeldung	Hinweis auf Wartungsaktivität o.ä.	120 - 149	EC_ID.NC50	MaintenanceMsg01.EC	Betriebsstunden, Behälterstand, Reparaturschalter usw.
Wartungsmeldung	Hinweis auf Wartungsaktivität o.ä.	120 - 149	EC_ID.NC51	MaintenanceMsg02.EC	Filterende erreicht, Filter verschmutzt usw.
Systemmeldung	Störungsmeldung aus dem GA-System.	150 - 219	EC_ID.NC60	SystemMsg.EC	Gerätestörung, Batteriemeldung, Kommunikationsunterbrechung usw.
Handeingriff	Handeingriff	220	EC_ID.NC70	ManualOp.EC	Handeingriff
Freibleibend	Sonstige Meldungen	221 - 255	EC_ID.NC80	OtherMsg.EC	Betriebszustandswechsel, Betriebsarten, Trendspeicher voll usw.

Syntax

```
FUNCTION_BLOCK FB_BA_EvtGroups EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  LifeSafety          : FB_BA_EvtCategory;
  SafetyMsg           : FB_BA_EvtCategory;
  AlarmMsg            : FB_BA_EvtCategory;
  FaultMsg            : FB_BA_EvtCategory;
  MaintenanceMsg01    : FB_BA_EvtCategory;
  MaintenanceMsg02    : FB_BA_EvtCategory;
  SystemMsg           : FB_BA_EvtCategory;
  ManualOp            : FB_BA_EvtCategory;
  OtherMsg            : FB_BA_EvtCategory;
END_VAR
```

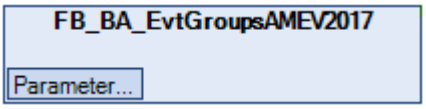
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
LifeSafety	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Gefahr für Leben“ auf.
SafetyMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Gefahrenmeldung, Sicherheitsmeldung“ auf.
AlarmMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Alarmmeldung“ auf.
FaultMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Störungsmeldung“ auf.
MaintenanceMsg01	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Wartungsmeldung“ auf.
MaintenanceMsg02	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Wartungsmeldung“ auf.
SystemMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Systemmeldung“ auf.
ManualOp	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Handeingriff“ auf.
OtherMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Sonstige Meldungen“ auf.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

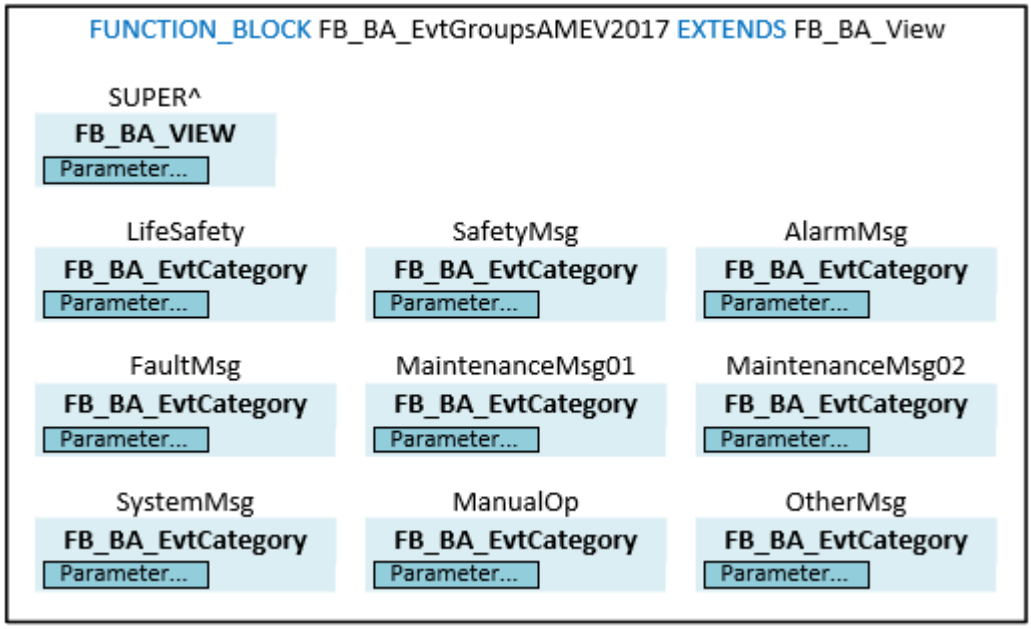
6.1.4.2.1.2.4.2 FB_BA_EvtGroupsAMEV2017



Das Template erzeugt die Meldeklassen nach den Anforderungen des AMEV (Arbeitskreis Maschinen- und Elektrotechnik staatlicher und kommunaler Verwaltungen).

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Meldungsklassenliste

Ereigniskategorie	Bedeutung	Priorität	Mel-dungs-klasse	NC-Objekt EC-Objekt	Beispiel
Gefahrenmeldung (Life Safety)	Gefahr für Leben	00 - 29	EC_ID.NC10	LifeSafety.EC	Brandalarm, Überfall
Gefahrenmeldung (Property Safety)	Sicherheitsmeldung	30 - 59	EC_ID.NC20	SafetyMsg.EC	Einbruch, unberechtigter Zutritt
Alarmmeldung	Meldung signalisiert Anlagenausfall oder erfordert sofortigen Eingriff.	60 - 89	EC_ID.NC30	AlarmMsg.EC	Sicherheitstemperaturbegrenzer (STB), Sicherheitsdruckbegrenzer (SDB), Übertemperatur der Warmwasserbereitung (WWB), Sicherheitsventile, Hauptpumpen, Keilriemenwächter, Frequenzumformer, Kälteanlagen, Spannungsausfall usw.
Störungsmeldung	Meldung weist auf einen anormalen Betriebszustand hin.	90 - 119	EC_ID.NC40	FaultMsg.EC	Temperaturwächter (TW), Druckwächter (DW), Temperaturüberwachung von Wärmetauscher (WT) und WWB, Motorschutz, Aufzug-Sammelstörmeldung, Netzdrücke usw.
Wartungsmeldung	Hinweis auf Wartungsaktivität o.ä.	120 - 149	EC_ID.NC50	MaintenanceMsg01.EC	Betriebsstunden, Behälterstand, Reparaturschalter usw.
Wartungsmeldung	Hinweis auf Wartungsaktivität o.ä.	120 - 149	EC_ID.NC51	MaintenanceMsg02.EC	Filterende erreicht, Filter verschmutzt usw.
Systemmeldung	Störungsmeldung aus dem GA-System.	150 - 219	EC_ID.NC60	SystemMsg.EC	Gerätestörung, Batteriemeldung, Kommunikationsunterbrechung usw.
Handmeldung	Handeingriff	220	EC_ID.NC70	ManualOp.EC	Handeingriff
Freibleibend	Sonstige Meldungen	221 - 255	EC_ID.NC80	OtherMsg.EC	Betriebszustandswechsel, Betriebsarten, Trendspeicher voll usw.

Syntax

```
FUNCTION_BLOCK FB_BA_EvtGroupsAMEV2017 EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  LifeSafety          : FB_BA_EvtCategory;
  SafetyMsg           : FB_BA_EvtCategory;
  AlarmMsg            : FB_BA_EvtCategory;
  FaultMsg            : FB_BA_EvtCategory;
  MaintenanceMsg01    : FB_BA_EvtCategory;
  MaintenanceMsg02    : FB_BA_EvtCategory;
  SystemMsg           : FB_BA_EvtCategory;
  ManualOp            : FB_BA_EvtCategory;
  OtherMsg            : FB_BA_EvtCategory;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
LifeSafety	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Gefahr für Leben“ auf.
SafetyMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Gefahrenmeldung, Sicherheitsmeldung“ auf.
AlarmMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Alarmmeldung“ auf.
FaultMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Störungsmeldung“ auf.
MaintenanceMsg01	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Wartungsmeldung“ auf.
MaintenanceMsg02	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Wartungsmeldung“ auf.
SystemMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Systemmeldung“ auf.
ManualOp	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Handeingriff“ auf.
OtherMsg	FB_BA_EvtCategory [▶ 1020]	Das Template ruft die Ereigniskategorie „Sonstige Meldungen“ auf.

Voraussetzungen

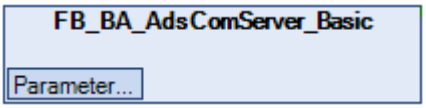
Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3 BuildingAutomation

Templates für gewerkeübergreifende Funktionen.

6.1.4.2.1.3.1 Communication

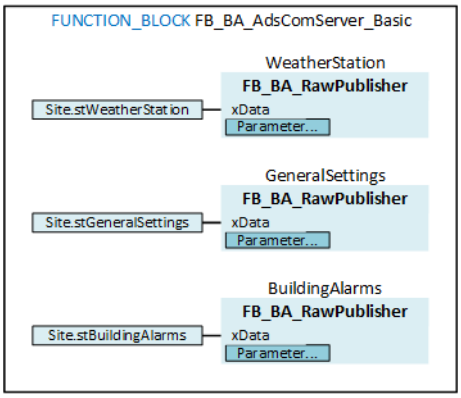
6.1.4.2.1.3.1.1 FB_BA_AdsComServer_Basic



Das Template hat die Aufgabe Daten und Parameter, welche dem Gewerk "Gebäudeautomation allgemein" zuzuordnen sind, aus der globalen Variablenliste Site [▶ 1135] zu lesen und via FB_BA_RawPublisher [▶ 143] innerhalb des GA-Netzwerkes zur Verfügung zu stellen.

Der Gegenbaustein, der diese Daten als Client empfängt, ist FB_BA_AdsComClient_Basic [▶ 861].

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComServer
VAR_INPUT CONSTANT
  WeatherStation     : FB_BA_RawPublisher;
  GeneralSettings    : FB_BA_RawPublisher;
  BuildingAlarms     : FB_BA_RawPublisher;
END_VAR
```

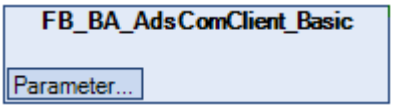
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
WeatherStation	FB_BA_RawPublisher [▶ 143]	Der Publisher *WeatherStation* veröffentlich die Struktur der Wetterstationsdaten Site.stWeatherStation [▶ 1135].
GeneralSettings	FB_BA_RawPublisher [▶ 143]	Der Publisher *GeneralSettings* veröffentlicht die Struktur der Witterungsparameter Site.stSettings [▶ 1135], welche im Template FB_BA_WeatherData [▶ 875] gebildet wird.
BuildingAlarms	FB_BA_RawPublisher [▶ 143]	Der Publisher *BuildingAlarms* veröffentlicht die Struktur der gebäudeweit gültigen Alarme Site.stBuildingAlarms [▶ 1135], welche in den Templates FB_BA_BurglarAlarmSystem und FB_BA_FireAlarmSystem gebildet werden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

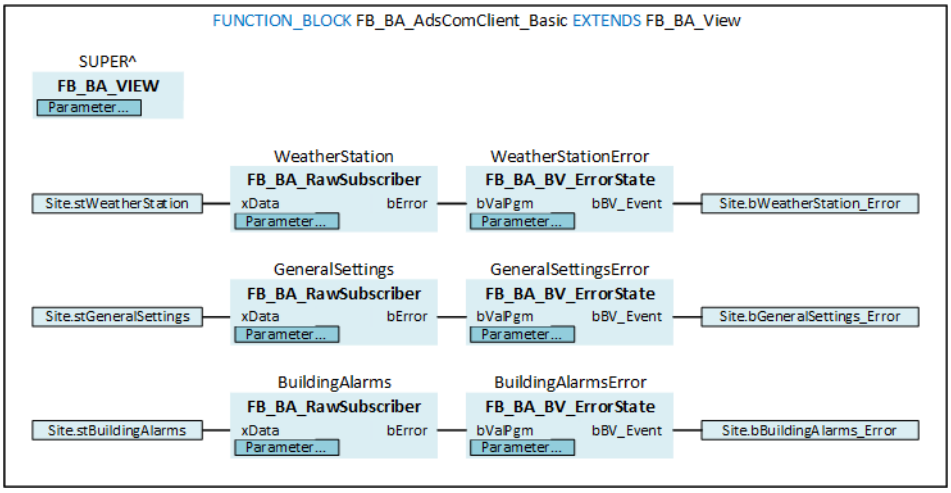
6.1.4.2.1.3.1.2 FB_BA_AdsComClient_Basic



Das Template liest globale Daten, welche dem Gewerk "Gebäudeautomation allgemein" zuzuordnen sind, von einem anderen Controller, der diese Daten mit dem Gegenbaustein FB_BA_AdsComServer_Basic [▶ 860] zur Verfügung stellt. Die gelesenen Informationen werden in die globale Variablenliste Site [▶ 1135] kopiert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  WeatherStation          : FB_BA_RawSubscriber;
  WeatherStationError     : FB_BA_BV_ErrorState;
  GeneralSettings         : FB_BA_RawSubscriber;
  GeneralSettingsError    : FB_BA_BV_ErrorState;
  BuildingAlarms          : FB_BA_RawSubscriber;
  BuildingAlarmsError     : FB_BA_BV_ErrorState;
END_VAR
```

VAR_INPUT CONSTANT

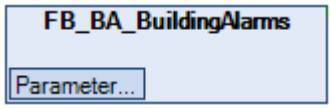
Name	Typ	Beschreibung
WeatherStation	FB_BA_RawSubscriber [▶ 162]	Der Subscriber greift auf die TwinCAT Netzwerk-Struktur *WeatherStation* zu und speichert die Daten in die angelegte Struktur *stWeatherStation* der GVL Site [▶ 1135] ab.
WeatherStationError	FB_BA_BV_ErrorState [▶ 1048]	Binäres Objekt zur Anzeige der Kommunikationsstörung des Subscribers *WeatherStation*.
GeneralSettings	FB_BA_RawSubscriber [▶ 162]	Der Subscriber greift auf die TwinCAT Netzwerk-Struktur *GeneralSettings* zu und speichert die Daten in die angelegte Struktur *stGeneralSettings* der GVL Site [▶ 1135] ab.
GeneralSettingsError	FB_BA_BV_ErrorState [▶ 1048]	Binäres Objekt zur Anzeige der Kommunikationsstörung des Subscribers *GeneralSettings*.
BuildingAlarms	FB_BA_RawSubscriber [▶ 162]	Der Subscriber greift auf die TwinCAT Netzwerk-Struktur *BuildingAlarms* zu und speichert die Daten in die angelegte Struktur *stBuildingAlarms* der GVL Site [▶ 1135] ab.
BuildingAlarmsError	FB_BA_BV_ErrorState [▶ 1048]	Binäres Objekt zur Anzeige der Kommunikationsstörung des Subscribers *BuildingAlarms*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.3.2 GlobalEventsAndSettings

6.1.4.2.1.3.2.1 FB_BA_BuildingAlarms



Dieses Template fasst gebäudespezifische Sicherheitskriterien zusammen und stellt sie in einer global deklarierten Struktur *stBuildingAlarms* (siehe globale Variablenliste Site [▶ 1135]) zur Verfügung.

Die Kriterien sind im Einzelnen:

1. Feueralarm (*FireAlert*): dieser Alarm wird direkt über ein binäres Eingangsobjekt eingelesen.
2. Einbruchalarm (*Bgly/Burglary*): dieser Alarm wird direkt über ein binäres Eingangsobjekt eingelesen.
3. Zentrale Abschaltung (*CentSwiOff*), eigentlich kein Alarm. Dieses Eingabeobjekt ermöglicht es, an dieser zentralen Stelle etwa Handmerker gebäudeweit zurückzusetzen oder Licht abzuschalten.
4. Zwangsbelüftung (*ForcedVenilation*): in diesem Template auf Reserve, zur Auslösung einer Zwangsbelüftung.
5. Entrauchung (*SmokeExtraction*): in diesem Template auf Reserve, zur Auslösung eines Entrauchungsbetriebs.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingAlarms EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  FireAlert     : FB_BA_BI_IO;
  Bgly          : FB_BA_BI_IO;
  CentSwiOff    : FB_BA_BV_Op;
END_VAR
```

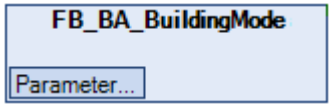
Eingänge CONSTANT

Name	Typ	Beschreibung
FireAlert	FB_BA_BI_IO [▶ 210]	Binäres Eingangsobjekt „Feueralarm“.
Bgly	FB_BA_BI_IO [▶ 210]	Binäres Eingangsobjekt „Einbruch“.
CentSwiOff	FB_BA_BV_Op [▶ 221]	Binäres Eingabeobjekt „Zentralschalter Aus“.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.2.2 FB_BA_BuildingMode



Auswahl des **Gebäudemodus** anhand eines Zeitschaltplans (*Sched*) und eines Handeingriffs (*OpModMan*).

Der Handeingriff hat folgende Modi (*OpModMan*):

Wert	Bedeutung
1	Automatikbetrieb
2	manuelle Anwahl Default
3	manuelle Anwahl Nightwatch
4	manuelle Anwahl Cleaning

Ist **Automatikbetrieb** gewählt, setzt der Baustein *DeMuxManMod* den Ausgang *bQ01.* Da dieser Ausgang nicht mit dem *PrioSwi* verbunden ist, ist der Handbetrieb am *PrioSwi* deaktiviert und der Zeitschaltplan *Sched* ist aktiv. Dieser kann folgende Werte annehmen:

Wert	Bedeutung
1	Default
2	Nightwatch
3	Cleaning

Der aktuell angewählte Gebäudemodus wird dann über den Baustein *OpModPr* zur Anzeige gebracht und per Publisher zur Verfügung gestellt.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingMode EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  OpModMan        : FB_BA_MV_Op;
  OpModPr         : FB_BA_MV_Op;
  Sched           : FB_BA_SchedM;
END_VAR
VAR
  DeMuxManMod     : FB_BA_DMUX_B04;
  DeMuxSched      : FB_BA_DMUX_B04;
  PrioSwi         : FB_BA_PrioSwi_UDI08;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
OpModMan	FB_BA_MV_Op [▶ 244]	Das Multistate-Value-Objekt repräsentiert einen Betriebsartenschalter mit den Betriebsarten Auto, Hand-Aus und Hand-Ein.
OpModPr	FB_BA_MV_Op [▶ 244]	Das Multistate-Value-Objekt zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.
Sched	FB_BA_SchedM [▶ 230]	Zeitschaltobjekt (Automatik) für das Gebäudeenergieniveau „BuildingEnergyLevel“.

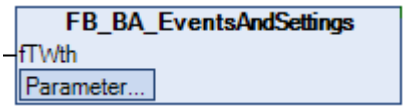
Variablen

Name	Typ	Beschreibung
DeMuxManMode	FB_BA_DMUX_B04 [▶ 650]	Konvertierung des Multistate-Wertes der Handanwahl zu einem Binärausgang.
DeMuxSched	FB_BA_DMUX_B04 [▶ 436]	Konvertierung des Multistate - Wertes des Zeitschaltplans zu einem Binärausgang.
PrioSwi	FB_BA_PrioSwi_UDI08 [▶ 443]	Priorisierende Rückumwandlung der Status zu einem resultierenden Multistate - bzw. Enumerationswert für das Gebäudeenergieniveau.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.2.3 FB_BA_EventsAndSettings



Das Template stellt die Anlagen-Ebene dar.

Es dient der Aufbereitung und Verteilung globaler Daten, welche system- bzw. gebäudeweit für die Steuerung und Regelung der Anlagen- und Raumautomation erforderlich sind.

Mit dem Funktionsbaustein FB_BA_BuildingAlarms [▶ 863] werden globale Ereignisse erfasst, welche in einem Gebäude oder einer Liegenschaft an allen Automationsstationen des Systems benötigt werden.

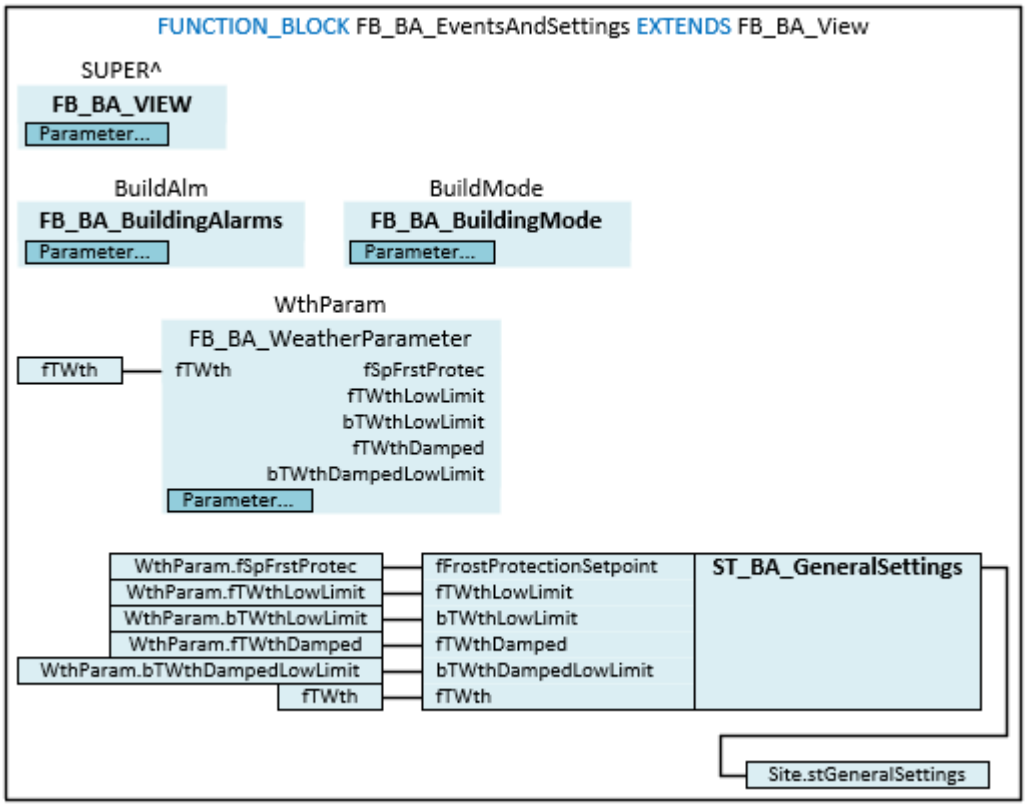
Der Funktionsbaustein FB_BA_WeatherParameter dient der globalen Bereitstellung witterungsspezifischer Parameter und Sollwerte.

Der Funktionsbaustein FB_BA_BuildingMode [▶ 863] dient der globalen Bereitstellung von Gebäudebetriebsarten.

Die Ablage der globalen Gebäudedaten erfolgt in der GVL Site [▶ 1135].

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Illustration

```
FUNCTION_BLOCK FB_BA_BuildingGlobal EXTENDS FB_BA_View
VAR_INPUT
  fTWth        : REAL;
END_VAR
VAR_INPUT CONSTANT
  BuildAlm     : FB_BA_BuildingAlarms;
  BuildMode    : FB_BA_BuildingMode;
  WthParam     : FB_BA_WeatherParameter;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fTWth	REAL	Aktueller Wert der Außentemperatur.

Eingänge CONSTANT

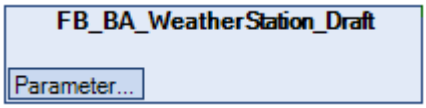
Name	Typ	Beschreibung
BuildAlm	FB_BA_BuildingAlarms [▶ 863]	Template für die Gebäudealarme.
BuildMode	FB_BA_BuildingMode [▶ 863]	Template die Gebäude-Betriebsmodi.
WthParam	FB_BA_WeatherParameter [▶ 876]	Template zur Auswertung der Außentemperatur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

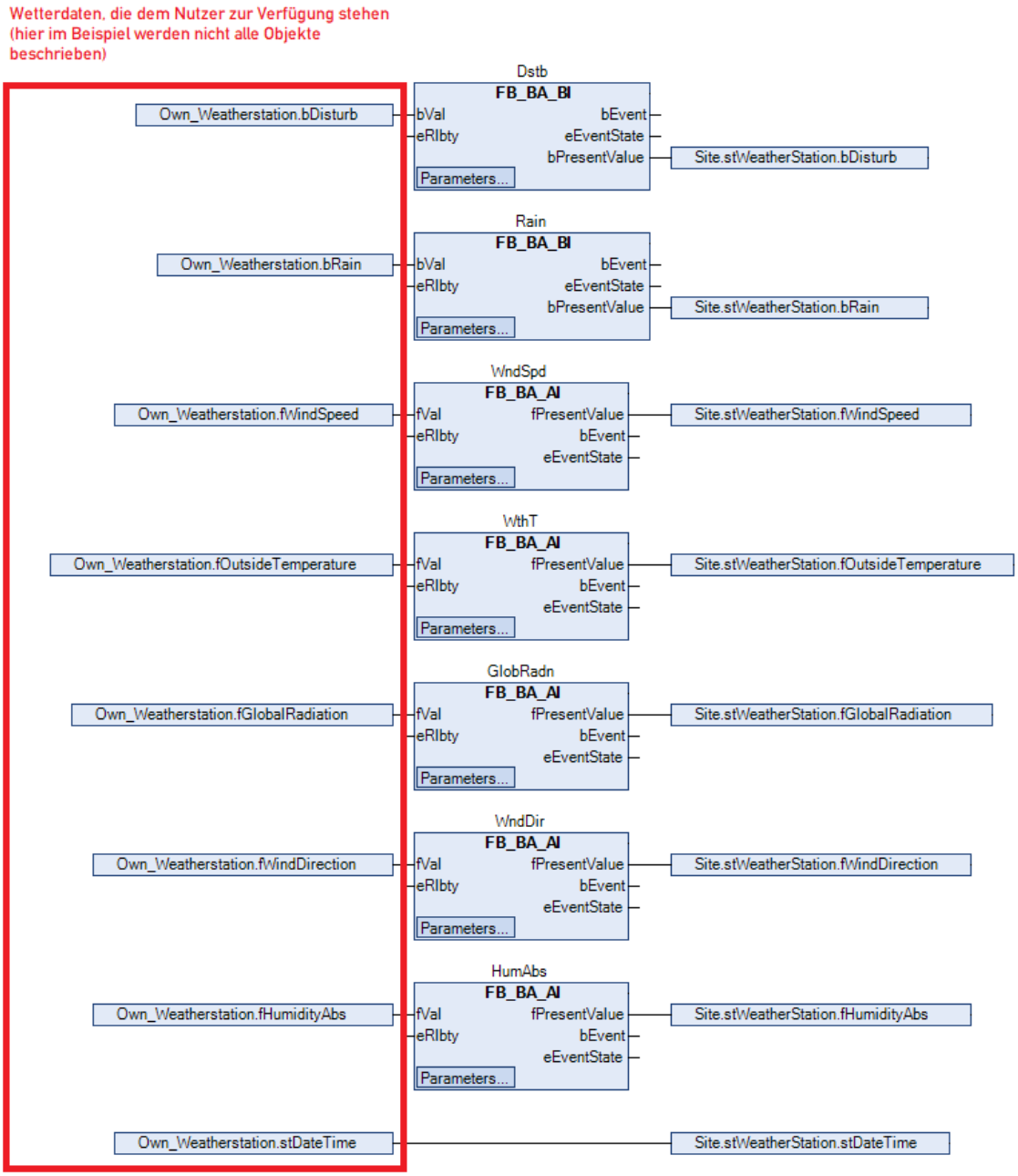
6.1.4.2.1.3.3 WeatherStation

6.1.4.2.1.3.3.1 FB_BA_Weatherstation_Draft



Dieses Template stellt eine Programmiervorlage für eine Wetterstation dar, für die kein vorbereitetes Template zur Verfügung steht. An die Objekte *Dstb … SunElv* sind, sofern vorhanden, die Werte der vorhandenen Wetterstation zu verknüpfen, sowie die Zeit. Für die Zeit vom Typ TIMESTRUCT ist kein Objekt vorhanden, hier muss die Platzhaltervariable ersetzt werden. Diese dient lediglich der fehlerfreien Übersetzung des Basisprojekts.

Die jeweiligen Werte *fPresentValue* werden in einer global deklarierten Variablenstruktur stWeatherstation (siehe Site [▶ 1135]) zusammengefasst.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_WeatherStation_Draft EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  Dstb                      : FB_BA_BI;
  Rain                      : FB_BA_BI;
  WthT                      : FB_BA_AI;
  DewPtT                    : FB_BA_AI;
  PrssAbs                   : FB_BA_AI;
  PrssRel                   : FB_BA_AI;
  HumAbs                    : FB_BA_AI;
  HumRel                    : FB_BA_AI;
  Brightness                : FB_BA_AI;
  Dawn                      : FB_BA_AI;
  GlobRadn                  : FB_BA_AI;
  WndDir                    : FB_BA_AI;
  WndSpd                    : FB_BA_AI;
  Latd                      : FB_BA_AI;
```

```
  Lngt                      : FB_BA_AI;
  SunAzm                    : FB_BA_AI;
  SunElv                    : FB_BA_AI;
END_VAR

VAR
  stDateTime_PLACEHOLDER    : TIMESTRUCT;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
Dstb	FB_BA_BI [▶ 209]	Die Wetterstation meldet eine Störung.
Rain	FB_BA_BI [▶ 209]	Regensensor.
WthT	FB_BA_AI [▶ 195]	Außentemperatur [°C].
DewPtT	FB_BA_AI [▶ 195]	Taupunkttemperatur [°C].
PrssAbs	FB_BA_AI [▶ 195]	Absoluter Luftdruck [hPa].
PrssRel	FB_BA_AI [▶ 195]	Relativer Luftdruck [hPa].
HumAbs	FB_BA_AI [▶ 195]	Absolute Feuchte [g/m³].
HumRel	FB_BA_AI [▶ 195]	Relative Absolute Feuchte [g/m³].
Brightness_N	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Norden [Lux].
Brightness_S	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Süden [Lux].
Brightness_E	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Osten [Lux].
Brightness_W	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Westen [Lux].
Dawn	FB_BA_AI [▶ 195]	Dämmerung [Lux].
GlobRadn	FB_BA_AI [▶ 195]	Globalstrahlung [W/m²].
WndDir	FB_BA_AI [▶ 195]	Windrichtung [°].
WndSpd	FB_BA_AI [▶ 195]	Windgeschwindigkeit [m/s]
Latd	FB_BA_AI [▶ 195]	Geographische Breite des Aufstellungsortes [°].
Lngt	FB_BA_AI [▶ 195]	Geographische Länge des Aufstellungsortes [°].
SunAzm	FB_BA_AI [▶ 195]	Aktueller Sonnenstand [°].
SunElv	FB_BA_AI [▶ 195]	Aktuelle Sonnenhöhe [°].

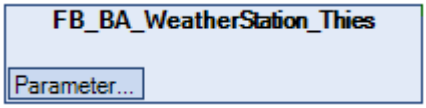
Variablen

Name	Typ	Beschreibung
stDateTime_PLACEHOLDER	TIMESTRUCT	Platzhalter-Variable: Statt dieser Variable kann eine geeignete Zeitstruktur verknüpft werden, welche die aktuelle Zeit über die Wetterstation zur Verfügung stellt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.3.2 FB_BA_Weatherstation_Thies



Dieses Template bereitet die Daten, welche über den Baustein FB_BA_ThiesWSC11 [▶ 1133] aus einer Thies-Wetterstation gelesen werden, auf und stellt sie in einer global deklarierten Variablenstruktur *stWeatherstation* (siehe globale Variablenliste Site [▶ 1135]) zur Verfügung.

Darüber hinaus wird in diesem Template die Hardwareanbindung über die Variablen *stRawDataIn* und *stRawDataOut* hergestellt.

Hier finden Sie eine detaillierte Beschreibung zur Einbindung der Wetterstation [▶ 1116].



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_WeatherStation_Thies EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  SerialCommRst                : FB_BA_Bv_OP;
  WthStRst                     : FB_BA_Bv_OP;
  SerialCommErr                : FB_BA_BV;
  Dstb                         : FB_BA_BV;
  Rain                         : FB_BA_BI;
  WthT                         : FB_BA_AI;
  DewPtT                       : FB_BA_AI;
  PrssAbs                      : FB_BA_AI;
  PrssRel                      : FB_BA_AI;
  HumAbs                       : FB_BA_AI;
  HumRel                       : FB_BA_AI;
  Brightness_N                 : FB_BA_AI;
  Brightness_S                 : FB_BA_AI;
  Brightness_E                 : FB_BA_AI;
  Brightness_W                 : FB_BA_AI;
  Dawn                         : FB_BA_AI;
  GlobRadn                     : FB_BA_AI;
  WndDir                       : FB_BA_AI;
  WndSpd                       : FB_BA_AI;
  Latd                         : FB_BA_AI;
  Lngt                         : FB_BA_AI;
  SunAzm                       : FB_BA_AI;
  SunElv                       : FB_BA_AI;
END_VAR

VAR
  ThiesWSC11                   : FB_BA_ThiesWSC11;
  bResetWeatherStation         : BOOL;
  bResetSerialCommunication    : BOOL;
  DataConversion               : FB_BA_ThiesData;
  tonSerialCommErr             : TON;
  tonDstb                      : TON;

  stRawDataIn     AT %I*       : KL6InData22B;
  stRawDataOut    AT %Q*       : KL6OutData22B;
  fbSerialCtrl                 : SerialLineControl;
  fbKL6Configuration           : KL6configuration;
  stTxBuff                     : ComBuffer;
  stRxBuff                     : ComBuffer;

  bSerialConfigError           : BOOL;
  nSerialConfigErrorID         : UDINT;
  bSerialCommError             : BOOL;
  nSerialCommErrorID           : UDINT;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
{attribute 'parameterUnit':= 's'}
  nSerialCommErrDelay          : UDINT := 10;
{attribute 'parameterUnit':= 's'}
  nDstbDelay                   : UDINT := 10;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
SerialCommRst	FB_BA_Bv_OP [▶ 221]	Neustarten der seriellen Kommunikation, falls diese durch einen Konfigurationsfehler oder während des Betriebes dauerhaft in Störung gegangen ist.
WthStRst	FB_BA_Bv_OP [▶ 221]	Neustarten der Wetterstation selbst, falls diese dauerhaft in Störung gegangen ist. Hierbei wird die SPS-Routine beginnend mit der Konfiguration und anschließender zyklischer Abfrage neu gestartet. Ein Hardware-Reset erfolgt nicht.
SerialCommErr	FB_BA_BV [▶ 218]	Quittierpflichtige Meldung: Serielle Kommunikation oder Konfiguration dauerhaft in Störung.
Dstb	FB_BA_BV [▶ 218]	Die Wetterstation meldet eine Störung.
Rain	FB_BA_BI [▶ 209]	Regensensor.
WthT	FB_BA_AI [▶ 195]	Außentemperatur [°C].
DewPtT	FB_BA_AI [▶ 195]	Taupunkttemperatur [°C].
PrssAbs	FB_BA_AI [▶ 195]	Absoluter Luftdruck [hPa].
PrssRel	FB_BA_AI [▶ 195]	Relativer Luftdruck [hPa].
HumAbs	FB_BA_AI [▶ 195]	Absolute Feuchte [g/m^3].
HumRel	FB_BA_AI [▶ 195]	Relative Absolute Feuchte [g/m^3].
Brightness_N	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Norden [Lux].
Brightness_S	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Süden [Lux].
Brightness_E	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Osten [Lux].
Brightness_W	FB_BA_AI [▶ 195]	Richtungsabhängiger Helligkeitssensor Westen [Lux].
Dawn	FB_BA_AI [▶ 195]	Dämmerung [Lux].
GlobRadn	FB_BA_AI [▶ 195]	Globalstrahlung [W/m²].
WndDir	FB_BA_AI [▶ 195]	Windrichtung [°].
WndSpd	FB_BA_AI [▶ 195]	Windgeschwindigkeit [m/s]
Latd	FB_BA_AI [▶ 195]	Geographische Breite des Aufstellungsortes [°].
Lngt	FB_BA_AI [▶ 195]	Geographische Länge des Aufstellungsortes [°].
SunAzm	FB_BA_AI [▶ 195]	Aktueller Sonnenstand [°].
SunElv	FB_BA_AI [▶ 195]	Aktuelle Sonnenhöhe [°].

Variablen

Name	Typ	Beschreibung
ThiesWSC11	FB_BA_ThiesWSC11 [▶ 1133]	Baustein zum Ausladen der Daten aus der Thies WSC11
bResetWeatherStation	BOOL	Reset Wetterstation.
bResetSerialCommunication	BOOL	Reset serielle Kommunikation.
DataConversion	FB_BA_ThiesData [▶ 1134]	Wandelt die Thies-Wetterstationsdaten in projektspezifische Daten um. Beispiel Helligkeit: Umwandlung von kLux in Lux, da nur diese Einheit von BACnet unterstützt wird.
tonSerialCommErr	TON	Fehlerverzögerung Kommunikationsfehler.
tonDstb	TON	Fehlerverzögerung Wetterstationsfehler
stRawDataIn	KL6InData22B	Eingangs-Rohwerte von der seriellen Klemme.
stRawDataOut	KL6InData22B	Ausgangs-Rohwerte zur seriellen Klemme.
fbSerialCtrl	SerialLineControl	Kommunikationsbaustein zur seriellen Klemme. Läuft in der schnellen Task unter der Methode FastCycle.
fbKL6Configuration	KL6configuration	Konfigurationsbaustein zur seriellen Klemme. Läuft in der schnellen Task unter der Methode FastCycle. Hiermit werden die seriellen Parameter (Baud-Rate, etc.) eingestellt, nicht aber das Prozessabbild (siehe Thies Wetterstation [▶ 1116]).
stTxBuff	ComBuffer	Kommunikationsvariable zwischen dem *fbSerialCtrl* der schnellen Task und dem *ThiesWSC11* der normalen SPS-Task.
stRxBuff	ComBuffer	Kommunikationsvariable zwischen dem *fbSerialCtrl* der schnellen Task und dem *ThiesWSC11* der normalen SPS-Task.
bSerialConfigError	BOOL	Konfigurationsfehler der seriellen Kommunikation.
nSerialConfigErrorID	UDINT	Fehlernummer.
bSerialCommError	BOOL	Kommunikationsfehler der seriellen Kommunikation.
nSerialCommErrorID	UDINT	Fehlernummer.

Eingänge CONSTANT PERSISTENT

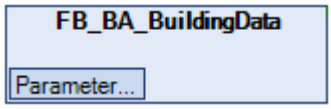
Name	Typ	Beschreibung
nSerialCommErrDelay	UDINT	Fehlerverzögerung Kommunikationsfehler [s].
nDstbDelay	UDINT	Fehlerverzögerung Wetterstationsfehler [s].

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.4 General

6.1.4.2.1.3.4.1 FB_BA_BuildingData

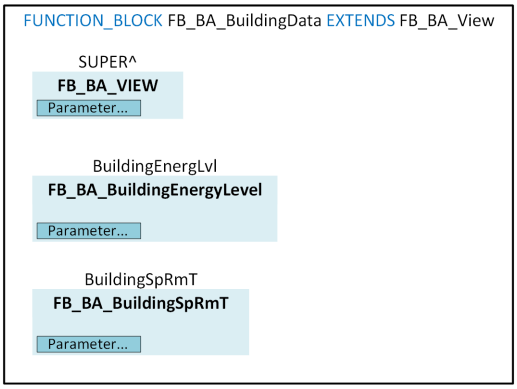


Dieses Template auf Anlagenebene dient zu Aufruf von Sub-Templates, welche gebäudeweite Daten und Betriebsarten zur Verfügung stellen.

Typischerweise wird dieses Template im Gebäuderechner aufgerufen.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
VAR_INPUT CONSTANT
  BuildingMode        : FB_BA_BuildingMode;
  BuildingEnergLvl    : FB_BA_BuildingEnergyLevel;
  BuildingSpRmT       : FB_BA_BuildingSpRmT;
  BuildingLight       : FB_BA_BuildingLighting;
END_VAR
```

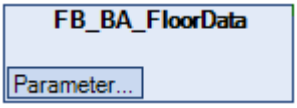
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
BuildingMode	FB_BA_BuildingMode [▶ 863]	Das Untertemplate *BuildingMode* definiert anhand eines Zeitschaltplanes mit Handeingriff die Gebäudebetriebsart.
BuildingEnergLvl	FB_BA_BuildingEnergyLevel [▶ 903]	Das Untertemplate *BuildingEnergLvl* definiert anhand eines Zeitschaltplanes mit Handeingriff das Gebäude-Energieniveau.
BuildingSpRmT	FB_BA_BuildingSpRmT [▶ 873]	Das Untertemplate *BuildingSpRmT* dient zur Eingabe der Raumtemperatursollwerte Heizen / Kühlen für die verschiedenen Gebäude-Energieniveaus. Darüber hinaus wird anhand der Außentemperatur jeweils ein Sommer- und ein Winterkompensationswert für die Raumtemperatur ermittelt.
BuildingLight	FB_BA_BuildingLighting [▶ 933]	Das Untertemplate *BuildingLight* stellt gebäudeübergreifende Steuertelegramme für die Lichtfunktionen zusammen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.4.2 FB_BA_FloorData

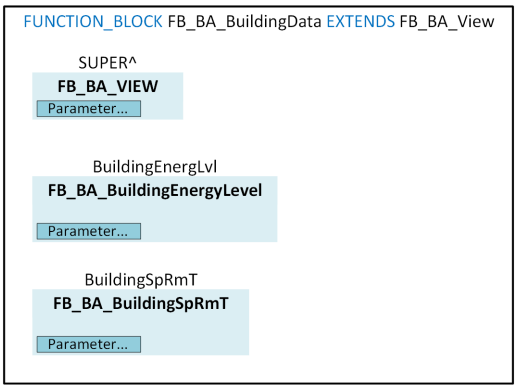


Dieses Template auf Anlagenebene dient zu Aufruf von Sub-Templates, welche etagenweite Daten und Betriebsarten zur Verfügung stellen.

Typischerweise wird dieses Template in den Etagencontrollern aufgerufen.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
VAR_INPUT CONSTANT
  FloorLighting    : FB_BA_FloorLighting;
END_VAR
```

VAR_INPUT CONSTANT

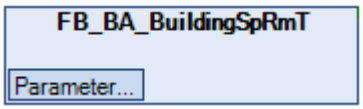
Name	Typ	Beschreibung
FloorLighting	FB_BA_FloorLighting [▶ 935]	Das Untertemplate ruft der Beleuchtung zugeordnete Funktionen im Etagenbereich auf.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.4.3 Functions

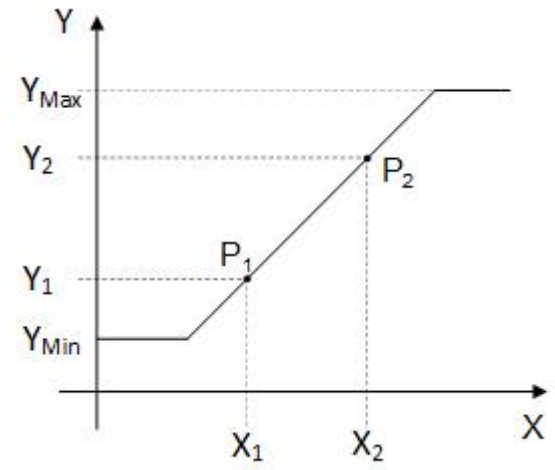
6.1.4.2.1.3.4.3.1 FB_BA_BuildingSpRmT



In diesem Template werden die Grundsollwerte Protection Cooling .. Comfort Cooling und Protection Heating .. Comfort Heating mit Korrekturwerten beaufschlagt und als gebäudeweite Sollwerte in einer global deklarierten Strukturvariable *stBuildingSpRmT* (siehe globale Variablenliste Site [▶ 1135]) zur Verfügung gestellt.

Die Korrektur wirkt sich dabei jeweils auf die Precomfort- und Comfort-Werte aus, siehe FB_BA_SpRmT [▶ 312] . Es handelt sich dabei um:

- **zentrale Sollwertverschiebung**
  Diese beiden Werte, ShiftHtg und ShiftCol, werden jeweils auf die entsprechenden Werte für Precomfort und Comfort aufaddiert.
- **witterungsgeführte Sollwertverschiebung (Sommer- bzw. Winterkompensation)**
  Anhand der Außentemperatur aus den Wetterstationsdaten *stWeatherStation* (siehe globale Variablenliste Site [▶ 1135]) werden hier noch einmal Korrekturwerte errechnet, welche auf die Werte für Precomfort und Comfort addiert werden.
  Die Berechnung erfolgt über die Interpolationsbausteine FB_BA_Scale_02 [▶ 1060], die sich in der *FB_init*-Methode parametrieren lassen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingSpRmT EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  SumCpsn      : FB_BA_Scale_02;
  WinCpsn      : FB_BA_Scale_02;
  PrtcHtg      : FB_BA_AV_Op;
  EcoHtg       : FB_BA_AV_Op;
  PreCmfHtg    : FB_BA_AV_Op;
  CmfHtg       : FB_BA_AV_Op;
  PrtcCol      : FB_BA_AV_Op;
  EcoCol       : FB_BA_AV_Op;
  PreCmfCol    : FB_BA_AV_Op;
  CmfCol       : FB_BA_AV_Op;
  ShiftHtg     : FB_BA_AV_Op;
  ShiftCol     : FB_BA_AV_Op;
END_VAR

VAR
  SpRmT        : FB_BA_SpRmT;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
SumCpsn	FB_BA_Scale_02 [▶ 1060]	Wert Sommerkompensation [K]
WinCpsn	FB_BA_Scale_02 [▶ 1060]	Wert Winterkompensation [K]
PrtcHtg	FB_BA_AV_Op [▶ 207]	Grundsollwert Protection Heating [°C]
EcoHtg	FB_BA_AV_Op [▶ 207]	Grundsollwert Economy Heating [°C]
PreCmfHtg	FB_BA_AV_Op [▶ 207]	Grundsollwert Precomfort Heating [°C]
CmfHtg	FB_BA_AV_Op [▶ 207]	Grundsollwert Comfort Heating [°C]
PrtcCol	FB_BA_AV_Op [▶ 207]	Grundsollwert Protection Cooling [°C]
EcoCol	FB_BA_AV_Op [▶ 207]	Grundsollwert Economy Cooling [°C]
PreCmfCol	FB_BA_AV_Op [▶ 207]	Grundsollwert Precomfort Cooling [°C]
CmfCol	FB_BA_AV_Op [▶ 207]	Grundsollwert Comfort Cooling [°C]
ShiftHtg	FB_BA_AV_Op [▶ 207]	Sollwertverschiebung Heizen [K]
ShitfCol	FB_BA_AV_Op [▶ 207]	Sollwertverschiebung Kühlen [K]

Variablen

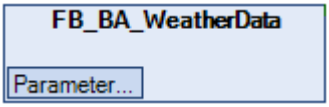
Name	Typ	Beschreibung
SpRmT	FB_BA_SpRmT [▶ 312]	Baustein zur Berechnung der beaufschlagten Temperatursollwerte.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.5 WeatherData

6.1.4.2.1.3.5.1 FB_BA_WeatherData



Das Template dient der Aufbereitung und Verteilung globaler Daten, welche system- bzw. gebäudeweit für die Steuerung und Regelung der Anlagenautomation erforderlich sind.

Der Funktionsbaustein FB_BA_WeatherParameter [▶ 876] dient der globalen Bereitstellung witterungsspezifischer Parameter und Sollwerte.

Die Ablage der globalen Gebäudedaten erfolgt in der GVL Site [▶ 1135].

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Vererbungshierarchie

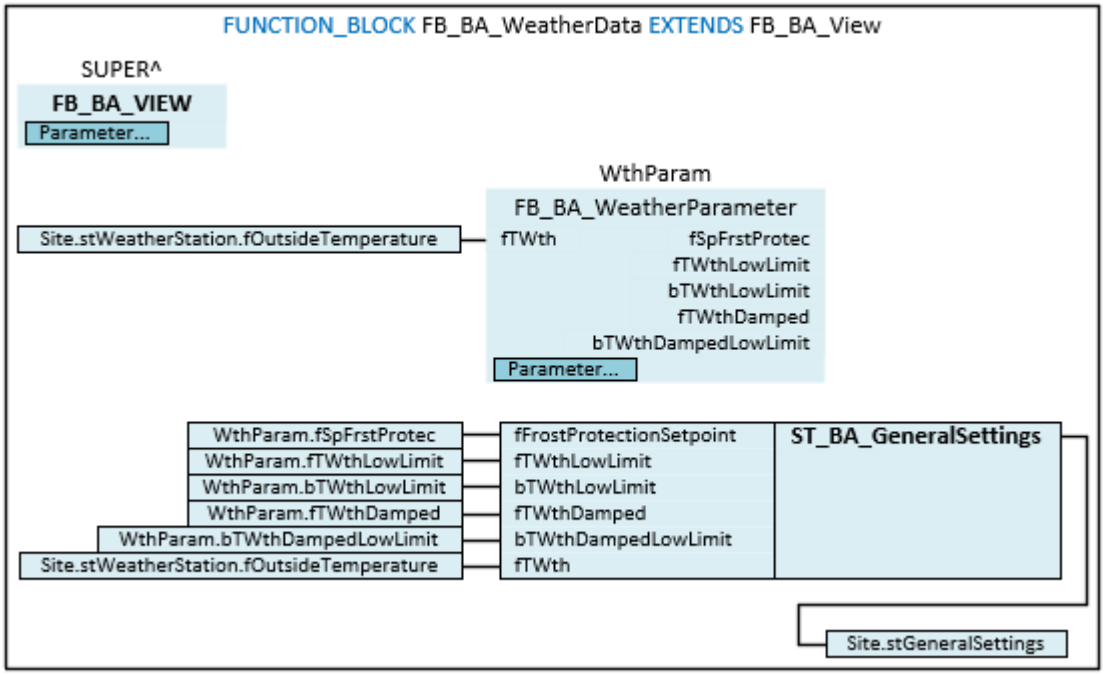
FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_View [▶ 246]

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_WeatherData EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  WthParam    : FB_BA_WeatherParameter;
END_VAR
```

Eingänge CONSTANT

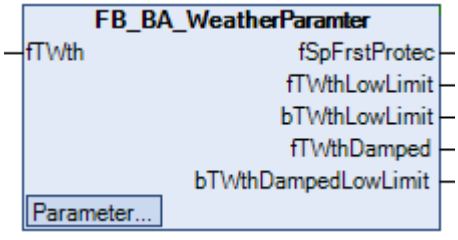
Name	Typ	Beschreibung
WthParam	FB_BA_WeatherParameter [▶ 876]	Sub-Template zur Auswertung und Erstellung der Witterungsparameter.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.5.2 Parameter

6.1.4.2.1.3.5.2.1 FB_BA_WeatherParameter



Das Template berechnet aus dem Messwert der Außentemperatur verschiedene Werte, welche Systemweit für die Steuerung und Regelung von Anlagen in der Heizungs-, Lüftungs- und Klimatechnik benötigt werden.

Das Objekt *SpFrstProtec* dient der Eingabe eines Frostschutzsollwertes. Der Frostschutzsollwert wird systemweit in allen HLK-Templates mit einem wasserseitigen Frostschutz verwendet.

Der Funktionsbaustein *TWthLowLimitHys* errechnet, ob die Außentemperatur unterhalb des kritischen Wertes von *TWthLowLimitVal* liegt. In Abhängigkeit des Wertes von der Variablen *bTWthLowLimit* wird die Frostschutzfunktion der HLK-Anlagen in dem Gebäude aktiviert.

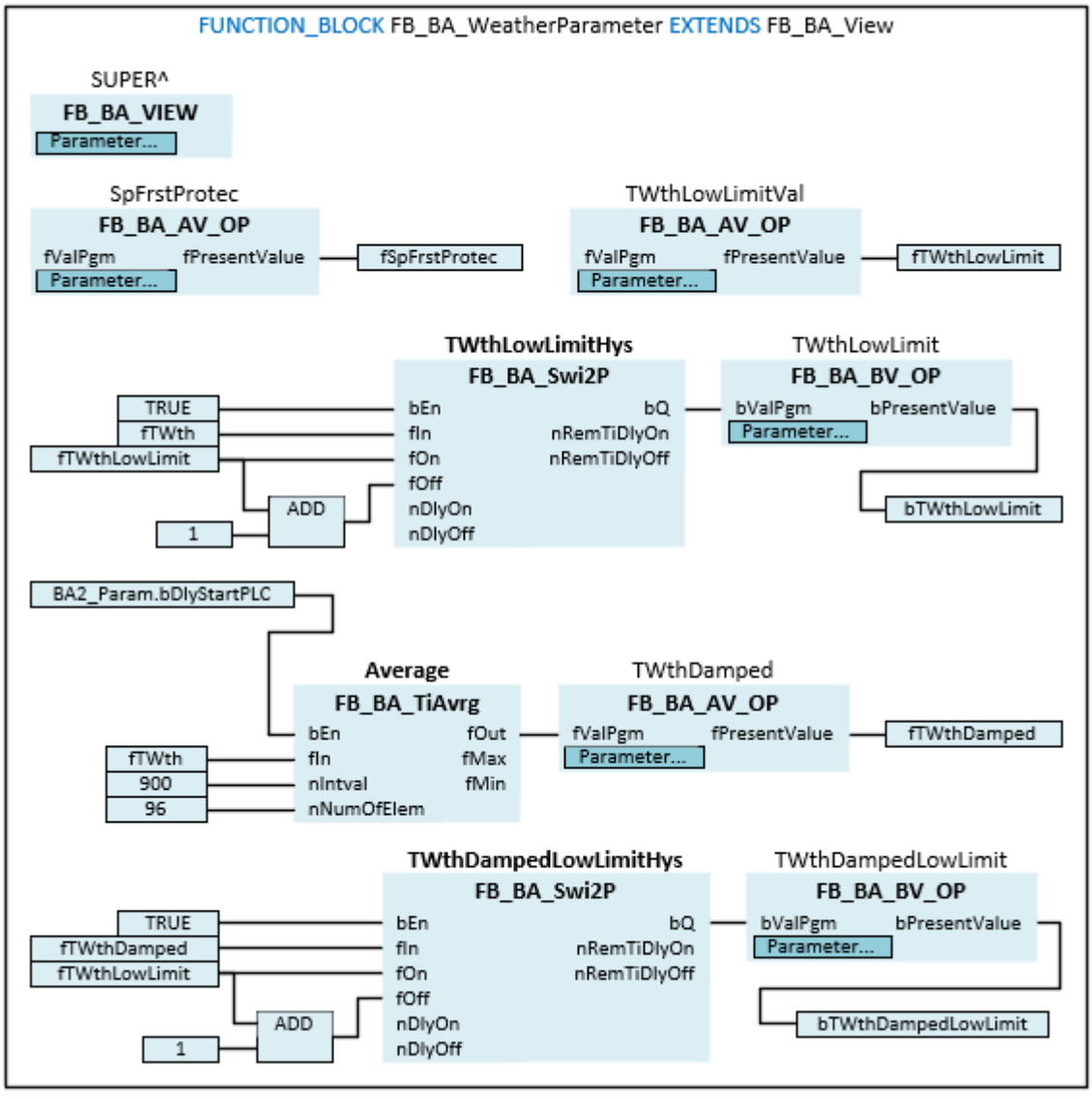
Die witterungsbedingte Freigabe der Heizungsanlagen erfolgt nicht in Abhängigkeit der aktuellen, sondern in Abhängigkeit der gedämpften Außentemperatur. Die Dämpfung der Außentemperatur erfolgt mit dem Funktionsbaustein *Average* (Bildung eines gleitenden Mittelwerts). Das Objekt *TWthDamped* dient der Anzeige des Wertes der gedämpften Außentemperatur.

Mit dem Funktionsbaustein *TWthDampedLowLimitHys* wird überprüft, ob sich die gedämpfte Außentemperatur unterhalb eines Wertes befinden, ab dem Heizungsanlagen in einem Gebäude frei gegeben werden sollen. Die globale witterungsbedingte Freigabe wird mit dem Objekt "TWthDampedLowLimit" angezeigt und zur Weiterverarbeitung in anderen Templates auf die Variable *bTWthDampedLowLimit* geschrieben.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Settings EXTENDS FB_BA_View
VAR_INPUT
  fTWth                    : REAL;
END_VAR
VAR_OUTPUT
  fSpFrstProtec            : REAL;
  fTWthLowLimit            : REAL;
  bTWthLowLimit            : BOOL;
  fTWthDamped              : REAL;
  bTWthDampedLowLimit      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SpFrstProtec             : FB_BA_AV_Op;
  TWthLowLimitVal          : FB_BA_AV_Op;
  TWthLowLimit             : FB_BA_BV_Op;
  TWthDamped               : FB_BA_AV_Op;
  TWthDampedLowLimit       : FB_BA_BV_Op;
END_VAR
VAR
  TWthLowLimitHys          : FB_BA_Swi2P;
  Average                  : FB_BA_TiAvrg;
  TWthDampedLowLimitHys    : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fTWth	REAL	Aktueller Wert der Außentemperatur.

Ausgänge

Name	Typ	Beschreibung
fSpFrstProtec	REAL	Frostschutzsollwert, z. B. für Heizkreise im Schutzmodus.
fTWthLowLimit	REAL	Unterer Grenzwert der Außentemperatur. Unterhalb dieses Wertes werden alle Frostschutzfunktionen in HVAC-Anlagen aktiviert.
bTWthLowLimit	BOOL	Der Ausgang zeigt die Betriebsmeldung *Außentemperatur unterer Grenzwert* an. Die Variable ist TRUE, wenn die Außentemperatur unter dem Wert von *fTWthLowCrit* liegt.
fTWthDamped	REAL	Aktueller Wert der gedämpften Außentemperatur.
bTWthDampedLowLimit	BOOL	Der Ausgang zeigt die Betriebsmeldung *Außentemperatur gedämpft unterer Grenzwert* an. Die Variable ist TRUE, wenn die gedämpfte Außentemperatur unter dem Wert von *fTWthLowCrit* liegt.

Eingänge CONSTANT

Name	Typ	Beschreibung
SpFrstProtec	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des Frostschutzsollwert, z. B. für Heizkreise im Schutzmodus.
TWthLowLimitVal	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des unteren Grenzwertes der Außentemperatur. Unterhalb dieses Wertes werden alle Frostschutzfunktionen in HVAC-Anlagen aktiviert.
TWthLowLimit	FB_BA_BV_Op [▶ 221]	Binäres Objekt, welches die Betriebsmeldung *Außentemperatur unterer Grenzwert* anzeigt.
TWthDamped	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Anzeige der gedämpften Außentemperatur. Unterhalb dieses Wertes werden alle Frostschutzfunktionen in HVAC-Anlagen aktiviert.
TWthDampedLowLimit	FB_BA_BV_Op [▶ 221]	Binäres Objekt, welches die Betriebsmeldung *Außentemperatur gedämpft unterer Grenzwert* anzeigt.

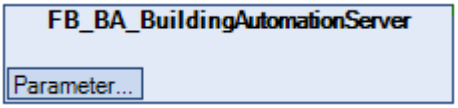
Variablen

Name	Typ	Beschreibung
TWthLowLimitHys	FB_BA_Swi2P	Zweipunktschalter, welcher die Außentemperatur *fWth* in ein binäres Schaltsignal für *TWthLowLimit* wandelt.
Average	FB_BA_TiAvrg [▶ 474]	Die Mittelwertbildung Average ermittelt anhand der Außentemperatur *fWth* die gedämpfte Außentemperatur *fTWthDamped*.
TWthDampedLowLimitHys	FB_BA_Swi2P	Zweipunktschalter, welcher die gedämpfte Außentemperatur *fTWthDamped* in ein binäres Schaltsignal für *TWthDampedLowLimit* wandelt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.3.6 FB_BA_BuildingAutomationServer



Aufruftemplate Gewerk "Gebäudeautomation" - Server-Ausführung.

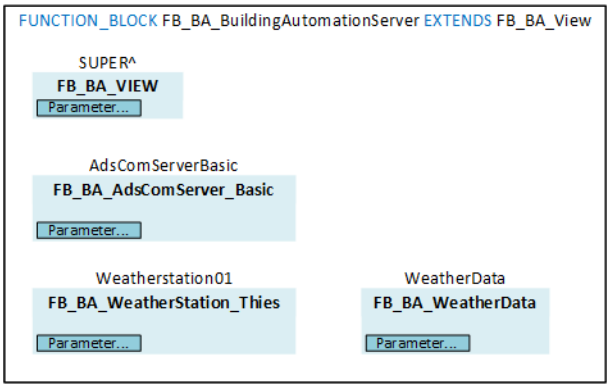
Dieses Template auf der Ebene "Gewerk" ruft alle Untertemplates auf, deren Daten idealerweise in einem Gebäuderechner aufbereitet werden.

Darüber hinaus werden durch ein ADS-Server-Template FB_BA_AdsComServer_Basic [▶ 860] grundlegende Daten gebäudeweit zur Verfügung gestellt.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_BuildingAutomationServer EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComServerBasic    : FB_BA_AdsComServer_Basic;
  Weatherstation01     : FB_BA_WeatherStation_Thies;
  WeatherData          : FB_BA_WeatherData;
END_VAR
```

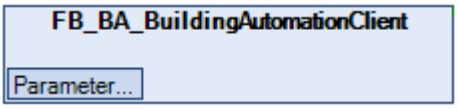
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
AdsComServerBasic	FB_BA_AdsComServer_Basic [▶ 860]	Das Untertemplate hat die Aufgabe Daten und Parameter innerhalb des GA-Netzwerkes zu übermitteln.
Weatherstation01	FB_BA_WeatherStation_Thies [▶ 868]	Das Untertemplate dient zum Datenaustausch mit einer Wetterstation vom Typ WSC11 der Firma Thies. Die Endung "01" am Instanznamen soll darauf hinweisen, dass es durchaus möglich ist, dass ein Gebäude mehrere Wetterstationen haben kann, welche dann an dieser Stelle einzupflegen sind.
WeatherData	FB_BA_WeatherData [▶ 875]	Das Untertemplate dient der Aufbereitung und Verteilung globaler Daten welche system- bzw. gebäudeweit für die Steuerung und Regelung der Anlagenautomation erforderlich sind.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.3.7 FB_BA_BuildingAutomationClient



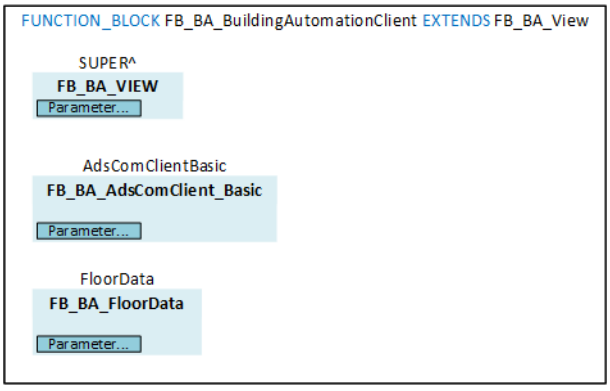
Aufruftemplate Gewerk "Gebäudeautomation" - Client-Ausführung.

Dieses Template auf der Ebene "Gewerk" ruft alle Untertemplate auf, deren Daten idealerweise in einem Etagenrechner aufbereitet werden.

Das sind zum einen Etagendaten aus dem Template FloorData [▶ 873], sowie grundlegende Daten, welche von einem Gebäuderechner übermittelt, gelesen werden (FB_BA_AdsComClient_Basic [▶ 861]).

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_BuildingAutomation EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComClientBasic    : FB_BA_AdsComClient_Basic;
  FloorData            : FB_BA_FloorData;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
AdsComClientBasic	FB_BA_AdsComClient_Basic [▶ 861]	Das Untertemplate hat die Aufgabe Daten und Parameter zu lesen, welche von anderen Rechnern, idealerweise vom Gebäuderechner, gesendet wurden.
FloorData	FB_BA_FloorData [▶ 873]	Das Untertemplate ruft weitere Templates auf, welche etagenweite Daten und Betriebsarten zur Verfügung stellen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4 RoomAutomation

Templates zur Erstellung von Raumautomationslösungen.

6.1.4.2.1.4.1 Common

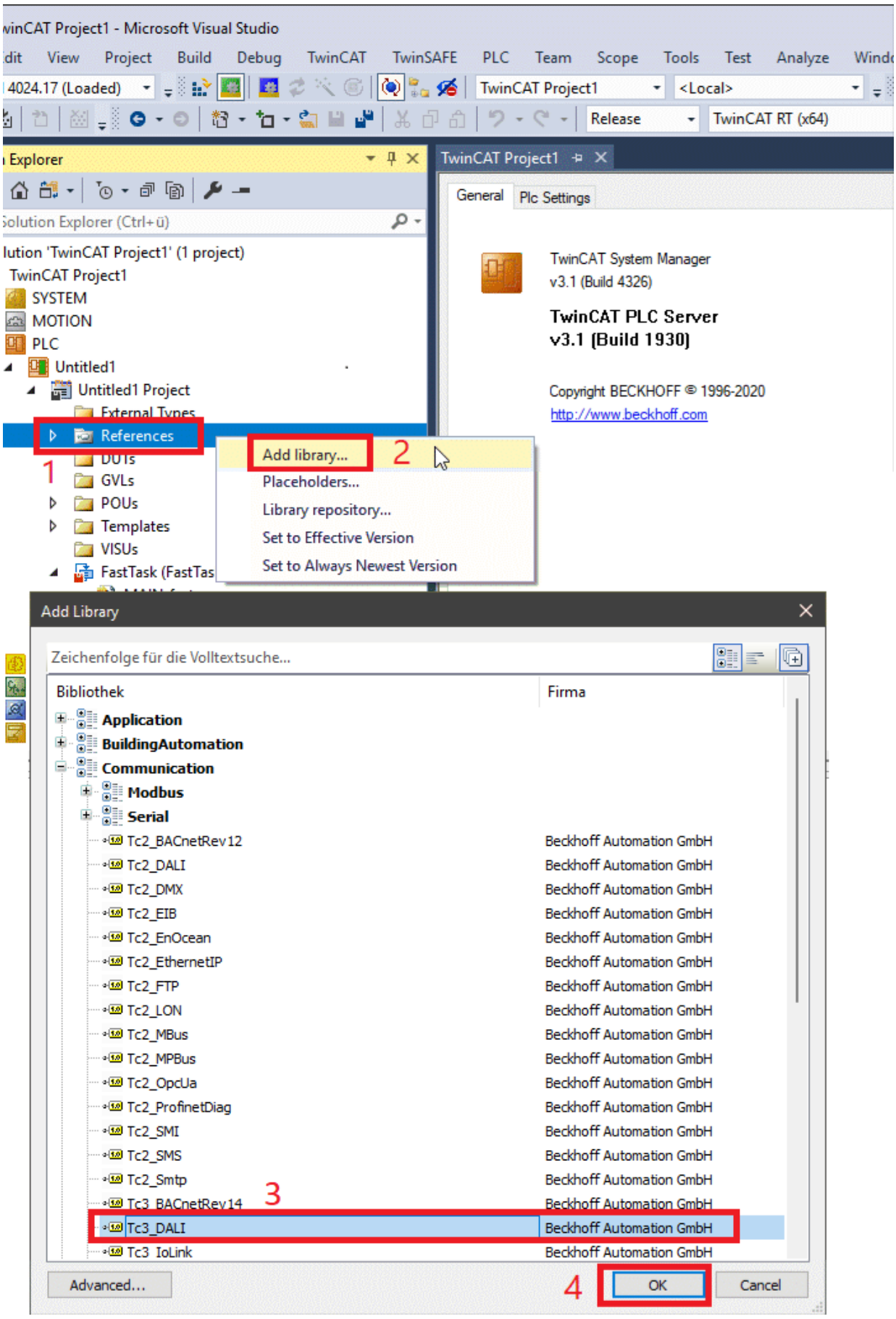
6.1.4.2.1.4.1.1 DALIFunctionBlocks

6.1.4.2.1.4.1.1.1 DALI-Kommunikation

Diese Anleitung bzw. die zur Verfügung gestellten DALI-Templates beziehen sich auf die Anwendung mit einer KL6821 im Zusammenspiel mit der Bibliothek TC3_DALI.

Hinzufügen der Bibliothek

Das alleinige Hinzufügen von DALI-basierten Templates in ein Projekt bindet nicht automatisch die benötigte TC3_DALI-Bibliothek mit ein. Diese muss manuell eingefügt werden:



Mit einem Rechtsklick auf "Referenzen" (1) "Bibliothek hinzufügen" (2) auswählen. In dem sich öffnenden Fenster die Bibliothek "TC3_DALI" wählen (3) und mit "OK" bestätigen (4).

Kommunikationsbausteine und schnelle Task

Alle DALI-basierten Templates benötigen den Kommunikationsbaustein FB_KL6821Communication, der in einer schnellen Kommunikationstask läuft und die Verbindung zu der DALI-Klemme herstellt, an dem die DALI-Linie des betreffenden Gerätes angeschlossen ist.

Die genaue Vorgehensweise ist im Beckhoff Information System innerhalb der Beschreibung der TC3_DALI zu finden.

Pro DALI-Linie ist dabei ein Kommunikationsbaustein zu instanziieren. Sind diese in der schnellen Task eingefügt und aufgerufen, stehen sie den DALI-basierten Templates zur Verfügung.

Verknüpfung im Projekt

Der Datenaustausch zwischen dem Kommunikationsbaustein und den DALI-basierten Templates erfolgt über ein Interface, das vom Kommunikationsbaustein zur Verfügung gestellt wird. Alle DALI-basierten Templates haben dieses Interface als Parameter (VAR_INPUT CONSTANT PERSISTENT) implementiert, so dass dort, im FB_Init, immer nur der jeweilige Kommunikationsbaustein verknüpft werden muss.

Beispiel:

In einer schnellen Task werden für 4 DALI-Linien Kommunikationsbausteine aufgerufen:

DALICommunication

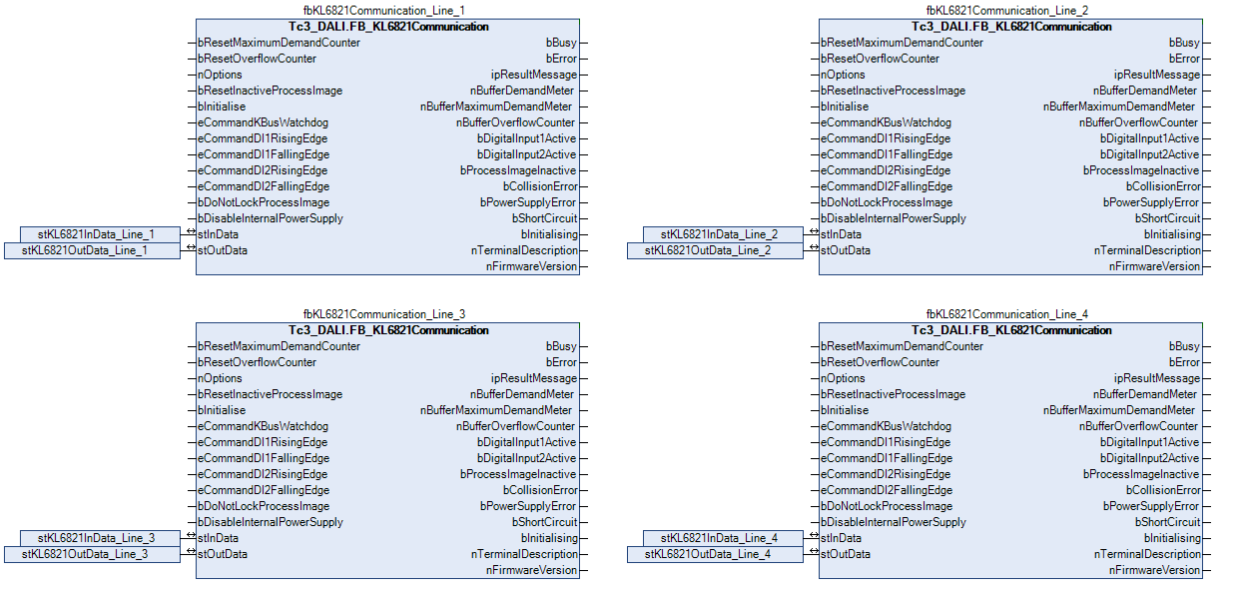
```
PROGRAM DALICommunication
VAR
    stKL6821InData_Line_1          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_1         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_1           : Tc3_DALI.FB_KL6821Communication;

    stKL6821InData_Line_2          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_2         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_2           : Tc3_DALI.FB_KL6821Communication;

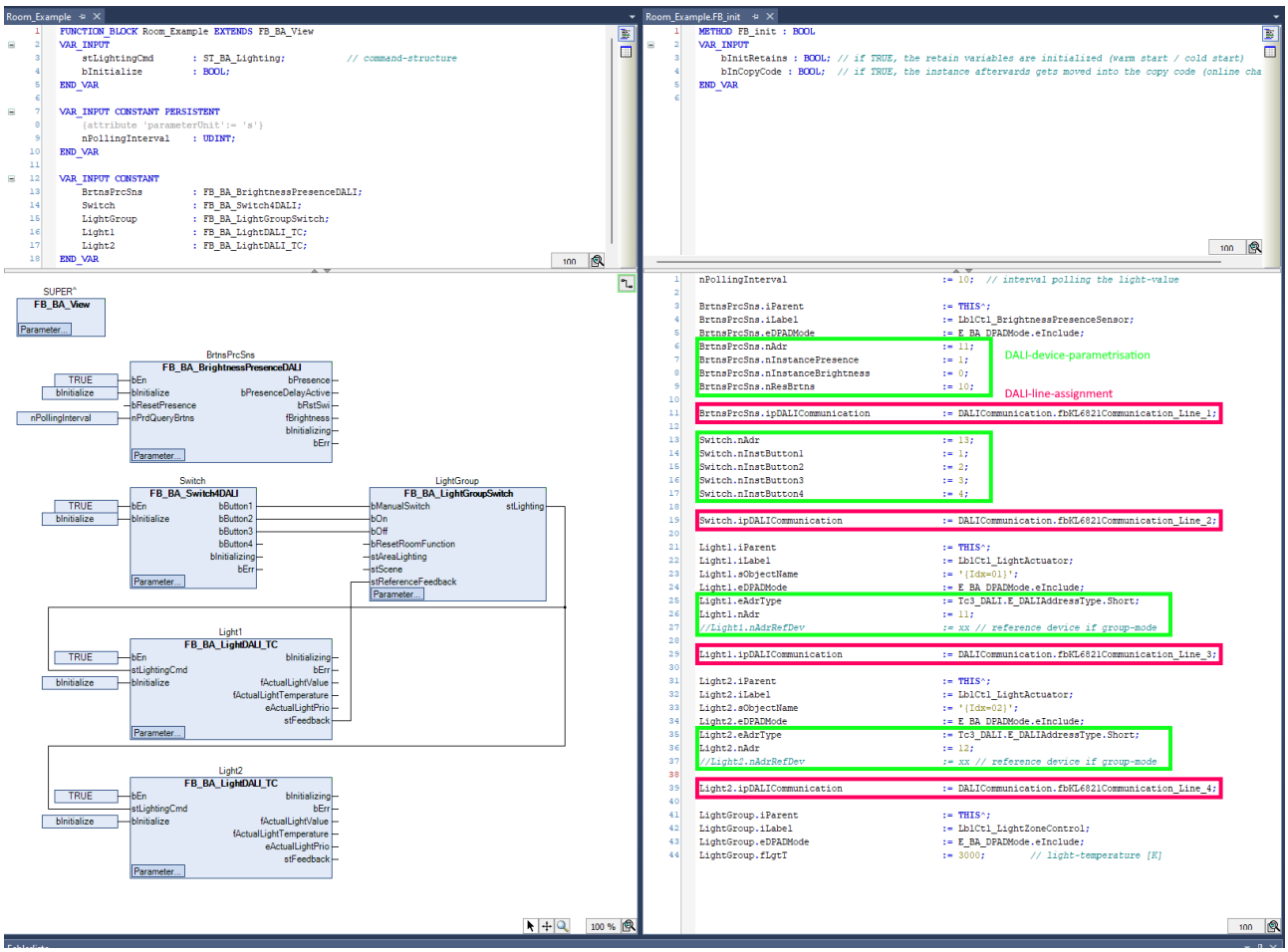
    stKL6821InData_Line_3          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_3         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_3           : Tc3_DALI.FB_KL6821Communication;

    stKL6821InData_Line_4          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData_Line_4         AT %Q*  : Tc3_DALI.ST_KL6821OutData;
    fbKL6821Communication_Line_4           : Tc3_DALI.FB_KL6821Communication;

END_VAR
```



In einem Raum-Template werden dann DALI-Templates aufgerufen. Diese werden im FB_Init des Raumtemplates vorinitialisiert unter anderem auch mit der Verknüpfung zum entsprechenden DALI-Kommunikationsbaustein:



6.1.4.2.1.4.1.1.2 FB_BA_BrtnsPrcDALI



Dieser Baustein dient dem Auslesen eines kombinierten DALI-Helligkeits- und Präsenzmelders.

Hinweise zur Funktion des verwendeten DALI-Gerätes entnehmen Sie bitte der Gerätedokumentation des Herstellers.

Funktion

Das Gerät wird durch die Instanznummer der Präsenz, *nInstancePresence*, und der Instanznummer der Helligkeit, *nInstanceBrightness* spezifiziert. Hinzu kommt die gerätespezifische Auflösung, *nResBrtns*. Diese drei Kenngrößen sind von Hersteller zu Hersteller verschieden.

Die Helligkeit wird zyklisch (Intervall *nPrdQueryBrtns* [s]) abgefragt. Bei der Präsenz hingegen wird das geräteinterne Ereignissystem (notifications) verwendet, welches selbsttätig ein Telegramm losschickt, sobald Präsenz bzw. Abwesenheit erkannt wird.

Der DALI-Helligkeits- und Präsenzbaustein kann über den Eingang *bInitialize* mit teils parametriebaren (siehe VAR_INPUT CONSTANT PERSISTENT), und teils **fest voreingestellten** Werten initialisiert werden (siehe VAR CONSTANT).

Syntax

```
FUNCTION_BLOCK FB_BA_BrtnsPrcDALI
VAR_INPUT
  bEn                   : BOOL;
  bInitialize           : BOOL;
  nPrdQueryBrtns        : UDINT;
  nAdr                  : BYTE;
  nInstPrc              : BYTE;
  nInstBrtns            : BYTE;
  nResBrtns             : BYTE := 10;
  ipDALICommunication   : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_OUTPUT
  bPrc                  : BOOL;
  nBrtns                : UINT;
  bInitializing         : BOOL;
  bErr                  : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nDeadtime             : BYTE := 2;
  nHold                 : BYTE := 90;
  nReport               : BYTE := 60;
END_VAR
VAR CONSTANT
  nEventFilter          : DWORD := 7;
  eEventPriority        : Tc3_DALI.E_DALIEventPriority := Tc3_DALI.E_DALIEventPriority.Middle;
  eEventScheme          : Tc3_DALI.E_DALIEventScheme := Tc3_DALI.E_DALIEventScheme.DeviceInstance;
END_VAR
```

VAR_INPUT

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteines: ein TRUE-Signal an diesem Eingang aktiviert die Funktion.
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in das DALI-Gerät überträgt, siehe FB_BA_Swi4DALI [▶ 894].
nPrdQueryBrtns	UDINT	Abfrageintervall der Helligkeit in Sekunden. Die Deklaration dieser Variable als Eingang soll eine variable Abfragegeschwindigkeit ermöglichen. So genügt bei einer Konstantlichtregelung beispielsweise zunächst ein langsames Abfrageintervall, bis eine größere auszugleichende Abweichung erkannt wird, die dann mit einem kürzeren Intervall ausgeregelt wird. Eine "0" an diesem Eingang verriegelt die Abfrage.
nAdr	BYTE	DALI-Kurzadresse des Sensors.
nInstPrc	BYTE	Herstellerspezifische Nummer der abzufragenden Präsenz-Instanz.
nInstBrtns	BYTE	Herstellerspezifische Nummer der abzufragenden Helligkeits-Instanz.
nResBrtns	BYTE	Herstellerspezifische Helligkeits-Auflösung.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

VAR_OUTPUT

Name	Typ	Beschreibung
bPrc	BOOL	Präsenzmelde-Ausgang.
nBrtns	BOOL	Gemessene Helligkeit. Die Einheit ist herstellerspezifisch.
bInitializing	BOOL	Der Sensor befindet sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in das DALI-Gerät übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.

VAR_INPUT CONSTANT PERSISTENT

Die folgenden Parameter werden durch die Initialisierungsroutine in das DALI-Vorschaltgerät bzw. die DALI-Geräte übertragen.

Name	Typ	Beschreibung
nDeadtime	BYTE	Dead-Time des Präsenzmelders als Vielfaches von 50ms, voreingestellt auf 2 (=100ms).
nHold	BYTE	Hold-Time des Präsenzmelders als Vielfaches von 10s, voreingestellt auf 90 (=900s).
nReport	BYTE	Report-Time des Präsenzmelders in Sekunden, voreingestellt auf 60s.

VAR CONSTANT

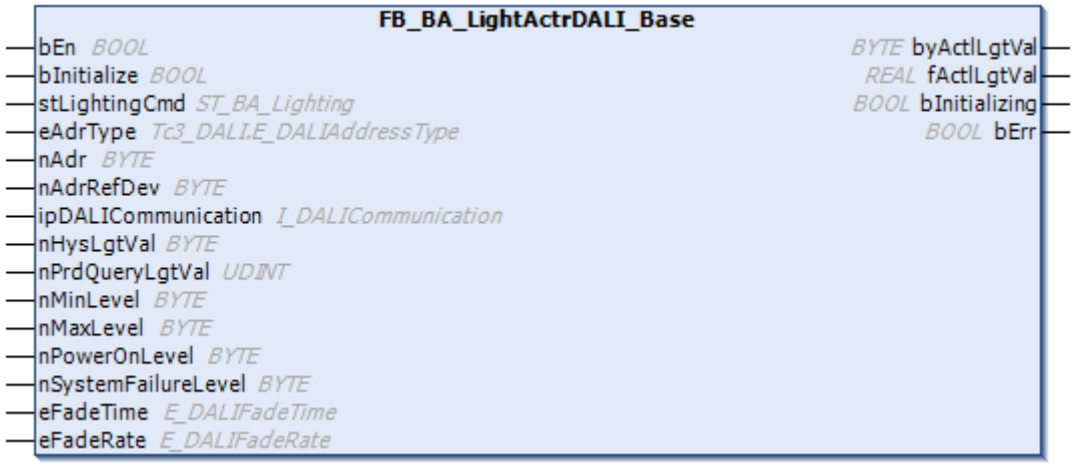
Die folgenden Parameter sind unveränderbar und werden durch die Initialisierungsroutine in das DALI-Gerät übertragen.

Name	Typ	Beschreibung
nEventFilter	DWORD	DALI-Ereignisfilter-Maske für Präsenzmelder. Dieser Wert ist auf 7 eingestellt und fragt damit die Zustände *Occupied*, *Vacant* und *Repeat Event* ab.
eEventPriority	Tc3_DALI.E_DALIEventPriority	DALI-Ereignispriorität, mit der Input Notification Events von der Instanz des DALI-Steuergeräts versendet werden, eingestellt auf *Middle*.
eEventScheme	Tc3_DALI.E_DALIEventScheme	DALI-Ereignisschema. Dieses ist auf *DeviceInstance* eingestellt (Geräte- / Instanz-Adressierung mit Kurzadresse und Instanznummer).

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.1.1.3 FB_BA_LightActrDALI_Base



Dieser Baustein dient zur Ansteuerung eines DALI-Lichtaktors, einer DALI-Gruppe oder von DALI-Geräten mit Sammelbefehl (broadcast), wobei nur der Lichthelligkeitswert verändert wird.

Funktion

Im Baustein werden 2 Routinen durchlaufen:

- Das Setzen des Lichtwertes, wenn dieser vom aktuellen Wert abweicht. Beaufschlagt ist dieses mit einem Hysteresebereich *nHysLgtVal*, siehe unten.
- Das Auslesen des Lichtwertes. Dieses geschieht zyklisch im Intervall *nPrdQueryLgtVal* in Sekunden.

Als "aktueller Wert" wird zunächst immer der Wert angenommen, welcher fehlerfrei gesendet wurde. Die zyklische Abfrage der Lesefunktion wird den Wert korrigieren, sollte ein Telegramm dennoch "verloren" gegangen oder das Vorschaltgerät in irgendeiner Form anderweitig angesteuert worden sein.

Diese Vorgehensweise stellt sicher, dass zum einen der DALI-Bus nicht zu sehr belastet wird, auf der anderen Seite jedoch der tatsächliche Lichtwert immer wieder einmal gelesen wird.

Die Ansteuerung des Bausteins erfolgt über Werte aus dem Lichttelegramm stLightingCmd [▶ 277]. Dabei werden die Steuerwerte des Lichts auf die Grenzen limitiert, welche zuvor durch eine Initialisierungsroutine mit anderen Konfigurationsparametern in das Vorschaltgerät, bzw. die Geräte übertragen wurden, siehe VAR_INPUT CONSTANT PERSISTENT. Die Initialisierung erfolgt über den Eingang *bInitialize*. Hier werden auch **fest eingestellte** Werte übertragen, siehe VAR CONSTANT.

Das Setzen des Lichtwertes erfolgt intern ausschließlich über den DALI-Befehl FB_DALI102DirectArcPowerControl. Damit ist die FadeTime (Stufenzeit) die Zeit, mit der jeweils auf den neuen Lichtwert gedimmt wird.

Voreingestellt ist diese Dimmzeit auf 0s, um Dimmrampen so allein in der SPS realisieren zu können.

Ist eine Gruppen- oder Sammelansteuerung gewählt (broadcast), so richten sich die Steuer- und Initialisierungsbefehle an alle Geräte. Lesebefehle hingegen werden nur an das Referenzgerät gesendet.

Syntax

```
FUNCTION_BLOCK FB_BA_LightActrDALI_Base
VAR_INPUT
  bEn                           : BOOL;
  bInitialize                   : BOOL;
  stLightingCmd                 : ST_BA_Lighting;
  eAdrType                      : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                          : BYTE;
  nAdrRefDev                    : BYTE;
  ipDALICommunication           : I_DALICommunication;
END_VAR
VAR_OUTPUT
  byActlLgtVal                  : BYTE;
  fActlLgtVal                   : REAL;
  bInitializing                 : BOOL;
  bErr                          : BOOL;
```

```
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nHysLgtVal                    : BYTE;
  nPrdQueryLgtVal               : UDINT;

  nMinLevel                     : BYTE := 126;
  nMaxLevel                     : BYTE := 254;
  nPowerOnLevel                 : BYTE := 254;
  nSystemFailureLevel           : BYTE := 254;
  eFadeTime                     : Tc3_DALI.E_DALIFadeTime := Tc3_DALI.E_DALIFadeTime.Disabled;
  eFadeRate                     : Tc3_DALI.E_DALIFadeRate := Tc3_DALI.E_DALIFadeRate.N358StepsPerSe
c;
END_VAR
VAR CONSTANT
  eExtendedFadeTimeBase         : Tc3_DALI.E_DALIExtendedFadeTimeBase := Tc3_DALI.E_DALIExtendedFad
eTimeBase.Base01;
  eExtendedFadeTimeMultiplier   : Tc3_DALI.E_DALIExtendedFadeTimeMultiplier := Tc3_DALI.E_DALIExten
dedFadeTimeMultiplier.Disabled;
END_VAR
```

VAR_INPUT

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteines: ein TRUE-Signal an diesem Eingang aktiviert die Funktion.
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in das DALI-Gerät überträgt, siehe FB_BA_Swi4DALI [▶ 894].
stLightingCmd	ST_BA_Lighting [▶ 277]	Lichtsteuertelegramm
eAdrType	Tc3_DALI.E_DALIAddressType	Definiert, ob der Eingang *nAdr* eine Kurzadresse (0…63) oder eine Gruppenadresse (0…31) enthält. Der Eingang *nAdr* hat keine Bedeutung, wenn ein Sammelruf (Broadcast) geschickt wird
nAdr	BYTE	DALI-Kurzadresse des Sensors.
nAdrRefDev	BYTE	DALI-Adresse eines Referenzgerätes, sollte Gruppen- oder Broadcast-Steuerung gewählt sein. Dieses Referenzgerät repräsentiert dann alle anderen angesteuerten Geräte.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

VAR_OUTPUT

Name	Typ	Beschreibung
byActlLgtVal	BYTE	Aktueller Lichtwert im DALI-Format (0…254).
fActlLgtVal	REAL	Aktueller Lichtwert in Prozent.
bInitializing	BOOL	Der Sensor befindet sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in das DALI-Gerät übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.

VAR_INPUT CONSTANT PERSISTENT

Name	Typ	Beschreibung
nHysLgtVal	BYTE	Hysteresebereich um den Lichtvorgabewert. Ein erneuter Setzbefehl wird dann gestartet, wenn der Sollwert *nHysLgtVal*/2 über oder unter dem Istwert liegt.
nPrdQueryLgtVal	UDINT	Abfrageintervall des Lichtwertes [s].

Die folgenden Parameter werden durch die Initialisierungsroutine in das DALI-Vorschaltgerät bzw. die DALI-Geräte übertragen.

Name	Typ	Beschreibung
nMinLevel	BYTE	Minimaler Lichtstellwert im DALI-Ausgabeformat (0…254). Voreingestellt auf 126.
nMaxLevel	BYTE	Maximaler Lichtstellwert im DALI-Ausgabeformat (0…254). Voreingestellt auf 254.
nPowerOnLevel	BYTE	Lichtstellwert beim Anlegen der Versorgungsspannung an das Vorschaltgerät. Voreingestellt auf 254.
nSystemFailureLevel	BYTE	Lichtstellwert bei einem Fehler am DALI-Bus. Voreingestellt auf 254.
eFadeTime	Tc3_DALI.E_DALIFadeTime	Die Fade Time (Stufenzeit) legt die Zeit des aktuellen Ausgangswerts zum geforderten Wert fest. Die FadeTime ist auf "disabled" voreingestellt, was zusammen mit der deaktivierten "Extended FadeTime" (VAR CONSTANT) einer Stufenzeit von 0s entspricht.
eFadeRate	Tc3_DALI.E_DALIFadeRate	Die Fade Rate (Stufengeschwindigkeit) legt die Veränderungsgeschwindigkeit (in Stufen pro Sekunde) für die Änderung des Ausgangswerts fest. Für das Verhalten dieses Bausteines ist sie nicht ausschlaggebend. Sie ist auf 358 Stufen/s voreingestellt.

VAR CONSTANT

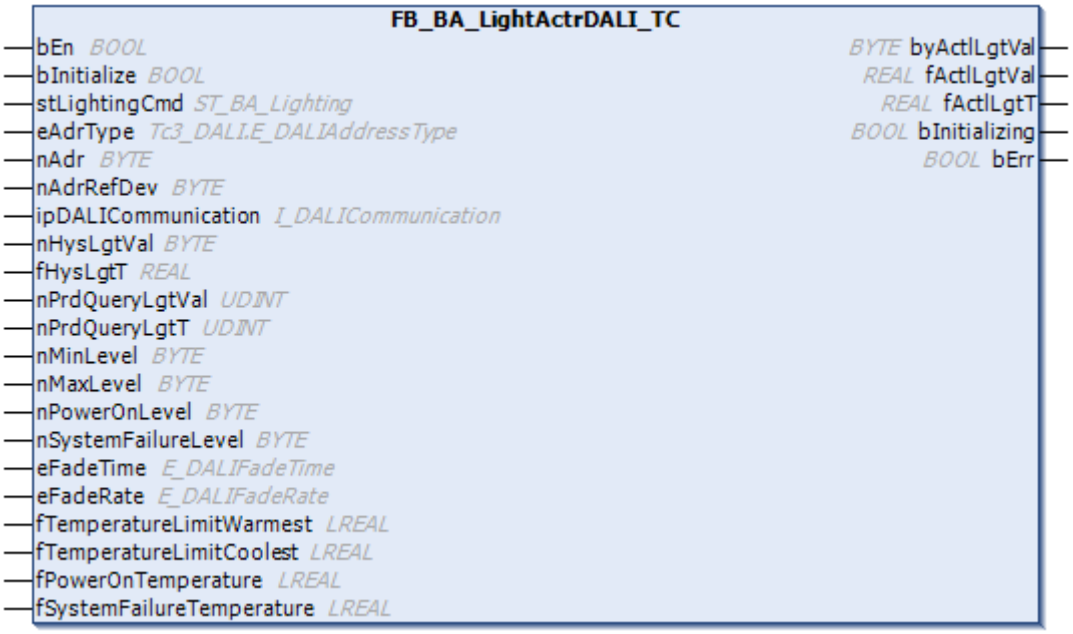
Die folgenden Parameter sind unveränderbar und werden durch die Initialisierungsroutine in das DALI-Gerät übertragen.

Name	Typ	Beschreibung
eExtendedFadeTimeBase	Tc3_DALI.E_DALIExtendedFadeTimeBase	Die Extended Fade Time ist für sehr lange Dimmrampen bis 16 min gedacht. Sie ist für diesen Anwendungsfall deaktiviert. *eExtendedFadeTimeBase* bezeichnet die Zeitbasis und ist auf den kleinsten Wert "Base01" voreingestellt.
eExtendedFadeTimeMultiplier	Tc3_DALI.E_DALIExtendedFadeTimeMultiplier	*eExtendedFadeTimeMultiplier* ist der Multiplikator zur Zeitbasis und ist auf den Wert "Disabled" eingestellt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.1.1.4 FB_BA_LightActrDALI_TC



Dieser Baustein dient zur Ansteuerung eines DALI-Lichtaktors, einer DALI-Gruppe oder von DALI-Geräten mit Sammelbefehl (broadcast), wobei Farbtemperatursteuerung (Temperature-Control) unterstützt wird.

Funktion

Im Baustein werden 4 Routinen durchlaufen:

- Das Setzen des Lichtwertes, wenn dieser vom aktuellen Wert abweicht. Beaufschlagt ist dieses mit einem Hysteresebereich *nHysLgtVal*, siehe unten.
- Das Setzen der Farbtemperatur, wenn diese vom aktuellen Wert abweicht. Beaufschlagt ist dieses mit einem Hysteresebereich *fHysLgtT*, siehe unten.
- Das Auslesen des Lichtwertes. Dieses geschieht zyklisch im Intervall *nPrdQueryLgtVal* in Sekunden.
- Das Auslesen der Farbtemperatur. Dieses geschieht zyklisch im Intervall *nPrdQueryLgtT* in Sekunden.

Als "aktueller Wert" wird zunächst immer der Wert angenommen, welcher fehlerfrei gesendet wurde. Die zyklische Abfrage der jeweiligen Lesefunktion wird den Wert korrigieren, sollte ein Telegramm dennoch "verloren" gegangen oder das Vorschaltgerät in irgendeiner Form anderweitig angesteuert worden sein.

Diese Vorgehensweise stellt sicher, dass zum einen der DALI-Bus nicht zu sehr belastet wird, auf der anderen Seite jedoch der tatsächliche Lichtwert immer wieder gelesen wird.

Die Ansteuerung des Bausteins erfolgt über Werte aus dem Lichttelegramm stLightingCmd [▶ 277]. Dabei werden die Steuerwerte des Lichts und der Farbtemperatur auf die Grenzen limitiert, welche zuvor durch eine Initialisierungsroutine mit anderen Konfigurationsparametern in das Vorschaltgerät, bzw. die Geräte übertragen wurden, siehe VAR_INPUT CONSTANT PERSISTENT. Die Initialisierung erfolgt über den Eingang *bInitialize*. Hier werden auch fest eingestellte Werte übertragen, siehe VAR CONSTANT.

Ein Farbtemperatur-Sollwert von "0" wird explizit ignoriert. Dadurch lassen sich Telegramme erzeugen, welche nur die Helligkeit, nicht aber die zuvor anderweitig eingestellte Farbtemperatur beeinflussen.

Das Setzen des Lichtwertes erfolgt intern ausschließlich über den DALI-Befehl FB_DALI102DirectArcPowerControl. Damit ist die FadeTime (Stufenzeit) die Zeit, mit der jeweils auf den neuen Lichtwert gedimmt wird.

Voreingestellt ist diese Dimmzeit auf 0 s, um Dimmrampen so ausschließlich in der SPS realisieren zu können.

Ist eine Gruppen- oder Sammelansteuerung gewählt (broadcast), so richten sich die Steuer- und Initialisierungsbefehle an alle Geräte. Lesebefehle hingegen werden nur an das Referenzgerät gesendet.

Syntax

```
FUNCTION_BLOCK FB_BA_LightDALI_TC
VAR_INPUT
  bEn                          : BOOL;
  bInitialize                  : BOOL;
  stLightingCmd                : ST_BA_Lighting;
  eAdrType                     : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                         : BYTE;
  nAdrRefDev                   : BYTE;
  ipDALICommunication          : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_OUTPUT
  byActlLgtVal                 : BYTE;
  fActlLgtVal                  : REAL;
  fActlLgtT                    : REAL;
  bInitializing                : BOOL;
  bErr                         : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nHysLgtVal                   : BYTE;
  fHysLgtT                     : REAL;
  nPrdQueryLgtVal              : UDINT;
  nPrdQueryLgtT                : UDINT;

  nMinLevel                    : BYTE := 126;
  nMaxLevel                    : BYTE := 254;
  nPowerOnLevel                : BYTE := 254;
  nSystemFailureLevel          : BYTE := 254;
  eFadeTime                    : Tc3_DALI.E_DALIFadeTime := Tc3_DALI.E_DALIFadeTime.Disabled;
  eFadeRate                    : Tc3_DALI.E_DALIFadeRate := Tc3_DALI.E_DALIFadeRate.N358StepsPerSec;
  fTemperatureLimitWarmest     : LREAL := 2700;
  fTemperatureLimitCoolest     : LREAL := 6500;
  fPowerOnTemperature          : LREAL := 3000;
  fSystemFailureTemperature    : LREAL := 3000;
END_VAR
VAR CONSTANT
  eExtendedFadeTimeBase        : Tc3_DALI.E_DALIExtendedFadeTimeBase := Tc3_DALI.E_DALIExtendedFadeT
imeBase.Base01;
  eExtendedFadeTimeMultiplier  : Tc3_DALI.E_DALIExtendedFadeTimeMultiplier := Tc3_DALI.E_DALIExtende
dFadeTimeMultiplier.Disabled;
END_VAR
```

VAR_INPUT

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteines: ein TRUE-Signal an diesem Eingang aktiviert die Funktion.
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in das DALI-Gerät überträgt, siehe FB_BA_Swi4DALI [▶ 894].
stLightingCmd	ST_BA_Lighting [▶ 277]	Lichtsteuertelegramm
eAdrType	Tc3_DALI.E_DALIAddressType	Definiert, ob der Eingang *nAdr* eine Kurzadresse (0…63) oder eine Gruppenadresse (0…31) enthält. Der Eingang *nAdr* hat keine Bedeutung, wenn ein Sammelruf (Broadcast) geschickt wird
nAdr	BYTE	DALI-Kurzadresse des Sensors.
nAdrRefDev	BYTE	DALI-Adresse eines Referenzgerätes, sollte Gruppen- oder Broadcast-Steuerung gewählt sein. Dieses Referenzgerät repräsentiert dann alle anderen angesteuerten Geräte.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

VAR_OUTPUT

Name	Typ	Beschreibung
byActlLgtVal	BYTE	Aktueller Lichtwert im DALI-Format (0…254).
fActlLgtVal	REAL	Aktueller Lichtwert in Prozent.
fActlLgtT	REAL	Aktuelle Farbtemperatur in Kelvin.
bInitializing	BOOL	Der Sensor befindet sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in das DALI-Gerät übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.

VAR_INPUT CONSTANT PERSISTENT

Name	Typ	Beschreibung
nHysLgtVal	BYTE	Hysteresebereich um den Lichtvorgabewert. Ein erneuter Setzbefehl wird dann gestartet, wenn der Sollwert *nHysLgtVal*/2 über oder unter dem Istwert liegt.
fHysLgtT	REAL	Hysteresebereich um den Farbtemperaturvorgabewert. Ein erneuter Setzbefehl wird dann gestartet, wenn der Sollwert *nHysLgtVal*/2 über oder unter dem Istwert liegt.
nPrdQueryLgtVal	UDINT	Abfrageintervall des Lichtwertes [s].
nPrdQueryLgtT	UDINT	Abfrageintervall der Farbtemperatur [s].



Die folgenden Parameter werden durch die Initialisierungsroutine in das DALI-Vorschaltgerät bzw. die DALI-Geräte übertragen.

Name	Typ	Beschreibung
nMinLevel	BYTE	Minimaler Lichtstellwert im DALI-Ausgabeformat (0…254). Voreingestellt auf 126.
nMaxLevel	BYTE	Maximaler Lichtstellwert im DALI-Ausgabeformat (0…254). Voreingestellt auf 254.
nPowerOnLevel	BYTE	Lichtstellwert beim Anlegen der Versorgungsspannung an das Vorschaltgerät. Voreingestellt auf 254.
nSystemFailureLevel	BYTE	Lichtstellwert bei einem Fehler am DALI-Bus. Voreingestellt auf 254.
eFadeTime	Tc3_DALI.E_DALIFadeTime	Die Fade Time (Stufenzeit) legt die Zeit des aktuellen Ausgangswerts zum geforderten Wert fest. Die FadeTime ist auf "disabled" voreingestellt, was zusammen mit der deaktivierten "Extended FadeTime" (VAR CONSTANT) einer Stufenzeit von 0s entspricht.
eFadeRate	Tc3_DALI.E_DALIFadeRate	Die Fade Rate (Stufengeschwindigkeit) legt die Veränderungsgeschwindigkeit (in Stufen pro Sekunde) für die Änderung des Ausgangswerts fest. Für das Verhalten dieses Bausteines ist sie nicht ausschlaggebend. Sie ist auf 358 Stufen/s voreingestellt.
fTemperatureLimitWarmest	LREAL	Untere Grenze des Farbtemperaturbereiches in Kelvin. Voreingestellt auf 2700K.
fTemperatureLimitCoolest	LREAL	Obere Grenze des Farbtemperaturbereiches in Kelvin. Voreingestellt auf 6500K.
fPowerOnTemperature	LREAL	Farbtemperaturwert beim Anlegen der Versorgungsspannung an das Vorschaltgerät. Voreingestellt auf 3000K.
fSystemFailureTemperature	LREAL	Farbtemperaturwert bei einem Fehler am DALI-Bus. Voreingestellt auf 3000K.

VAR CONSTANT

Die folgenden Parameter sind unveränderbar und werden durch die Initialisierungsroutine in das DALI-Gerät übertragen.

Name	Typ	Beschreibung
eExtendedFadeTimeBase	Tc3_DALI.E_DALIExtendedFadeTimeBase	Die Extended Fade Time ist für sehr lange Dimmrampen bis 16 min gedacht. Sie ist für diesen Anwendungsfall deaktiviert. *eExtendedFadeTimeBase* bezeichnet die Zeitbasis und ist auf den kleinsten Wert "Base01" voreingestellt.
eExtendedFadeTimeMultiplier	Tc3_DALI.E_DALIExtendedFadeTimeMultiplier	*eExtendedFadeTimeMultiplier* ist der Multiplikator zur Zeitbasis und ist auf den Wert "Disabled" eingestellt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.1.1.5 FB_BA_Swi4DALI



Dieser Baustein dient zum Auslesen eines DALI-4-fach-Tasters.

Hinweise zur Funktion des verwendeten DALI-Gerätes entnehmen Sie bitte der Gerätedokumentation des Herstellers.

Funktion

Das Gerät selbst ist spezifiziert durch die Instanznummern der Taster, *nInstButton1…nInstButton4*.

Es wird grundsätzlich das geräteinterne Ereignissystem (notifications) verwendet, welches selbsttätig ein Telegramm losschickt, sobald ein Tastendruck oder ein Loslassen einer Taste erkannt wird.

Die Zustände der Tasten werden zusätzlich zyklisch abgefragt. So wird sichergestellt, dass der Zustand der Tasten immer wieder korrigiert wird, sollte einmal ein Telegramm "verloren gehen".

Der DALI-Tasterbaustein kann über den Eingang *bInitialize* mit **fest eingestellten** Parametern initialisiert werden, siehe VAR CONSTANT.

Syntax

```
FUNCTION_BLOCK FB_BA_Swi4DALI
VAR_INPUT
  bEn                   : BOOL;
  bInitialize           : BOOL;
  nAdr                  : BYTE;
  nInstButton1          : BYTE;
  nInstButton2          : BYTE;
  nInstButton3          : BYTE;
  nInstButton4          : BYTE;
  ipDALICommunication   : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_OUTPUT
  bButton1              : BOOL;
  bButton2              : BOOL;
  bButton3              : BOOL;
  bButton4              : BOOL;
  bInitializing         : BOOL;
  bErr                  : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nPrdQuery             : UDINT := 60;
END_VAR
VAR CONSTANT
  nEventFilter          : DWORD := 131;
  eEventPriority        : Tc3_DALI.E_DALIEventPriority := Tc3_DALI.E_DALIEventPriority.Middle;
  eEventScheme          : Tc3_DALI.E_DALIEventScheme := Tc3_DALI.E_DALIEventScheme.DeviceInstance;
  nRepeatTimer          : UINT := 160;
  nShortTimer           : UINT := 500;
  nDoubleTimer          : UINT := 0;
  nStuckTimer           : BYTE := 20;
END_VAR
```

VAR_INPUT

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteines: ein TRUE-Signal an diesem Eingang aktiviert die Funktion.
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in das DALI-Gerät überträgt, siehe FB_BA_Swi4DALI [▶ 894].
nAdr	BYTE	DALI-Kurzadresse des Tastsensors.
nInstButton1… nInstButton4	BYTE	Nummern der abzufragenden Taster-Instanzen.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

VAR_OUTPUT

Name	Typ	Beschreibung
bButton1…bButton4	BOOL	Zustand der einzelnen Taster (TRUE = gedrückt).
bInitializing	BOOL	Der Sensor befindet sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in das DALI-Gerät übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.

VAR_INPUT CONSTANT PERSISTENT

Name	Typ	Beschreibung
nPrdQuery	UDINT	Abfrageintervall der Schalterzustände. Diese Abfrage dient lediglich der Korrektur, falls durch einen Telegrammfehler ein falscher Status gemerkt wird.

VAR CONSTANT

Die folgenden Parameter sind unveränderbar und werden durch die Initialisierungsroutine in das DALI-Gerät übertragen.

Name	Typ	Beschreibung
nEventFilter	DWORD	DALI-Ereignisfilter-Maske für Präsenzmelder. Dieser Wert ist auf 7 eingestellt und fragt damit die Zustände *Occupied*, *Vacant* und *Repeat Event* ab.
eEventPriority	Tc3_DALI.E_DALIEventPriority	DALI-Ereignispriorität, mit der Input Notification Events von der Instanz des DALI-Steuergeräts versendet werden, eingestellt auf *Middle*.
eEventScheme	Tc3_DALI.E_DALIEventScheme	DALI-Ereignisschema. Dieses ist auf *DeviceInstance* eingestellt (Geräte- / Instanz-Adressierung mit Kurzadresse und Instanznummer).
nRepeatTimer	UINT	Repeat-Timer als Vielfaches von 20ms. Eingestellt auf 8 (=160ms).
nShortTimer	UINT	Short-Timer als Vielfaches von 20ms. Eingestellt auf 25 (=500ms).
nDoubleTimer	UINT	Double-Timer als Vielfaches von 20ms. Eingestellt auf 0 (=0ms). Damit ist dieser Wert auf den herstellerspezifischen Minimalwert gesetzt.
nStuckTimer	BYTE	Stuck-Timer in Sekunden. Eingestellt auf 20s.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.1.2 Sensors

6.1.4.2.1.4.1.2.1 FB_BA_Switch4DALI



Das Template dient dem Auslesen eines 4-fach-Schalters.

Datenaustausch HMI

Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_BaseSwitch* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variable:

- **bInitializeRm:** Start DALI-Initialisierungsroutine aus der HMI.

Funktion

Das Gerät selbst wird spezifiziert durch die Instanznummern der Taster, *nInstButton1* … *nInstButton4*.

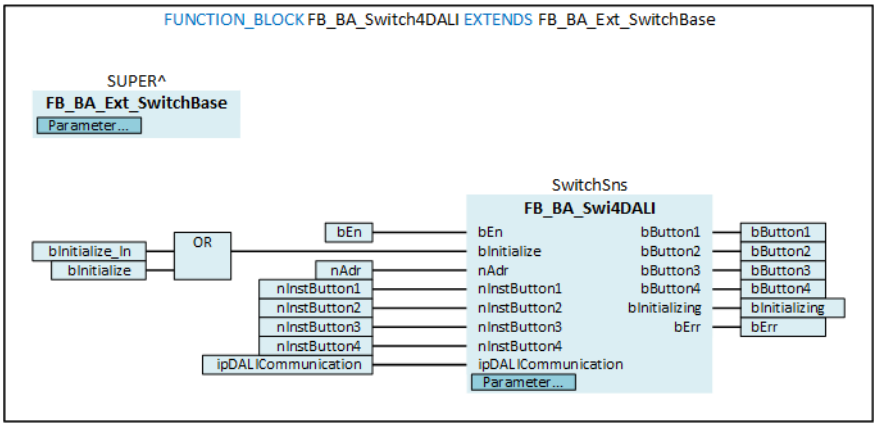
Es wird grundsätzlich das geräteinterne Ereignissystem (notifications) verwendet, welches selbsttätig ein Telegramm losschickt, sobald ein Tastendruck bzw. ein Loslassen einer Taste erkannt wird.

Im internen Baustein FB_BA_Swi4DALI [▶ 894] werden die Zustände der Tasten zusätzlich zyklisch abgefragt. So wird sichergestellt, dass der Zustand der Tasten immer wieder korrigiert wird, sollte einmal ein Telegramm "verloren" gehen.

Der DALI-Tasterbaustein kann über den Eingang *bInitialize* oder über die HMI (*bInitializeRm*) mit seinen eingestellten Parametern initialisiert werden, siehe FB_BA_Swi4DALI [▶ 894].

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Switch4DALI EXTENDS FB_BA_Ext_SwitchBase
VAR_INPUT
  bEn                   : BOOL;
  bInitialize           : BOOL;
END_VAR
VAR_OUTPUT
  bButton1              : BOOL;
  bButton2              : BOOL;
  bButton3              : BOOL;
  bButton4              : BOOL;
  bInitializing         : BOOL;
  bErr                  : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nAdr                  : BYTE;
  nInstButton1          : BYTE := 0;
  nInstButton2          : BYTE := 1;
  nInstButton3          : BYTE := 2;
  nInstButton4          : BYTE := 3;
  ipDALICommunication   : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_INPUT CONSTANT
  SwitchSns             : FB_BA_Swi4DALI;
END_VAR
```

VAR_INPUT

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteines: ein TRUE-Signal an diesem Eingang aktiviert die Funktion.
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in das DALI-Gerät überträgt, siehe FB_BA_Swi4DALI [▶ 894].

VAR_OUTPUT

Name	Typ	Beschreibung
bButton1…bButton4	BOOL	Zustand der einzelnen Taster (TRUE = gedrückt).
bInitializing	BOOL	Der Tastsensor befindet sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in das DALI-Gerät übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.

VAR_INPUT CONSTANT PERSISTENT

Name	Typ	Beschreibung
nAdr	BYTE	DALI-Einzeladresse des Tastsensors.
nInstButton1… nInstButton4	BYTE	Nummern der abzufragenden Taster-Instanzen.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

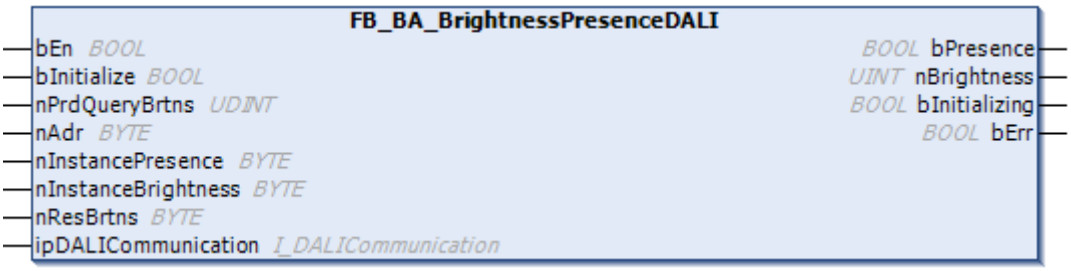
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
SwitchSns	FB_BA_Swi4DALI [▶ 894]	Baustein zum Auslesen des Tastsensors.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.1.2.2 FB_BA_BrightnessPresenceDALI



Dieses Template dient zum Auslesen eines kombinierten DALI-Helligkeits- und Präsenzmelders.

Datenaustausch HMI

Der Datenaustausch mit der HMI wird in der Basisklasse *FB_BA_BaseBrightnessPresence* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variable:

- **bInitializeRm:** Start DALI-Initialisierungsroutine aus der HMI.

Funktion

Das Gerät selbst wird spezifiziert durch die Instanznummer der Präsenz, *nInstancePresence*, und der Instanznummer der Helligkeit, *nInstanceBrightness*. Hinzu kommt die gerätespezifische Auflösung, *nResBrtns*. Diese drei Kenngrössen sind von Hersteller zu Hersteller verschieden.

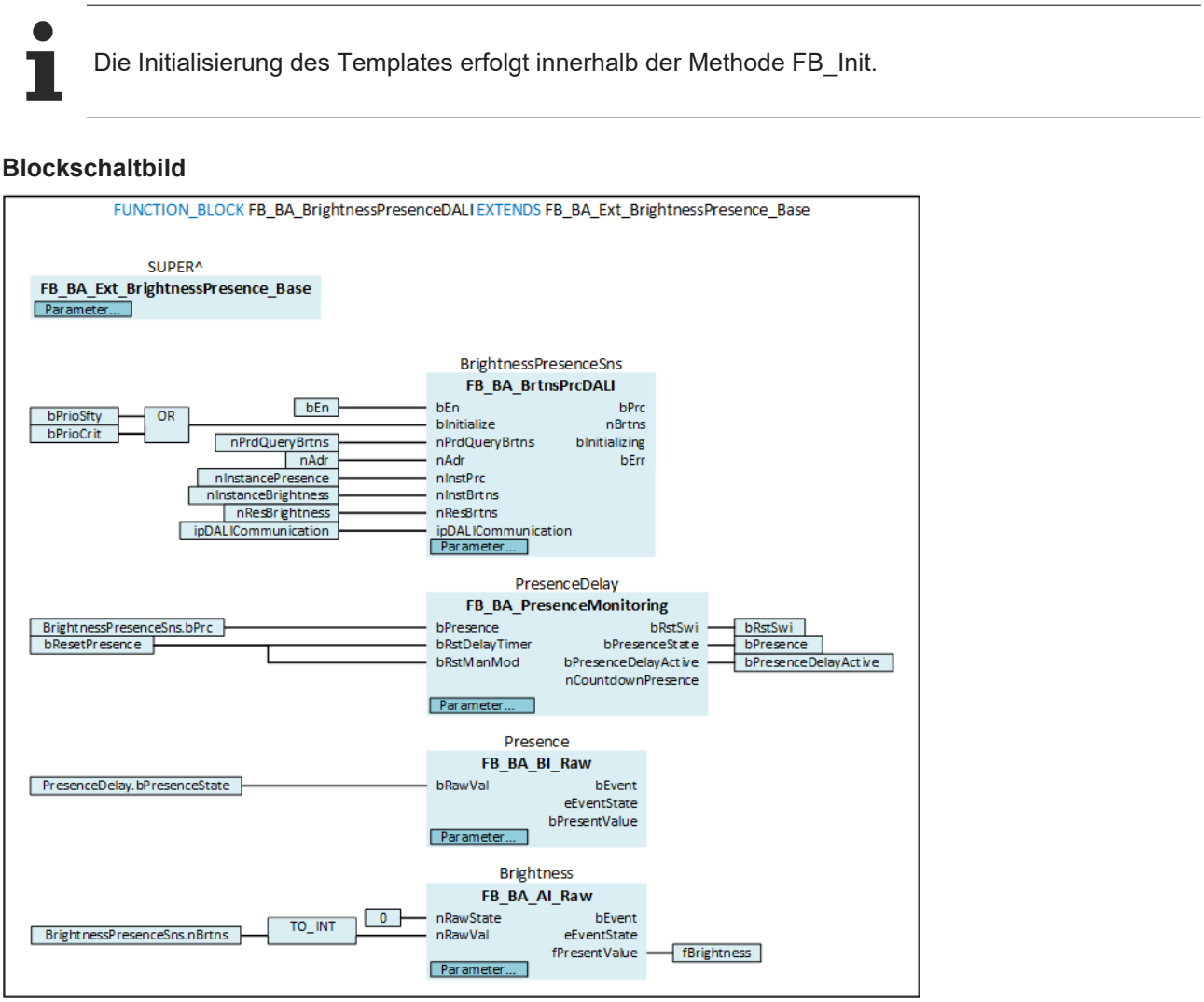
Im internen Baustein BrightnessPresenceSns [▶ 884] wird die Helligkeit zyklisch (Intervall *nPrdQueryBrtns* [s]) abgefragt. Bei der Präsenz hingegen wird das geräteinterne Ereignissystem (notifications) verwendet, welches selbsttätig ein Telegramm losschickt, sobald Präsenz bzw. Abwesenheit erkannt wird.

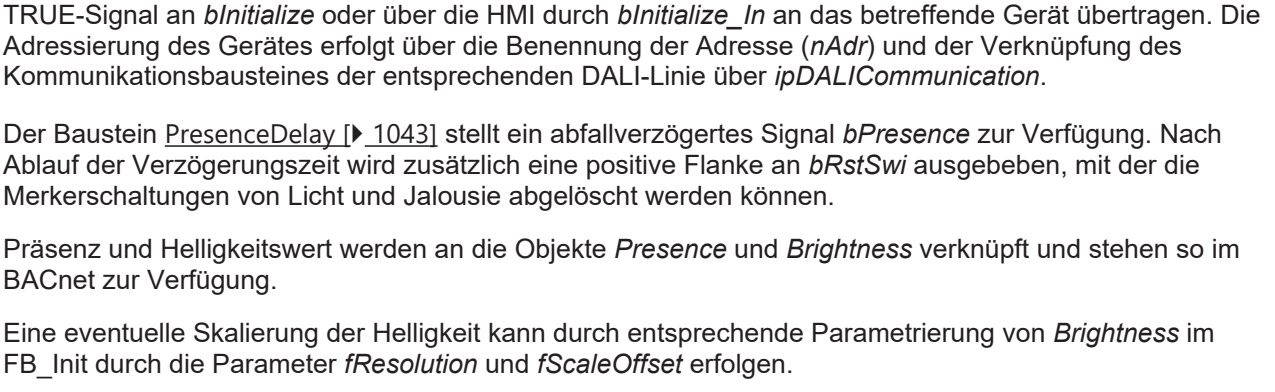
Die DALI-spezifischen Parameter dieses Bausteines (*nInstPrc*, *nInstBrtns* und *nResBrtns*) werden durch Ein TRUE-Signal an *bInitialize* oder über die HMI durch *bInitialize_In* an das betreffende Gerät übertragen. Die Adressierung des Gerätes erfolgt über die Benennung der Adresse (*nAdr*) und der Verknüpfung des Kommunikationsbausteines der entsprechenden DALI-Linie über *ipDALICommunication*.

Der Baustein PresenceDelay [▶ 1043] stellt ein abfallverzögertes Signal *bPresence* zur Verfügung. Nach Ablauf der Verzögerungszeit wird zusätzlich eine positive Flanke an *bRstSwi* ausgebeben, mit der die Merkerschaltungen von Licht und Jalousie abgelöscht werden können.

Präsenz und Helligkeitswert werden an die Objekte *Presence* und *Brightness* verknüpft und stehen so im BACnet zur Verfügung.

Eine eventuelle Skalierung der Helligkeit kann durch entsprechende Parametrierung von *Brightness* im FB_Init durch die Parameter *fResolution* und *fScaleOffset* erfolgen.





Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild

Syntax

```
FUNCTION_BLOCK FB_BA_BrightnessPresenceDALI EXTENDS FB_BA_Ext_BrightnessPresence_Base
VAR_INPUT
  bEn                       : BOOL;
  bInitialize               : BOOL;
  bResetPresence            : BOOL;
  nPrdQueryBrtns            : UDINT  := 1;
END_VAR
VAR_OUTPUT
  bPresence                 : BOOL;
  bPresenceDelayActive      : BOOL;
```

```
  bRstSwi                 : BOOL;
  fBrightness             : REAL;
  bInitializing           : BOOL;
  bErr                    : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nAdr                    : BYTE;
  nInstancePresence       : BYTE;
  nInstanceBrightness     : BYTE;
  nResBrtns               : BYTE;
  ipDALICommunication     : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_INPUT CONSTANT
  BrightnessPresenceSns   : FB_BA_BrtnsPrcDALI;
  PresenceDelay           : FB_BA_PresenceMonitoring;
  Brightness              : FB_BA_AI_RAW;
  Presence                : FB_BA_BI_RAW;
END_VAR
```

VAR_INPUT

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteines: ein TRUE-Signal an diesem Eingang aktiviert die Funktion.
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in das DALI-Gerät überträgt, siehe FB_BA_Swi4DALI [▶ 894].
bResetPresence	BOOL	Setzt die interne Abfallverzögerung der Präsenz zurück und triggert dadurch einmal den Ausgang *bRstSwi*.
nPrdQueryBrtns	UDINT	Abfrageintervall der Helligkeit in Sekunden. Die Deklaration dieser Variable als Eingang soll eine variable Abfragegeschwindigkeit ermöglichen. So genügt bei einer Konstantlichtregelung beispielsweise zunächst ein langsames Abfrageintervall bis eine größere auszugleichende Abweichung erkannt wird, die dann mit einem kürzeren Abfrageintervall ausgeregelt wird.

VAR_OUTPUT

Name	Typ	Beschreibung
bPresence	BOOL	Präsenzmelde-Ausgang.
bPresenceDelayActive	BOOL	Dieser Ausgang steht dann auf TRUE, wenn keine Präsenz mehr erkannt wird und der Präsenzmelde-Ausgang *bPresence* nur noch durch die interne Abfallverzögerung gesetzt ist.
bRstSwi	BOOL	Rücksetzausgang für Handübersteuerungen. Wird für einen SPS-Zyklus nach Ablauf der der internen Präsenzverzögerung (siehe PresenceDelay [▶ 1043]) oder bei einem TRUE-Signal am Eingang bResetPresence gesetzt,
fBrightness	REAL	Gemessene Helligkeit. Die Einheit ist herstellerspezifisch. Eine Umrechnung in Lux muss, falls gewünscht, beim Hersteller erfragt werden.
bInitializing	BOOL	Der Sensor befindet sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in das DALI-Gerät übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.

VAR_INPUT CONSTANT PERSISTENT

Name	Typ	Beschreibung
nAdr	BYTE	DALI-Einzeladresse des Sensors.
nInstancePresence	BYTE	Nummer der abzufragenden Präsenz-Instanz.
nInstanceBrightness	BYTE	Nummer der abzufragenden Helligkeits-Instanz.
nResBrtns	BYTE	Herstellerspezifische Helligkeits-Auflösung.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

VAR_INPUT CONSTANT

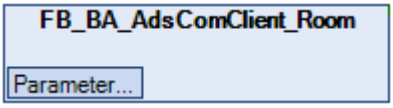
Name	Typ	Beschreibung
BrightnessPresenceSns	FB_BA_BrtnsPrcDALI [▶ 884]	Baustein zum Auslesen des DALI-Helligkeits- und Präsenzsensors.
PresenceDelay	FB_BA_PresenceMonitoring [▶ 1043]	Universelles Präsenzmeldetemplate mit Rücksetzeingängen für Verzögerungstimer und Handfunktion.
Brightness	FB_BA_AI_RAW [▶ 200]	Dieser Baustein dient zum einen der Umrechnung des herstellerspezifischen Rohwertes der Helligkeit auf eine gewünschte Einheit, beispielsweise Lux und zum anderen zur Darstellung des Wertes im BACnet.
Presence	FB_BA_BI_RAW [▶ 211]	Baustein zur Darstellung des Präsenzzustandes im BACnet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.2 Communication

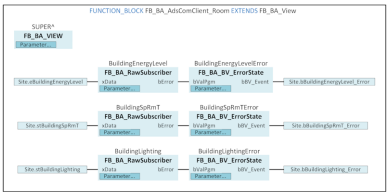
6.1.4.2.1.4.2.1 FB_BA_AdsComClient_Room



Das Template liest globale Daten, welche dem Gewerk "Raumautomation" zuzuordnen sind, von einem anderen Controller, der diese Daten mit dem Gegenbaustein FB_BA_AdsComServer_Room [▶ 902] zur Verfügung stellt. Die gelesenen Informationen werden in die globale Variablenliste Site [▶ 1135] kopiert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  BuildingEnergyLevel         : FB_BA_RawSubscriber;
  BuildingEnergyLevelError    : FB_BA_BV_ErrorState;
  BuildingSpRmT               : FB_BA_RawSubscriber;
  BuildingSpRmTError          : FB_BA_BV_ErrorState;
  BuildingLighting            : FB_BA_RawSubscriber;
  BuildingLightingError       : FB_BA_BV_ErrorState;
END_VAR
```

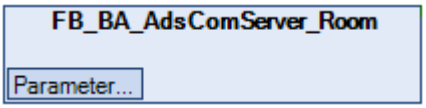
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
BuildingEnergyLevel	FB_BA_RawSubscriber [▶ 162]	Der Subscriber greift auf die TwinCAT Netzwerk-Struktur *eBuildingEnergyLevel* zu und speichert die Daten in die angelegte Enumeration *eBuildingEnergyLevel* der GVL Site [▶ 1135] ab.
BuildingEnergyLevelError	FB_BA_BV_ErrorState [▶ 1048]	Binäres Objekt zur Anzeige der Kommunikationsstörung des Subscribers *BuildingEnergyLevel*.
BuildingSpRmT	FB_BA_RawSubscriber [▶ 162]	Der Subscriber greift auf die TwinCAT Netzwerk-Struktur *BuildingSpRmT* zu und speichert die Daten in die angelegte Struktur *stBuildingSpRmT* der GVL Site [▶ 1135] ab.
BuildingSpRmTError	FB_BA_BV_ErrorState [▶ 1048]	Binäres Objekt zur Anzeige der Kommunikationsstörung des Subscribers *BuildingSpRmT*.
BuildingLighting	FB_BA_RawSubscriber [▶ 162]	Der Subscriber greift auf die TwinCAT Netzwerk-Struktur *BuildingLighting* zu und speichert die Daten in die angelegte Struktur *stBuildingLighting* der GVL Site [▶ 1135] ab.
BuildingLightingError	FB_BA_BV_ErrorState [▶ 1048]	Binäres Objekt zur Anzeige der Kommunikationsstörung des Subscribers *BuildingLighting*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.2.2 FB_BA_AdsComServer_Room

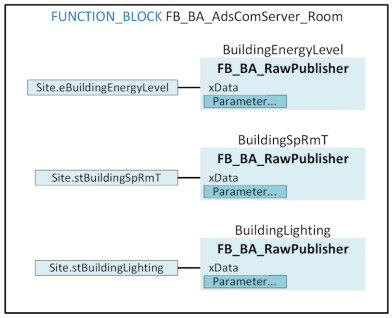


Das Template hat die Aufgabe Daten und Parameter, welche dem Gewerk "Raumautomation" zuzuordnen sind, aus der globalen Variablenliste Site zu lesen und via FB_BA_RawPublisher [▶ 143] innerhalb des GA-Netzwerkes zur Verfügung zu stellen.

Der Gegenbaustein, der diese Daten als Client empfängt, ist FB_BA_AdsComClient_Room [▶ 901].

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComServer
VAR_INPUT CONSTANT
  BuildingEnergyLevel    : FB_BA_RawPublisher;
  BuildingSpRmT          : FB_BA_RawPublisher;
  BuildingLighting       : FB_BA_RawPublisher;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
BuildingEnergyLevel	FB_BA_RawPublisher [▶ 143]	Der Publisher veröffentlicht das aktuelle Gebäudeenergieniveau Site.eBuildingEnergyLevel [▶ 1135], welches im Template FB_BA_BuildingEnergyLevel [▶ 903] gebildet wird.
BuildingSpRmT	FB_BA_RawPublisher [▶ 143]	Der Publisher veröffentlicht die Struktur der gebäudeweit gültigen Raumtemperaturdaten Site.stBuildingSpRmT [▶ 1135], welche im Template FB_BA_BuildingSpRmT [▶ 873] gebildet wird.
BuildingLighting	FB_BA_RawPublisher [▶ 143]	Der Publisher veröffentlicht die Struktur der gebäudeweit gültigen Beleuchtungsdaten Site.stBuildingLighting [▶ 1135], welche im Template FB_BA_BuildingLighting [▶ 933] gebildet wird.

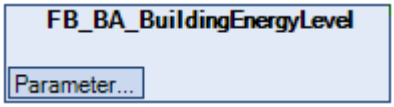
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.3 HeatingCooling

Vorlagen für das Heizen und Kühlen.

6.1.4.2.1.4.3.1 FB_BA_BuildingEnergyLevel



Auswahl des **Gebäudeenergieniveaus** anhand eines Zeitschaltplanes (*Sched*) und eines Handbetriebs (*EnergLvlMan*).

Der Handbetrieb (*EnergLvlMan*) hat folgende Werte:

Wert	Bedeutung
1	Automatikbetrieb
2	manuelle Anwahl Protection
3	manuelle Anwahl Economy
4	manuelle Anwahl Precomfort
5	manuelle Anwahl Comfort

Ist **Automatikbetrieb** gewählt, setzt der Baustein *DeMuxEnergLvl* den Ausgang *bQ01.* Da dieser Ausgang nicht mit dem *PrioSwi* verbunden ist, ist der Handbetrieb am *PrioSwi* deaktiviert und der Zeitschaltplan *Sched* ist aktiv. Dieser kann folgende Werte annehmen:

Wert	Bedeutung
1	Protection
2	Economy
3	Precomfort
4	Comfort

Das aktuell angewählte Energieniveau wird dann über den Baustein *EnergLvlPr* zur Anzeige gebracht und per Publisher zur Verfügung gestellt.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Illustration

```
FUNCTION_BLOCK FB_BA_BuildingEnergyLevel EXTENDS FB_BA_View

VAR_INPUT CONSTANT
  EnergLvlMan      : FB_BA_MV_Op;
  EnergLvlPr       : FB_BA_MV_Op;
  Sched            : FB_BA_SchedM;
END_VAR

VAR
  DeMuxEnergLvl    : FB_BA_DMUX_B08;
  DeMuxSched       : FB_BA_DMUX_B04;
  PrioSwi          : FB_BA_PrioSwi_UDI08;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
EnergLvlMan	FB_BA_MV_Op [▶ 244]	Eingabeobjekt Hand für das Gebäudeenergieniveau „BuildingEnergyLevel“.
EnergLvlPr	FB_BA_MV_Op [▶ 244]	Resultierender Modus.
Sched	FB_BA_SchedM [▶ 230]	Zeitschaltobjekt (Automatik) für das Gebäudeenergieniveau „BuildingEnergyLevel“.

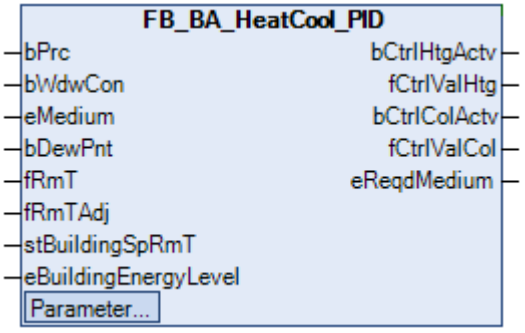
Variablen

Name	Typ	Beschreibung
DeMuxEnergLvl	FB_BA_DMUX_B08 [▶ 436]	Konvertierung des Multistate - Wertes des Handbetriebs zu einem Binärausgang.
DeMuxSched	FB_BA_DMUX_B04 [▶ 436]	Konvertierung des Multistate - Wertes des Zeitschaltplans zu einem Binärausgang.
PrioSwi	FB_BA_PrioSwi_UDI08 [▶ 443]	Priorisierende Rückumwandlung der Status zu einem resultierenden Multistate - bzw. Enumerationswert für das Gebäudeenergieniveau.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.3.2 FB_BA_HeatCool_PID



Dieses Template dient zur Regelung einer Heiz- Kühlzone.

Mit der Prioritätenauswahl *EnergLvlSlcn* wird zunächst das aktuell gültige Energieniveau bestimmt.

Der Fensterkontakt hat dabei immer Vorrang und schaltet bei geöffnetem Fenster auf das Energieniveau „Protection“. Bei Präsenz im Raum wird stets das Niveau „Comfort“ aktiviert.

Der Funktionsbausteins *RmTAdj* dient einer lokalen Verschiebung des Raumtemperatursollwerts für die Energieniveaus „Pre-Comfort“ und „Comfort“.

Die Sollwertverschiebung erfolgt dabei über einen lokalen Eingabewert *fRmTAdj* oder über einen externen Wert *fRmTAdj_In* der über die Basisklasse *FB_BA_Ext_SunblindAngle* (interner Baustein) etwa über eine HMI eingelesen wird. Gültig ist dabei immer der Wert, welcher als letzter geändert wurde, dafür ist der Baustein FB_BA_LastWriterWins_R04 [▶ 457] verantwortlich. Der Raumtemperaturwert ist auf +/- *fRmTAdjLimit* (im FB_init vordefiniert auf 1.0) begrenzt.

Die Instanz *FnctSel* des Funktionsbausteins FB_BA_FnctSel [▶ 306] aktiviert den Heiz- oder Kühlregler der Temperaturzone.

Datenaustausch HMI

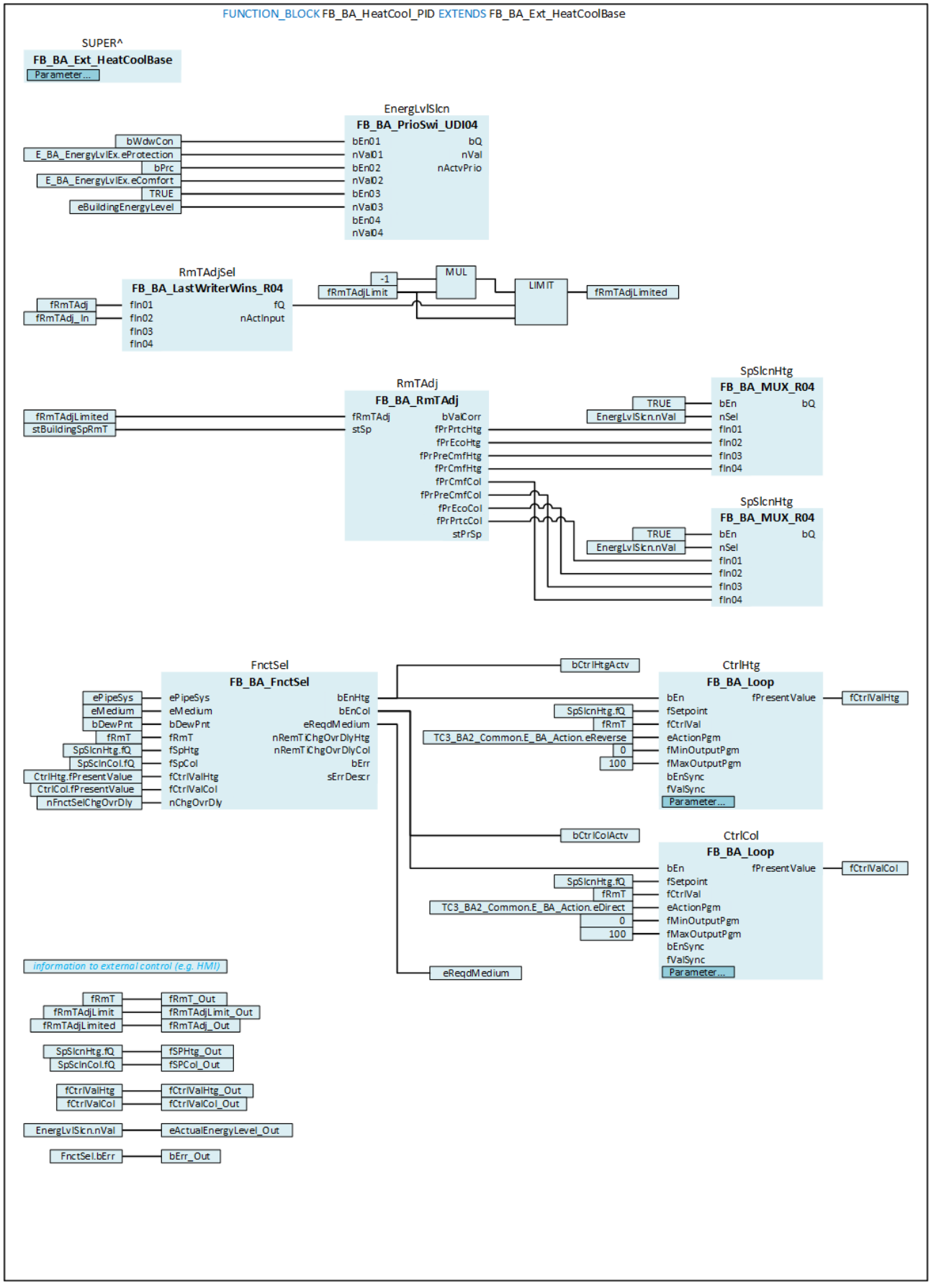
Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_HeatCoolBase* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

- **fRmTAdj_In**: Eingabe Raumsollwertverschiebung von der HMI.
- **fRmTAdj_Out**: Ausgabe aktuelle Raumsollwertverschiebung an die HMI.
- **fRmTAdjLimit_Out**: Ausgabe der +/- Bereichsgrenze der Raumsollwertverschiebung an die HMI.
- **fRmT_Out**: Ausgabe aktuelle Raumtemperatur an die HMI.
- **fSPHtg_Out**: Ausgabe Sollwert Heizen an die HMI.
- **fSPCol_Out**: Ausgabe Sollwert Kühlen an die HMI.
- **fCtrlValHtg_Out**: Ausgabe Ausgang Heizregler an die HMI.
- **fCtrlValCol_Out**: Ausgabe Ausgang Kühlregler an die HMI.
- **eActualEnergyLevel_Out**: Ausgabe aktuelles Energieniveau an die HMI.
- **bErr_Out**: Ausgabe Status "Fehler an *FnctSel*" an die HMI.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_HeatCool_PID EXTENDS FB_BA_Ext_HeatCoolBase
VAR_INPUT
  bPrc                    : BOOL;
  bWdwCon                 : BOOL;
  eMedium                 : E_BA_Medium;
  bDewPnt                 : BOOL;
  fRmT                    : REAL;
  fRmTAdj                 : REAL;
  stBuildingSpRmT         : ST_BA_SpRmT;
  eBuildingEnergyLevel    : E_BA_EnergyLvlEx;

END_VAR
VAR_OUTPUT
  bCtrlHtgActv            : BOOL;
  fCtrlValHtg             : REAL;
  bCtrlColActv            : BOOL;
  fCtrlValCol             : REAL;
  eReqdMedium             : E_BA_Medium;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  ePipeSys                : E_BA_PipeSys;
  nFnctSelChgOvrDly       : UDINT;
  fRmTAdjLimit            : REAL;
END_VAR
VAR_INPUT CONSTANT
  CtrlHtg                 : FB_BA_Loop;
  CtrlCol                 : FB_BA_Loop;
END_VAR
VAR
  RmTAdj                  : FB_BA_RmTAdj;
  EnergLvlSlcn            : FB_BA_PrioSwi_UDI04;
  SpSlcnHtg               : FB_BA_MUX_R04;
  SpSclnCol               : FB_BA_MUX_R04;
  FnctSel                 : FB_BA_FnctSel;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bPrc	BOOL	Präsenzerkennung
bWdwCon	BOOL	Fensterkontakt (offen = TRUE)
eMedium	E_BA_Medium [▶ 272]	Anstehendes Medium (Heiz- oder Kühlmedium, nur wichtig für Zweileiterbetrieb)
bDewPnt	BOOL	Taupunktwächter (Alarm = TRUE)
fRmT	BOOL	Raumtemperatur [°C]
fRmTAdj	REAL	Raumsollwertverschiebung [K]
stBuildingSpRmT	ST_BA_SpRmT [▶ 277]	Struktur der Raumsollwerte (Protection Cooling..Comfort Cooling und Protection Heating .. Comfort Heating)
eBuildingEnergyLevel	E_BA_EnergyLvlEx [▶ 713]	Aktuelles Gebäudeenergieniveau

Ausgänge

Name	Typ	Beschreibung
bCtrlHtgActv	BOOL	Heizregler ist aktiv
fCtrlValHtg	REAL	Stellwert Heizventil
bCtrlColActv	BOOL	Kühlregler ist aktiv
fCtrlValCol	REAL	Stellwert Kühlventil
eReqdMedium	E_BA_Medium [▶ 272]	Gefordertes Medium (Heiz- oder Kühlmedium)

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
ePipeSys	E_BA_PipeSys [▶ 272]	Auswahl Zwei- oder Vierleitersystem.
nFnctSelChgOvrDly	UDINT	Umschaltverzögerung [s] von Heizen auf Kühlen und umgekehrt.
fRmTAdjLimit	REAL	Einstellbarer Bereich [K] der Sollwertverschiebung *fRmTAdj* : *-fRmTAdjLimit <= fRmTAdj <= +fRmTAdjLimit*

Eingänge CONSTANT

Name	Typ	Beschreibung
CtrlHtg	FB_BA_Loop [▶ 225]	Heizregler
CtrlCol	FB_BA_Loop [▶ 225]	Kühlregler

Variablen

Name	Typ	Beschreibung
RmTAdj	FB_BA_RmTAdj [▶ 309]	Baustein der die entsprechenden Sollwerte mit der Verschiebung beaufschlagt und explizit ausgibt.
EnergLvlSlcn	FB_BA_PrioSwi_UDI04 [▶ 443]	Priorisierende Auswahl und Umwandlung der möglichen Energieniveaus in einen Zahlenwert.
SpSlcnHtg	FB_BA_MUX_R04 [▶ 441]	Auswahl des Heizsollwertes
SpSclnCol	FB_BA_MUX_R04 [▶ 441]	Auswahl des Kühlsollwertes
FnctSel	FB_BA_FnctSel [▶ 306]	Funktionsauswahl Heizen oder Kühlen.

Voraussetzungen

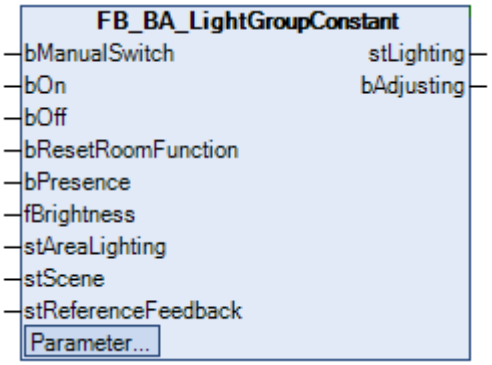
Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.4 Lighting

6.1.4.2.1.4.4.1 Functions

6.1.4.2.1.4.4.1.1 Zone

6.1.4.2.1.4.4.1.1.1 FB_BA_LightGroupConstant



In einer Licht-Zone werden ein oder mehrere Lichtaktoren zur gleichzeitigen Ansteuerung zusammengefasst, je nachdem wie es die örtlichen Gegebenheiten erfordern.

Dieses Template stellt eine Licht-Zone (Raum) mit Konstantlichtregelung und manueller Übersteuerung dar.

Funktion

Ist über den Parameter *eLightActivationMode* Vollautomatik eingestellt, so schaltet sich die Lichtfunktion ConstLgtCtrl [▶ 324] bei Präsenzerkennung am Eingang *bPresence* in den Regelbetrieb. Ist hingegen Halbautomatk angewählt, wird der erste Tastendruck am Template *bManualSwitch* den Baustein in den Regelbetrieb versetzen. Weitere Tastendrücke oder eine Wertvorgabe über die HMI (*fSetLightValueMan_In* / *SetLightValueMan_In*) brechen den Regelbetrieb ab und der Baustein befindet sich in der Hand-Übersteuerung.

In diesem Modus lässt sich das Licht über den Templateeingang *bManualSwitch* wechselweise ein- und ausschalten bzw. bei Langtastendruck (t>*SwiOvrTi*) auf- und abdimmen. Die Eingänge *bOn* und *bOff* lassen das Licht gezielt ein- und ausschalten während über eine HMI durch den Befehl *bSetLightValueMan_In* der Lichtwert *fSetLightValueMan_In* direkt gesetzt wird.

Der Licht-Steuerbaustein [▶ 324] zeigt seinen derzeitigen Status "Regelbetrieb" oder "Handübersteuerung" an den Ausgängen *bControlMode* und *bManualMode* an; beide Ausgänge gleichzeitig werden nicht gesetzt. Mit diesen Ausgängen wird entweder über den Baustein CnstLgtAuto [▶ 316] ein Lichtausgabetelegramm mit der Priorität eAutomaticLight [▶ 273] erzeugt oder über ManMode [▶ 316] ein Telegramm der Priorität eManual [▶ 273]. Die unterschiedliche Telegrammpriorität wird am Baustein PrioSwi [▶ 317] der internen HMI-Variable *eActualPrio_Out* zugewiesen. So kann über eine HMI eine Unterscheidung Regelbetrieb oder Handübersteuerung zur Anzeige gebracht werden.

Bei einer fallenden Flanke am Templateeingang *bPresence* oder durch Setzen von *bResetRoomFunction* wird der Lichtsteuerbaustein deaktiviert, wobei er, falls angewählt, noch über eine Ausschaltrampe mit Verweilzeit auf einem Vorab-Ausschhaltwert sein Abschalten ankündigt. Danach werden beide Bausteinausgänge *bControlMode* und *bManualMode* abgelöscht.

Neben den internen Telegrammen von *bControlMode* und *ManMode* werden auch die eingangsseitigen *stAreaLighting* und *stScene* mit auf den Selektor (*PrioSwi*) geführt.

Dabei stellt *stArea* das resultierende Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building) dar, während stScene für eine beliebige, die Konstantlichtregelung zu übersteuernde Lichtszene reserviert ist.

Der Telegrammselektor (*PrioSwi*) leitet das höchst priorisierte Telegramm durch - bei gleicher Priorität das zuletzt gesendete (last writer wins).

Der Sollwert für die Konstantlichtregelung wird mithilfe eines Analogobjekts SpBrightness [▶ 207] eingestellt.

Eingang stReferenceFeedback

Über diesen Eingang werden Informationen über den angesteuerten Lichtaktor bzw. den Referenzaktor einer Gruppe in die Lichtsteuerfunktion zurückgeführt.

Hierbei ist der Lichtwert des Referenzaktors von besonderer Bedeutung: Ist der Lichtaktor bereits von einer anderen Funktion gleicher oder niederer Priorität angesteuert und das hier beschriebene Template "übernimmt" nun die Kontrolle, so muss es den Aktor ausgehend von seinem bestehenden Lichtwert aus ansteuern.

Darüber hinaus werden aktuelle Werte und Status übertragen, die über die HMI zur Anzeige gebracht werden können.

Datenaustausch HMI

Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_LightConstCtrl* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

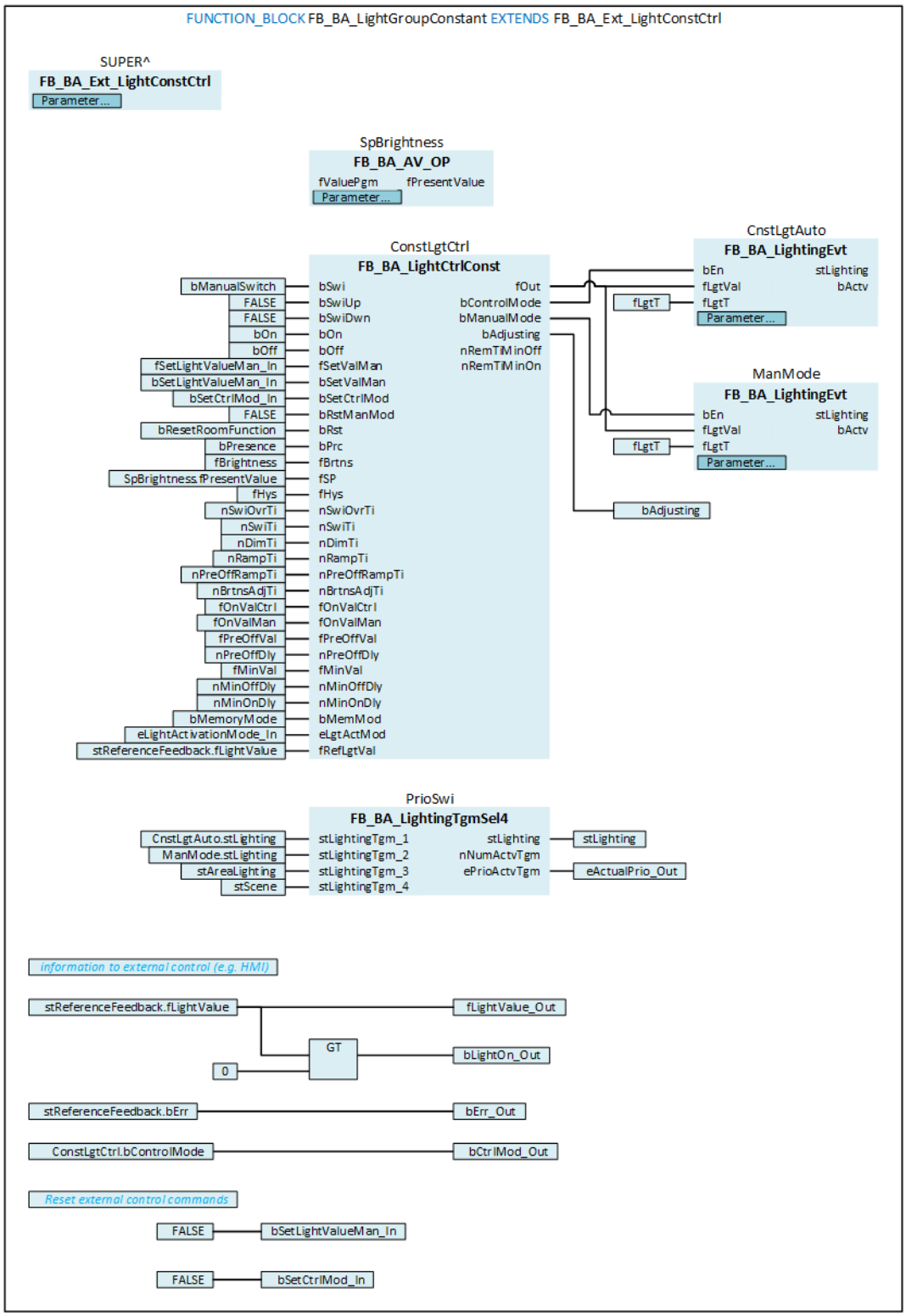
- **bSetLightValueMan_In**: Befehl in den Handbetrieb zu wechseln.
- **fSetLightValueMan_In**: Lichtwert, der dabei angenommen werden soll.
- **bSetCtrlMod_In**: Befehl in den Konstantlichtregelbetrieb zu wechseln.
- **eLightActivationMode_In**: Aktivierungsmodus der Lichtsteuerfunktion [▶ 324] aus der HMI (Vollautomatik oder Halbautomatik). Der über die HMI gewählte Modus persistent gespeichert.

- **fLightValue_Out**: Ausgabe prozentualer Lichtwert an die HMI.
- **bLightOn_Out**: Ausgabeinformation "Licht Ein" an die HMI.
- **bErr_Out**: Ausgabeinformation "Referenzaktor in Störung" an die HMI.
- **eActualPrio_Out**: Aktuell steuernde Telegrammpriorität an die HMI.
- **bCtrlMod_Out**: Ausgabeinformation "Lichtfunktion ist im Regelbetrieb" an die HMI.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupConstant EXTENDS FB_BA_Ext_LightConstCtrl
VAR_INPUT
  bManualSwitch          : BOOL;
  bOn                    : BOOL;
  bOff                   : BOOL;
  bResetRoomFunction     : BOOL;
  bPresence              : BOOL;
  fBrightness            : REAL;

  stAreaLighting         : ST_BA_Lighting;
  stScene                : ST_BA_Lighting;
```

```
  stReferenceFeedback    : ST_BA_LightActuatorFeedback;
END_VAR
VAR_OUTPUT
  stLighting             : ST_BA_Lighting;
  bAdjusting             : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  fHys                   : REAL;
  nSwiOvrTi              : UDINT;
  nSwiTi                 : UDINT;
  nDimTi                 : UDINT;
  nRampTi                : UDINT;
  nPreOffRampTi          : UDINT;
  nBrtnsAdjTi            : UDINT;
  fOnValCtrl             : REAL;
  fOnValMan              : REAL;
  fPreOffVal             : REAL;
  nPreOffDly             : UDINT;
  fMinVal                : REAL;
  nMinOffDly             : UDINT;
  nMinOnDly              : UDINT;
  bMemoryMode            : BOOL;
  fLgtT                  : REAL;
END_VAR
VAR_INPUT CONSTANT
  SpBrightness           : FB_BA_AV_Op;
  ManMode                : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eManual);
  CnstLgtAuto            : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eAutomaticLight);
END_VAR
VAR
  ConstLgtCtrl           : FB_BA_LightCtrlConst;
  PrioSwi                : FB_BA_LightingTgmSel4;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bManualSwitch	BOOL	Ist der Konstantlichtregelbaustein noch nicht aktiv, so schaltet ein kurzer Tastendruck zunächst die Regelung ein, danach lassen weitere Signale den Baustein in die Handfunktion übergehen und es gilt: kurzer Tastendruck: an/aus, lange Tastendrücke: wechselweise hoch- oder herunterdimmen.
bOn	BOOL	Ist der Konstantlichtregelbaustein noch nicht aktiv, so schaltet ein TRUE-Signal zunächst die Regelung ein, danach bezieht sich dieser Eingang nur auf das Einschalten im Hand-Modus.
bOff	BOOL	Schaltet das Licht aus, der Konstantlichtregelbaustein befindet sich weiterhin im Handbetrieb. War der Konstantlichtregelbaustein vorher im Regelbetrieb, so befindet er sich nun im Handbetrieb.
bResetRoomFunction	BOOL	Dieser Eingang schaltet den Konstantlichtregelbaustein ab. Die Abschaltung erfolgt über eine Rampe und Verweilzeit auf einem Basislichtwert, siehe Parameter: *fPreOffVal*, *nPreOffDly* und *nPreOffRampT*.
bPresence	BOOL	Präsenzmeldeeingang. Ist der Konstantlichtregelbaustein im Vollautomatikbetrieb konfiguriert, so kann über diesen Eingang eine steigende Flanke die Funktion ein- und eine fallende Flanke ausschalten. Im Halbautomatikbetrieb schaltet nur eine fallende Flanke an diesem Eingang den Konstantlichtregelbaustein aus.
fBrightness	REAL	Aktuelle Helligkeit für die Konstantlichtautomatik: Bereich und Einheit richten sich nach dem Helligkeitssensor.
stAreaLighting	ST_BA_Lighting [▶ 277]	Resultierendes Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building), welches im Baustein FB_BA_AreaLighting [▶ 936] gebildet wird.
stScene	ST_BA_Lighting [▶ 277]	Reservierter Telegramm-Eingang für eine Szenenansteuerung.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldeeingang des angesteuerten Lichtaktors bzw. der Führungsleuchte der angesteuerten Gruppe.

Ausgänge

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Resultierendes Ausgabeprogramm.
bAdjusting	BOOL	Der Konstantlichtregelbaustein befindet sich im Ausregelbetrieb. Dieses Signal kann dazu genutzt werden, Helligkeitssensoren, welche nicht analog, sondern über eine Kommunikation arbeiten, häufiger abzufragen und damit ein günstigeres Regelverhalten zu erzielen.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fHys	REAL	Konstantlicht-Automatik: Hystereseband. Die Konstantlichtregelung steuert den Ausgang so weit hoch, bzw. herunter, bis die gemessene Helligkeit den Zielwert erreicht, bzw. leicht über- oder unterschreitet. Das Licht gilt dann als ausgeregelt. Erst wenn die gemessene Helligkeit den Sollwert um *fHys*/2 über- oder unterschreitet, wird erneut nachgeregelt. Bereich und Einheit richten sich nach dem Helligkeitssensor, siehe oben: *fBrtnsSP* und *fBrtnsSen*. In *FB_Init* voreingestellt auf 50.
nSwiOvrTi	UDINT	Unterscheidungszeit [ms] zwischen Kurz- und Langtastendruck. In *FB_Init* voreingestellt auf 250ms.
nSwiTi	UDINT	Rampe für die Schaltfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 2s.
nDimTi	UDINT	Rampe für die Dimmfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 10s.
nRampTi	UDINT	Regelrampe der Konstantlicht-Automatik in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 60s.
nPreOffRampTi	UDINT	Rampe, mit vor dem Abschalten auf einen Basiswert *fPreOffVal* gefahren wird. In *FB_Init* voreingestellt auf 2s.
nBrtnsAdjTi	UDINT	Wartezeit in Sekunden nach Einschalten des Lichts, damit der Helligkeitssensor den korrekten Wert erfasst.
fOnValCtrl	REAL	Einschaltwert für den Regelbetrieb, sollte dieser bei vorher ausgeschalteter Funktion aktiviert werden. In *FB_Init* voreingestellt auf 50%.
fOnValMan	REAL	Einschaltwert der Handfunktion, sollte nicht über den Parameter *eOperationalMode* "Memory-Mode" angewählt sein. In *FB_Init* voreingestellt auf 100%.
fPreOffVal / nPreOffDly	REAL / UDINT	Basiswert und Haltezeit auf diesem Wert vor dem Ausschalten der Gesamtfunktion. Unterschreitet der aktuelle Lichtwert den Basiswert bereits, so ist diese Funktion nicht aktiv und es wird unmittelbar abgeschaltet. In *FB_Init* voreingestellt auf 4% und 10s.
fMinVal	REAL	Konstantlicht-Automatik: Mindestausgabewert. Wird dieser Wert intern unterschritten (d.h. es ist hell genug, dass kein künstliches Licht benötigt wird), so schaltet die Konstantlichtregelung das Licht nach Ablauf von *nOffDly* in Sekunden ab. Ist das Licht ausgeschaltet und die Regelung erkennt wieder Bedarf an Licht über dem Minimumwert, so schaltet die Regelung nach *nOnDly* in Sekunden das Licht wieder auf zunächst *fMinVal* ein. In *FB_Init* voreingestellt auf 4%
nMinOffDly	UDINT	Konstantlicht-Automatik: Abschaltwartezeit in Sekunden, siehe *fMinVal*. In *FB_Init* voreingestellt auf 300s.
nMinOnDly	UDINT	Konstantlicht-Automatik: Einschaltwartezeit in Sekunden, siehe *fMinVal*. In *FB_Init* voreingestellt auf 300s.
bMemoryMode	BOOL	Memory-Mode: Beim Einschalten des Lichts im Handbetrieb nimmt das Licht den Wert an, den die Funktion vor dem letzten Ausschalten innehatte. Ist der "Memory-Mode" nicht aktiv, so ist der Einschaltwert über *fOnValMan* definiert. In *FB_Init* voreingestellt auf FALSE.
fLgtT	REAL	Lichttemperatur: In *FB_Init* voreingestellt auf 3000K.

Eingänge CONSTANT

Name	Typ	Beschreibung
SpBrightness	FB_BA_AV_Op [▶ 207]	Analoges Eingabeobjekt: Sollwert der Regelungs [lx].
ManMode	FB_BA_LightingEvt [▶ 316]	Telegrammerzeuger für die Lichtwerte im Handbetrieb (*bManualMode* = TRUE). Die Priorität ist voreingestellt auf *eManualActuator*.
CnstLgtAuto	FB_BA_LightingEvt [▶ 316]	Telegrammerzeuger für die Lichtwerte im Regelbetrieb (*bControlMode* = TRUE). Die Priorität ist voreingestellt auf *eConstantLightControl*.

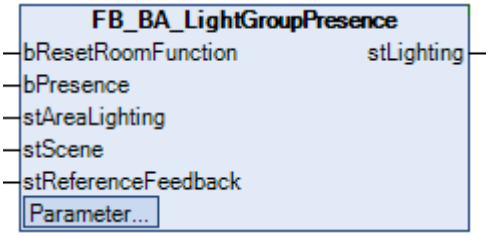
VAR

Name	Typ	Beschreibung
ConstLgtCtrl	FB_BA_LightCtrlConst [▶ 324]	Konstantlichtregelbaustein.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 317]	Telegrammauswahl.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.4.1.1.2 FB_BA_LightGroupPresence



In einer Licht-Zone werden ein oder mehrere Lichtaktoren zur gleichzeitigen Ansteuerung zusammengefasst, je nachdem wie es die örtlichen Gegebenheiten erfordern.

Dieses Template stellt eine Licht-Zone (Raum) dar, welche nur über Präsenz geschaltet wird. Typische Anwendungsfälle sind Sanitärbereiche oder innenliegende Flure.

Funktion

Eine steigende Flanke an *bPresence* schaltet das Licht über den internen Lichtsteuerbaustein LightSwitch [▶ 318] ein.

Der Ausgang *bAutomaticMode* wechselt auf TRUE und der Lichtwert wird über den Baustein LightEvent [▶ 316] als Telegramm ST_BA_Lighting [▶ 277] ausgegeben.

Bei einer fallenden Flanke an *bPresence* wird das Licht zunächst über eine Rampe auf einen Minimalwert gedimmt (falls parametriert, siehe FB_BA_LightCtrl [▶ 318]) und dann abgeschaltet. Der Ausgang *bAutmaticMode* wechselt auf FALSE und das Lichttelegramm wird am Baustein LightEvent [▶ 316] deaktiviert.

Über eine steigende Flanke den Eingang *bResetRoomFunction* kann eine vorzeitige Abschaltung und Deaktivierung erreicht werden.

Eingang stReferenceFeedback

Über diesen Eingang werden Informationen über den angesteuerten Lichtaktor bzw. den Referenzaktor einer Gruppe in die Lichtsteuerfunktion zurückgeführt.

Hierbei ist der Lichtwert des Referenzaktors von besonderer Bedeutung: Ist der Lichtaktor bereits von einer anderen Funktion gleicher oder niederer Priorität angesteuert und das hier beschriebene Template "übernimmt" nun die Kontrolle, so muss es den Aktor ausgehend von seinem bestehenden Lichtwert aus ansteuern.

Darüber hinaus werden aktuelle Werte und Status übertragen, die über die HMI zur Anzeige gebracht werden können.

Datenaustausch HMI

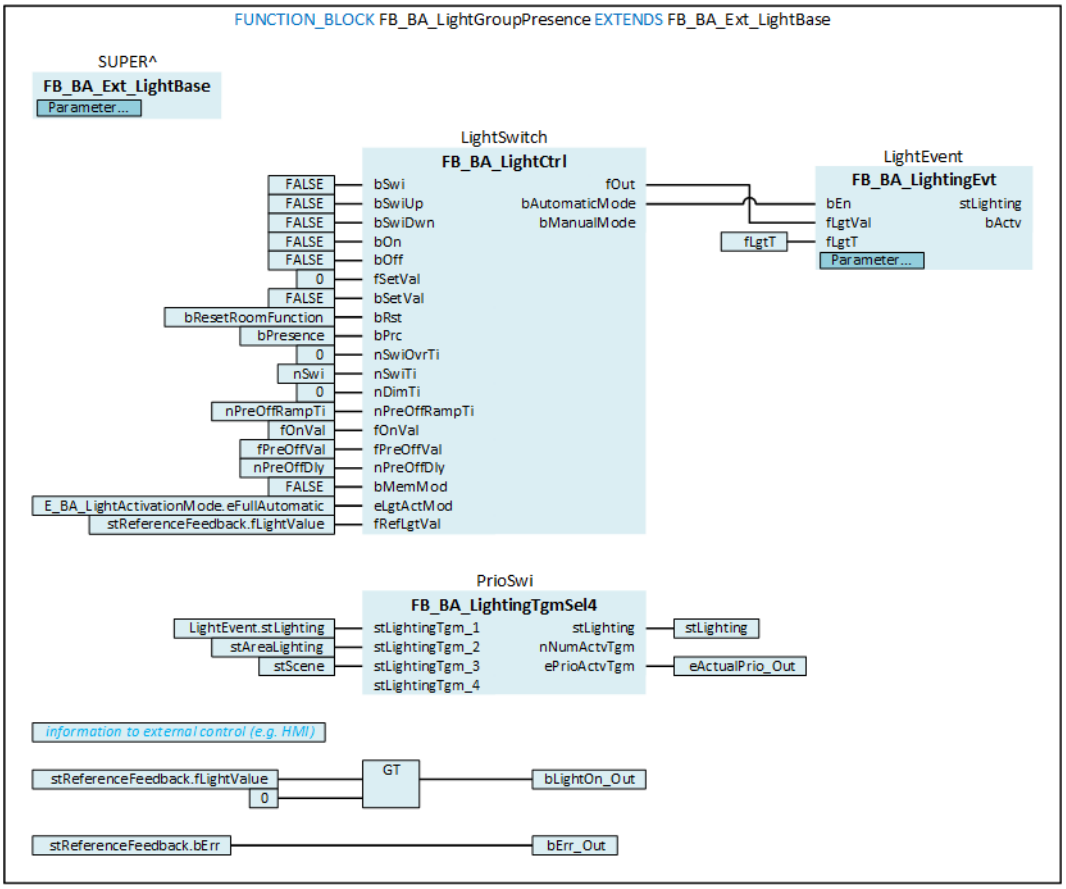
Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_LightBase* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

- **bLightOn_Out**: Ausgabeinformation "Licht Ein" an die HMI.
- **bErr_Out**: Ausgabeinformation "Referenzaktor in Störung" an die HMI.
- **eActualPrio_Out**: Aktuell steuernde Telegrammpriorität an die HMI.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupPresence EXTENDS FB_BA_Ext_LightBase
VAR_INPUT
  bResetRoomFunction     : BOOL;
  bPresence              : BOOL;
  stAreaLighting         : ST_BA_Lighting;
  stScene                : ST_BA_Lighting;
  stReferenceFeedback    : ST_BA_LightActuatorFeedback;
```

```
END_VAR
VAR_OUTPUT
  stLighting             : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiTi                 : UDINT;
  nPreOffRampTi          : UDINT;
  fOnVal                 : REAL;
  fPreOffVal             : REAL;
  nPreOffDly             : UDINT;
  fLgtT                  : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  LightEvent             : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eSimple);
END_VAR
VAR
  LightSwitch            : FB_BA_LightCtrl;
  PrioSwi                : FB_BA_LightingTgmSel4;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bResetRoomFunction	BOOL	Dieser Eingang schaltet den Lichtbaustein *LightSwitch* ab - er ist dann nicht mehr aktiv (Ausgang *bAutmaticMode*/ *bManualMode* = FALSE). Die Abschaltung erfolgt über eine Rampe und Verweilzeit auf einem Basislichtwert, siehe Parameter: *fPreOffVal*, *nPreOffDly* und *nPreOffRampT*.
bPresence	BOOL	Präsenzmeldeeingang. Eine steigende Flanke die schaltet das Licht ein und eine fallende Flanke aus. Das Ein- und Ausschalten über Präsenz geht mit einer Aktivierung und Deaktivierung des Lichtbausteins [▶ 318] einher.
stAreaLighting	ST_BA_Lighting [▶ 277]	Resultierendes Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building), welches im Baustein FB_BA_AreaLighting [▶ 936] gebildet wird.
stScene	ST_BA_Lighting [▶ 277]	Reservierter Telegramm-Eingang für eine Szenenansteuerung.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldeeingang des angesteuerten Lichtaktors bzw. der Führungsleuchte der angesteuerten Gruppe.

Ausgänge

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Resultierendes Ausgabeprogramm.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nSwiTi	UDINT	Rampe für die Schaltfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 2s.
nPreOffRampTi	UDINT	Rampe, mit vor dem Abschalten auf einen Basiswert *fPreOffVal* gefahren wird. In *FB_Init* voreingestellt auf 2s.
fOnVal	REAL	Einschaltwert. In *FB_Init* voreingestellt auf 100%.
fPreOffVal / nPreOffDly	REAL / UDINT	Basiswert und Haltezeit auf diesem Wert vor dem Ausschalten der Gesamtfunktion. Unterschreitet der aktuelle Lichtwert den Basiswert bereits, so ist diese Funktion nicht aktiv und es wird unmittelbar abgeschaltet. In *FB_Init* voreingestellt auf 4% und 10s.
fLgtT	REAL	Lichttemperatur: In *FB_Init* voreingestellt auf 3000K.

Eingänge CONSTANT

Name	Typ	Beschreibung
LightEvent	FB_BA_LightingEvt [▶ 316]	Telegrammerzeuger für die Lichtwerte. Die Priorität ist voreingestellt auf *eManualActuator*.

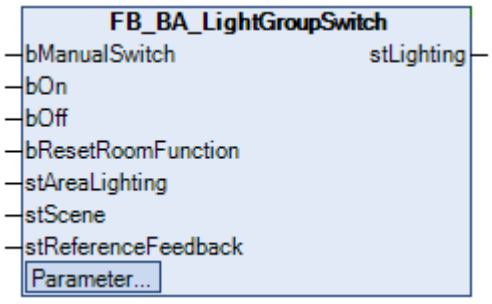
VAR

Name	Typ	Beschreibung
LightSwitch	FB_BA_LightCtrl [▶ 318]	Lichtsteuerbaustein.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 317]	Telegrammauswahl.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.4.1.1.3 FB_BA_LightGroupSwitch



In einer Licht-Zone werden ein oder mehrere Lichtaktoren zur gleichzeitigen Ansteuerung zusammengefasst, je nachdem wie es die örtlichen Gegebenheiten erfordern.

Dieses Template stellt eine Licht-Zone (Raum) dar, welche nur über Schalter bedient wird.

Funktion

Über den Templateeingang *bManualSwitch* lässt sich das Licht über den Steuerbaustein LightSwitch [▶ 318] wechselweise ein- und ausschalten bzw. bei Langtastendruck (t > *SwiOvrTi*) auf- und abdimmen.

Die Eingänge *bOn* und *bOff* lassen das Licht gezielt ein- und ausschalten während über eine HMI durch den Befehl *bSetLightValueMan_In* der Lichtwert *fSetLightValueMan_In* direkt gesetzt wird.

Durch Handbefehle (nicht Präsenz) geschaltet arbeitet der Lichtbaustein FB_BA_LightCtrl [▶ 318] normalerweise so, dass er immer aktiv ist und dies am Ausgang *bManualMode* anzeigt. Das ist dadurch begründet, dass der Anwender auch einmal gezielt das Licht abschalten möchte und es dadurch nicht zu der Situation kommen darf, dass der Baustein inaktiv wird und ein Lichttelegramm niedrigerer Priorität die Aktoren schaltet.

Normalerweise deaktiviert dann eine fallende Flanke am Präsenzeingang *bPrc* die Funktion. Da dies ohne Präsenzerkennung nicht möglich ist, ist der Lichtfunktion eine Logik voran geschaltet, die bei Erreichen von 0% am Lichtausgang einen Triggerimpuls an den Reset-Eingang *bRst* sendet und den Lichtsteuereingang so trotzdem deaktiviert. Wahlweise kann die Deaktivierung auch über den Templateeingang *bResetRoomFunction* erfolgen.

Im Falle des eingeschalteten Lichts wird der Baustein LightEvent [▶ 316] durch *bManualMode* = TRUE aktiviert und ein Lichttelegramm [▶ 277] mit dem gesetzten Lichtwert ausgegeben.

Neben dem internen Telegramm von *LightEvent* werden die eingangsseitigen Telegramme *stAreaLighting* und *stScene* mit auf einen Selektor (*PrioSwi*) geführt.

Dabei stellt *stArea* das resultierende Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building) dar, während *stScene* für eine beliebige, die Konstantlichtregelung zu übersteuernde Lichtszene reserviert ist.

Der Telegrammselektor (*PrioSwi*) leitet das höchst priorisierte Telegramm durch - bei gleicher Priorität das zuletzt gesendete (last writer wins).

Eingang stReferenceFeedback

Über diesen Eingang werden Informationen über den angesteuerten Lichtaktor bzw. den Referenzaktor einer Gruppe in die Lichtsteuerfunktion zurückgeführt.

Hierbei ist der Lichtwert des Referenzaktors von besonderer Bedeutung: Ist der Lichtaktor bereits von einer anderen Funktion gleicher oder niederer Priorität angesteuert und das hier beschriebene Template "übernimmt" nun die Kontrolle, so muss es den Aktor ausgehend von seinem bestehenden Lichtwert aus ansteuern.

Darüber hinaus werden aktuelle Werte und Status übertragen, die über die HMI zur Anzeige gebracht werden können.

Datenaustausch HMI

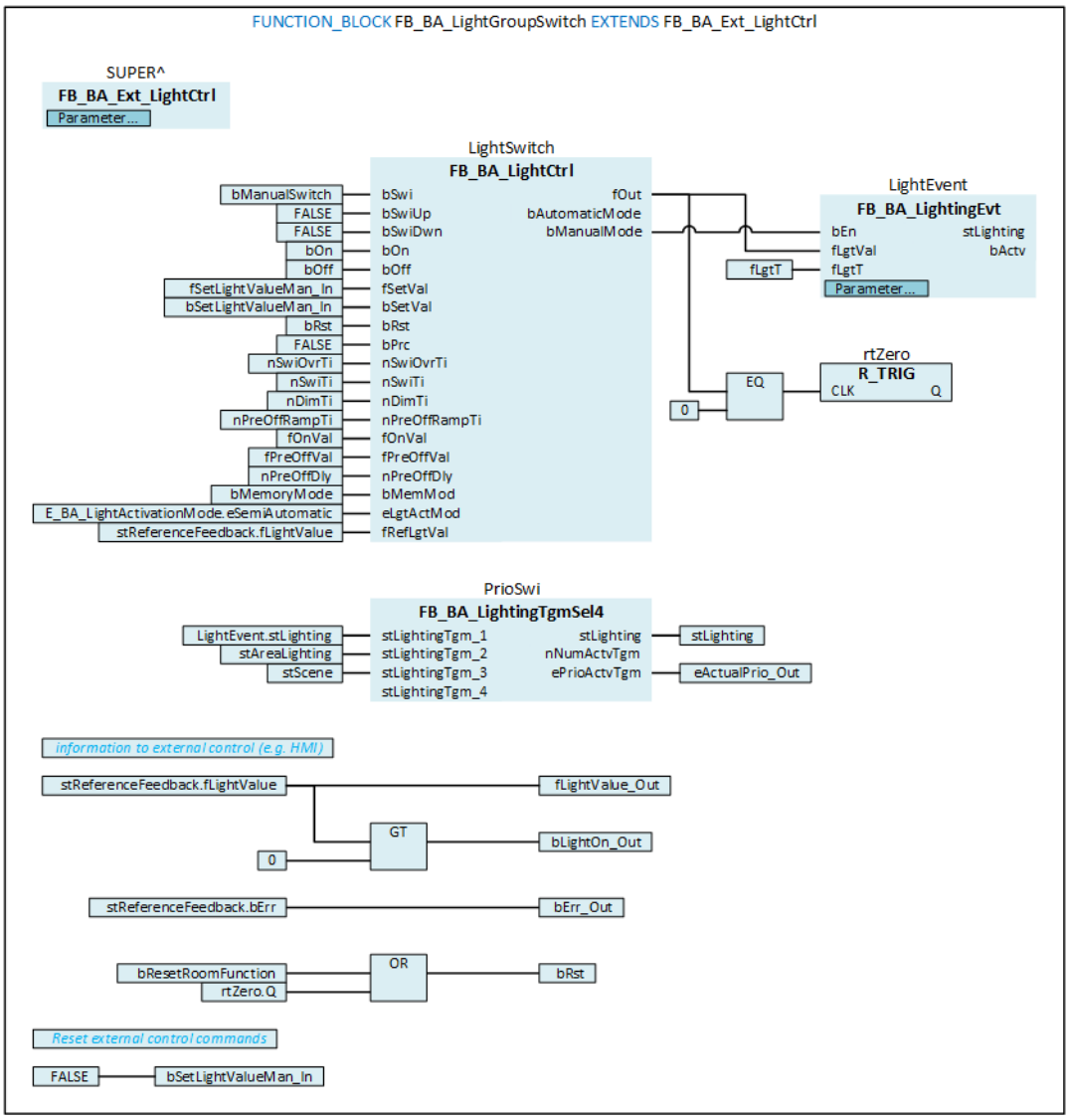
Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_LightCtrl* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

- **fSetLightValueMan_In**: Lichtwert, der über die HMI gesetzt werden soll.
- **bSetLightValueMan_In**: Befehl von der HMI, dass der obige Lichtwert gesetzt werden soll.
- **bLightOn_Out**: Ausgabeinformation "Licht Ein" an die HMI.
- **fLightValue_Out**: Ausgabeinformation Lichtwert an die HMI.
- **bErr_Out**: Ausgabeinformation "Referenzaktor in Störung" an die HMI.
- **eActualPrio_Out**: Aktuell steuernde Telegrammpriorität an die HMI.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupSwitch EXTENDS FB_BA_Ext_LightCtrl
VAR_INPUT
  bManualSwitch           : BOOL;
  bOn                     : BOOL;
  bOff                    : BOOL;
  bResetRoomFunction      : BOOL;
  stAreaLighting          : ST_BA_Lighting;
  stScene                 : ST_BA_Lighting;
  stReferenceFeedback     : ST_BA_LightActuatorFeedback;
END_VAR
VAR_OUTPUT
  stLighting              : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiOvrTi               : UDINT;
  nSwiTi                  : UDINT;
  nDimTi                  : UDINT;
  nPreOffRampTi           : UDINT;
  fOnVal                  : REAL;
  fPreOffVal              : REAL;
  nPreOffDly              : UDINT;
  bMemoryMode             : BOOL;
  eLightActivationMode    : E_BA_LightActivationMode;
  fLgtT                   : REAL;
END_VAR
VAR_INPUT CONSTANT
  LightEvent              : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eSimple);
END_VAR
VAR
```

```
  LightSwitch             : FB_BA_LightCtrl;
  PrioSwi                 : FB_BA_LightingTgmSel4;
  rtZero                  : R_TRIG;
  bRst                    : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bManualSwitch	BOOL	Kurze Tastendrücke an diesem Eingang schalten das Licht an und aus. Lange Tastendrücke (t > *SwiOvrTi*) lassen das Licht auf- und abdimmen.
bOn	BOOL	Schaltet das Licht ein und aktiviert den Lichtsteuerbaustein [▶ 936].
bOff	BOOL	Schaltet das Licht aus. Bedingt durch eine Zusatzbeschaltung, die bei Erreichen von 0% am Lichtsteuerbaustein [▶ 936] einen Triggerimpuls an dessen Eingang *bRst* gibt, wird auch gleichzeitig der Baustein deakiviert.
bResetRoomFunction	BOOL	Dieser Eingang schaltet den Lichtbaustein *LightSwitch* ab - er ist dann nicht mehr aktiv (Ausgang *bAutmaticMode*/ *bManualMode* = FALSE). Die Abschaltung erfolgt über eine Rampe und Verweilzeit auf einem Basislichtwert, siehe Parameter: *fPreOffVal*, *nPreOffDly* und *nPreOffRampT*.
stAreaLighting	ST_BA_Lighting [▶ 277]	Resultierendes Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building), welches im Baustein FB_BA_AreaLighting [▶ 936] gebildet wird.
stScene	ST_BA_Lighting [▶ 277]	Reservierter Telegramm-Eingang für eine Szenenansteuerung.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldeeingang des angesteuerten Lichtaktors bzw. der Führungsleuchte der angesteuerten Gruppe.

Ausgänge

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Resultierendes Ausgabeprogramm.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nSwiOvrTi	UDINT	Unterscheidungszeit [ms] zwischen Kurz- und Langtastendruck. In *FB_Init* voreingestellt auf 250 ms.
nSwiTi	UDINT	Rampe für die Schaltfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 2 s.
nDimTi	UDINT	Rampe für die Dimmfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 10 s.
nPreOffRampTi	UDINT	Rampe, mit vor dem Abschalten auf einen Basiswert *fPreOffVal* gefahren wird. In *FB_Init* voreingestellt auf 2 s.
fOnVal	REAL	Einschaltwert. In *FB_Init* voreingestellt auf 100%.
fPreOffVal / nPreOffDly	REAL / UDINT	Basiswert und Haltezeit auf diesem Wert vor dem Ausschalten der Gesamtfunktion. Unterschreitet der aktuelle Lichtwert den Basiswert bereits, so ist diese Funktion nicht aktiv und es wird unmittelbar abgeschaltet. In *FB_Init* voreingestellt auf 4% und 10s.
bMemoryMode	BOOL	Beim Einschalten des Lichts im Handbetrieb nimmt das Licht den Wert an, den die Funktion vor dem letzten Ausschalten innehatte. Ist der "Memory-Mode" nicht aktiv, so ist der Einschaltwert über *fOnValMan* definiert. In *FB_Init* voreingestellt auf FALSE.
eLightActivationMode	E_BA_LightActivationMode [▶ 272]	Auswahl der Funktion der Beleuchtungsautomatik: • Halbautomatik: Die Lichtfunktion wird nur über einen Tastendruck aktiviert, über eine wegfallende Präsenz wird sie automatisch deaktiviert. • Vollautomatik: Die Lichtfunktion kann sowohl über eine Präsenzerkennung als auch über einen Tastendruck aktiviert werden, über eine wegfallende Präsenz wird sie automatisch deaktiviert.
fLgtT	REAL	Lichttemperatur: In *FB_Init* voreingestellt auf 3000K.

Eingänge CONSTANT

Name	Typ	Beschreibung
LightEvent	FB_BA_LightingEvt [▶ 316]	Telegrammerzeuger für die Lichtwerte. Die Priorität ist voreingestellt auf *eManualActuator*.

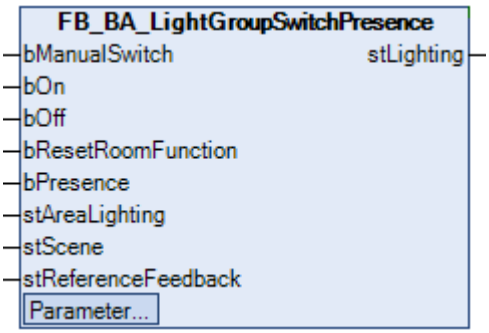
VAR

Name	Typ	Beschreibung
LightSwitch	FB_BA_LightCtrl [▶ 318]	Lichtsteuerbaustein.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 317]	Telegrammauswahl.
rtZero	R_TRIG	Trigger, der vom Lichtausgabewert "0" aktiviert wird.
bRst	BOOL	Interne Variable zum Deaktivieren der Gesamtfunktion. Es wird kein gültiges Lichtsteuertelegramm *stLighting* mehr ausgegeben.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.4.1.1.4 FB_BA_LightGroupSwitchPresence



In einer Licht-Zone werden ein oder mehrere Lichtaktoren zur gleichzeitigen Ansteuerung zusammengefasst, je nachdem wie es die örtlichen Gegebenheiten erfordern.

Dieses Template stellt eine Licht-Zone (Raum) dar, welche sowohl über Schalter als auch über Präsenzmelder bedient wird.

Funktion

Über den Templateeingang *bManualSwitch* lässt sich das Licht über den Steuerbaustein LightSwitch wechselweise ein- und ausschalten bzw. bei Langtastendruck (t > *SwiOvrTi*) auf- und abdimmen.

Die Eingänge *bOn* und *bOff* lassen das Licht gezielt ein- und ausschalten während über eine HMI durch den Befehl *bSetLightValueMan_In* der Lichtwert *fSetLightValueMan_In* direkt gesetzt wird.

Ist über den Parameter *eLightActivationMode* Vollautomatik-Betrieb eingestellt, so schaltet sich das Licht bei Präsenzerkennung am Eingang *bPresence* automatisch ein und bei fallender Flanke automatisch aus. Ist hingegen Halbautomatik-Betrieb angewählt, so erfolgt über den Präsenzeingang nur eine Abschaltung.

Der Licht-Steuerbaustein unterscheidet, ob er über Präsenz oder über Hand aktiviert bzw. Hand-übersteuert wurde. Im ersten Fall wird der Ausgang *bAutomaticMode* gesetzt, im zweiten Fall *bManualMode*; beide Ausgänge gleichzeitig werden nicht gesetzt. Mit diesen Ausgängen wird entweder über den Baustein AutoMode [▶ 316] ein Lichtausgabetelegramm mit der Priorität eAutomaticLight [▶ 273] erzeugt oder über ManMode [▶ 316] ein Telegramm der Priorität eManual [▶ 273]. Die unterschiedliche Telegrammpriorität wird am Baustein PrioSwi der internen HMI-Variable *eActualPrio_Out* zugewiesen. So kann über eine HMI eine Unterscheidung Automatik oder Hand/Handübersteuerung zur Anzeige gebracht werden.

Bei einer fallenden Flanke am Templateeingang *bPresence* oder durch das Setzen von *bResetRoomFunction* wird der Lichtsteuerbaustein deaktiviert, wobei er, falls angewählt, noch über eine Ausschaltrampe mit Verweilzeit auf einem Vor-Ausschhaltwert sein Abschalten ankündigt. Danach werden beide Bausteinausgänge *bAutomaticMode* und *bManualMode* abgelöscht.

Neben den internen Telegrammen von *AutoMode* und *ManMode* werden auch die eingangsseitigen stAreaLighting und stScene mit auf den Selektor (PrioSwi) geführt.

Dabei stellt *stArea* das resultierende Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building) dar, während *stScene* für eine beliebige, die Konstantlichtregelung zu übersteuernde Lichtszene reserviert ist.

Der Telegrammselektor (*PrioSwi*) leitet das höchst priorisierte Telegramm durch - bei gleicher Priorität das zuletzt gesendete (last writer wins).

Eingang stReferenceFeedback

Über diesen Eingang werden Informationen über den angesteuerten Lichtaktor bzw. den Referenzaktor einer Gruppe in die Lichtsteuerfunktion zurückgeführt.

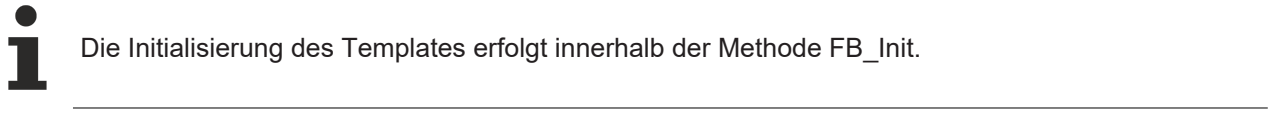
Hierbei ist der Lichtwert des Referenzaktors von besonderer Bedeutung: Ist der Lichtaktor bereits von einer anderen Funktion gleicher oder niederer Priorität angesteuert und das hier beschriebene Template "übernimmt" nun die Kontrolle, so muss es den Aktor ausgehend von seinem bestehenden Lichtwert aus ansteuern.

Darüber hinaus werden aktuelle Werte und Status übertragen, die über die HMI zur Anzeige gebracht werden können.

Datenaustausch HMI

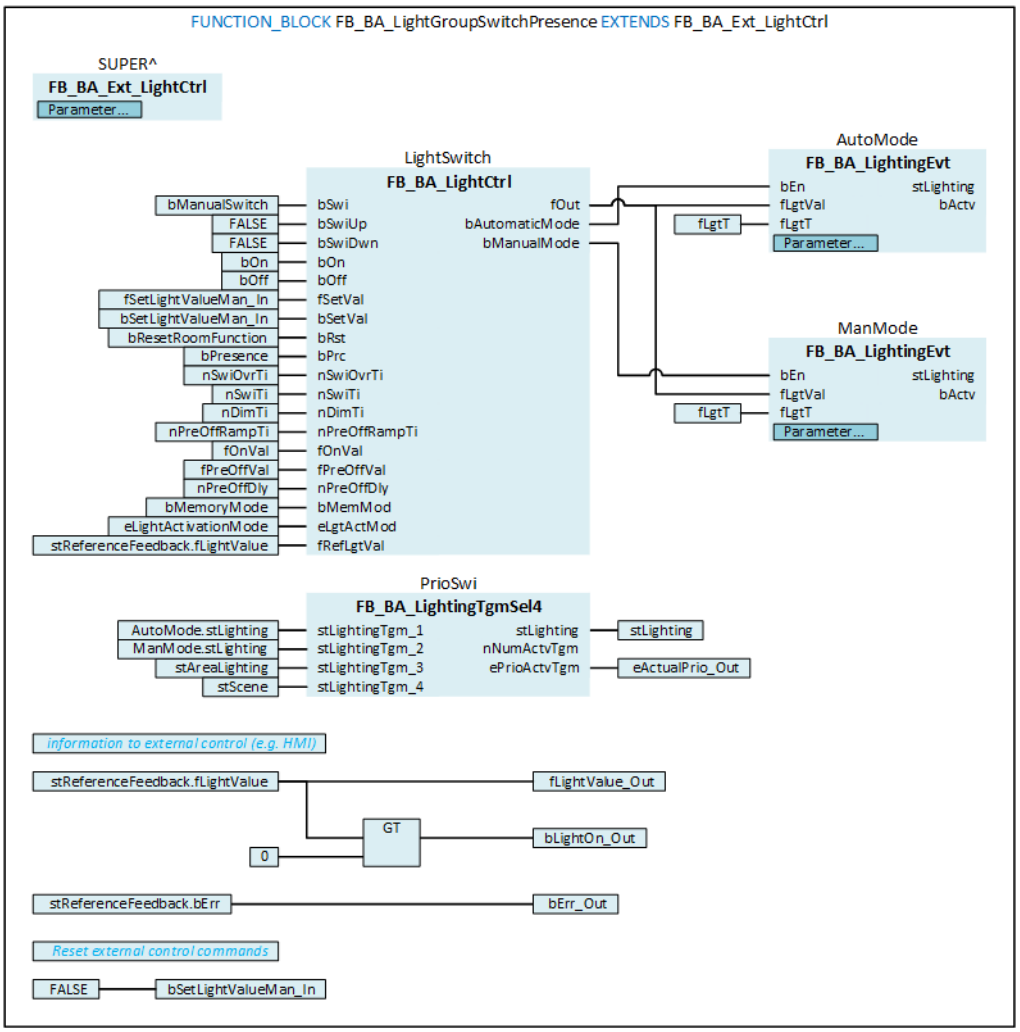
Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_LightCtrl* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

- **fSetLightValueMan_In**: Lichtwert, der über die HMI gesetzt werden soll.
- **bSetLightValueMan_In**: Befehl von der HMI, dass der obige Lichtwert gesetzt werden soll.
- **bLightOn_Out**: Ausgabeinformation "Licht Ein" an die HMI.
- **fLightValue_Out**: Ausgabeinformation Lichtwert an die HMI.
- **bErr_Out**: Ausgabeinformation "Referenzaktor in Störung" an die HMI.
- **eActualPrio_Out**: Aktuell steuernde Telegrammpriorität an die HMI.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupSwitch EXTENDS FB_BA_Ext_LightCtrl
VAR_INPUT
  bManualSwitch           : BOOL;
  bOn                     : BOOL;
  bOff                    : BOOL;
  bResetRoomFunction      : BOOL;
  bPresence               : BOOL;
  stAreaLighting          : ST_BA_Lighting;
  stScene                 : ST_BA_Lighting;
  stReferenceFeedback     : ST_BA_LightActuatorFeedback;
```

```
END_VAR
VAR_OUTPUT
  stLighting              : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiOvrTi               : UDINT;
  nSwiTi                  : UDINT;
  nDimTi                  : UDINT;
  nPreOffRampTi           : UDINT;
  fOnVal                  : REAL;
  fPreOffVal              : REAL;
  nPreOffDly              : UDINT;
  bMemoryMode             : BOOL;
  eLightActivationMode    : E_BA_LightActivationMode;
  fLgtT                   : REAL;
END_VAR
VAR_INPUT CONSTANT
  ManMode                 : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eManual);
  AutoMode                : FB_BA_LightingEvt :=( ePrio:= E_BA_LightingPrio.eAutomaticLight);
END_VAR
VAR
  LightSwitch             : FB_BA_LightCtrl;
  PrioSwi                 : FB_BA_LightingTgmSel4;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bManualSwitch	BOOL	Kurze Tastendrücke an diesem Eingang schalten das Licht an und aus. Lange Tastendrücke (t > *SwiOvrTi*) lassen das Licht auf- und abdimmen.
bOn	BOOL	Schaltet das Licht ein und aktiviert den Lichtsteuerbaustein [▶ 936].
bOff	BOOL	Schaltet das Licht aus. Bedingt durch eine Zusatzbeschaltung, die bei Erreichen von 0% am Lichtsteuerbaustein [▶ 936] einen Triggerimpuls an dessen Eingang *bRst* gibt, wird auch gleichzeitig der Baustein deakiviert.
bResetRoomFunction	BOOL	Dieser Eingang schaltet den Lichtbaustein *LightSwitch* ab - er ist dann nicht mehr aktiv (Ausgang *bAutmaticMode*/ *bManualMode* = FALSE). Die Abschaltung erfolgt über eine Rampe und Verweilzeit auf einem Basislichtwert, siehe Parameter: *fPreOffVal*, *nPreOffDly* und *nPreOffRampT*.
bPresence	BOOL	Präsenzmeldeeingang. Ist über den Parameter eLightActivationMode [▶ 272] Vollautomatik angewählt, so wird der Lichtbaustein [▶ 318] mit einer steigenden Flanke als "automatisch" aktiviert (Ausgang *bAutomaticMode* = TRUE), sofern er nicht vorher schon auf Hand/Hand-übersteuert stand (Ausgang *bManualMode* = TRUE). Eine fallende Flanke deaktiviert den Lichtsteuerbaustein (Ausgang bAutomaticMode=FALSE, bMaualMode=FALSE).
stAreaLighting	ST_BA_Lighting [▶ 277]	Resultierendes Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building), welches im Baustein FB_BA_AreaLighting [▶ 936] gebildet wird.
stScene	ST_BA_Lighting [▶ 277]	Reservierter Telegramm-Eingang für eine Szenenansteuerung.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldeeingang des angesteuerten Lichtaktors bzw. der Führungsleuchte der angesteuerten Gruppe.

Ausgänge

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Resultierendes Ausgabeprogramm.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nSwiOvrTi	UDINT	Unterscheidungszeit [ms] zwischen Kurz- und Langtastendruck. In *FB_Init* voreingestellt auf 250 ms.
nSwiTi	UDINT	Rampe für die Schaltfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 2 s.
nDimTi	UDINT	Rampe für die Dimmfunktionen in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In *FB_Init* voreingestellt auf 10 s.
nPreOffRampTi	UDINT	Rampe, mit vor dem Abschalten auf einen Basiswert *fPreOffVal* gefahren wird. In *FB_Init* voreingestellt auf 2 s.
fOnVal	REAL	Einschaltwert. In *FB_Init* voreingestellt auf 100%.
fPreOffVal / nPreOffDly	REAL / UDINT	Basiswert und Haltezeit auf diesem Wert vor dem Ausschalten der Gesamtfunktion. Unterschreitet der aktuelle Lichtwert den Basiswert bereits, so ist diese Funktion nicht aktiv und es wird unmittelbar abgeschaltet. In *FB_Init* voreingestellt auf 4% und 10s.
bMemoryMode	BOOL	Beim Einschalten des Lichts im Handbetrieb nimmt das Licht den Wert an, den die Funktion vor dem letzten Ausschalten innehatte. Ist der "Memory-Mode" nicht aktiv, so ist der Einschaltwert über *fOnValMan* definiert. In *FB_Init* voreingestellt auf FALSE.
eLightActivationMode	E_BA_LightActivationMode [▶ 272]	Auswahl der Funktion der Beleuchtungsautomatik: • Halbautomatik: Die Lichtfunktion wird nur über einen Tastendruck aktiviert, über eine wegfallende Präsenz wird sie automatisch deaktiviert. • Vollautomatik: Die Lichtfunktion kann sowohl über eine Präsenzerkennung als auch über einen Tastendruck aktiviert werden, über eine wegfallende Präsenz wird sie automatisch deaktiviert.
fLgtT	REAL	Lichttemperatur: In *FB_Init* voreingestellt auf 3000K.

Eingänge CONSTANT

Name	Typ	Beschreibung
ManMode	FB_BA_LightingEvt [▶ 316]	Telegrammerzeuger für die Lichtwerte. Die Priorität ist voreingestellt auf *eManualActuator*.
AutoMode	FB_BA_LightingEvt [▶ 316]	Telegrammerzeuger für die Lichtwerte, sollte der Baustein über den Präsenzmelder aktiviert worden und (noch) nicht handübersteuert sein. Die Priorität ist voreingestellt auf *eAutomaticLight*.

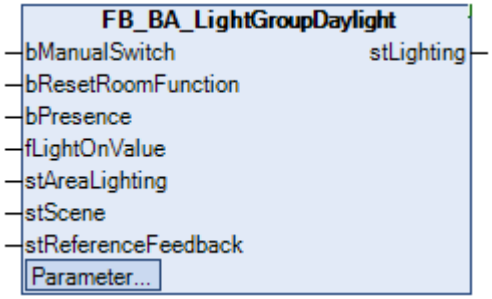
VAR

Name	Typ	Beschreibung
LightSwitch	FB_BA_LightCtrl [▶ 318]	Lichtsteuerbaustein.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 317]	Telegrammauswahl.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.4.1.1.5 FB_BA_LightGroupDaylight



Dieses Template stellt eine tageslichtabhängige Lichtsteuerung dar, welche präsenzabhängig automatisch ein- und ausschaltet. Sie ist für Flure und Durchgangsbereiche gedacht. Eine vorgelagerte, optimierende Einschaltfuntkion FB_BA_LightCtrlValue [▶ 321] sorgt für die Vorgabe eines Lichteinschaltwertes, der die nachgelagerte Konstantlichtregelung nahe an den gewünschten Sollwert schaltet. Sobald das Licht über Präsenz eingeschaltet ist, wird es automatisch dem Sollwert entsprechend nachgeregelt.

Funktion

Im Gegensatz zum Template FB_BA_LightGroupConstant [▶ 908] (Konstantlichtregelung) wird hier nur der Regelbetrieb des Bausteines FB_BA_LightCtrlConst [▶ 324] genutzt. Ein manuelles Übersteuern der Konstantlichtregelung ist für Flure und Durchgangsbereiche nicht notwendig, weshalb auf einen Schaltereingang verzichtet wird. Ist dennoch ein manuelles Einschalten erwünscht, so ist der Schalter wie ein Präsenzmelder zu behandeln und **nicht** an die Eingänge *bSwi*, *bSwiUp* oder *bSwiDwn* zu verknüpfen, da diese die Übersteuerung ohne Regelung auslösen.

Eine steigende Flanke an *bPresence* aktiviert den Baustein und er wird direkt auf den durch FB_BA_LightCtrlValue [▶ 321] optimierten Einschaltwert gesetzt.

Bei einer fallenden Flanke am Templateeingang *bPresence* oder durch Setzen von *bResetRoomFunction* wird der Lichtsteuerbaustein deaktiviert, wobei er, falls angewählt, noch über eine Ausschaltrampe mit Verweilzeit auf einem Vor-Ausschhaltwert sein Abschalten ankündigt. Danach ist die Funktion inaktiv und der Ausgang *bControlMode* wird abgelöscht.

Neben den internen Telegrammen von *LightEvent* und *OnValue* (Einlernfunktion) werden auch die eingangsseitigen *stAreaLighting* und *stScene* mit auf den Selektor (*PrioSwi*) geführt.

Dabei stellt *stArea* das resultierende Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building) dar, während *stScene* für eine beliebige, die Konstantlichtregelung zu übersteuernde Lichtszene reserviert ist.

Das Telegramm *OnValue.stTeachLighting* ist nur während des Einlernvorgangs von *OnValue* (FB_BA_LightCtrlValue [▶ 321]) aktiv.

Der Telegrammselektor (*PrioSwi*) leitet das höchst priorisierte Telegramm durch - bei gleicher Priorität das zuletzt gesendete (last writer wins).

Der Sollwert für die Konstantlichtregelung wird mithilfe eines Analogobjekts SpBrightness [▶ 207] eingestellt.

Eingang stReferenceFeedback

Über diesen Eingang werden Informationen über den angesteuerten Lichtaktor bzw. den Referenzaktor einer Gruppe in die Lichtsteuerfunktion zurückgeführt.

Hierbei ist der Lichtwert des Referenzaktors von besonderer Bedeutung: Ist der Lichtaktor bereits von einer anderen Funktion gleicher oder niederer Priorität angesteuert und das hier beschriebene Template "übernimmt" nun die Kontrolle, so muss es den Aktor ausgehend von seinem bestehenden Lichtwert aus ansteuern.

Der Lichtvorgabebaustein *fOnValue* benötigt ebenfalls die Information des aktuellen Lichtwertes der angesteuerten Gruppe. Da er selbst nur den Helligkeitszugewinn ermitteln kann, kann er auch nur den Aufschlag auf den aktuellen Lichtwert benennen - der absolute Wert ergibt sich dann aus der Summe von aktuellem Lichtwert und Aufschlag.

Darüber hinaus werden aktuelle Werte und Status übertragen, die über die HMI zur Anzeige gebracht werden können.

Datenaustausch HMI

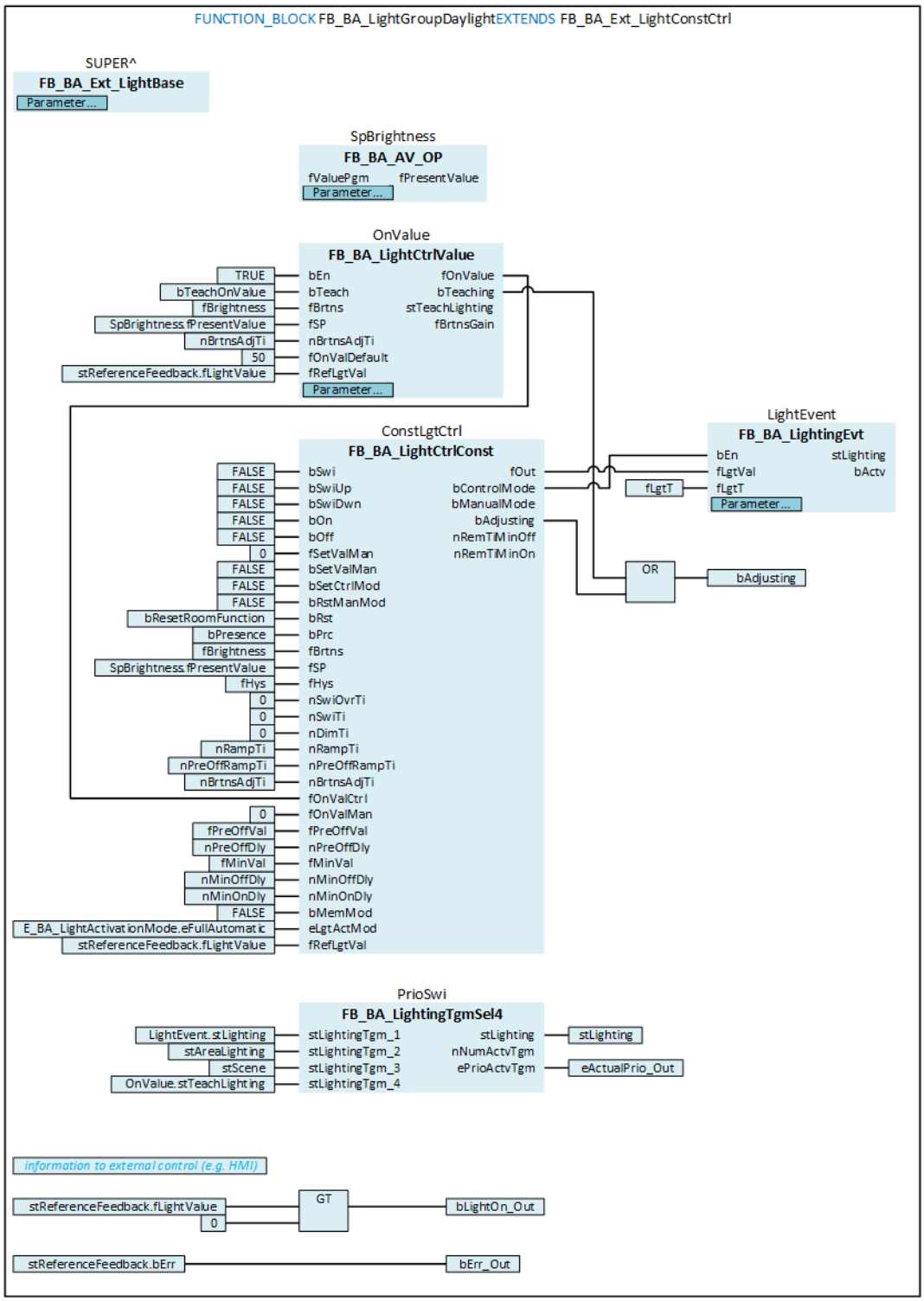
Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_LightBase* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

- **bLightOn_Out**: Ausgabeinformation "Licht Ein" an die HMI.
- **bErr_Out**: Ausgabeinformation "Referenzaktor in Störung" an die HMI.
- **eActualPrio_Out**: Aktuell steuernde Telegrammpriorität an die HMI.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightGroupDaylight EXTENDS FB_BA_Ext_LightBase
VAR_INPUT
  bResetRoomFunction     : BOOL;
  bPresence              : BOOL;
  bTeachOnValue          : BOOL;
  fBrightness            : REAL;
  stAreaLighting         : ST_BA_Lighting;
  stScene                : ST_BA_Lighting;
  stReferenceFeedback    : ST_BA_LightActuatorFeedback;
END_VAR

VAR_OUTPUT
  stLighting             : ST_BA_Lighting;
```

```
  bAdjusting              : BOOL;
END_VAR

VAR_INPUT CONSTANT PERSISTENT
  fHys                    : REAL;
  nRampTi                 : UDINT;
  nPreOffRampTi           : UDINT;
  nBrtnsAdjTi             : UDINT;
  fPreOffVal              : REAL;
  nPreOffDly              : UDINT;
  fMinVal                 : REAL;
  nMinOffDly              : UDINT;
  nMinOnDly               : UDINT;
  fLgtT                   : REAL;
END_VAR

VAR_INPUT CONSTANT
  SpBrightness            : FB_BA_AV_Op;
  OnValue                 : FB_BA_LightCtrlValue := ( ePrio := E_BA_LightingPrio.eMaintenance);
  LightEvent              : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eSimple);
END_VAR
VAR
  ConstLgtCtrl            : FB_BA_LightCtrlConst;
  PrioSwi                 : FB_BA_LightingTgmSel4;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bResetRoomFunction	BOOL	Dieser Eingang schaltet den Konstantlichtregelbaustein ab. Die Abschaltung erfolgt über eine Rampe und Verweilzeit auf einem Basislichtwert, siehe Parameter: *fPreOffVal*, *nPreOffDly* und *nPreOffRampT*.
bPresence	BOOL	Präsenzmeldeeingang. Im Template ist der Konstantlichtregelbaustein im Vollautomatikbetrieb vorkonfiguriert, so dass eine steigende Flanke an diesem Eingang die Funktion ein- und eine fallende Flanke die Funktion ausschaltet.
bTeachOnValue	BOOL	Startet den Einlernvorgang am Baustein FB_BA_LightCtrlValue [▶ 321]. Über ein Lichtsteuertelegramm der Priorität *eMaintenance* wird das Licht einmal auf 0% und einmal auf 100% geschaltet. Durch die hohe Priorität des Telegrammes (*eMaintenance*) werden Hand- und Automatiktelegramme am Prioritätenschalter *PrioSwi* übersteuert.
fBrightness	REAL	Aktuelle Helligkeit für die Konstantlichtautomatik: Bereich und Einheit richten sich nach dem Helligkeitssensor.
stAreaLighting	ST_BA_Lighting [▶ 277]	Resultierendes Telegramm aus den übergeordneten Ebenen (Bereich/area, Etage/floor und Gebäude/building), welches im Baustein FB_BA_AreaLighting [▶ 936] gebildet wird.
stScene	ST_BA_Lighting [▶ 277]	Reservierter Telegramm-Eingang für eine Szenenansteuerung.
stReferenceFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldeeingang des angesteuerten Lichtaktors bzw. der Führungsleuchte der angesteuerten Gruppe.

Ausgänge

Name	Typ	Beschreibung
stLighting	ST_BA_Lighting [▶ 277]	Resultierendes Ausgabeprogramm.
bAdjusting	BOOL	Der Konstantlichtregelbaustein befindet sich im Ausregelbetrieb oder aber die Einschaltfuntkion im Einlernbetrieb. Dieses Signal kann dazu genutzt werden, Helligkeitssensoren, welche nicht analog, sondern über eine Kommunikation arbeiten, häufiger abzufragen und damit ein günstigeres Regelverhalten zu erzielen.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fHys	REAL	Konstantlicht-Automatik: Hystereseband. Die Konstantlichtregelung steuert den Ausgang so weit hoch, bzw. herunter, bis die gemessene Helligkeit den Zielwert erreicht, bzw. leicht über- oder unterschreitet. Das Licht gilt dann als ausgeregelt. Erst wenn die gemessene Helligkeit den Sollwert um *fHys*/2 über- oder unterschreitet, wird erneut nachgeregelt. Bereich und Einheit richten sich nach dem Helligkeitssensor, siehe oben: *fBrtnsSP* und *fBrtnsSen*. In FB_Init voreingestellt auf 50.
nRampTi	UDINT	Regelrampe der Konstantlicht-Automatik in Sekunden, bezogen auf ein Dimmen von 0 bis 100%. In FB_Init voreingestellt auf 60 s.
nPreOffRampTi	UDINT	Rampe, mit vor dem Abschalten auf einen Basiswert *fPreOffVal* gefahren wird. In *FB_Init* voreingestellt auf 2 s.
nBrtnsAdjTi	UDINT	Wartezeit in Sekunden nach Einschalten des Lichts, damit der Helligkeitssensor den korrekten Wert erfasst. Wird sowohl von der Lichtvorgabefunktion *OnValue* als auch von dem Lichtsteuerbaustein *ConstLgtCtrl* genutzt.
fPreOffVal / nPreOffDly	REAL / UDINT	Basiswert und Haltezeit auf diesem Wert vor dem Ausschalten der Gesamtfunktion. Unterschreitet der aktuelle Lichtwert den Basiswert bereits, so ist diese Funktion nicht aktiv und es wird unmittelbar abgeschaltet. In *FB_Init* voreingestellt auf 4% und 10 s.
fMinVal	REAL	Konstantlicht-Automatik: Mindestausgabewert. Wird dieser Wert intern unterschritten (d.h. es ist hell genug, dass kein künstliches Licht benötigt wird), so schaltet die Konstantlichtregelung das Licht nach Ablauf von *nOffDly* in Sekunden ab. Ist das Licht ausgeschaltet und die Regelung erkennt wieder Bedarf an Licht über dem Minimumwert, so schaltet die Regelung nach *nOnDly* in Sekunden das Licht wieder auf zunächst *fMinVal* ein. In FB_Init voreingestellt auf 4%
nMinOffDly	UDINT	Konstantlicht-Automatik: Abschaltwartezeit in Sekunden, siehe *fMinVal*. In FB_Init voreingestellt auf 300s.
nMinOnDly	UDINT	Konstantlicht-Automatik: Einschaltwartezeit in Sekunden, siehe *fMinVal*. In FB_Init voreingestellt auf 300s.
fLgtT	REAL	Lichttemperatur: In *FB_Init* voreingestellt auf 3000K.

Eingänge CONSTANT

Name	Typ	Beschreibung
SpBrightness	FB_BA_AV_Op [▶ 207]	Analoges Eingabeobjekt: Sollwert der Regelung [lx].
OnValue	FB_BA_LightCtrlValue [▶ 321]	Optimierende Einschaltfuntkion
LightEvent	FB_BA_LightingEvt [▶ 316]	Telegrammerzeuger für die Lichtwerte. Die Priorität ist voreingestellt auf *eManualActuator*.

VAR

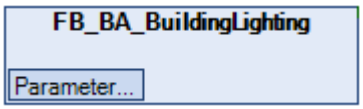
Name	Typ	Beschreibung
ConstLgtCtrl	FB_BA_LightCtrlConst [▶ 324]	Konstanlichtregelbaustein.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 317]	Telegrammauswahl.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.4.1.2 Building

6.1.4.2.1.4.4.1.2.1 FB_BA_BuildingLighting



Dieses Template stellt gebäudeübergreifende Steuertelegramme für die Lichtfunktionen zusammen.

Üblicherweise werden die Alarme "Feuer" und "Einbruch" im Gebäuderechner (Building Controller) aufgenommen (FB_BA_FireAlarmSystem und FB_BA_BurglarAlarmSystem). Von dort aus gelangen sie mit anderen Informationen über den Kommunikations-Server (FB_BA_AdsComServer_Basic [▶ 860]) in die Etagencontroller, wo sie im Gegenbaustein FB_BA_AdsComClient_Basic [▶ 861] empfangen und in der lokalen Site [▶ 1135]-Variablenliste zentral zur Verfügung gestellt werden.

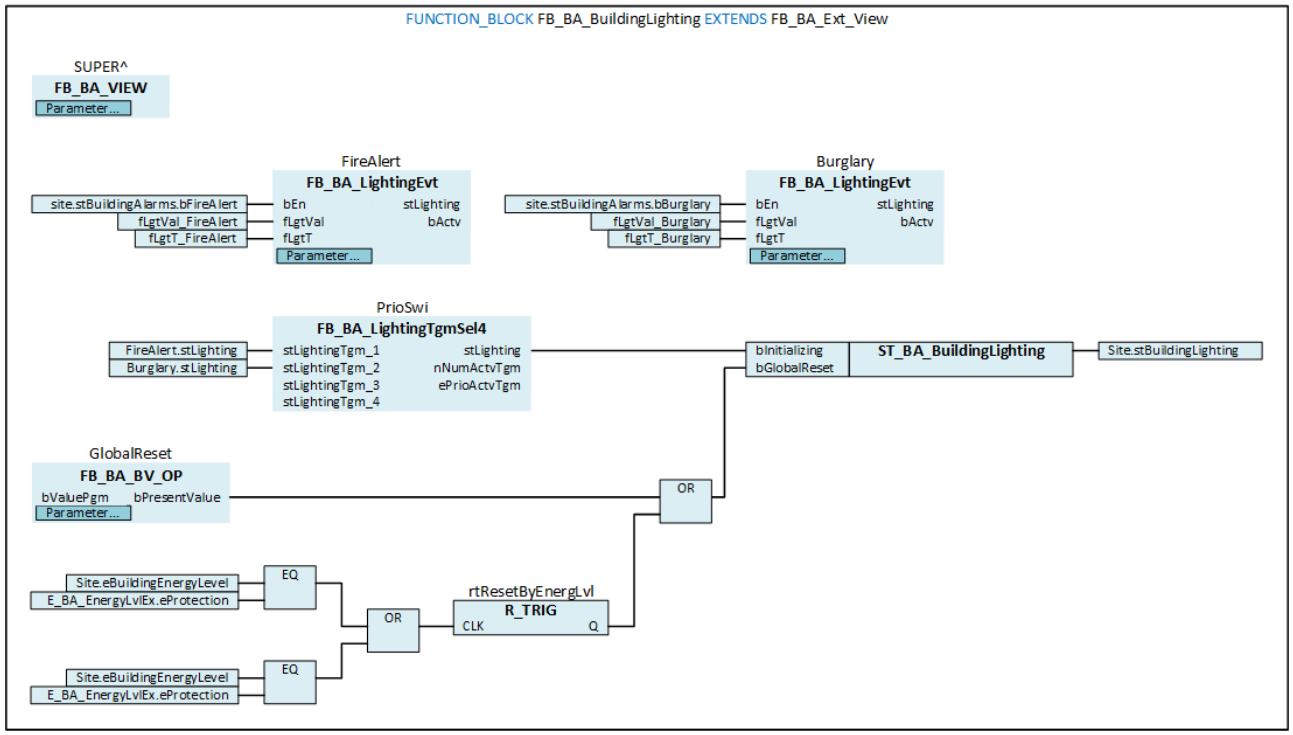
Die Alarme Feuer und Einbruch können jeweils ein eigenes Lichttelegramm über die Event-Bausteine [▶ 316] FireAlert bzw. Burglary auslösen. Die Telegramme werden an einen Prioritätenbaustein PrioSwi [▶ 317] geführt.

Neben den Alarmen wird in diesem Template ein globaler Reset der Lichtfunktionen zur Verfügung gestellt. Er besteht zum einen aus der manuellen Auslösung über ein Eingabeobjekt GlobalReset [▶ 221], welches im FB_Init als Pushbutton, also tastend parametriert ist, zum anderen wird beim Erreichen des Gebäude-Energieniveaus „Protection“ oder „Economy“ ein Triggersignal gebildet, welches "Abwesenheit" signalisieren soll.

Das resultierende Lichttelegramm vom *PrioSwi* und der globale Reset "Licht" werden über die Struktur stBuildingLighting [▶ 715] in der lokalen Site [▶ 1135]-Variablenliste zur Verfügung gestellt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_BuildingLighting EXTENDS FB_BA_View
VAR_INPUT CONSTANT PERSISTENT
  fLgtVal_FireAlert     : REAL;
  fLgtT_FireAlert       : REAL;
  fLgtVal_Burglary      : REAL;
  fLgtT_Burglary        : REAL;
END_VAR
VAR_INPUT CONSTANT
  FireAlert             : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eFire);
  Burglary              : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eBurglary);
  GlobalReset           : FB_BA_BV_Op;
END_VAR
VAR
  rtResetByEnergLvl     : R_TRIG;
  PrioSwi               : FB_BA_LightingTgmSel4;
END_VAR
```

VAR_INPUT CONSTANT PERSISTENT

Name	Typ	Beschreibung
fLgtVal_FireAlert	REAL	Eingabe Lichtwert [%] im Brandfall. In *FB_Init* voreingestellt auf 100%.
fLgtT_FireAlert	REAL	Eingabe Lichttemperatur [K] im Brandfall. In *FB_Init* voreingestellt auf 4000K.
fLgtVal_Burglary	REAL	Eingabe Lichtwert [%] im Falle eines Einbruchs. In *FB_Init* voreingestellt auf 100%.
fLgtT_Burglary	REAL	Eingabe Lichttemperatur [K] im Falle eines Einbruchs. In *FB_Init* voreingestellt auf 4000K.

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
FireAlert	FB_BA_LightingEvt [▶ 316]	Ereignis-Telegrammbaustein für den Feueralarm.
Burglary	FB_BA_LightingEvt [▶ 316]	Ereignis-Telegrammbaustein für den Einbruchalarm.
GlobalReset	FB_BA_BV_Op [▶ 221]	Binäres Eingabeobjekt: Bedienmöglichkeit die Lichtfunktionen zurückzusetzen. Dieses Objekt ist im *FB_Init* als tastend parametriert, das heißt, es erzeugt einen TRUE-Impuls.

VAR

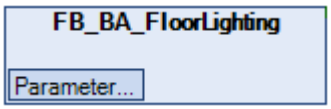
Name	Typ	Beschreibung
rtResetByEnergLvl	R_TRIG	Triggert das aus den Energieniveaus gebildete Reset-Signal.
PrioSwi	FB_BA_LightingTgmSel4 [▶ 317]	Telegrammauswahlbaustein für die Lichttelegramme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.4.1.3 Floor

6.1.4.2.1.4.4.1.3.1 FB_BA_FloorLighting



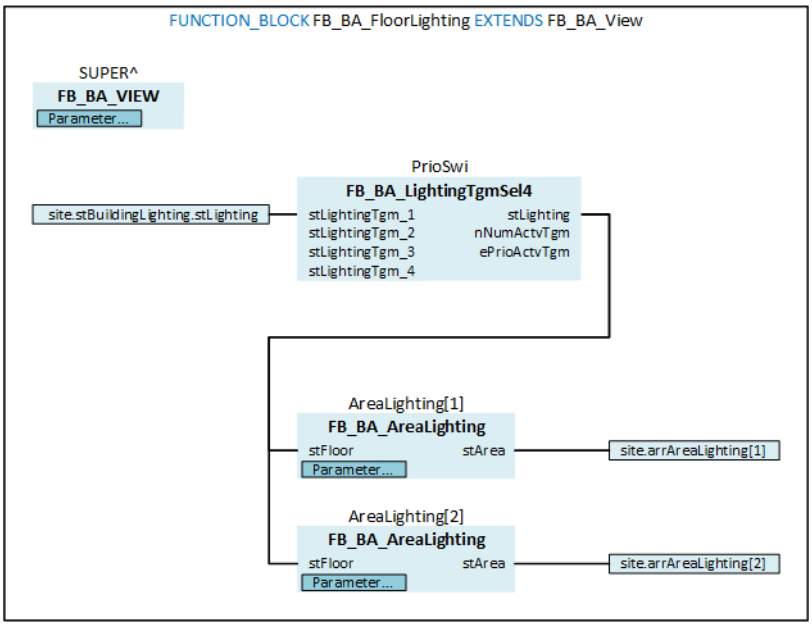
Dieses Template dient zur Implementation verschiedener Lichtfunktionen auf Etagenebene. Es ist denkbar einfach gestaltet und soll nach Bedürfnissen frei erweitert werden.

Das resultierende Gebäude-Lichttelegramm aus der lokalen Site [▶ 1135]-Variablenliste wird an den Prioritätenbaustein PrioSwi [▶ 317] geführt, welcher weitere Eingänge für Lichttelegramme etwa aus Zeitschaltplänen bereit hält. Das daraus resultierende Telegramm wird an die Eingänge der Licht-Bereichsbausteine *AreaLighting[1]* und *AreaLighting[2]* gelegt. Deren Ausgangstelegramme werden zurück auf die lokale Site [▶ 1135]-Variablenliste geführt.

Der Licht-Bereichsbaustein [▶ 936] enthält exemplarisch Zeitschaltpläne und Handbedienungen. Die dort aufgeführten Bausteine können für dieses Template herauskopiert und ergänzt bzw. reduziert werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_FloorLighting EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AreaLighting    : ARRAY[1..BA2_Param.nMaxNumberOfAreas] OF FB_BA_AreaLighting;
END_VAR
VAR
  PrioSwiLight    : FB_BA_LightingTgmSel4;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
AreaLighting	FB_BA_AreaLighting [▶ 936]	Licht-Bereichsbaustein mit Telegrammfunktionen, welche den Bereich innerhalb einer Etage betreffen.

VAR

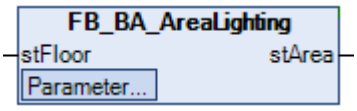
Name	Typ	Beschreibung
PrioSwiLight	FB_BA_LightingTgmSel4 [▶ 317]	Telegrammauswahlbaustein für die Lichttelegramme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.4.1.4 Area

6.1.4.2.1.4.4.1.4.1 FB_BA_AreaLighting



Auswahl des Lichtbetriebs für einen Bereich anhand eines Zeitschaltplanes (*Sched*) und eines Handeingriffs (*OpModMan*).

Der **Handeingriff** hat folgende Modi:

Wert	Bedeutung
1	Automatikbetrieb.
2	Manuelle Anwahl Nachtwächterrundgang.
3	Manuelle Anwahl Gebäudereinigung.
4	Manuelle Anwahl Wartung.
5	Manuelle Anwahl Benutzerdefiniert 1.
6	Manuelle Anwahl Benutzerdefiniert 2.

Ist **Automatikbetrieb** gewählt, so setzt der Baustein *DeMuxManMod* den Ausgang *bQ01*. Da dieser Ausgang nicht mit dem *PrioSwi* verbunden ist, ist der Handbetrieb am *PrioSwi* deaktiviert und der Zeitschaltplan *Sched* ist aktiv. Dieser kann folgende Werte annehmen:

Wert	Bedeutung
1	Default.
2	Nachtwächterrundgang.
3	Gebäudereinigung.
4	Benutzerdefiniert 1.
5	Benutzerdefiniert 2.

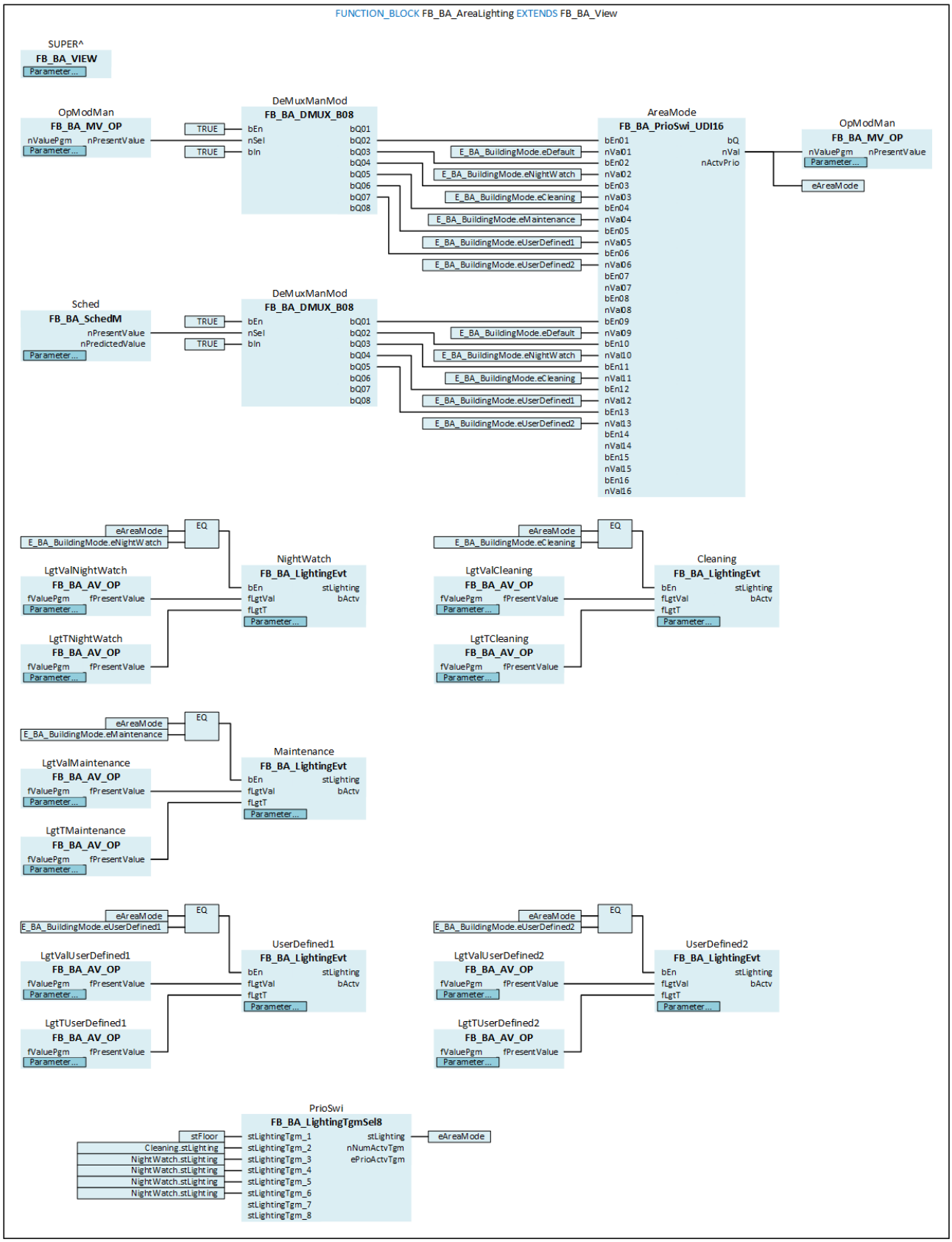
Der aktuell angewählte Lichtbetrieb *eAreaMode*, welcher vom selben Typ wie der Gebäudemodus [▶ 709] ist, wird dann über den Baustein *OpModPr* zur Anzeige gebracht.

Anhand des aktuell gültigen Lichtbetriebs wird einer der Eventbausteine [▶ 316] aktiviert. Alle möglichen Lichttelegramme werden neben dem eingangsseitigen Etagentelegramm *stFloor* an den Prioritätenbaustein *PrioSwi* geführt. Das daraus resultierende Telegramm steht am Ausgang *stArea* den Raumtemplates zur Verfügung.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AreaLighting EXTENDS FB_BA_View
VAR_INPUT
  stFloor               : ST_BA_Lighting;
END_VAR
VAR_OUTPUT
```

```
  stArea                : ST_BA_Lighting;
END_VAR
VAR_INPUT CONSTANT
  OpModMan              : FB_BA_MV_Op;
  OpModPr               : FB_BA_MV_Op;
  Sched                 : FB_BA_SchedM;

  LgtValNightWatch      : FB_BA_AV_Op;
  LgtTNightWatch        : FB_BA_AV_Op;
  NightWatch            : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eNightWatch);

  LgtValCleaning        : FB_BA_AV_Op;
  LgtTCleaning          : FB_BA_AV_Op;
  Cleaning              : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eCleaning);

  LgtValMaintenance     : FB_BA_AV_Op;
  LgtTMaintenance       : FB_BA_AV_Op;
  Maintenance           : FB_BA_LightingEvt := ( ePrio := E_BA_LightingPrio.eMaintenance);

  LgtValUserDefined1    : FB_BA_AV_Op;
  LgtTUserDefined1      : FB_BA_AV_Op;
  UserDefined1          : FB_BA_LightingEvt := ( ePrio := 10);

  LgtValUserDefined2    : FB_BA_AV_Op;
  LgtTUserDefined2      : FB_BA_AV_Op;
  UserDefined2          : FB_BA_LightingEvt := ( ePrio := 10);
END_VAR
VAR
  DeMuxManMod           : FB_BA_DMUX_B08;
  DeMuxSched            : FB_BA_DMUX_B08;
  AreaMode              : FB_BA_PrioSwi_UDI16;
  eAreaMode             : E_BA_BuildingMode;
  PrioSwi               : FB_BA_LightingTgmSel8;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stFloor	ST_BA_Lighting [▶ 277]	Lichtsteuertelegramm aus der übergeordneten Ebene "Etage".

Ausgänge

Name	Typ	Beschreibung
stArea	ST_BA_Lighting [▶ 277]	Resultierendes Lichtsteuertelegramm aus diesem Template, welches die Ebene "Bereich" repräsentiert.

Eingänge CONSTANT

Name	Typ	Beschreibung
OpModMan	FB_BA_MV_Op [▶ 244]	Eingabeobjekt Hand für den Gebäudemodus (BuildingMode).
OpModPr	FB_BA_MV_Op [▶ 244]	Resultierender Modus
Sched	FB_BA_SchedM [▶ 230]	Zeitschaltobjekt (Automatik) für den Gebäudemodus (BuildingMode).
LgtValNightWatch	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für den Lichtwert Nachtwächterrundgang.
LgtTNightWatch	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für die Lichttemperatur Nachtwächterrundgang.
NightWatch	FB_BA_LightingEvt [▶ 316]	Eventbaustein Nachtwächterrundgang.
LgtValCleaning	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für den Lichtwert Gebäudereinigung.
LgtTCleaning	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für die Lichttemperatur Gebäudereinigung.
Cleaning	FB_BA_LightingEvt [▶ 316]	Eventbaustein Gebäudereinigung.
LgtValMaintenance	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für den Lichtwert Wartung.
LgtTMaintenance	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für die Lichttemperatur Wartung.
Maintenance	FB_BA_LightingEvt [▶ 316]	Eventbaustein Wartung.
LgtValUserDefined1	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für den Lichtwert "Benutzerdefiniert1".
LgtTUserDefined1	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für die Lichttemperatur "Benutzerdefiniert1".
UserDefined1	FB_BA_LightingEvt [▶ 316]	Eventbaustein "Benutzerdefiniert1".
LgtValUserDefined2	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für den Lichtwert "Benutzerdefiniert2".
LgtTUserDefined2	FB_BA_AV_Op [▶ 207]	Eingabeobjekt für die Lichttemperatur "Benutzerdefiniert2".
UserDefined2	FB_BA_LightingEvt [▶ 316]	Eventbaustein "Benutzerdefiniert2".

VAR

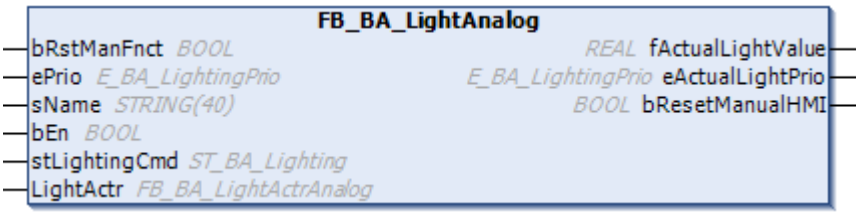
Name	Typ	Beschreibung
DeMuxManMod	FB_BA_DMUX_B08 [▶ 436]	Konvertierung des Multistate-Wertes Handanwahl zu einem Binärausgang.
DeMuxSched	FB_BA_DMUX_B08 [▶ 436]	Konvertierung des Multistate-Wertes Zeitschaltplan zu einem Binärausgang.
AreaMode	FB_BA_PrioSwi_UDI16 [▶ 443]	Priorisierende Rückumwandlung der Status zu einem resultierenden Multistate - bzw. Enumerationswert für den Lichtbetrieb im Gebäudebereich.
eAreaMode	E_BA_BuildingMode [▶ 709]	Aktuell gültiger Lichtbetrieb im Gebäudebereich.
PrioSwi	FB_BA_LightingTgmSel8 [▶ 317]	Telegrammauswahlbaustein für die Lichttelegramme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.4.2 Actuators

6.1.4.2.1.4.4.2.1 FB_BA_LightAnalog



Dieses Template dient zur Ansteuerung eines analogen Lichtaktors.

Kernstück ist der Analogausgabebaustein FB_BA_AO_Raw [▶ 204], der zum einen den Lichtwert im BACnet zur Verfügung stellt und zum anderen eine Umrechnung des Lichtwertes von 0 bis 100 % auf 0 bis 32767 realisiert. Die allokierte Variable *nLgtVal* ermöglicht eine direkte Verknüpfung zu einer Dimmerklemme (KL2751, KL2761).

Der Baustein lässt bei der Ansteuerung des Lichtaktors die Farbtemperatur außer Acht.

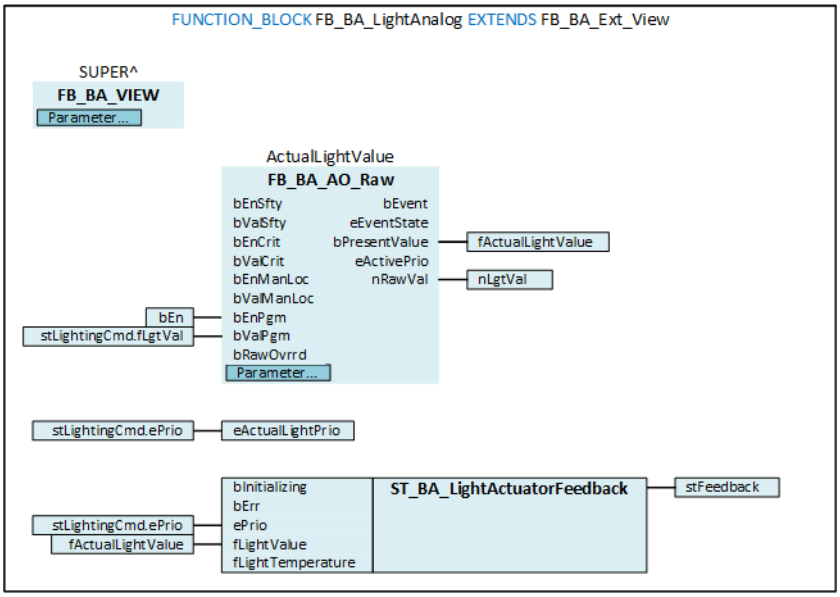
Das Template besitzt eine Rückmeldestruktur stFeedback [▶ 715].

Ist der programmierte Lichtaktor ein Einzelaktor oder der Referenzaktor einer Gruppe, so ist dieses Rückmeldetelegramm an die ansteuernde Raum- (Zonen-) Anwendungsfunktion zu verknüpfen, um auf diesem Wege Informationen über den Zustand des Aktors zu erlangen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightAnalog  EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  stLightingCmd         : ST_BA_Lighting;
END_VAR
VAR_OUTPUT
  fActualLightValue     : REAL;
  eActualLightPrio      : BYTE;
  stFeedback            : ST_BA_LightActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT
```

```
  ActualLightValue     : FB_BA_AO_Raw;
END_VAR
VAR
  nLgtVal      AT %Q*  : INT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteines.
stLightingCmd	ST_BA_Lighting [▶ 277]	Resultierendes Telegramm aus der übergeordneten Zone (Raum).

Ausgänge

Name	Typ	Beschreibung
fActualLightValue	REAL	Aktueller Lichtwert in Prozent.
eActualLightPrio	BYTE	Aktuelle Priorität des Befehlstelegramms *stLightingCmd*.
stFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldetelegramm zur Verknüpfung an die ansteuernde Raum- (Zonen-) Anwendungsfunktion. Auf diesem Wege gelangen Informationen über den Zustand des Lichtaktors zurück in die Anwendungsfunktion.

Eingänge CONSTANT

Name	Typ	Beschreibung
ActualLightValue	FB_BA_AO_Raw [▶ 204]	Baustein zur Umrechnung des prozentualen Lichtwertes auf den Wertebereich 0..32767 und zur gleichzeitigen Darstellung im BACnet.

VAR

Name	Typ	Beschreibung
nLgtVal	INT	Ausgangsvariable zur Verknüpfung mit der Dimmerklemme.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.4.2.2 FB_BA_LightDALI_Base



Dieses Template dient der Ansteuerung eines DALI-Lichtaktors, einer DALI-Gruppe oder DALI-Geräten mit Sammelbefehl (broadcast), wobei nur der Lichthelligkeitswert verändert wird.

Kernstück ist der Lichtsteuerbaustein LightActr [▶ 887]. Dieser stellt die Verbindung zu einer DALI-Leuchte (bzw. -Gruppe) her und wandelt Änderungen im Lichttelegramm *stLightingCmd* in DALI-Steuersignale um. Darüber hinaus kann über den Eingang *bInitialize* die DALI-Leuchte (bzw. -Gruppe) mit den am Baustein eingestellten Parametern (Minimum-Wert, Maximum-Wert etc.) initialisiert werden.

Zu beachten ist, dass bei einer Änderung des Licht-Stellwertes im Befehlstelegramm die Ausgabewerte *byActlLgtVal* und *fActlLgtVal* zunächst diesem Stellwert entsprechen. Intern, im Baustein FB_BA_LightActrDALI_Base [▶ 887], wird jedoch der tatsächliche Lichtwert zyklisch abgefragt. Damit ist Darstellung des wahren Zustands der DALI-Leuchte gewährleistet, ohne den DALI-Bus unnötig stark zu belasten.

Der Baustein lässt bei der Ansteuerung des Lichtaktors die Farbtemperatur außer Acht.

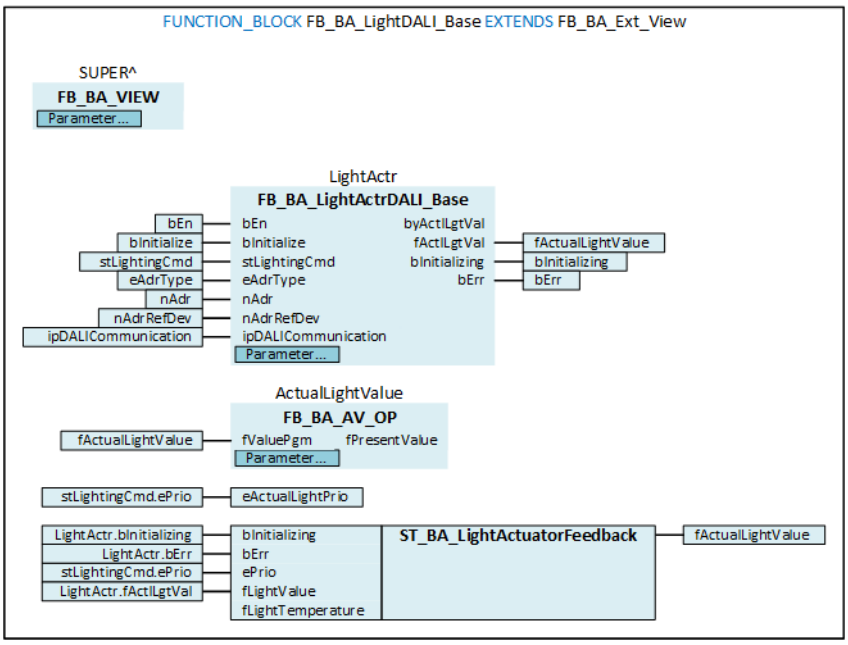
Über ein Analogobjekt FB_BA_AV_Op [▶ 207] wird der aktuelle Lichtwert im BACnet zur Verfügung gestellt.

Das Template besitzt eine Rückmeldestruktur stFeedback [▶ 715].

Ist der programmierte Lichtaktor ein Einzelaktor oder der Referenzaktor einer Gruppe, so ist dieses Rückmeldetelegramm an die ansteuernde Raum- (Zonen-) Anwenderfunktion zu verknüpfen, um auf diesem Wege Informationen über den Zustand des Aktors zu erlangen.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightDALI_Base EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  stLightingCmd         : ST_BA_Lighting;
  bInitialize           : BOOL;
END_VAR
VAR_OUTPUT
  bInitializing         : BOOL;
  bErr                  : BOOL;
  fActualLightValue     : REAL;
  eActualLightPrio      : BYTE;
  stFeedback            : ST_BA_LightActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  eAdrType              : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                  : BYTE;
  nAdrRefDev            : BYTE;
```

```
  ipDALICommunication    : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_INPUT CONSTANT
  LightActr              : FB_BA_LightActrDALI_Base;
  ActualLightValue       : FB_BA_AV_Op;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteins: Ein TRUE-Signal aktiviert die Funktion.
stLightingCmd	ST_BA_Lighting [▶ 277]	Befehlstelegramm aus der übergeordneten Zone (Raum).
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in die Vorschaltgeräte überträgt, (siehe FB_BA_LightActrDALI_Base [▶ 887]).

Ausgänge

Name	Typ	Beschreibung
bInitializing	BOOL	Der Lichtaktor bzw. die Aktoren befinden sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in die angesteuerten Vorschaltgeräte übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.
fActualLightValue	REAL	Aktueller Lichtwert in Prozent.
eActualLightPrio	E_BA_LightingPrio [▶ 273]	Aktuelle Priorität des Befehlstelegramms *stLightingCmd*.
stFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldetelegramm zur Verknüpfung an die ansteuernde Raum- (Zonen-) Anwendungsfunktion. Auf diesem Wege gelangen Informationen über den Zustand des Lichtaktors zurück in die Anwendungsfunktion.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eAdrType	Tc3_DALI.E_DALIAddressType	Anwahl der Steuerungsart: Einzel- Gruppen oder Broadcast-Steuerung, voreingestellt auf Einzelansteuerung.
nAdr	BYTE	DALI-Adresse des Aktors, bzw. der Gruppe.
nAdrRefDev	BYTE	DALI-Adresse eines Referenzgerätes, sollte Gruppen- oder Broadcast-Steuerung gewählt sein. Dieses Referenzgerät repräsentiert dann alle anderen angesteuerten Geräte.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

Eingänge CONSTANT

Name	Typ	Beschreibung
LightActr	FB_BA_LightActrDALI_Base [▶ 887]	Lichtsteuerbaustein ohne Ansteuerung der Farbtemperatur.
ActualLightValue	FB_BA_AV_Op [▶ 207]	Baustein zur Darstellung des prozentualen Lichtwertes im BACnet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.4.2.3 FB_BA_LightDALI_TC



Dieses Template dient zur Ansteuerung eines DALI-Lichtaktors, einer DALI-Gruppe oder DALI-Geräten mit Sammelbefehl (broadcast), wobei die Farbtemperatursteuerung (Temperature-Control) unterstützt wird.

Kernstück ist der Lichtsteuerbaustein LightActr [▶ 890]. Dieser stellt die Verbindung zu einer DALI-Leuchte (bzw. -Gruppe) her und wandelt Änderungen im Lichttelegramm *stLightingCmd* in DALI-Steuersignale um. Darüber hinaus kann über den Eingang *bInitialize* die DALI-Leuchte (bzw. -Gruppe) mit den am Baustein eingestellten Parametern (Minimum-Wert, Maximum-Wert etc.) initialisiert werden.

Zu beachten ist, dass bei einer Änderung des Licht-Stellwertes oder der Farbtemperatur im Befehlstelegramm die Ausgabewerte *byActlLgtVal*, *fActlLgtVal* und *fActlLgtT* zunächst diesen Stellwerten entsprechen. Intern, im Baustein FB_BA_LightActrDALI_TC [▶ 890], wird jedoch der tatsächliche Lichtwert und die tatsächliche Farbtemperatur jeweils zyklisch abgefragt. Damit ist Darstellung des wahren Zustands der DALI-Leuchte gewährleistet, ohne den DALI-Bus unnötig stark zu belasten.

Über ein Analogobjekt FB_BA_AV_Op [▶ 207] wird der aktuelle Lichtwert im BACnet zur Verfügung gestellt, während die aktuelle Farbtemperatur nur als Template-Ausgang dargestellt ist.

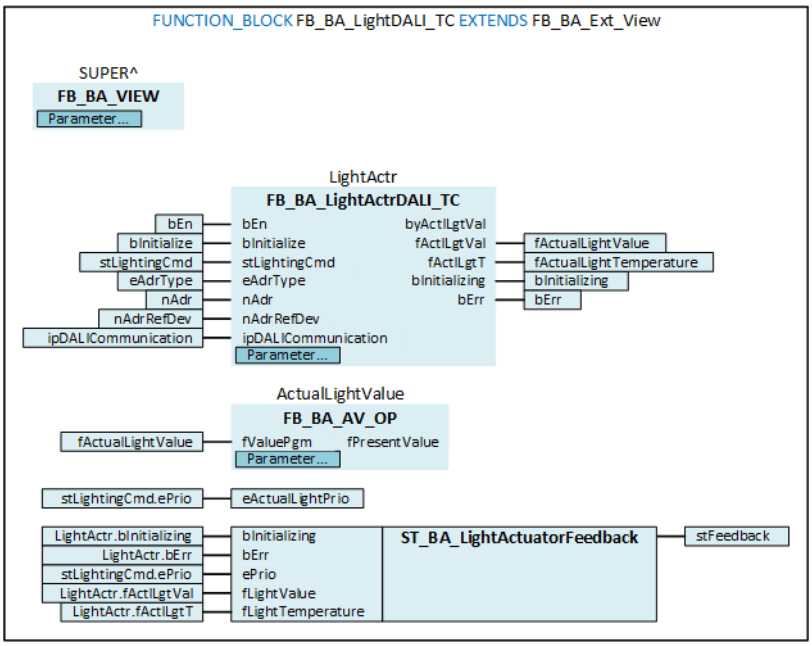
Das Template besitzt eine Rückmeldestruktur stFeedback [▶ 715].

Ist der programmierte Lichtaktor ein Einzelaktor oder der Referenzaktor einer Gruppe, so ist dieses Rückmeldetelegramm an die ansteuernde Raum- (Zonen-) Anwenderfunktion zu verknüpfen, um auf diesem Wege Informationen über den Zustand des Aktors zu erlangen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_LightDALI_TC EXTENDS FB_BA_View
VAR_INPUT
  bEn                          : BOOL;
  stLightingCmd                : ST_BA_Lighting;
  bInitialize                  : BOOL;
END_VAR
VAR_OUTPUT
  bInitializing                : BOOL;
  bErr                         : BOOL;
  fActualLightValue            : REAL;
  fActualLightTemperature      : REAL;
  eActualLightPrio             : BYTE;
  stFeedback                   : ST_BA_LightActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  eAdrType                     : Tc3_DALI.E_DALIAddressType := Tc3_DALI.E_DALIAddressType.Short;
  nAdr                         : BYTE;
  nAdrRefDev                   : BYTE;
  ipDALICommunication          : Tc3_DALI.I_DALICommunication;
END_VAR
VAR_INPUT CONSTANT
  LightActr                    : FB_BA_LightActrDALI_Base;
  ActualLightValue             : FB_BA_AV_Op;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Bausteins: Ein TRUE-Signal aktiviert die Funktion.
stLightingCmd	ST_BA_Lighting [▶ 277]	Befehlstelegramm aus der übergeordneten Zone (Raum).
bInitialize	BOOL	Eine positive Flanke an diesem Eingang startet die DALI-Initialisierungsroutine, welche die eingestellten Parameter in die Vorschaltgeräte überträgt, (siehe FB_BA_LightActrDALI_Base [▶ 887]).

Ausgänge

Name	Typ	Beschreibung
bInitializing	BOOL	Der Lichtaktor bzw. die Aktoren befinden sich in der DALI-Initialisierungsphase, d.h. die eingetragenen Parameter werden in die angesteuerten Vorschaltgeräte übertragen.
bErr	BOOL	Fehlerausgang. Ein Klartext wird in TwinCAT im Ausgabefenster in der Fehlerliste (Error List) ausgegeben.
fActualLightValue	REAL	Aktueller Lichtwert in Prozent.
fActualLightTemperature	REAL	Aktuelle Farbtemperatur in Kelvin.
eActualLightPrio	E_BA_LightingPrio [▶ 273]	Aktuelle Priorität des Befehlstelegramms *stLightingCmd*.
stFeedback	ST_BA_LightActuatorFeedback [▶ 715]	Rückmeldetelegramm zur Verknüpfung an die ansteuernde Raum- (Zonen-) Anwenderfunktion. Auf diesem Wege gelangen Informationen über den Zustand des Lichtaktors zurück in die Anwendungsfunktion.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eAdrType	Tc3_DALI.E_DALIAddressType	Anwahl der Steuerungsart: Einzel- Gruppen oder Broadcast-Steuerung, voreingestellt auf Einzelansteuerung.
nAdr	BYTE	DALI-Adresse des Aktors, bzw. der Gruppe.
nAdrRefDev	BYTE	DALI-Adresse eines Referenzgerätes, sollte Gruppen- oder Broadcast-Steuerung gewählt sein. Dieses Referenzgerät repräsentiert dann alle anderen angesteuerten Geräte.
ipDALICommunication	Tc3_DALI.I_DALICommunication	Interface-Pointer auf den DALI-Kommunikationsbaustein.

Eingänge CONSTANT

Name	Typ	Beschreibung
LightActr	FB_BA_LightActrDALI_Base [▶ 887]	Lichtsteuerbaustein ohne Ansteuerung der Farbtemperatur.
ActualLightValue	FB_BA_AV_Op [▶ 207]	Baustein zur Darstellung des prozentualen Lichtwertes im BACnet.

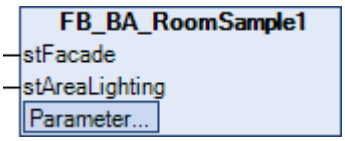
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.5 RoomSamples

Mögliche Raumvorlagen.

6.1.4.2.1.4.5.1 FB_BA_RoomSample1



Dieses Template stellt einen einfachen Raum mit einem Heiz-Kühlregler, einer Sonnenschutz-Zonenfunktion mit zwei Jalousien und einer Sonnenschutz-Zonenfunktion mit zwei DALI-Leuchten dar.

Gesteuert und überwacht wird der Raum zusätzlich mit einem DALI-Präsenzmelder, welcher außerdem die Raumhelligkeit zur Verfügung stellt.

DALI-Ansteuerung

Da dieses Raumtemplate ein Beispiel mit DALI-Komponenten darstellt, erfordert es eine schnelle Task mit einem DALI-Kommunikationsbaustein.

Der Datenaustausch mit den DALI-Geräten erfolgt dann über ein Interface und wird im FB_Init vordefiniert, hier am Beispiel des Helligkeits- und Präsenzsensors:

```
BrtnsPrcSns.iParent               := THIS^;
BrtnsPrcSns.iLabel                := LblCtl_BrightnessPresenceSensor;
BrtnsPrcSns.eDPADMode             := E_BA_DPADMode.eInclude;
BrtnsPrcSns.nAdr                  := 11;
BrtnsPrcSns.nInstancePresence     := 1;
BrtnsPrcSns.nInstanceBrightness   := 0;
BrtnsPrcSns.nResBrtns             := 10;
BrtnsPrcSns.ipDALICommunication   := DALICommunication.fbKL6821Communication;
```

In der letzten Zeile wird der DALI-Kommunikationsbaustein dem Kommunikations-Interfacepointer des Helligkeits- und Präsenzsensors zugewiesen.

Dieses Beispiel implementiert im Programm *DALICommunication* den Funktionsbaustein *fbKL6821Communication*:

```
PROGRAM DALICommunication
VAR

    stKL6821InData          AT %I*  : Tc3_DALI.ST_KL6821InData;
    stKL6821OutData         AT %Q*  : Tc3_DALI.ST_KL6821OutData;

    fbKL6821Communication           : Tc3_DALI.FB_KL6821Communication;

    bError                          : BOOL;
    bResetMaximumDemandCounter      : BOOL;
    bResetOverflowCounter           : BOOL;
    nOptions                        : DWORD;
```

```

fbKL6821Communication(  bError                      => bError,
                        bResetMaximumDemandCounter  := bResetMaximumDemandCounter,
                        bResetOverflowCounter       := bResetOverflowCounter,
                        nOptions                    := nOptions,
                        bBusy                       => bBusy,
                        nBufferDemandMeter          => nBufferDemandMeter,
                        nBufferMaximumDemandMeter   => nBufferMaximumDemandMeter,
                        nBufferOverflowCounter      => nBufferOverflowCounter,
                        bResetInactiveProcessImage  := bResetInactiveProcessImage,
                        bInitialise                 := bInitialise,
                        bDigitalInput1Active        => bDigitalInput1Active,
                        bDigitalInput2Active        => bDigitalInput2Active,
                        bProcessImageInactive       => bProcessImageInactive,
                        bCollisionError             => bCollisionError,
                        bPowerSupplyError           => bPowerSupplyError,
                        bShortCircuit               => bShortCircuit,
                        bInitialising               => bInitialising,
                        nTerminalDescription        => nTerminalDescription,
                        nFirmwareVersion            => nFirmwareVersion,
                        stInData                    := stKL6821InData,
                        stOutData                   := stKL6821OutData);
```

Das Programm *DALICommunication* ist in einer schnellen Task aufzurufen - eine Anleitung dazu findet sich im Kapitel: Schnelle Task für serielle Kommunikation [▶ 55].

Helligkeits- und Präsenzmelder

Der Baustein FB_BA_BrightnessPresenceDALI [▶ 898] repräsentiert DALI-Geräte, die gleichermaßen Helligkeit und Präsenz auslesen können. In der Anleitung des Bausteins ist beschrieben, dass die Präsenz ereignisgesteuert ausgegeben, die Helligkeit jedoch in einem Intervall der einstellbaren Dauer *nPrdQueryBrtns* [s] ausgelesen wird.

In diesem Beispiel werden eine kurze Dauer *nPollingIntervalFast* und eine lange *nPollingIntervalNormal* vorgegeben: die kurze ist für die Helligkeitsadaption einer Konstantlichtregelung gedacht, sollte diese einen Nachregelauftrag erhalten (*LightZoneConstant.bAdjusting*), die lange Dauer dann im Normalbetrieb, um den DALI-Bus nicht weiter zu belasten.

Heiz- und Kühlregler

Kern der Heiz- und Kühlfunktion ist der Baustein FB_BA_HeatCool_PID [▶ 905].

Die globale Variablenliste Site [▶ 1135] teilt dem Baustein mit, ob Heiz- oder Kühlbetrieb ansteht (*eHeatCoolMedium*), welche Temperaturen für welches Energieniveau gelten (*stBuildingSpRmT*) und welches Energieniveau gebäudeweit gerade gültig ist (*eBuildingEnergyLevel*).

Generell soll nur bei Präsenz die volle Heiz- und Kühlleistung im Energieniveau *Comfort* zur Verfügung stehen. Dies wird dem Baustein über den Präsenzmelder des Raumes am Eingang *bPrc* gemeldet.

Zwei Fensterkontakte *WdwCon1* und *WdwCon2* überwachen das Heizen/Kühlen. Diese Kontakte sind so vorgesehen, dass sie bei geschlossenem Fenster ein TRUE-Signal liefern. Wird eines der Fenster geöffnet, erhält der Baustein FB_BA_HeatCool_PID [▶ 905] am Eingang *bWdwCon* ein TRUE-Signal und schaltet intern auf das Energieniveau *Protection* um nicht unnötig gegen die Außentemperatur zu steuern.

Der Taupunktwächter, *DewPointSensor*, ist wie die Fensterkontakte nach dem Ruhestromprinzip ausgelegt. Im kritischen Fall wird er ein FALSE-Signal ausgeben, welches negiert als TRUE am Eingang *bDewPnt* des FB_BA_HeatCool_PID [▶ 905] gelangt. Damit wird der Kühlmodus sofort beendet und verriegelt.

Ein analoger Sensor *RmTSen* stellt der Regelung die Raumtemperatur zur Verfügung, über einen Sollwertsteller *RmTAdj* kann der Raumtemperatur-Sollwert in geringen Grenzen verändert werden. Die im FB_Init voreingestellten Werte für *fRmTAdjMin* und *fRmTAdjMax* liegen bei -5K und 5K.



Die Raumtemperaturanpassung ist nur dann wirksam, wenn das Energieniveau *Comfort* oder *Precomfort* anliegt, siehe hierzu FB_BA_RmTAdj [▶ 309].

Jalousieansteuerung

Für die Jalousiesteuerung ist in diesem Beispiel ein Zonenbaustein FB_BA_SunblindZone vordefiniert, welcher parallel 2 Jalousie Aktor Bausteine FB_BA_SunBld [▶ 988] ansteuert.

Dabei bündelt der Zonenbaustein die gesamte Generierung der Automatikbefehle, der Handansteuerung sowie die Verknüpfung zu einer Visualisierung (HMI).

Die Jalousie AktorBausteine beinhalten lediglich die Fahrprofile und die Verknüpfung zur Hardware. Jede Jalousie oder Jalousiegruppe, welche einzeln angesteuert werden soll, benötigt daher einen Zonenbaustein.

Der Zonenbaustein erhält über den Eingang *stFacade* Daten wie Sturm- und Vereisungsschutz oder aber Sonnenschutz und Thermoautomatik. Er ist damit einer bestimmten Fassade zugeordnet. Würde es sich um einen Eckraum mit Jalousien unterschiedlicher Ausrichtung handeln, müsste ein weiterer Zonenbaustein hinzugefügt werden.

Über den Präsenzmeldeeingang *bPrcDetc*, an dem der Helligkeits- und Präsenzmelder verknüpft ist, entscheidet der Baustein, ob der Sonnenschutz aktiv sein kann (Präsenz) oder aber die Thermoautomatik (Nicht-Präsenz).

Die Thermoautomatik braucht darüber hinaus Informationen über die Raumtemperatur, sowie den aktuell gültigen Gebäudesollwert, um mithilfe der Jalousien die Strahlungswärme der Sonne zu nutzen oder davor zu schützen. Sonnenschutz und Thermoautomatik sind innerhalb des FB_BA_SunblindZone zusätzlich generell abwählbar.

Die Tasteingänge *bSunbldSwiUp* und *bSunbldSwiDwn* sind hier als einfache Variablen deklariert und es obliegt dem Anwender diese mit seiner Applikation zu verknüpfen.

Zur Zustandsbeurteilung der Jalousiegruppe und zur Anzeige in einer Visualisierung ist ein Jalousie Aktor als Referenz zu wählen. Sein Ausgang *stFeedback* muss mit dem Eingang *stReferenceFeedback* des Zonenbausteins verbunden werden. Bei einer Einzeljalousie ist diese selbst der Referenz Aktor.

Lichtansteuerung

Ähnlich wie bei den Jalousien ist auch beim Licht das Kernstück ein Zonenbaustein, der die Generierung der übergeordneten Befehle (Gebäude, Etage, Bereich) mit der lokalen Handbedienung und der Ansteuerung und Anzeige einer möglichen Visualisierung zusammenfasst. Die Aktor Bausteine stellen lediglich die Ansteuerung der Leuchtmittel dar, welche direkt über eine Busklemme oder über ein Protokoll (DALI) sein kann.

Auch hier benötigt jeder einzeln anzusteuernde Aktor einen Zonenbaustein. Ebenfalls ist zur Zustandsbeurteilung einer Gruppe ein Referenz Aktor zu wählen, dessen Ausgang *stFeedback* mit dem Eingang *stReferenceFeedback* des Zonenbausteins verknüpft werden muss. Bei einer Einzelleuchte ist diese selbst der Referenz Aktor.

In diesem Beispiel ist die Lichtzone eine Konstantlichtregelung FB_BA_LightGroupConstant [▶ 908].

Der Helligkeits- und Präsenzmeldebaustein FB_BA_BrightnessPresenceDALI [▶ 898] kann, sofern Vollautomatik gewählt ist, über den Präsenzmeldeeingang *bPrcDetc* die Konstantlichtregelung aktivieren. Wird keine Präsenz mehr erkannt, wird über den Eingang *bResetRoomFunction* die Lichtsteuerung zurückgesetzt. Die Automatikfunktion, bzw. deren lokale Handübersteuerung, ist dann nicht mehr aktiv und Lichttelegramme niedrigerer Priorität aus Gebäude, Etage oder Bereich, welche über den Eingang *stAreaLighting* einfließen, können die Lichtsteuerung übernehmen.

Der Eingang *fBrightness* empfängt die gemessene Lichtstärke des Helligkeitssensors zur Lichtregelung. Der Tasteingang *bLightSwi* ist hier als einfache Variable deklariert und es obliegt dem Anwender diese mit seiner Applikation zu verknüpfen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild

Bei Bedarf kann das https://infosys.beckhoff.com/content/1031/TF8040_TC3_BuildingAutomation/Resources/18447665419.zip heruntergeladen werden.

Syntax

```
FUNCTION_BLOCK FB_BA_RoomSample1 EXTENDS FB_BA_View
VAR_INPUT
  stFacade                : ST_BA_Facade;
  stAreaLighting          : ST_BA_Lighting;
END_VAR

VAR_INPUT CONSTANT PERSISTENT
  fRmTAdjMin              : REAL;
  fRmTAdjMax              : REAL;
  fLgtT                   : REAL;
  nPollingIntervalFast    : UDINT;
  nPollingIntervalNormal  : UDINT;
END_VAR

VAR_INPUT CONSTANT
  BrtnsPrcSns             : FB_BA_BrightnessPresenceDALI;
  Ctrl                    : FB_BA_HeatCool_PID;
  VlvHtg                  : FB_BA_ActuatorAnalog;
  VlvCol                  : FB_BA_ActuatorAnalog;
  WdwCon1                 : FB_BA_SensorBinary;
  WdwCon2                 : FB_BA_SensorBinary;
  DewPointSensor          : FB_BA_SensorBinary;
  RmTSen                  : FB_BA_SensorAnalog;
  RmTAdj                  : FB_BA_SensorAnalog;
  SunBldZone              : FB_BA_SunblindZone;
```

```
  SunBld1                 : FB_BA_SunBld;
  SunBld2                 : FB_BA_SunBld;
  LightZoneConstant       : FB_BA_LightGroupConstant;
  Light1                  : FB_BA_LightDALI_TC;
  Light2                  : FB_BA_LightDALI_TC;
END_VAR

VAR
  bSunbldSwiUp            : BOOL;
  bSunbldSwiDwn           : BOOL;
  bLightSwi               : BOOL;
  RmTAvg                  : FB_BA_EnAvrg02;
  bInitializeSensor       : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stFacade	ST_BA_Facade [▶ 717]	Fassadenspezifische Jalousiedaten und -telegramme.
stAreaLighting	ST_BA_Lighting [▶ 277]	Resultierendes Lichttelegramm aus dem Gebäude, der Etage und letztendlich dem Bereich (Priorisierung).

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fRmTAdjMin	REAL	Mindestwert der Temperatursollwertverschiebung, vordefiniert auf -5K.
fRmTAdjMax	REAL	Mindestwert der Temperatursollwertverschiebung, vordefiniert auf 5K.
fLgtT	REAL	Feste Lichttemperatur der Konstantlichtregelung, vordefiniert auf 3000K.
nPollingIntervalFast	UDINT	Schnelles Abrufintervall des Helligkeitssensors, vordefiniert auf 1s.
nPollingIntervalNormal	UDINT	Langsames Abrufintervall des Helligkeitssensors, vordefiniert auf 15 s.

Eingänge CONSTANT

Name	Typ	Beschreibung
BrtnsPrcSns	FB_BA_BrightnessPresenceDALI [▶ 898]	Lese- und Auswertebaustein für einen DALI-Helligkeits- und Präsenzmelder.
Ctrl	FB_BA_HeatCool_PID [▶ 905]	Heiz-Kühl-Regelbaustein.
VlvHtg	FB_BA_ActuatorAnalog [▶ 1000]	Analoges Ausgangsobjekt Heizventil.
VlvCol	FB_BA_ActuatorAnalog [▶ 1000]	Analoges Ausgangsobjekt Kühlventil.
WdwCon1	FB_BA_SensorBinary [▶ 1107]	Binäres Eingangsobjekt Fensterkontakt.
WdwCon2	FB_BA_SensorBinary [▶ 1107]	Binäres Eingangsobjekt Fensterkontakt.
DewPointSensor	FB_BA_SensorBinary [▶ 1107]	Binäres Eingangsobjekt Taupunktwächter.
RmTSen	FB_BA_SensorAnalog [▶ 1106]	Analoges Eingangsobjekt Raumtemperatursensor.
RmTAdj	FB_BA_SensorAnalog [▶ 1106]	Analoges Eingangsobjekt Sollwertanpassung.
SunBldZone	FB_BA_SunblindZone [▶ 980]	Baustein Jalousiespezifischer Zonenfunktionen.
SunBld1	FB_BA_SunBld [▶ 988]	Aktor Baustein für eine Jalousie.
SunBld2	FB_BA_SunBld [▶ 988]	Aktor Baustein für eine Jalousie.
LightZoneConstant	FB_BA_LightGroupConstant [▶ 908]	Licht-Zonenbaustein einer Konstantlichtregelung.
Light1	FB_BA_LightDALI_TC [▶ 945]	Aktor Baustein für eine DALI-Leuchte.
Light2	FB_BA_LightDALI_TC [▶ 945]	Aktor Baustein für eine DALI-Leuchte.

VAR

Name	Typ	Beschreibung
bSunbldSwiUp	BOOL	Variable zur Tasterbelegung "Jalousie manuell auf".
bSunbldSwiDwn	BOOL	Variable zur Tasterbelegung "Jalousie manuell ab".
bLightSwi	BOOL	Variable zur Tasterbelegung "Licht manuell schalten".
bInitializeSensor	BOOL	Variable zur Neuinitialisierung des DALI-Helligkeits- und Präsenzmelder.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.64	TF8040 | TwinCAT Building Automation ab V5.10.1.0

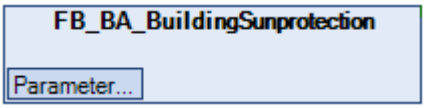
6.1.4.2.1.4.6 SunProtection

Vorlagen zur Erstellung eines individuellen Sonnenschutzes.

6.1.4.2.1.4.6.1 Functions

6.1.4.2.1.4.6.1.1 Building

6.1.4.2.1.4.6.1.1.1 FB_BA_BuildingSunprotection



Dieses Template stellt gebäudeübergreifende Kriterien für die Jalousiefunktionen zusammen, die dann in den Fassadeninstanzen des FB_BA_Facade [▶ 956] weiter genutzt werden. Dies sind im Einzelnen:

- **resultierendes Telegramm Gebäudealarme**
- **Schutztelegramm Feuer**
  Das Template FB_BA_BuildingAlarms [▶ 863] liefert über eine global deklarierte Strukturvariable stBuildingAlarms (siehe globale Variablenliste Site [▶ 1135]) die Information "Feueralarm". Ist dieser Alarm aktiv, so werden die Jalousien über ein FB_BA_SunBldEvt [▶ 388] ganz hochgefahren.
- **Schutztelegramm Einbruch**
  Das Template FB_BA_BuildingAlarms [▶ 863] liefert über eine global deklarierte Strukturvariable stBuildingAlarms (siehe globale Variablenliste Site [▶ 1135]) die Information "Einbruchalarm". Ist dieser Alarm aktiv, so werden die Jalousien über ein FB_BA_SunBldEvt [▶ 388] ganz hochgefahren. Dadurch ist der Einbrecher von außen weniger blickgeschützt.
- **Schutztelegramm Vereisung**
  Eine bevorstehende Vereisung wird dadurch erkannt, dass während einer Niederschlagserkennung die gemessene Außentemperatur unterhalb des Frost-Grenzwertes liegt - dieser ist hier durch das Objekt *SpIce* vorgegeben und auf -2 °C voreingestellt. Übersteigt die Außentemperatur den Frost-Grenzwert für die Zeit, die am Objekt *DeiceTi* vorgegeben wird, so ist der Frostschutz wieder aufgehoben. Im Falle des Vereisungsalarms wird am Baustein FB_BA_SunBldIcePrtc [▶ 389] ein Telegramm ausgegeben, was die Jalousien ganz hochfahren lässt.
- **Thermoautomatik**
  In der Regel wird von einer Wetterstation die globale Wärmeeinstrahlung nicht richtungsabhängig gemessen. Daher ist es sinnvoll, einen Ein- und einen Ausschaltschwellwert pro Gebäude zu definieren. Die Objekte *GlobalThAutoValOn* und *GlobalThAutoValOff* definieren dabei in W/m² diese Schwellwerte und bilden über einen Hystereseschalter die gebäudeweite Freigabe der Thermoautomatik (Objekt *GlobalThAutoRlse*).
- **Dämmerungsautomatik**
  Die Dämmerung, welche am Tage nur einen kurzen Zeitraum definiert, wird ebenfalls Gebäude-übergreifend definiert: die vier richtungsabhängigen Helligkeitswerte der Wetterstation werden gemittelt und mit Schwellwerten *GlobalTwiLgtAutoValOn* und *GlobalTwiLgtAutoValOff* belegt.
- **Sonnenschutz**
  Gebäudeweit wird hier das Intervall für die Nachpositionierung der Lamellen bei den Sonnenschutzfunktionen vorgegeben.
- **Reset der Handfunktionen**
  Ein gebäudeweites Kriterium für den Reset der Handfunktionen wird hier definiert. Es richtet sich nach dem Gebäudezeitschaltplan der Energieniveaus, welche einen Rückschluss auf Abwesenheit geben. Alternativ steht ein Binär-Eingabeobjekt zur Verfügung.

Die oben genannten Schutztelegramme werden auf einem Prioritätenschalter FB_BA_SunBldTgmSel4 [▶ 400] zu einem resultierenden Telegramm zusammengefasst.

Dieses wird am Ende des Templates mit den gebäudespezifischen Sonnenschutzdaten in einer global deklarierten Variablenstruktur *stBuildingSunBlind* (siehe globale Variablenliste Site [▶ 1135]) zur Verfügung gestellt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_BuildingSunprotection EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  FireAlert                   : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eFire);
  Burglary                    : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eBurglary);
  IceProtection               : FB_BA_IceProtection;
  GlobalThAutoRlse            : FB_BA_Hys_03;
  GlobalTwiLgtAutoRlse        : FB_BA_Hys_03;
  SunPrtcPosIntval            : FB_BA_AV_Op_SP;
  GlobalResetManMode          : FB_BA_BV_Op_Val;
END_VAR
VAR
  BuildingAlarms              : FB_BA_SunBldTgmSel4;
  GlobalTwiLgtAutoValOnOff    : FB_BA_Swi2P;
  rtManResetEnergLvl          : R_TRIG;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
FireAlert	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für den Feueralarm: lässt die Jalousie ganz nach oben fahren.
Burglary	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für den Einbruch: lässt die Jalousie ganz nach oben fahren.
IceProtection	FB_BA_IceProtection [▶ 955]	Unter-Template zur Darstellung des Vereisungsschutzes.
GlobalThAutoRlse	FB_BA_Hys_03 [▶ 1026]	Hysterese-Template zur Darstellung der globalen Thermo-Automatik-Freigabe bedingt durch die Globalstrahlung. Beinhaltet Grenzwerte, Verzögerungen und ein Binärobjekt zur Anzeige.
GlobalTwiLgtAutoRlse	FB_BA_Hys_03 [▶ 1026]	Hysterese-Template zur Darstellung der globalen Dämmerungsautomatik-Freigabe bedingt durch die mittlere Helligkeit. Beinhaltet Grenzwerte, Verzögerungen und ein Binärobjekt zur Anzeige.
SunPrtcPosIntval	FB_BA_AV_Op_SP [▶ 1047]	Analoges Eingabeobjekt: Nachstell-Intervall des Lamellenwinkels [min].
GlobalResetManMode	FB_BA_BV_Op_Val [▶ 1049]	Binäres Eingabeobjekt: Bedienmöglichkeit, um Handfunktionen zurückzusetzen.

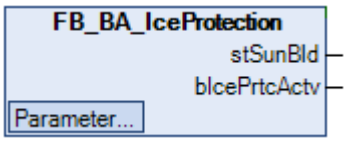
Variablen

Name	Typ	Beschreibung
BuildingAlarms	FB_BA_SunBldTgmSel4 [▶ 400]	Prioritäts-Auswahlbaustein.
GlobalTwiLgtAutoValOnOff	FB_BA_Swi2P	Hysterese-Baustein zum Ein-Ausschalten der Dämmerungsautomatik (globales Kriterium).
rtManResetEnergLvl	R_TRIG	Trigger Reset Handfunktion für Kriterium Energieniveau (Dauersignal).

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.6.1.1.2 FB_BA_IceProtection



Untertemplate Vereisungsschutz.

Der Baustein FB_BA_IceProtection realisiert den Vereisungsschutz. Sinkt die Temperatur unter den Gefrierpunkt, welcher am Objekt *SPIce* eingegeben wird und übermittelt die Wetterstation über die Site-Variablenliste [▶ 1135] Niederschlag (*Site.stWeatherStation.bRain*), so ist der Vereisungsfall gegeben und die Jalousien werden ganz hochgefahren.

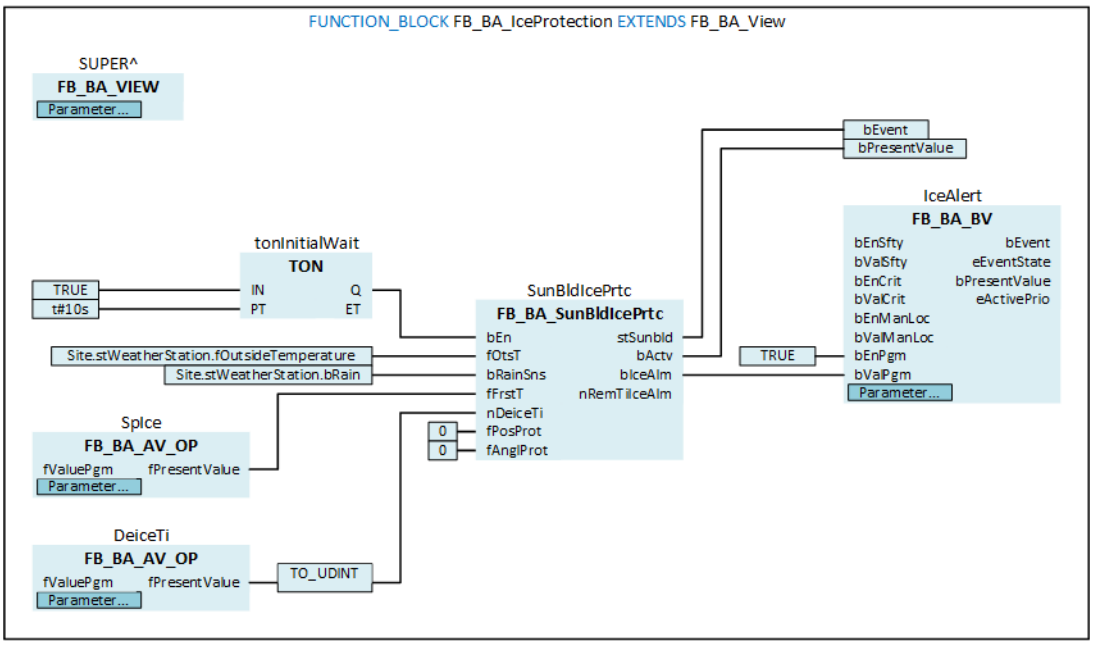
Erst wenn die Außentemperatur wieder für eine am Objekt *DeiceTi* eingestellte Enteisungszeit (in Sekunden) die Vereisungstemperatur überschreitet, wird der Alarm aufgehoben.

Das Telegramm ist dann nicht mehr aktiv.

Um sicherzustellen, dass aktuelle Temperaturwerte bei Programmstart vorliegen, wird die Aktivierung des Vereisungs-Alarmbausteines initial um 10s verzögert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_IceProtection EXTENDS FB_BA_View
VAR_OUTPUT
  stSunBld          : ST_BA_SunBld;
  bIcePrtcActv      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SpIce             : FB_BA_AV_Op;
  DeiceTi           : FB_BA_AV_Op;
  IceAlert          : FB_BA_BV;
  SunBldIcePrtc     : FB_BA_SunBldIcePrtc;
END_VAR
VAR
  tonInitialWait    : TON;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Vereisungsschutz-Telegramm.
bIcePrtcActv	BOOL	Kontrollausgang "Vereisungsschutz aktiv".

Eingänge CONSTANT

Name	Typ	Beschreibung
SpIce	FB_BA_AV_Op [▶ 207]	Eingabe Grenzwert Vereisung [°C].
DeiceTi	FB_BA_AV_Op [▶ 207]	Eingabe Enteisungszeit [s].
IceAlert	FB_BA_BV [▶ 218]	Anzeigeobjekt Status Vereisungsschschutz.
SunBldIcePrtc	FB_BA_SunBldIcePrtc [▶ 389]	Vereisungsschutz der Lamellen.

Variablen

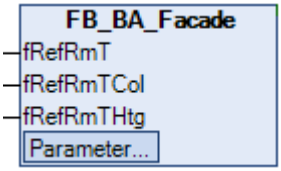
Name	Typ	Beschreibung
tonInitialWait	TON	Initial-Verzögerung der Bausteinfreigabe.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.6.1.2 Facade

6.1.4.2.1.4.6.1.2.1 FB_BA_Facade



Dieses Template stellt die Sonnenschutztelegramme zusammen, welche jeweils für eine ganze Fassade gültig sind.

Die Telegramme werden im Hauptteil (main part) größtenteils in Untertemplates gebildet, dann auf einen Telegrammselektor FB_BA_SunBldTgmSel8 [▶ 401] geführt und um das resultierende Alarmtelegramm aus den Gebäudedaten (Feuer, Einbruch oder Vereisung) ergänzt.

Das durchgereichte Telegramm des Selektors wird zusammen mit den Freigaben von Thermo- und Dämmerungsautomatik sowie den aktuellen Sonnenschutz-Daten auf die Ausgabestruktur stFacade gelegt - die Priorität des Telegramms wird im Untertemplate FacadeInformation [▶ 962] zur Anzeige gebracht.

Telegramme

Folgende Telegramme liegen sortiert nach Eingang am Telegrammselektor [▶ 401] *SunBldTgmResult* an:

- **Schutztelegramm Kommunikationsfehler (CommError)**
  In der globalen Variablenliste Site [▶ 1135] sind ebenfalls die Fehlerzustände der Subscriber hinterlegt, die nur dann auf TRUE wechseln können, falls der entsprechende Subscriber tatsächlich benutzt wird. Im Falle eines Subscriber-Ausfalls oder aber durch eine Wetterstations-Störung werden die Jalousien sicherheitshalber hochgefahren.
- **Schutztelegramm Sturm (**WindProtection [▶ 973]**)**

- **Positioniertelegramm Wartung (**Maintenance [▶ 963]**)**
- **Positioniertelegramm Fassaden-Thermoautomatik (**ThermoAutomatic [▶ 969]**)**
- **Positioniertelegramm Dämmerungsautomatik (**TwilightAutomatic [▶ 972]**)**
- **Positioniertelegramm Parkposition (**ParkPosition [▶ 964]**)**
- **Alarmtelegramm Feuer/Einbruch/Vereisung**
  Dieses Telegramm (*Site.stBuildingSunBlind.stSunBld*) wird üblicherweise auf dem Gebäudecontroller im Template FB_BA_BuildingSunprotection [▶ 953] gebildet und auf die Site-Variablenliste [▶ 1135] gelegt. Es beinhaltet die gebäudeweiten Telegramme für Vereisung, Einbruch und Feuer.

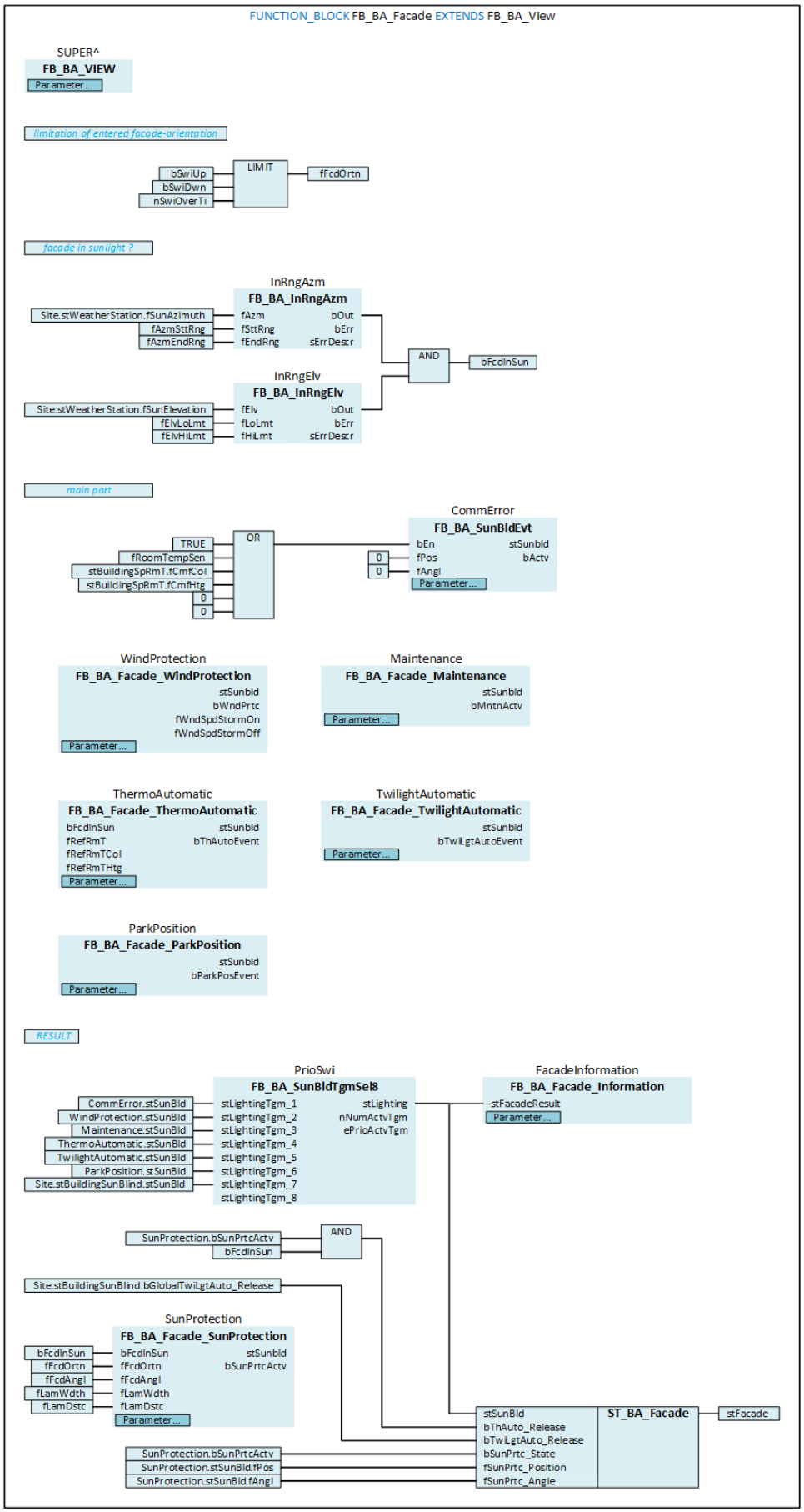
Sonnenschutzberechnung

Für jede Fassade wird der Sonnenschutz getrennt berechnet. Dieses erfolgt im Unter-Template SunProtection [▶ 966]. Die errechneten Werte werden zusammen mit den Freigaben von Thermo- und Dämmerungsautomatik auf die Ausgabestruktur stFacade [▶ 717] gelegt.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Facade EXTENDS FB_BA_View
VAR_INPUT
  fRefRmT                 : REAL;
  fRefRmTCol              : REAL;
  fRefRmTHtg              : REAL;
END_VAR
VAR_OUTPUT
  stFacade                : ST_BA_Facade;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  fFcdOrtn                : REAL;
  fFcdAngl                : REAL;
  fLamWdth                : REAL;
  fLamDstc                : REAL;
  fAzmSttRng              : REAL;
  fAzmEndRng              : REAL;
  fElvLoLmt               : REAL;
  fElvHiLmt               : REAL;
END_VAR
VAR_INPUT CONSTANT
  CommError               : FB_BA_SunBldEvt :=(ePrio:=E_BA_SunBldPrio.eCommError);
  WindProtection          : FB_BA_Facade_WindProtection;
  Maintenance             : FB_BA_Facade_Maintenance;
  ThermoAutomatic         : FB_BA_Facade_ThermoAutomatic;
  TwilightAutomatic       : FB_BA_Facade_TwilightAutomatic;
  ParkPosition            : FB_BA_Facade_ParkPosition;
 SunProtection            : FB_BA_Facade_SunProtection;
 FacadeInformation        : FB_BA_Facade_Information;
END_VAR
VAR
  InRngAzm                : FB_BA_InRngAzm;
  InRngElv                : FB_BA_InRngElv;
  bFcdInSun               : BOOL;
  SunBldTgmResult         : FB_BA_SunBldTgmSel8;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fRefRmT	REAL	Raumtemperatur des Referenzraumes für die fassadenweite Thermoautomatik.
fRefRmTCol	REAL	Raumtemperatursollwert Kühlen des Referenzraumes für die fassadenweite Thermoautomatik.
fRefRmTHtg	REAL	Raumtemperatursollwert Heizen des Referenzraumes für die fassadenweite Thermoautomatik.

Ausgänge

Name	Typ	Beschreibung
stFacade	ST_BA_Facade [▶ 717]	Ausgabestruktur der gesammelten Fassadendaten.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fFcdOrtn	REAL	Fassadenausrichtung Nordhalbkugel: Nord = 0°, Ost = 90°, Süd = 180°, West = 270°, auf der Südhalbkugel gilt: Süd = 0°, Ost = 90°, Nord = 180°, West = 270°.
fFcdAngl	REAL	Neigung der Fassade [°]. Nach unten geneigt ist der Winkel kleiner, nach oben größer Null.
fLamWdth	REAL	Lamellenbreite [mm].
fLamDstc	REAL	Lamellenabstand [mm].
fAzmSttRng / fAzmEndRng	REAL	Die Fassade gilt als von der Sonne beschienen, wenn der Sonnenstand +/-90° der Fassadenorientierung ist. Mit *fAzmSttRng* / *fAzmEndRng* kann der Bereich eingeschränkt werden.
fElvLoLmt / fElvHiLmt	REAL	Die Fassade gilt als von der Sonne beschienen, wenn die Sonnenhöhe zwischen 0° und 90° liegt. Mit *fElvLoLmt* / *fElvHiLmt* kann der Bereich eingeschränkt werden.

Eingänge CONSTANT

Name	Typ	Beschreibung
CommError	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Kommunikationsstörung
WindProtection	FB_BA_Facade_WindProtection [▶ 973]	Untertemplate Sturmschutz
Maintenance	FB_BA_Facade_Maintenance [▶ 963]	Untertemplate Wartung
ThermoAutomatic	FB_BA_Facade_ThermoAutomatic [▶ 969]	Untertemplate Thermoautomatik
TwilightAutomatic	FB_BA_Facade_TwilightAutomatic [▶ 972]	Untertemplate Dämmerungsautomatik
ParkPosition	FB_BA_Facade_ParkPosition [▶ 964]	Untertemplate Parkposition
SunProtection	FB_BA_Facade_SunProtection [▶ 966]	Untertemplate Sonnenschutz
FacadeInformation	FB_BA_Facade_Information [▶ 962]	Untertemplate Fassadeninformationen

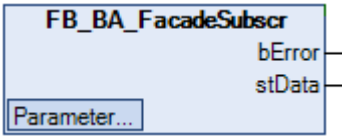
Variablen

Name	Typ	Beschreibung
InRngAzm	FB_BA_InRngAzm [▶ 357]	Sonnenrichtung innerhalb der definierten Grenzen
InRngElv	FB_BA_InRngElv [▶ 360]	Sonnenhöhe innerhalb der definierten Grenzen
bFcdInSun	BOOL	Fassade im Sonnenstand
SunBldTgmResult	FB_BA_SunBldTgmSel8 [▶ 401]	Telegrammauswahlbaustein für das resultierende Jalousietelegramm der Fassade

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.6.1.2.2 FB_BA_FacadeSubscr



Dieses Template dient dem Empfang der Fassadendaten vom Typ ST_BA_Facade [▶ 717] und wird typischerweise in den Etagencontrollern eingesetzt.

Es stellt eine Erweiterung des FB_BA_Subscriber dar: Im Falle des Kommunikationsausfalls wird das Jalousietelegramm innerhalb der Fassadendaten *stSunbld* mit einem aktiven Telegramm Position 0 % überschrieben.

Syntax

```
FUNCTION_BLOCK FB_BA_FacadeSubscr EXTENDS FB_BA_Subscriber
VAR_OUTPUT
  stData          : ST_BA_Facade;
END_VAR
VAR
   _fbHighPrio    : FB_BA_SunBldEvt := (ePrio := E_BA_SunBldPrio.eCommError);
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
stData	ST_BA_Facade [▶ 717]	Gelesenes, im Fehlerfall überschriebenes Fassadentelegramm.

Variablen

Name	Typ	Beschreibung
_fbHighPrio	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein, welcher im Fall fehlerhafter Kommunikation ein hochpriorisiertes (*ePrio := E_BA_SunBldPrio.eCommError*) Telegramm mit Position 0% aktiviert und damit die Jalousie ganz hoch fahren lässt.

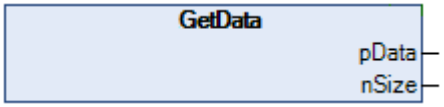
Methoden

Name	Beschreibung
GetData [▶ 961]	Ermittelt Adresse und Größe des Ausgabetelegramms.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.GetData
4.2.
1.4.
6.1.
2.2.
1



Intern genutzte Methode, die Adresse und Größe des Ausgabetelegramms ermittelt.

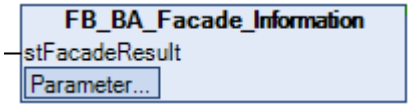
Syntax

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

Ausgänge

Name	Typ	Beschreibung
pData	pData	Pointer auf die abonnierten Daten.
nSize	DINT	Größe der abonnierten Daten.

6.1.4.2.1.4.6.1.2.3 FB_BA_Facade_Information



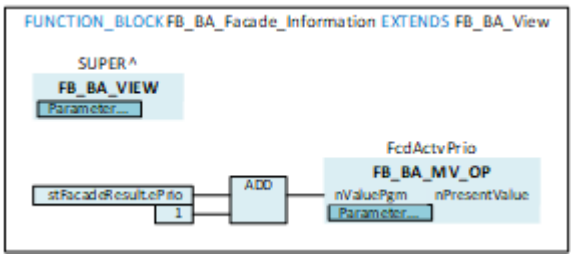
Untertemplate Fassadeninformation.

Dieses Template bringt die Priorität des aktuellen Sonnenschutztelegramms im BACnet zur Ansicht.

Da die Enumeration der Prioritäten mit "0" beginnt, die Texte des MV-Objektes jedoch mit "1" (siehe *FB_Init*), muss zur Enumeration eine eins hinzuaddiert werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_Information EXTENDS FB_BA_View
VAR_INPUT
  stFacadeResult    : ST_BA_SunBld;
END_VAR
VAR_INPUT CONSTANT
  FcdActvPrio       : FB_BA_MV_Op;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stFacadeResult	ST_BA_SunBld [▶ 279]	Resultierendes Telegramm aus dem übergeordneten Fassaden-Template [▶ 956].

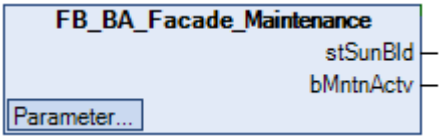
Eingänge CONSTANT

Name	Typ	Beschreibung
FcdActvPrio	FB_BA_MV_Op [▶ 244]	Baustein zur Darstellung der aktuellen Telegrammpriorität im BACnet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.6.1.2.4 FB_BA_Facade_Maintenance



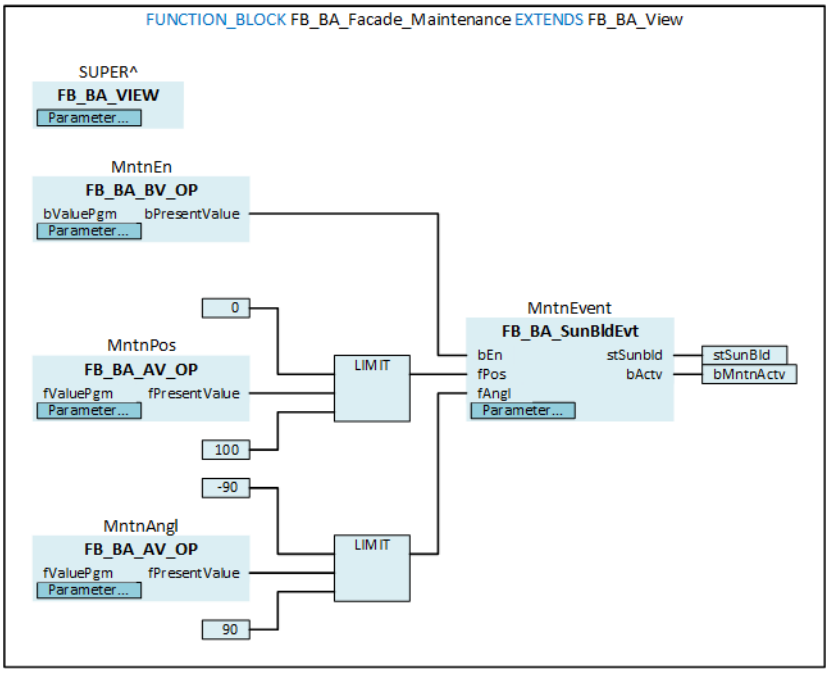
Untertemplate Wartungsposition.

Diese Funktion lässt für Wartungsarbeiten über Objekte eine Position und einen Winkel eingeben und das entsprechende Sonnenschutz-Telegramm aktivieren,

welches dann am Ausgang *stSunblind* dem übergeordneten Fassaden-Template [▶ 956] zur Verfügung steht.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_Maintenance EXTENDS FB_BA_View
VAR_OUTPUT
  stSunBld     : ST_BA_SunBld;
  bMntnActv    : BOOL;
END_VAR
VAR_INPUT CONSTANT
  MntnEn       : FB_BA_BV_Op;
  MntnPos      : FB_BA_AV_Op;
  MntnAngl     : FB_BA_AV_Op;
  MntnEvent    : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eMaintenance);
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Wartungs-Telegramm.
bMntnActv	BOOL	Kontrollausgang "Wartung aktiv".

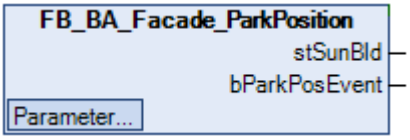
Eingänge CONSTANT

Name	Typ	Beschreibung
MntnEn	FB_BA_BV_Op [▶ 221]	Baustein zur Aktivierung des Wartungstelegramms.
MntnPos	FB_BA_AV_Op [▶ 207]	Baustein zum Eintrag der Position.
MntnAngl	FB_BA_AV_Op [▶ 207]	Baustein zum Eintrag des Winkels.
MntnEvent	FB_BA_SunBldEvt [▶ 388]	Baustein zur Telegrammerzeugung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.6.1.2.5 FB_BA_Facade_ParkPosition



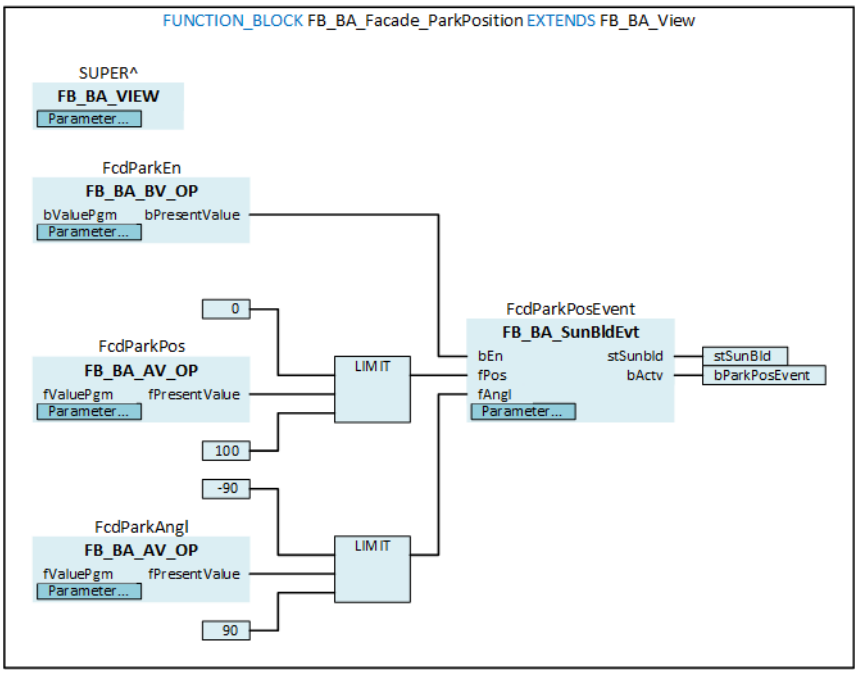
Untertemplate Parkposition.

Diese Funktion lässt eine Park-Position und einen Park-Winkel eingeben und das entsprechende Sonnenschutz-Telegramm aktivieren,

welches dann am Ausgang *stSunblind* dem übergeordneten Fassaden-Template [▶ 956] zur Verfügung steht.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_ParkPosition EXTENDS FB_BA_View
VAR_OUTPUT
  stSunBld           : ST_BA_SunBld;
  bParkPosEvent      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  FcdParkEn          : FB_BA_BV_Op;
  FcdParkPos         : FB_BA_AV_Op;
  FcdParkAngl        : FB_BA_AV_Op;
  FcdParkPosEvent    : FB_BA_SunBldEvt := (ePrio:=E_BA_SunBldPrio.eParkPosition);
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Parkposition-Telegramm.
bParkPosEvent	BOOL	Kontrollausgang "Parkposition aktiv".

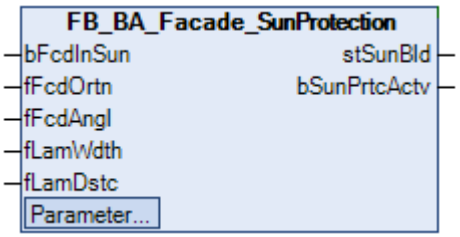
Eingänge CONSTANT

Name	Typ	Beschreibung
FcdParkEn	FB_BA_BV_Op [▶ 221]	Baustein zur Aktivierung des Wartungstelegramms.
FcdParkPos	FB_BA_AV_Op [▶ 207]	Baustein zum Eintrag der Position.
FcdParkAngl	FB_BA_AV_Op [▶ 207]	Baustein zum Eintrag des Winkels.
FcdParkPosEvent	FB_BA_SunBldEvt [▶ 388]	Baustein zur Telegrammerzeugung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.6.1.2.6 FB_BA_Facade_SunProtection



Untertemplate Sonnenschutzautomatik.

Dieses Template ruft eine Sonnenschutzfunktion mit fester Position und Lamellennachführung auf.

Im oberen Teil wird zunächst anhand der Fassadenorientierung (Nord = 0°, Ost = 90°, Süd = 180°, West = 270°) und aus den 4 Helligkeitssensoren die Helligkeit für die betrachtete Fassade interpoliert (*BrtnsCardinalInterpolation*).

Da die Blendwirkung der Sonne auch von der Sonnenhöhe anhängt werden jeweils 7 Helligkiswerte für die Ein - und Ausschaltschwelle des Sonnenschutzes herangezogen. Diese sind als AV-Objekte definiert und beschreiben sie Schwellwerte bei 0°, 15°, 30°, 45°, 60°, 75° und 90° Sonnenelevation.

Anhand der aktuellen Sonnenhöhe wird der Ein- und Ausschaltschwellwert mit jeweils einem Baustein FB_BA_Chrct07 [▶ 471] interpoliert.

Die jeweils Errechneten Schwellwerte für Ein und Aus werden jeweils über ein AV-Objekt im BACnet zur Verfügung gestellt.

Im sich anschließenden Teil erzeugt ein Hysterese-Baustein die eigentliche Freigabe für die Sonnenschutzfunktion.

Dazu wird die interpolierte himmelsrichtungsabhängige Helligkeit als Istwert herangezogen.

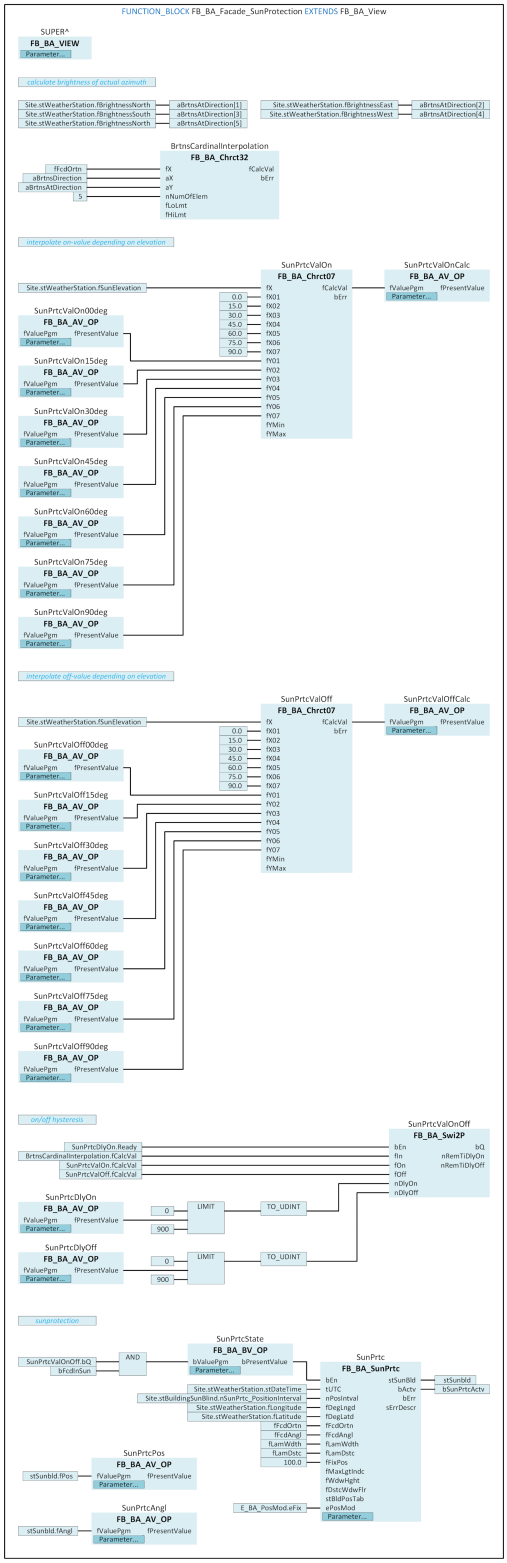
Zusätzlich ist eine zeitliche Verzögerung der Grenzwerte einprogrammiert, die einem schnellen Helligkeitswechsel Rechnung trägt.

Die Freigabe des Sonnenschutzbausteines wird mit dem Kriterium "*bFcdInSun*" (Fassade ist im Sonnenbereich) eingeschränkt und anschließend über ein BV-Objekt zur Anzeige im BACnet geführt. Die globalen Daten für die Sonnenschutzfunktion - Datum/Uhrzeit, Positionierintervall, Längen- und Breitengrad der Liegenschaft - stammen aus der lokalen Site [▶ 1135]-Variablenliste, während Fassadenorientierung, Fassadenneigung, Lamellenbreite und Lamellenabstand als Eingänge aus dem übergeordneten FB_BA_Facade [▶ 956] als Parameter stammen. Der ermittelte Jalousiewinkel sowie die Position werden als Analogobjekte im BACnet zur Ansicht gebracht.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_SunProtection EXTENDS FB_BA_View
VAR_INPUT
  bFcdInSun                     : BOOL;
  fFcdOrtn                      : REAL;
  fFcdAngl                      : REAL;
  fLamWdth                      : REAL;
  fLamDstc                      : REAL;
END_VAR
VAR_OUTPUT
  stSunbld                      : ST_BA_SunBld;
  bSunPrtcActv                  : BOOL;
```

```
END_VAR
VAR_INPUT CONSTANT
SunPrtcValOn00deg               : FB_BA_AV_Op;
  SunPrtcValOn15deg             : FB_BA_AV_Op;
  SunPrtcValOn30deg             : FB_BA_AV_Op;
  SunPrtcValOn45deg             : FB_BA_AV_Op;
  SunPrtcValOn60deg             : FB_BA_AV_Op;
  SunPrtcValOn75deg             : FB_BA_AV_Op;
  SunPrtcValOn90deg             : FB_BA_AV_Op;
  SunPrtcValOff00deg            : FB_BA_AV_Op;
  SunPrtcValOff15deg            : FB_BA_AV_Op;
  SunPrtcValOff30deg            : FB_BA_AV_Op;
  SunPrtcValOff45deg            : FB_BA_AV_Op;
  SunPrtcValOff60deg            : FB_BA_AV_Op;
  SunPrtcValOff75deg            : FB_BA_AV_Op;
  SunPrtcValOff90deg            : FB_BA_AV_Op;
  SunPrtcValOnCalc              : FB_BA_AV_Op;
  SunPrtcValOffCalc             : FB_BA_AV_Op;
  SunPrtcDlyOn                  : FB_BA_AV_Op;
  SunPrtcDlyOff                 : FB_BA_AV_Op;
  SunPrtcState                  : FB_BA_BV_Op;
  SunPrtcPos                    : FB_BA_AV_Op;
  SunPrtcAngl                   : FB_BA_AV_Op;
  SunPrtc                       : FB_BA_SunPrtc;
END_VAR
VAR
  aBrtnsDirection               : ARRAY[1..32] OF REAL :=[0.0, 90.0, 180.0, 270.0, 360.0];
  aBrtnsAtDirection             : ARRAY[1..32] OF REAL;
  BrtnsCardinalInterpolation    : FB_BA_Chrct32;
  SunPrtcValOn                  : FB_BA_Chrct07;
  SunPrtcValOff                 : FB_BA_Chrct07;
  SunPrtcValOnOff               : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bFcdInSun	BOOL	Die Fassade liegt in Bezug auf die aktuelle Sonnenhöhe und Sonnenrichtung im Sonnenbereich. Dieses Kriterium wird im übergeordneten FB_BA_Facade [▶ 956] gebildet.
fFcdOrtn	REAL	Fassadenausrichtung Nordhalbkugel: Nord = 0°, Ost = 90°, Süd = 180°, West = 270°, auf der Südhalbkugel gilt: Süd = 0°, Ost = 90°, Nord = 180°, West = 270°.
fFcdAngl	REAL	Neigung der Fassade [°]. Nach unten geneigt ist der Winkel kleiner, nach oben größer Null.
fLamWdth	REAL	Lamellenbreite [mm].
fLamDstc	REAL	Lamellenabstand [mm].

Ausgänge

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Parkposition-Telegramm.
bSunPrtcActv	BOOL	Kontrollausgang "Sonnenschutz aktiv".

Eingänge CONSTANT

Name	Typ	Beschreibung
SunPrtcValOn00deg … SunPrtcValOn90deg	FB_BA_AV_Op [▶ 207]	Helligkeitsschwellwerte Sonnenschutz Ein [lx] für Elevationen von 0°, 15°, 30°, 45°, 60°, 75° und 90°.
SunPrtcValOff00deg … SunPrtcValOff90deg	FB_BA_AV_Op [▶ 207]	Helligkeitsschwellwerte Sonnenschutz Aus [lx] für Elevationen von 0°, 15°, 30°, 45°, 60°, 75° und 90°.
SunPrtcValOnCalc	FB_BA_AV_Op [▶ 207]	Errechneter Helligkeitsschwellwert Sonnenschutz Ein [lx] für die Anzeige im BACnet.
SunPrtcValOffCalc	FB_BA_AV_Op [▶ 207]	Errechneter Helligkeitsschwellwert Sonnenschutz Aus [lx] für die Anzeige im BACnet.
SunPrtcDlyOn	FB_BA_AV_Op [▶ 207]	Verzögerung Sonnenschutz ein [s].
SunPrtcDlyOff	FB_BA_AV_Op [▶ 207]	Verzögerung Sonnenschutz aus [s].
SunPrtcState	FB_BA_BV_Op [▶ 221]	Anzeigeobjekt Status Sonnenschutz. Dieses Objekt zeigt an, ob die Freigabe des Sonnenschutzes seitens der Fassade gegeben ist. Ob die Jalousien tatsächlich die Sonnenschutzposition einnehmen, hängt auch von Kriterien innerhalb der Raumzonen ab (z.B. Präsenz).
SunPrtcPos	FB_BA_AV_Op [▶ 207]	Errechnete Sonnenschutzposition [%] für die Anzeige im BACnet.
SunPrtcAngl	FB_BA_AV_Op [▶ 207]	Errechnete Sonnenschutzwinkel [°] für die Anzeige im BACnet.
SunPrtc	FB_BA_SunPrtc [▶ 406]	Berechnungsbaustein für den Sonnenschutz.

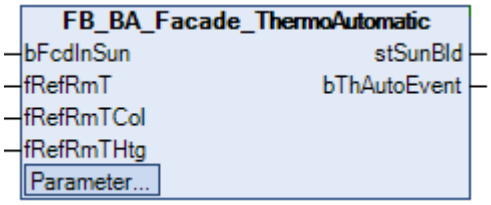
Variablen

Name	Typ	Beschreibung
aBrtnsDirection	ARRAY[1..32] OF REAL	Voreingestelltes Feld von Himmelsrichtungen zur Wichtung der Helligkeit.
aBrtnsAtDirection	ARRAY[1..32] OF REAL	Eingabefeld für die Helligkeiten bezogen auf die o.a. Himmelsrichtungen.
BrtnsCardinalInterpolation	FB_BA_Chrct32 [▶ 473]	Berechnungsbaustein für die Helligkeit.
SunPrtcValOn	FB_BA_Chrct07 [▶ 471]	Berechnungsbaustein für die elevationsabhängige Helligkeitsschwelle Ein.
SunPrtcValOff	FB_BA_Chrct07 [▶ 471]	Berechnungsbaustein für die elevationsabhängige Helligkeitsschwelle Aus.
SunPrtcValOnOff	FB_BA_Swi2P	Hysteresebaustein zum Ein-Ausschalten des Sonnenschutzes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.6.1.2.7 FB_BA_Facade_ThermoAutomatic



Untertemplate fassadenweite Thermoautomatik.

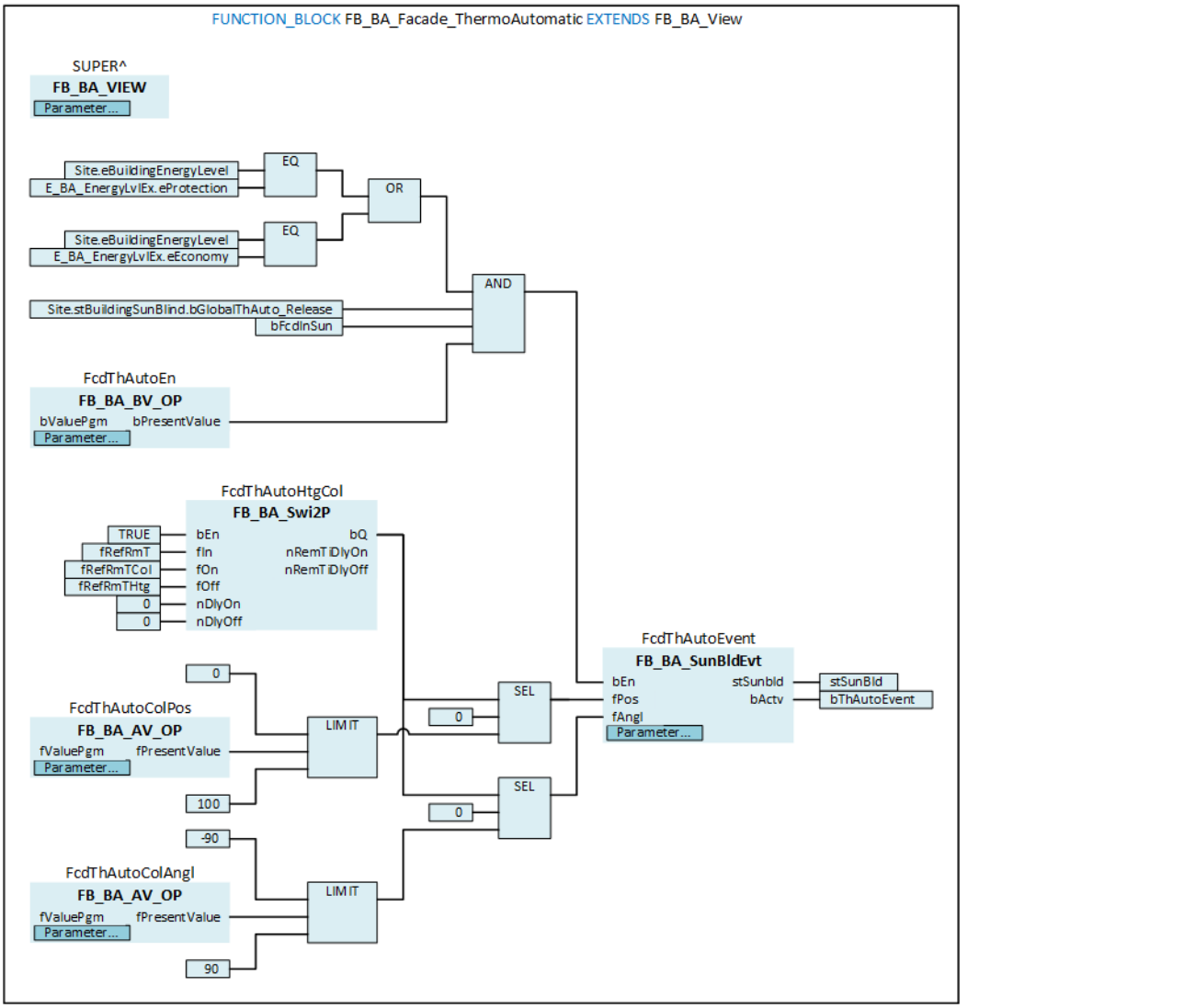
Eine Fassadenweite Thermo-Automatik benötigt die Temperatur eines Referenzraumes sowie dessen Heiz- und Kühlsollwert. Diese Daten werden über die Eingangsvariablen *fRefRmT*, *fRefRmTCol* und *fRefRmTHtg* vorgegeben.

Vier Bedingungen müssen erfüllt sein, dass die Fassaden-Thermoautomatik aktiv werden kann:

- das Gebäude muss unbelegt sein, dies wird anhand des aktuell gültigen Energieniveaus ermittelt.
- die globale strahlungsabhängige Freigabe aus dem Template FB_BA_BuildingSunprotection [▶ 953] muss gegeben sein. Dieses Kriterium ist auf die lokale Site [▶ 1135]-Variablenliste geführt und wird von dort gelesen (*Site.stBuildingSunBlind.bGlobalThAuto_Release*).
- die Fassade muss von der Sonne beschienen sein (*bFcdInSun*).
- die fassadenweite Thermoautomatik ist über das Objekt *FcdThAutoEn* angewählt.

Sind diese Bedingungen erfüllt, so fährt die Jalousie wechselweise auf Kühlposition (Objekte *FcdThAutoColPos* und *FcdThAutoColAngl*) wenn die Referenztemperatur den Kühlwert überschreitet oder auf Position 0, wenn der Heizwert unterschritten wird.

Die Thermoautomatik lässt sich über das Objekt *FcdThAutoEn* an- und abwählen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild

Syntax

```
FUNCTION_BLOCK FB_BA_Facade_ThermoAutomatic EXTENDS FB_BA_View
VAR_INPUT
  bFcdInSun              : BOOL;
  fRefRmT                : REAL;
  fRefRmTCol             : REAL;
  ReffRmTHtg             : REAL;
END_VAR
VAR_OUTPUT
  stSunBld               : ST_BA_SunBld;
  bThAutoEvent           : BOOL;
END_VAR
VAR_INPUT CONSTANT
  FcdThAutoEn            : FB_BA_BV_Op;
  FcdThAutoColPos        : FB_BA_AV_Op;
  FcdThAutoColAngl       : FB_BA_AV_Op;
  FcdThAutoEvent         : FB_BA_SunBldEvt :=(ePrio:=E_BA_SunBldPrio.eFacadeThermoAutomatic);
END_VAR
VAR
  FcdThAutoHtgCol        : FB_BA_Swi2P;
  bFcdThAuto_Enable      : BOOL;
  fFcdThAuto_Position    : REAL;
  fFcdThAuto_Angle       : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bFcdInSun	BOOL	Die Fassade liegt in Bezug auf die aktuelle Sonnenhöhe und Sonnenrichtung im Sonnenbereich. Dieses Kriterium wird im übergeordneten FB_BA_Facade [▶ 956] gebildet.
fRefRmT	REAL	Raumtemperatur des Referenzraumes für die fassadenweite Thermoautomatik.
fRefRmTCol	REAL	Raumtemperatursollwert Kühlen des Referenzraumes für die fassadenweite Thermoautomatik.
RefRmTHtg	REAL	Raumtemperatursollwert Heizendes Referenzraumes für die fassadenweite Thermoautomatik.

Ausgänge

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Thermoautomatik-Telegramm.
bThAutoEvent	BOOL	Kontrollausgang "fassadenweite Thermoautomatik aktiv".

Eingänge CONSTANT

Name	Typ	Beschreibung
FcdThAutoEn	FB_BA_BV_Op [▶ 221]	Fassaden-Thermoautomatik Freigabe.
FcdThAutoColPos	FB_BA_AV_Op [▶ 207]	Fassaden-Thermoautomatik Kühlstellung Position [%].
FcdThAutoColAngl	FB_BA_AV_Op [▶ 207]	Fassaden-Thermoautomatik Kühlstellung Winkel [°].
FcdThAutoEvent	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Fassaden-Thermoautomatik.

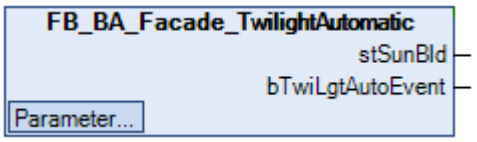
Variablen

Name	Typ	Beschreibung
FcdThAutoHtgCol	FB_BA_Swi2P	Hysteresebaustein zur Umschaltung Heizen/Kühlen für die Fassaden-Thermoautomatik.
bFcdThAuto_Enable	BOOL	Freigabe Thermoautomatik.
fFcdThAuto_Position	REAL	Thermoautomatik Position [%].
fFcdThAuto_Angle	REAL	Thermoautomatik Winkel [°].

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.6.1.2.8 FB_BA_Facade_TwilightAutomatic



Untertemplate fassadenweite Dämmerungsautomatik.

Diese Funktion ist so konzipiert, dass sie nur bei Abwesenheit aktiv ist, um unnötigen Lichtaustritt aus dem Gebäude zu verhindern.

Drei Bedingungen müssen erfüllt sein, dass die Fassadenautomatik aktiv werden kann:

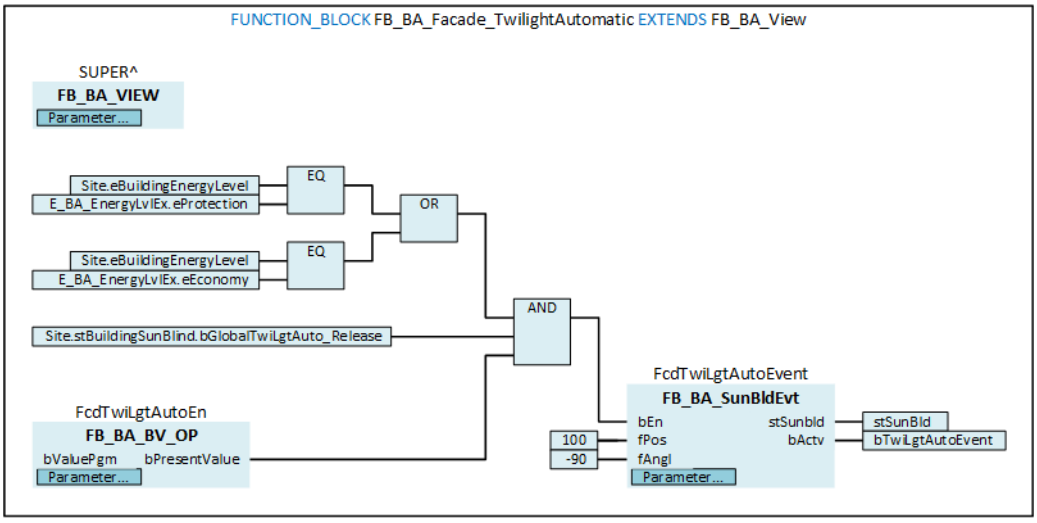
- das Gebäude muss unbelegt sein, dies wird anhand des aktuell gültigen Energieniveaus ermittelt.
- die helligkeitsabhängige Freigabe aus dem Template FB_BA_BuildingSunprotection [▶ 953] muss gegeben sein. Dieses Kriterium ist auf die lokale Site [▶ 1135]-Variablenliste geführt und wird von dort gelesen (*Site.stBuildingSunBlind.bGlobalTwiLgtAuto_Release*).
- die fassadenweite Dämmerungsautomatik ist über das Objekt *FcdTwiLgtAutoEn* angewählt.

Ist sie freigegeben und über das Objekt *FcdTwiLgtAutoEn* angewählt, wird im Dämmerungsfall die Jalousie ganz herunterfahren.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_TwilightAutomatic EXTENDS FB_BA_View
VAR_INPUT
END_VAR
VAR_OUTPUT
  stSunBld              : ST_BA_SunBld;
  bTwiLgtAutoEvent      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  FcdTwiLgtAutoEn       : FB_BA_BV_Op;
  FcdTwiLgtAutoEvent    : FB_BA_SunBldEvt :=(ePrio:=E_BA_SunBldPrio.eFacadeTwilightAutomatic);
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Dämmerungsautomatik-Telegramm.
bTwiLgtAutoEvent	BOOL	Kontrollausgang "fassadenweite Dämmerungsautomatik aktiv".

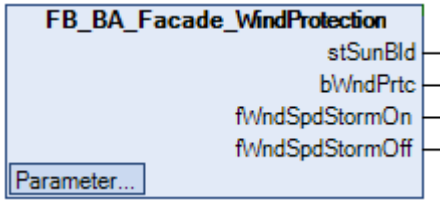
Eingänge CONSTANT

Name	Typ	Beschreibung
FcdTwiLgtAutoEn	FB_BA_BV_Op [▶ 221]	Baustein zur Aktivierung des Dämmerungsautomatik-Telegramms.
FcdTwiLgtAutoEvent	FB_BA_SunBldEvt [▶ 388]	Baustein zur Telegrammerzeugung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.6.1.2.9 FB_BA_Facade_WindProtection



Untertemplate Sturmschutz.

Die Daten für Sturmschutz - Windstärke und Windrichtung - werden mit anderen Wetterdaten von dem Wetterstationstemplate [▶ 868] zunächst auf die lokale Site [▶ 1135]-Variablenliste geführt. Von dort aus stehen sie als

- *Site.stWeatherStation.fWindSpeed*
- *Site.stWeatherStation.fWindDirection*

diesem Template zur Verfügung.

Im oberen Teil der Funktion werden zunächst die Ein- und Ausschaltschwellen für die Aktivierung je nach Windrichtung gewichtet. In den Array-Tabellen *StormOnInterpolation* für die Einschaltschwelle und *StormOffInterpolation* für die Ausschaltschwelle stehen jeweils Grenzwerte für unterschiedliche Windrichtungen in 30°-Schritten zur Verfügung.

Die Idee dahinter ist, dass ein direktes Auftreffen des Windes auf die Fassade stärker gewichtet wird (niedrigerer Grenzwert), als wenn der Wind von der Seite kommt (höherer Grenzwert).

Da dieses Template universell, also für alle Fassadenrichtungen ausgeliefert wird, ist im *FB_Init* die Einschaltschwelle für **alle** Windrichtungen mit 12 m/s (starker Wind) und die Ausschaltschwelle mit 8 m/s vordefiniert. Die Werte sind nach Bedarf zu verfeinern.

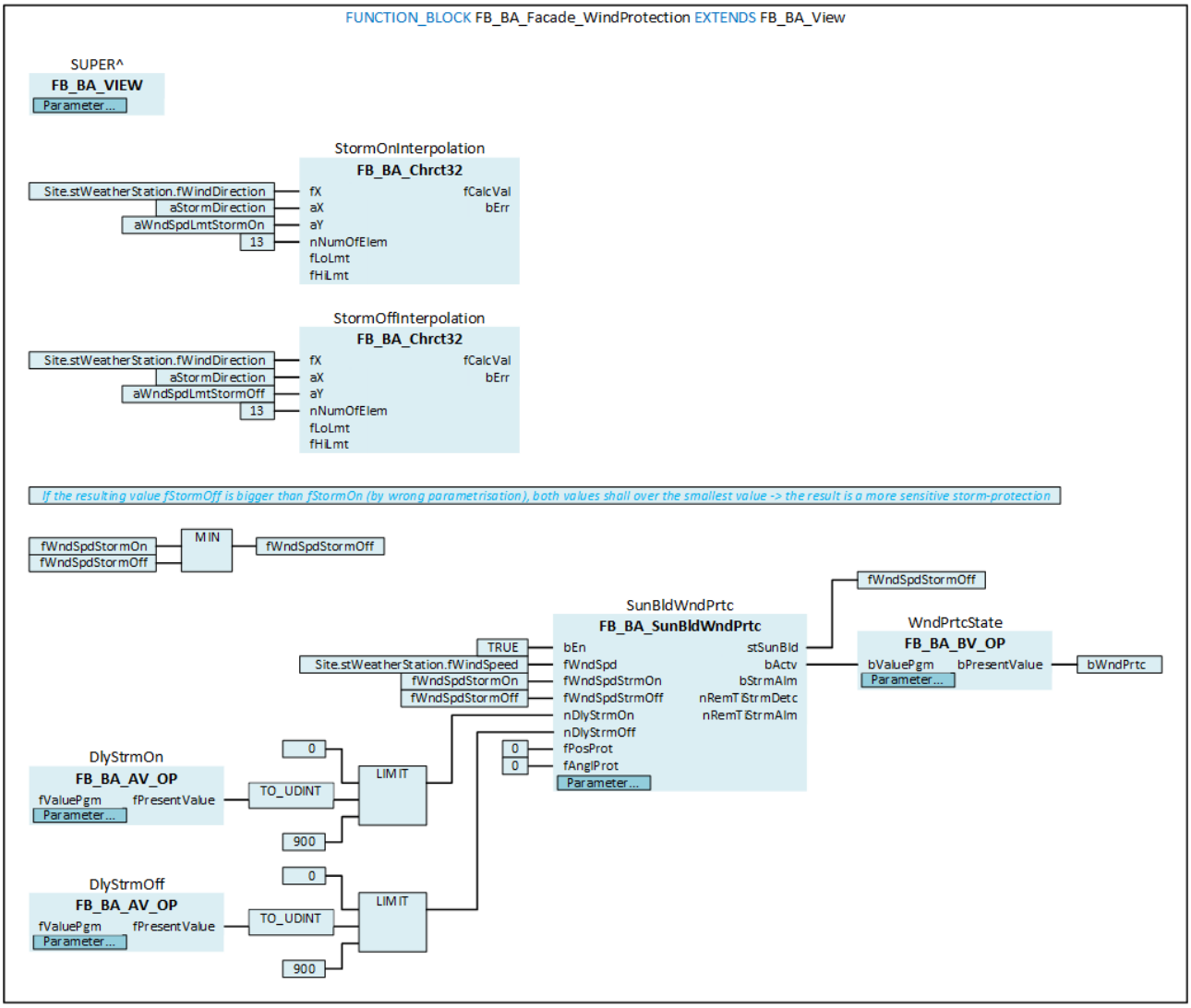
Zu beachten ist auch, dass die Grenzwerte tatsächlich einheitenlos sind und entsprechend der Wetterstation gewählt werden müssen: würde die Wetterstation die Windstäke in km/h ausgeben, sind die Werte in der Größenordnung von 40km/h für Einschalten und 28km/h für Ausschalten passender.

Der Baustein FB_BA_SunBldWndPrtc [▶ 404] realisiert den Sturmschutz. An ihn werden die aktuelle Windgeschwindigkeit aus der Site [▶ 1135]-Variablenliste und die gewichteten Ein- und Ausschaltschwellen geführt. Darüber hinaus besitzt er noch Eingänge für eine zeitliche Ein- und Ausschaltverzögerung, welche über die Eingabeobjekte DlyStrmOn [▶ 207] und DlyStrmOff [▶ 207] eingestellt werden können. Beide Werte sind in diesem Template auf 60 s begrenzt.

Mit dem binären Anzeigeobjekt WndPrtcState [▶ 221] wird der Status des Sturmschutzes im BACnet zur Anzeige gebracht.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Facade_WindProtection EXTENDS FB_BA_View
VAR_OUTPUT
  stSunbld                : ST_BA_SunBld;
  bWndPrtc                : BOOL;
  fWndSpdStormOn          : REAL;
  fWndSpdStormOff         : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  aWndSpdLmtStormOn       : ARRAY[1..32] OF REAL;
  aWndSpdLmtStormOff      : ARRAY[1..32] OF REAL;
END_VAR
VAR_INPUT CONSTANT
  DlyStrmOn               : FB_BA_AV_Op;
  DlyStrmOff              : FB_BA_AV_Op;
  WndPrtcState            : FB_BA_BV_Op;
  SunBldWndPrtc           : FB_BA_SunBldWndPrtc;
END_VAR
VAR
  aStormDirection         : ARRAY[1..32] OF REAL :=[0.0, 30.0, 60.0, 90.0, 120.0, 150.0, 180.0, 210
.0, 240.0, 270.0, 300.0, 330.0, 360.0];
```

```
  StormOnInterpolation     : FB_BA_Chrct32;
  StormOffInterpolation    : FB_BA_Chrct32;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
stSunBld	ST_BA_SunBld [▶ 279]	Sturmschutz-Telegramm.
bWndPrtc	BOOL	Kontrollausgang "Sturmschutz aktiv".
fWndSpdStormOn	REAL	Berechneter Wert Grenzwert Sturmerkennung.
fWndSpdStormOff	REAL	Berechneter Wert Grenzwert Sturm abgeflaut.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
aWndSpdLmtStormOn	ARRAY[1..32] OF REAL	*aWndSpdLmtStormOn[01]* … *aWndSpdLmtStormOn[13]*: Diese Werte definieren die Interpolationskurve für den richtungsabhängigen Einschaltschwellwert der Sturmerkennung. Dabei gilt *aWndSpdLmtStormOn[01]* für die Windrichtung bei 0° (Nord). Die weiteren Feldelemente stellen die Stützpunkte in 30°-Schritten dar. Die Einheit der im *FB_Init* gewählten Werte ist m/s. Sollte die Wetterstation die Windstärke in einer anderen Einheit zur Verfügung stellen, so sind die Werte entsprechend anders zu wählen. Es werden für die Interpolation der Windrichtung nur 13 Werte beschrieben. Der entsprechende Baustein ist dahingehend parametriert.
aWndSpdLmtStormOff	ARRAY[1..32] OF REAL	*aWndSpdLmtStormOff[01]* .. *aWndSpdLmtStormOff[13]*: Diese Werte definieren die Interpolationskurve für den richtungsabhängigen Ausschaltschwellwert der Sturmerkennung. Dabei gilt *aWndSpdLmtStormOff[01]* für die Windrichtung bei 0° (Nord). Die weiteren Feldelemente stellen die Stützpunkte in 30°-Schritten dar. Die Einheit der im *FB_Init* gewählten Werte ist m/s. Sollte die Wetterstation die Windstärke in einer anderen Einheit zur Verfügung stellen, so sind die Werte entsprechend anders zu wählen. Es werden für die Interpolation der Windrichtung nur 13 Werte beschrieben. Der entsprechende Baustein ist dahin gehend parametriert.

Eingänge CONSTANT

Name	Typ	Beschreibung
DlyStrmOn	FB_BA_AV_Op [▶ 207]	Verzögerungswert Sturmerkennung [s].
DlyStrmOff	FB_BA_AV_Op [▶ 207]	Verzögerungswert Sturm abgeflaut [s].
WndPrtcState	FB_BA_BV_Op [▶ 221]	Anzeigeobjekt Status Sturmschutz.
SunBldWndPrtc	FB_BA_SunBldWndPrtc [▶ 404]	Auslöse- und Telegrammbaustein für den Sturmschutz.

Variablen

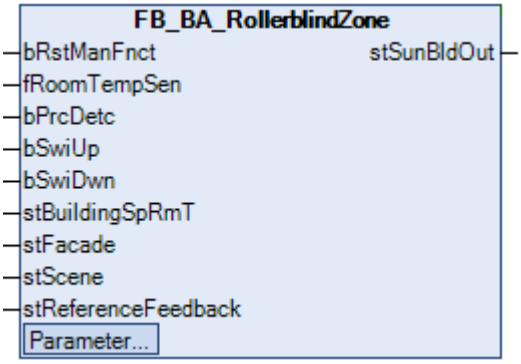
Name	Typ	Beschreibung
aStormDirection	ARRAY[1..32] OF REAL	Voreingestelltes Feld von Himmelsrichtungen zur Wichtung der Sturmgrenzwerte.
StormOnInterpolation	FB_BA_Chrct32 [▶ 473]	Berechnungsbaustein für den Grenzwert Sturmerkennung.
StormOffInterpolation	FB_BA_Chrct32 [▶ 473]	Berechnungsbaustein für den Grenzwert Sturm abgeflaut.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.6.1.3 Zone

6.1.4.2.1.4.6.1.3.1 FB_BA_RollerblindZone



In einer Jalousie-Zone werden eine oder mehrere Jalousien zur gleichzeitigen Ansteuerung zusammengefasst, je nachdem wie es die örtlichen Gegebenheiten erfordern. Das Template bündelt dabei die vorgefertigten Daten der Fassaden und entscheidet aufgrund einer Zonenpräsenzerkennung und örtlicher An- und Abwahl, welche Funktionalität dabei aktiv ist.

Zusammen mit dem resultierenden Telegramm der hoch-priorisierten Funktionen aus der Fassade werden diese Funktionalitäten am Ende des Templates auf einen Telegrammselektor gelegt. Dieser entscheidet dann anhand der Priorität, welches Telegramm im übergeordneten Raumtemplate weiter an die Jalousien gereicht wird.

Funktionen

- **Handfunktion**
  Das Zonentemplate enthält eine Handfunktion, die das manuelle Ansteuern der Jalousie über die Tastenfunktionen *bSwiUp* / *bSwiDwn* ermöglicht. Die Umschaltzeit auf Selbsthaltung, *nSwiOverTi* [ms], ist auf 250 ms vorparametriert. Über den Eingang *bRstManFnct* (von der Basisklasse *FB_BA_Ext_SunblindPosition* (interner Baustein) geerbt) wird die Handfunktion abgelöscht.
- **Thermoautomatik**
  Die Themoautomatik gilt als aktiv, wenn sie lokal angewählt ist (Parameter *bThAutoSlcn*), wenn keine Präsenz erkannt wird (Eingang *bPrcDetc*) und das Fassadentemplate FB_BA_Facade [▶ 956] die Freigabe für diese Gebäudeseite ermittelt hat. Ist sie aktiv, wird anhand der Raumtemperatur (Eingang *fRoomTempSen*) und den Gebäudesollwerten für Heizen und Kühlen über die Positionierung entschieden: Liegt die Raumtemperatur über dem Gebäudewert für Komfort-Kühlen, so fahren die Jalousien auf eine vordefinierbare Position. Sinkt die Temperatur hingegen unter den Wert für Komfort-Heizen, so fahren die Jalousien komplett auf.

- **Sonnenschutzautomatik**
  Die Sonnenschutzautomatik ist im Gegensatz zur Thermoautomatik nur aktiv, wenn Präsenz erkannt wird (Eingang *bPrcDetc*). Auch sie muss lokal angewählt sein (Parameter *bSunPrtcSlcn*) und vom Fassadentemplate die Freigabe bekommen, unter anderem durch die richtungsabhängige Helligkeit. Die Jalousieposition wird ebenfalls in der Fassade ermittelt.
- **Dämmerungsautomatik**
  Die Freigabe der Dämmerungsautomatig erfolgt in der Fassade aufgrund der Helligkeitswerte. Ist die Automatik für die Zone angewählt (Eingang *bTwiLgtAutoSlcn*) so fährt die Jalousie im Dämmerungsfall auf eine vordefinierte Position (Parameter *fTwiLgtAutoPos*).

Eingang stReferenceFeedback

Über diesen Eingang werden Informationen über den angesteuerten Jalousieaktor bzw. den Referenzaktor einer Gruppe in die Jalousiesteuerfunktion zurückgeführt.

Hierbei handelt es sich um Positionsangaben und Status des Referenzaktors.

Datenaustausch HMI

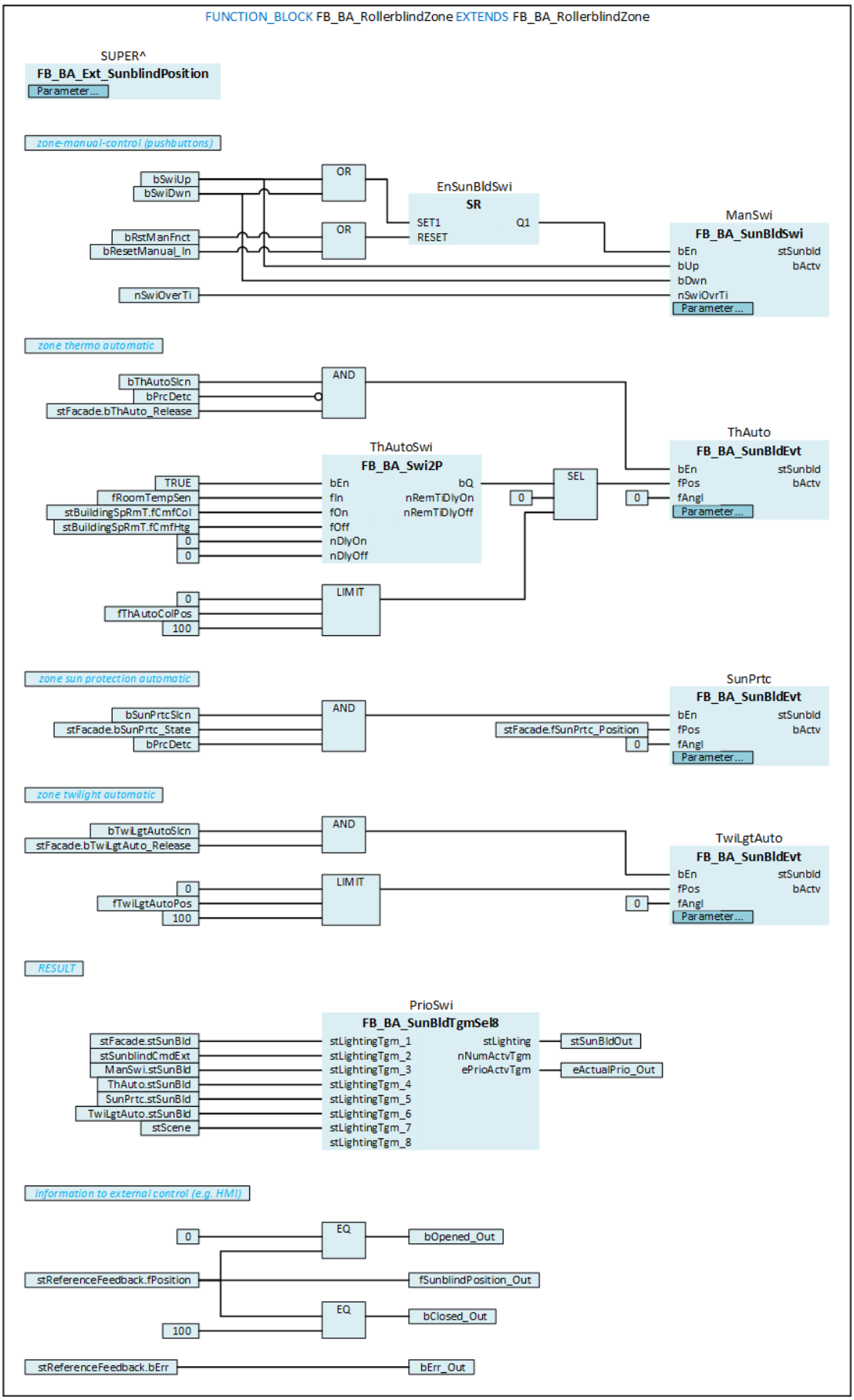
Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_SunblindPosition* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

- **bResetManual_In**: Befehl von der HMI, die Handfunktion abzulöschen.
- **fSunblindPosition_Out**: Ausgabeinformation Position an die HMI.
- **bOpened_Out**: Ausgabeinformation "Jalousie komplett geöffnet" an die HMI.
- **bClosed_Out**: Ausgabeinformation "Jalousie komplett geschlossen" an die HMI.
- **bErr_Out**: Ausgabeinformation "Referenzaktor in Störung" an die HMI.
- **eActualPrio_Out**: Aktuell steuernde Telegrammpriorität an die HMI.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_RollerblindZone EXTENDS FB_BA_Ext_SunblindPosition
VAR_INPUT
  fRoomTempSen           : REAL;
  bPrcDetc               : BOOL;
  bSwiUp                 : BOOL;
  bSwiDwn                : BOOL;
  bRstManFnct            : BOOL;
  stBuildingSpRmT        : ST_BA_SpRmT;
  stFacade               : ST_BA_Facade;
  stScene                : ST_BA_Sunbld;
  stReferenceFeedback    : ST_BA_SunblindActuatorFeedback;
END_VAR
VAR_OUTPUT
  stSunBldOut            : ST_BA_SunBld;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bThAutoSlcn            : BOOL;
  fThAutoColPos          : REAL;
  bSunPrtcSlcn           : BOOL;
  bTwiLgtAutoSlcn        : BOOL;
  fTwiLgtAutoPos         : REAL;
  nSwiOverTi             : UDINT;
END_VAR
VAR_INPUT CONSTANT
  ManSwi                 : FB_BA_SunBldSwi := (ePrio:= E_BA_SunBldPrio.eManualGroup);
  ThAuto                 : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupThermoAuto);
  SunPrtc                : FB_BA_SunBldEvt : = (ePrio:= E_BA_SunBldPrio.eSunProtection);
  TwiLgtAuto             : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupTwiLightAuto);
END_VAR
VAR
  ThAutoSwi              : FB_BA_Swi2P;
  PrioSwi                : FB_BA_SunBldTgmSel8;
  EnSunBldSwi            : SR;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fRoomTempSen	REAL	Raumtemperatursensor [°C].
bPrcDetc	BOOL	Präsenzerkennung.
bSwiUp	BOOL	Tastenfunktion Jalousie "auf".
bSwiDwn	BOOL	Tastenfunktion Jalousie "ab".
bRstManFnct	BOOL	Ablöscheingang für den Jalousie-Handmerker.
stBuildingSpRmT	ST_BA_SpRmT [▶ 277]	Struktur der Raumsollwerte (Protection Cooling … Comfort Cooling und Protection Heating … Comfort Heating).
stFacade	ST_BA_Facade [▶ 717]	Struktur der Fassadendaten (Fassadentelegramme, Freigaben für Dämmerungs- und Thermoautomatik, etc).
stScene	ST_BA_Sunbld [▶ 279]	Reservierter Telegramm-Eingang für eine Szenenansteuerung.
stReferenceFeedback	ST_BA_SunblindActuatorFeedback [▶ 714]	Rückmeldeeingang des angesteuerten Jalousieaktors bzw. des Referenzaktors der angesteuerten Gruppe.

Ausgänge

Name	Typ	Beschreibung
stSunBldOut	ST_BA_Sunbld [▶ 279]	Resultierendes Zonentelegramm.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bThAutoSlcn	BOOL	Thermoautomatik Freigabe.
fThAutoColPos	REAL	Thermoautomatik Kühlposition [%].
bSunPrtcSlcn	BOOL	Verschattungsautomatik Freigabe.
bTwiLgtAutoSlcn	BOOL	Dämmerungsautomatik Freigabe.
fTwiLgtAutoPos	REAL	Dämmerungsautomatik Kühlposition [%].
nSwiOverTi	UDINT	Taster-Funktion der Zone: Umschaltzeit [ms] vom Tipp- bis zum Selbsthaltungsbetrieb.

Eingänge CONSTANT

Name	Typ	Beschreibung
ManSwi	FB_BA_SunBldSwi [▶ 398]	Gruppen- (Zonen-) Tasterbaustein.
ThAuto	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Gruppen- (Zonen-) Thermoautomatik.
SunPrtc	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Gruppen- (Zonen-) Verschattungsautomatik.
TwiLgtAuto	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Gruppen- (Zonen-) Dämmerungsautomatik.

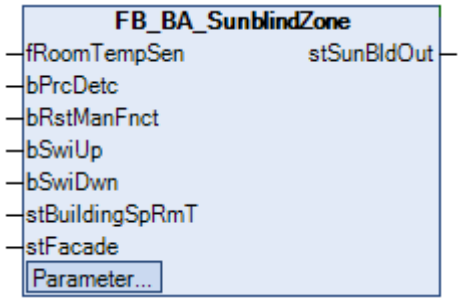
Variablen

Name	Typ	Beschreibung
ThAutoSwi	FB_BA_Swi2P	Hysteresebaustein zur Umschaltung Heizen/Kühlen für die Gruppen- (Zonen-) Thermoautomatik.
PrioSwi	FB_BA_SunBldTgmSel8 [▶ 401]	Auswahl resultierendes Telegramm.
EnSunBldSwi	SR	Umschaltung Freigabe/Reset.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.6.1.3.2 FB_BA_SunblindZone



In einer Jalousie-Zone werden ein oder mehrere Jalousien zur gleichzeitigen Ansteuerung zusammengefasst. Das Template *FB_BA_SunblindZone* bündelt dabei die vorgefertigten Daten der Fassaden und entscheidet aufgrund einer Zonenpräsenzerkennung und örtlicher An- und Abwahl, welche Funktionalität dabei aktiv ist.

Zusammen mit dem resultierenden Telegramm der hoch-priorisierten Funktionen aus der Fassade werden diese Funktionalitäten am Ende des Templates auf einen Telegrammselektor gelegt. Dieser entscheidet dann anhand der Priorität, welches Telegramm im übergeordneten Raumtemplate weiter an die Jalousien gereicht wird.

Funktionen

- **Handfunktion**
  Das Zonentemplate enthält eine Handfunktion, die das manuelle Ansteuern der Jalousie über die Tastenfunktionen *bSwiUp/bSwiDwn* ermöglicht. Die Umschaltzeit auf Selbsthaltung, *nSwiOverTi* [ms], ist auf 250 ms vorparametriert. Über den Eingang *bRstManFnct* wird die Handfunktion abgelöscht.
- **Thermoautomatik**
  Die Thermoautomatik gilt als aktiv, wenn sie lokal angewählt ist (Parameter *bThAutoSlcn*), wenn keine Präsenz erkannt wird (Eingang *bPrcDetc*) und das Fassadentemplate FB_BA_Facade [▶ 956] die Freigabe für diese Gebäudeseite ermittelt hat. Ist sie aktiv, wird anhand der Raumtemperatur (Eingang *fRoomTempSen*) und den Gebäudesollwerten für Heizen und Kühlen über die Positionierung entschieden: Liegt die Raumtemperatur über dem Gebäudewert für Komfort-Kühlen, so fahren die Jalousien auf eine vordefinierbare Position. Sinkt die Temperatur hingegen unter den Wert für Komfort-Heizen, so fahren die Jalousien komplett auf.
- **Sonnenschutzautomatik**
  Die Sonnenschutzautomatik ist im Gegensatz zur Thermoautomatik nur aktiv, wenn Präsenz erkannt wird (Eingang *bPrcDetc*). Auch sie muss lokal angewählt sein (Parameter *bSunPrtcSlcn*) und vom Fassadentemplate die Freigabe bekommen, unter anderem durch die richtungsabhängige Helligkeit. Die Jalousieposition und der Winkel werden ebenfalls in der Fassade ermittelt.
- **Dämmerungsautomatik**
  Die Freigabe der Dämmerungsautomatik erfolgt in der Fassade aufgrund der Helligkeitswerte. Ist die Automatik für die Zone angewählt (Eingang *bTwiLgtAutoSlcn*) so fährt die Jalousie im Dämmerungsfall auf eine vordefinierte Position (Parameter *fTwiLgtAutoPos/fTwiLgtAutoAngl*).

Eingang stReferenceFeedback

Über diesen Eingang werden Informationen über den angesteuerten Jalousieaktor bzw. den Referenzaktor einer Gruppe in die Jalousiesteuerfunktion zurückgeführt.

Hierbei handelt es sich um Positionsangaben und Status des Referenzaktors.

Datenaustausch HMI

Der Datenaustausch mit der HMI wird hier in der Basisklasse *FB_BA_Ext_SunblindAngle* (interner Baustein) realisiert. In diesem Template sichtbar ist die Verwendung folgender Variablen:

- **bResetManual_In**: Befehl von der HMI, die Handfunktion abzulöschen.
- **fSunblindPosition_Out**: Ausgabeinformation Position an die HMI.
- **fSunblindAngle_Degree**: Ausgabeinformation Winkel in Grad an die HMI.
- **bOpened_Out**: Ausgabeinformation "Jalousie komplett geöffnet" an die HMI.
- **bClosed_Out**: Ausgabeinformation "Jalousie komplett geschlossen" an die HMI.
- **bErr_Out**: Ausgabeinformation "Referenzaktor in Störung" an die HMI.
- **eActualPrio_Out**: Aktuell steuernde Telegrammpriorität an die HMI.
- **fAnglLmtUp:** Oberer Lamellenwinkel des Referenzaktors.
- **fAnglLmtDwn:** Unterer Lamellenwinkel des Referenzaktors.

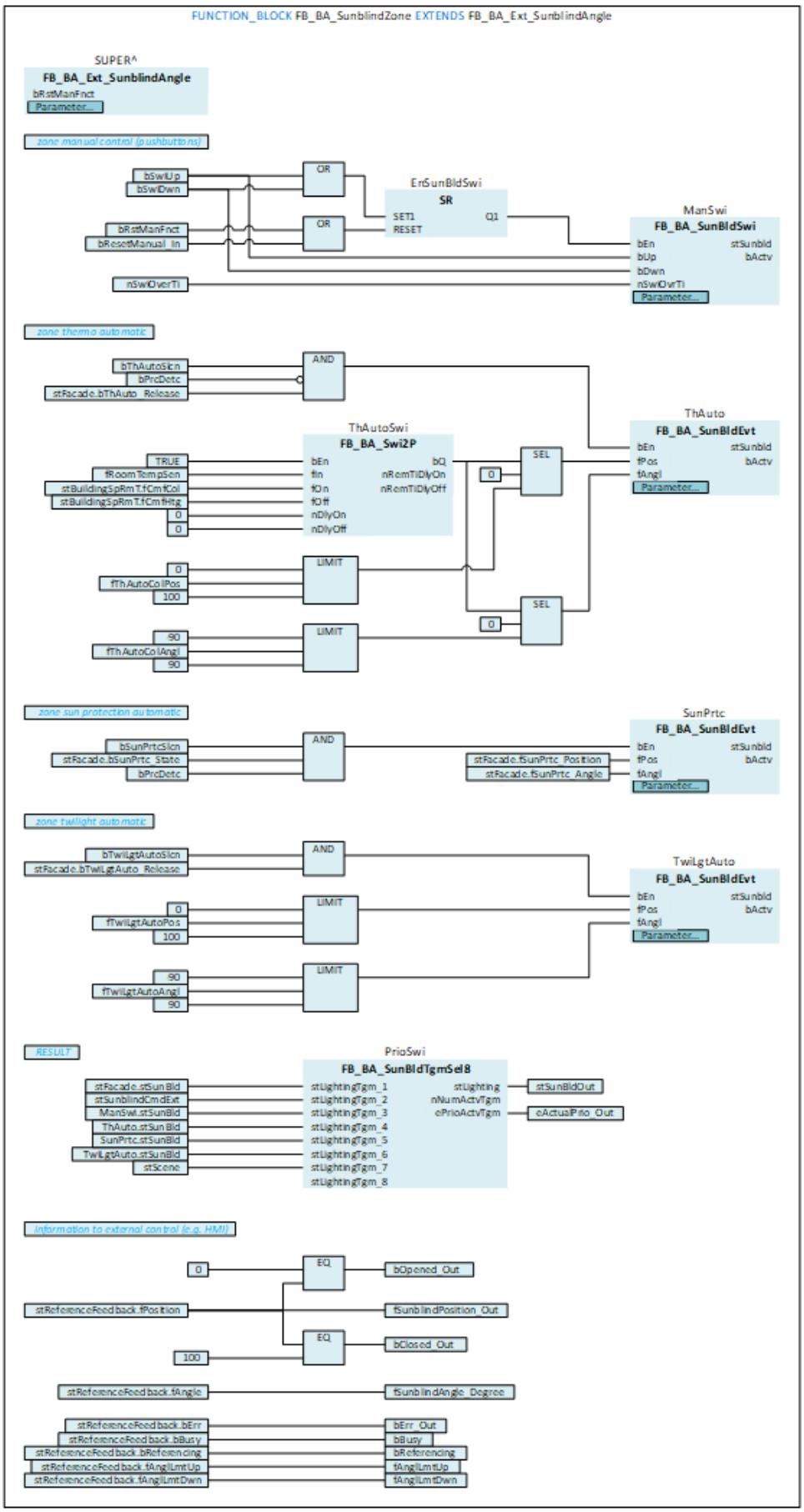


Über die HMI werden die anzusteuernden Lamellenwinkel in 0% (Lamelle an oberer Grenzposition *fAnglLmtUp*) bis 100% (Lamelle an unterer Grenzposition *fAnglLmtDwn*) angegeben, wobei 50% die Nullstellung (0° = Lamelle waagerecht) ist.

Es ist darauf zu achten, dass an dieser Stelle der Obere Lamellenwinkel größer 0° und der untere Lamellenwinkel kleiner 0° ist, um die Berechnung richtig ausführen zu können.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_SunblindZone EXTENDS FB_BA_Ext_SunblindAngle
VAR_INPUT
  fRoomTempSen           : REAL;
  bPrcDetc               : BOOL;
  bSwiUp                 : BOOL;
  bSwiDwn                : BOOL;
  bRstManFnct            : BOOL;
  stBuildingSpRmT        : ST_BA_SpRmT;
  stFacade               : ST_BA_Facade;
  stScene                : ST_BA_Sunbld;
  stReferenceFeedback    : ST_BA_SunblindActuatorFeedback;
END_VAR
VAR_OUTPUT
  stSunBldOut            : ST_BA_SunBld;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bThAutoSlcn            : BOOL;
  fThAutoColPos          : REAL;
  fThAutoColAngl         : REAL;
  bSunPrtcSlcn           : BOOL;
  bTwiLgtAutoSlcn        : BOOL;
  fTwiLgtAutoPos         : REAL;
  fTwiLgtAutoAngl        : REAL;
  nSwiOverTi             : UDINT;
END_VAR
VAR_INPUT CONSTANT
  ManSwi                 : FB_BA_SunBldSwi := (ePrio:= E_BA_SunBldPrio.eManualGroup);
  ThAuto                 : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupThermoAuto);
  SunPrtc                : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eSunProtection);
  TwiLgtAuto             : FB_BA_SunBldEvt := (ePrio:= E_BA_SunBldPrio.eGroupTwiLightAuto);

END_VAR
VAR
  ThAutoSwi              : FB_BA_Swi2P;
  PrioSwi                : FB_BA_SunBldTgmSel8;
  EnSunBldSwi            : SR;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fRoomtempSen	REAL	Raumtemperarursensor [°C].
bPrcDetc	BOOL	Präsenzerkennung.
bSwiUp	BOOL	Lokaler Taster "auf".
bSwiDwn	BOOL	Lokaler Taster "ab".
bRstManFnct	BOOL	Eingang zum Rücksetzen aller internen Handfunktionen, sowohl die über die Eingänge *bSwiUp/bSwiDwn*, als auch der externen Ansteuerung (z.B. HMI).
stBuildingSpRmT	ST_BA_SpRmT [▶ 277]	Struktur der Raumsollwerte (Protection Cooling..Comfort Cooling und Protection Heating .. Comfort Heating).
stFacade	ST_BA_Facade [▶ 717]	Fassadenspezifische Jalousiedaten und -telegramme.
stScene	ST_BA_Sunbld [▶ 279]	Reservierter Telegramm-Eingang für eine Szenenansteuerung.
stReferenceFeedback	ST_BA_SunblindActuatorFeedback [▶ 714]	Rückmeldeeingang des angesteuerten Jalousieaktors bzw. des Referenzaktors der angesteuerten Gruppe.

Ausgänge

Name	Typ	Beschreibung
stSunBldOut	ST_BA_SunBld [▶ 279]	resultierendes Zonentelegramm.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bThAutoSlcn	BOOL	Thermo-Automatik Freigabe.
fThAutoColPos	REAL	Thermo-Automatik Kühl-Position [%].
fThAutoColAngl	REAL	Thermo-Automatik Kühl-Winkel [°].
bSunPrtcSlcn	BOOL	Verschattungs-Automatik Freigabe.
bTwiLgtAutoSlcn	BOOL	Dämmerungs-Automatik Freigabe.
fTwiLgtAutoPos	REAL	Dämmerungs-Automatik Position [%].
fTwiLgtAutoAngl	REAL	Dämmerungs-Automatik Position [°].
nSwiOverTi	UDINT	Umschaltzeit [ms] des Handschalters *SunBldSwi* zur Selbsthaltung.

Eingänge CONSTANT

Name	Typ	Beschreibung
ManSwi	FB_BA_SunBldSwi [▶ 398]	Gruppen- (Zonen-) Tasterbaustein.
ThAuto	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Gruppen- (Zonen-) Thermoautomatik.
SunPrtc	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Gruppen- (Zonen-) Verschattungsautomatik.
TwiLgtAuto	FB_BA_SunBldEvt [▶ 388]	Telegrammbaustein für die Gruppen- (Zonen-) Dämmerungsautomatik.

VAR

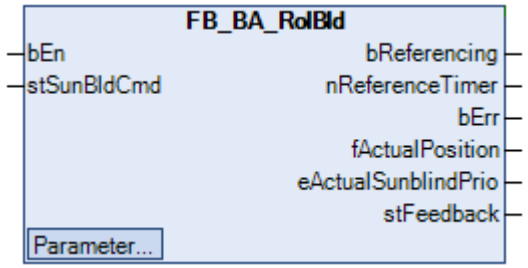
Name	Typ	Beschreibung
ThAutoSwi	FB_BA_Swi2P	Hysteresebaustein zur Umschaltung Heizen/Kühlen für die Gruppen- (Zonen-) Thermoautomatik.
PrioSwi	FB_BA_SunBldTgmSel8 [▶ 401]	Auswahl des resultierenden Telegramms.
EnSunBldSwi	SR	Aktivierungsspeicher des oben genannten Bausteins.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.6.2 Actuators

6.1.4.2.1.4.6.2.1 FB_BA_RolBld



Dieses Template dient zur Ansteuerung eines Jalousie-Aktors ohne Lamellenverstellung.

Das resultierende Diagramm aus der Zone, *stSunBldCmd*, wird über einen Ansteuerungs-Verzögerer FB_BA_SunBldPosDly [▶ 391] geführt.

Dieser verzögert alle Automatik-Telegramme und soll sicherstellen, dass im Falle einer globalen Ansteuerung von Jalousien (z. B. Feueralarm) nicht alle Jalousien gleichzeitig fahren und somit der Anzugsstrom der Motoren begrenzt bleibt.

Über ein Analogobjekt FB_BA_AV_Op [▶ 207] wird die aktuelle Jalousieposition im BACnet zur Verfügung gestellt.

Das Template besitzt am Ausgang eine Rückmeldestruktur stFeedback [▶ 714].

Ist der programmierte Jalousieaktor ein Einzelaktor oder der Referenzaktor einer Gruppe, so ist dieses Rückmeldetelegramm an die ansteuernde Raum- (Zonen-) Anwenderfunktion zu verknüpfen, um auf diesem Wege Informationen über den Zustand des Aktors zu erlangen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_RolBld EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  stSunBldCmd           : ST_BA_SunBld;
END_VAR
VAR_OUTPUT
  bReferencing          : BOOL;
  nReferenceTimer       : UDINT;
  bErr                  : BOOL;
  fActualPosition       : REAL;
  eActualSunblindPrio   : BYTE;
  stFeedback            : ST_BA_SunblindActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiOverTi            : UDINT;
  nPositioningDelay     : UDINT;
  nTiUp                 : UDINT;
  nTiDwn                : UDINT;
END_VAR
VAR_INPUT CONSTANT
  ActualPosition        : FB_BA_AV_Op;
END_VAR
VAR
  PositioningDelay      : FB_BA_SunBldPosDly;
  RolBldActr            : FB_BA_RolBldActr;
  bCmdUp      AT %Q*    : BOOL;
  bCmdDown    AT %Q*    : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe der Bausteinfunktion.
stSunBldCmd	ST_BA_SunBld [▶ 279]	Resultierendes Telegramm aus der übergeordneten Zone (Raum).

Ausgänge

Name	Typ	Beschreibung
bReferencing	BOOL	Jalousie wird gerade referenziert.
nReferenceTimer	UDINT	Ablaufende Referenzierzeit [s].
bErr	BOOL	Der interne Baustein *FB_BA_RolBldActr* ist falsch parametriert.
fActualPosition	REAL	Aktuelle Position (errechnet).
eActualSunblindPrio	BYTE	Aktuelle Priorität, mit dem der Jalousieaktor angesteuert wird.
stFeedback	ST_BA_SunblindActuatorFeedback [▶ 714]	Rückmeldetelegramm zur Verknüpfung an die ansteuernde Raum- (Zonen-) Anwenderfunktion. Auf diesem Wege gelangen Informationen über den Zustand des Jalousieaktors zurück in die Anwenderfunktion.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nSwiOverTi	UDINT	Umschaltzeit [ms] des Handschalters *SunBldSwi* zur Selbsthaltung.
nPositioningDelay	UDINT	Übermittlungsverzögerung der Zonen-Telegramme [ms] um Anzugsströme zeitlich zu verteilen und dadurch zu begrenzen.
nTiUp	UDINT	Komplette Hochfahrzeit [ms].
nTiDwn	UDINT	Komplette Zeit zum Herunterfahren [ms].

Eingänge CONSTANT

Name	Typ	Beschreibung
ActualPosition	FB_BA_AV_Op [▶ 207]	Baustein zur Darstellung der Position [%] im BACnet.

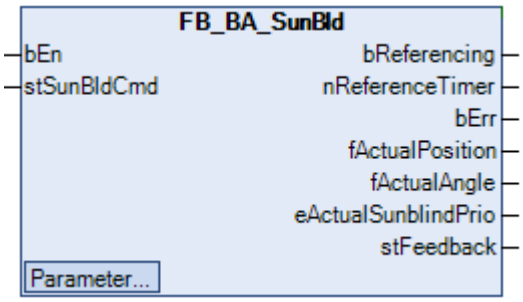
Variablen

Name	Typ	Beschreibung
PositioningDelay	FB_BA_SunBldPosDly [▶ 391]	Verzögerung der Telegramme aus der Zone (Gruppe) um hohe Anzugsströme durch Gleichzeitigkeit zu verhindern.
RolBldActr	FB_BA_RolBldActr [▶ 373]	Baustein zur Ansteuerung einer Jalousie ohne Lamellenverstellung.
bCmdUp	BOOL	Ausgangsvariable Befehl „hoch“.
bCmdDown	BOOL	Ausgangsvariable Befehl „runter“.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.6.2.2 FB_BA_SunBld



Dieses Template dient zur Ansteuerung eines Jalousie-Aktors mit Lamellenverstellung.

Das resultierende Diagramm aus der Zone, *stSunBldCmd*, wird über einen Ansteuerungs-Verzögerer FB_BA_SunBldPosDly [▶ 391] geführt.

Dieser verzögert alle Automatik-Telegramme und soll sicherstellen, dass im Falle einer globalen Ansteuerung von Jalousien (z. B. Feueralarm) nicht alle Jalousien gleichzeitig fahren und somit der Anzugsstrom der Motoren begrenzt bleibt.

Über zwei Analogobjekte FB_BA_AV_Op [▶ 207] werden die aktuelle Jalousieposition sowie der aktuelle Winkel im BACnet zur Verfügung gestellt.

Das Template besitzt am Ausgang eine Rückmeldestruktur stFeedback [▶ 714].

Ist der programmierte Jalousieaktor ein Einzelaktor oder der Referenzaktor einer Gruppe, so ist dieses Rückmeldetelegramm an die ansteuernde Raum- (Zonen-) Anwenderfunktion zu verknüpfen, um auf diesem Wege Informationen über den Zustand des Aktors zu erlangen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_SunBld EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  stSunBldCmd           : ST_BA_SunBld;
END_VAR
VAR_OUTPUT
  bReferencing          : BOOL;
  nReferenceTimer       : UDINT;
  bErr                  : BOOL;
  fActualPosition       : REAL;
  fActualAngle          : REAL;
  eActualSunblindPrio   : BYTE;
  stFeedback            : ST_BA_SunblindActuatorFeedback;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nSwiOverTi            : UDINT;
  nPositioningDelay     : UDINT;
  nTiUp                 : UDINT;
  nTiDwn                : UDINT;
  nTurnTiUp             : UDINT;
  nTurnTiDwn            : UDINT;
  nBckLshTiUp           : UDINT;
  nBckLshTiDwn          : UDINT;
  fAnglLmtUp            : REAL;
  fAnglLmtDwn           : REAL;
END_VAR
VAR_INPUT CONSTANT
  ActualPosition        : FB_BA_AV_Op;
  ActualAngle           : FB_BA_AV_Op;
END_VAR
VAR
  PositioningDly        : FB_BA_SunBldPosDly;
  SunBldActr            : FB_BA_SunBldActr;
```

```
  bCmdUp      AT %Q*     : BOOL;
  bCmdDown    AT %Q*     : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe der Bausteinfunktion.
stSunBldCmd	ST_BA_SunBld [▶ 279]	Resultierendes Telegramm aus der übergeordneten Zone (Raum).

Ausgänge

Name	Typ	Beschreibung
bReferencing	BOOL	Jalousie wird gerade referenziert.
nReferenceTimer	UDINT	Ablaufende Referenzierzeit [s].
bErr	BOOL	Der interne Baustein *FB_BA_SunBldActr* ist falsch parametriert.
fActualPosition	REAL	Aktuelle Position (errechnet).
fActualAngle	REAL	Aktueller Winkel (errechnet).
eActualSunblindPrio	BYTE	Aktuelle Priorität, mit dem der Jalousieaktor angesteuert wird.
stFeedback	ST_BA_SunblindActuatorFeedback [▶ 714]	Rückmeldetelegramm zur Verknüpfung an die ansteuernde Raum- (Zonen-) Anwenderfunktion. Auf diesem Wege gelangen Informationen über den Zustand des Jalousieaktors zurück in die Anwenderfunktion.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nSwiOverTi	UDINT	Umschaltzeit [ms] des Handschalters *SunBldSwi* zur Selbsthaltung.
nPositioningDelay	UDINT	Übermittlungsverzögerung der Zonen-Telegramme [ms] um Anzugsströme zeitlich zu verteilen und dadurch zu begrenzen.
nTiUp	UDINT	Komplette Hochfahrzeit [ms].
nTiDwn	UDINT	Komplette Zeit zum Herunterfahren [ms].
nTurnTiUp	UDINT	Zeit zum Umschwenken der Lamellen in obere Richtung [ms].
nTurnTiDwn	UDINT	Zeit zum Umschwenken der Lamellen in untere Richtung [ms].
nBckLshTiUp	UDINT	Zeit zum Ausfahren der Umkehrlose in obere Richtung [ms].
nBckLshTiDwn	UDINT	Zeit zum Ausfahren der Umkehrlose in untere Richtung [ms].
fAnglLmtUp	REAL	Höchste Stellung der Lamellen [°].
fAnglLmtDwn	REAL	Niedrigste Stellung der Lamellen [°]. Diese Stellung ist erreicht, wenn die Jalousie ganz heruntergefahren ist.

Eingänge CONSTANT

Name	Typ	Beschreibung
ActualPosition	FB_BA_AV_Op [▶ 207]	Baustein zur Darstellung der Position [%] im BACnet.
ActualAngle	FB_BA_AV_Op [▶ 207]	Baustein zur Darstellung des Winkels [°] im BACnet.

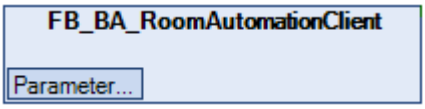
Variablen

Name	Typ	Beschreibung
PositioningDelay	FB_BA_SunBldPosDly [▶ 391]	Verzögerung der Telegramme aus der Zone (Gruppe) um hohe Anzugsströme durch Gleichzeitigkeit zu verhindern.
SunBldActr	FB_BA_SunBldActr [▶ 383]	Baustein zur Ansteuerung einer Jalousie.
bCmdUp	BOOL	Ausgangsvariable Befehl „hoch“.
bCmdDown	BOOL	Ausgangsvariable Befehl „runter“.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.4.7 FB_BA_RoomAutomationClient



Aufruftemplate Gewerk "Raumautomation" - Client-Ausführung.

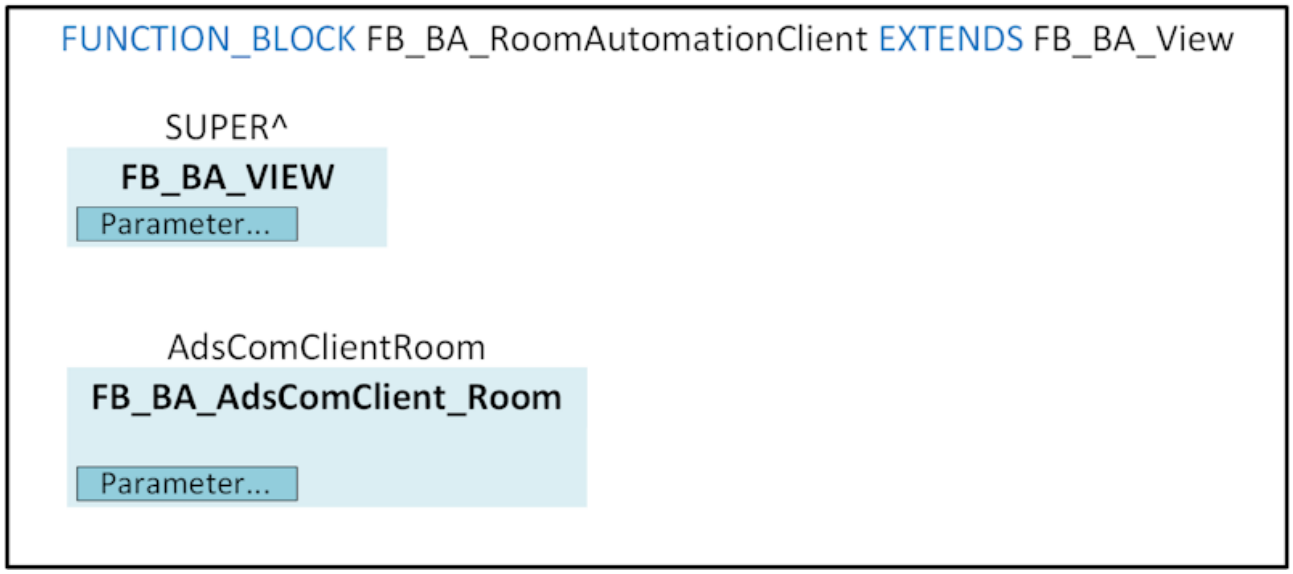
Dieses Template auf der Ebene "Gewerk" ruft alle Untertemplates auf, deren Daten idealerweise in einem Etagenrechner aufbereitet werden.

Das sind zum einen dem Gewerk "Raumautomation" zuzuordnende Daten, welche von einem Gebäuderechner übermittelt, gelesen werden (FB_BA_AdsComClient_Room [▶ 901]), zum anderen befinden sich in diesem Template die Aufrufe aller Räume.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_RoomAutomationClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComClientRoom    : FB_BA_AdsComClient_Room;
  SampleRoom          : FB_BA_RoomSample1;
END_VAR
```

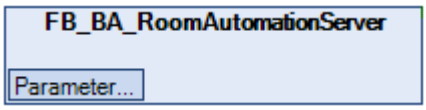
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
AdsComClientRoom	FB_BA_AdsComClient_Room [▶ 901]	Das Untertemplate hat die Aufgabe Daten und Parameter zu lesen, welche von anderen Rechnern, idealerweise vom Gebäuderechner, gesendet wurden.
SampleRoom	FB_BA_RoomSample1 [▶ 947]	Das Untertemplate stellt einen Beispielraum mit Verschattung, Beleuchtung und Klimatisierung dar. Dieser Raum hat einen Verschattungsbereich innerhalb einer Fassade, dessen Daten er über *site.arrFacade[2]* empfängt. Beleuchtungstechnisch ist der Raum dem Bereich 1 zugeordnet, was sich durch den Empfang von *site.arrAreaLighting[1]* ausdrückt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.4.8 FB_BA_RoomAutomationServer



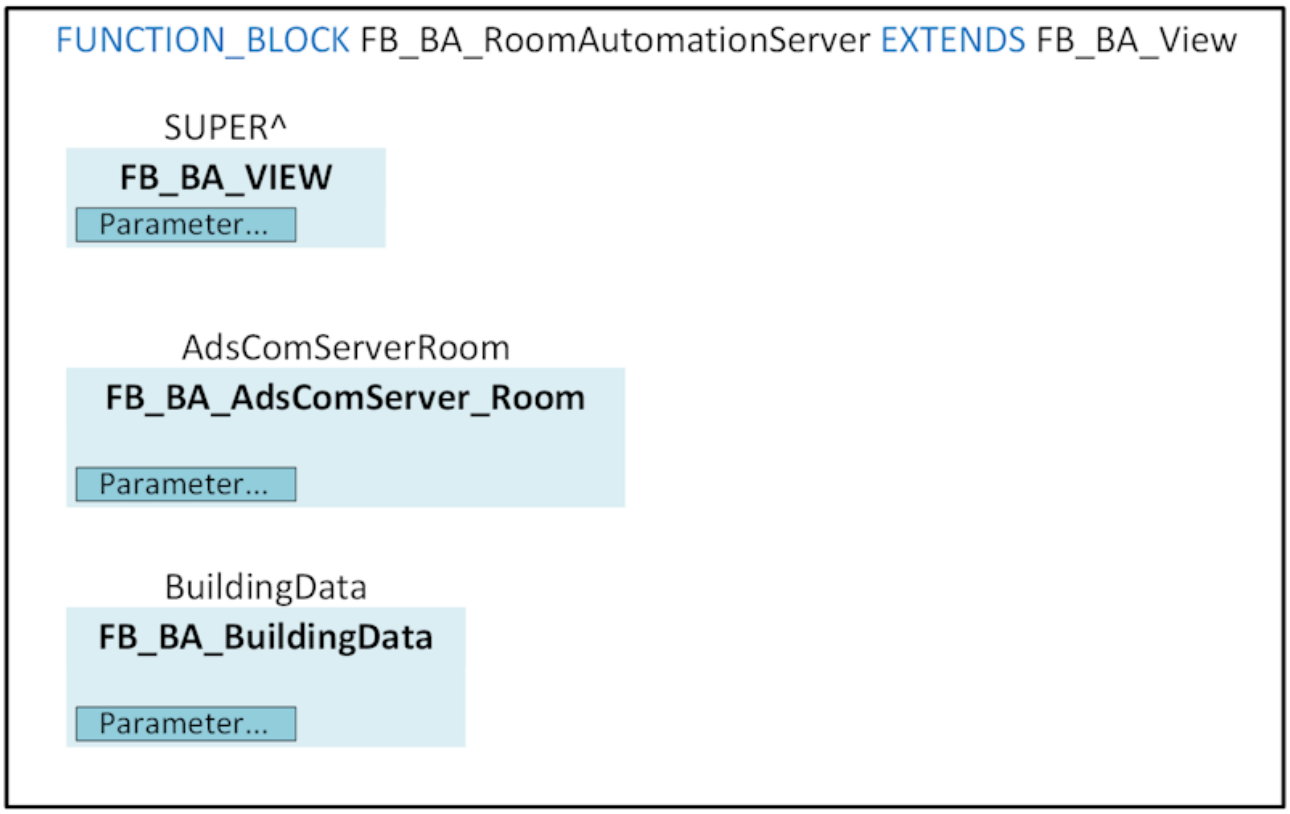
Aufruftemplate Gewerk "Raumautomation" - Server-Ausführung.

Dieses Template auf der Ebene "Gewerk" ruft alle Untertemplates auf, deren Daten idealerweise in einem Gebäuderechner aufbereitet werden.

Darüber hinaus werden durch ein ADS-Server-Template raumautomations-spezifische Daten gebäudeweit zur Verfügung gestellt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_RoomAutomationServer EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComServerRoom    : FB_BA_AdsComServer_Room;
  BuildingData        : FB_BA_BuildingData;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
AdsComServerRoom	FB_BA_AdsComServer_Room [▶ 902]	Das Untertemplate hat die Aufgabe Daten und Parameter innerhalb des GA-Netzwerkes zu übermitteln.
BuildingData	FB_BA_BuildingData [▶ 872]	Das Untertemplate ruft weitere Templates auf, welche gebäudeweite Daten und Betriebsarten zur Verfügung stellen.

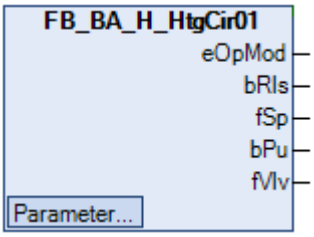
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.5 HeatingSystem

6.1.4.2.1.5.1 Distribution

6.1.4.2.1.5.1.1 FB_BA_H_HtgCir01



Template dient der Programmierung eines statischen Heizkreises.

Die wesentlichen Bestandteile des Templates sind:

- Vorlauftemperaturregelung.
- Heizkurve außentemperaturabhängig mit Nachtabsenkung.
- Betriebsartenauswahl.
- Steuerung der Heizkreispumpe.
- Ansteuerung eines stetigen Regelventils.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_H_HtgCir01 EXTENDS FB_BA_View
VAR_OUTPUT
  eOpMod       : E_BA_EnergyLvl;
  bRls         : BOOL;
  fSp          : REAL;
  bPu          : BOOL;
  fVlv         : REAL;
END_VAR
VAR_INPUT CONSTANT
  TFl          : FB_BA_SensorAnalog_Raw;
  TRt          : FB_BA_SensorAnalog_Raw;
  Vlv          : FB_BA_Vlv;
  Pu           : FB_BA_Pulst;
  Sp           : FB_BA_H_HtgCir_Sp;
  OpMod        : FB_BA_H_OpMod;
  HtgLmt       : FB_BA_HtgLmt;
  TFlCtrl      : FB_BA_PID;
  PlantLock    : FB_BA_PlantLock;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
eOpMod	E_BA_EnergyLvl [▶ 713]	Betriebsart des Heizkreises.
bRls	BOOL	Die Variable zeigt an, dass der Heizkreis in Betrieb ist.
fSp	REAL	Errechneter Sollwert der Heizkennlinie.
bPu	BOOL	Freigabe der Heizkreispumpe.
fVlv	REAL	Errechneter Stellwert für das Ventil.

Eingänge CONSTANT

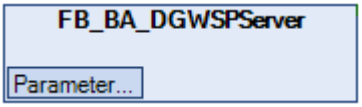
Name	Typ	Beschreibung
TFl	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert die Vorlauftemperatur.
TRt	FB_BA_SensorAnalog [▶ 1106]	Der Funktionsbaustein repräsentiert die Rücklauftemperatur.
Vlv	FB_BA_Vlv [▶ 1111]	Regelventil
Pu	FB_BA_Pu1st [▶ 1093]	Heizkreispumpe
Sp	FB_BA_H_HtgCir_Sp	Der Funktionsbaustein berechnet den Sollwert der Vorlauftemperatur in Abhängigkeit der Außentemperatur.
OpMod	FB_BA_H_OpMod	Betriebsartenauswahl des Heizkreises (Tag-, Nacht-, Schutzbetrieb).
HtgLmt	FB_BA_HtgLmt	Der Funktionsbaustein gibt den Heizbetrieb unterhalb einer Heizgrenztemperatur frei.
TFlCtrl	FB_BA_PID [▶ 1055]	Rücklauftemperaturfühler
PlantLock	FB_BA_PlantLock [▶ 138]	Mit dem Aufruf des Funktionsbausteins werden auf dieser und tiefer gelegenen Ebenen des Heizkreises die relevanten Störungen der ereignisfähigen Objekte gesammelt und ausgegeben. Diese relevanten Störungen lösen die Betriebsart „Störung“ aus. Der Heizkreis wird damit in die Betriebsart „Protection“ geschaltet.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.1.6 DoorsGatesWindowsSunProtection

6.1.4.2.1.6.1 FB_BA_DGWSPServer



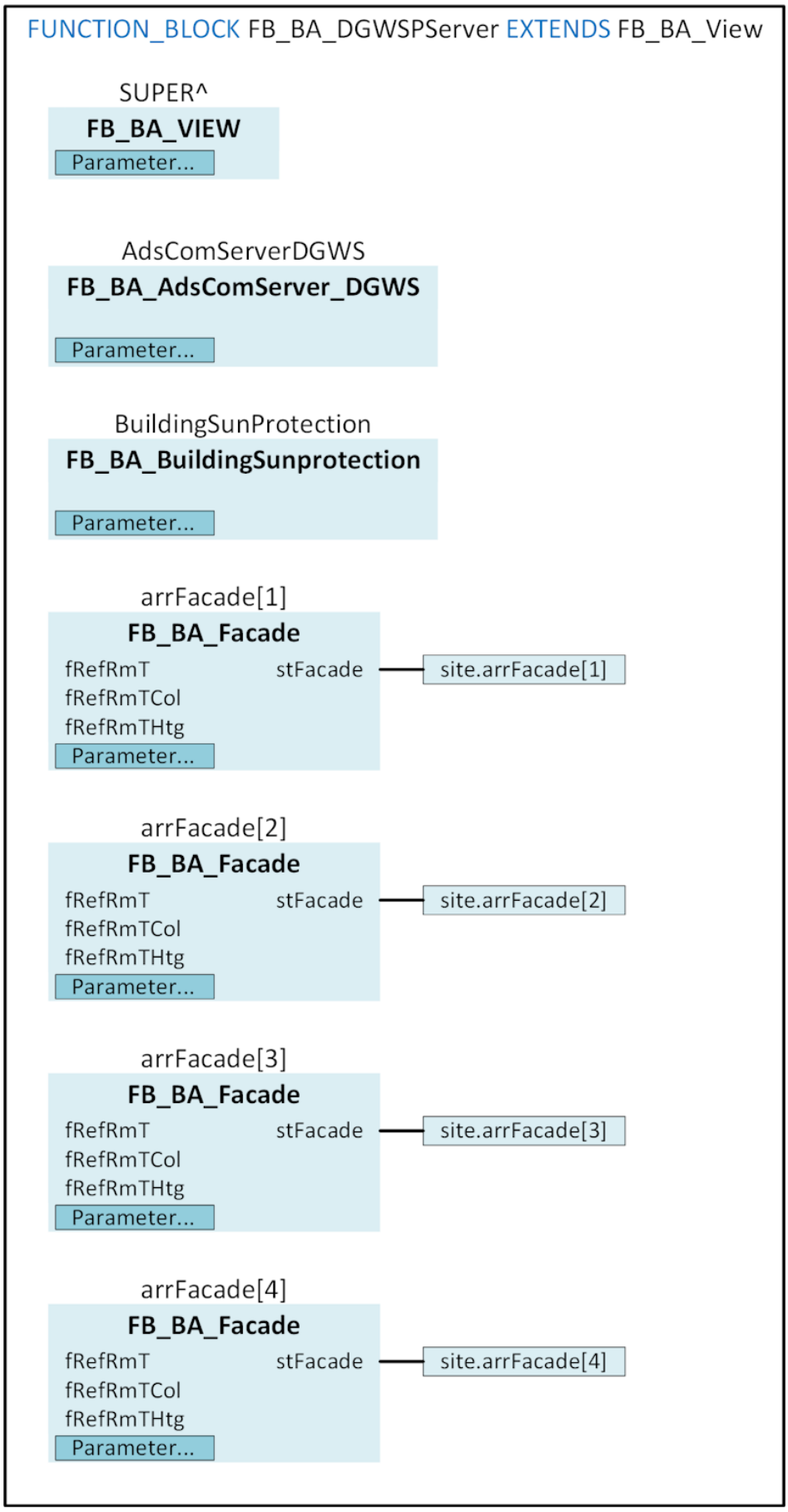
Aufruftemplate Gewerk "Türe, Tore, Fenster, Sonnenschutz" - Server-Ausführung.

Dieses Template auf der Ebene "Gewerk" ruft alle Untertemplates auf, deren Daten idealerweise in einem Gebäuderechner aufbereitet werden. Das sind zum einen allgemeine Daten des Sonnenschutzes, sowie die Daten aller Fassaden.

Darüber hinaus werden durch ein ADS-Server-Template gewerkespezifische Daten gebäudeweit zur Verfügung gestellt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_DGWSPServer EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComServerDGWS          : FB_BA_AdsComServer_DGWS;
  BuildingSunProtection     : FB_BA_BuildingSunprotection;
  arrFacade                 : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_Facade;
END_VAR
```

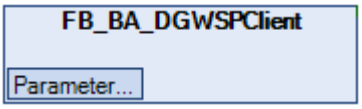
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
AdsComServerDGWS	FB_BA_AdsComServer_DGWS [▶ 998]	Das Untertemplate hat die Aufgabe Daten und Parameter innerhalb des GA-Netzwerkes zu übermitteln.
BuildingSunProtection	FB_BA_BuildingSunprotection [▶ 953]	Das Untertemplate stellt gebäudeweite Daten und Betriebsarten zur Verfügung.
arrFacade	FB_BA_Facade [▶ 956]	Diese Templates von Typ *FB_BA_Facade* stellen die Sonnenschutztelegramme zusammen, welche jeweils für eine ganzen Fassade gültig sind. In der globalen Parameterliste BA2_Param [▶ 1137] sind mit *nMaxNumberOfFacades* 4 Fassaden vordefiniert, welche hier im *FB_Init* exemplarisch auf Norden, Osten, Süden und Westen initialisiert sind.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.6.2 FB_BA_DGWSPClient

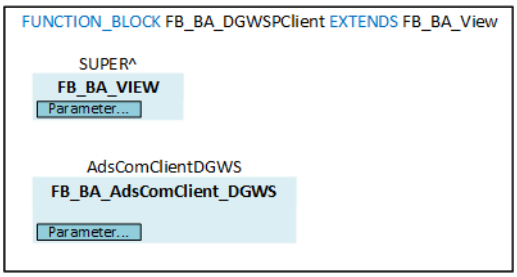


Aufruftemplate Gewerk "Türe, Tore, Fenster, Sonnenschutz" - Client-Ausführung.

Dieses Template auf der Ebene "Gewerk" ruft alle Untertemplates auf, deren Daten idealerweise in einem Etagenrechner aufbereitet werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_DGWSPClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  AdsComClientDGWS    : FB_BA_AdsComClient_DGWS;
END_VAR
```

VAR_INPUT CONSTANT

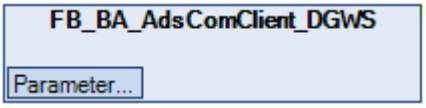
Name	Typ	Beschreibung
AdsComClientDGWS	FB_BA_AdsComClient_DGWS [▶ 997]	Das Untertemplate hat die Aufgabe Daten und Parameter zu lesen, welche von anderen Rechnern, idealerweise vom Gebäuderechner, gesendet wurden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.1.6.3 Communication

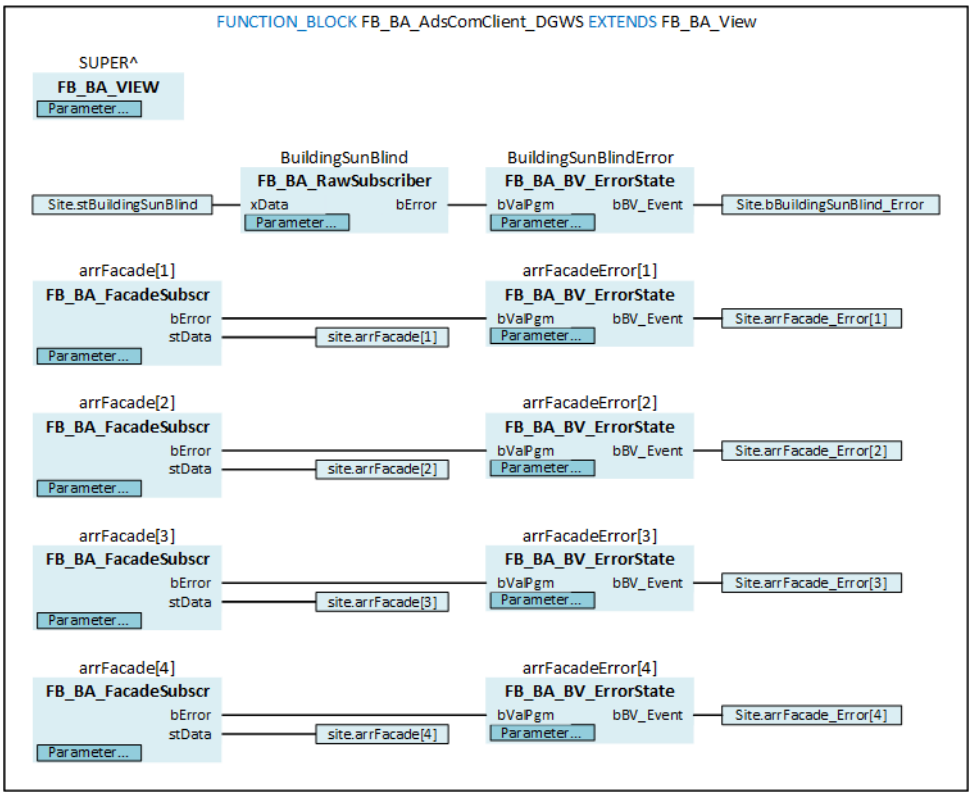
6.1.4.2.1.6.3.1 FB_BA_AdsComClient_DGWS



Das Template liest globale Daten, welche dem Gewerk "Türen, Tore, Fenster, Sonnenschutz" zuzuordnen sind, von einem anderen Controller, der diese Daten mit dem Gegenbaustein FB_BA_AdsComServer_DGWS [▶ 998] zur Verfügung stellt. Die gelesenen Informationen werden in die globale Variablenliste Site [▶ 1135] kopiert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComClient EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  BuildingSunBlind         : FB_BA_RawSubscriber;
  BuildingSunBlindError    : FB_BA_BV_ErrorState;
  arrFacade                : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_FacadeSubscr;
  arrFacadeError           : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_BV_ErrorState;
END_VAR
```

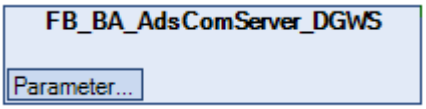
VAR_INPUT CONSTANT

Name	Typ	Beschreibung
BuildingSunBlind	FB_BA_RawSubscriber [▶ 162]	Der Subscriber greift auf die TwinCAT Netzwerk-Struktur *BuildingSunBlind* zu und speichert die Daten in die angelegte Struktur *stBuildingSunBlind* der GVL Site [▶ 1135] ab.
BuildingSunBlindError	FB_BA_BV_ErrorState [▶ 1048]	Binäres Objekt zur Anzeige der Kommunikationsstörung des Subscribers *BuildingSunBlind*.
arrFacade	FB_BA_FacadeSubscr [▶ 961]	Diese Templates empfangen die Fassadendaten, welche an anderer Stelle über einen FB_BA_Facade [▶ 956] definert bzw. zur Verfügung gestellt wurden. In der globalen Parameterliste BA2_Param [▶ 1137] sind mit *nMaxNumberOfFacades* 4 Fassaden vordefiniert, welche hier im *FB_Init* verbal auf Norden, Osten, Süden und Westen initialisiert sind.
arrFacadeError	FB_BA_BV_ErrorState [▶ 1048]	Binäre Objekte zur Anzeige der Kommunikationsstörung der Fassaden-Subscriber.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

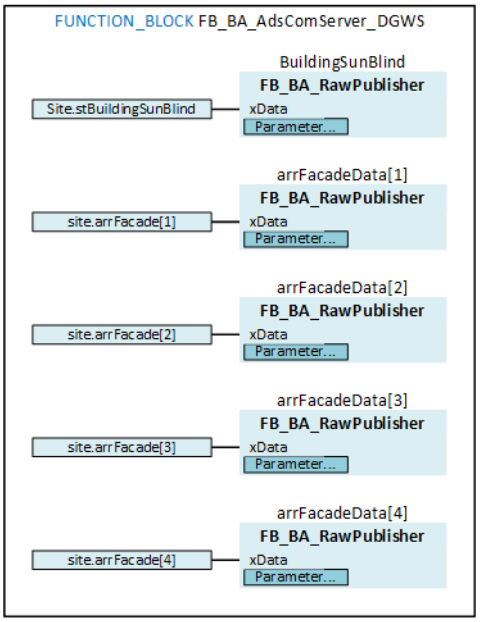
6.1.4.2.1.6.3.2 FB_BA_AdsComServer_DGWS



Das Template hat die Aufgabe Daten und Parameter, welche dem Gewerk "Türen, Tore, Fenster, Sonnenschutz" zuzuordnen sind, aus der globalen Variablenliste Site [▶ 1135] zu lesen und via FB_BA_RawPublisher [▶ 143] innerhalb des GA-Netzwerkes zur Verfügung zu stellen.

Der Gegenbaustein, der diese Daten als Client empfängt, ist FB_BA_AdsComClient_DGWS [▶ 997].

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AdsComServer
VAR_INPUT CONSTANT
  BuildingSunBlind    : FB_BA_RawPublisher;
  arrFacadeData       : ARRAY[1..BA2_Param.nMaxNumberOfFacades] OF FB_BA_RawPublisher;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
BuildingSunBlind	FB_BA_RawPublisher [▶ 143]	Der Publisher veröffentlicht die Struktur der gebäudeweit gültigen Sonnenschutzdaten Site.stBuildingSunBlind [▶ 1135], welche im Template FB_BA_BuildingSunprotection [▶ 953] gebildet wird.
arrFacadeData	FB_BA_RawPublisher [▶ 143]	Die Publisher *arrFacadeData[1..4]* veröffentlichen die Strukturen der gebäudeweit gültigen Verschattungsdaten site.arrFacade[1..4] [▶ 1135], welche in den Instanzen des Templates FB_BA_Facade [▶ 956] gebildet wird. Die Anzahl der Fassaden-Instanzen richtet sich nach dem Gebäudegrundriss. Die Fassadenanzahl "4" ist hier als typisch verwendet worden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

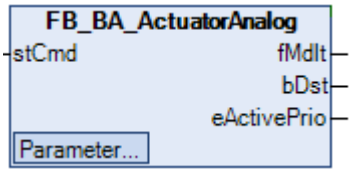
6.1.4.2.2 Universal

Templates zum universellen Einsatz in verschiedenen Gebäudefunktionen.

6.1.4.2.2.1 Aggregate

6.1.4.2.2.1.1 Analog

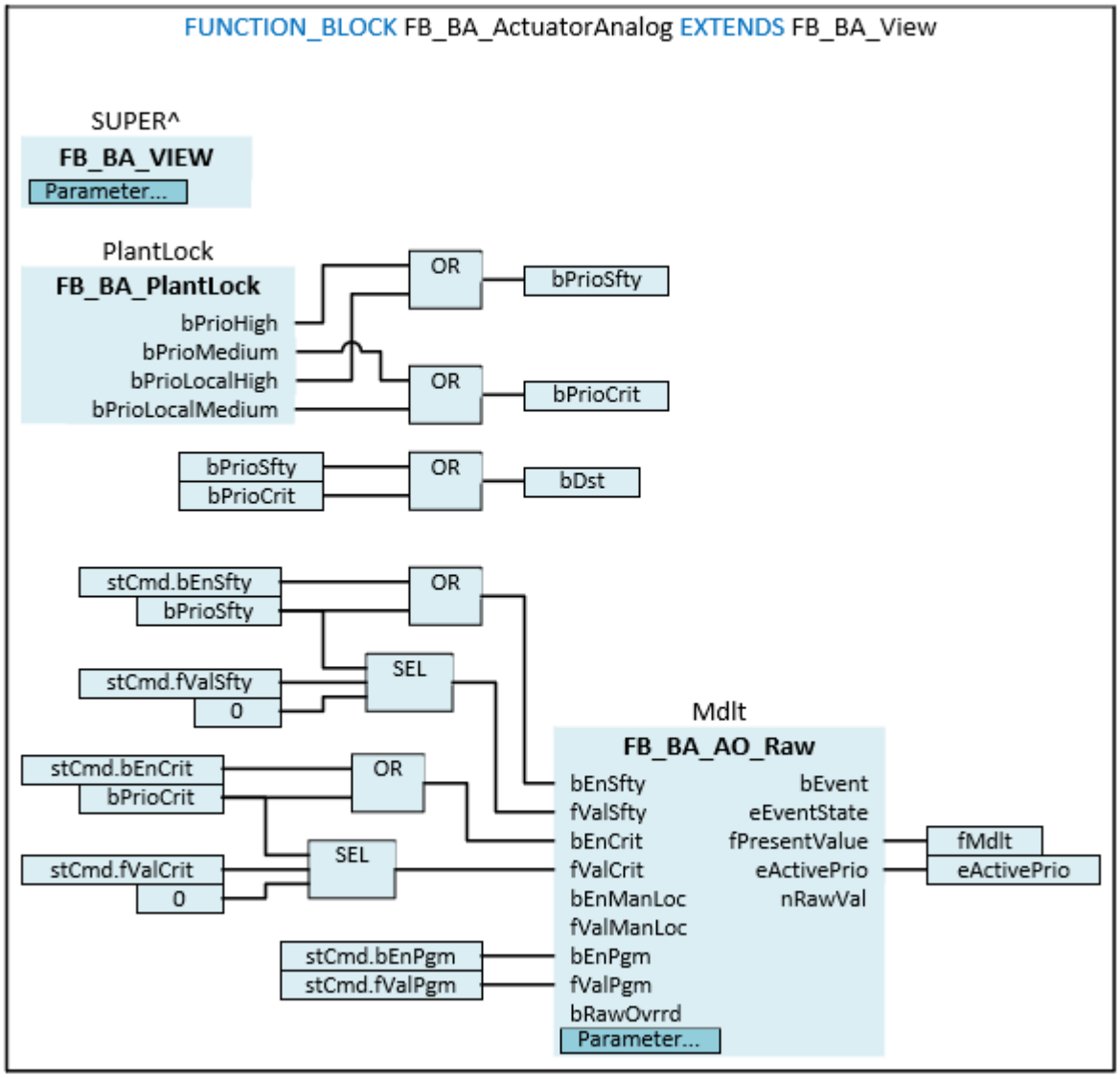
6.1.4.2.2.1.1.1 FB_BA_ActuatorAnalog



Das Template dient der Ansteuerung analoger Aggregate. Es besteht im Wesentlichen aus einem AO-Objekt für die Ansteuerung eines Aggregates und dem Funktionsbaustein *PlantLock*, welcher alle sicherheitsrelevanten Störungen sammelt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
  stCmd          : ST_BA_Analog;
END_VAR
VAR_OUTPUT
  fMdlt          : REAL;
  bDst           : BOOL;
  eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
  Mdlt           : FB_BA_AO_Raw;
  PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty      : BOOL;
  bPrioCrit      : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stCmd	ST_BA_Analog [▶ 281]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Schaltwerte übermittelt. Das Kommando mit der höchsten Priorität wird am Analog-Ausgabe-Objekt *Mdlt* ausgegeben.

Ausgänge

Name	Typ	Beschreibung
fMdlt	REAL	Aktueller Wert des Analog-Output-Objektes.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

Eingänge CONSTANT

Name	Typ	Beschreibung
Mdlt	FB_BA_AO_Raw [▶ 204]	Aktueller Wert des Analog-Output-Objektes.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

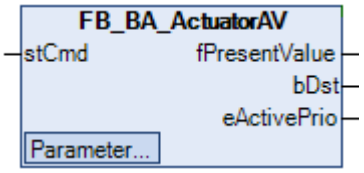
Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.1.1.2 FB_BA_ActuatorAV

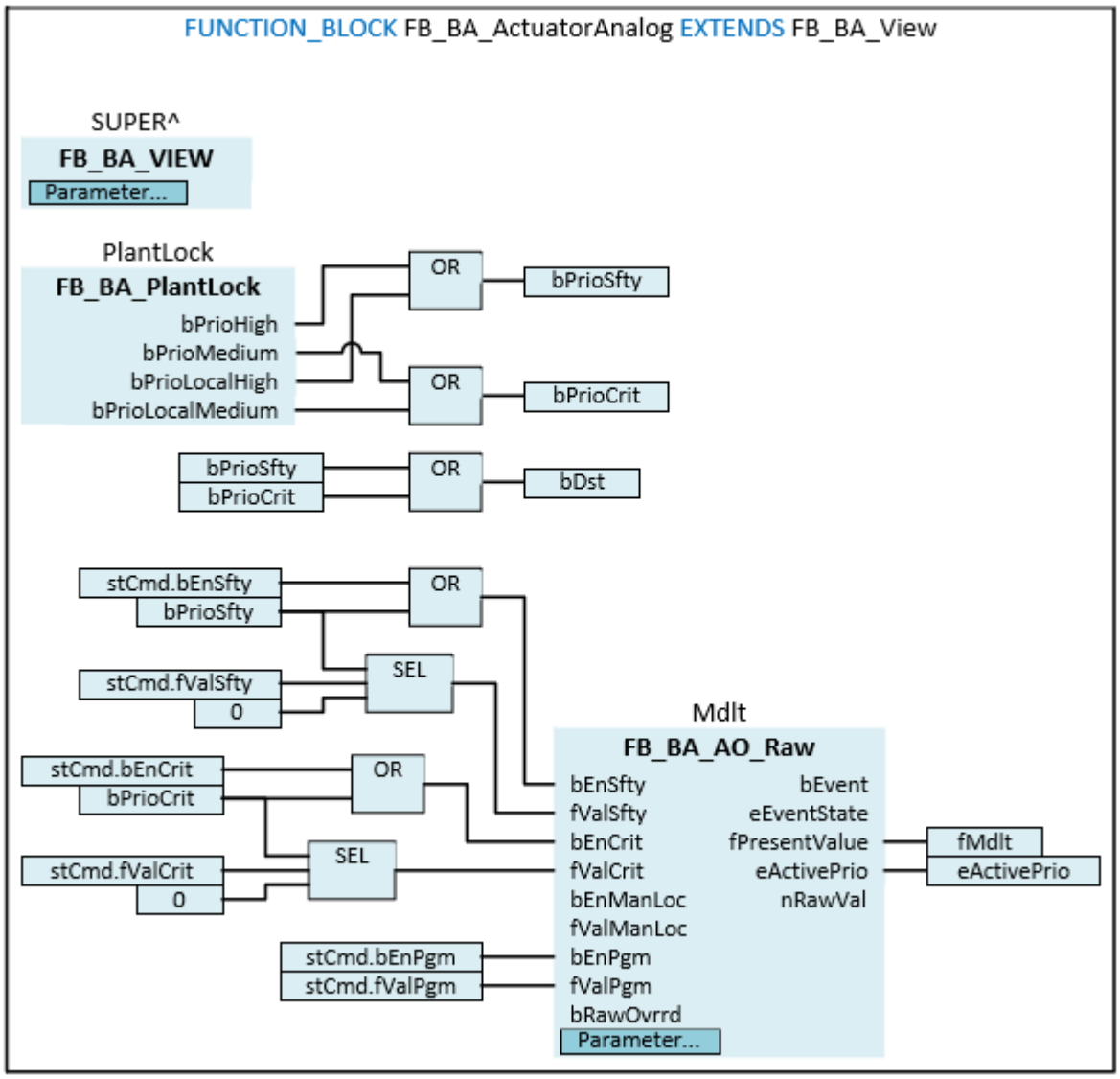


Das Template dient der Ansteuerung analoger Aggregate. Es besteht im Wesentlichen aus einem AV-Objekt für die Ansteuerung eines Aggregates und dem Funktionsbaustein PlantLock, welcher alle sicherheitsrelevanten Störungen sammelt.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorAV EXTENDS FB_BA_View
VAR_INPUT
  stCmd             : ST_BA_Analog;
END_VAR
VAR_OUTPUT
  fPresentValue     : REAL;
  bDst              : BOOL;
```

```
  eActivePrio      : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
  Val              : FB_BA_AV;
  PlantLock        : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty        : BOOL;
  bPrioCrit        : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stCmd	ST_BA_Analog [▶ 281]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Schaltwerte übermittelt. Das Kommando mit der höchsten Priorität wird am Analog-Werte-Objekt *Val* ausgegeben.

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller Wert des Analog-Werte-Objektes.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

Eingänge CONSTANT

Name	Typ	Beschreibung
Val	FB_BA_AV [▶ 205]	Das Analog-Werte-Objekt ermittelt den aktuellen Stellwert.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

Variablen

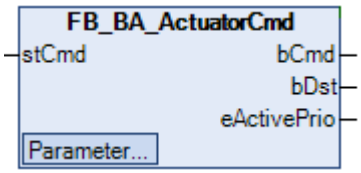
Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.1.2 Binary

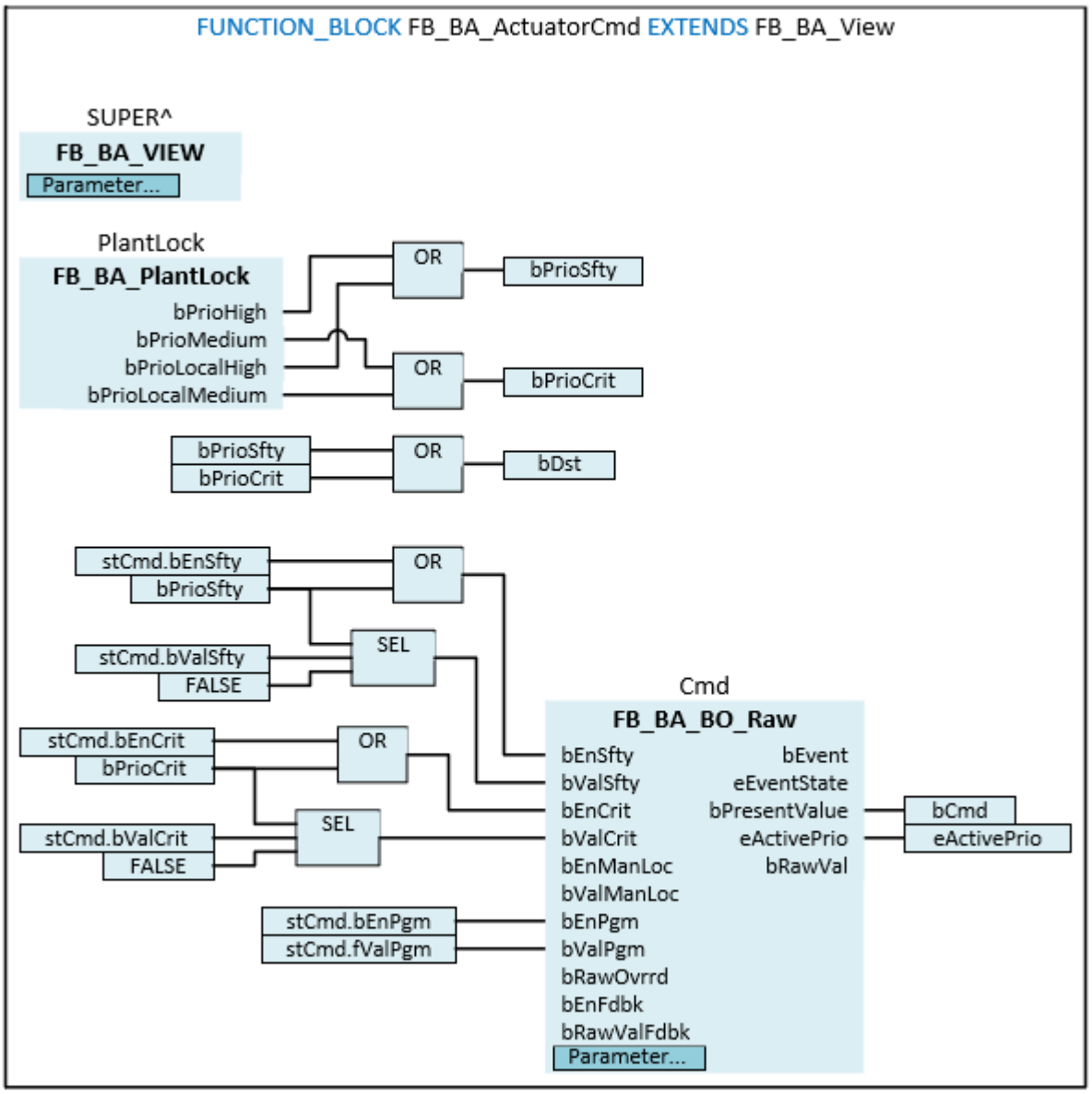
6.1.4.2.2.1.2.1 FB_BA_ActuatorCmd



Das Template dient der Ansteuerung binärer Aggregate. Es besteht im Wesentlichen aus einem BO-Objekt für die Ansteuerung eines Aggregates und dem Funktionsbaustein *PlantLock*, welcher alle sicherheitsrelevanten Störungen sammelt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorAnalog EXTENDS FB_BA_View
VAR_INPUT
  stCmd          : ST_BA_Binary;
END_VAR
VAR_OUTPUT
  bCmd           : BOOL;
  bDst           : BOOL;
  eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
  Cmd            : FB_BA_BO_Raw;
  PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty      : BOOL;
  bPrioCrit      : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stCmd	ST_BA_Binary [▶ 282]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Schaltwerte übermittelt. Das Kommando mit der höchsten Priorität wird am Biären-Ausgabe-Objekt *Cmd* ausgegeben.

Ausgänge

Name	Typ	Beschreibung
bCmd	BOOL	Aktueller Wert des Binären-Ausgangs-Objektes.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

Eingänge CONSTANT

Name	Typ	Beschreibung
Cmd	FB_BA_BO_Raw [▶ 216]	Das Binär-Ausgangs-Objekt dient der Ausgabe eines Schaltbefehls und übermittelt diesen an die IO-Ebene.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

Variablen

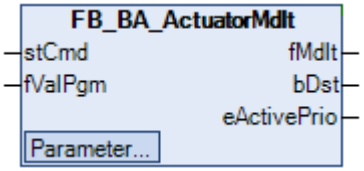
Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.1.3 Modulation

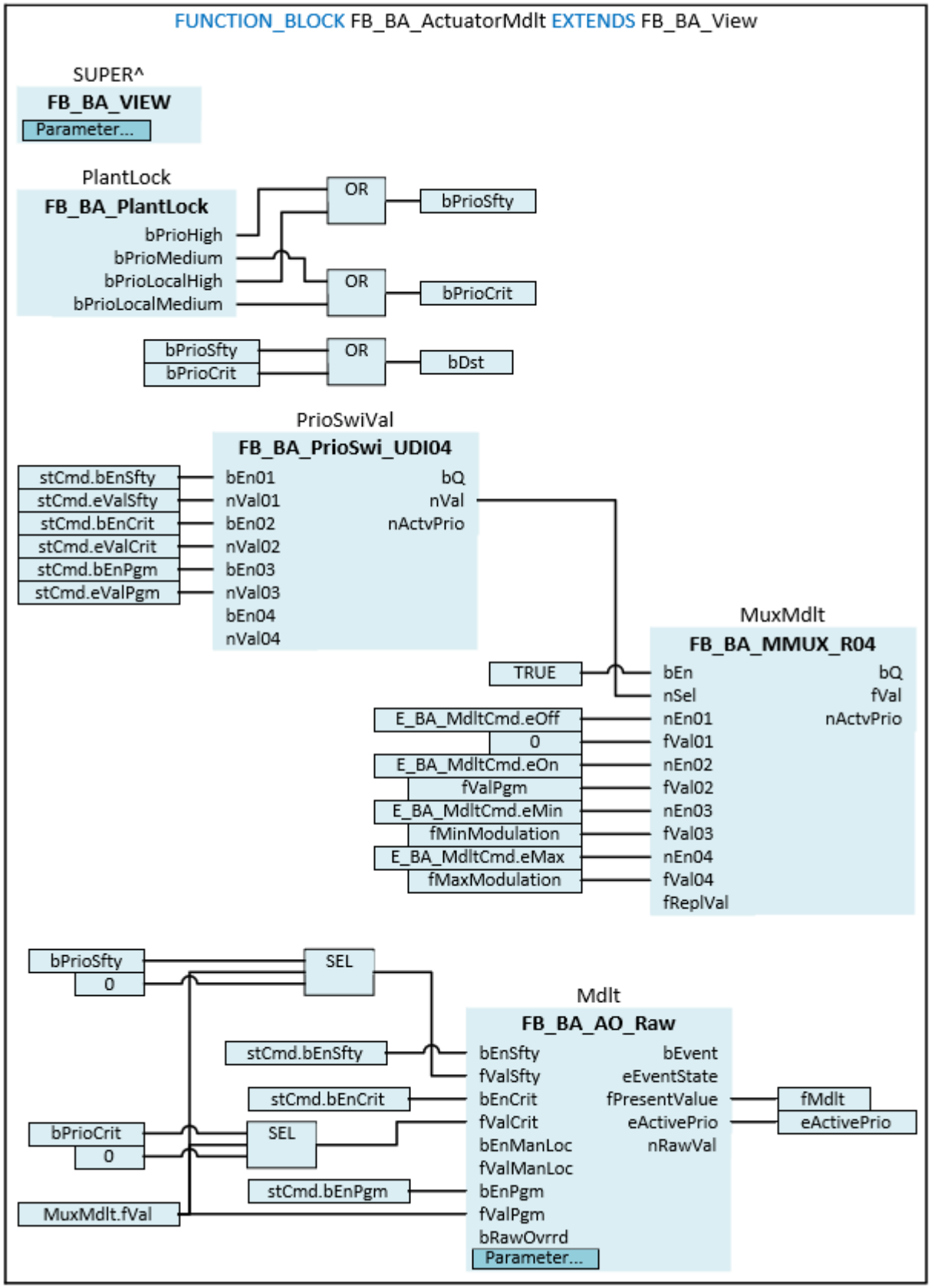
6.1.4.2.2.1.3.1 FB_BA_ActuatorMdlt



Das Template dient der Ansteuerung modulierender Aggregate. Es besteht im Wesentlichen aus einem AO-Objekt für die Ansteuerung eines Aggregates, dem Prioritätenschalter *PrioSwiVal* zur Ermittlung des Modulationsbefehl, dem Multiplexer *MuxMdlt* zur Ermittlung des Stellwertes und dem Funktionsbaustein *PlantLock*, welcher alle sicherheitsrelevanten Störungen sammelt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorMdlt EXTENDS FB_BA_View
VAR_INPUT
  stCmd             : ST_BA_Mdlt;
  fValPgm           : REAL;
END_VAR
```

```
VAR_OUTPUT
  fMdlt             : REAL;
  bDst              : BOOL;
  eActivePrio       : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory'  := 'Behaviour'}
  fMinModulation    : REAL := 20;
  {attribute 'parameterCategory'  := 'Behaviour'}
  fMaxModulation    : REAL := 100;
END_VAR
VAR_INPUT CONSTANT
  Mdlt              : FB_BA_AO_Raw;
  PlantLock         : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty         : BOOL;
  bPrioCrit         : BOOL;

  PrioSwiVal        : FB_BA_PrioSwi_UDI04;
  MuxMdlt           : FB_BA_MMUX_R04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stCmd	ST_BA_Mdlt [▶ 281]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Modulationsbefehle übermittelt. Das Kommando mit der höchsten Priorität bestimmt am Binär-Ausgangs-Objekt *Cmd* den Schaltbefehl and am Analog-Ausgabe-Objekt *Mdlt* den Stellbefehl.
fValPgm	REAL	Über die Eingangsvariable *fValPgm* wird dem Template das Stellsignal für Modulationsbefehl *E_BA_MdltCmd.eOn* zugeführt.

Ausgänge

Name	Typ	Beschreibung
fMdlt	REAL	Aktueller Wert des Analog-Output-Objektes.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fMinModulation	REAL	Konstanter minimaler Wert, wenn der Modulationsbefehl E_BA_MdltCmd.eMin ansteht.
fMaxModulation	REAL	Konstanter maximaler Wert, wenn der Modulationsbefehl E_BA_MdltCmd.eMax ansteht.

Eingänge CONSTANT

Name	Typ	Beschreibung
Mdlt	FB_BA_AO_Raw [▶ 204]	Aktueller Wert des Analog-Output-Objektes.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

Variablen

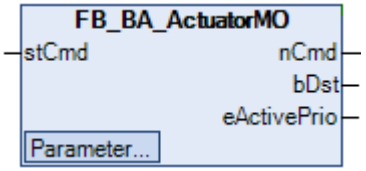
Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter PrioSwiVal [▶ 443] ermittelt anhand der Kommandostruktur *stCmd* den Modulationsbefehl für die Multiplexer *MuxCmd* und *MuxMdlt*.
MuxMdlt	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer MuxMdlt [▶ 438] ermittelt den aktuellen Stellwert aus den Modulationswerten *fValPgm, fMinModulation* und *fMaxModulation* und dem Modulationsbefehl des Prioritätenschalters *PrioSwiVal.* Das resultierende Ergebnis wird an das Analog-Ausgangs-Objekt *Mdlt* übermittelt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.1.4 Multistate

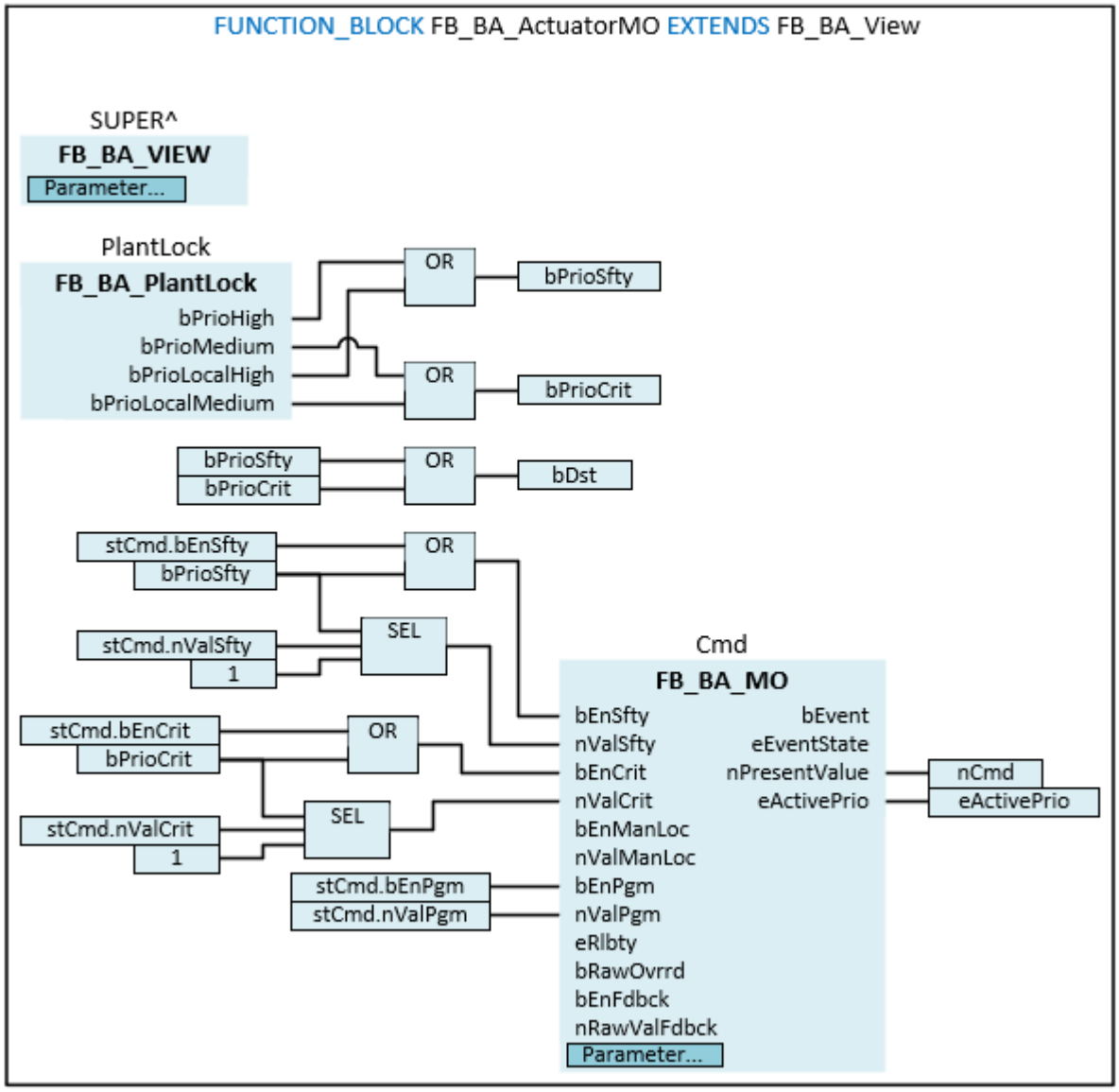
6.1.4.2.2.1.4.1 FB_BA_ActuatorMO



Das Template dient der Ansteuerung mehrstufiger Aggregate. Es besteht im Wesentlichen aus einem MO-Objekt für die Ansteuerung eines Aggregates und dem Funktionsbaustein *PlantLock*, welcher alle sicherheitsrelevanten Störungen sammelt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorMO EXTENDS FB_BA_View
VAR_INPUT
  stCmd          : ST_BA_Multistate;
END_VAR
VAR_OUTPUT
  nCmd           : UDINT;
  bDst           : BOOL;
  eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
  Cmd            : FB_BA_MO;
  PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty      : BOOL;
  bPrioCrit      : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stCmd	ST_BA_Multistate [▶ 281]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Schaltwerte übermittelt. Das Kommando mit der höchsten Priorität wird am Multistate-Ausgabe-Objekt *Cmd* ausgegeben.

Ausgänge

Name	Typ	Beschreibung
nCmd	UDINT	Aktueller Schaltwert des Multistate-Ausgabe-Objekt.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

Eingänge CONSTANT

Name	Typ	Beschreibung
Cmd	FB_BA_MO [▶ 237]	Das Multistate-Ausgabe-Objekt dient der Ausgabe des aktuellen Schaltwertes.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

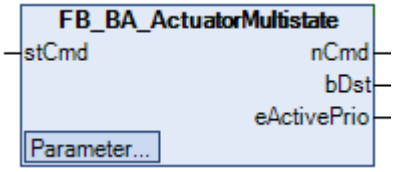
Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

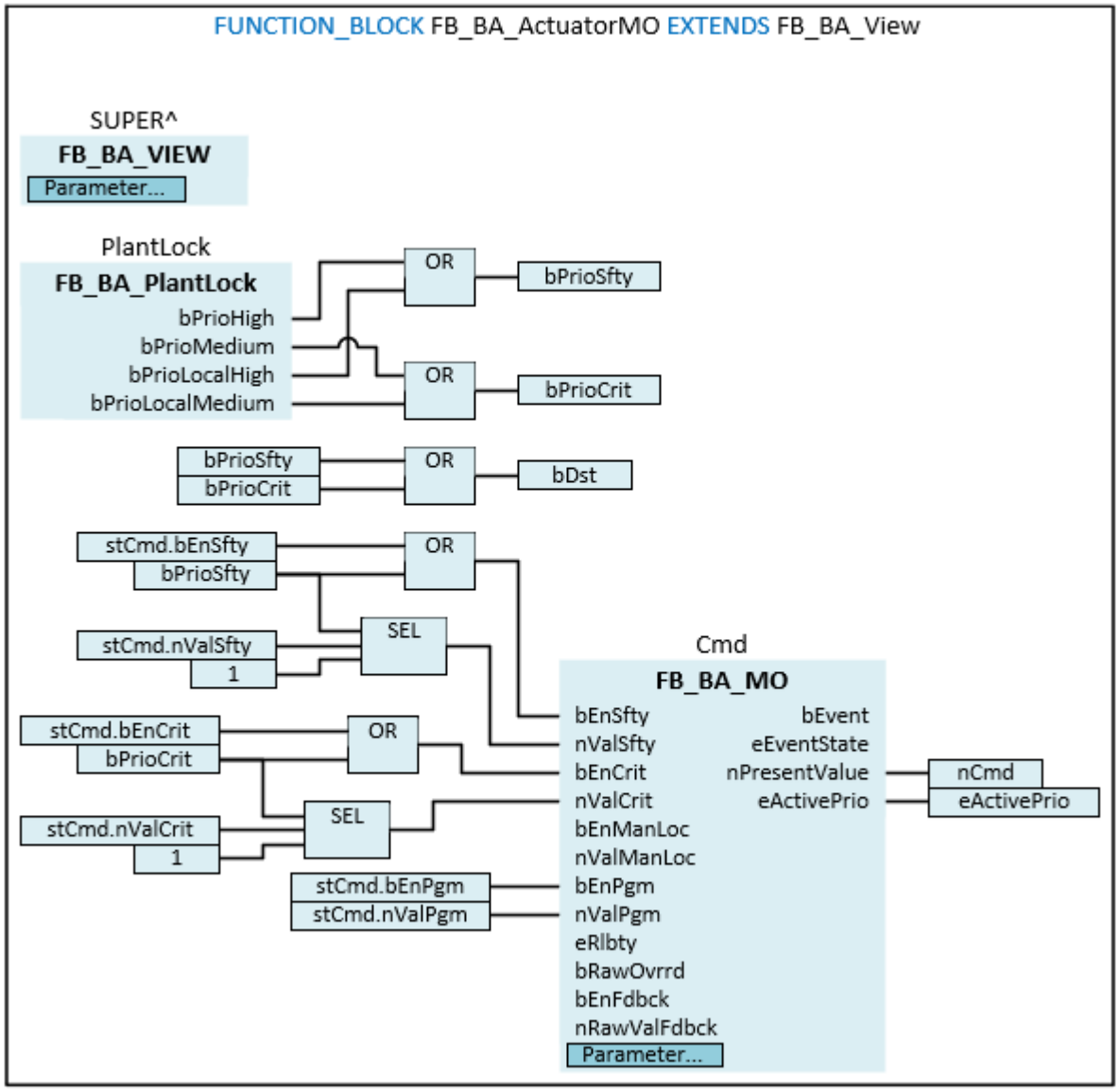
6.1.4.2.2.1.4.2 FB_BA_ActuatorMultistate



Das Template dient der Ansteuerung mehrstufiger Aggregate. Es besteht im Wesentlichen aus einem MO-Objekt für die Ansteuerung eines Aggregates und dem Funktionsbaustein *PlantLock*, welcher alle sicherheitsrelevanten Störungen sammelt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_ActuatorMultistate EXTENDS FB_BA_View
VAR_INPUT
  stCmd          : ST_BA_Multistate;
END_VAR
VAR_OUTPUT
  nCmd           : UDINT;
  bDst           : BOOL;
  eActivePrio    : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
  Cmd            : FB_BA_MO_Raw;
  PlantLock      : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty      : BOOL;
  bPrioCrit      : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stCmd	ST_BA_Multistate [▶ 281]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Schaltwerte übermittelt. Das Kommando mit der höchsten Priorität wird am Multistate-Ausgabe-Objekt *Cmd* ausgegeben.

Ausgänge

Name	Typ	Beschreibung
nCmd	UDINT	Aktueller Schaltwert des Multistate-Ausgabe-Objekt.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

Eingänge CONSTANT

Name	Typ	Beschreibung
Cmd	FB_BA_MO_Raw [▶ 240]	Das Multistate-Ausgabe-Objekt dient der Ausgabe des aktuellen Schaltwertes.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.

Voraussetzungen

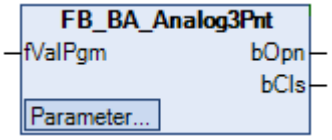
Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2 Control

Volragen zur Präsenzerfassung und Skalierung.

6.1.4.2.2.2.1 Analog3Point

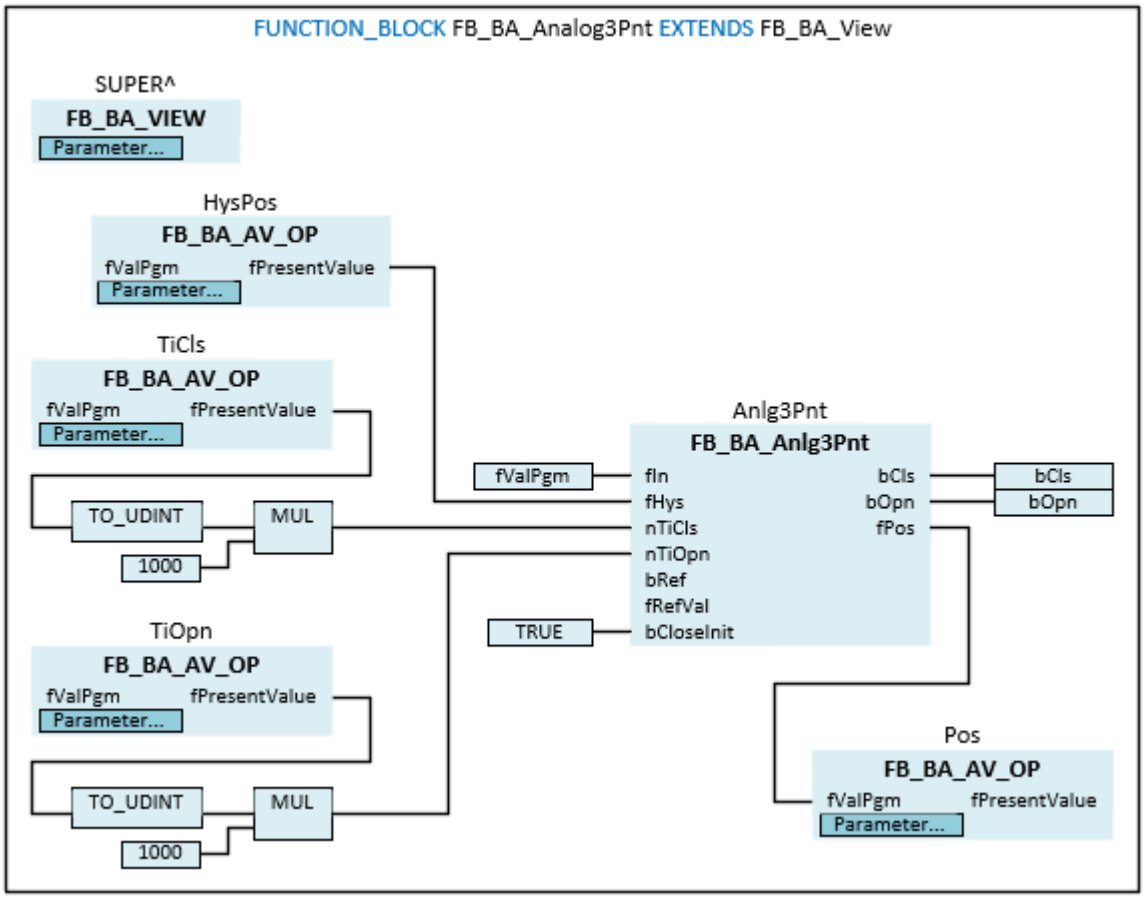
6.1.4.2.2.2.1.1 FB_BA_Analog3Pnt



Das Template wandelt ein stetiges Stellsignal für die Positionierung eines 3-Punktstellorgans in die binären Schaltbefehle Auf/Zu um.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Analog3Pnt EXTENDS FB_BA_View
VAR_INPUT
  fValPgm     : REAL;
END_VAR
VAR_OUTPUT
  bOpn        : BOOL;
  bCls        : BOOL;
END_VAR
VAR_INPUT CONSTANT
  HysPos      : FB_BA_AV_Op;
  TiOpn       : FB_BA_AV_Op;
  TiCls       : FB_BA_AV_Op;
  Pos         : FB_BA_AV_Op;
END_VAR
VAR
  Anlg3Pnt    : FB_BA_Anlg3Pnt;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fValPgm	REAL	Stellwert für die Position des Stellantriebes.

Ausgänge

Name	Typ	Beschreibung
bOpn	BOOL	Ausgang für das Auffahren des Stellantriebs.
bCls	BOOL	Ausgang für das Zufahren des Stellantriebs.

Eingänge CONSTANT

Name	Typ	Beschreibung
HysPos	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Hysteresewertes zum Starten der Positionsänderung.
TiOpn	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes für die Auffahrzeit.
TiCls	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes für die Zufahrzeit.
Pos	FB_BA_AV_Op [▶ 207]	Anzeige der errechneten Position.

VAR

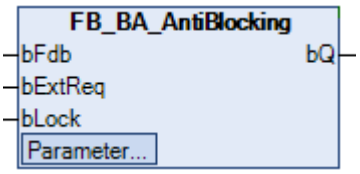
Name	Typ	Beschreibung
Anlg3Pnt	FB_BA_Anlg3Pnt [▶ 417]	Der Funktionsbaustein ist das Kernstück des Templates und zur Ansteuerung eines 3-Punkt Antriebs vorgesehen. Er wandelt ein stetiges Signal für die Positionierung in die binären Befehle Auf/Zu.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.2 AntiBlocking

6.1.4.2.2.2.2.1 FB_BA_AntiBlocking



Das Template verhindert das Blockieren von Pumpen oder Stellantrieben nach längeren Zeiten ohne Bewegung durch die Ausgabe eines Einschaltimpulses.

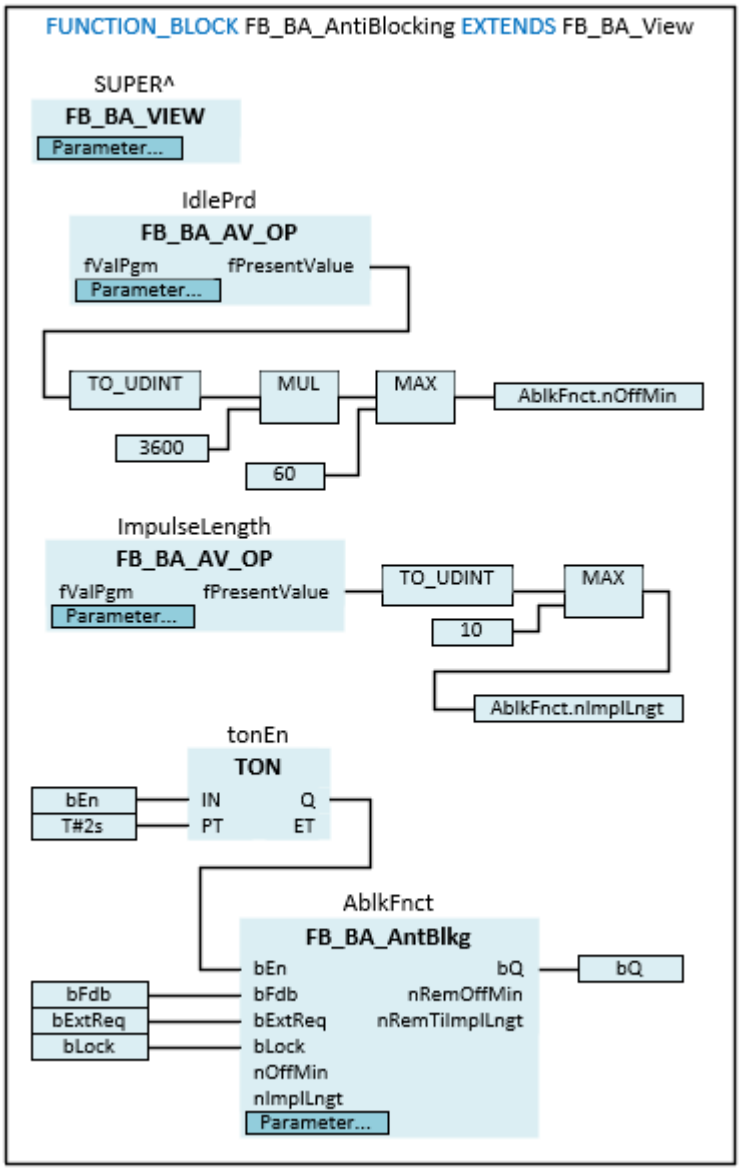
Eine Impulsausgabe erfolgt generell nur, wenn der Funktionsbaustein FB_BA_AntBlkg an *bEn* freigeschaltet ist.

Die maximale Dauer des Stillstands bis zur Ausgabe eines Impulses wird durch den Wert der Variablen *nOffMin* bestimmt. Zur Erfassung der Stillstandzeit muss der Eingang *bFdb* mit der Betriebsrückmeldung des Aggregates verbunden werden. Die Länge des Impulses wird mit der Variablen *nImplLngt* parametriert. Für diese Funktion muss die Betriebsart *E_BA_AntBlkgMode.eOffTime* (siehe E_BA_AntBlkgMode) eingestellt sein.

Falls der Antiblockierschutz-Impuls nicht in Abhängigkeit der Stillstandszeit, sondern zyklisch von einem Zeitschaltplan ausgehen sollen, ist hierfür der Eingang *bExtReq* zu verwenden. Bei einer steigenden Flanke an *bExtReq* wird unmittelbar ein Impuls an *bQ* ausgegeben. Für diese Funktion muss die Betriebsart *E_BA_AntBlkgMode.eExternalRequest* (siehe E_BA_AntBlkgMode) eingestellt sein.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AntBlkg EXTENDS FB_BA_View
VAR_INPUT
  bFdb              : BOOL;
  bExtReq           : BOOL;
  bLock             : BOOL;
END_VAR
VAR_OUTPUT
  bQ                : BOOL;
END_VAR
VAR_INPUT CONSTANT
  IdlePrd           : FB_BA_AV_Op;
  ImpulseLength     : FB_BA_AV_Op;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory':='Operation'}
  bEn               : BOOL := TRUE;
END_VAR
VAR
```

```
  AblkFnct          : FB_BA_AntBlkg;
  tonEn             : TON;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bFdb	BOOL	Eingang zum Anschluss des Feedbacksignals eines Motors oder Ventils. Dieser Eingang wird nur in der Betriebsart *E_BA_AntBlkgMode.eOffTime* berücksichtigt.
bExtReq	BOOL	Aktiv in der Betriebsart *E_BA_AntBlkgMode.eExternalRequest*. Externe Anforderung eines Impulses zum Beispiel von einem Zeitschaltplan. Mit einer steigenden Flanke wird der Antiblockierschutz-Impuls gestartet.
bLock	BOOL	Aktiv in den Betriebsarten *E_BA_AntBlkgMode.eExternalRequest* oder *E_BA_AntBlkgMode.eOffTime*. Um zu verhindern, dass z. B. die Pumpe und das Ventil eines Erhitzers gleichzeitig einen Puls bekommt, wird die Ausgabe des Pulses immer so lange unterdrückt, bis *bLock* wieder FALSE ist. Wird *bLock* während der Ausgabe eines Blockierschutz Pulses TRUE, dann wird der Antiblockierschutz-Impuls unterbrochen. Nachdem bLock wieder FALSE ist, wird der Antiblockierschutz- Impuls neu gestartet.

Ausgänge

Name	Typ	Beschreibung
bQ	BOOL	Ausgabe des Antiblockierschutz-Impulses.

Eingänge CONSTANT

Name	Typ	Beschreibung
IdlePrd	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe der maximalen Dauer einer Pumpenstillstandsperiode bis zur Ausgabe eines Antiblockierschutz-Impulses.
ImpulseLength	FB_BA_AV_Op [▶ 207]	Eine steigende Flanke an diesem Eingang setzt den Ausgangswert *fOut* auf den Eingangswert *fIn*.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates. Wenn *bEn* FALSE ist, so ist der Meldeausgang *bQ* ebenfalls FALSE.

Variablen

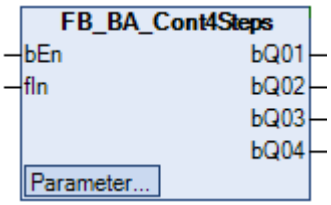
Name	Typ	Beschreibung
bAblkFnct	FB_BA_AntBlkg [▶ 419]	Der Funktionsbaustein *AblkFnct* zur Ausgabe eines Antiblockierschutz-Impulses ist das Kernstück dieses Templates.
tonEn	TON	Einschaltverzögerung der Funktion nach dem Hochlaufen der Steuerung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.3 ContinuousSteps

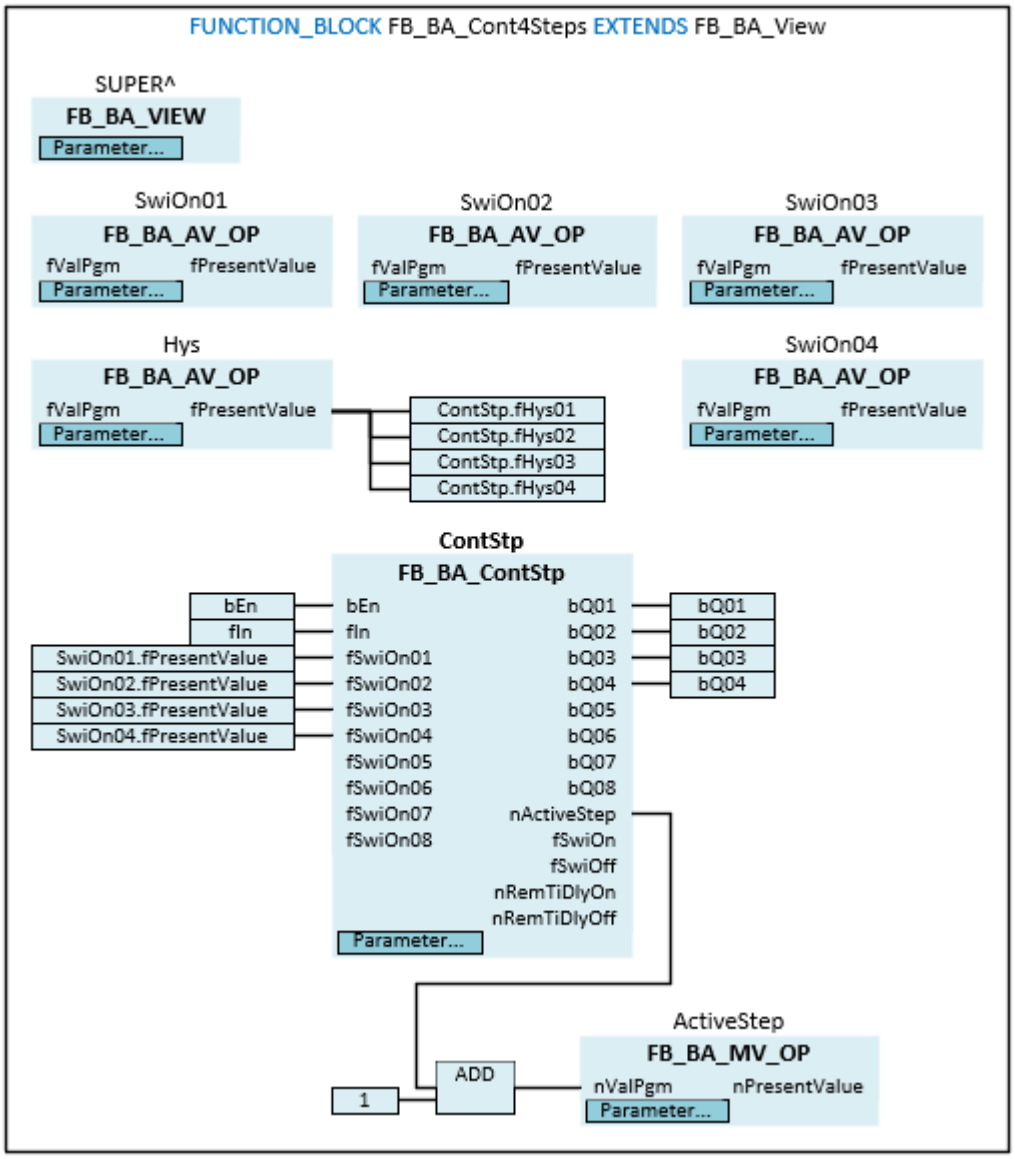
6.1.4.2.2.2.3.1 FB_BA_Cont4Steps



Das Template ermittelt, abhängig vom stetigen Eingangssignal *fIn*, die resultierenden Schaltstufen eines 4-stufigen Aggregats.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Cont4Steps EXTENDS FB_BA_View
VAR_INPUT
  bEn           : BOOL := TRUE;
  fIn           : REAL;
END_VAR
VAR_OUTPUT
  bQ01          : BOOL;
  bQ02          : BOOL;
  bQ03          : BOOL;
  bQ04          : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SwitchOn01    : FB_BA_AV_Op;
  SwitchOn02    : FB_BA_AV_Op;
  SwitchOn03    : FB_BA_AV_Op;
  SwitchOn04    : FB_BA_AV_Op;
  Hys           : FB_BA_AV_Op;
  ActiveStep    : FB_BA_MV_Op;
END_VAR
VAR  ContStp    : FB_BA_ContStp;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates. Wenn *bEn* FALSE ist, so sind sämtliche Meldeausgänge *bQ0x* ebenfalls FALSE.
fIn	REAL	Stetiger Eingangswert, von dem die Schaltzustände abgeleitet werden.

Ausgänge

Name	Typ	Beschreibung
bQ01	BOOL	Anzeige des Status Stufe 01
bQ02	BOOL	Anzeige des Status Stufe 02
bQ03	BOOL	Anzeige des Status Stufe 03
bQ04	BOOL	Anzeige des Status Stufe 04

Eingänge CONSTANT

Name	Typ	Beschreibung
SwitchOn01	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Einschaltpunktes Stufe 01.
SwitchOn02	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Einschaltpunktes Stufe 02.
SwitchOn03	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Einschaltpunktes Stufe 03.
SwitchOn04	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Einschaltpunktes Stufe 04.
Hys	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe der Hysterese für die Einschaltpunkte.
ActiveStep	FB_BA_AV_Op [▶ 207]	MV-Objekt zur Anzeige wie viele Stufen eingeschaltet sind.

Variablen

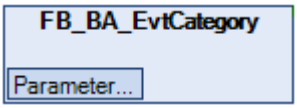
Name	Typ	Beschreibung
ContStp	FB_BA_ContStp [▶ 427]	Der Funktionsbaustein ermittelt abhängig vom stetigen Eingangssignal *fIn* die resultierenden Schaltstufen eines mehrstufigen Aggregats und ist das Kernstück dieses Templates

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.2.4 EventClasses

6.1.4.2.2.2.4.1 FB_BA_EvtCategory



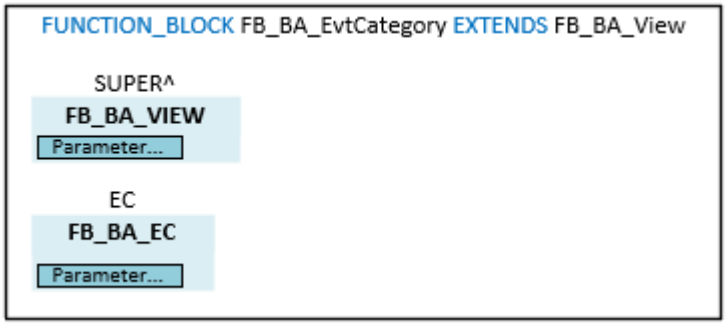
Das Template beinhaltet ein Meldungsklassen-Objekt.

Jedem BACnet-Objekt, das Meldungen mittels Intrinsic Reporting oder Algorithmic Change Reporting erzeugen soll, ist ein Meldungsklassen-Objekt (Notification Class Object) zuzuordnen, dass die Informationen für die Verteilung der Ereignismeldungen beinhaltet.

Das Meldungsklassen-Objekt definiert, welche Prioritäten den Ereignismeldungen zugeordnet werden, ob die Ereignisse eine Quittierung benötigen und welche Empfänger die Meldungen erhalten sollen.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_EvtCategory EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  EC    : FB_BA_EC;
END_VAR
```

Eingänge CONSTANT

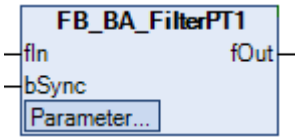
Name	Typ	Beschreibung
EC	FB_BA_EC [▶ 223]	Meldungsklassen-Objekt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.5 Filter

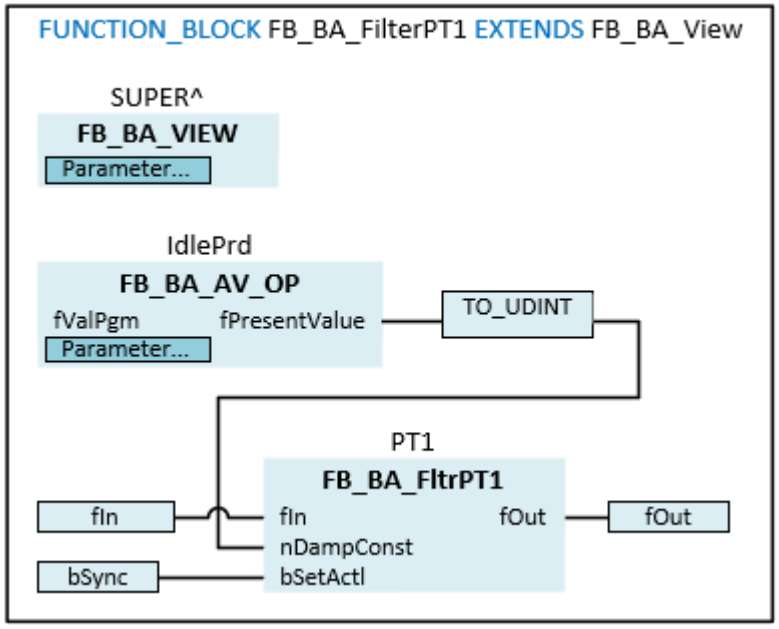
6.1.4.2.2.2.5.1 FB_BA_FilterPT1



Filter erster Ordnung.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_FilterPT1 EXTENDS FB_BA_View
VAR_INPUT
  fIn                : REAL;
  bSync              : BOOL;
END_VAR
VAR_OUTPUT
  fOut               : REAL;
END_VAR
VAR_INPUT CONSTANT
  DampingConstant    : FB_BA_AV_Op;
END_VAR
VAR
  PT1                : FB_BA_FltrPT1;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fIn	REAL	Eingangssignal.
bSync	BOOL	Eine steigende Flanke an diesem Eingang setzt den Ausgangswert *fOut* auf den Eingangswert *fIn*.

Ausgänge

Name	Typ	Beschreibung
fOut	REAL	Gedämpftes Ausgangssignal.

Eingänge CONSTANT

Name	Typ	Beschreibung
DampingConstant	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe Filterzeitkonstante.

Variablen

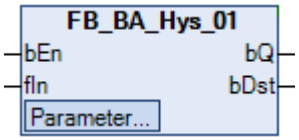
Name	Typ	Beschreibung
PT1	FB_BA_FltrPT1	Meldungsklassen-Objekt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.6 Hysteresis

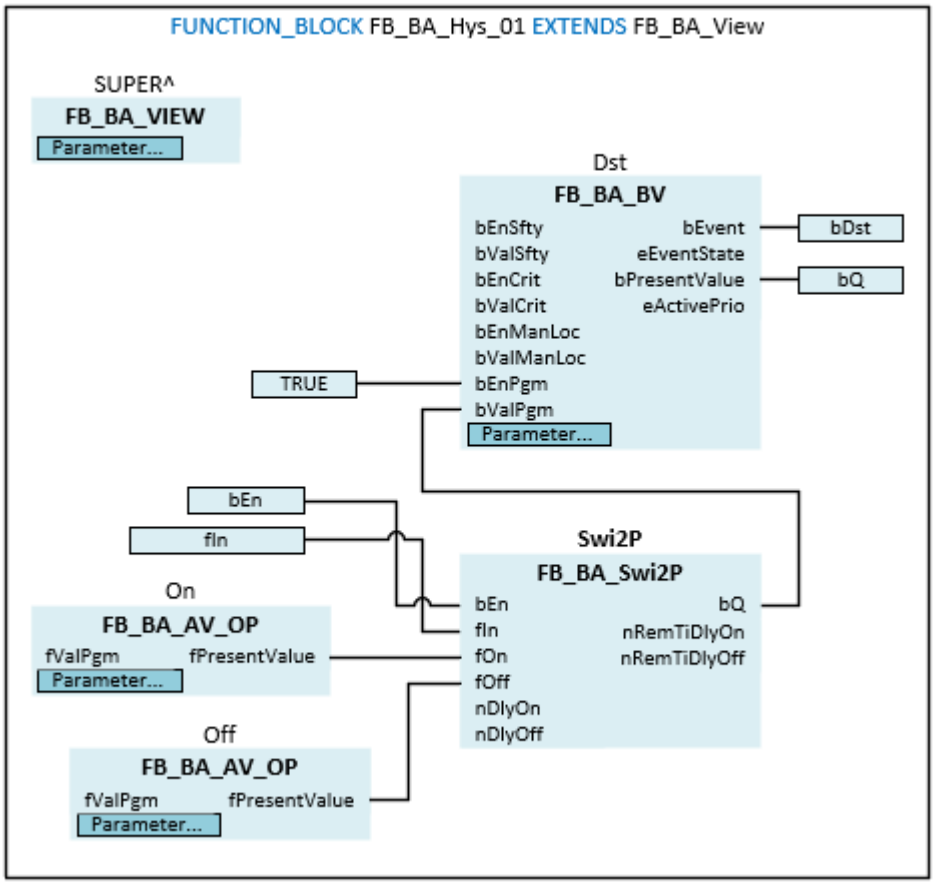
6.1.4.2.2.2.6.1 FB_BA_Hys_01



Das Template stellt eine Hysterese-Funktion mit festen Schaltpunkten dar.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Hys_01 EXTENDS FB_BA_View
VAR_INPUT
  bEn       : BOOL;
  fIn       : REAL;
END_VAR
VAR_OUTPUT
  bQ        : BOOL;
  bDst      : BOOL;
END_VAR
VAR_INPUT CONSTANT
```

```
  On       : FB_BA_AV_Op;
  Off      : FB_BA_AV_Op;
  Q        : FB_BA_BV;
END_VAR
VAR
  Swi2P    : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins.
fIn	REAL	Istwert

Ausgänge

Name	Typ	Beschreibung
bQ	BOOL	Ausgabe des aktuellen Zustands der Hysterese-Funktion.
bDst	BOOL	Anzeige einer Störung bzw. das BV-Objekt ist aktiv. *bDst* ist nur aktiv, wenn das Property *bEventDetectionEnable* des BV-Objektes auf TRUE eingestellt wurde. Die Überwachung der binären Rückmeldung zeigt eine Störung an.

Eingänge CONSTANT

Name	Typ	Beschreibung
On	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des oberen Grenzwertes der Hysterese-Funktion.
Off	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des unteren Grenzwertes der Hysterese-Funktion.
Q	FB_BA_BV [▶ 218]	Das binäre Objekt zeigt den aktuellen Zustand der Hysterese-Funktion.

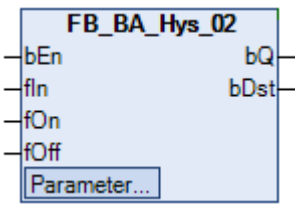
Variablen

Name	Typ	Beschreibung
Swi2P	FB_BA_Swi2P	Der Funktionsbaustein Swi2P ist das Kernstück der Hysterese-Funktion.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

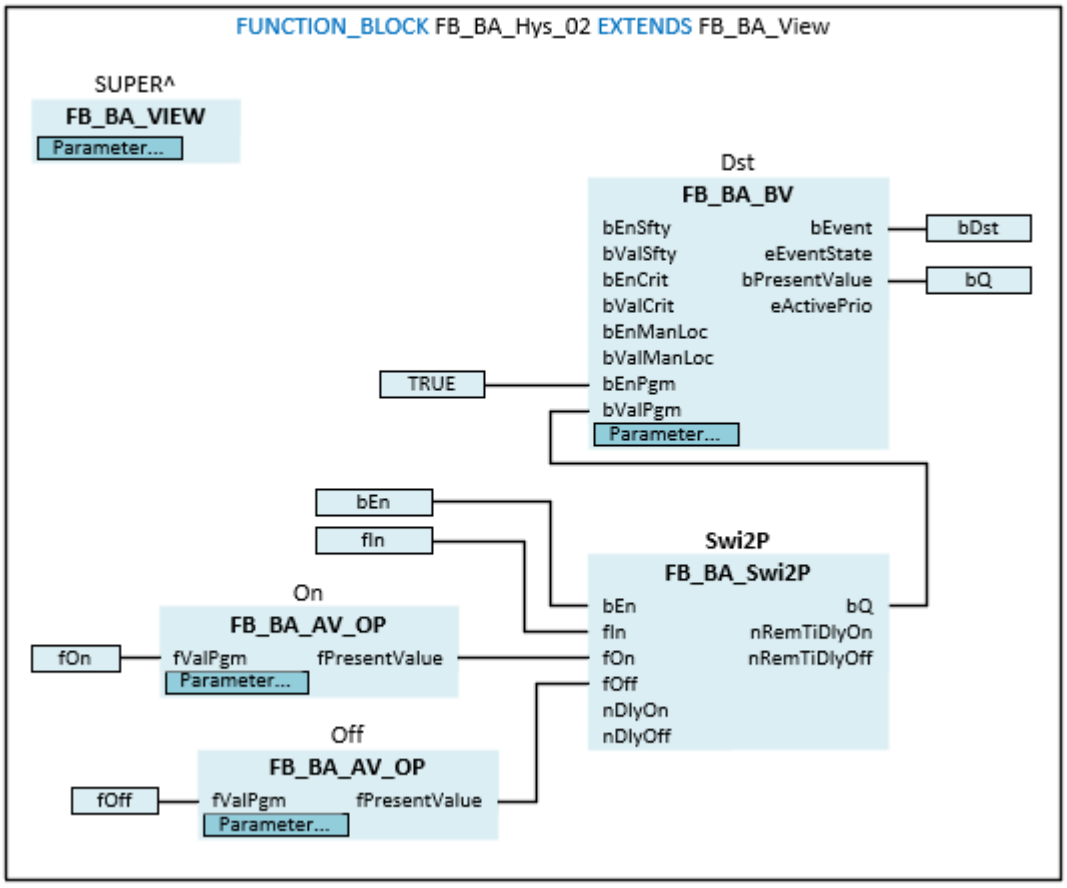
6.1.4.2.2.6.2 FB_BA_Hys_02



Das Template stellt eine gleitende Grenzwertüberwachung mit zwei Schaltpunkten dar.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Hys_01 EXTENDS FB_BA_View
VAR_INPUT
  bEn       : BOOL;
  fIn       : REAL;
  fOn       : REAL;
  fOff      : REAL;
END_VAR
VAR_OUTPUT
  bQ        : BOOL;
  bDst      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  On        : FB_BA_AV_Op;
  Off       : FB_BA_AV_Op;
  Q         : FB_BA_BV;
END_VAR
VAR
  Swi2P     : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins.
fIn	REAL	Istwert
fOn	REAL	Dynamischer Einschaltpunkt
fOff	REAL	Dynamischer Ausschaltpunkt

Ausgänge

Name	Typ	Beschreibung
bQ	BOOL	Ausgabe des aktuellen Zustands der Hysterese-Funktion.
bDst	BOOL	Anzeige einer Störung bzw. das BV-Objekt ist aktiv. *bDst* ist nur aktiv, wenn das Property *bEventDetectionEnable* des BV-Objektes auf TRUE eingestellt wurde. Die Überwachung der binären Rückmeldung zeigt eine Störung an.

Eingänge CONSTANT

Name	Typ	Beschreibung
On	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des oberen Grenzwertes der Hysterese-Funktion.
Off	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des unteren Grenzwertes der Hysterese-Funktion.
Q	FB_BA_BV [▶ 218]	Das binäre Objekt zeigt den aktuellen Zustand der Hysterese-Funktion.

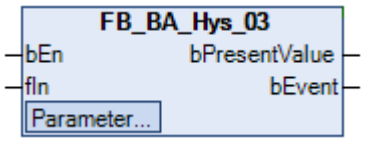
Variablen

Name	Typ	Beschreibung
Swi2P	FB_BA_Swi2P	Der Funktionsbaustein Swi2P ist das Kernstück der Hysterese-Funktion.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.6.3 FB_BA_Hys_03

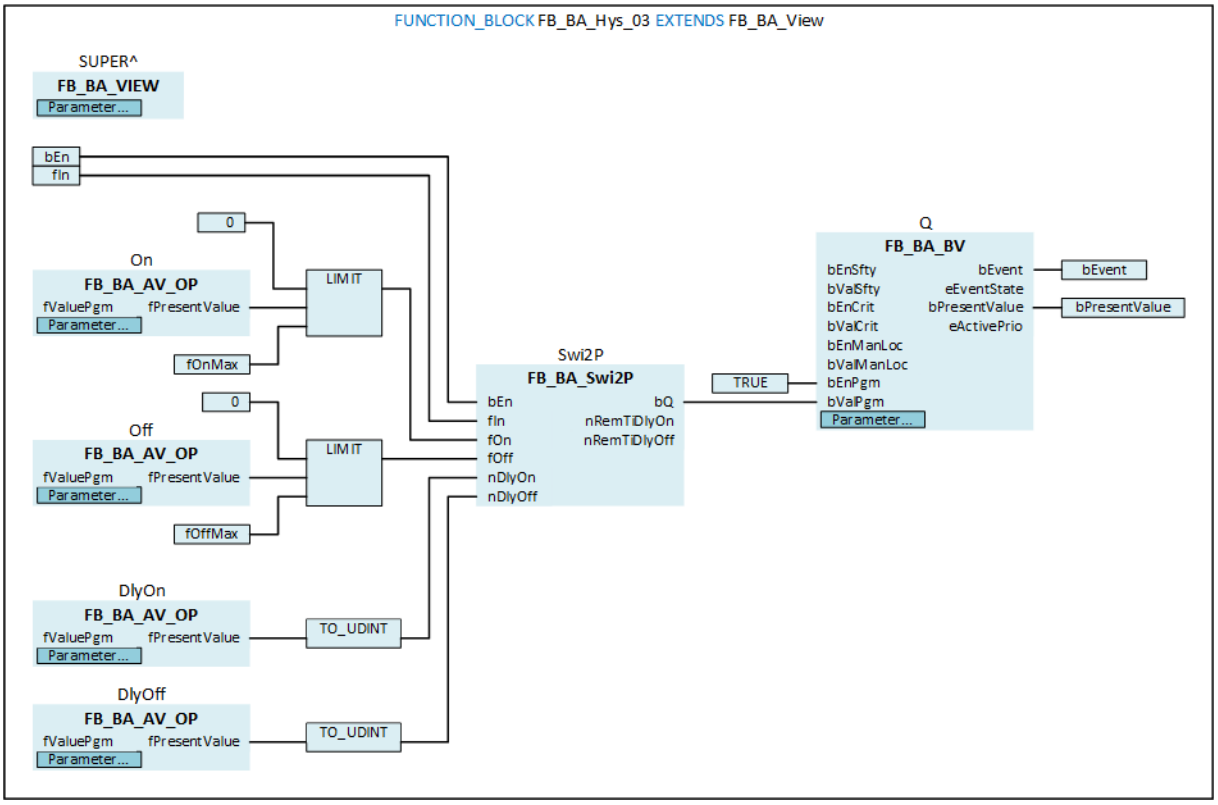


Das Template stellt eine gleitende Grenzwertüberwachung mit zwei Schaltpunkten dar, welche im Gegensatz zu FB_BA_Hys_02 [▶ 1024] mit Ein- und Ausschaltverzögerungen beaufschlagt sind.

Die Eingabe der Ein- und Ausschaltschwelle sowie der Zeitverzögerungen erfolgt über analoge Objekte.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Hys_03 EXTENDS FB_BA_View
VAR_INPUT
  bEn                           : BOOL;
  fIn                           : REAL;
END_VAR
VAR_OUTPUT
  bPresentValue                 : BOOL;
  bEvent                        : BOOL;
END_VAR
VAR_INPUT CONSTANT
  On                            : FB_BA_AV_Op;
  Off                           : FB_BA_AV_Op;
  DlyOn                         : FB_BA_AV_Op;
  DlyOff                        : FB_BA_AV_Op;
  Q                             : FB_BA_BV;
END_VAR
VAR
  Swi2P                         : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins.
fIn	REAL	Istwert

Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Ausgabe des aktuellen Zustands der Hysterese-Funktion.
bEvent	BOOL	Anzeige einer Störung bzw. das BV-Objekt ist aktiv. *bEvent* ist nur aktiv, wenn das Property *bEventDetectionEnable* des BV-Objektes auf TRUE eingestellt wurde. Die Überwachung der binären Rückmeldung zeigt eine Störung an.

Eingänge CONSTANT

Name	Typ	Beschreibung
On	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des oberen Grenzwertes der Hysterese-Funktion.
Off	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des unteren Grenzwertes der Hysterese-Funktion.
DlyOn	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe der Einschaltverzögerung in Sekunden.
DlyOff	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe der Ausschaltverzögerung in Sekunden.
Q	FB_BA_BV [▶ 218]	Das binäre Objekt zeigt den aktuellen Zustand der Hysterese-Funktion.

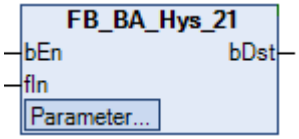
Variablen

Name	Typ	Beschreibung
Swi2P	FB_BA_Swi2P	Der Funktionsbaustein Swi2P ist das Kernstück der Hysterese-Funktion.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.6.4 FB_BA_Hys_21



Das Template stellt zwei Hysterese Funktionen mit festen Schaltpunkten dar.

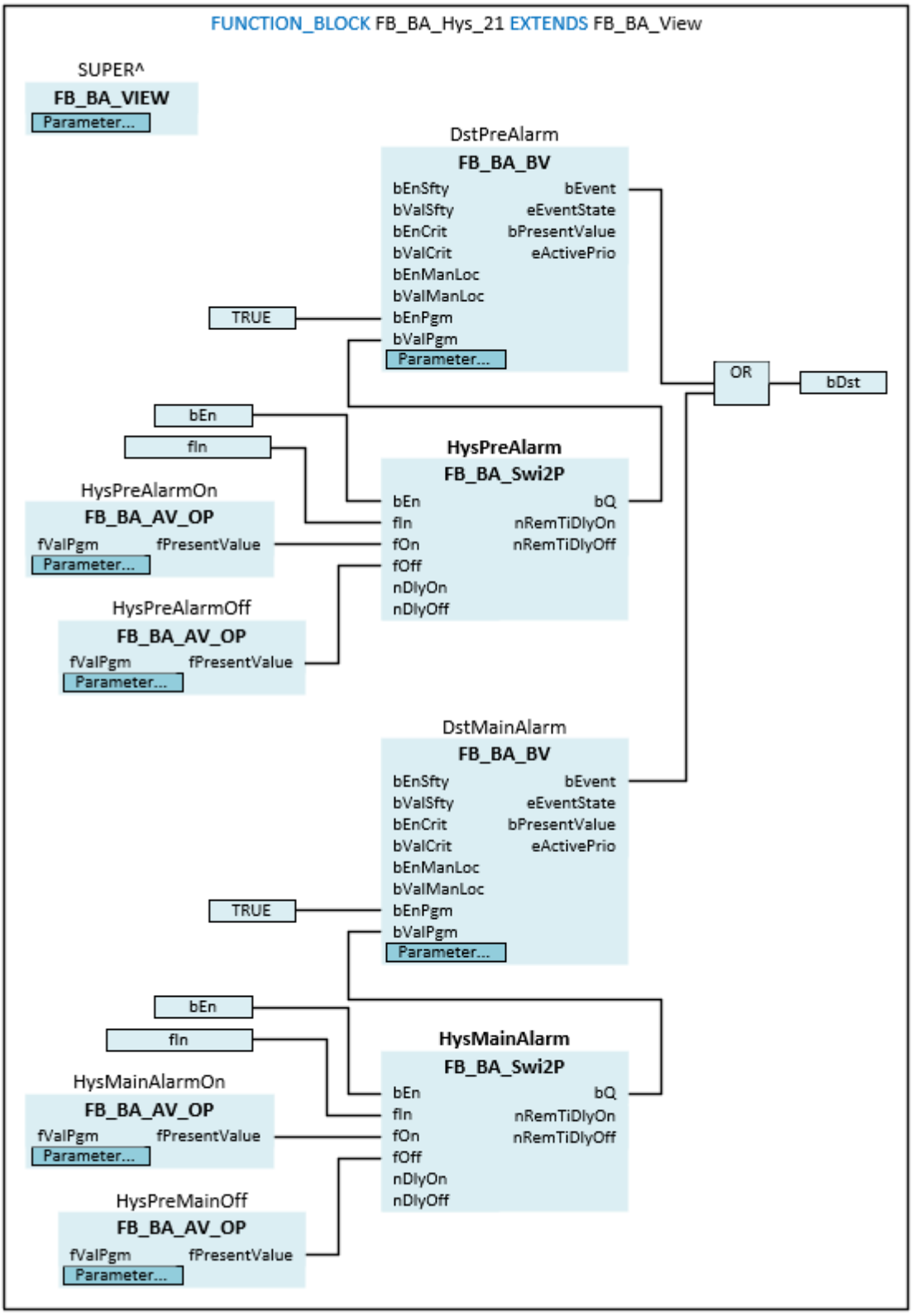
Es dient zur Überwachung von analogen Werten, wie z. B. eines Luftfilters.

Die Hysteresefunktion *HysPreAlarm* löst einen Voralarm aus. An einem Luftfilter kann z. B. eine Wartung stattfinden.

Die Hysteresefunktion *HysMainAlarm* löst einen Hauptalarm aus und kann als anlagenabschaltende Störung verwendet werden. Der Luftfilter muss gewartet werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Hys_21 EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
  fIn                : REAL;
END_VAR
VAR_OUTPUT
  bDst               : BOOL;
END_VAR
VAR_INPUT CONSTANT
```

```
  HysPreAlarmOn      : FB_BA_AV_Op;
  HysPreAlarmOff     : FB_BA_AV_Op;
  HysMainAlarmOn     : FB_BA_AV_Op;
  HysMainAlarmOff    : FB_BA_AV_Op;
  DstPreAlarm        : FB_BA_BV;
  DstMainAlarm       : FB_BA_BV;
END_VAR
VAR
  HysPreAlarm        : FB_BA_Swi2P;
  HysMainAlarm       : FB_BA_Swi2P;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Funktionsbausteins.
fIn	REAL	Istwert

Ausgänge

Name	Typ	Beschreibung
bDst	BOOL	Anzeige einer Störung bzw. das BV-Objekt ist aktiv. *bDst* ist nur aktiv, wenn das Property *bEventDetectionEnable* des BV-Objektes auf TRUE eingestellt wurde. Die Überwachung der binären Rückmeldung zeigt eine Störung an.

Eingänge CONSTANT

Name	Typ	Beschreibung
HysPreAlarmOn	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des oberen Grenzwertes der Hysterese-Funktion *HysPreAlarm*.
HysPreAlarmOff	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des unteren Grenzwertes der Hysterese-Funktion *HysPreAlarm*.
HysMainAlarmOn	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des oberen Grenzwertes der Hysterese-Funktion *HysMainAlarm*.
HysMainAlarmOff	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des unteren Grenzwertes der Hysterese-Funktion *HysMainAlarm*.
DstPreAlarm	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige des Voralarms der Hysteresefunktion *HysPreAlarm*. Über das Intrinsic Reporting wird eine Meldung ausgelöst.
DstMainAlarm	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige des Hauptalarms der Hysteresefunktion *HysMainAlarm*. Über das Intrinsic Reporting wird eine Meldung ausgelöst. Eine anlagenabschaltende Störung muss in der Methode FB_init über *eEnPlantLock* parametriert werden.

Variablen

Name	Typ	Beschreibung
HysPreAlarm	FB_BA_Swi2P	Der Funktionsbaustein ist das Kernstück der Hysterese-Funktion Voralarm.
HysMainAlarm	FB_BA_Swi2P	Der Funktionsbaustein ist das Kernstück der Hysterese-Funktion Hauptalarm.

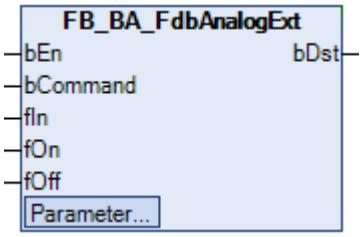
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.2.7 Monitoring

Präsenzerfassung.

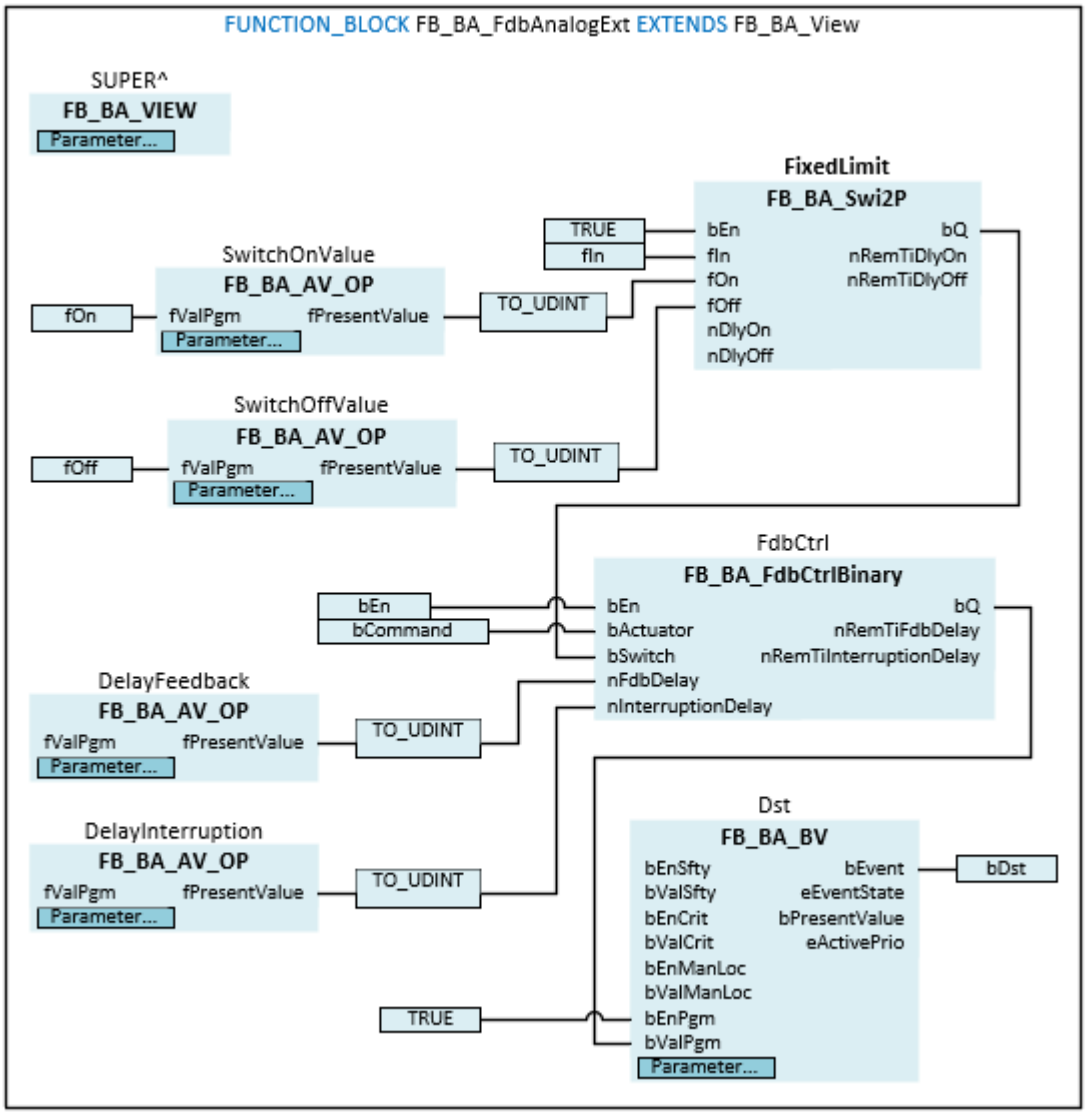
6.1.4.2.2.2.7.1 FB_BA_FdbAnalogExt



Das Template dient der Überwachung von analogen Werten mit dynamischen Schaltwerten, wie z. B. der Differenzdrucküberwachung eines Ventilators.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_FdbAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  bCommand              : BOOL;
  fIn                   : REAL;
  fOn                   : REAL := 10.0;
  fOff                  : REAL := 2.0;
END_VAR
VAR_OUTPUT
  bDst                  : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SwitchOnValue         : FB_BA_AV_Op;
  SwitchOffValue        : FB_BA_AV_Op;
  DelayFeedback         : FB_BA_AV_Op;
  DelayInterruption     : FB_BA_AV_Op;
  Dst                   : FB_BA_BV;
END_VAR
VAR
  FixedLimit            : FB_BA_Swi2P;
  FdbCtrl               : FB_BA_FdbCtrlBinary;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
bCommand	BOOL	An dem Eingang wird der Schaltaktorausgang des zu überwachenden Aggregates angeschlossen.
fIn	REAL	An dem Eingang wird der zu überwachende, analoge Wert angeschlossen, z. B. ein Differenzdruckfühler.
fOn	REAL	An dem Eingang wird der Einschaltwert für *FixedLimit* angeschlossen. Bei einer Differenzdrucküberwachung muss hier ein Druckwert gewählt werden, der im laufenden Prozess nicht unterschritten werden darf.
fOff	REAL	An dem Eingang wird der Ausschaltwert für *FixedLimit* angeschlossen. Dieser Wert muss knapp unterhalb des Einschaltwertes *fOn* liegen.

Ausgänge

Name	Typ	Beschreibung
bDst	BOOL	Binäres Objekt zur Anzeige der Störung.

Eingänge CONSTANT

Name	Typ	Beschreibung
SwitchOnValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des Einschaltwertes für *fOn*.
SwitchOffValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des Ausschaltwertes für *fOff*.
DelayFeedback	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Zeitverzögerung der Information „Aggregat betriebsbereit“. Bei einer Differenzdrucküberwachung muss hier eine Zeitverzögerung angegeben werden, ab der anzunehmen ist, dass die Anlage den benötigten Differenzdruck aufgebaut hat.
DelayInterruption	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Verzögerungszeit, um eine Störmeldung auszulösen. Die Meldung des Differenzdruckfühlers kann verzögert werden, um Druckschwankungen zu überbrücken.
Dst	FB_BA_BV [▶ 218]	Binäres Objekt zur Anzeige der Störung.

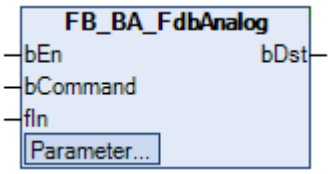
Variablen

Name	Typ	Beschreibung
FixedLimit	FB_BA_Swi2P	Wandlung des analogen Werts *fIn* in ein binäres Schaltsignal für *FdbCtrl*.
FdbCtrl	FB_BA_FdbCtrlBinary [▶ 477]	Überwachung der binären Rückmeldung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

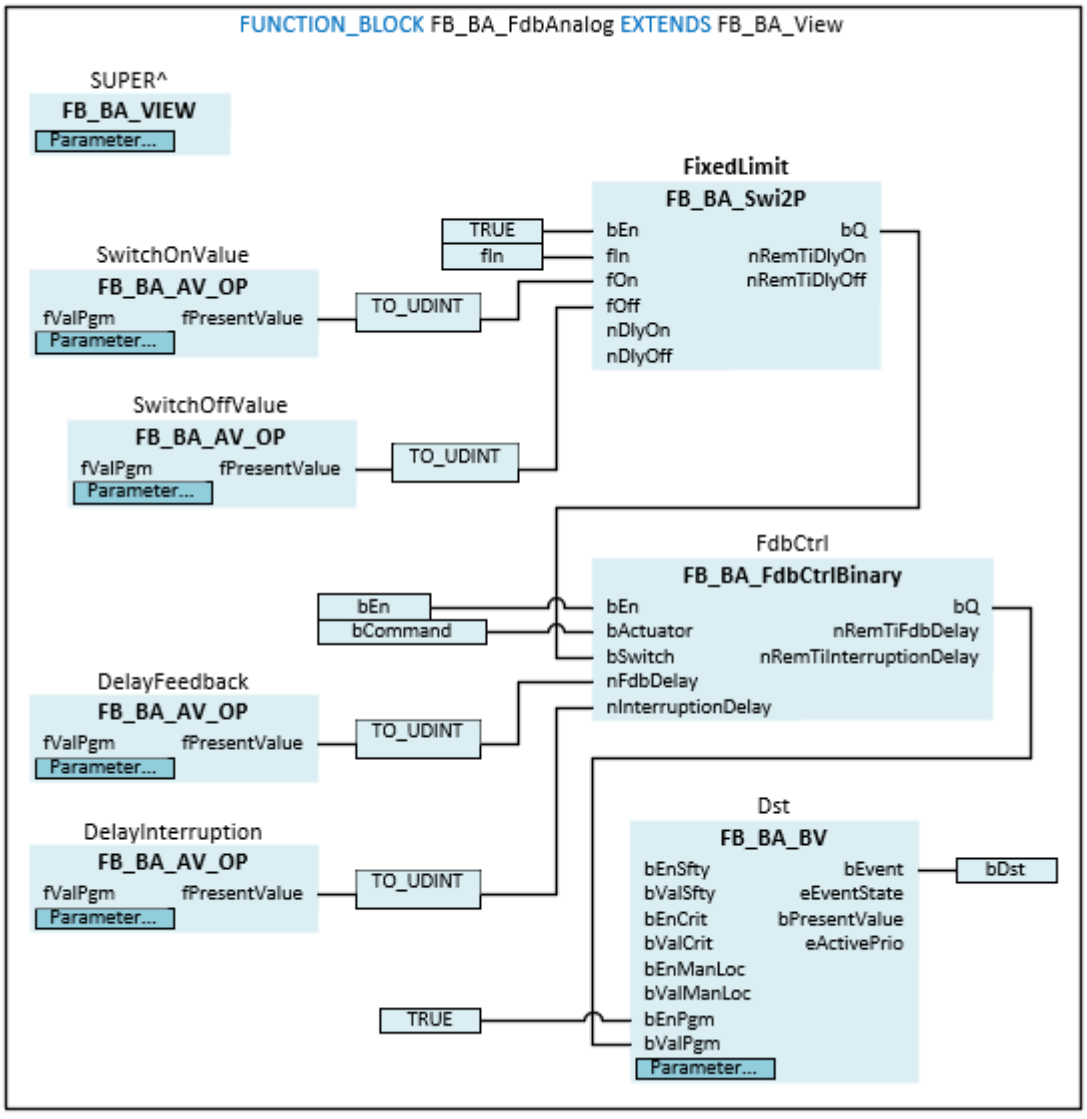
6.1.4.2.2.7.2 FB_BA_FdbAnalog



Das Template dient der Überwachung von analogen Werten mit festen Schaltwerten, wie z. B. einer Differenzdrucküberwachung eines Ventilators.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_FdbAnalog EXTENDS FB_BA_View
VAR_INPUT
  bEn                               : BOOL;
  bCommand                          : BOOL;
  fIn                               : REAL;
END_VAR
VAR_OUTPUT
  bDst                              : BOOL;
END_VAR
```

```
VAR_INPUT CONSTANT
  SwitchOnValue                 : FB_BA_AV_Op;
  SwitchOffValue                : FB_BA_AV_Op;
  DelayFeedback                 : FB_BA_AV_Op;
  DelayInterruption             : FB_BA_AV_Op;
  Dst                           : FB_BA_BV;
END_VAR
VAR
  FixedLimit                    : FB_BA_Swi2P;
  FdbCtrl                       : FB_BA_FdbCtrlBinary;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
bCommand	BOOL	An dem Eingang wird der Schaltaktorausgang des zu überwachenden Aggregates angeschlossen.
fIn	REAL	An dem Eingang wird der zu überwachende, analoge Wert angeschlossen, z. B. ein Differenzdruckfühler.

Ausgänge

Name	Typ	Beschreibung
bDst	BOOL	Binäres Objekt zur Anzeige der Störung.

Eingänge CONSTANT

Name	Typ	Beschreibung
SwitchOnValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des Einschaltwertes für *FixedLimit*. Bei einer Differenzdrucküberwachung muss hier ein Druckwert gewählt werden, der im laufenden Prozess nicht unterschritten werden darf.
SwitchOffValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des Ausschaltwertes für *FixedLimit*. Dieser Wert muss knapp unterhalb des Einschaltwertes *SwitchOnValue* liegen.
DelayFeedback	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Zeitverzögerung der Information „Aggregat betriebsbereit“. Bei einer Differenzdrucküberwachung muss hier eine Zeitverzögerung angegeben werden, ab der anzunehmen ist, dass die Anlage den benötigten Differenzdruck aufgebaut hat.
DelayInterruption	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Verzögerungszeit, um eine Störmeldung auszulösen. Die Meldung des Differenzdruckfühlers kann verzögert werden, um Druckschwankungen zu überbrücken.
Dst	FB_BA_BV [▶ 218]	Binäres Objekt zur Anzeige der Störung.

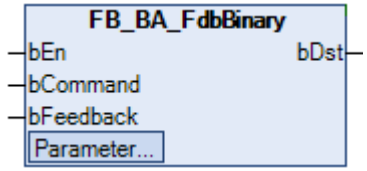
Variablen

Name	Typ	Beschreibung
FixedLimit	FB_BA_Swi2P	Wandlung des analogen Werts *fIn* in ein binäres Schaltsignal für *FdbCtrl*.
FdbCtrl	FB_BA_FdbCtrlBinary [▶ 477]	Überwachung der binären Rückmeldung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

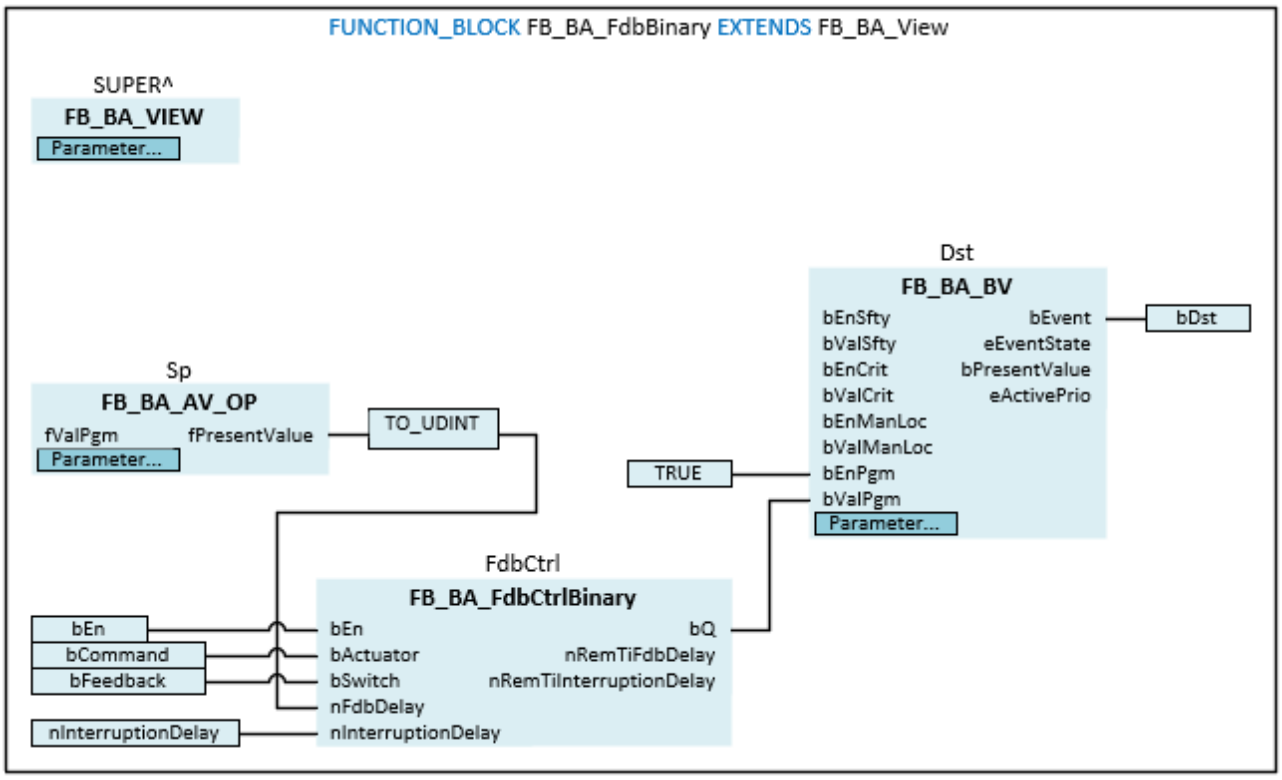
6.1.4.2.2.7.3 FB_BA_FdbBinary



Das Template dient zur Überwachung von binären Rückmeldungen, wie z. B. die Endlagen von Klappen oder Ventilen. Es kann aber auch für eine Differenzdrucküberwachung mittels Differenzdruckschalter eingesetzt werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_FdbBinary EXTENDS FB_BA_View
VAR_INPUT
  bEn                   : BOOL;
  bCommand              : BOOL;
  bFeedback             : BOOL;
END_VAR
VAR_OUTPUT
  bDst                  : BOOL;
END_VAR
VAR_INPUT CONSTANT
  DelayFeedback         : FB_BA_AV_Op;
  Dst                   : FB_BA_BV;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterUnit':= 's'}
```

```
  nInterruptionDelay    : UDINT := 1;
END_VAR
VAR
  FdbCtrl               : FB_BA_FdbCtrlBinary;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
bCommand	BOOL	An dem Eingang wird der Schaltaktorausgang des zu überwachenden Aggregates angeschlossen.
bFeedback	BOOL	An dem Eingang wird das Feedback-Signal des zu überwachenden Aggregates angeschlossen, z. B. ein Differenzdruckschalter, Strömungswächter oder Endlagenschalter.

Ausgänge

Name	Typ	Beschreibung
bDst	BOOL	Binäres Objekt zur Anzeige der Störung.

Eingänge CONSTANT

Name	Typ	Beschreibung
DelayFeedback	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Zeitverzögerung der Rückmeldung. Die Verfahrzeit des Aktors kann verwendet werden. Bei einer Differenzdrucküberwachung muss hier eine Zeitverzögerung angegeben werden, damit die Anlage den benötigten Differenzdruck aufbauen kann.
Dst	FB_BA_BV [▶ 218]	Binäres Objekt zur Anzeige der Störung.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nInterruptionDelay	UDINT	Analog-Wert-Objekt zur Eingabe des Einschaltwertes für *FixedLimit*. Bei einer Differenzdrucküberwachung muss hier ein Druckwert gewählt werden, der im laufenden Prozess nicht unterschritten werden darf.

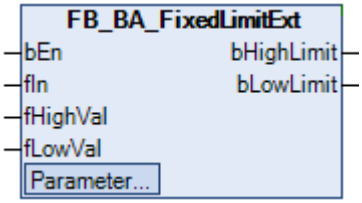
Variablen

Name	Typ	Beschreibung
FdbCtrl	FB_BA_FdbCtrlBinary [▶ 477]	Überwachung der binären Rückmeldung der Aktoren.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.7.4 FB_BA_FixedLimitExt



Das Template stellt einen Grenzwertschalter mit dynamischen Grenzwerten dar.

Um den zu überwachenden Wert *fIn* herum wird ein Toleranzbereich definiert.

Der Toleranzbereich ergibt sich aus einem oberen Grenzwert *HighLimitValue* und einem unteren Grenzwert *LowLimitValue*.

Übersteigt der Wert *fIn* den oberen Grenzwert des Toleranzbereiches, dann wird der Ausgang *bHighLimit* gesetzt.

Eine Ansprechverzögerung des Ausgangs *bHighLimit* ist mit der Zeitvariable *TiDly* zu parametrieren.

Das binäre Objekt *HighLimitOn* dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.

Unterschreitet der Wert *fIn* den unteren Grenzwert des Toleranzbereiches, dann wird der Ausgang *bLowLimit* gesetzt.

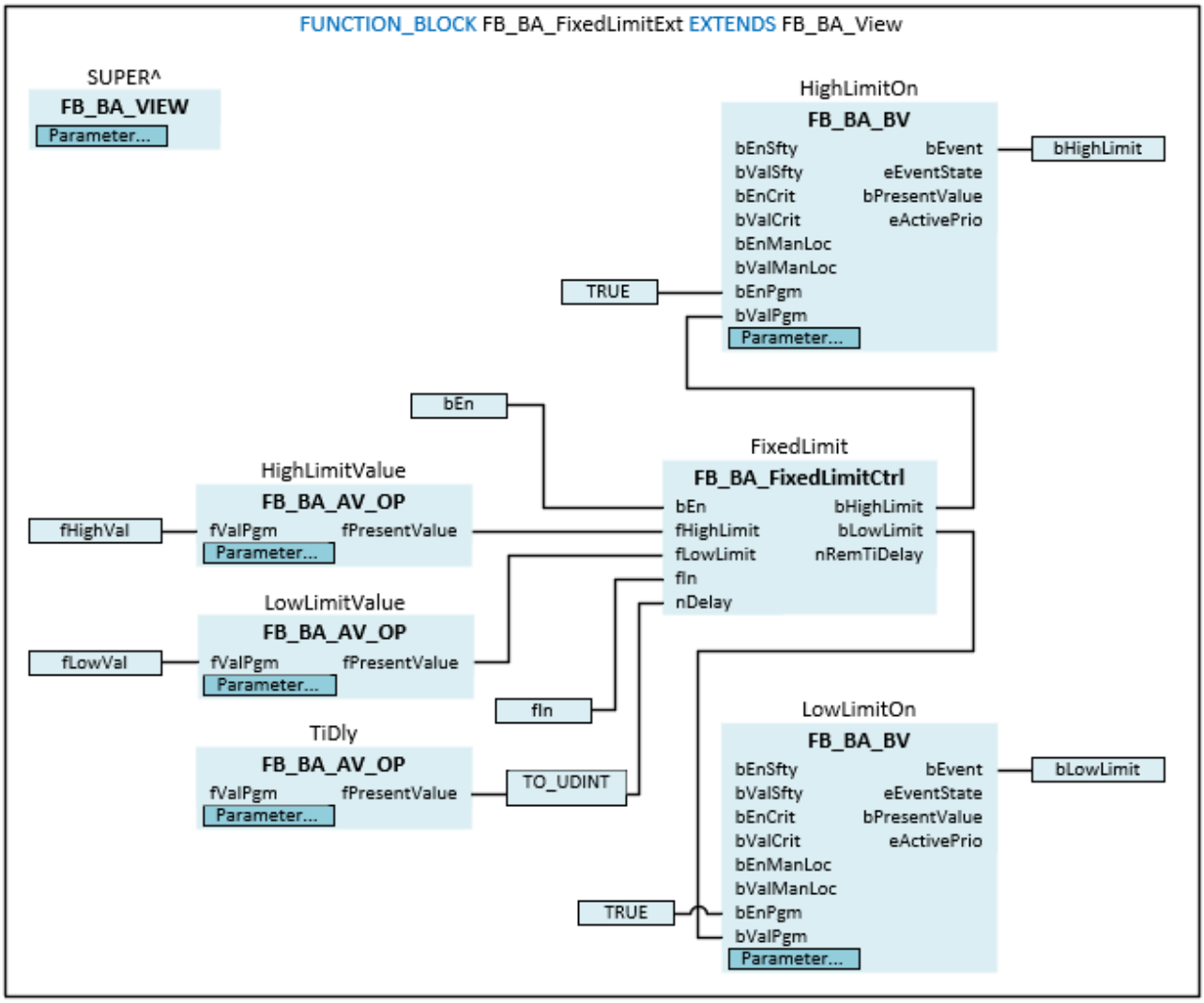
Eine Ansprechverzögerung des Ausgangs *bLowLimit* ist mit der Zeitvariable *TiDly* zu parametrieren.

Das binäre Objekt *LowLimitOn* dient zur Anzeige von *bLowLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_FixedLimitExt EXTENDS FB_BA_View
VAR_INPUT
  bEn                : BOOL;
  fIn                : REAL;
  fHighVal           : REAL;
  fLowVal            : REAL;
END_VAR
VAR_OUTPUT
  bHighLimit         : BOOL;
  bLowLimit          : BOOL;
END_VAR
VAR_INPUT CONSTANT
  HighLimitValue     : FB_BA_AV_Op;
  LowLimitValue      : FB_BA_AV_Op;
  TiDly              : FB_BA_AV_Op;
  HighLimitOn        : FB_BA_BV;
  LowLimitOn         : FB_BA_BV;
END_VAR
VAR
  FixedLimit         : FB_BA_FixedLimitCtrl;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
fIn	REAL	An dem Eingang wird der zu überwachende analoge Wert angeschlossen, z. B. ein Differenzdruckfühler.
fHighVal	REAL	An dem Eingang wird der obere Grenzwert des Toleranzbereiches angeschlossen.
fLowVal	REAL	An dem Eingang wird der untere Grenzwert des Toleranzbereiches angeschlossen.

Ausgänge

Name	Typ	Beschreibung
bHighLimit	BOOL	Der Ausgang zeigt an, dass der obere Grenzwert des Toleranzbereiches überschritten wurde.
bLowLimit	BOOL	Der Ausgang zeigt an, dass der untere Grenzwert des Toleranzbereiches unterschritten wurde.

Eingänge CONSTANT

Name	Typ	Beschreibung
HighLimitValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des oberen Grenzwertes des Toleranzbereiches.
LowLimitValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des unteren Grenzwertes des Toleranzbereiches.
TiDly	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Ansprechverzögerung der Ausgänge *bHighLimit* und *bLowLimit.*
HighLimitOn	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.
LowLimitOn	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.

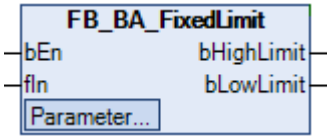
Variablen

Name	Typ	Beschreibung
FixedLimit	FB_BA_FixedLimitCtrl [▶ 478]	Kernstück des Templates.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.7.5 FB_BA_FixedLimit



Das Template stellt einen Grenzwertschalter mit festen Grenzen dar.

Um den zu überwachenden Wert *fIn* herum wird ein Toleranzbereich definiert.

Der Toleranzbereich ergibt sich aus einem oberen Grenzwert *HighLimitValue* und einem unteren Grenzwert *LowLimitValue*.

Übersteigt der Wert *fIn* den oberen Grenzwert des Toleranzbereiches, dann wird der Ausgang *bHighLimit* gesetzt.

Eine Ansprechverzögerung des Ausgangs *bHighLimit* ist mit der Zeitvariable *TiDly* zu parametrieren.

Das binäre Objekt *HighLimitOn* dient der Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.

Unterschreitet der Wert *fIn* den unteren Grenzwert des Toleranzbereiches, dann wird der Ausgang *bLowLimit* gesetzt.

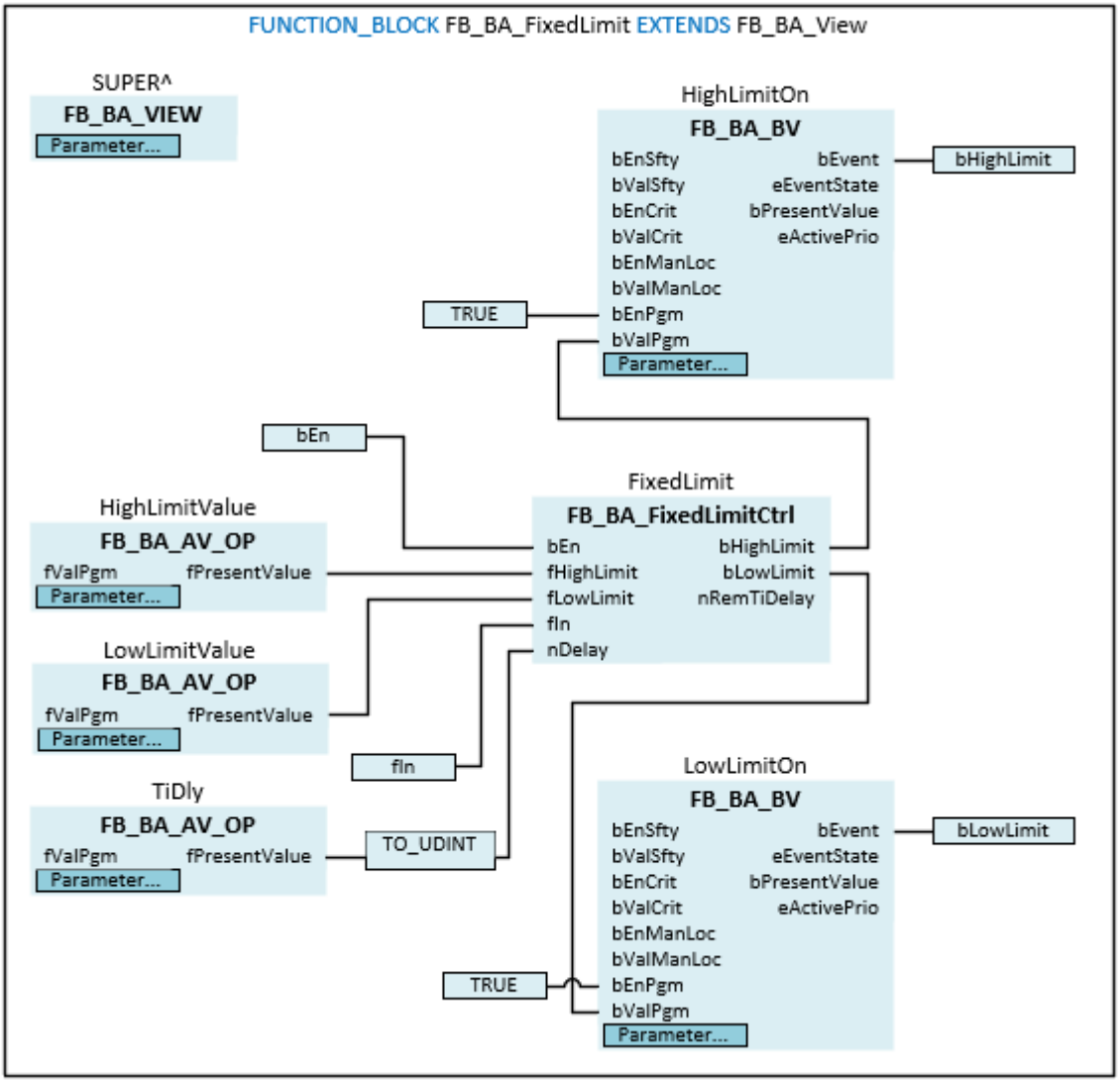
Eine Ansprechverzögerung des Ausgangs *bLowLimit* und ist mit der Zeitvariable *TiDly* zu parametrieren.

Das binäre Objekt *LowLimitOn* dient zur Anzeige von *bLowLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_FixedLimit EXTENDS FB_BA_View
VAR_INPUT
  bEn               : BOOL;
  fIn               : REAL;
END_VAR
VAR_OUTPUT
  bHighLimit        : BOOL;
  bLowLimit         : BOOL;
END_VAR
VAR_INPUT CONSTANT
  HighLimitValue    : FB_BA_AV_Op;
  LowLimitValue     : FB_BA_AV_Op;
  TiDly             : FB_BA_AV_Op;
  HighLimitOn       : FB_BA_BV;
  LowLimitOn        : FB_BA_BV;
END_VAR
VAR
  FixedLimit        : FB_BA_FixedLimitCtrl;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
fIn	REAL	An dem Eingang wird der zu überwachende analoge Wert angeschlossen, z. B. ein Differenzdruckfühler.

Ausgänge

Name	Typ	Beschreibung
bHighLimit	BOOL	Der Ausgang zeigt an, dass der obere Grenzwert des Toleranzbereiches überschritten wurde.
bLowLimit	BOOL	Der Ausgang zeigt an, dass der untere Grenzwert des Toleranzbereiches unterschritten wurde.

Eingänge CONSTANT

Name	Typ	Beschreibung
HighLimitValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des oberen Grenzwertes des Toleranzbereiches.
LowLimitValue	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe des unteren Grenzwertes des Toleranzbereiches.
TiDly	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Ansprechverzögerung der Ausgänge *bHighLimit* und *bLowLimit*.
HighLimitOn	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.
LowLimitOn	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.

Variablen

Name	Typ	Beschreibung
FixedLimit	FB_BA_FixedLimitCtrl [▶ 478]	Kernstück des Templates.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.7.6 FB_BA_PresenceMonitoring



Universelles Präsenzmeldetemplate mit Rücksetzeingängen für Verzögerungstimer und Handfunktion.

Über den Eingang *bPresence* wird eine ausschaltverzögerte Präsenzmeldung am Ausgang *bPresenceState* ausgegeben.

Die Ausschaltverzögerung ist durch *nDlyPrc* [s] definiert. Nach Ablauf dieser Zeit wird nicht nur *bPresenceState* wieder auf FALSE gesetzt, sondern auch ein TRUE-Impuls am Ausgang *bRstSwi*. Dieser Impuls kann zur Rücksetzung von Handübersteuerungen beispielsweise an Jalousie- oder Lichtfunktionen genutzt werden.

Wird am Eingang *bRstDelayTimer* ein TRUE-Signal gegeben, so wird der Verzögerungstimer abgelöscht und der Ausgang *bPresenceState* geht auf FALSE, bis durch den Eingang *bPresence* wieder Präsenz erkannt wird.

Ein TRUE-Signal am Eingang *bRstManMod* triggert den Ausgang *bRstSwi* gezielt.

Beide Reset-Funktionen sind für eine zentrale Abschaltung, bei der davon ausgegangen wird, dass niemand mehr am Platz ist, wichtig.

Am Ausgang *nCountdownPresence* kann zu Inbetriebnahme Zwecken die verbleibende Zeit des Verzögerungs-Timers in Sekunden abgelesen werden.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_PresenceMonitoring
VAR_INPUT
  bPresence             : BOOL;
  bRstDelayTimer        : BOOL;
  bRstManMod            : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nDlyPrc               : UDINT;
END_VAR
VAR_OUTPUT
  bRstSwi               : BOOL;
  bPresenceState        : BOOL;
  nCountdownPresence    : UDINT;
END_VAR
VAR
  tofPrcDetc            : TOF;
  rtRstDelayTimer       : R_TRIG;
  rtRstManMod           : R_TRIG;
  ftPrc                 : F_TRIG;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bPresence	BOOL	Präsenzmeldeeingang, der ausschaltverzögert an den Ausgang *bPresenceState* weitergegeben wird.
bRstDelayTimer	BOOL	Rücksetzeingang für die Ausschaltverzögerung. Ein TRUE-Signal an diesem Eingang setzt den internen Timer zurück.
bRstManMode	BOOL	Rücksetzeingang für die Handübersteuerung. Ein TRUE-Signal an diesem Eingang erzeugt eine positive Flanke am Ausgang *bRstSwi*.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nDlyPrc	UDINT	Ausschaltverzögerungszeit [s]. In *FB_Init* voreingestellt auf 3600.

Ausgänge

Name	Typ	Beschreibung
bRstSwi	BOOL	Rücksetzausgang für Handübersteuerungen.
bPresenceState	BOOL	Abschaltverzögerte Präsenzmeldung.
nCountDownPresence	UDINT	Verbleibende Zeit des Verzögerungs-Timers in Sekunden.

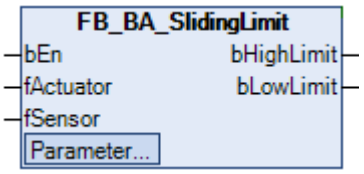
Variablen

Name	Typ	Beschreibung
tofPrcDetc	TOF	Ausschaltverzögerung Präsenz.
rtRstDelayTimer	R_TRIG	Trigger für den Reset des Timers.
rtRstManMod	R_TRIG	Trigger für den Reset der Handübersteuerung.
ftPrc	F_TRIG	Trigger für Reset der Handübersteuerung, jedoch durch den Wegfall der Präsenz gesteuert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.7.7 FB_BA_SlidingLimit



Das Template stellt eine gleitende Grenzwertüberwachung dar.

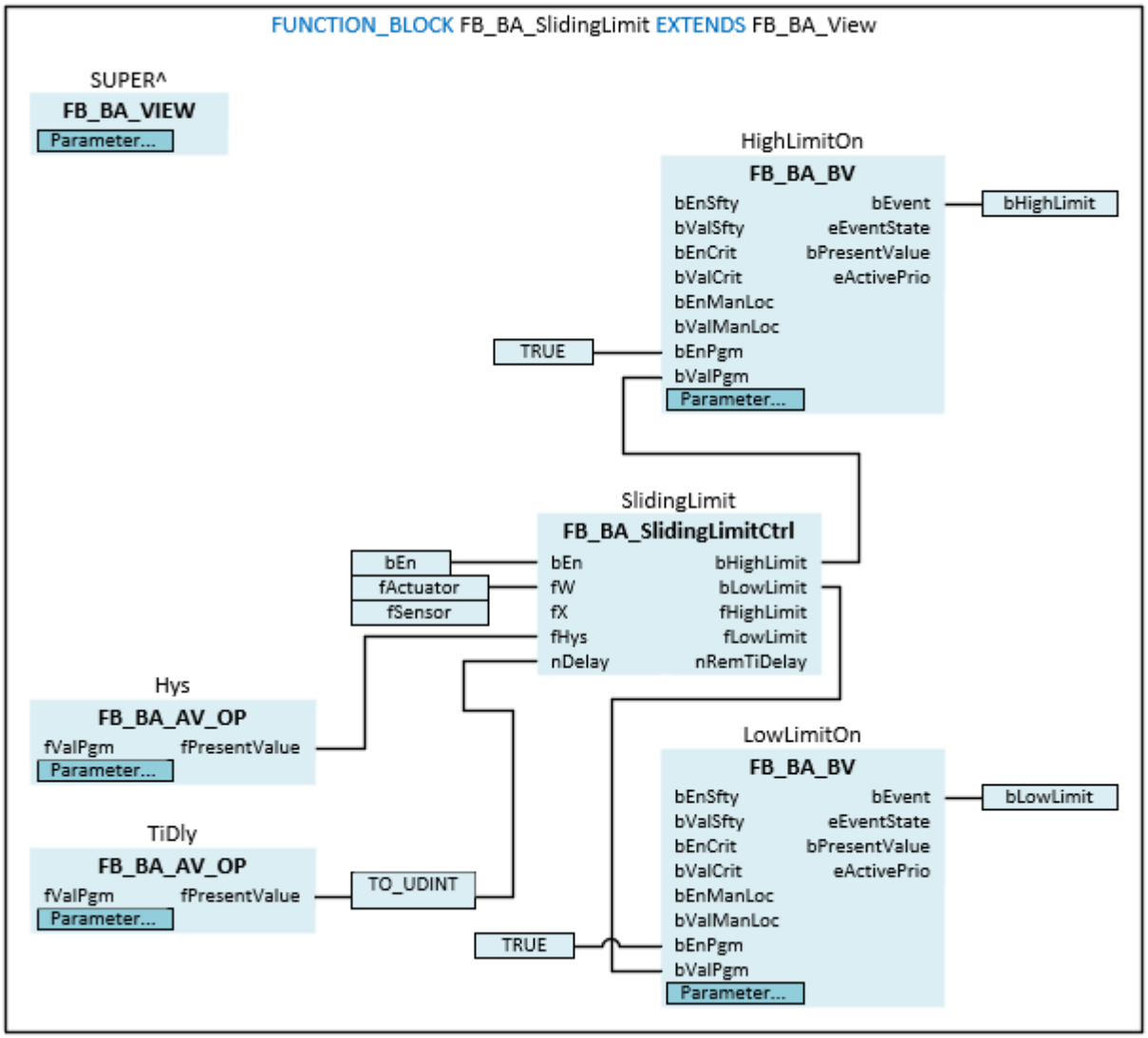
Nach dem Start wird zunächst geprüft, ob der Istwert *fSensor* der Regelung innerhalb des Toleranzbereiches zwischen dem unteren Grenzwert *fLowLimit* und dem oberen Grenzwert *fHighLimit* des Funktionsbausteines *SlidingLimit* liegt. Liegt der Istwert außerhalb dieses Toleranzbereiches und die Verzögerungszeit *TiDly* ist abgelaufen, dann wird je nach Verlassen des Toleranzbereiches eine der Variablen *bHighLimit* oder *bLowLimit* gesetzt.

Das binäre Objekt *HighLimitOn* dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.

Das binäre Objekt *LowLimitOn* dient zur Anzeige von *bLowLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_SlidingLimit EXTENDS FB_BA_View
VAR_INPUT
  bEn              : BOOL;
  fActuator        : REAL;
  fSensor          : REAL;
END_VAR
VAR_OUTPUT
  bHighLimit       : BOOL;
  bLowLimit        : BOOL;
END_VAR
VAR_INPUT CONSTANT
  Hys              : FB_BA_AV_Op;
  TiDly            : FB_BA_AV_Op;
  HighLimitOn      : FB_BA_BV;
  LowLimitOn       : FB_BA_BV;
END_VAR
```

```
VAR
  SlidingLimit    : FB_BA_SlidingLimitCtrl;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
fActuator	REAL	An dem Eingang wird der Basiswert des Toleranzbereiches angeschlossen. Dieser Wert könnte z. B. eine Ventilstellung sein.
fSensor	REAL	An dem Eingang wird der zu überwachende analoge Wert angeschlossen. Dieser Wert könnte z.B. das Rückmeldesignal eines Ventils sein.

Ausgänge

Name	Typ	Beschreibung
bHighLimit	BOOL	Der Ausgang zeigt an, dass der obere Grenzwert des Toleranzbereiches überschritten wurde.
bLowLimit	BOOL	Der Ausgang zeigt an, dass der untere Grenzwert des Toleranzbereiches unterschritten wurde.

Eingänge CONSTANT

Name	Typ	Beschreibung
Hys	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Hysterese des Toleranzbereiches. Unterer Grenzwert des Funktionsbausteines SlidingLimit: *fLowLimit* = *fActuator* - (*Hys* / 2) Oberer Grenzwert des Funktionsbausteines SlidingLimit: *fHighLimit* = *fActuator* + (*Hys* / 2)
TiDly	FB_BA_AV_Op [▶ 207]	Analog-Wert-Objekt zur Eingabe der Ansprechverzögerung der Ausgänge *bHighLimit* und *bLowLimit.*
HighLimitOn	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.
LowLimitOn	FB_BA_BV [▶ 218]	Das binäre Objekt dient zur Anzeige von *bHighLimit* und kann durch Änderung der Parametrierung als Störmeldeobjekt verwendet werden.

Variablen

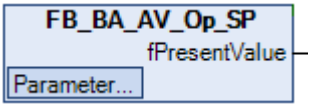
Name	Typ	Beschreibung
SlidingLimit	FB_BA_SlidingLimitCtrl [▶ 480]	Kernstück des Templates.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.8 Object

6.1.4.2.2.8.1 FB_BA_AV_Op_SP



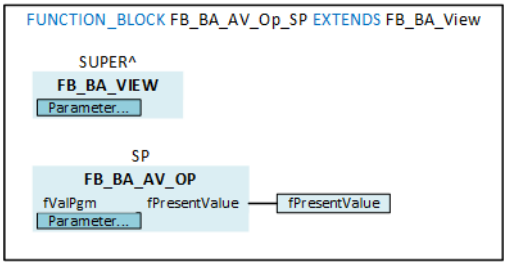
Das Template FB_BA_AV_Op_SP dient der Eingabe eines Sollwertes innerhalb eines Templates auf Anlagenebene.

Es stellt selbst eine Hülle dar, welche die Aggregat-Ebene repräsentiert und das innenliegende Analogobjekt auf die Funktionsebene setzt.

Im *FB_Init* vordefiniert ist ein Label, der das innenliegende Analogobjekt [▶ 207] als "Sollwert" beschreibt, die vorgewählte Einheit ist [°C].

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_AV_Op_SP EXTENDS FB_BA_View
VAR_OUTPUT
  fPresentValue    : REAL;
END_VAR
VAR_INPUT CONSTANT
  SP               : FB_BA_AV_Op;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Eingegebener Wert.

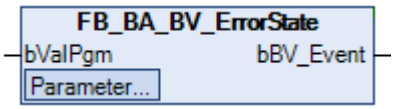
Eingänge CONSTANT

Name	Typ	Beschreibung
SP	FB_BA_AV_Op [▶ 207]	Analogwert-Objekt zur Eingabe des Sollwertes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.8.2 FB_BA_BV_ErrorState



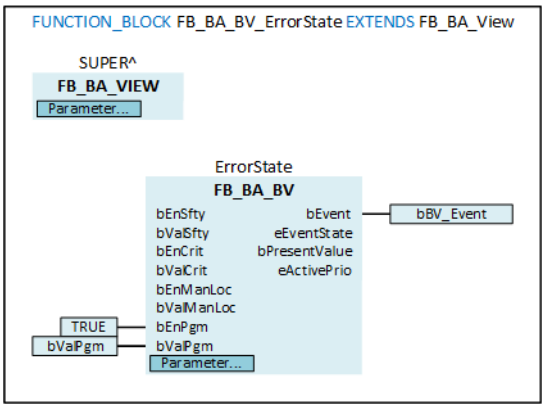
Das Template ist eine Fehlerzustandsmeldung innerhalb eines Templates auf Anlagenebene.

Es stellt selbst eine Hülle dar, welche die Aggregat-Ebene repräsentiert und das innenliegende Binärobjekt [▶ 218] auf Funktionsebene setzt.

Im *FB_Init* vordefiniert ist ein Label, der das innenliegende Binärobjekt als "Fehler" beschreibt.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_BV_ErrorState EXTENDS FB_BA_View
VAR_INPUT
  bValPgm       : BOOL;
END_VAR
VAR_INPUT CONSTANT
  ErrorState    : FB_BA_BV;
END_VAR
VAR_OUTPUT
  bBV_Event     : BOOL;
END_VAR
```

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
bValPgm	BOOL	(Fehler-) Eingang aus dem Programm.

VAR_INPUT CONSTANT

Name	Typ	Beschreibung
ErrorState	FB_BA_BV [▶ 218]	Binärwert-Objekt zur Darstellung der Fehlermeldung.

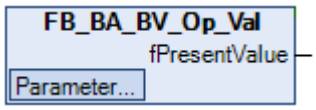
Ausgänge

Name	Typ	Beschreibung
bBVEvent	BOOL	Binärer Zustand der Meldung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.56	TF8040 | TwinCAT Building Automation ab V5.8.0.0

6.1.4.2.2.8.3 FB_BA_BV_Op_Val



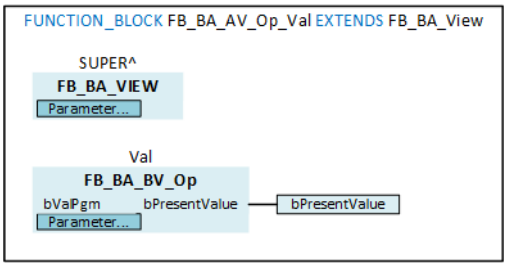
Das Template FB_BA_BV_OP_Val ist ein Schalter innerhalb eines Templates auf der Anlagenebene.

Es stellt selbst eine Hülle dar, welche die Aggregat-Ebene repräsentiert und das innenliegende Binärobjekt [▶ 221] auf der Funktionsebene setzt.

Im *FB_Init* vordefiniert ist ein Label, der das innenliegende Binärobjekt als "Taster" beschreibt, die vorgewählte Funktion ist "rastend" (E_BA_ToggleMode.eSwitch).

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_BV_OP_Val EXTENDS FB_BA_View
VAR_OUTPUT
  fPresentValue    : BOOL;
END_VAR
VAR_INPUT CONSTANT
  Val              : FB_BA_BV_Op;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Binärer Zustand des Schalters.

Eingänge CONSTANT

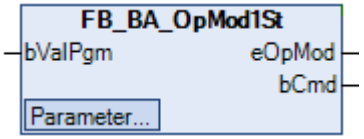
Name	Typ	Beschreibung
Val	FB_BA_BV_Op [▶ 221]	Binärwert-Objekt zur Darstellung des Schalters.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.9 OperatingMode

6.1.4.2.2.2.9.1 FB_BA_OpMod1St

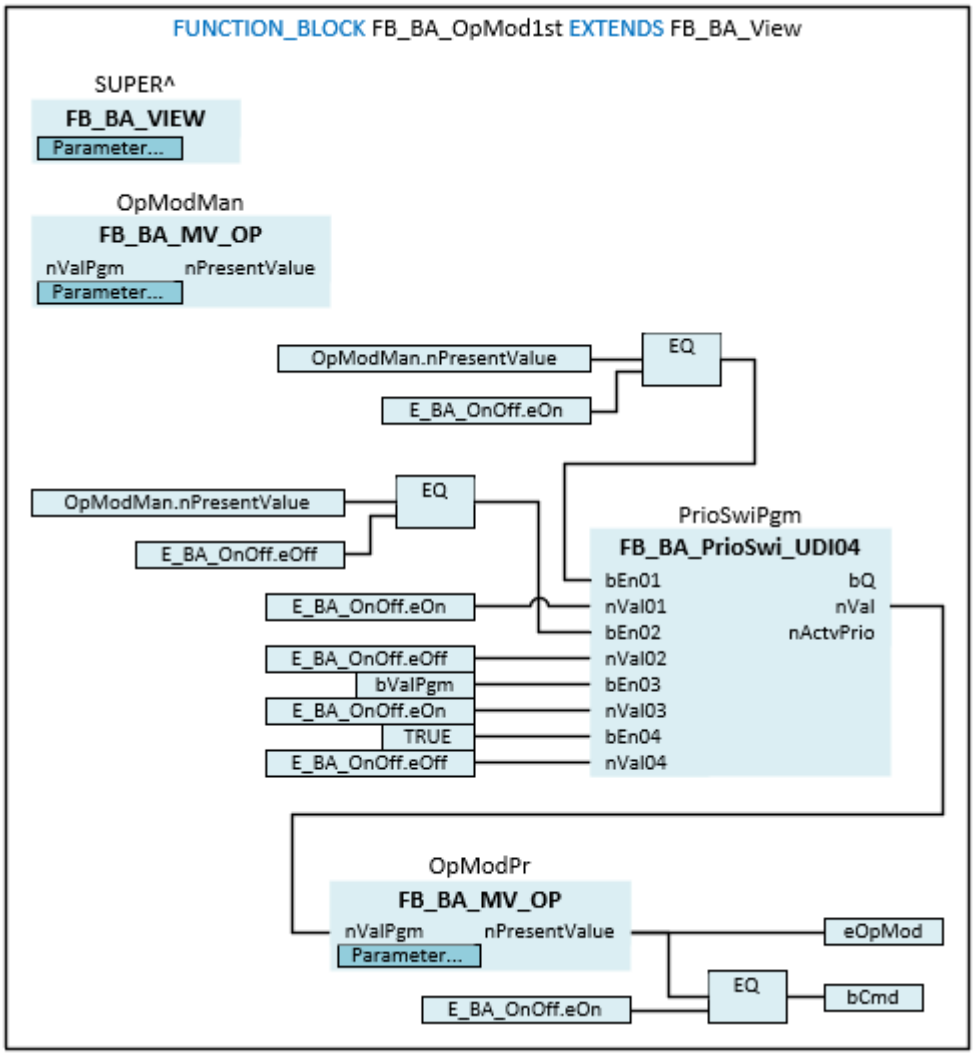


Das Template bildet einen einstufigen Anlagenwahlschalter ab mit den Betriebsarten „Auto“, „Hand Aus“ und „Hand Ein“.

Über den Eingang *bValPgm* kann in der Betriebsart Auto eine Anlage ein- oder ausgeschaltet werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_OpMod1St EXTENDS FB_BA_View
VAR_INPUT
  bValPgm      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  OpModMan     : FB_BA_MV_Op;
  OpModPr      : FB_BA_MV_Op;
```

```
END_VAR
VAR_OUTPUT
  eOpMod       : E_BA_OnOff;
  bCmd         : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bValPgm	BOOL	Über den Eingang kann in der Betriebsart „Auto“ eine Anlage ein- oder ausgeschaltet werden.

Eingänge CONSTANT

Name	Typ	Beschreibung
OpModMan	FB_BA_MV_Op [▶ 244]	Das Multistate Value Objekt repräsentiert einen Betriebsartenschalter mit den Betriebsarten „Auto“, „Hand Aus“ und „Hand Ein“.
OpModPr	FB_BA_MV_Op [▶ 244]	Das Multistate Value Objekt zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.

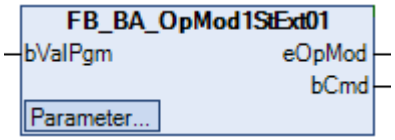
Ausgänge

Name	Typ	Beschreibung
eOpMod	E_BA_OnOff [▶ 713]	Zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.
bCmd	BOOL	Der Ausgang zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.9.2 FB_BA_OpMod1StExt01

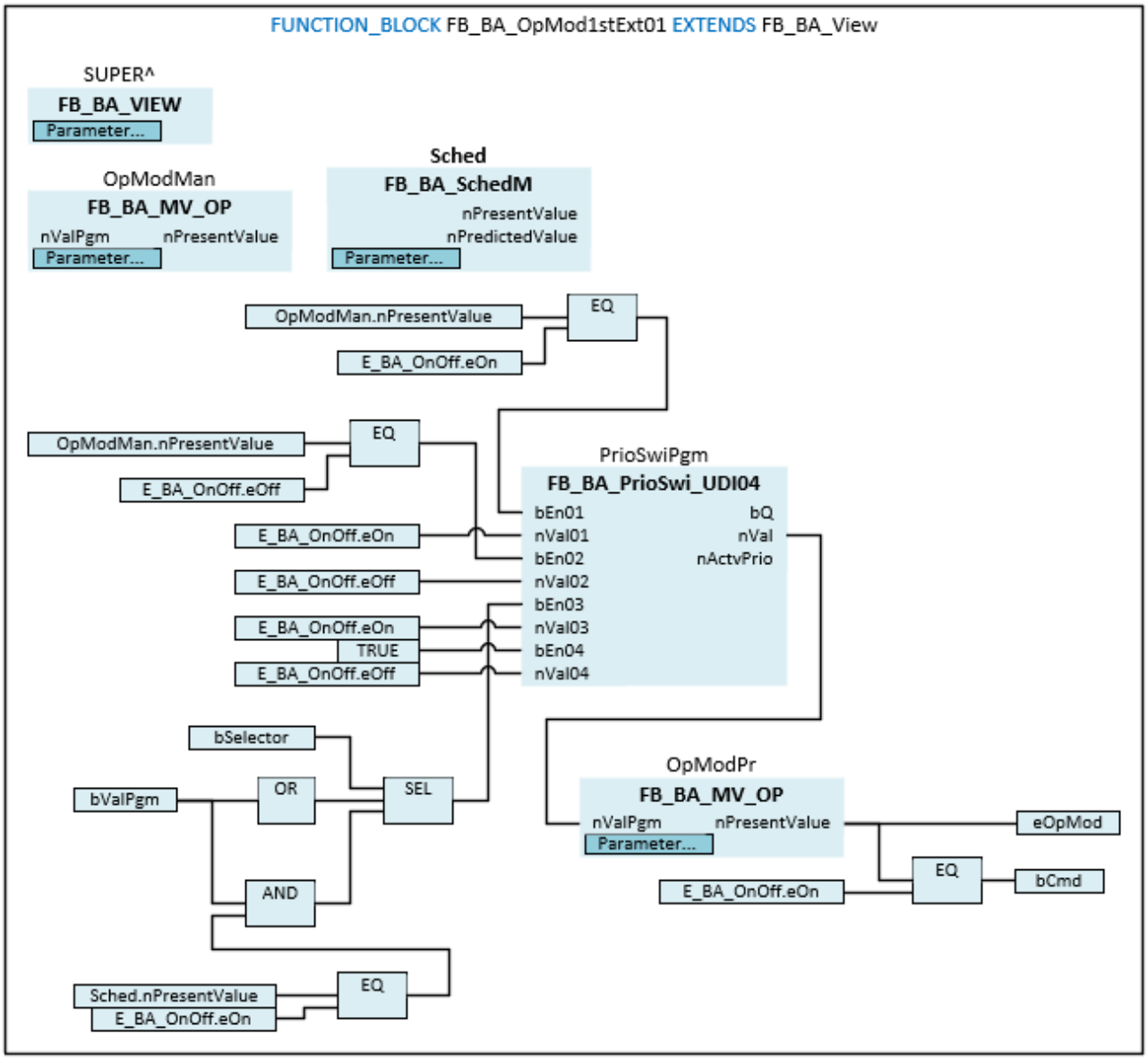


Das Template bildet einen einstufigen Anlagenwahlschalter ab mit den Betriebsarten „Auto“, „Hand Aus“ und „Hand Ein“.

Mittels der Variable *bSelector* kann in der Betriebsart „Auto“ eine Anlage so parametriert werden, dass entweder über den Eingang *bValPgm* oder über den Zeitschaltplan *Sched* die Anlage ein- oder ausgeschaltet wird.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_OpMod1St EXTENDS FB_BA_View
VAR_INPUT
  bValPgm          : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  bSelector        : BOOL;
END_VAR
VAR_INPUT CONSTANT
  OpModMan         : FB_BA_MV_Op;
  Sched            : FB_BA_SchedM;
  OpModPr          : FB_BA_MV_Op;
END_VAR
VAR_OUTPUT
  eOpMod           : E_BA_OnOff;
  bCmd             : BOOL;
END_VAR
VAR
  PrioSwiOpMod     : FB_BA_PrioSwi_UDI04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bValPgm	BOOL	Über den Eingang kann in der Betriebsart „Auto“ eine Anlage ein- oder ausgeschaltet werden.

Eingänge CONSTANT

Name	Typ	Beschreibung
bSelector	BOOL	Mittels der Variable kann in der Betriebsart „Auto“ eine Anlage so parametriert werden, dass entweder über den Eingang *bValPgm* oder über den Zeitschaltplan *Sched* die Anlage ein- oder ausgeschaltet wird.

Eingänge CONSTANT

Name	Typ	Beschreibung
OpModMan	FB_BA_MV_Op [▶ 244]	Das Multistate Value Objekt repräsentiert einen Betriebsartenschalter mit den Betriebsarten „Auto“, „Hand Aus“ und „Hand Ein“.
Sched	FB_BA_SchedM [▶ 230]	Über den Zeitschaltplan kann in der Betriebsart „Auto“ und den Parameter *bSelector* = TRUE eine Anlage ein- oder ausgeschaltet werden.
OpModPr	FB_BA_MV_Op [▶ 244]	Das Multistate Value Objekt zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.

Ausgänge

Name	Typ	Beschreibung
eOpMod	E_BA_OnOff [▶ 713]	Zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.
bCmd	BOOL	Der Ausgang zeigt den Zustand der aktuell gültigen Anlagenbetriebsart an.

Variablen

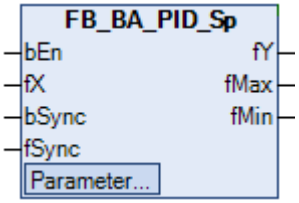
Name	Typ	Beschreibung
PrioSwiOpMod	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter erfasst die Betriebsarten, priorisiert diese und gibt das Ergebnis an *OpModPr* weiter.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.10 PID

6.1.4.2.2.2.10.1 FB_BA_PID_Sp



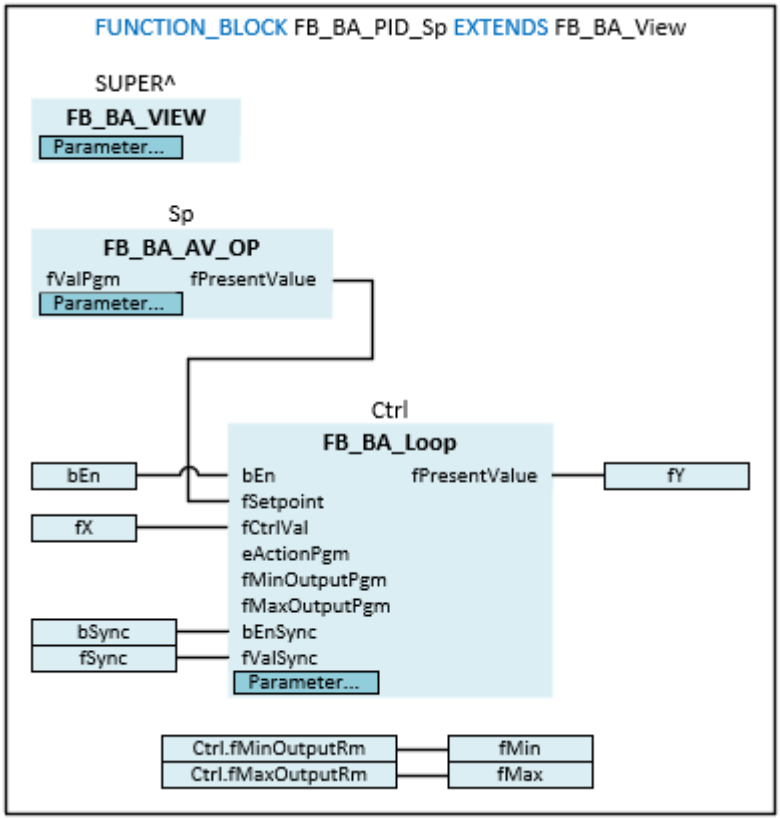
Das Template ist ein universeller PID-Regler.

Die Freigabe des PID-Reglers erfolgt anhand der Eingangsvariable *bEn*.

Der Sollwert wird über das AV-Objekt *Sp* eingegeben.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_PID EXTENDS FB_BA_View
VAR_INPUT
  bEn       : BOOL;
  fX        : REAL;
  bSync     : BOOL;
  fSync     : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
  fMax      : REAL;
  fMin      : REAL;
END_VAR
VAR_INPUT CONSTANT
  Sp        : FB_BA_AV_Op;
  Ctrl      : FB_BA_Loop;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates
fX	REAL	Istwert
bSync	BOOL	Bei einer steigenden Flanke an diesem Eingang wird das Loop-Objekt auf den Wert von *fSync* aufsynchronisiert.
fSync	REAL	Synchronisationswert

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Ausgabe der Stellgröße
fMax	REAL	Maximalwert des Reglers
fMin	REAL	Minimalwert des Reglers

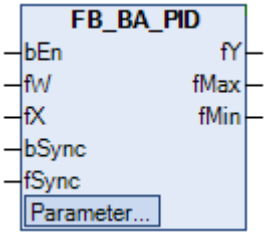
Eingänge CONSTANT

Name	Typ	Beschreibung
Sp	FB_BA_AV_Op [▶ 207]	Eingabe des Sollwerts
Ctrl	FB_BA_Loop [▶ 225]	PID-Regler

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.10.2 FB_BA_PID

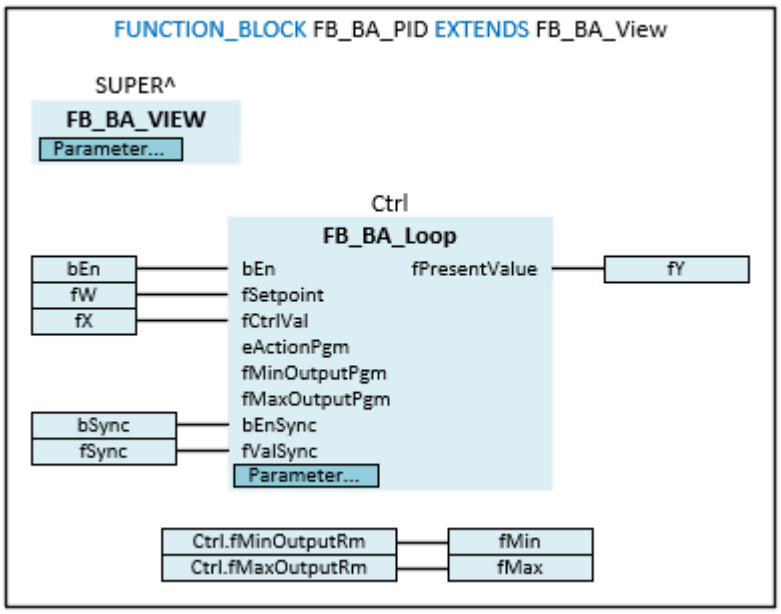


Das Template ist ein universeller PID-Regler.

Die Freigabe des PID-Reglers erfolgt anhand der Eingangsvariable *bEn*.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_PID EXTENDS FB_BA_View
VAR_INPUT
  bEn       : BOOL;
  fW        : REAL;
  fX        : REAL;
  bSync     : BOOL;
  fSync     : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
  fMax      : REAL;
  fMin      : REAL;
END_VAR
VAR_INPUT CONSTANT
  Ctrl      : FB_BA_Loop;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates
fW	REAL	Sollwert
fX	REAL	Istwert
bSync	BOOL	Bei einer steigenden Flanke an diesem Eingang wird das Loop-Objekt auf den Wert von *fSync* aufsynchronisiert.
fSync	REAL	Synchronisationswert

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Ausgabe der Stellgröße
fMax	REAL	Maximalwert des Reglers
fMin	REAL	Minimalwert des Reglers

Eingänge CONSTANT

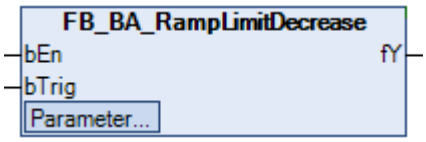
Name	Typ	Beschreibung
Ctrl	FB_BA_Loop [▶ 225]	PID-Regler

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.11 Ramp

6.1.4.2.2.2.11.1 FB_BA_RampLimitDecrease

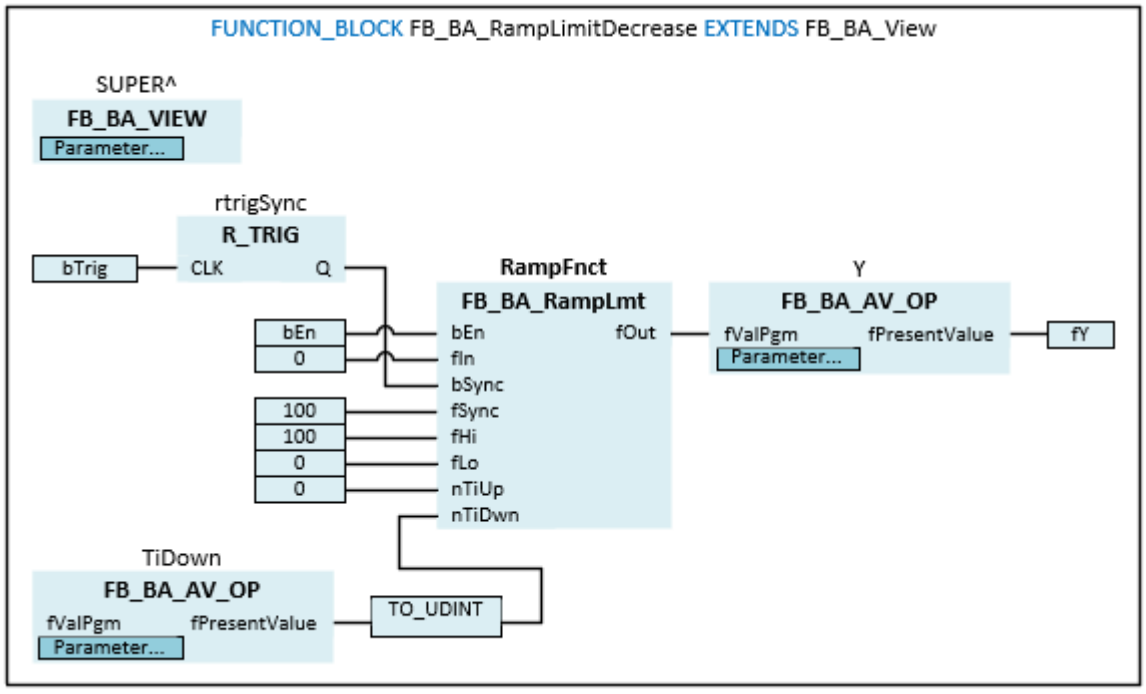


Das Template realisiert eine fallende Rampenbegrenzung von 100 nach 0 und bleibt auf diesem Wert stehen.

Die Rampenbegrenzung wird ausgelöst durch eine steigende Flanke an dem Eingang *bTrig* und zeitlich durch *TiDown* vorgegeben.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_RampLimitDecrease EXTENDS FB_BA_View
VAR_INPUT
  bEn           : BOOL;
  bTrig         : BOOL;
END_VAR
```

```
VAR_OUTPUT
  fY            : REAL;
END_VAR
VAR_INPUT CONSTANT
  TiDown        : FB_BA_AV_Op;
  Y             : FB_BA_AV_Op;
END_VAR
VAR
  RampFnct      : FB_BA_RampLmt;
  rtrigSync     : R_TRIG;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
bTrig	BOOL	Durch eine steigende Flanke an diesem Eingang wird die Rampenbegrenzung aktiviert.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Ausgang des Rampenbegrenzungswertes.

Eingänge CONSTANT

Name	Typ	Beschreibung
TiDown	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe der Abfallzeit der Rampenbegrenzung.
Y	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Anzeige des Rampenbegrenzungswertes.

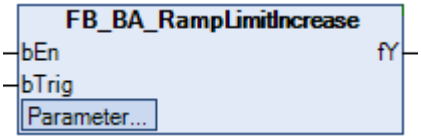
Variablen

Name	Typ	Beschreibung
RampFnct	FB_BA_RampLmt	Der Funktionsbaustein zur Ausgabe einer Rampenbegrenzung ist das Kernstück dieses Templates.
rtrigSync	R_TRIG	Der Funktionsbaustein löst eine steigende Flanke aus und aktiviert die Rampenbegrenzung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.11.2 FB_BA_RampLimitIncrease

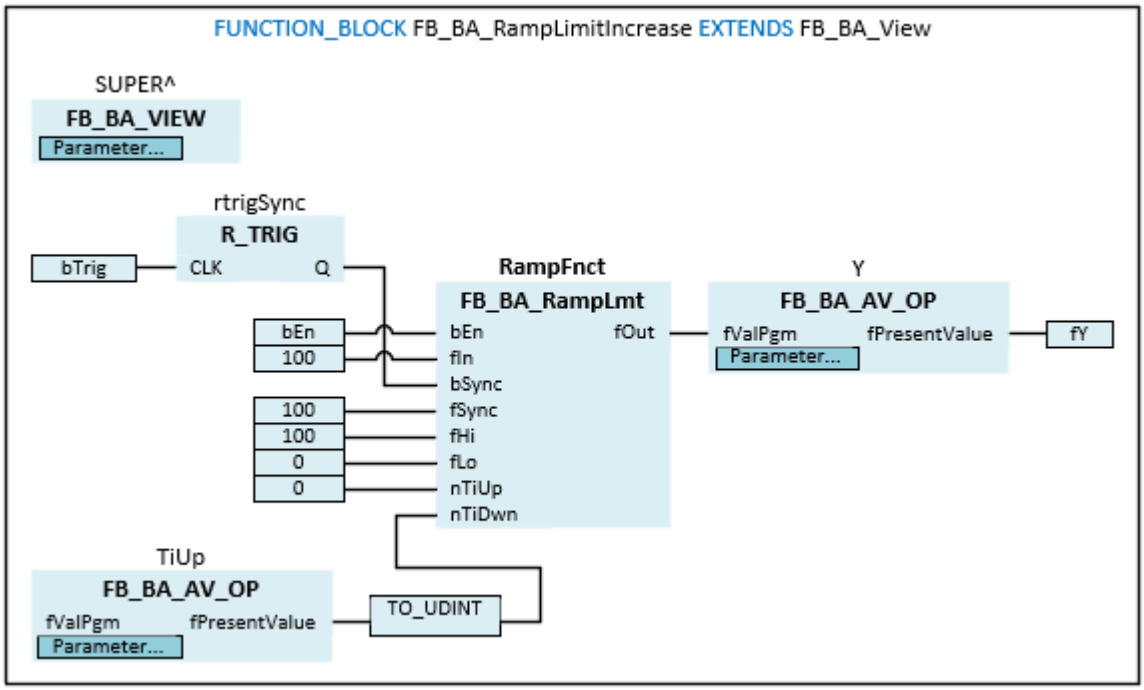


Das Template realisiert eine steigende Rampenbegrenzung von 0 nach 100 und bleibt auf diesem Wert stehen.

Die Rampenbegrenzung wird ausgelöst durch eine steigende Flanke an dem Eingang *bTrig* und zeitlich durch *TiUp* vorgegeben.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_RampLimitIncrease EXTENDS FB_BA_View
VAR_INPUT
  bEn          : BOOL;
  bTrig        : BOOL;
END_VAR
VAR_OUTPUT
  fY           : REAL;
END_VAR
VAR_INPUT CONSTANT
  TiUp         : FB_BA_AV_Op;
  Y            : FB_BA_AV_Op;
END_VAR
VAR
  RampFnct     : FB_BA_RampLmt;
  rtrigSync    : R_TRIG;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Allgemeine Freigabe des Templates.
bTrig	BOOL	Durch eine steigende Flanke an diesem Eingang wird die Rampenbegrenzung aktiviert.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Ausgang des Rampenbegrenzungswertes.

Eingänge CONSTANT

Name	Typ	Beschreibung
TiUp	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe der Zeit zum Steigen der Rampenbegrenzung.
Y	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Anzeige des Rampenbegrenzungswertes.

Variablen

Name	Typ	Beschreibung
RampFnct	FB_BA_RampLmt	Der Funktionsbaustein zur Ausgabe einer Rampenbegrenzung ist das Kernstück dieses Templates.
rtrigSync	R_TRIG	Der Funktionsbaustein löst eine steigende Flanke aus und aktiviert die Rampenbegrenzung.

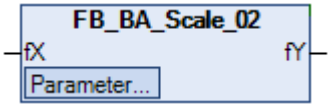
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.12 Scale

Templates zur Skalierung von Analogwerten.

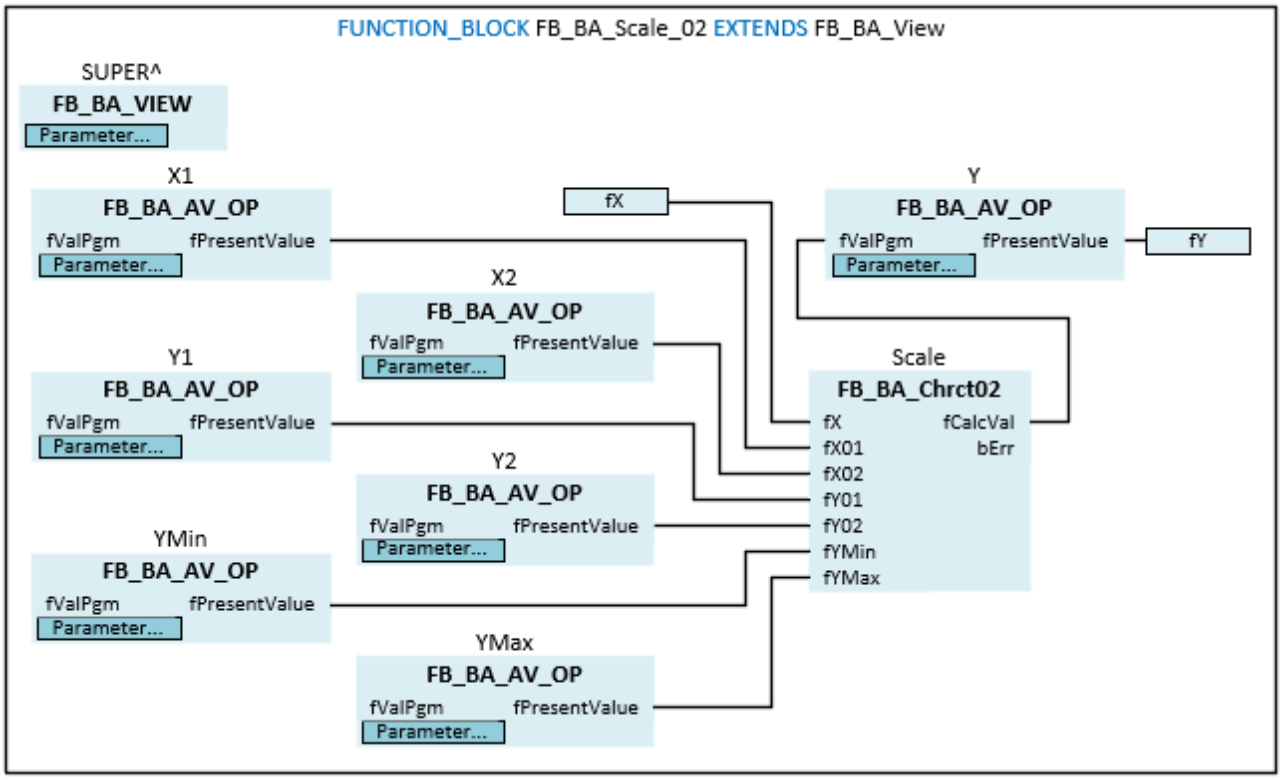
6.1.4.2.2.2.12.1 FB_BA_Scale_02



Das Template stellt eine Linear-Interpolation mit zwei Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*X1/Y1*] bis [*X2/Y2*] bestimmt. Der errechnete Ausgangswert *fY* wird durch *YMin* bzw. *YMax* limitiert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
  fX        : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
END_VAR
VAR_INPUT CONSTANT
  X1        : FB_BA_AV_Op;
  X2        : FB_BA_AV_Op;
  Y1        : FB_BA_AV_Op;
  Y2        : FB_BA_AV_Op;
  YMin      : FB_BA_AV_Op;
  YMax      : FB_BA_AV_Op;
  Y         : FB_BA_AV_Op;
END_VAR
VAR
  Scale     : FB_BA_Chrct02;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Errechneter Ausgangswert der Kennlinie.

Eingänge CONSTANT

Name	Typ	Beschreibung
X1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X1 vorgegeben.
X2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X2 vorgegeben.
Y1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y1 vorgegeben.
Y2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y2 vorgegeben.
YMin	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die minimale Begrenzung von *fY* vorgegeben.
YMax	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die maximale Begrenzung von *fY* vorgegeben.
Y	FB_BA_AV_Op [▶ 207]	Ausgabe errechneter Ausgangswert der Kennlinie.

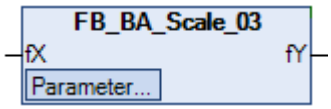
Variablen

Name	Typ	Beschreibung
Scale	FB_BA_Chrct02	Der Baustein stellt eine Linear-Interpolation mit zwei Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

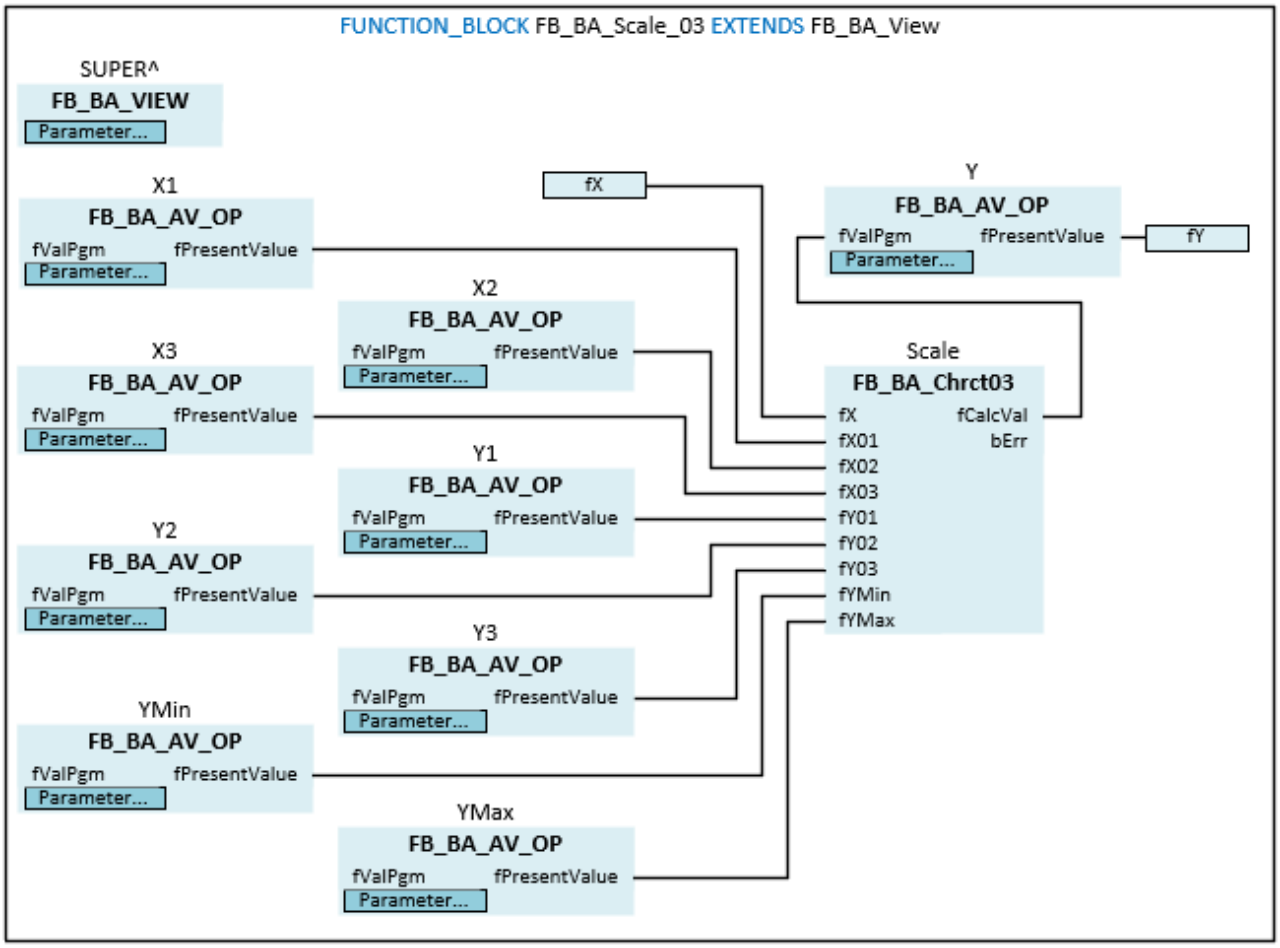
6.1.4.2.2.12.2 FB_BA_Scale_03



Das Template stellt eine Linear-Interpolation mit drei Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*X1/Y1*] bis [*X3/Y3*] bestimmt. Der errechnete Ausgangswert *fY* wird durch *YMin* bzw. *YMax* limitiert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
  fX        : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
END_VAR
VAR_INPUT CONSTANT
  X1        : FB_BA_AV_Op;
  X2        : FB_BA_AV_Op;
  X3        : FB_BA_AV_Op;
  Y1        : FB_BA_AV_Op;
  Y2        : FB_BA_AV_Op;
  Y3        : FB_BA_AV_Op;
  YMin      : FB_BA_AV_Op;
  YMax      : FB_BA_AV_Op;
  Y         : FB_BA_AV_Op;
END_VAR
VAR
  Scale     : FB_BA_Chrct03;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Errechneter Ausgangswert der Kennlinie.

Eingänge CONSTANT

Name	Typ	Beschreibung
X1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X1 vorgegeben.
X2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X2 vorgegeben.
X3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X3 vorgegeben.
Y1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y1 vorgegeben.
Y2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y2 vorgegeben.
Y3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y3 vorgegeben.
YMin	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die minimale Begrenzung von *fY* vorgegeben.
YMax	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die maximale Begrenzung von *fY* vorgegeben.
Y	FB_BA_AV_Op [▶ 207]	Ausgabe errechneter Ausgangswert der Kennlinie.

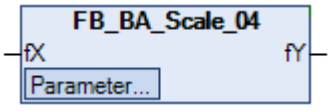
Variablen

Name	Typ	Beschreibung
Scale	FB_BA_Chrct03	Der Baustein stellt eine Linear-Interpolation mit drei Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

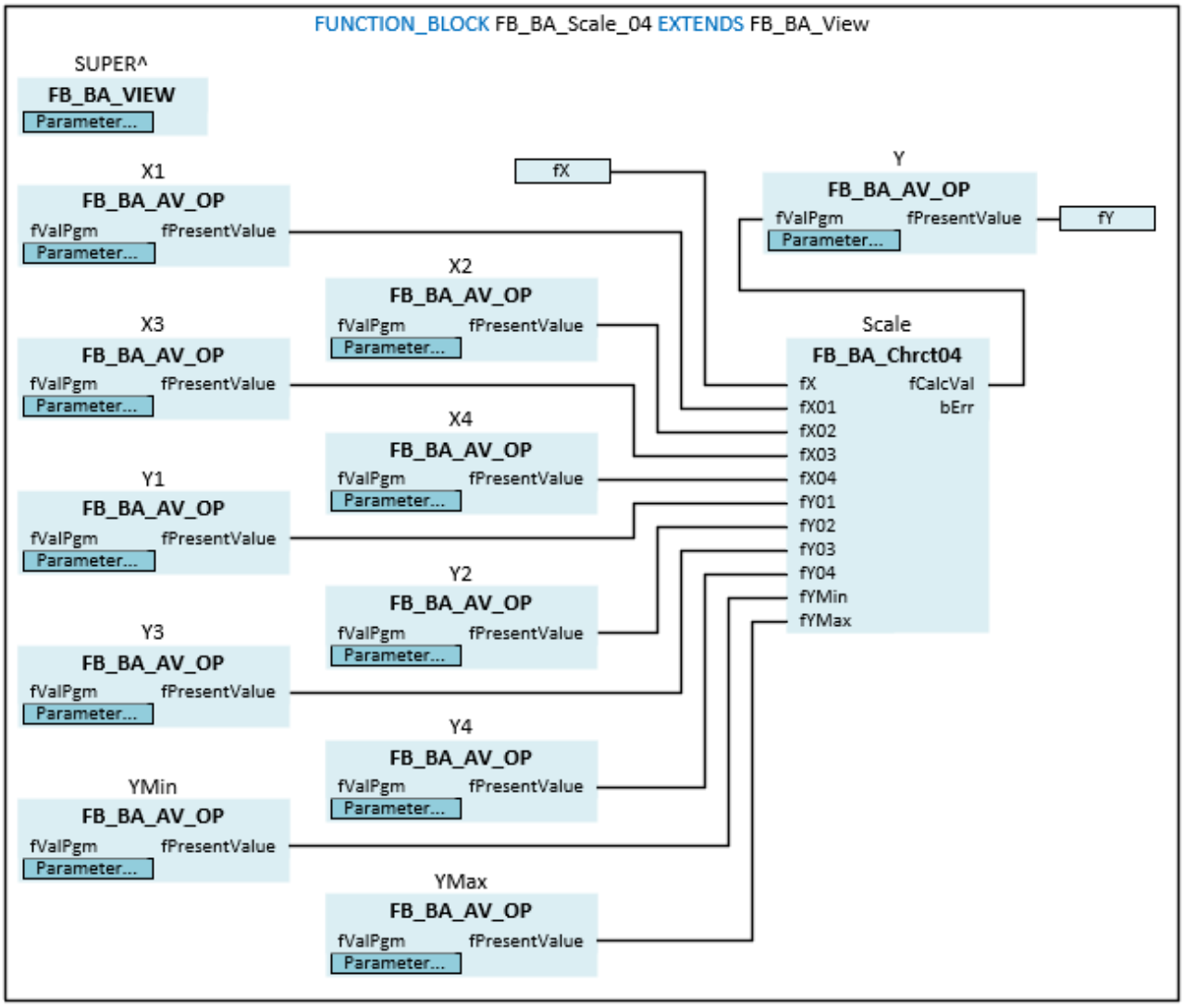
6.1.4.2.2.2.12.3 FB_BA_Scale_04



Das Template stellt eine Linear-Interpolation mit vier Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*X1/Y1*] bis [*X4/Y4*] bestimmt. Der errechnete Ausgangswert *fY* wird durch *YMin* bzw. *YMax* limitiert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
  fX        : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
END_VAR
VAR_INPUT CONSTANT
  X1        : FB_BA_AV_Op;
  X2        : FB_BA_AV_Op;
  X3        : FB_BA_AV_Op;
  X4        : FB_BA_AV_Op;
  Y1        : FB_BA_AV_Op;
  Y2        : FB_BA_AV_Op;
  Y3        : FB_BA_AV_Op;
  Y4        : FB_BA_AV_Op;
  YMin      : FB_BA_AV_Op;
  YMax      : FB_BA_AV_Op;
  Y         : FB_BA_AV_Op;
END_VAR
VAR
  Scale     : FB_BA_Chrct04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Errechneter Ausgangswert der Kennlinie.

Eingänge CONSTANT

Name	Typ	Beschreibung
X1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X1 vorgegeben.
X2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X2 vorgegeben.
X3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X3 vorgegeben.
X4	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X4 vorgegeben.
Y1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y1 vorgegeben.
Y2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y2 vorgegeben.
Y3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y3 vorgegeben.
Y4	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y4 vorgegeben.
YMin	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die minimale Begrenzung von *fY* vorgegeben.
YMax	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die maximale Begrenzung von *fY* vorgegeben.
Y	FB_BA_AV_Op [▶ 207]	Ausgabe errechneter Ausgangswert der Kennlinie.

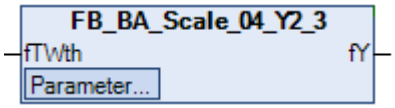
Variablen

Name	Typ	Beschreibung
Scale	FB_BA_Chrc04	Der Baustein stellt eine Linear-Interpolation mit vier Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

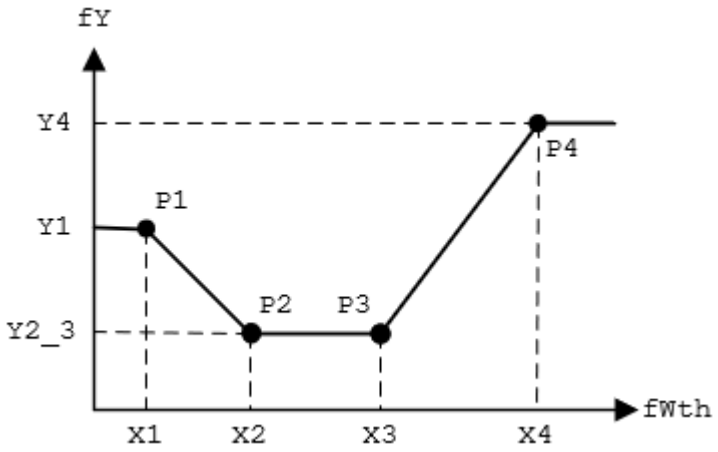
6.1.4.2.2.12.4 FB_BA_Scale_04_Y2_3



Das Template ist ein Sollwertprogramm für eine Abluft/Zuluft Kaskade mit nur einem Raumtemperatursollwert inklusive Sommer- und Winterkompensation.

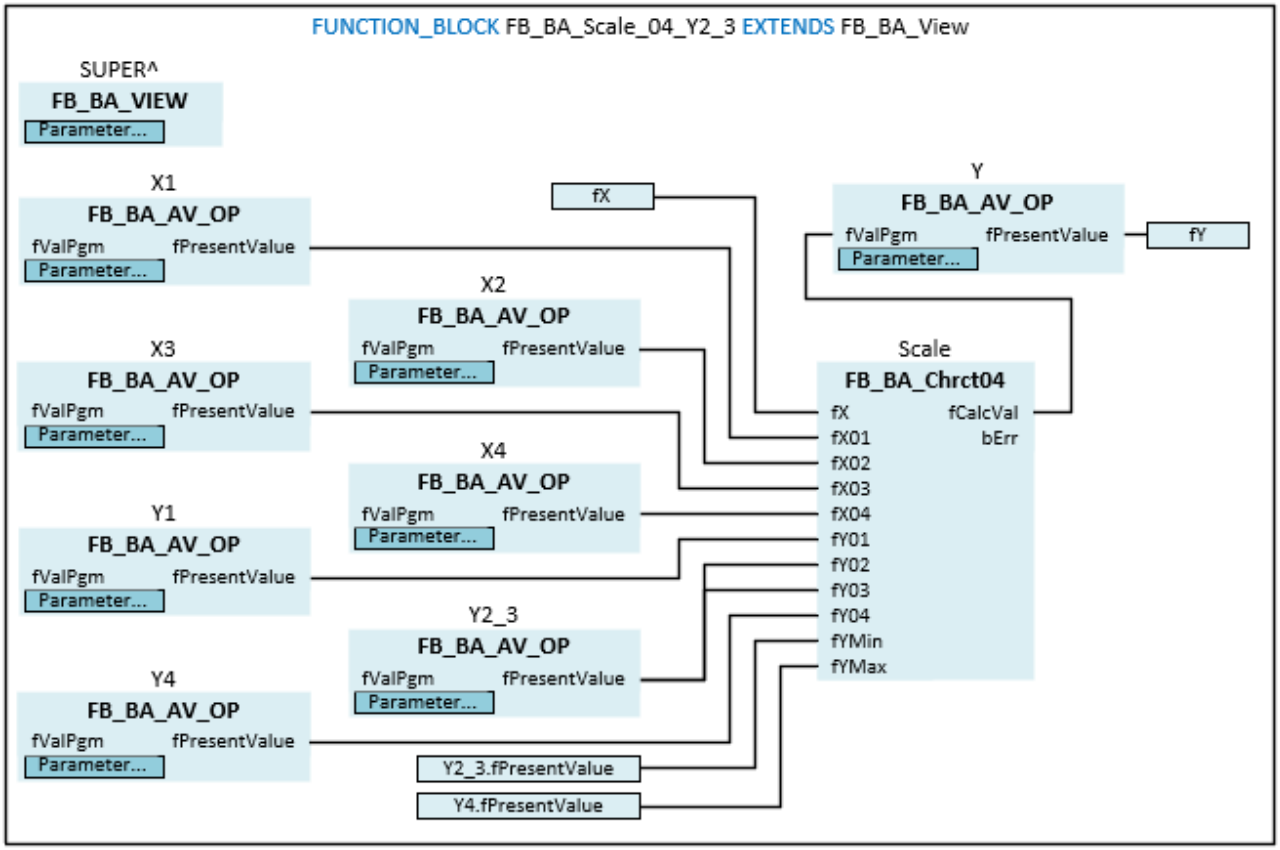
Sollwertprogramm für eine Zulufttemperatur-Regelung mit einem Zulufttemperatur-Sollwert inklusive Sommer- und Winterkompensation über eine Kennlinie.

Die Bestimmung des Zulufttemperatur-Sollwertes wird durch den Funktionsbaustein Scale in Abhängigkeit der Außentemperatur erfolgen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
  fTWth     : REAL;
END_VAR
VAR_OUTPUT
  fY        : REAL;
END_VAR
```

```
VAR_INPUT CONSTANT
  X1       : FB_BA_AV_Op;
  X2       : FB_BA_AV_Op;
  X3       : FB_BA_AV_Op;
  X4       : FB_BA_AV_Op;
  Y1       : FB_BA_AV_Op;
  Y2_3     : FB_BA_AV_Op;
  Y4       : FB_BA_AV_Op;
  Y        : FB_BA_AV_Op;
END_VAR
VAR
  Scale    : FB_BA_Chrct04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fTWth	REAL	Aktueller Wert der Außentemperatur.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Errechneter Sollwert der Raumtemperatur.

Eingänge CONSTANT

Name	Typ	Beschreibung
X1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X1 vorgegeben.
X2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X2 vorgegeben.
X3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X3 vorgegeben.
X4	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X4 vorgegeben.
Y1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y1 vorgegeben.
Y2_3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die Stützpunkte Y2/ Y3 vorgegeben.
Y4	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y4 vorgegeben.
Y	FB_BA_AV_Op [▶ 207]	Ausgabe des errechneten, einfachen Raumtemperatur-Sollwertes. Dieser wird an dem Ausgang *fY* ausgegeben.

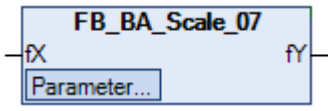
Variablen

Name	Typ	Beschreibung
Scale	FB_BA_Chrct04	Der Baustein stellt eine Linear-Interpolation mit vier Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Der Funktionsbaustein berechnet die Sollwertkennlinie für die aktuelle Raumtemperatur in Abhängigkeit der Außentemperatur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

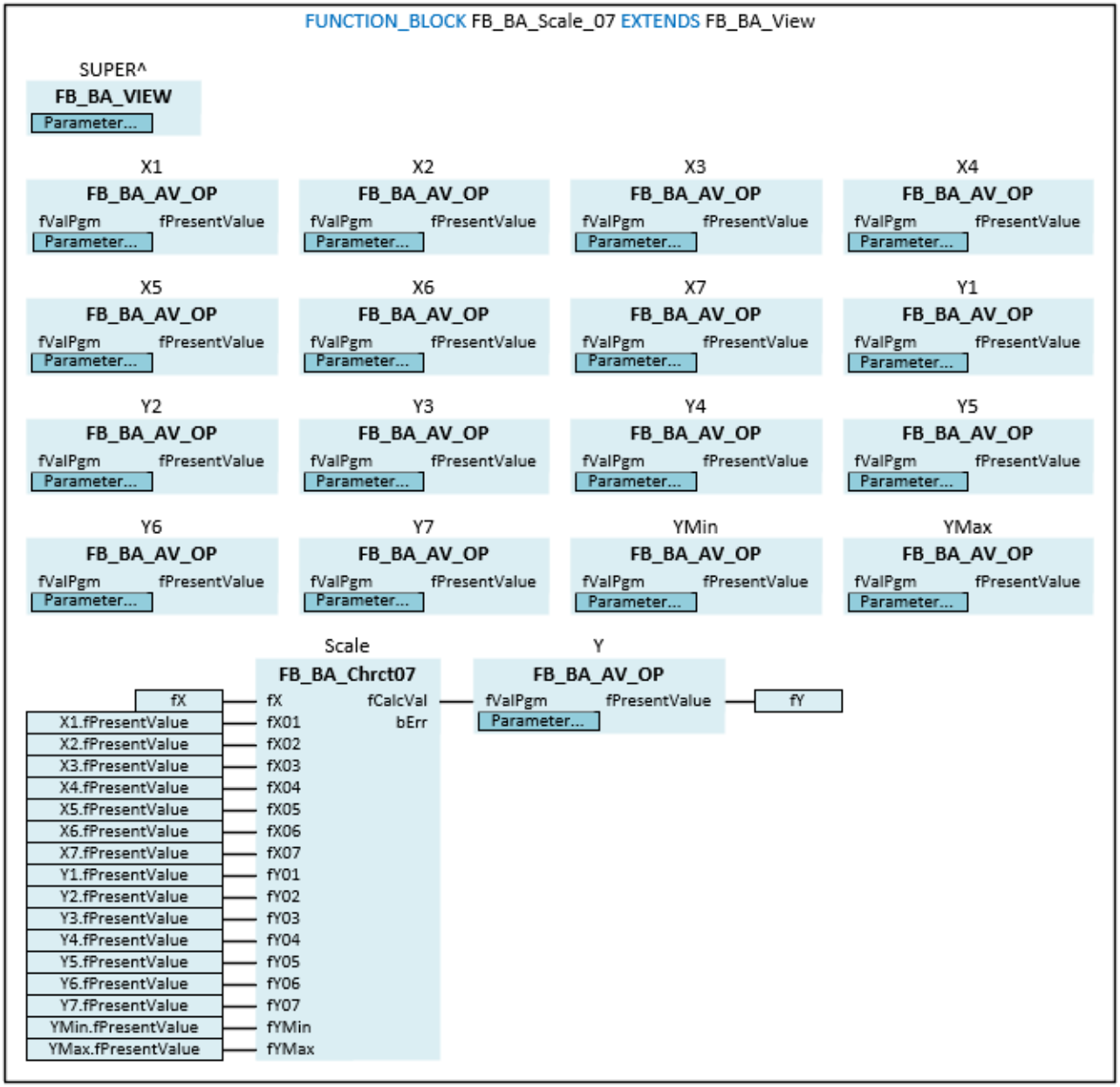
6.1.4.2.2.12.5 FB_BA_Scale_07



Das Template stellt eine Linear-Interpolation mit sieben Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen. Die Kennlinie wird durch die Stützpunkte [*X1/Y1*] bis [*X7/Y7*] bestimmt. Der errechnete Ausgangswert *fY* wird durch *YMin* bzw. *YMax* limitiert.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Scale_04 EXTENDS FB_BA_View
VAR_INPUT
  fX        : REAL;
END_VAR
VAR_OUTPUT
```

```
  fY       : REAL;
END_VAR
VAR_INPUT CONSTANT
  X1       : FB_BA_AV_Op;
  X2       : FB_BA_AV_Op;
  X3       : FB_BA_AV_Op;
  X4       : FB_BA_AV_Op;
  X5       : FB_BA_AV_Op;
  X6       : FB_BA_AV_Op;
  X7       : FB_BA_AV_Op;
  Y1       : FB_BA_AV_Op;
  Y2       : FB_BA_AV_Op;
  Y3       : FB_BA_AV_Op;
  Y4       : FB_BA_AV_Op;
  Y5       : FB_BA_AV_Op;
  Y6       : FB_BA_AV_Op;
  Y7       : FB_BA_AV_Op;
  YMin     : FB_BA_AV_Op;
  YMax     : FB_BA_AV_Op;
  Y        : FB_BA_AV_Op;
END_VAR
VAR
  Scale    : FB_BA_Chrct07;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fX	REAL	Eingangswert der Kennlinie.

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Errechneter Ausgangswert der Kennlinie.

Eingänge CONSTANT

Name	Typ	Beschreibung
X1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X1 vorgegeben.
X2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X2 vorgegeben.
X3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X3 vorgegeben.
X4	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X4 vorgegeben.
X5	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X5 vorgegeben.
X6	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X6 vorgegeben.
X7	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt X7 vorgegeben.
Y1	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y1 vorgegeben.
Y2	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y2 vorgegeben.
Y3	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y3 vorgegeben.
Y4	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y4 vorgegeben.
Y5	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y5 vorgegeben.
Y6	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y6 vorgegeben.
Y7	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für den Stützpunkt Y7 vorgegeben.
YMin	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die minimale Begrenzung von *fY* vorgegeben.
YMax	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt wird der Wert für die maximale Begrenzung von *fY* vorgegeben.
Y	FB_BA_AV_Op [▶ 207]	Ausgabe errechneter Ausgangswert der Kennlinie.

Variablen

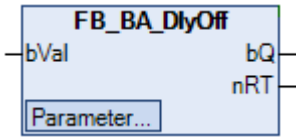
Name	Typ	Beschreibung
Scale	FB_BA_Chrct07	Der Baustein stellt eine Linear-Interpolation mit sieben Stützstellen dar und lässt sich zur Kennlinienerstellung nutzen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.2.13 TimeDelay

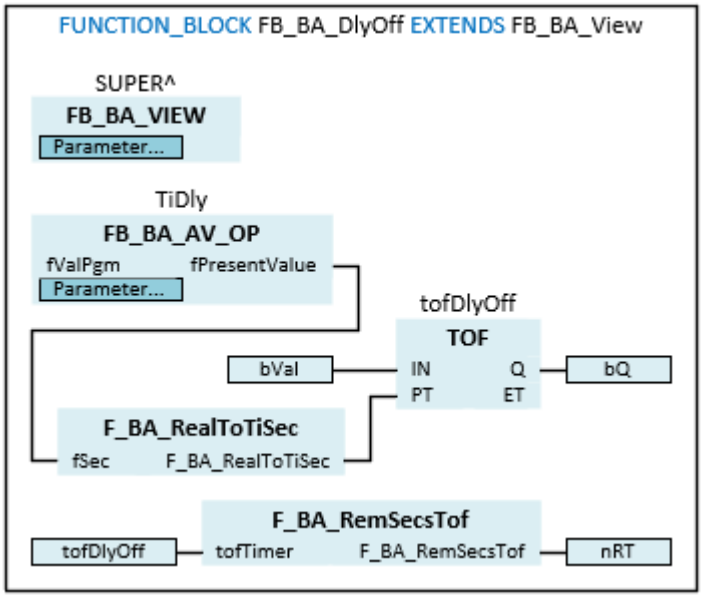
6.1.4.2.2.2.13.1 FB_BA_DlyOff



Das Template realisiert eine Ausschaltverzögerung und kann für die Nachlaufsteuerung von Pumpen eingesetzt werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_DlyOff EXTENDS FB_BA_View
VAR_INPUT
  bVal          : BOOL;
END_VAR
VAR_OUTPUT
  bQ            : BOOL;
  nRT           : UDINT;
END_VAR
VAR_INPUT CONSTANT
  TiDly         : FB_BA_AV_Op;
END_VAR
VAR
  tofDlyOff     : TOF;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bVal	BOOL	Eine fallende Flanke aktiviert die Ausschaltverzögerung.

Ausgänge

Name	Typ	Beschreibung
bQ	BOOL	Zustand der Ausschaltverzögerung.
nRT	UDINT	Restzeit der Ausschaltverzögerung. Nach Ablauf dieser Zeit wird *bQ* auf FALSE gesetzt.

Eingänge CONSTANT

Name	Typ	Beschreibung
TiDly	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes der Ausschaltverzögerung.

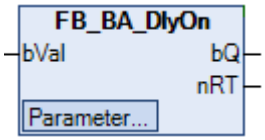
Variablen

Name	Typ	Beschreibung
tofDlyOff	TOF	Der Zeitglied ist das Kernstück dieses Templates zur Ausschaltverzögerung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

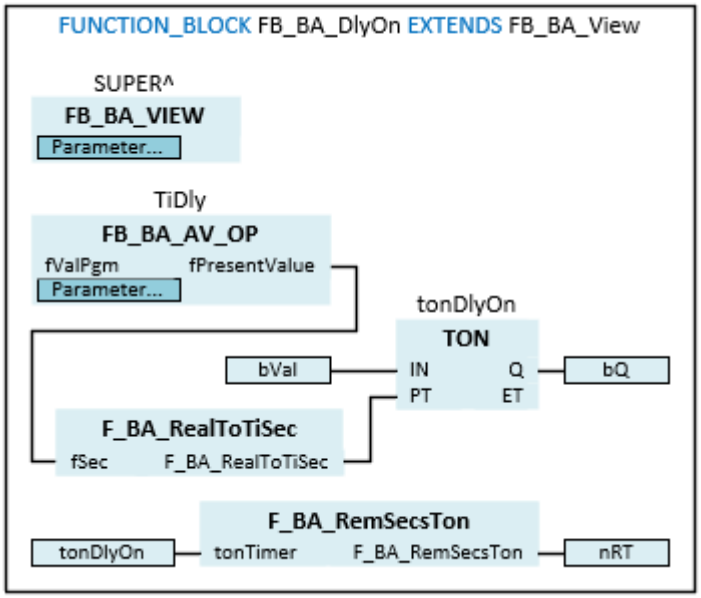
6.1.4.2.2.13.2 FB_BA_DlyOn



Das Template realisiert eine Einschaltverzögerung.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_DlyOff EXTENDS FB_BA_View
VAR_INPUT
  bVal        : BOOL;
END_VAR
VAR_OUTPUT
  bQ          : BOOL;
  nRT         : UDINT;
END_VAR
VAR_INPUT CONSTANT
  TiDly       : FB_BA_AV_Op;
END_VAR
VAR
  tonDlyOn    : TON;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bVal	BOOL	Eine steigende Flanke aktiviert die Einschaltverzögerung.

Ausgänge

Name	Typ	Beschreibung
bQ	BOOL	Zustand der Einschaltverzögerung.
nRT	UDINT	Restzeit der Einschaltverzögerung. Nach Ablauf dieser Zeit wird *bQ* auf TRUE gesetzt.

Eingänge CONSTANT

Name	Typ	Beschreibung
TiDly	FB_BA_AV_Op [▶ 207]	AV-Objekt zur Eingabe des Wertes der Einschaltverzögerung.

Variablen

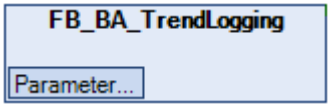
Name	Typ	Beschreibung
tofDlyOn	TON	Der Zeitglied ist das Kernstück dieses Templates zur Einschaltverzögerung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.14 Trend

6.1.4.2.2.2.14.1 FB_BA_TrendLogging



Das Template erfasst die Werte einer Datenquelle und zeichnet sie im Trendlog-Speicher auf. Die Verbindung zu den Trendlog-Objekten (Datenquellen, z. B. FB_BA_BO_IO [▶ 214], FB_BA_AO_IO [▶ 202]) geschieht über Referenzen.

Die Variable *TrendCount* definiert die Größe des Feldes der Trendlog-Objekte.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_TrendLogging EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  Trends          : ARRAY[1..TREND_COUNT] OF FB_BA_Trend;
END_VAR
VAR
  i               : UDINT;
END_VAR
VAR CONSTANT
  TREND_COUNT     : UDINT := 30;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
Trends	FB_BA_Trend [▶ 231]	Feld mit 30 Einträgen zu Trendlog-Objekten.

Variablen

Name	Typ	Beschreibung
i	UDINT	Zähler für die FOR-Schleife, um das Trendlog-Feld Trends aufzurufen.

Variablen

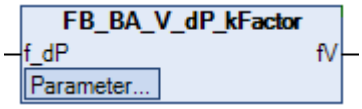
Name	Typ	Beschreibung
TrendCount	UDINT	Konstante zur Definition der Menge der zu erfassenden Trendlog-Objekte. Voreingestellt auf 30.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.2.15 VolumeFlowDetermination

6.1.4.2.2.2.15.1 FB_BA_V_dP_kFactor



Das Template wird eingesetzt zur Volumenstrombestimmung mittels Differenzdruck und k-Faktor.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_V_dP_kFactor
VAR_INPUT
  {attribute 'parameterUnit':='Pa'}
  f_dP                    : REAL;
END_VAR
VAR_OUTPUT
  {attribute 'parameterUnit':='m³/h'}
```

```
    fV                  : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory':='Unit'}
  eConversion_kFactor    : E_BA_Conversion_kFactor := E_BA_Conversion_kFactor.e_Pa;
  {attribute 'parameterCategory':='Config'}
  fK                  : REAL := 381;
END_VAR
VAR
    fConversionF        : REAL;
END_VAR
VAR CONSTANT
    f_Pa                : REAL := 1;
    f_hPa               : REAL := 0.1;
    f_mbar              : REAL := 0.1;
    f_mm_HG             : REAL := 0.086613;
    f_in_HG             : REAL := 0.017185;
    f_mm_WS             : REAL := 0.31933;
    f_psi               : REAL := 0.012043;
    f_inches_H2O        : REAL := 0.063361;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
f_dP	REAL	Erfassung des Differenzdrucks [Pa].

Ausgänge

Name	Typ	Beschreibung
fV	REAL	Ausgabe des errechneten Volumenstroms [m³/h].

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
eConversion_kFactor	E_BA_Conversion_kFactor [▶ 712]	Umrechnungsfaktor für die Einheit des k-Faktors.
fK	REAL	k-Faktor des Herstellers der Ventilatordüse (Venturi Prinzip).

Variablen

Name	Typ	Beschreibung
fConversionF	REAL	Zeigt an, welcher Umrechnungsfaktor durch die Aufzählungsvariable *eConversion_kFactor* ausgewählt wurde und wird zur Berechnung des Volumenstroms verwendet.

Variablen

Name	Typ	Beschreibung
f_Pa	REAL	Umrechnungsfaktor für den Druck [Pa].
f_hPa	REAL	Umrechnungsfaktor für den Druck [hPa].
f_mbar	REAL	Umrechnungsfaktor für den Druck [mbar].
f_mm_HG	REAL	Umrechnungsfaktor für die Einheit Millimeter Quecksilbersäule [mmHG, Torr].
f_in_HG	REAL	Umrechnungsfaktor für die Einheit Zoll Quecksilber [inHG].
f_mm_WS	REAL	Umrechnungsfaktor für die Einheit Millimeter Wassersäule [mmWS, mmH2O].
f_psi	REAL	Umrechnungsfaktor für den Druck [psi].
f_inchesn_H2O	REAL	Umrechnungsfaktor für die Einheit Zoll Wassersäule [inH2O].

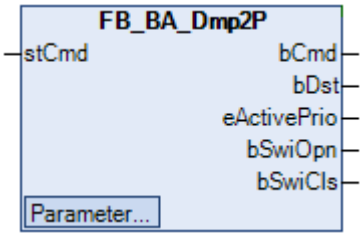
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.3 Damper

6.1.4.2.2.3.1 2P

6.1.4.2.2.3.1.1 FB_BA_Dmp2P

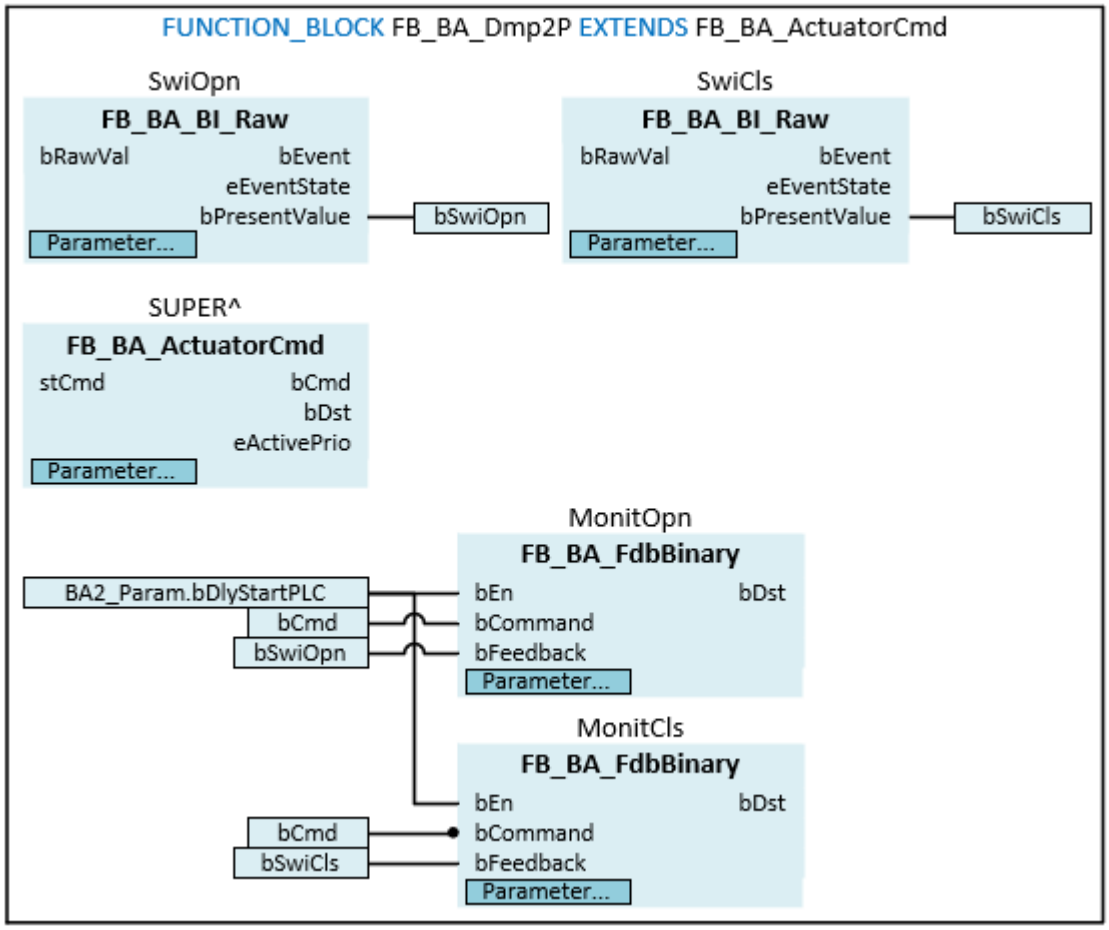


Das Template ist für die Ansteuerung einer Zweipunktklappe mit integrierter Überwachung beider Endlagen.

Es besteht im Wesentlichen aus der Basisklasse FB_BA_ActuatorCmd [▶ 1004] für die Ansteuerung eines binären Aggregates und dem Sammeln aller sicherheitsrelevanten Störungen. Die beiden Templates *MonitOpn* und *MonitCls* dienen der Überwachung der Endlagen.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Dmp2P EXTENDS FB_BA_ActuatorCmd
VAR_OUTPUT
  bSwiOpn       : BOOL;
  bSwiCls       : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SwiOpn        : FB_BA_BI_Raw;
  SwiCls        : FB_BA_BI_Raw;
  MonitOpen     : FB_BA_FdbBinary;
  MonitClose    : FB_BA_FdbBinary;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bSwiOpn	BOOL	Endlage „Auf“ ist erreicht.
bSwiCls	BOOL	Endlage „Zu“ ist erreicht.

Eingänge CONSTANT

Name	Typ	Beschreibung
SwiOpn	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung des Endlagenschalters „Auf“.
SwiCls	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung des Endlagenschalters „Zu“.
MonitOpen	FB_BA_FdbBinary [▶ 1036]	Das Template überwacht die Endlage „Auf“.
MonitClose	FB_BA_FdbBinary [▶ 1036]	Das Template überwacht die Endlage „Zu“.

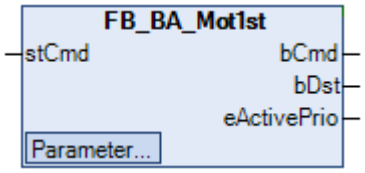
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.4 Motor

6.1.4.2.2.4.1 1st

6.1.4.2.2.4.1.1 FB_BA_Mot1st

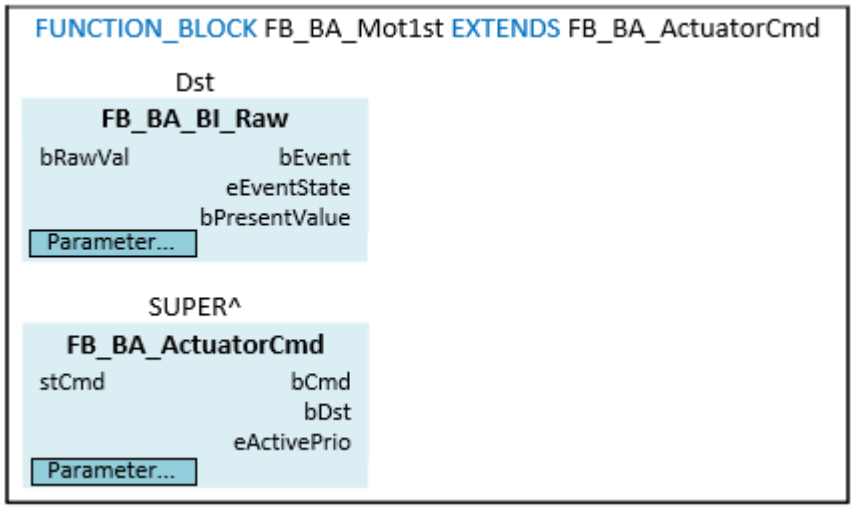


Das Template ist für die Ansteuerung eines einstufigen Motors mit Störmeldung, z. B. eines Ventilators.

Es besteht im Wesentlichen aus der Basisklasse FB_BA_ActuatorCmd [▶ 1004] für die Ansteuerung eines binären Aggregates und dem Sammeln aller sicherheitsrelevanten Störungen. Das BI-Objekt *Dst* zeigt eine Störung des Motors an.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Mot1st EXTENDS FB_BA_ActuatorCmd
VAR_INPUT CONSTANT
  Dst    : FB_BA_BI_Raw;
END_VAR
VAR
END_VAR
```

Eingänge CONSTANT

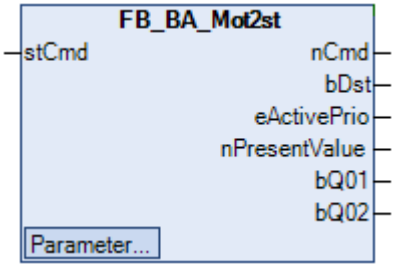
Name	Typ	Beschreibung
Dst	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung einer Störung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.4.2 2st

6.1.4.2.2.4.2.1 FB_BA_Mot2st



Das Template dient der Ansteuerung eines zweistufigen Motors mit Störmeldungen, z. B. eines Ventilators.

Es besteht im Wesentlichen aus dem Funktionsbaustein *StpCtrl* und der Basisklasse FB_BA_ActuatorMO [▶ 1009] für die Ansteuerung eines mehrstufigen Aggregates und dem Sammeln aller sicherheitsrelevanten Störungen.

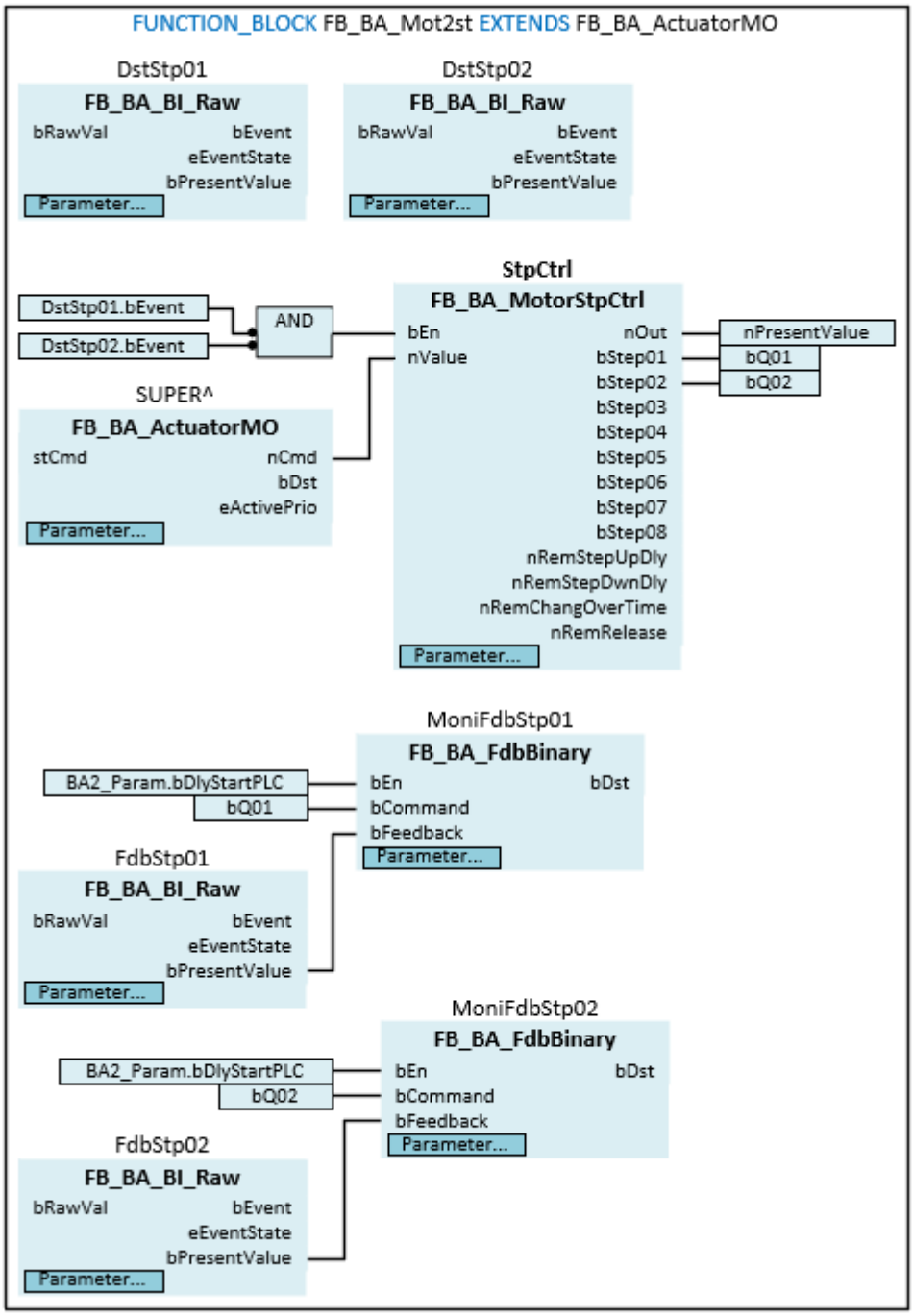
Die BI-Objekte *DstStp01* und *DstStp02* zeigen Störungen der jeweiligen Stufe an.

Die beiden Templates *MoniFdbStp01* und *MoniFdbStp02* dienen der Rückmeldeüberwachung der Stufen 1 + 2, *FdbStp01* und *FdbStp02*.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Mot2st EXTENDS FB_BA_ActuatorMO
VAR_OUTPUT
  nPresentValue     : UDINT;
  bQ01              : BOOL;
  bQ02              : BOOL;
END_VAR
VAR_INPUT CONSTANT
  DstStp01          : FB_BA_BI_Raw;
  DstStp02          : FB_BA_BI_Raw;
  FdbStp01          : FB_BA_BI_Raw;
  FdbStp02          : FB_BA_BI_Raw;
  MoniFdbStp01      : FB_BA_FdbBinary;
  MoniFdbStp02      : FB_BA_FdbBinary;
  StpCtrl           : FB_BA_MotorStpCtrl;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
nPresentValue	UDINT	Aktuelle Schaltstufe des Motors.
bQ01	BOOL	Variable für die Ansteuerung der Stufe 1 des Motors. Diese Variable muss mit einer Busklemme verknüpft werden.
bQ02	BOOL	Variable für die Ansteuerung der Stufe 2 des Motors. Diese Variable muss mit einer Busklemme verknüpft werden.

Eingänge CONSTANT

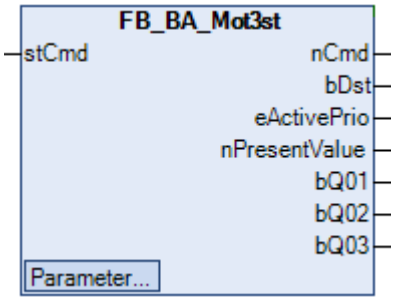
Name	Typ	Beschreibung
DstStp01	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Störung der Stufe 1 des Motors.
DstStp02	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Störung der Stufe 2 des Motors.
FdbStp01	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Rückmeldung der Stufe 1 des Motors.
FdbStp02	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Rückmeldung der Stufe 2 des Motors.
MoniFdbStp01	FB_BA_FdbBinary [▶ 1036]	Template, welches die Rückmeldung der Stufe 1 des Motors überwacht.
MoniFdbStp02	FB_BA_FdbBinary [▶ 1036]	Template, welches die Rückmeldung der Stufe 2 des Motors überwacht.
StpCtrl	FB_BA_MotorStpCtrl [▶ 423]	Der Funktionsbaustein *StpCtrl* erhält den nummerischen Schaltwert von der Basisklasse FB_BA_ActuatorMO [▶ 1009] und wandelt den Schaltwert in einzelne Schaltstufen um.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.4.3 3st

6.1.4.2.2.4.3.1 FB_BA_Mot3st



Das Template dient der Ansteuerung eines dreistufigen Motors mit Störmeldungen, z. B. eines Ventilators.

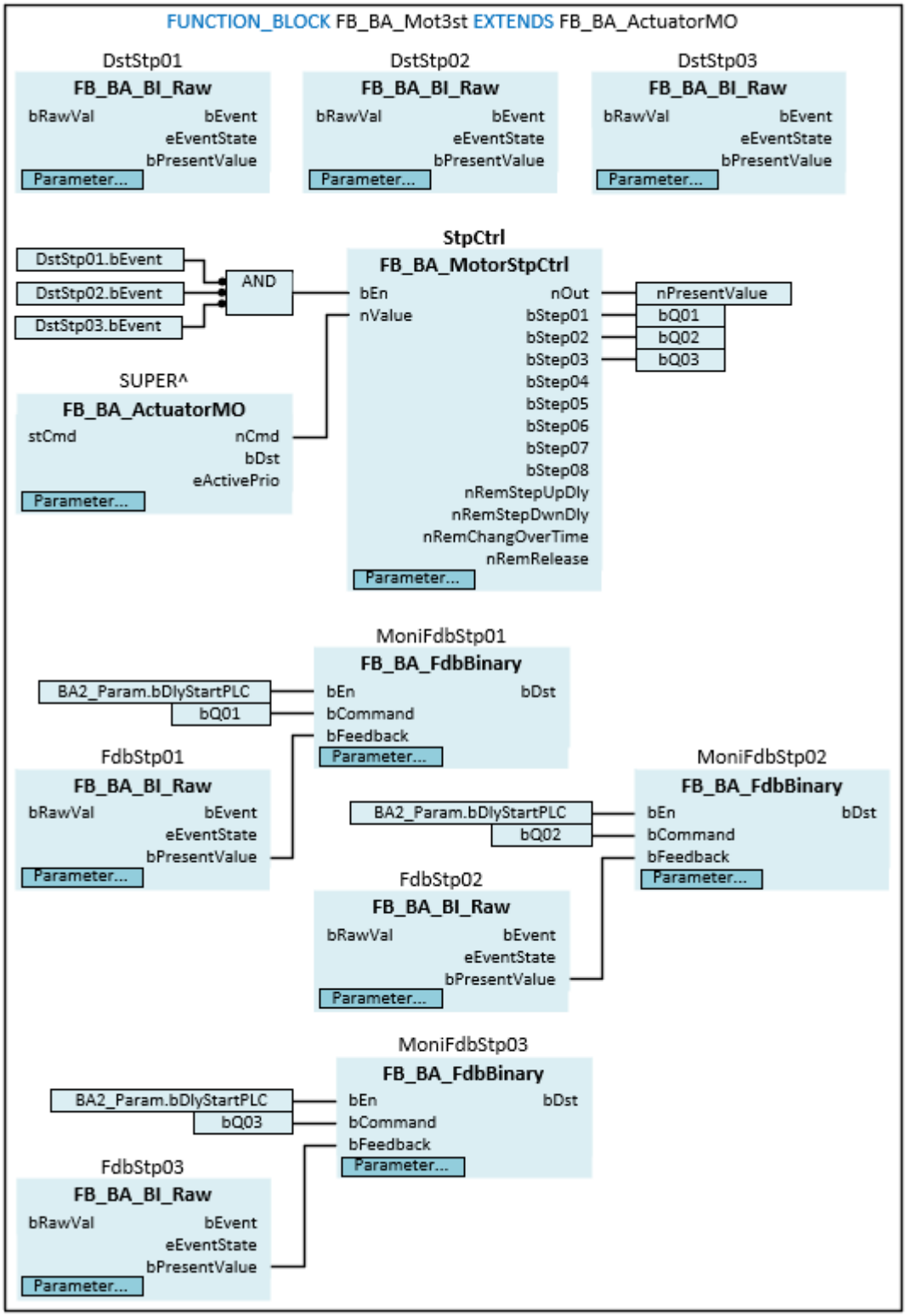
Es besteht im Wesentlichen aus dem Funktionsbaustein *StpCtrl* und der Basisklasse FB_BA_ActuatorMO [▶ 1009] für die Ansteuerung eines mehrstufigen Aggregates und dem Sammeln aller sicherheitsrelevanten Störungen.

Die BI-Objekte *DstStp01*, *DstStp02* und *DstStp03* zeigen Störungen der jeweiligen Stufe an.

Die beiden Templates *MoniFdbStp01*, *MoniFdbStp02* und *MoniFdbStp03* dienen der Rückmeldeüberwachung der Stufen 1 + 2 + 3, *FdbStp01*, *FdbStp02* und *FdbStp03*.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Mot3st EXTENDS FB_BA_ActuatorMO
VAR_OUTPUT
  nPresentValue     : UDINT;
  bQ01              : BOOL;
  bQ02              : BOOL;
  bQ03              : BOOL;
END_VAR
VAR_INPUT CONSTANT
  DstStp01          : FB_BA_BI_Raw;
  DstStp02          : FB_BA_BI_Raw;
  DstStp03          : FB_BA_BI_Raw;
  FdbStp01          : FB_BA_BI_Raw;
  FdbStp02          : FB_BA_BI_Raw;
  FdbStp03          : FB_BA_BI_Raw;
  MoniFdbStp01      : FB_BA_FdbBinary;
  MoniFdbStp02      : FB_BA_FdbBinary;
  MoniFdbStp03      : FB_BA_FdbBinary;
  StpCtrl           : FB_BA_MotorStpCtrl;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
nPresentValue	UDINT	Aktuelle Schaltstufe des Motors.
bQ01	BOOL	Variable für die Ansteuerung der Stufe 1 des Motors. Diese Variable muss mit einer Busklemme verknüpft werden.
bQ02	BOOL	Variable für die Ansteuerung der Stufe 2 des Motors. Diese Variable muss mit einer Busklemme verknüpft werden.

Eingänge CONSTANT

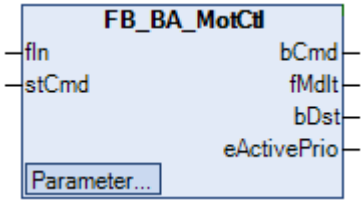
Name	Typ	Beschreibung
DstStp01	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Störung der Stufe 1 des Motors.
DstStp02	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Störung der Stufe 2 des Motors.
DstStp03	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Störung der Stufe 3 des Motors.
FdbStp01	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Rückmeldung der Stufe 1 des Motors.
FdbStp02	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Rückmeldung der Stufe 2 des Motors.
FdbStp03	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung der Rückmeldung der Stufe 3 des Motors.
MoniFdbStp01	FB_BA_FdbBinary [▶ 1036]	Template, welches die Rückmeldung der Stufe 1 des Motors überwacht.
MoniFdbStp02	FB_BA_FdbBinary [▶ 1036]	Template, welches die Rückmeldung der Stufe 2 des Motors überwacht.
MoniFdbStp03	FB_BA_FdbBinary [▶ 1036]	Template, welches die Rückmeldung der Stufe 3 des Motors überwacht.
StpCtrl	FB_BA_MotorStpCtrl [▶ 423]	Der Funktionsbaustein *StpCtrl* erhält den nummerischen Schaltwert von der Basisklasse FB_BA_ActuatorMO [▶ 1009] und wandelt den Schaltwert in einzelne Schaltstufen um.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.4.4 Control

6.1.4.2.2.4.4.1 FB_BA_MotCtl



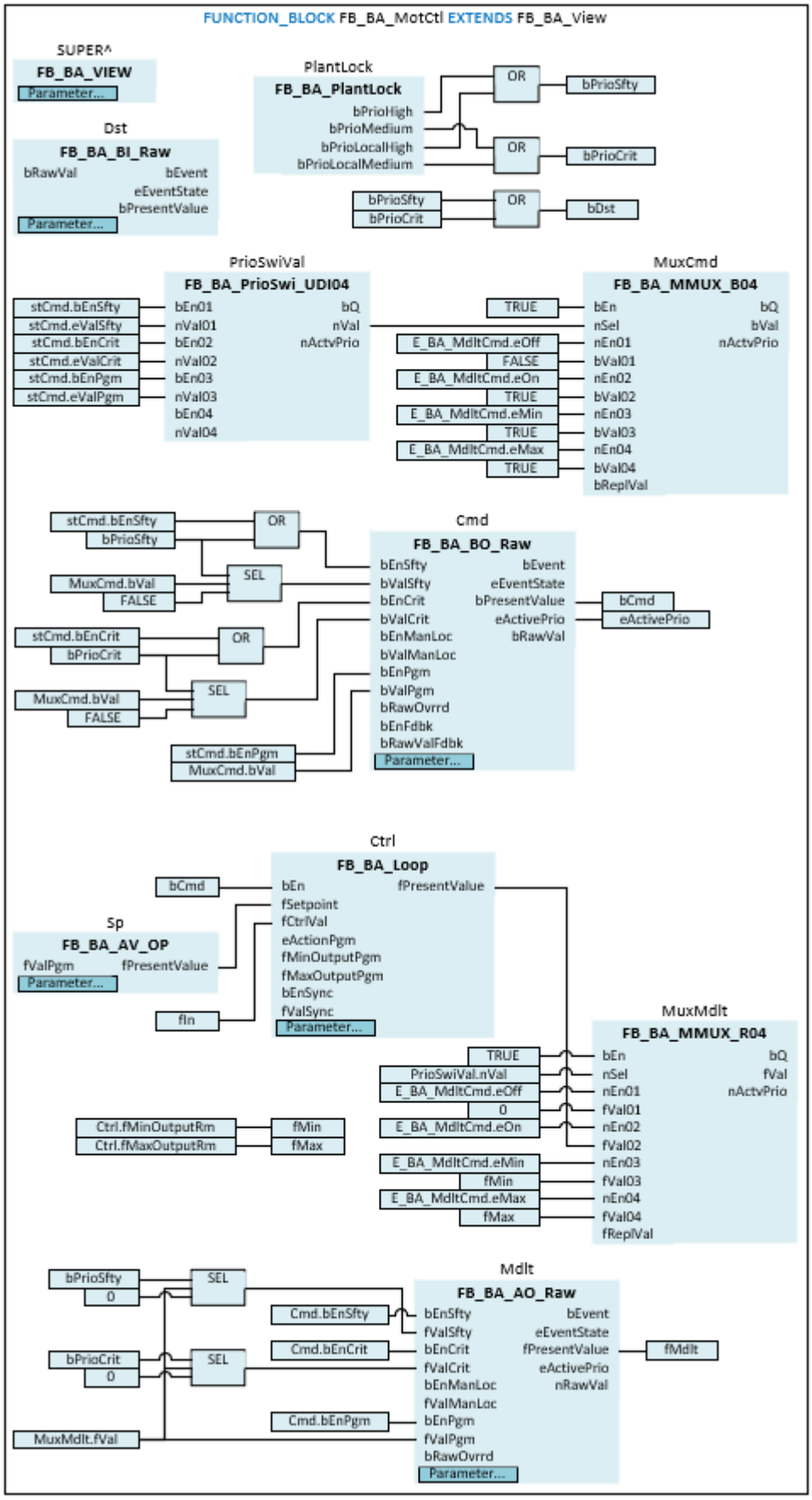
Das Template dient der Ansteuerung und Regelung eines Drehzahl-gesteuerten Motors.

Es besteht im Wesentlichen aus einem BO- und AO-Objekt für die Ansteuerung des Motors (Frequenzumformers), eines BI-Objektes zur Anzeige einer Störung und einem PID-Regler für die Drehzahlregelung. Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen. Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Modulationsbefehle übermittelt.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtl EXTENDS FB_BA_View
VAR_INPUT
  fIn             : REAL;
  stCmd           : ST_BA_Mdlt;
END_VAR
VAR_OUTPUT
  bCmd            : BOOL;
  fMdlt           : REAL;
  bDst            : BOOL;
  eActivePrio     : E_BA_Priority;
END_VAR
VAR_INPUT CONSTANT
  Dst             : FB_BA_BI_Raw;
  Cmd             : FB_BA_BO_Raw;
  Mdlt            : FB_BA_AO_Raw;
  Sp              : FB_BA_AV_Op;
  Ctrl            : FB_BA_Loop;
  PlantLock       : FB_BA_PlantLock;
END_VAR
VAR
  bPrioSfty       : BOOL;
  bPrioCrit       : BOOL;
  PrioSwiVal      : FB_BA_PrioSwi_UDI04;
  MuxCmd          : FB_BA_MMUX_B04;
  MuxMdlt         : FB_BA_MMUX_R04;
  fMax            : REAL;
  fMin            : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fIn	REAL	An dem Eingang wird der Istwert für den Regler *Ctrl* angeschlossen.
stCmd	ST_BA_Mdlt [▶ 281]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Modulationsbefehle übermittelt. Das Kommando mit der höchsten Priorität bestimmt am Binär-Ausgangs-Objekt *Cmd* den Schaltbefehl and am Analog-Ausgabe-Objekt *Mdlt* den Stellbefehl.

Ausgänge

Name	Typ	Beschreibung
bCmd	BOOL	Ausgabe des Schaltwertes.
fMdlt	REAL	Aktueller Wert des Analog-Output-Objektes.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

Eingänge CONSTANT

Name	Typ	Beschreibung
Dst	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung einer Störung.
Cmd	FB_BA_BO_Raw [▶ 216]	Das Binär-Ausgangs-Objekt dient der Ausgabe eines Schaltbefehls und übermittelt diesen an die IO-Ebene.
Mdlt	FB_BA_AO_Raw [▶ 204]	Aktueller Wert des Analog-Output-Objektes.
Sp	FB_BA_AV_Op [▶ 207]	Eingabe des Sollwerts für den PID-Regler *Ctrl*.
Ctrl	FB_BA_Loop [▶ 225]	PID-Regler.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

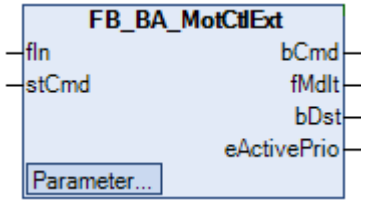
Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter PrioSwiVal [▶ 443] ermittelt anhand der Kommandostruktur *stCmd* den Modulationsbefehl für die Multiplexer *MuxCmd* und *MuxMdlt*.
MuxCmd	FB_BA_MMUX_B04 [▶ 438]	Der Multiplexer MuxCmd [▶ 438] ermittelt den aktuellen Schaltwert aus der Kommandierung des Prioritätenschalters *PrioSwiVal*. Das resultierende Ergebnis wird an das Binär-Ausgangs-Objekt *Cmd* übermittelt.
MuxMdlt	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer MuxMdlt [▶ 438] ermittelt den aktuellen Stellwert aus den Modulationswerten *Ctrl.fPresentValue*, *fMin, fMax* und dem Modulationsbefehl des Prioritätenschalters *PrioSwiVal*. Das resultierende Ergebnis wird an das Analog-Ausgangs-Objekt *Mdlt* übermittelt.
fMax	REAL	Maximalwert des Reglers, welcher ausgegeben wird, wenn der Modulationsbefehl *E_BA_MdltCmd.eMax* ansteht (siehe E_BA_Mdlt [▶ 275]).
fMin	REAL	Minimalwert des Reglers, welcher ausgegeben wird, wenn der Modulationsbefehl *E_BA_MdltCmd.eMin* ansteht (siehe E_BA_Mdlt [▶ 275]).

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.4.4.2 FB_BA_MotCtlExt

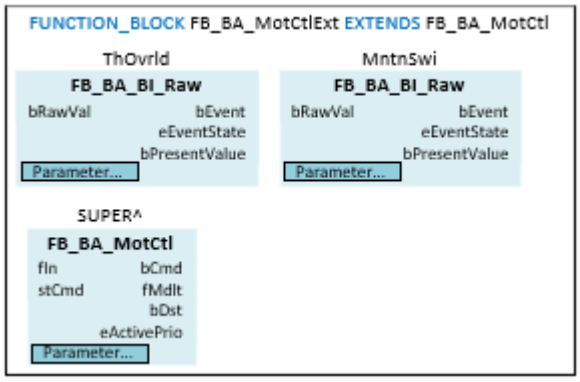


Das Template dient der Ansteuerung und Regelung eines Drehzahl-gesteuerten Motors. Es besteht im Wesentlichen aus der Basisklasse FB_BA_MotCtl [▶ 1085].

Der Unterschied zum Template FB_BA_MotCtl [▶ 1085] sind die zusätzlichen BI-Objekte *ThOvrld* und *MntnSwi*.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtlExt EXTENDS FB_BA_MotCtl
VAR_INPUT CONSTANT
  ThOvrld    : FB_BA_BI_Raw;
  MntnSwi    : FB_BA_BI_Raw;
END_VAR
```

Eingänge CONSTANT

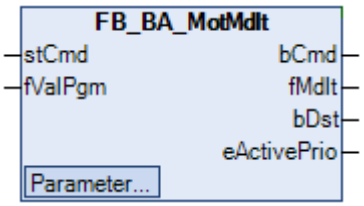
Name	Typ	Beschreibung
ThOvrld	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung der Störung "Thermische Überlastung".
MntnSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Wartungsschalters.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.4.5 Modulation

6.1.4.2.2.4.5.1 FB_BA_MotMdlt



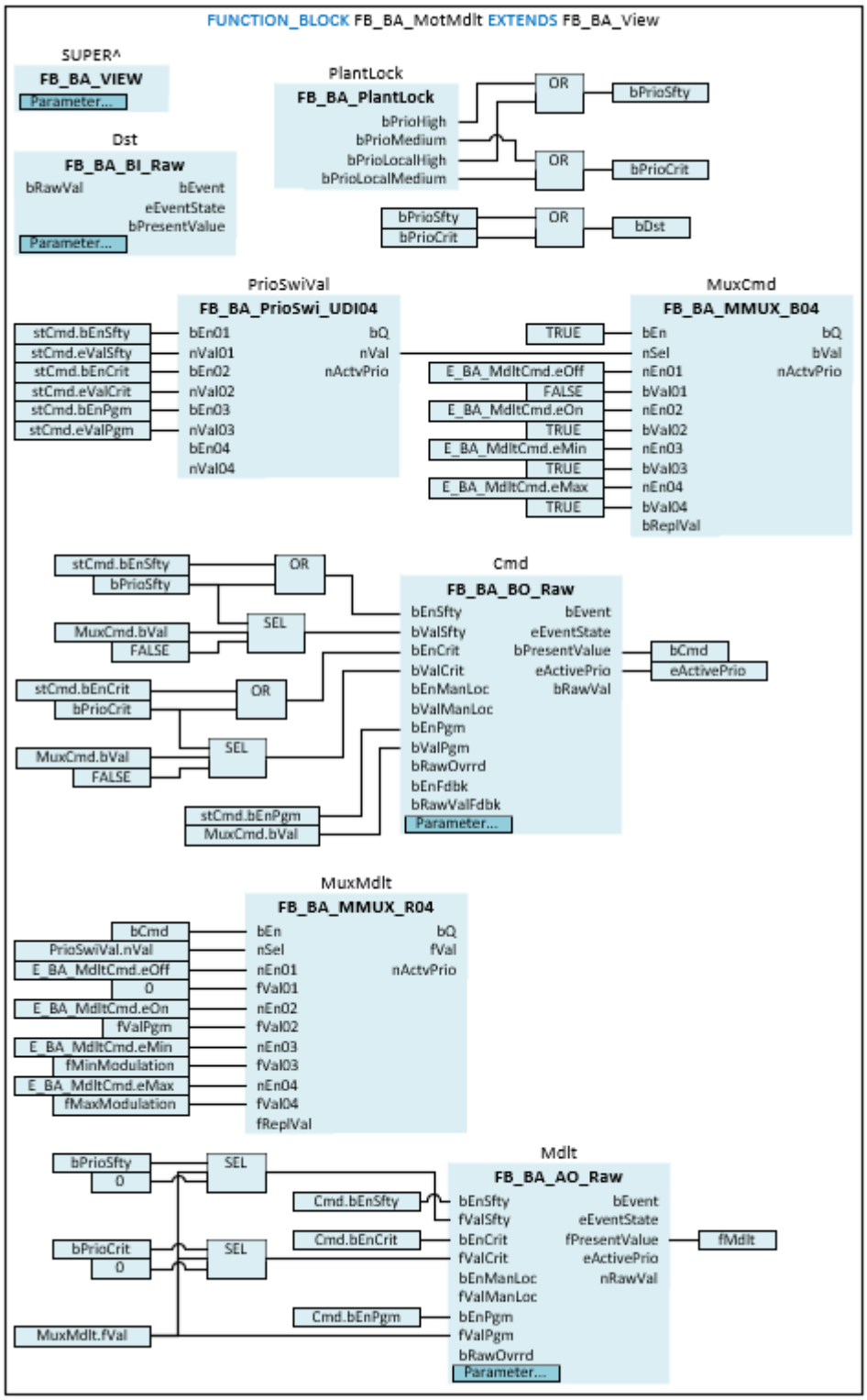
Das Template dient der Ansteuerung eines Drehzahl-geregelten Motors.

Es besteht im Wesentlichen aus einem BO- und AO-Objekt für die Ansteuerung des Motors (Frequenzumformer) und einem BI-Objekt zur Anzeige einer Störung. Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen. Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Modulationsbefehle übermittelt



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtl EXTENDS FB_BA_View
VAR_INPUT
  stCmd             : ST_BA_Mdlt;
  fValPgm           : REAL;
END_VAR
VAR_INPUT CONSTANT
  Dst               : FB_BA_BI_Raw;
  Cmd               : FB_BA_BO_Raw;
  Mdlt              : FB_BA_AO_Raw;
  PlantLock         : FB_BA_PlantLock;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  {attribute 'parameterCategory' := 'Behaviour'}
  fMinModulation    : REAL := 20;
  {attribute 'parameterCategory' := 'Behaviour'}
  fMaxModulation    : REAL := 100;
END_VAR
VAR_OUTPUT
  bCmd              : BOOL;
  fMdlt             : REAL;
  bDst              : BOOL;
  eActivePrio       : E_BA_Priority;
END_VAR
VAR
  bPrioSfty         : BOOL;
  bPrioCrit         : BOOL;
  PrioSwiVal        : FB_BA_PrioSwi_UDI04;
  MuxCmd            : FB_BA_MMUX_B04;
  MuxMdlt           : FB_BA_MMUX_R04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stCmd	ST_BA_Mdlt [▶ 281]	Über die Kommandostruktur *stCmd* werden dem Template die Freigaben und Modulationsbefehle übermittelt. Das Kommando mit der höchsten Priorität bestimmt am Binär-Ausgangs-Objekt *Cmd* den Schaltbefehl and am Analog-Ausgabe-Objekt *Mdlt* den Stellbefehl.
fValPgm	REAL	Stellwert für die Ansteuerung des Motors.

Eingänge CONSTANT

Name	Typ	Beschreibung
Dst	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung einer Störung.
Cmd	FB_BA_MO [▶ 237]	Das Multistate-Ausgabe-Objekt dient der Ausgabe des aktuellen Schaltwertes.
Mdlt	FB_BA_AO_Raw [▶ 204]	Aktueller Wert des Analog-Output-Objektes.
PlantLock	FB_BA_PlantLock [▶ 138]	Der Funktionsbaustein *PlantLock* sammelt alle sicherheitsrelevanten Störungen auf dieser Ebene der Projektstruktur und löst dementsprechend Schaltbefehle im Template aus.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
fMinModulation	REAL	Konstanter minimaler Wert, welcher ausgegeben wird, wenn der Modulationsbefehl *E_BA_MdltCmd.eMin* ansteht (siehe E_BA_Mdlt [▶ 275]).
fMaxModulation	REAL	Konstanter maximaler Wert, welcher ausgegeben wird, wenn der Modulationsbefehl *E_BA_MdltCmd.eMax* ansteht (siehe E_BA_Mdlt [▶ 275]).

Ausgänge

Name	Typ	Beschreibung
bCmd	BOOL	Ausgabe des Schaltwertes.
fMdlt	REAL	Aktueller Wert des Analog-Output-Objektes.
bDst	BOOL	Die Variable ist eine Auswertung der Lockprioritäten „Safety“ und „Critical“ der Projektstruktur und zeigt ein ausgelöstes Ereignis an.
eActivePrio	E_BA_Priority [▶ 112]	Anzeige der aktiven Priorität.

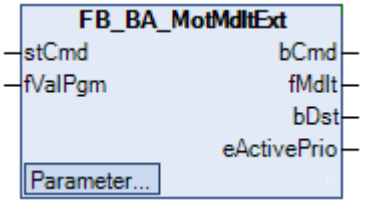
Variablen

Name	Typ	Beschreibung
bPrioSfty	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Safety“ der Projektstruktur.
bPrioCrit	BOOL	Die Variable ist eine Auswertung der Lockpriorität „Critical“ der Projektstruktur.
bPrioSwiVal	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter PrioSwiVal [▶ 443] ermittelt anhand der Kommandostruktur *stCmd* den Modulationsbefehl für die Multiplexer *MuxCmd* und *MuxMdlt*.
MuxCmd	FB_BA_MMUX_B04 [▶ 438]	Der Multiplexer MuxCmd [▶ 438] ermittelt den aktuellen Schaltwert aus der Kommandierung des Prioritätenschalters *PrioSwiVal*. Das resultierende Ergebnis wird an das Binär-Ausgangs-Objekt *Cmd* übermittelt.
MuxMdlt	FB_BA_MMUX_R04 [▶ 438]	Der Multiplexer MuxMdlt [▶ 438] ermittelt den aktuellen Stellwert aus den Modulationswerten *Ctrl.fPresentValue*, *fMin, fMax* und dem Modulationsbefehl des Prioritätenschalters *PrioSwiVal*. Das resultierende Ergebnis wird an das Analog-Ausgangs-Objekt *Mdlt* übermittelt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.4.5.2 FB_BA_MotMdltExt

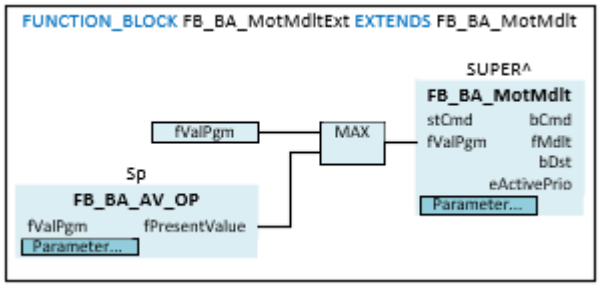


Das Template dient der Ansteuerung eines Drehzahl-gesteuerten Motors. Es besteht im Wesentlichen aus der Basisklasse FB_BA_MotMdlt [▶ 1089].

Der Unterschied zum Template FB_BA_MotMdlt [▶ 1089] ist das AV-Objekt *Sp* mit der dazu gehörigen MAX-Auswahl. Über das *Sp* Objekt könnte so ein Mindest-Stellwert für den Motor vorgegeben werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_MotCtlExt EXTENDS FB_BA_MotMdlt
VAR_INPUT CONSTANT
  Sp     : FB_BA_AV_Op;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
Sp	FB_BA_AV_Op [▶ 207]	Über das AV-Objekt kann der Mindest-Stellwert für den Motor vorgegeben werden. Eine MAX-Auswahl verwendet dann entweder diesen Sollwert oder den Wert vom Eingang *fValPgm* der Basisklasse FB_BA_MotMdlt [▶ 1089].

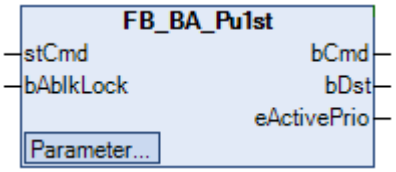
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.5 Pump

6.1.4.2.2.5.1 1st

6.1.4.2.2.5.1.1 FB_BA_Pu1st

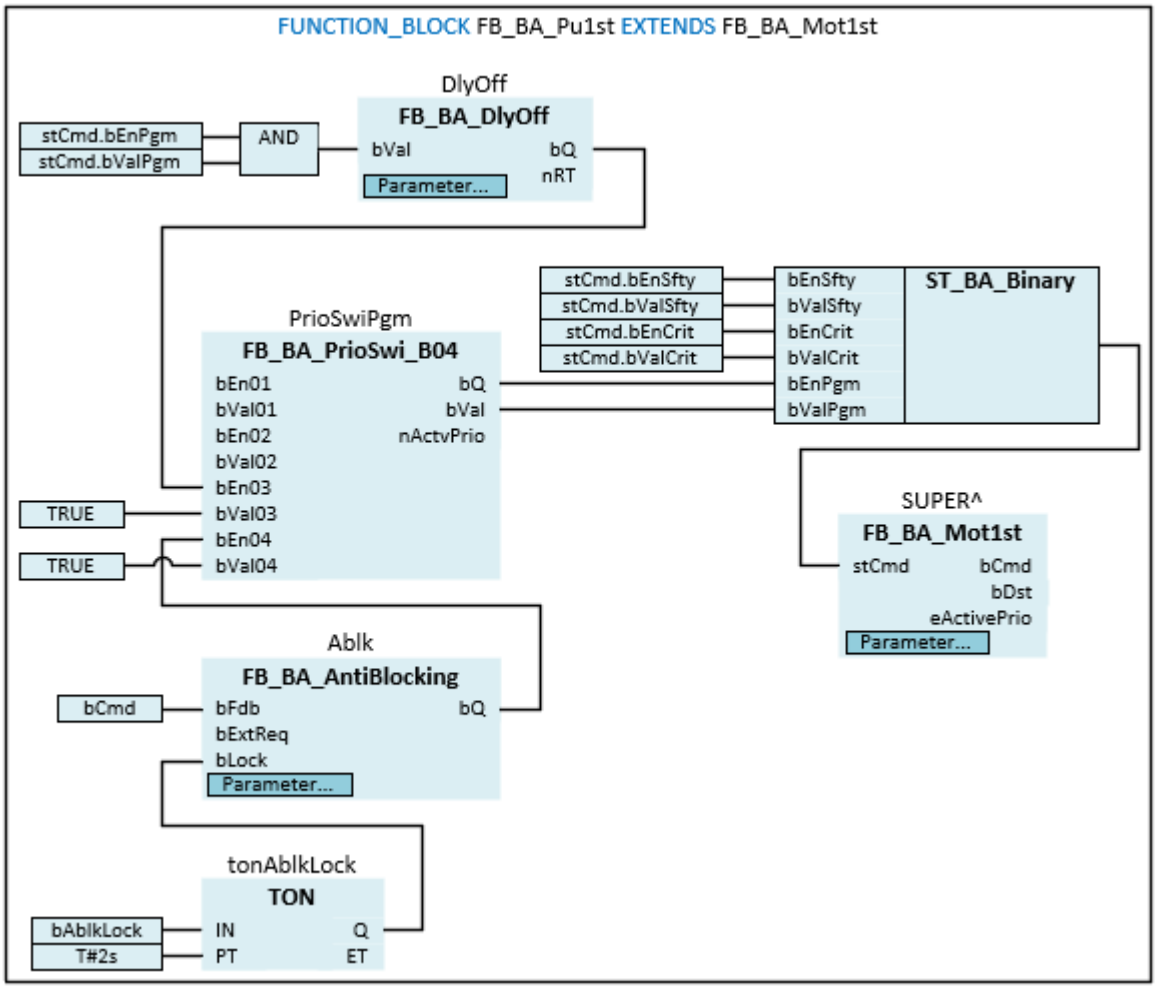


Das Template dient der Ansteuerung einer einstufigen Pumpe mit binären Ein- und Ausgängen. Es besteht im Wesentlichen aus der Basisklasse FB_BA_Mot1st [▶ 1079], der Ausschaltverzögerung DlyOff [▶ 1072] und der Antiblockierschutzfunktion Ablk [▶ 1015].

Die Pumpe wird extern über die Prioritäten der Kommandostruktur *stCmd* der Basisklasse FB_BA_Mot1st [▶ 1079] oder intern durch die Antiblockierschutzfunktion Ablk [▶ 1015] eingeschaltet. Die externe Anforderung über die Priorität „Programm“ wird durch das Template *DlyOff* verzögert ausgeschaltet.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Pu1st EXTENDS FB_BA_Mot1st
VAR_INPUT
  bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  DlyOff         : FB_BA_DlyOff;
  Ablk           : FB_BA_AntiBlocking;
END_VAR
VAR
  PrioSwiPgm     : FB_BA_PrioSwi_B04;
  tonAblkLock    : TON;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bAblkLock	BOOL	Ein TRUE an dieser Eingangsvariable unterbricht die Antiblockierschutzfunktion Ablk [▶ 1015]. Es soll verhindert werden, dass Pumpen und Ventile gleichzeitig einen Antiblockierschutz-Impuls bekommen.

Eingänge CONSTANT

Name	Typ	Beschreibung
DlyOff	FB_BA_DlyOff [▶ 1072]	Das Template dient als Pumpennachlauf.
Ablk	FB_BA_AntiBlocking [▶ 1015]	Antiblockierschutz.

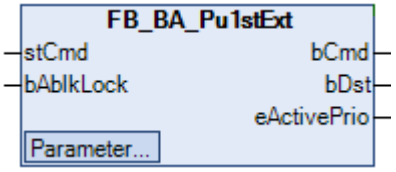
Variablen

Name	Typ	Beschreibung
PrioSwiPgm	FB_BA_PrioSwi_B04 [▶ 443]	Der Prioritätenschalter *PrioSwiPgm* ermittelt anhand der Ausschaltverzögerung DlyOff, der Antiblockierschutzfunktion Ablk [▶ 1015] und der Priorität Programm der Kommandostruktur *stCmd* der Basisklasse FB_BA_Mot1st [▶ 1079] den aktuellen Schaltwert für die Basisklasse FB_BA_Mot1st [▶ 1079].
tonAblkLock	TON	Ausschaltverzögerung des Antiblockierschutzimpulses.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.5.1.2 FB_BA_Pu1stExt

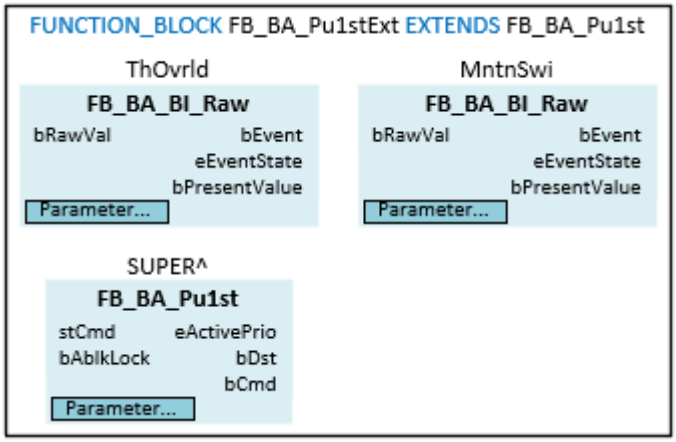


Das Template ist für die Ansteuerung einer einstufigen Pumpe mit binären Ein- und Ausgängen. Es besteht im Wesentlichen aus der Basisklasse FB_BA_Pu1st [▶ 1093].

Der Unterschied zum Template FB_BA_Pu1st [▶ 1093] sind die zusätzlichen BI-Objekte *ThOvrld* und *MntnSwi*.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Pu1stExt EXTENDS FB_BA_Pu1st
VAR_INPUT CONSTANT
  ThOvrld    : FB_BA_BI_Raw;
  MntnSwi    : FB_BA_BI_Raw;
END_VAR
```

Eingänge CONSTANT

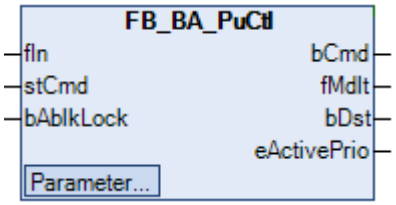
Name	Typ	Beschreibung
ThOvrld	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung der Störung "Thermische Überlastung".
MntnSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Wartungsschalters.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.5.2 Control

6.1.4.2.2.5.2.1 FB_BA_PuCtl



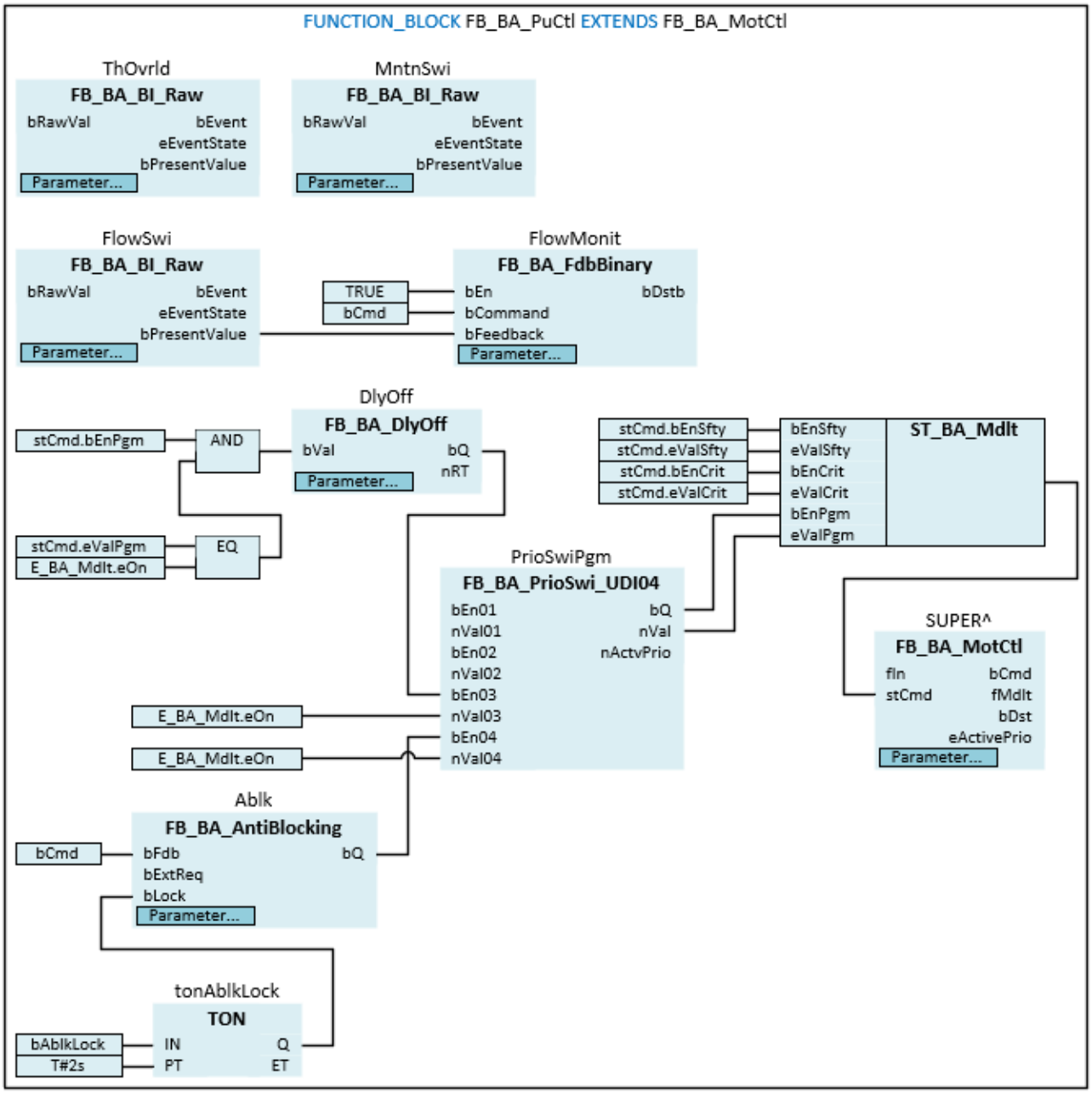
Das Template dient der Ansteuerung und Regelung einer Drehzahl-gesteuerten Pumpe mit analogen und binären Ein- und Ausgängen. Es besteht im Wesentlichen aus der Basisklasse FB_BA_MotCtl [▶ 1085], der Ausschaltverzögerung DlyOff [▶ 1072] und der Antiblockierschutzfunktion Ablk [▶ 1015]. Das Template *FlowMonit* steht für die Überwachung des Strömungswächters *FlowSwi*.

Die Pumpe wird extern über die Prioritäten der Kommandostruktur *stCmd* der Basisklasse FB_BA_MotCtl [▶ 1085] oder intern durch die Antiblockierschutzfunktion Ablk [▶ 1015] eingeschaltet. Die externe Anforderung über die Priorität „Program“ wird durch das Template Pumpennachlauf DlyOff [▶ 1072] verzögert ausgeschaltet.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_PuCtl EXTENDS FB_BA_MotCtl
VAR_INPUT
  bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  ThOvrld        : FB_BA_BI_Raw
  MntnSwi        : FB_BA_BI_Raw;
  FlowSwi        : FB_BA_BI_Raw;
  FlowMonit      : FB_BA_FdbBinary;
  DlyOff         : FB_BA_DlyOff;
  Ablk           : FB_BA_AntiBlocking;
END_VAR
VAR
  PrioSwiPgm     : FB_BA_PrioSwi_UDI04;
  tonAblkLock    : TON;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bAblkLock	BOOL	Ein TRUE an dieser Eingangsvariable unterbricht die Antiblockierschutzfunktion Ablk [▶ 1015]. Es soll verhindert werden, dass Pumpen und Ventile gleichzeitig einen Antiblockierschutz-Impuls bekommen.

Eingänge CONSTANT

Name	Typ	Beschreibung
ThOvrld	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung der Störung "Thermische Überlastung".
MntnSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Wartungsschalters.
FlowSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Strömungswächters.
FlowMonit	FB_BA_FdbBinary [▶ 1036]	Template zur Überwachung des Strömungswächters.
DlyOff	FB_BA_DlyOff [▶ 1072]	Das Template dient als Pumpennachlauf.
Ablk	FB_BA_AntiBlocking [▶ 1015]	Antiblockierschutz.

Variablen

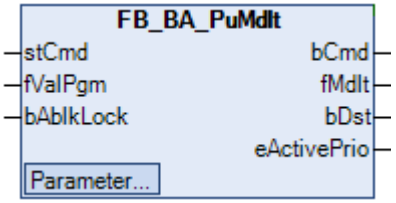
Name	Typ	Beschreibung
PrioSwiPgm	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter *PrioSwiPgm* ermittelt anhand der Ausschaltverzögerung DlyOff, der Antiblockierschutzfunktion Ablk [▶ 1015] und der Priorität Programm der Kommandostruktur *stCmd* der Basisklasse FB_BA_MotCtl [▶ 1085] den aktuellen Modulationswert für die Basisklasse FB_BA_MotCtl [▶ 1085].
tonAblkLock	TON	Ausschaltverzögerung des Antiblockierschutzimpulses.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.5.3 Modulation

6.1.4.2.2.5.3.1 FB_BA_PuMdlt

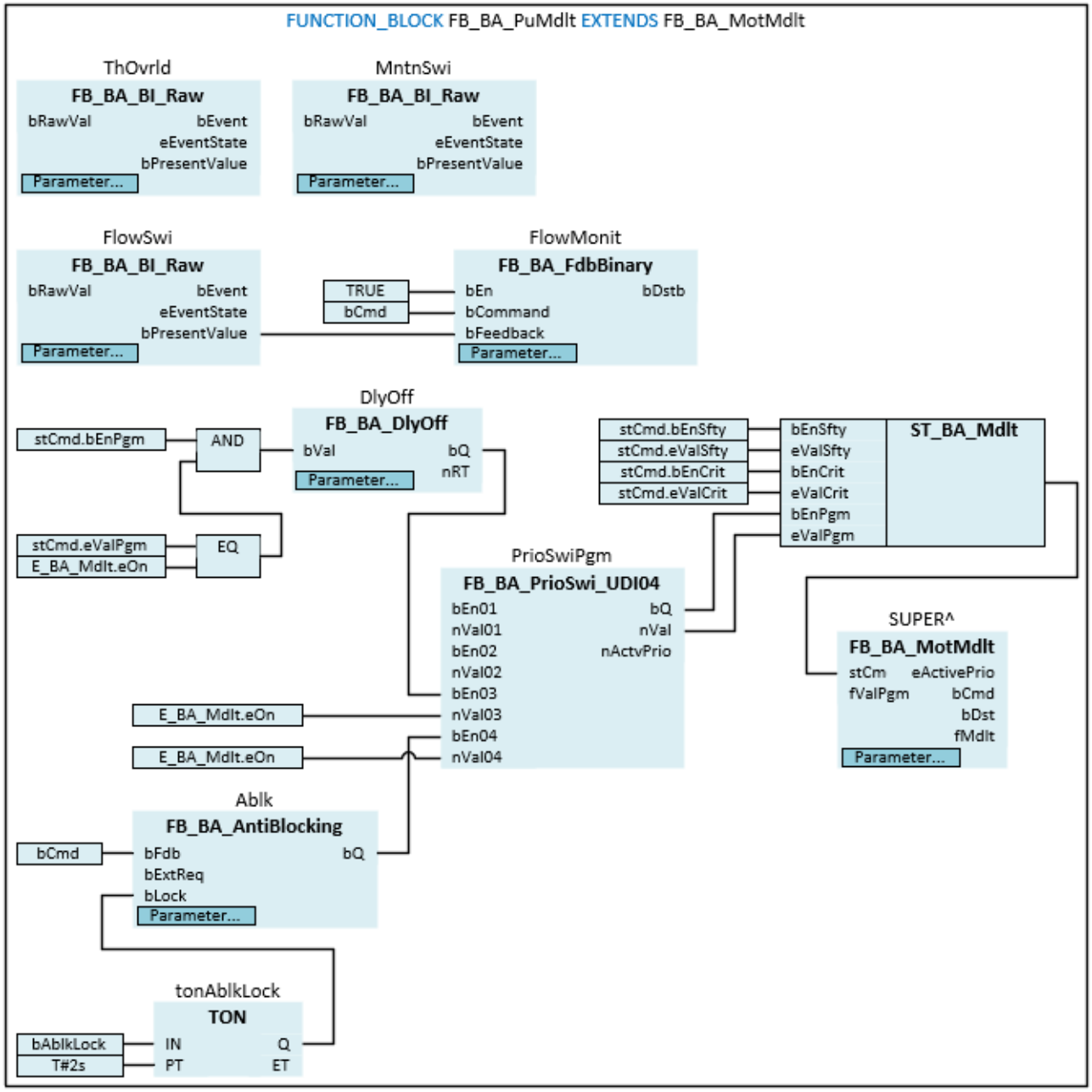


Das Template dient der Ansteuerung einer Drehzahl-gesteuerten Pumpe mit analogen und binären Ein- und Ausgängen. Es besteht im Wesentlichen aus der Basisklasse FB_BA_MotMdlt [▶ 1089], der Ausschaltverzögerung DlyOff [▶ 1072] und der Antiblockierschutzfunktion Ablk [▶ 1015]. Das Template *FlowMonit* steht für die Überwachung des Strömungswächters *FlowSwi*.

Die Pumpe wird extern über die Prioritäten der Kommandostruktur *stCmd* der Basisklasse FB_BA_MotMdlt [▶ 1089] oder intern durch die Antiblockierschutzfunktion Ablk [▶ 1015] eingeschaltet. Die externe Anforderung über die Priorität „Programm“ wird durch das Template Pumpennachlauf DlyOff [▶ 1072] verzögert ausgeschaltet.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_PuMdlt EXTENDS FB_BA_MotMdlt
VAR_INPUT
  bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  ThOvrld        : FB_BA_BI_Raw;
  MntnSwi        : FB_BA_BI_Raw;
  FlowSwi        : FB_BA_BI_Raw;
  FlowMonit      : FB_BA_FdbBinary;
```

```
  DlyOff          : FB_BA_DlyOff;
  Ablk            : FB_BA_AntiBlocking;
END_VAR
VAR
  PrioSwiPgm      : FB_BA_PrioSwi_UDI04;
  tonAblkLock     : TON;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bAblkLock	BOOL	Ein TRUE an dieser Eingangsvariable unterbricht die Antiblockierschutzfunktion Ablk [▶ 1015]. Es soll verhindert werden, dass Pumpen und Ventile gleichzeitig einen Antiblockierschutz-Impuls bekommen.

Eingänge CONSTANT

Name	Typ	Beschreibung
ThOvrld	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung der Störung "Thermische Überlastung".
MntnSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Wartungsschalters.
FlowSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Strömungswächters.
FlowMonit	FB_BA_FdbBinary [▶ 1036]	Template zur Überwachung des Strömungswächters.
DlyOff	FB_BA_DlyOff [▶ 1072]	Das Template dient als Pumpennachlauf.
Ablk	FB_BA_AntiBlocking [▶ 1015]	Antiblockierschutz.

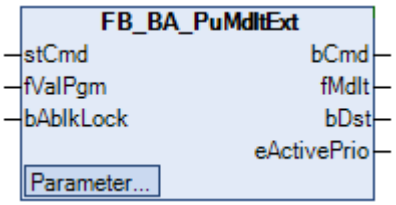
Variablen

Name	Typ	Beschreibung
PrioSwiPgm	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter *PrioSwiPgm* ermittelt anhand der Ausschaltverzögerung DlyOff, der Antiblockierschutzfunktion Ablk [▶ 1015] und der Priorität Programm der Kommandostruktur *stCmd* der Basisklasse FB_BA_Mot1st [▶ 1079] den aktuellen Schaltwert für die Basisklasse FB_BA_Mot1st [▶ 1079].
tonAblkLock	TON	Ausschaltverzögerung des Antiblockierschutzimpulses.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.5.3.2 FB_BA_PuMdltExt

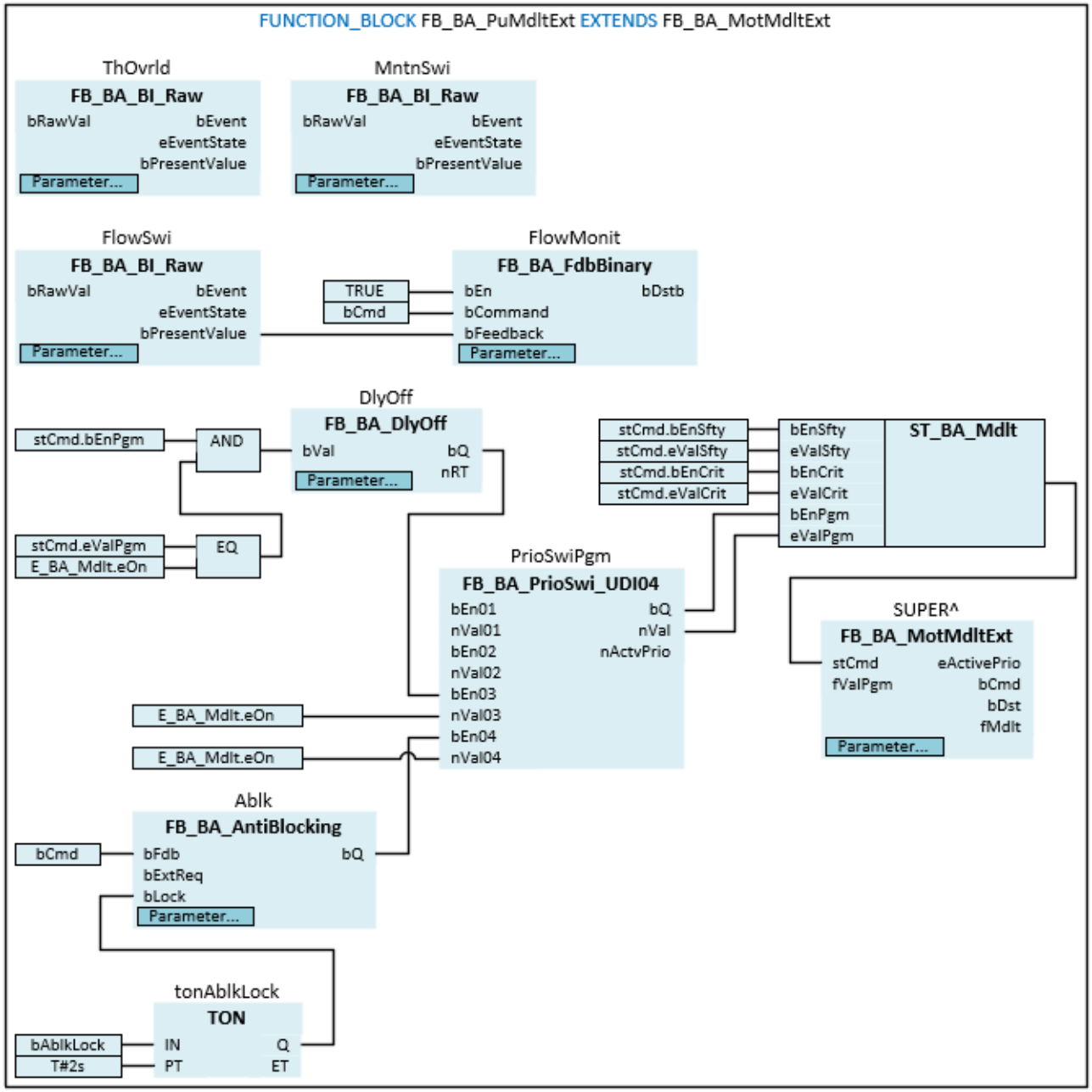


Das Template dient der Ansteuerung einer Drehzahl-gesteuerten Pumpe mit analogen und binären Ein- und Ausgängen. Es besteht im Wesentlichen aus der Basisklasse FB_BA_MotMdltExt [▶ 1092], der Ausschaltverzögerung DlyOff [▶ 1072] und der Antiblockierschutzfunktion Ablk [▶ 1015]. Das Template *FlowMonit* steht für die Überwachung des Strömungswächters *FlowSwi*.

Die Pumpe wird extern über die Prioritäten der Kommandostruktur *stCmd* der Basisklasse FB_BA_MotMdltExt [▶ 1092] oder intern durch die Antiblockierschutzfunktion Ablk [▶ 1015] eingeschaltet. Die externe Anforderung über die Priorität „Programm“ wird durch das Template Pumpennachlauf DlyOff [▶ 1072] verzögert ausgeschaltet.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_PuMdlt EXTENDS FB_BA_MotMdlt
VAR_INPUT
  bAblkLock      : BOOL;
END_VAR
VAR_INPUT CONSTANT
  ThOvrld        : FB_BA_BI_Raw;
  MntnSwi        : FB_BA_BI_Raw;
  FlowSwi        : FB_BA_BI_Raw;
  FlowMonit      : FB_BA_FdbBinary;
  DlyOff         : FB_BA_DlyOff;
  Ablk           : FB_BA_AntiBlocking;
END_VAR
VAR
  PrioSwiPgm     : FB_BA_PrioSwi_UDI04;
  tonAblkLock    : TON;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bAblkLock	BOOL	Ein TRUE an dieser Eingangsvariable unterbricht die Antiblockierschutzfunktion Ablk [▶ 1015]. Es soll verhindert werden, dass Pumpen und Ventile gleichzeitig einen Antiblockierschutz-Impuls bekommen.

Eingänge CONSTANT

Name	Typ	Beschreibung
ThOvrld	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung der Störung "Thermische Überlastung".
MntnSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Wartungsschalters.
FlowSwi	FB_BA_BI_Raw [▶ 211]	Das Binär-Eingangs-Objekt dient der Verarbeitung eines Strömungswächters.
FlowMonit	FB_BA_FdbBinary [▶ 1036]	Template zur Überwachung des Strömungswächters.
DlyOff	FB_BA_DlyOff [▶ 1072]	Das Template dient als Pumpennachlauf.
Ablk	FB_BA_AntiBlocking [▶ 1015]	Antiblockierschutz.

Variablen

Name	Typ	Beschreibung
PrioSwiPgm	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter *PrioSwiPgm* ermittelt anhand der Ausschaltverzögerung DlyOff, der Antiblockierschutzfunktion Ablk [▶ 1015] und der Priorität Programm der Kommandostruktur *stCmd* der Basisklasse FB_BA_Mot1st [▶ 1079] den aktuellen Schaltwert für die Basisklasse FB_BA_Mot1st [▶ 1079].
tonAblkLock	TON	Ausschaltverzögerung des Antiblockierschutzimpulses.

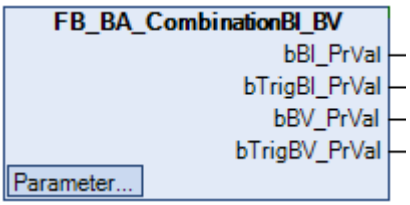
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.2.1.0

6.1.4.2.2.6 Sensor

Templates für Sensoren und Messwerterfassung.

6.1.4.2.2.6.1 FB_BA_CombinationBI_BV



Das Template ist eine Kombination aus einem BI- und BV-Objekt.

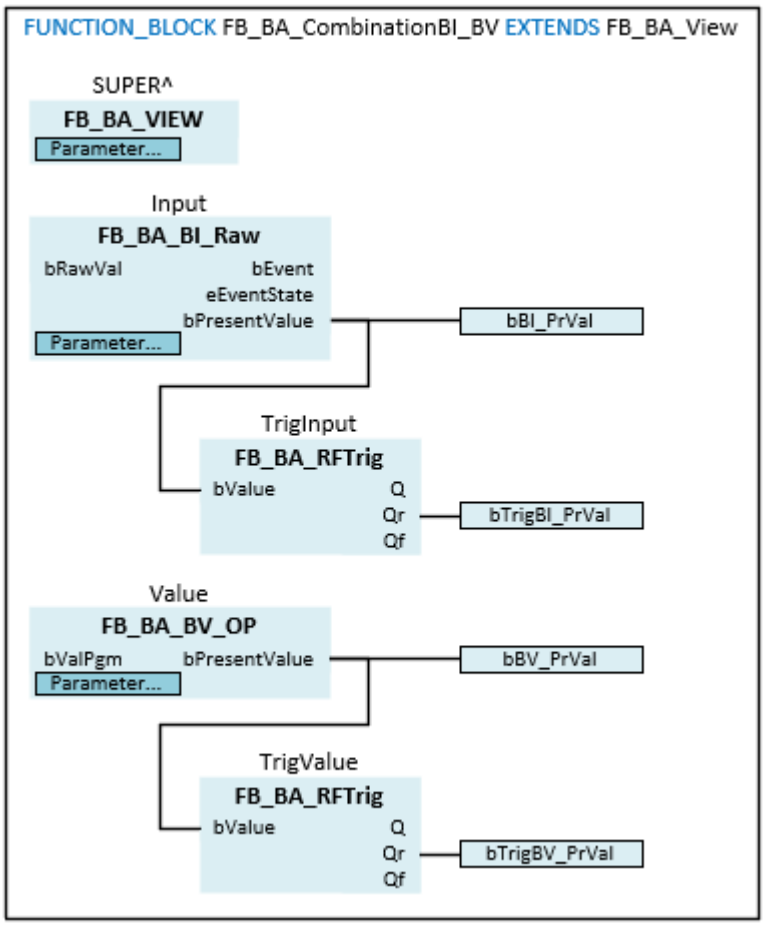
Das Binäre-Eingangs-Objekt *Input* erfasst einen binären Eingangswert von einer Busklemme und gibt diesen als einen Boolean-Prozesswert aus. Zusätzlich wird der Prozesswert von *Input* als steigende Flanke *bTrigBI_PrVal* ausgegeben.

Das Binäre-Werte-Objekt *Value* bildet einen Boolean-Prozesswert ab. Es kann als Schalter oder Taster parametriert werden durch den Parameter E_BA_ToggleMode. Der Ausgangswert des Value Objektes wird auch als steigende Flanke *bTrigBV_PrVal* ausgegeben.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_CombinationBI_BV EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  Input             : FB_BA_BI_Raw;
  Value             : FB_BA_BV_Op;
END_VAR
VAR_OUTPUT
  bBI_PrVal         : BOOL;
```

```
  bTrigBI_PrVal    : BOOL;
  bBV_PrVal        : BOOL;
  bTrigBV_PrVal    : BOOL;
END_VAR
VAR
  TrigInput        : FB_BA_RFTrig;
  TrigValue        : FB_BA_RFTrig;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
Input	FB_BA_BI_Raw [▶ 211]	Binäres-Eingangs-Objekt zur Anzeige eines Prozesswertes.
Value	FB_BA_BV_Op [▶ 221]	Binäres-Werte-Objekt zur Anzeige eines Prozesswertes. Es kann als Schalter oder Taster eingesetzt werden.

Ausgänge

Name	Typ	Beschreibung
bBI_PrVal	BOOL	Aktueller Wert des Binären-Eingangs-Objekt *Input*.
bTrigBI_PrVal	BOOL	Aktueller Wert des Ausgangs *Qr* des Funktionsbausteins *TrigInput*.
bBV_PrVal	BOOL	Aktueller Wert des Binären-Werte-Objekt *Value*
bTrigBV_PrVal	BOOL	Aktueller Wert des Ausgangs *Qr* des Funktionsbausteins *TrigValue*.

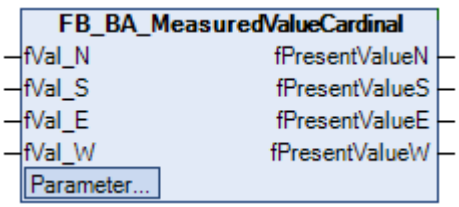
Variablen

Name	Typ	Beschreibung
TrigInput	FB_BA_RFTrig	Der Funktionsbaustein erzeugt aus dem Ausgangssignal des Binären-Eingangs-Objektes *Input* eine steigende Flanke.
TrigValue	FB_BA_RFTrig	Der Funktionsbaustein erzeugt aus dem Ausgangssignal des Binären-Werte-Objektes *Value* eine steigende Flanke.

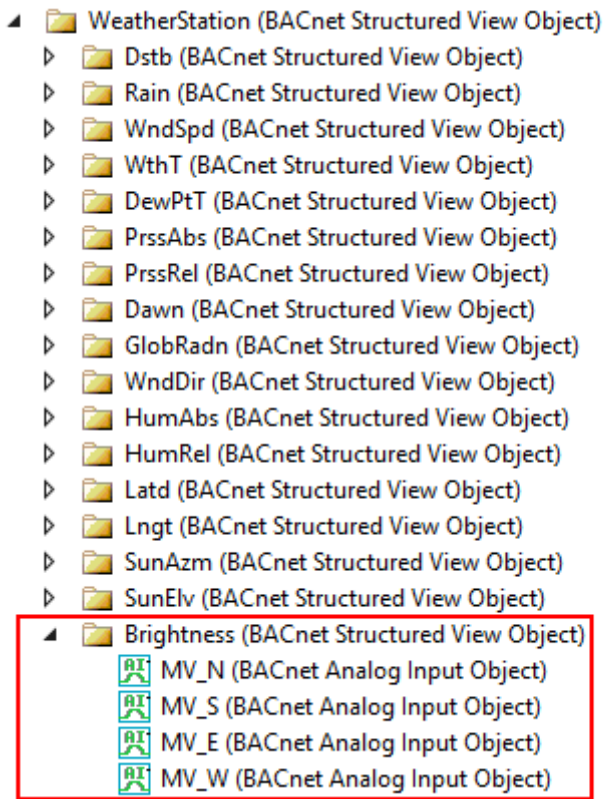
Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.6.2 FB_BA_MeasuredValueCardinal



Dieses Template stellt am Ausgang, in einer hierarchischen Darstellung, eine Sammlung von vier Werten dar, welche den Himmelsrichtungen (engl. "cardinal points") zugeordnet sind. Mit dieser Vorlage lassen sich auf einer Darstellungsebene, wie in der folgenden Grafik gezeigt, die Außenhelligkeiten für alle vier Himmelsrichtungen darstellen.



Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Syntax

```
FUNCTION_BLOCK FB_BA_MeasuredValueCardinal EXTENDS FB_BA_View
VAR_INPUT
  fVal_N            : REAL;
  fVal_S            : REAL;
  fVal_E            : REAL;
  fVal_W            : REAL;
END_VAR
VAR_INPUT CONSTANT
  MV_N              : FB_BA_AI;
  MV_S              : FB_BA_AI;
  MV_E              : FB_BA_AI;
  MV_W              : FB_BA_AI;
END_VAR
VAR_OUTPUT
  fPresentValueN    : REAL;
  fPresentValueS    : REAL;
  fPresentValueE    : REAL;
  fPresentValueW    : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fVal_N	REAL	Analoger Eingangswert für "Nord".
fVal_S	REAL	Analoger Eingangswert für "Süd".
fVal_E	REAL	Analoger Eingangswert für "Ost".
fVal_W	REAL	Analoger Eingangswert für "West".

Eingänge CONSTANT

Name	Typ	Beschreibung
MV_N	FB_BA_AI [▶ 195]	Analogeingabeobjekt für den Wert "Nord".
MV_S	FB_BA_AI [▶ 195]	Analogeingabeobjekt für den Wert "Süd".
MV_E	FB_BA_AI [▶ 195]	Analogeingabeobjekt für den Wert "Ost".
MV_W	FB_BA_AI [▶ 195]	Analogeingabeobjekt für den Wert "West".

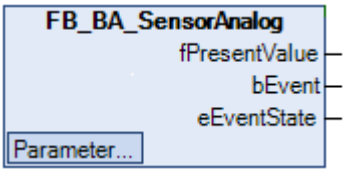
Ausgänge

Name	Typ	Beschreibung
fPresentValueN	REAL	Ausgabewert für "Nord".
fPresentValueS	REAL	Ausgabewert für "Süd".
fPresentValueE	REAL	Ausgabewert für "Ost".
fPresentValueW	REAL	Ausgabewert für "West".

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

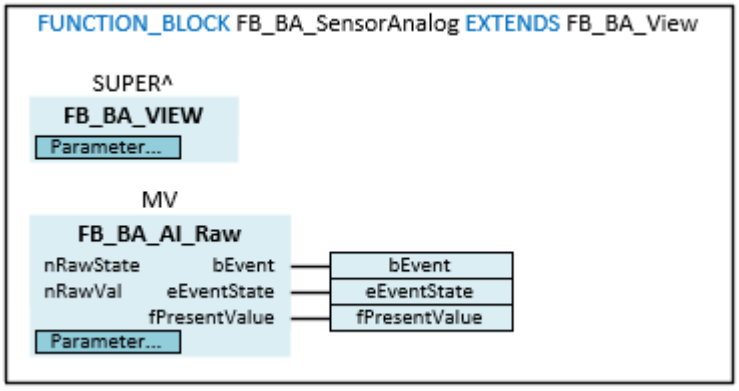
6.1.4.2.2.6.3 FB_BA_SensorAnalog



Der Funktionsbaustein *MV*, innerhalb des Templates, erfasst einen analogen Eingangswert von einer IO-Busklemme und rechnet diesen in einen Real-Prozesswert um.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_SensorAnalog EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  MV               : FB_BA_AI_Raw;
END_VAR
VAR_OUTPUT
  fPresentValue    : REAL;
```

```
  bEvent           : BOOL;
  eEventState      : E_BA_EventState;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
MV	FB_BA_AI_Raw [▶ 200]	Analog-Eingangs-Objekt zur Anzeige eines Prozesswertes.

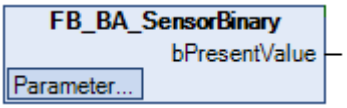
Ausgänge

Name	Typ	Beschreibung
fPresentValue	REAL	Aktueller Wert des Analog-Eingangs-Objekt MV zur Anzeige eines Prozesswertes.
bEvent	BOOL	Ein TRUE zeigt an, dass ein Ereignis ansteht.
eEventState	E_BA_EventState	Gibt den Ereignis Status des Analog-Input-Objektes *MV* aus.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

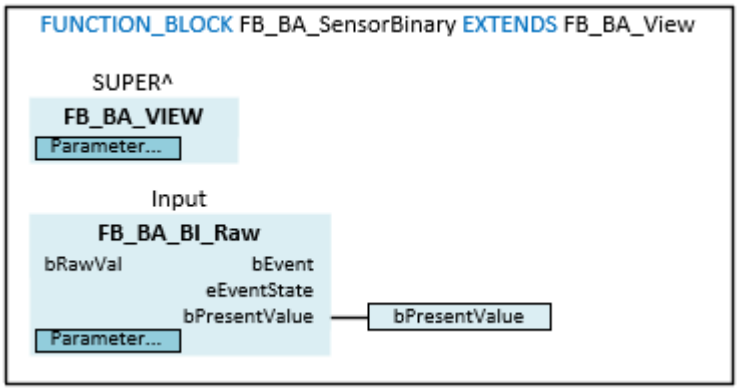
6.1.4.2.2.6.4 FB_BA_SensorBinary



Der Funktionsbaustein Input erfasst einen binären Eingangswert von einer IO-Busklemme und gibt diesen als einen Boolean-Prozesswert aus.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_SensorBinary EXTENDS FB_BA_View
VAR_INPUT CONSTANT
  Input            : FB_BA_BI_Raw;
END_VAR
VAR_OUTPUT
  bPresentValue    : BOOL;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
Input	FB_BA_BI_Raw [▶ 211]	Binäres-Eingangs-Objekt zur Anzeige eines Prozesswertes.

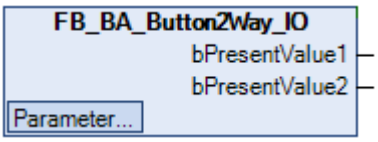
Ausgänge

Name	Typ	Beschreibung
bPresentValue	BOOL	Aktueller Wert des Binären-Eingangs-Objekts *Input* zur Anzeige eines Prozesswertes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.6.5 FB_BA_Button2Way



Das Template dient der direkten Verknüpfung der Eingänge eines Zweifach-Tasters. Die Zustände werden auf einer Hierarchie-Ebene mit zwei Objekten dargestellt und dem Projekt über die Ausgänge *bPresentValue1* und *bPresentValue2* zur Verfügung gestellt. Anwendung findet dieses Template beispielsweise bei der Einbindung von Jalousietastern.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Vererbungshierarchie

FB_BA_Base

FB_BA_BasePublisher

FB_BA_Object [▶ 269]

FB_BA_View [▶ 246]

Syntax

```
FUNCTION_BLOCK FB_BA_Button2Way EXTENDS FB_BA_VIEW
VAR_INPUT
END_VAR
VAR_INPUT CONSTANT
  Swi1              : FB_BA_BI_Raw;
  Swi2              : FB_BA_BI_Raw;
END_VAR
VAR_OUTPUT
  bPresentValue1    : BOOL;
  bPresentValue2    : BOOL;
END_VAR
```

Eingänge CONSTANT

Name	Typ	Beschreibung
Swi1	FB_BA_BI_Raw [▶ 211]	Binäreingabeobjekt für Schalter1.
Swi2	FB_BA_BI_Raw [▶ 211]	Binäreingabeobjekt für Schalter2.

Ausgänge

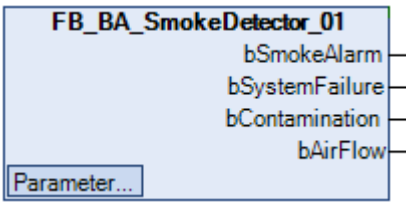
Name	Typ	Beschreibung
bPresentValue1	BOOL	Ausgabe: Zustand Schalter 1.
bPresentValue2	BOOL	Ausgabe: Zustand Schalter 2.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.7 SmokeDetector

6.1.4.2.2.7.1 FB_BA_SmokeDetector_01

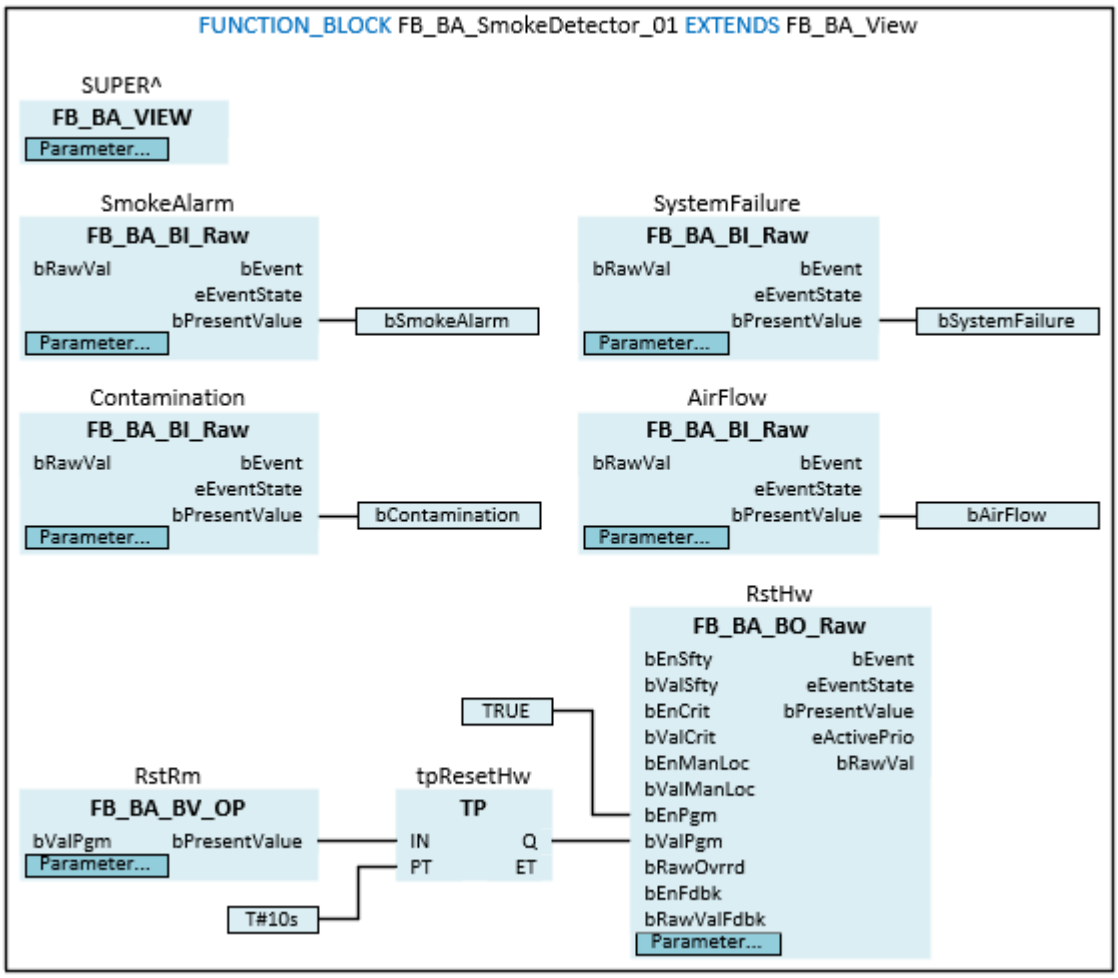


Das Template dient als Anzeige von Störungen und Warnungen eines Kanalrauchmelders.

Durch das Binär-Ausgangs-Objekt *RstHw* können diese Meldungen durch eine Fernentriegelung zurückgesetzt werden.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_SmokeDetector_01 EXTENDS FB_BA_View
VAR_OUTPUT
  bSmokeAlarm       : BOOL;
  bSystemFailure    : BOOL;
  bContamination    : BOOL;
  bAirFlow          : BOOL;
END_VAR
VAR_INPUT CONSTANT
  SmokeAlarm        : FB_BA_BI_Raw;
  SystemFailure     : FB_BA_BI_Raw;
  Contamination     : FB_BA_BI_Raw;
  AirFlow           : FB_BA_BI_Raw;
  RstHw             : FB_BA_BO_Raw;
  RstRm             : FB_BA_BV_Op;
END_VAR
VAR
  tpResetHw         : TP;
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bSmokeAlarm	BOOL	Aktueller Wert des Binär-Eingangs-Objektes *SmokeAlarm*.
bSystemFailure	BOOL	Aktueller Wert des Binär-Eingangs-Objektes *SystemFailure*.
bContamination	BOOL	Aktueller Wert des Binär-Eingangs-Objektes *Contamination*.
bAirFlow	BOOL	Aktueller Wert des Binär-Eingangs-Objektes *AirFlow*.

Eingänge CONSTANT

Name	Typ	Beschreibung
SmokeAlarm	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung eines Rauchalarms.
SystemFailure	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung einer Systemstörung des Kanalrauchmelders.
Contamination	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung einer Verschmutzung des Kanalrauchmelders.
AirFlow	FB_BA_BI_Raw [▶ 211]	Binär-Eingangs-Objekt dient der Verarbeitung einer zu geringen Luftströmung.
RstHw	FB_BA_BO_Raw [▶ 216]	Binär-Ausgangs-Objekt welches eine Fernentriegelung am Rauchmelder durchführt.
RstRm	FB_BA_BV_Op [▶ 221]	Das Binäre-Werte-Objekt *RstRm* löst eine Fernentriegelung am Rauchmelder von der Managementebene aus. Das *RstRm* Objekt ist durch den zusätzlichen Parameter eToggleMode als Tasterobjekt initialisiert.

Variablen

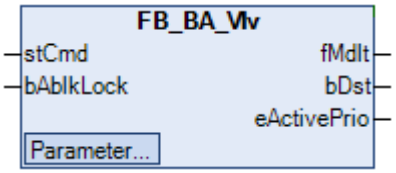
Name	Typ	Beschreibung
tpResetHw	TP	Das Zeitglied verlängert den Quittierimpuls für das Hereinwischen von Relaisschaltungen (z.B. Frostschutzrelais).

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.8 Valve

6.1.4.2.2.8.1 FB_BA_Vlv

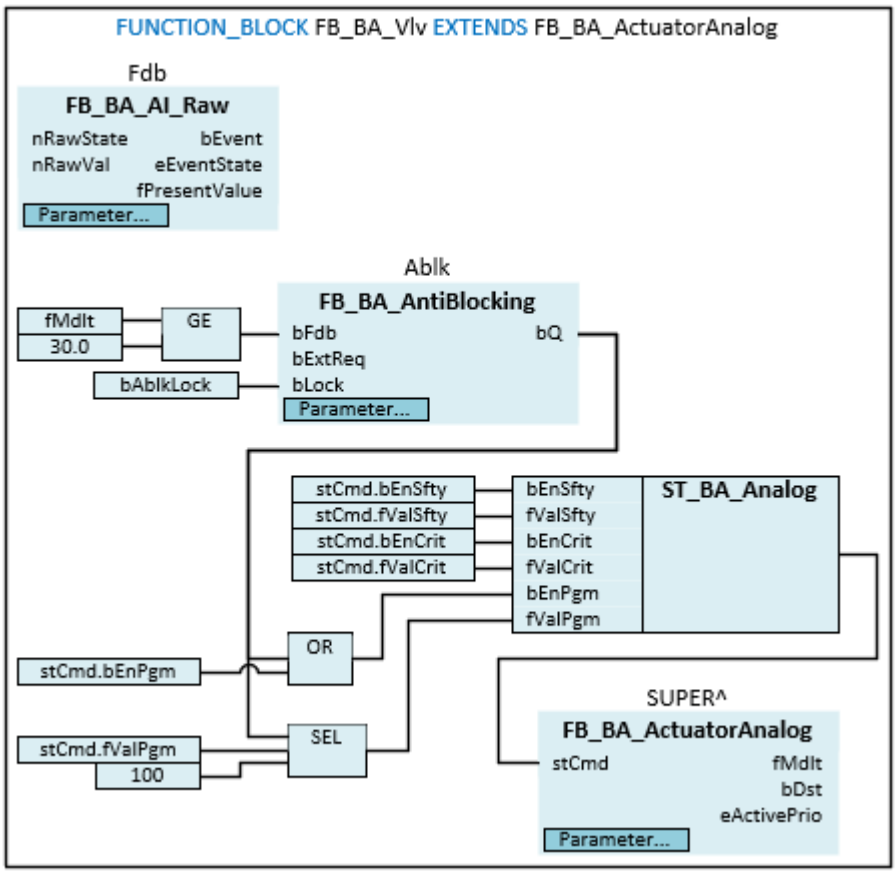


Das Template dient der Ansteuerung eines stetigen Ventils mit analogen Ein- und Ausgängen. Es besteht im Wesentlichen aus der Basisklasse FB_BA_ActuatorAnalog [▶ 1000], der Antiblockierschutzfunktion Ablk [▶ 1015] und dem AI-Objekt *Fdb* zur Erfassung der Stellungsrückmeldung vom Ventil.

Das Ventil wird extern über die Prioritäten der Kommandostruktur *stCmd* der Basisklasse FB_BA_ActuatorAnalog [▶ 1000] oder intern durch die Antiblockierschutzfunktion Ablk [▶ 1015] eingeschaltet.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Vlv EXTENDS FB_BA_ActuatorAnalog
VAR_INPUT
  bAblkLock    : BOOL;
END_VAR
VAR_INPUT CONSTANT
  Ablk         : FB_BA_AntiBlocking;
  Fdb          : FB_BA_AI_Raw;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bAblkLock	BOOL	Ein TRUE an dieser Eingangsvariable unterbricht die Antiblockierschutzfunktion Ablk [▶ 1015]. Es soll verhindert werden, dass Pumpen und Ventile gleichzeitig einen Antiblockierschutz-Impuls bekommen.

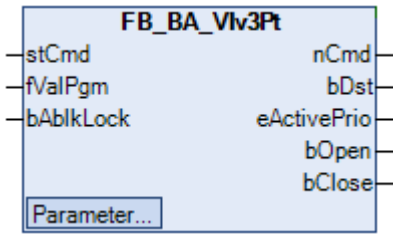
Eingänge CONSTANT

Name	Typ	Beschreibung
Ablk	FB_BA_AntiBlocking [▶ 1015]	Antiblockierschutz.
Fdb	FB_BA_AI_Raw [▶ 200]	Analog-Eingangs-Objekt zur Erfassung der Stellungsrückmeldung des Ventils.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.8.2 FB_BA_Vlv3pt

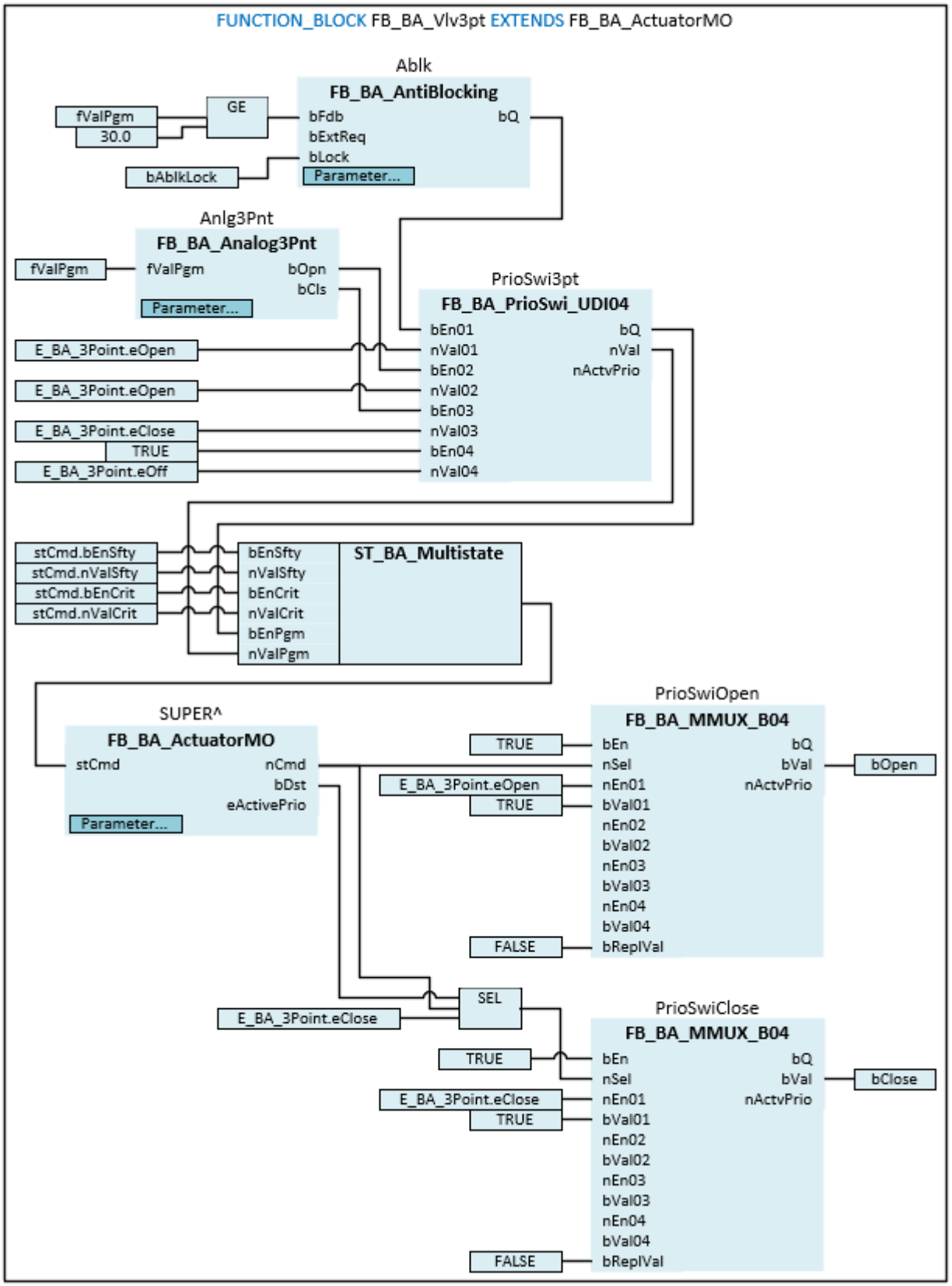


Das Template dient der Ansteuerung eines Dreipunktventils. Es besteht im Wesentlichen aus der Basisklasse FB_BA_ActuatorMO [▶ 1009], der Funktionsbaustein Anlg3Pnt [▶ 417] für die analoge Wandlung des Eingangssignal *fValPgm* in ein Dreipunktsignal und der Antiblockierschutzfunktion Ablk [▶ 1015].

Das Dreipunktventil wird extern über die Prioritäten der Kommandostruktur *stCmd* der Basisklasse FB_BA_ActuatorMO [▶ 1009] oder intern durch die Antiblockierschutzfunktion Ablk [▶ 1015] eingeschaltet.

Die Initialisierung des Templates erfolgt innerhalb der Methode FB_Init.

Blockschaltbild



Syntax

```
FUNCTION_BLOCK FB_BA_Vlv3pt EXTENDS FB_BA_ActuatorMO
VAR_INPUT
  fValPgm         : REAL;
  bAblkLock       : BOOL;
END_VAR
VAR_OUTPUT
```

```
  bOpen           : BOOL;
  bClose          : BOOL;
END_VAR
VAR_INPUT CONSTANT
  Anlg3Pnt        : FB_BA_Analog3Pnt;
  Ablk            : FB_BA_AntiBlocking;
END_VAR
VAR
  PrioSwi3pt      : FB_BA_PrioSwi_UDI04;
  PrioSwiOpen     : FB_BA_MMUX_B04;
  PrioSwiClose    : FB_BA_MMUX_B04;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fValPgm	REAL	Stetiges Eingangssignal für die analoge Wandlung in ein Dreipunktsignal. Dieses kann von einem PID-Regler kommen und einen Wert von 0-100% haben.
bAblkLock	BOOL	Ein TRUE an dieser Eingangsvariable unterbricht die Antiblockierschutzfunktion Ablk [▶ 1015]. Es soll verhindert werden, dass Pumpen und Ventile gleichzeitig einen Antiblockierschutz-Impuls bekommen.

Ausgänge

Name	Typ	Beschreibung
bOpen	BOOL	Variable für die Ansteuerung Auf des 3-Punkt-Ventils. Diese Variable muss mit einer Busklemme verknüpft werden.
bClose	BOOL	Variable für die Ansteuerung Zu des 3-Punkt-Ventils. Diese Variable muss mit einer Busklemme verknüpft werden.

Eingänge CONSTANT

Name	Typ	Beschreibung
Anlg3Pnt	FB_BA_Analog3Pnt [▶ 1013]	Das Template *Anlg3Pnt* wandelt das analoge Eingangssignal *fValPgm* in ein Dreipunktsignal um.
Ablk	FB_BA_AntiBlocking [▶ 1015]	Antiblockierschutz.

Variablen

Name	Typ	Beschreibung
PrioSwi3pt	FB_BA_PrioSwi_UDI04 [▶ 443]	Der Prioritätenschalter *PrioSwi3pt* ermittelt anhand des analogen 3Punkt Wandlers *Anlg3Pnt* und der Antiblockierschutzfunktion Ablk [▶ 1015] den aktuellen Schaltwert für die Priorität „Program“ der Kommandostruktur *stCmd*.
PrioSwiOpen	FB_BA_MMUX_B04 [▶ 438]	Der Multiplexer *PrioSwiOpen* erhält den nummerischen Schaltwert *nCmd* von der Basisklasse FB_BA_ActuatorMO [▶ 1009] und wandelt den Schaltwert in das 3Punkt-Signal *bOpen* um.
PrioSwiClose	FB_BA_MMUX_B04 [▶ 438]	Der Multiplexer *PrioSwiClose* erhält den nummerischen Schaltwert *nCmd* von der Basisklasse FB_BA_ActuatorMO [▶ 1009] und wandelt den Schaltwert in das 3Punkt-Signal *bClose* um.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.9 WeatherStation

Templates zur Datenerfassung von Wetterstationen.

6.1.4.2.2.9.1 Thies

6.1.4.2.2.9.1.1 Einbindung Thies Wetterstation

In dieser Anleitung wird die Einbindung einer Thies Wetterstsation Compact WSC11, 4.9056.10.000, mithilfe des Templates "FB_BA_Weatherstation_Thies" gezeigt.

Benötigte Hardware:

- Thies Wetterstsation Compact WSC11, 4.9056.10.000 (ASCII-Format)
- KL6041, vorkonfiguriert auf 22 Bytes Prozessabbild

Benötigte zusätzliche Bibliothek:

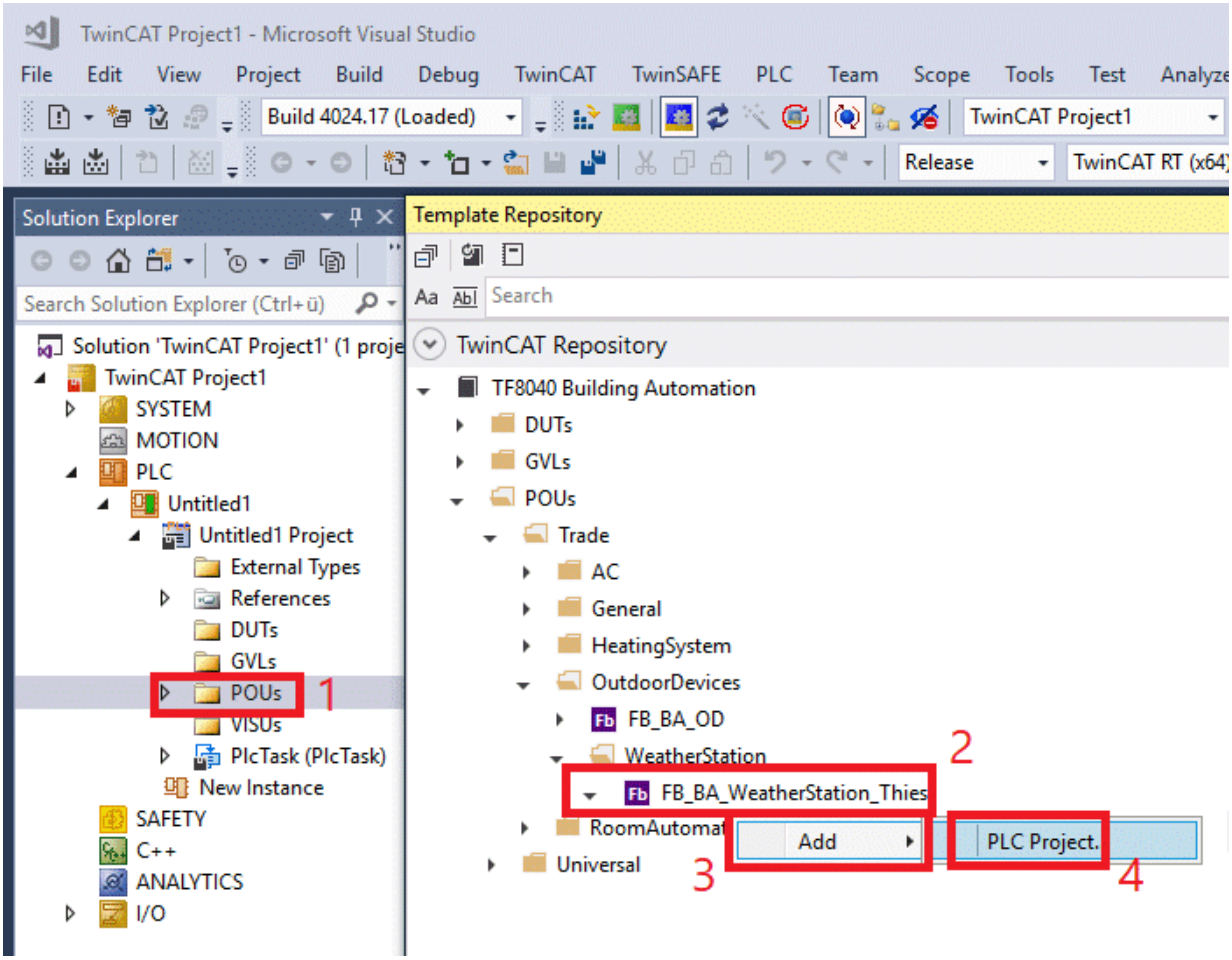
- Tc2_SerialCom

Hinzufügen des Templates:

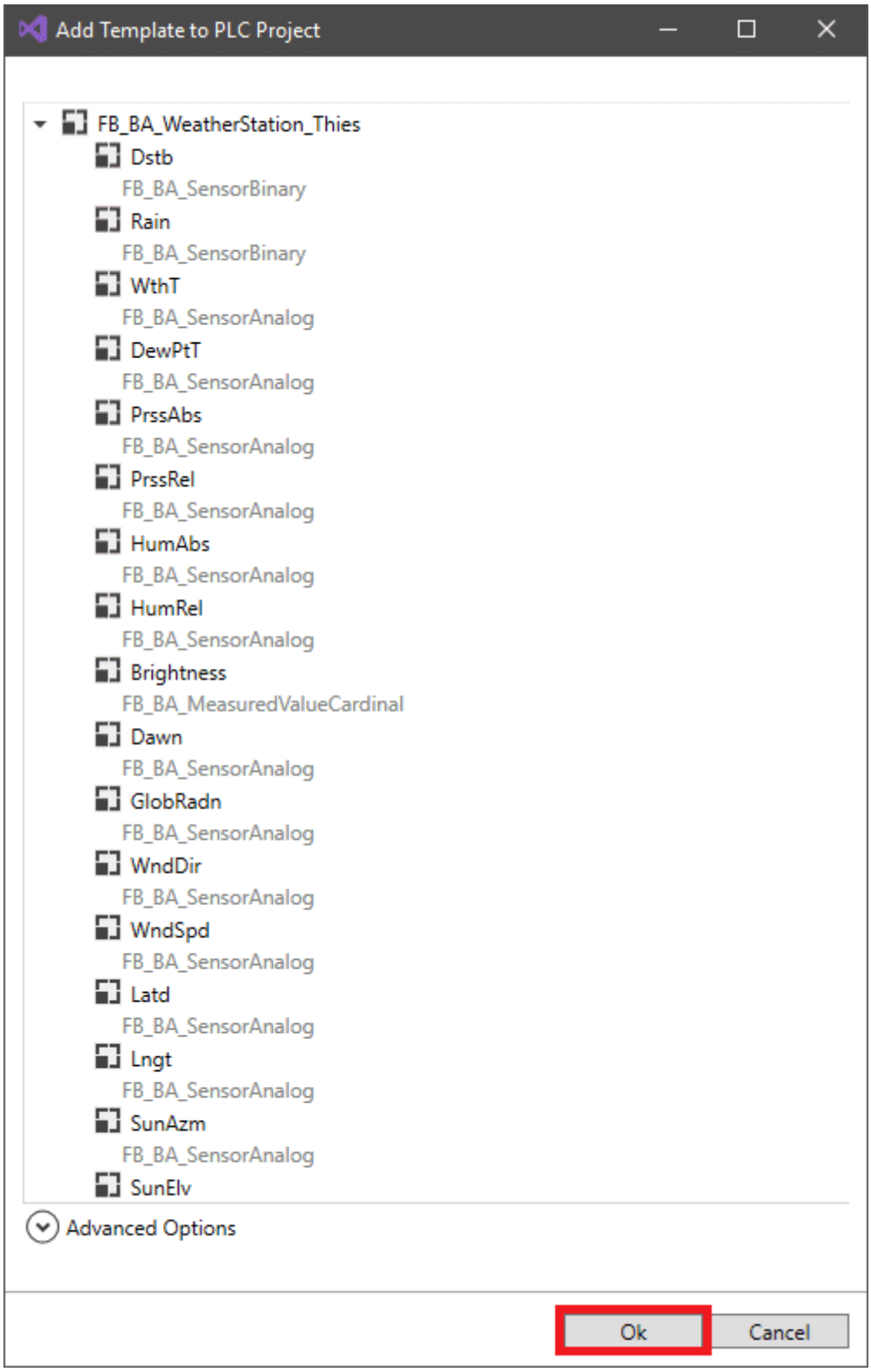
✓ Klicken Sie auf das gewünschte PLC-Projekt (1).

1. Führen Sie im Template-Repository einen Rechtsklick auf das Template **FB_BA_WeatherStation_Thies** aus (2)

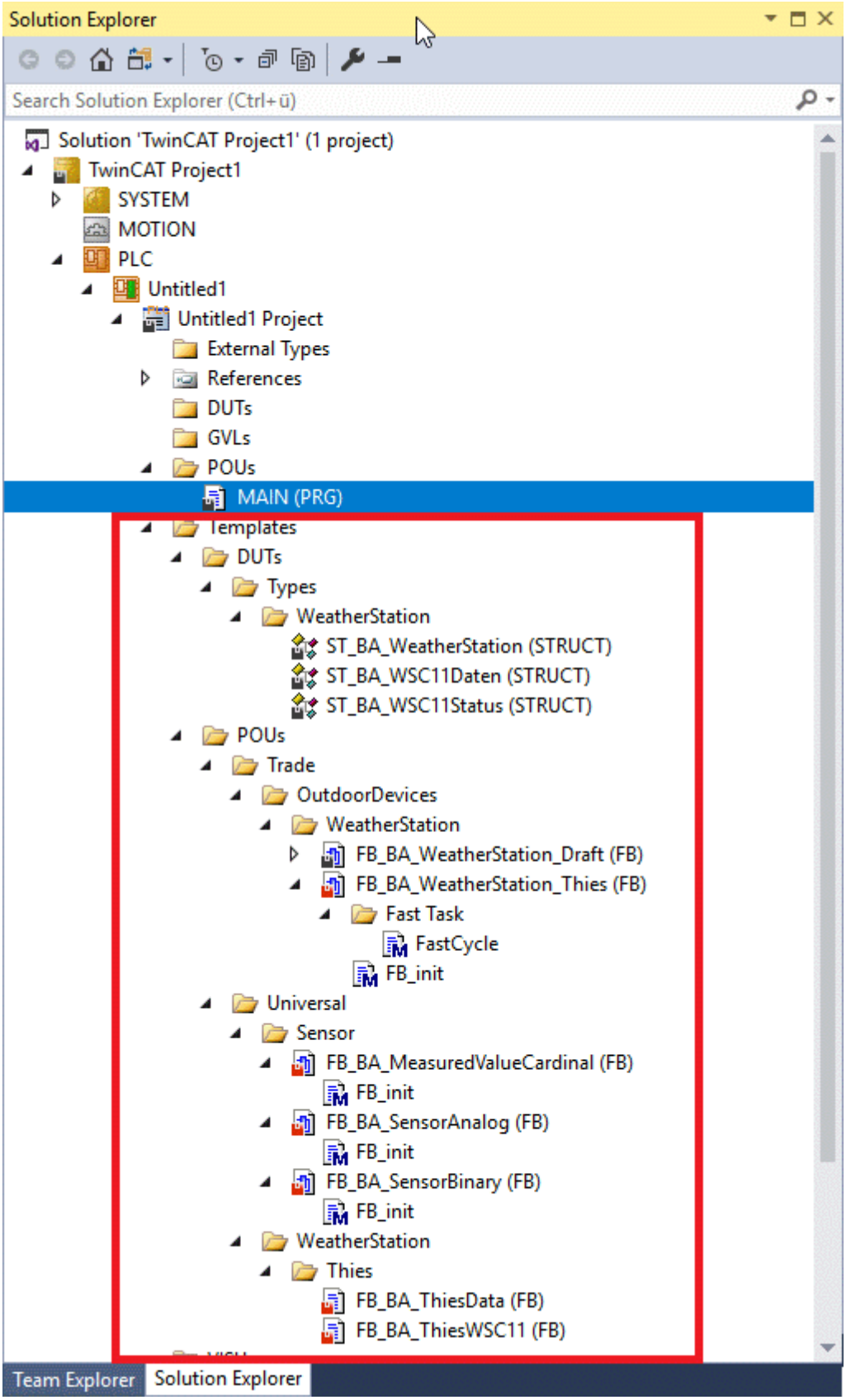
2. Fügen Sie das Template hinzu (3) und (4).



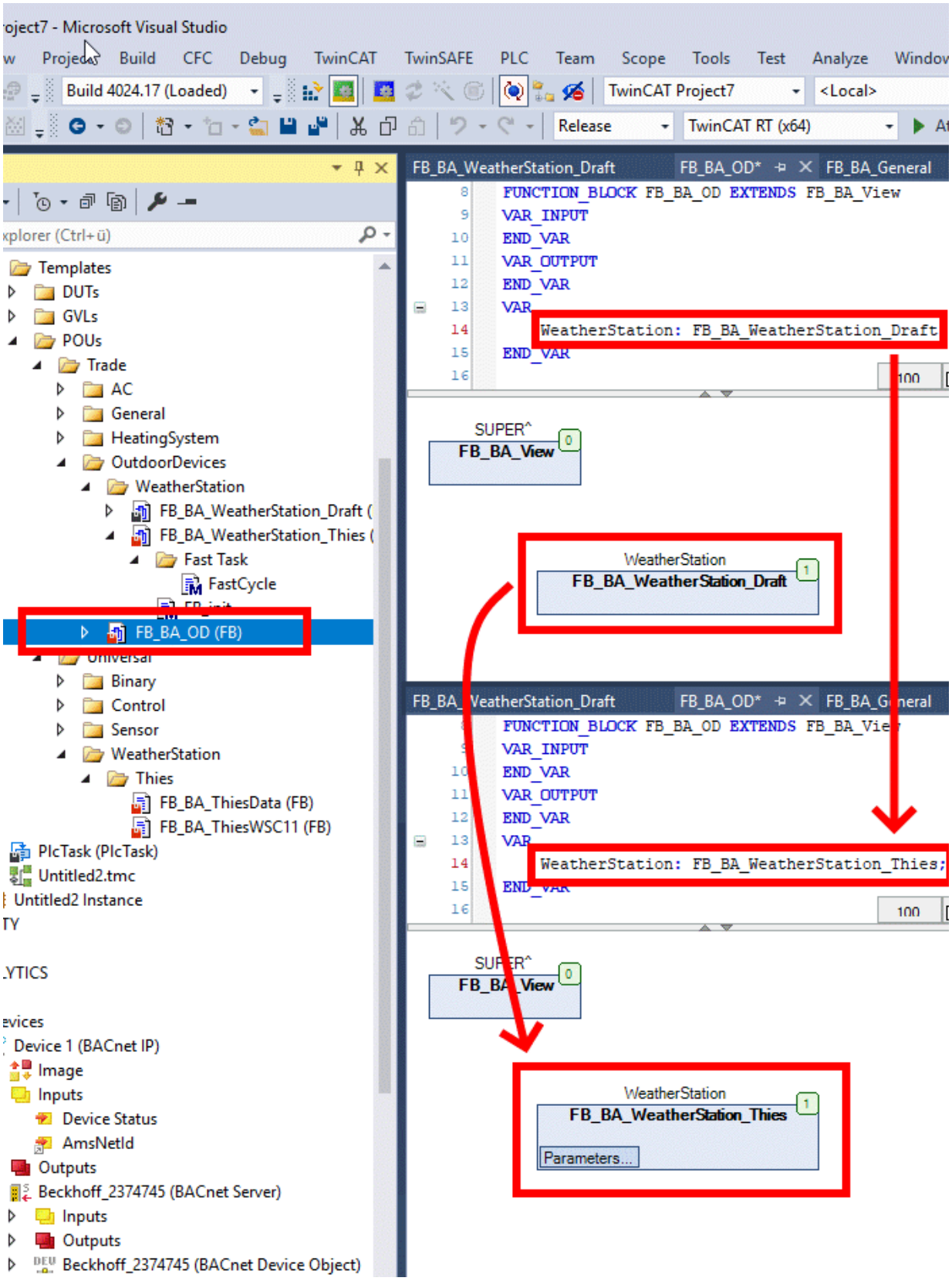
3. Der folgende Dialog mit der Auflistung der zusätzlich implementierten Bausteine öffnet sich. Schließen Sie den Dialog mit **OK**.



4. Alle neuen SPS-Bausteine werden nun in einen Ordner **Templates** hineingezogen und können bei Bedarf im Projekt verteilt werden.



⇨ Im Standard-SPS-Projekt ist bereits ein Wetterstations-Template *FB_BA_Weatherstation_Draft* eingebunden, welches alle notwendigen Anzeigeobjekte vorhält und zur individuellen Verknüpfung gedacht ist.



Hinzufügen einer schnellen Task für die serielle Kommunikation

Die serielle Klemme (KL6041) wird bei einem PLC-Neustart auf die folgenden Kommunikationsparameter konfiguriert:

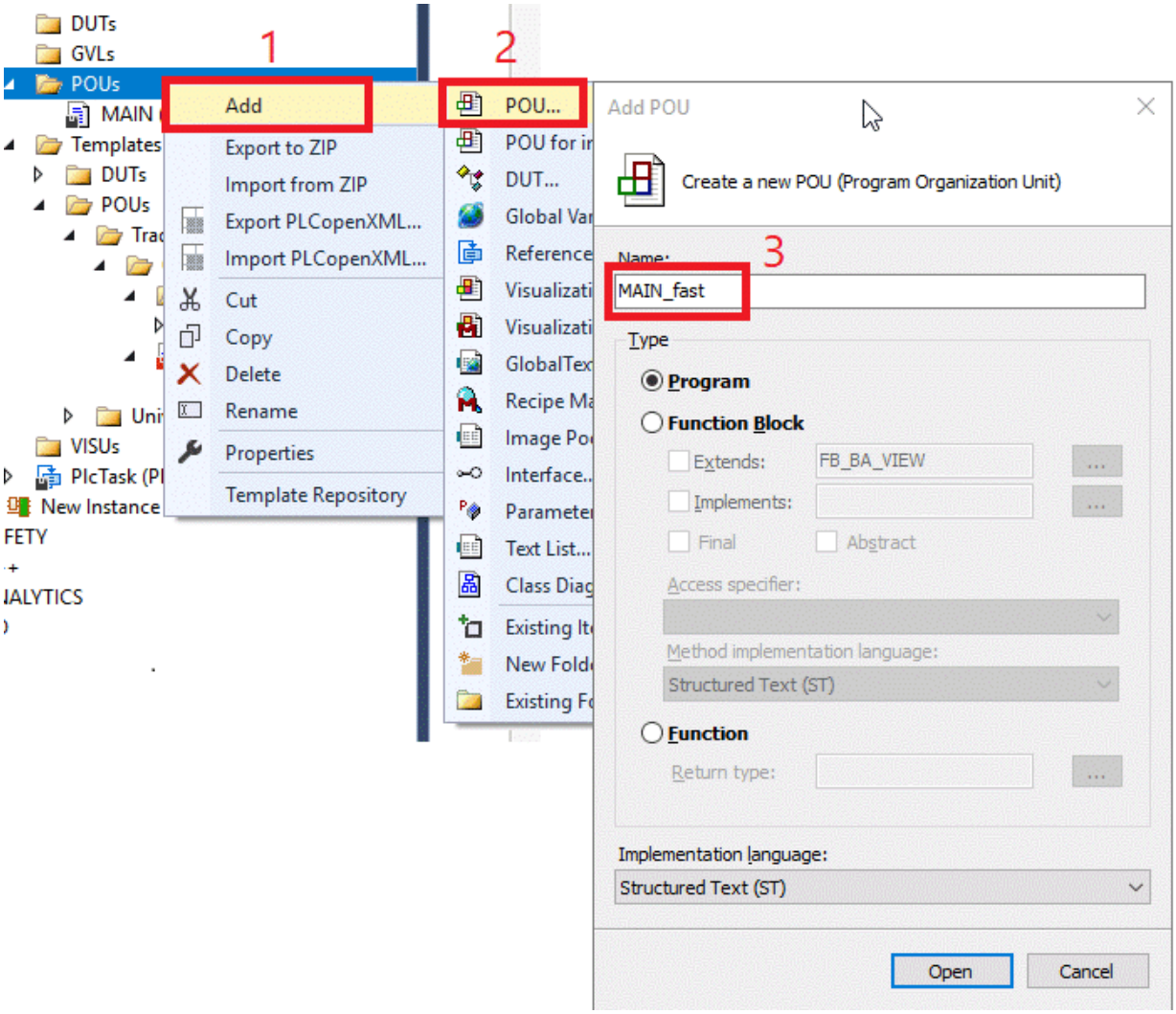
- Baudrate : 9600
- Datenbits : 8
- Parität : keine
- Stoppbits : 1
- Handshake : RS485 HALFDUPLEX

Das Prozessabbild muss zuvor auf 22 Bytes eingestellt sein.

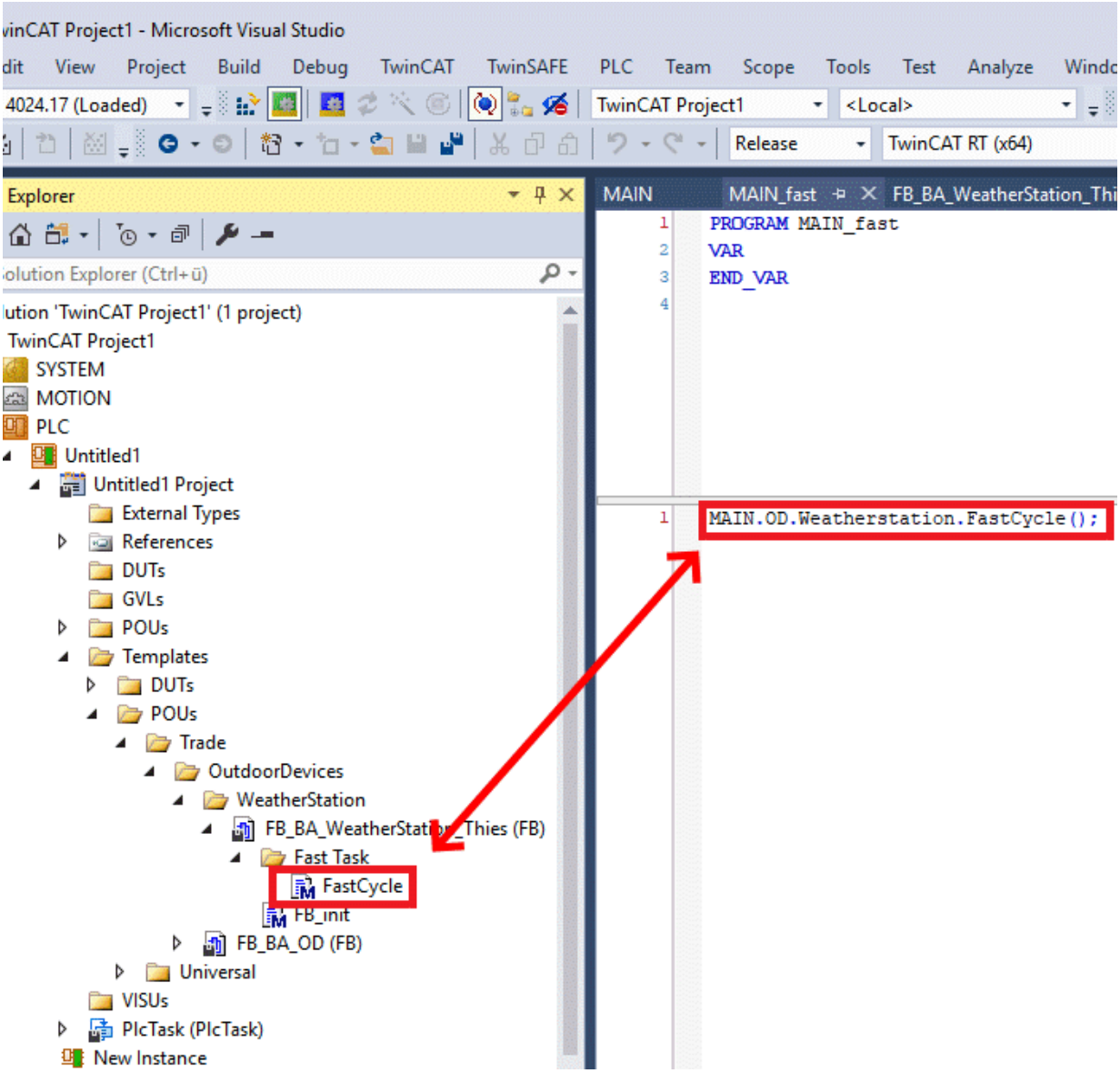
Danach beginnt die eigentliche Kommunikation zwischen Klemme und Wetterstation.

Die dazu benötigten Bausteine sind in der Tc2_SerialCom-Bibliothek vorhanden und werden in der Methode "FastCycle" des Templates "FB_BA_Weatherstation_Thies" aufgerufen. Dieser Aufruf muss einer schnelleren Task als der normalen SPS-Zyklus-Task zugeordnet werden.

1. Öffnen Sie mit einem Rechtklick auf POU das neue Fenster, wählen Sie dort **add** (Hinzufügen) (1) und **POU** (2)**.**

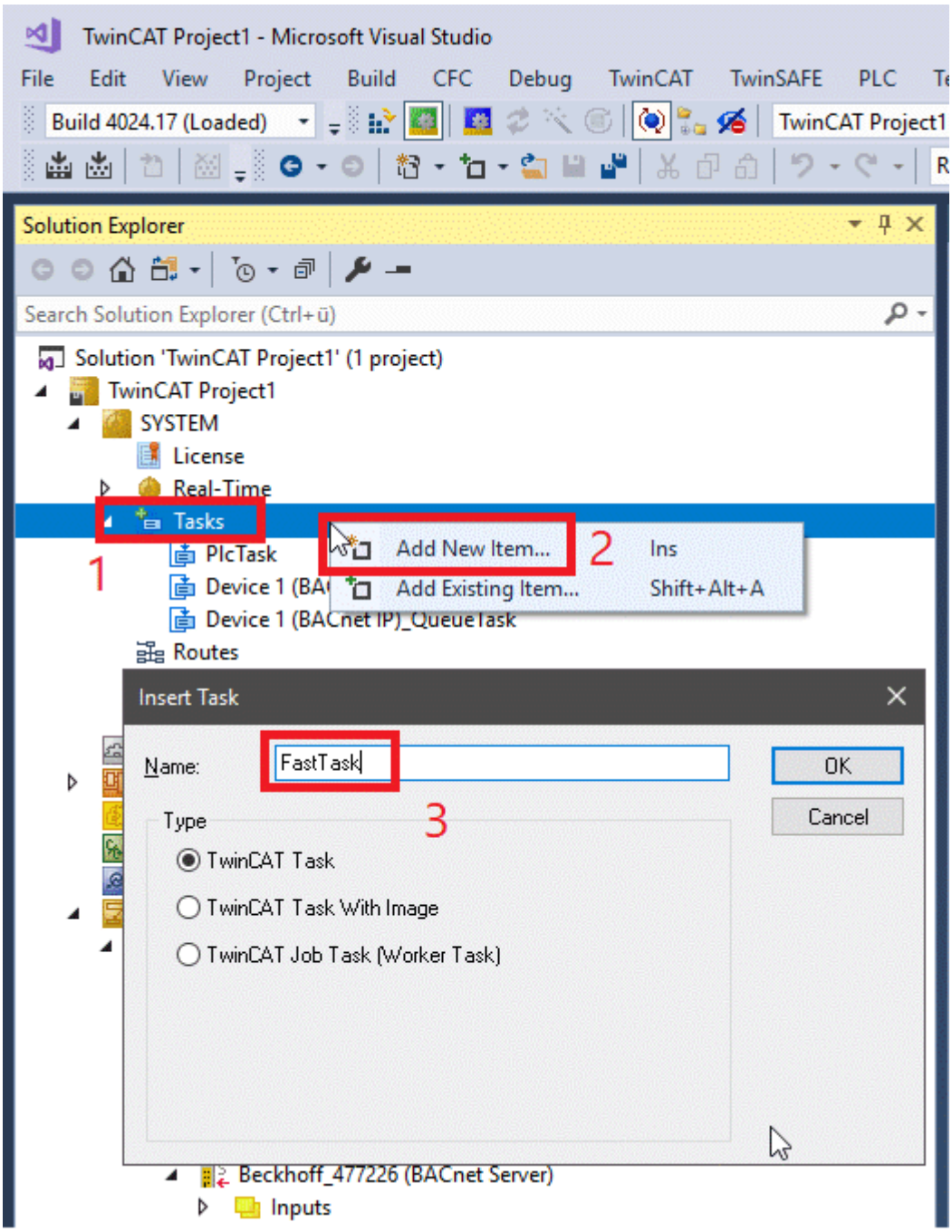


2. Ein Fenster öffnet sich. Geben Sie dort den Namen ein (Beispiel **MAIN_fast**) (3). Der POU-Typ ist ein Programm in strukturiertem Text (ST).

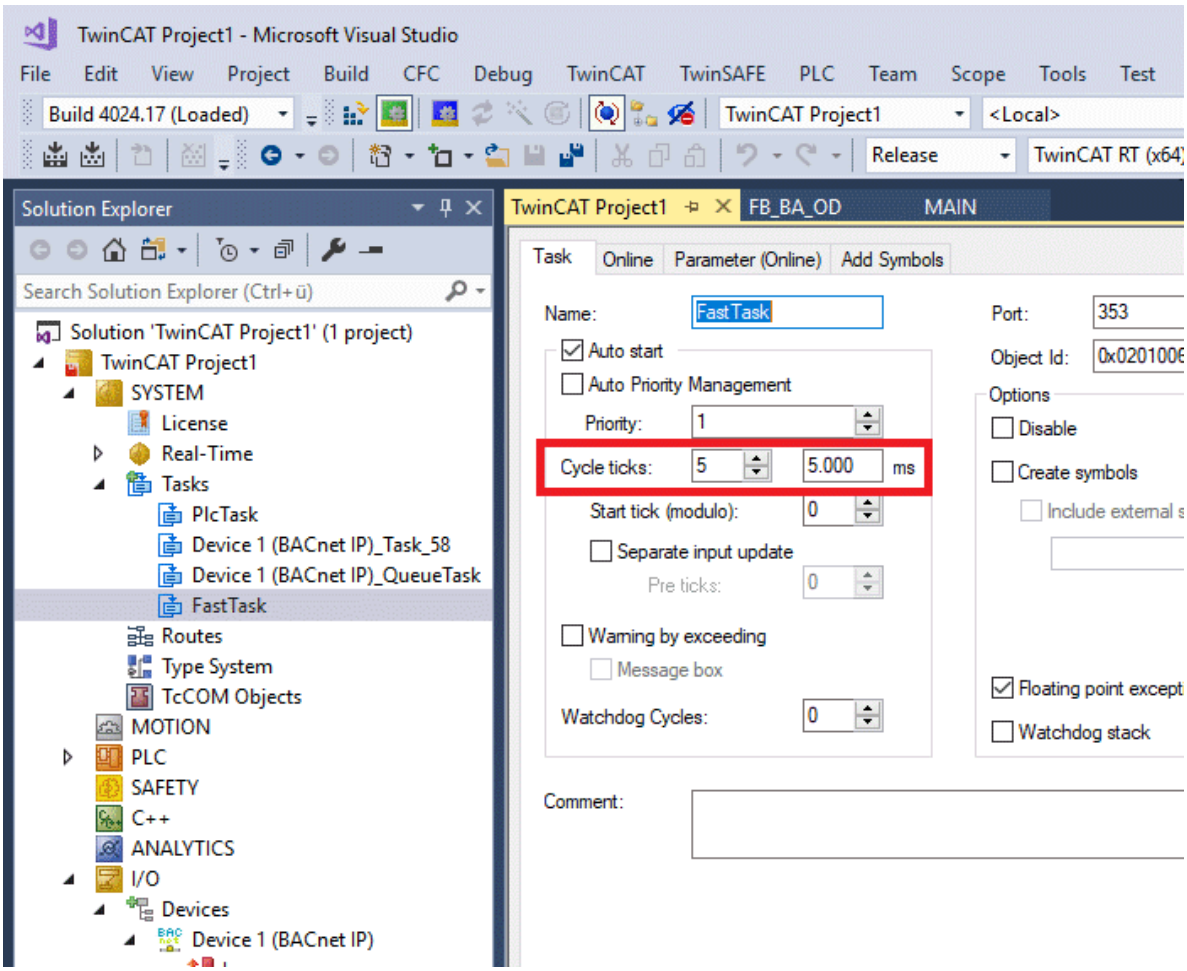


⇨ In diesem Programm wird nun die Methode **FastCycle** aufgerufen.

3. Fügen Sie mit einem Rechtsklick auf **Tasks** (1) ein neues Element hinzu (2). Benennen Sie diese mit einem aussagekräftigen Namen, beispielsweise **FastTask** (3).

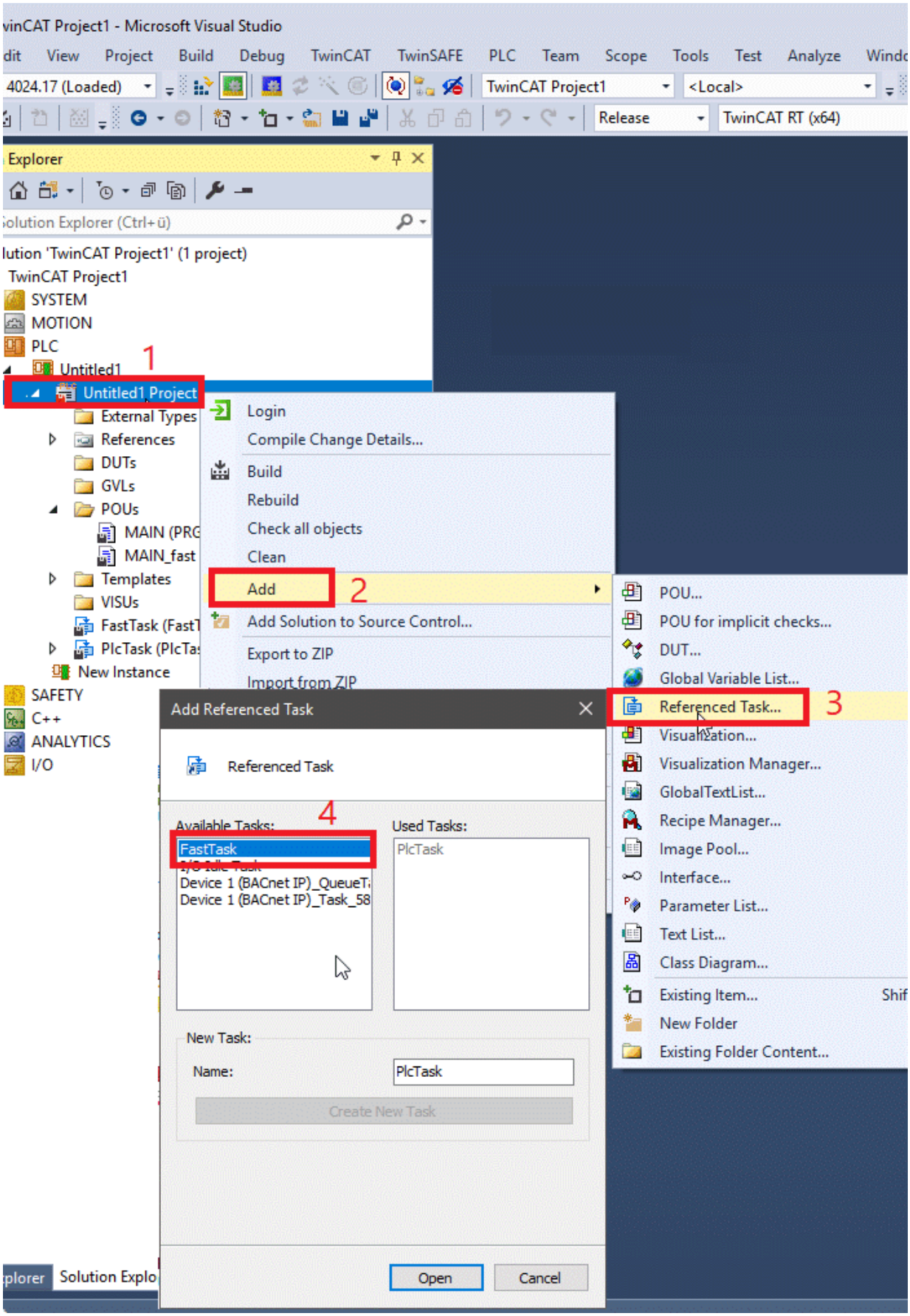


⇨ Diese neue Task muss nun auf eine kleine Zykluszeit eingestellt werden. Die hier gezeigten 5ms sind eine Empfehlung



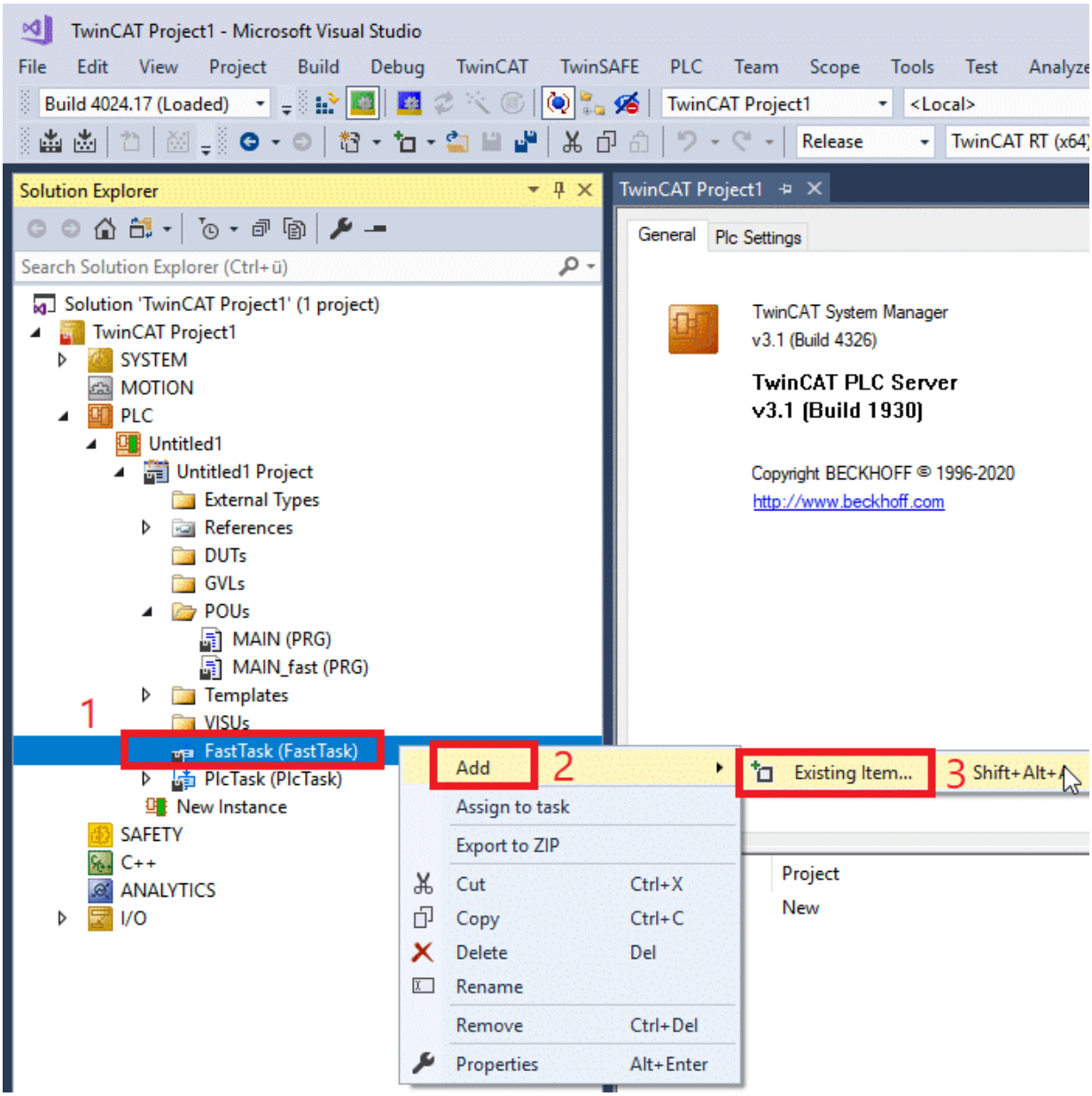
⇨ Damit diese Task dem Programmteil zur Verfügung steht, muss eine Taskreferenz erzeugt werden:

4. Gehen Sie mit einem Rechtsklick auf das Projekt (1), wählen Sie **add (Hinzufügen)** (2) danach **Taskreferenz** (3). Es öffnet sich ein Fenster. Wählen Sie dort die Task aus, auf die sich die Referenz beziehen soll (4).

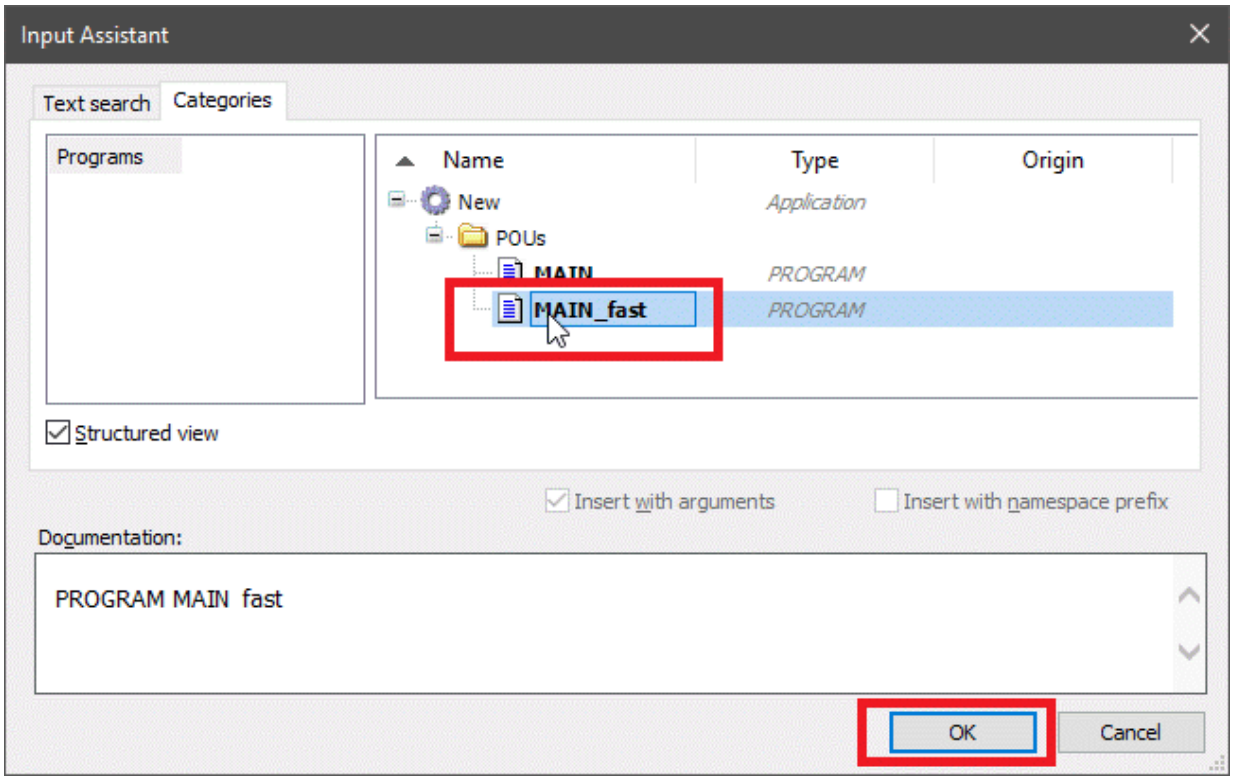


⇨ Die Taskreferenz erscheint nun im SPS-Teil unten. Ihr kann nun der Baustein zugordnet werden, der in dieser Task aufgerufen werden soll.

5. Öffnen Sie mit einem Rechtsklick auf die Task **Fast Task**(1) eine Dialog, wählen Sie dort zunächst **add (Hinzufügen)** (2) und danach **existing item (Vorhandenes Element)** (3).

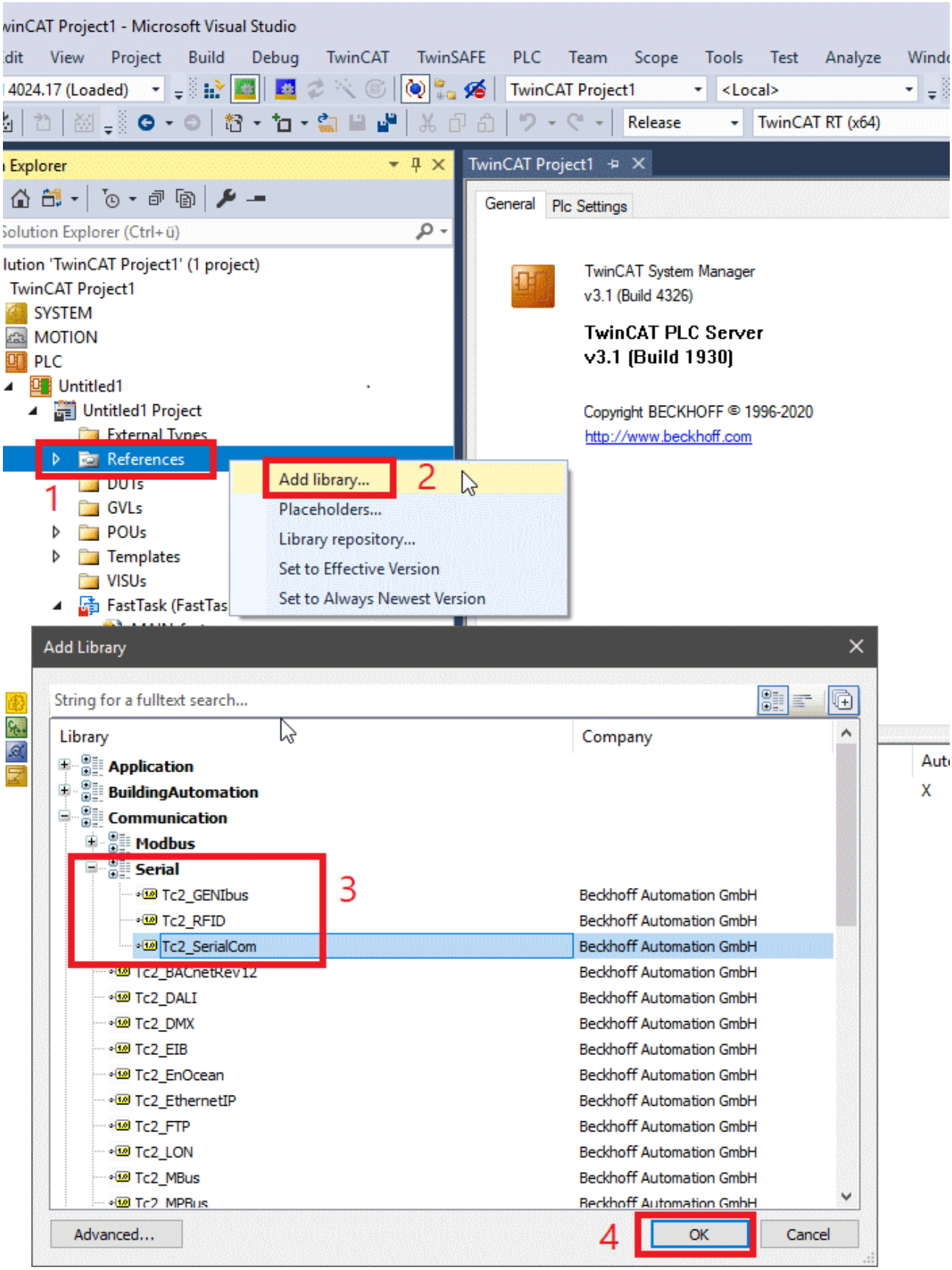


⇨ Es öffnet sich ein Fenster, in dem Sie das Aufrufprogramm wählen können.



Hinzufügen der seriellen Kommunikationsbibliothek

Das Hineinziehen des Templates "FB_BA_WeatherStation_Thies" fügt nicht automatisch die benötigte serielle-Kommunikations-Bibliothek hinzu. Diese muss manuell eingefügt werden:

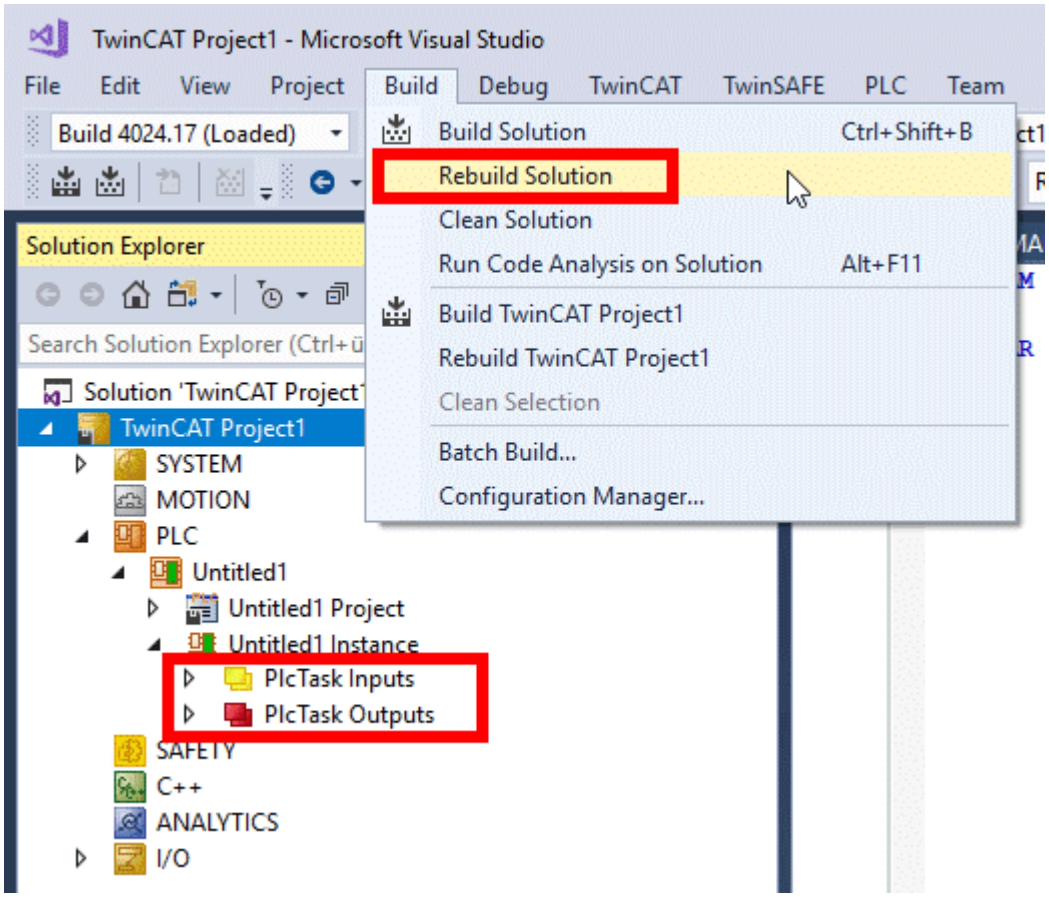


Mit einem Rechtsklick auf "Referenzen" (1) "Bibliothek hinzufügen" (2) auswählen. In dem sich öffnenden Fenster die Bibliothek "Tc2_SerialCom" wählen (3) und mit "OK" bestätigen (4).

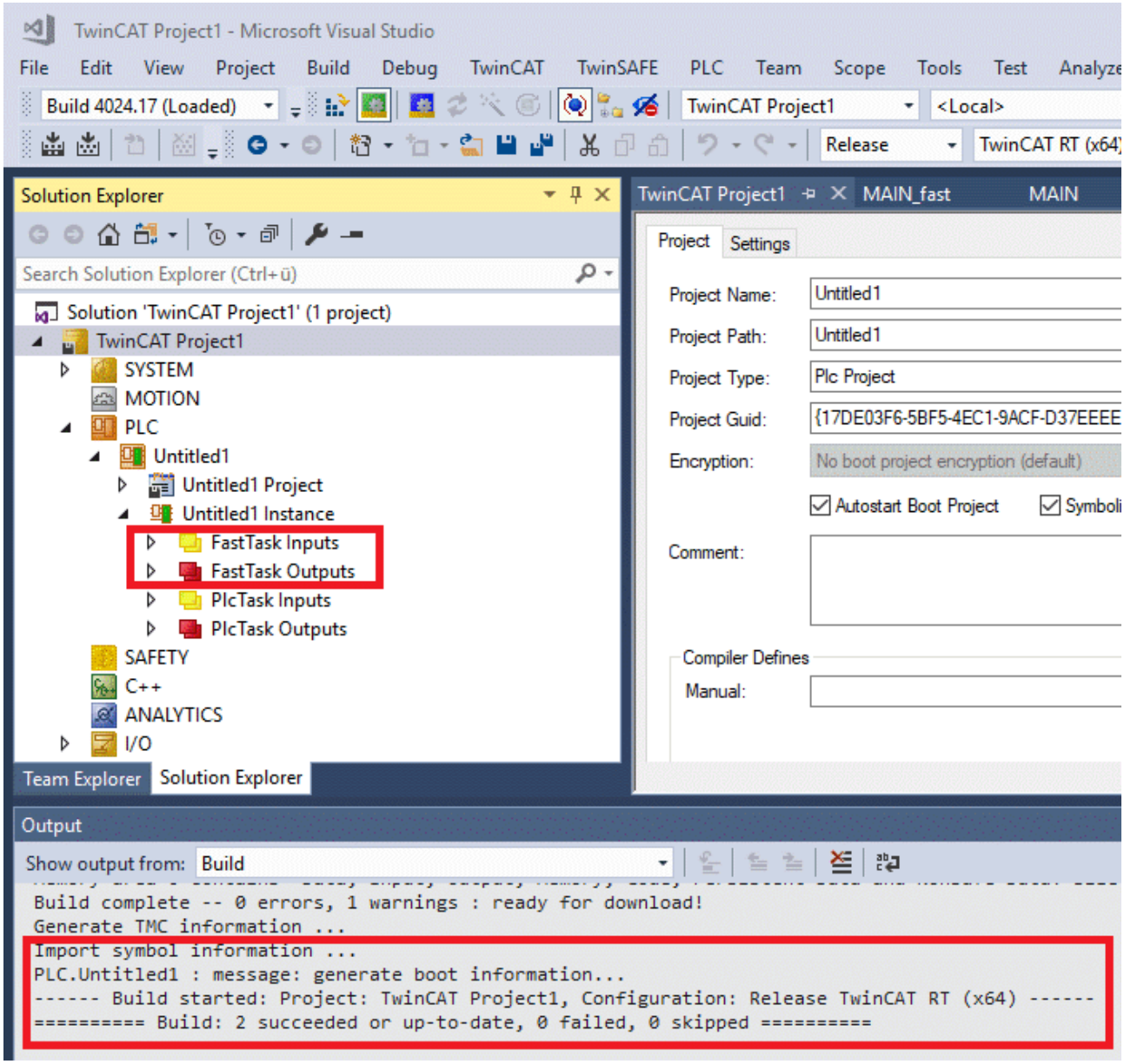
Programmverknüpfungen erstellen

Die Verknüpfungsvariablen zum Prozessabbild der seriellen Klemme befinden sich im Template "FB_BA_WeatherStation_Thies". Allein durch die Implementation des Templates stehen sie jedoch nicht automatisch für das Verknüpfen zur Verfügung. Hierzu muss die Projektmappe einmal neu erstellt werden.

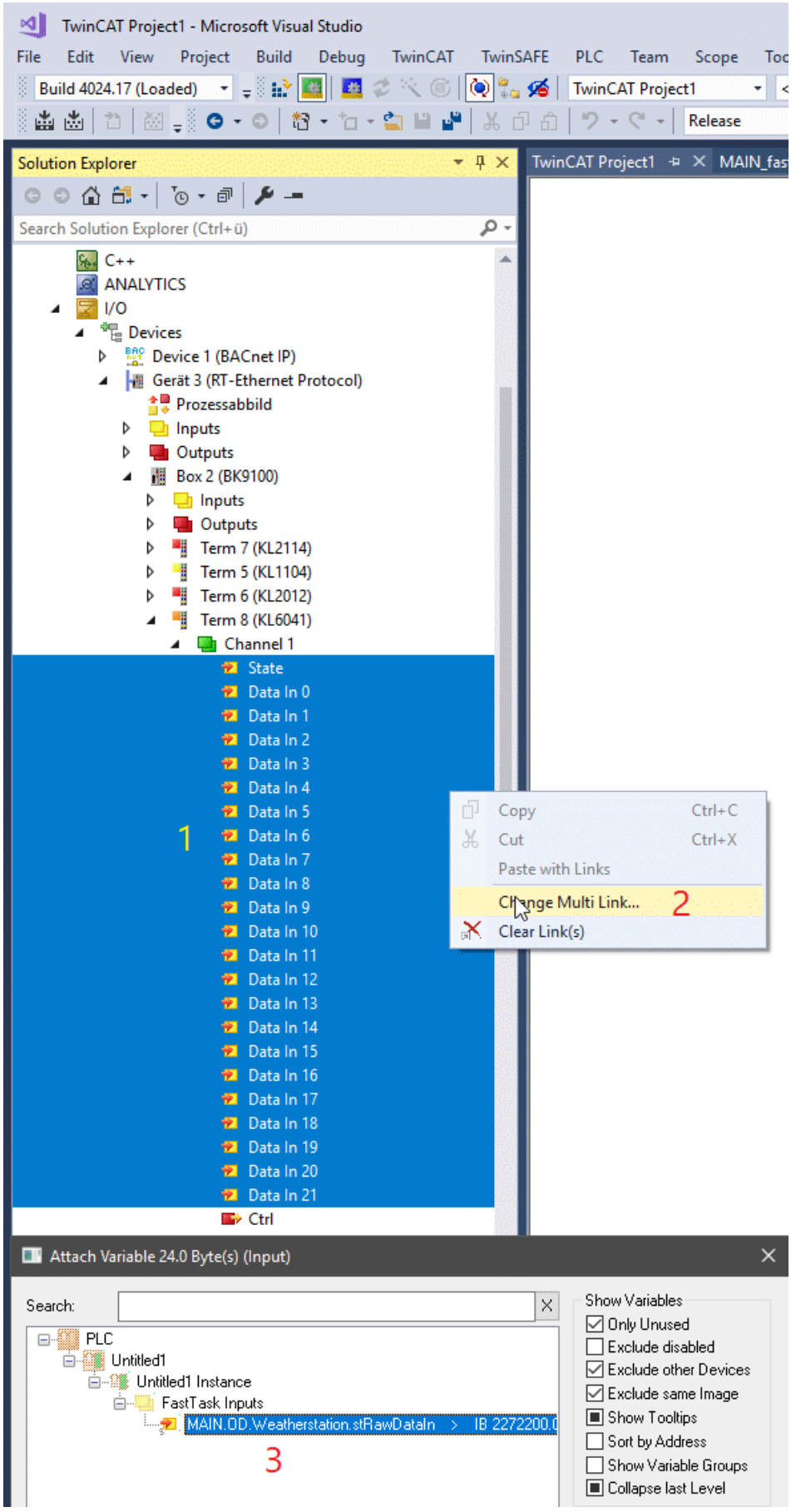
Wichtig ist an dieser Stelle, dass in dem Programm keine weiteren Übersetzungsfehler auftreten.



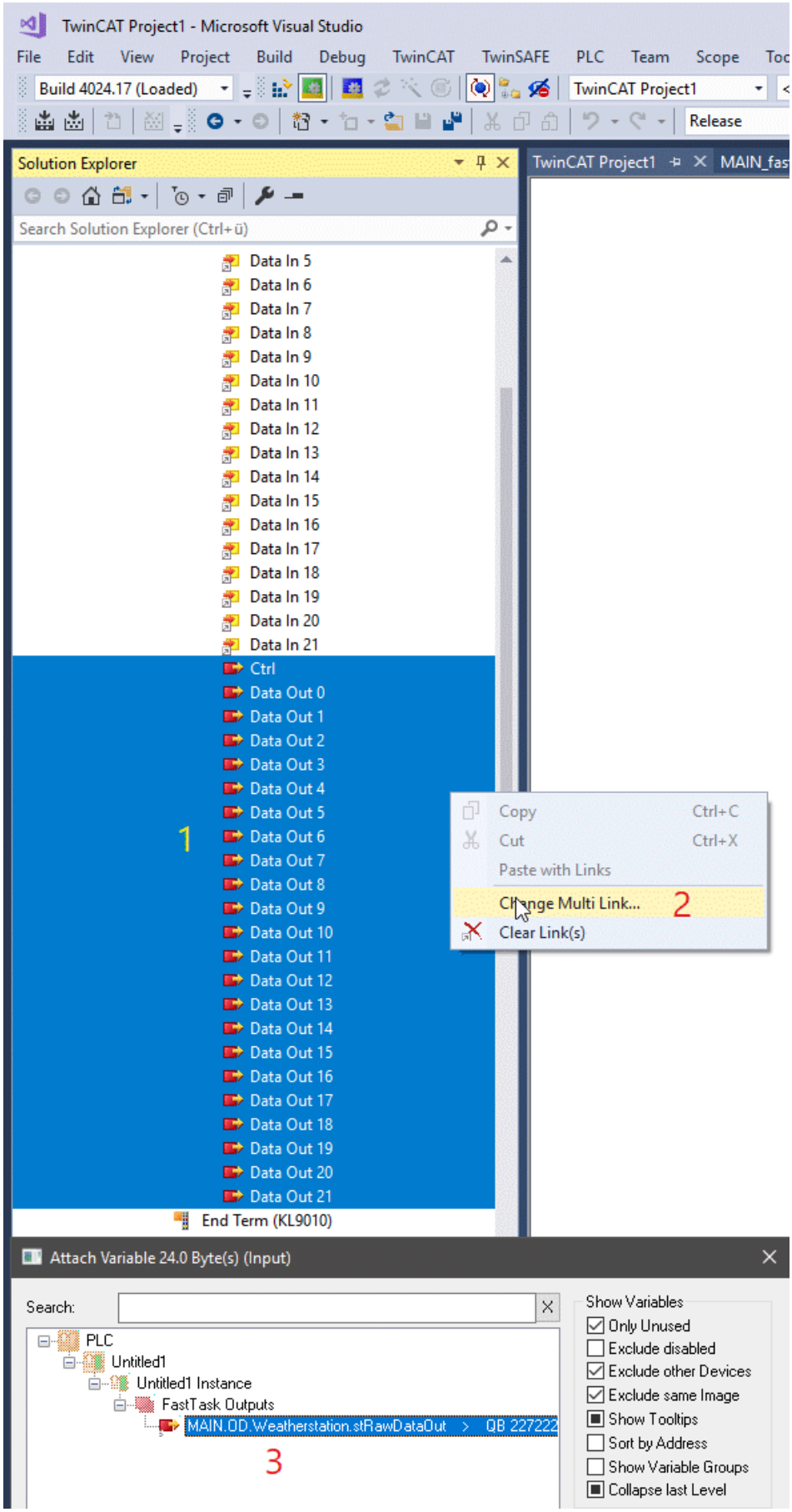
Unter "Erstellen" "Projektmappe neu erstellen" auswählen. Zu diesem Zeitpunkt hat nur die schon vorhandene SPS-Task ein Prozessabbild.



Nach einem fehlerfreien Erstellen stehen die Linkvariablen des Templates zur Verfügung und die Fast Task hält ebenfalls ein Prozessabbild-Bereich vor.

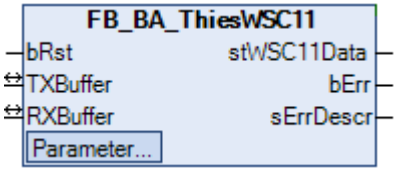


Zunächst werden alle Eingangsvariablen des Klemmen-Prozessabbildes markiert (Status anklicken und dann mit gehaltener Shift-Taste die Pfeil-runter-Taste betätigen) (1), danach Rechtsklick und "Multi-Verknüpfung" wählen (2). Mit "*stRawDataIn*" des Templates verknüpfen.



Das Verknüpfen der Ausgangsdaten erfolgt analog.

6.1.4.2.2.9.1.2 FB_BA_ThiesWSC11



Baustein zum zyklischen Auslesen und Aufbereiten der seriellen Daten aus einer Thies-WSC11-Wetterstation.

Dieser Baustein wandelt die seriellen Daten in eine definierte Struktur um.

Syntax

```
VAR_INPUT
  bRst                : BOOL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nUpdateTime         : UDINT;
  nConfigTimeout      : UDINT;
  nMaxCyclesOldData   : UDINT;
  nWndDatAvrgIntVal   : UDINT;
  bWndSpdLEDOn        : BOOL;
  bTwiLgtCalcAvrg     : BOOL;
  nWndDirNorthOffs    : UDINT;
  nStHgtAMSL          : UDINT;
  nRainOffDly         : UDINT;
  nDistMsgDly         : UDINT;
END_VAR
VAR_OUTPUT
  stWSC11Data         : ST_BA_WSC11Data;
  bErr                : BOOL;
  sErrDescr           : T_MaxString;
END_VAR
VAR_IN_OUT
  TXBuffer            : ComBuffer;
  RXBuffer            : ComBuffer;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bRst	BOOL	Eine steigende Flanke an diesem Eingang löst einen Software-Reset innerhalb der Wetterstation aus.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nUpdateTime	UDINT	Abrufintervall der Daten [500…60000 ms].
nConfigTimeout	UDINT	Timeout für die Konfigurationsroutine [s].
nMaxCyclesOldData	UDINT	Maximale Anzahl von Lesezyklen mit identischen Daten: darüber hinaus wird dies als Fehler angesehen - u.U. hervorgerufen durch Drahtbruch.
nWndDatAvrgIntVal	UDINT	Mittelungsintervall der Windrichtung und der Windgeschwindigkeit [1…10 min], 0 = Aus.
bWndSpdLEDOn	BOOL	Ein TRUE zeigt über die blaue LED an der Wetterstation das Ereignis „Wind“ an.
bTwiLgtCalcAvrg	BOOL	Auswahl der Werteberechnung für die Dämmerung: FALSE: die Summe der 4 Helligkeitssensoren wird genommen. TRUE: der Mittelwert der 4 Helligkeitssensoren wird genommen.
nWndDirNorthOffs	UDINT	Offset für die Windrichtung [0…360°]. Damit kann die Nordrichtung korrigiert werden.
nStHgtAMSL	UDINT	Stationshöhe über dem Meeresspiegel [0…3000 m].
nRainOffDly	UDINT	Abfallverzögerung der Regenerkennung [0…3600 s].
nDistMsgDly	UDINT	Verzögerung der Fehlermeldungen [0…60 s].

Ausgänge

Name	Typ	Beschreibung
stWSC11Data	ST_BA_WSC11Data [▶ 719]	Ausgabe der Daten
bErr	BOOL	Fehler
sErrDescr	T_MaxString	Textuelle Fehlerbeschreibung

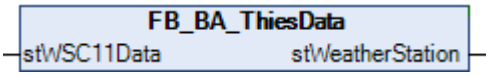
Ein-/Ausgänge

Name	Typ	Beschreibung
TXBuffer / RXBuffer	ComBuffer	Serieller Datenaustausch mit dem Lesebaustein SerialLineControl der schnellen Task.

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.2.2.9.1.3 FB_BA_ThiesData



In diesem Baustein werden Daten der Thies Wetterstation umgerechnet, so dass sie sich mit BACnet-Objekten darstellen lassen.

Das betrifft nur die Helligkeitswerte der vier Himmelsrichtungen: Während die Wetterstation diese in Kilolux ausgibt, sieht BACnet zur Darstellung nur die Einheit Lux vor, daher sind die Werte der Wetterstation mit 1000 zu multiplizieren.

Syntax

```
FUNCTION_BLOCK FB_BA_ThiesData
VAR_INPUT
  stWSC11Data         : ST_BA_WSC11Data;
END_VAR
VAR_OUTPUT
  stWeatherStation    : ST_BA_WeatherStation;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stWSC11Data	ST_BA_WSC11Data [▶ 719]	Daten aus der Thies-Wetterstation.

Ausgänge

Name	Typ	Beschreibung
stWeatherStation	ST_BA_WeatherStation [▶ 718]	Umgerechnete Daten

Voraussetzungen

Entwicklungsumgebung	Erforderliche Function
TwinCAT ab v3.1.4024.35	TF8040 | TwinCAT Building Automation ab V5.0.0.0

6.1.4.3 GVLs

6.1.4.3.1 Site

Die Site GVL ist eine Liste globaler Variablen, welche systemweit von allen Automationsstationen eines GA-Netzwerkes benötigt werden. Die Daten dienen zur Steuerung und Regelung der Anlagen- und Raumautomation.

Die Verteilung der Daten innerhalb des GA-Netzwerkes ist mittels der Templates FB_BA_AdsComClient und FB_BA_AdsComServer organisiert.

Unabhängig davon, ob es sich um den Server oder einen Client der Daten innerhalb des GA-Netzwerkes handelt, lesen bzw. schreiben alle Templates, welche Daten erzeugen bzw. lesen immer in die GVL-Site. Es befindet sich also in jeder Automationsstation die gleiche GVL-Site.

In den Clients z. B. eines Etagencontrollers für die Raumautomation, werden die Daten innerhalb des Templates FB_BA_AdsComClient über ADS vom Server gelesen und in die GVL kopiert. Alle Templates der Raumautomation greifen dann auf diese Daten der GVL-Site zu.

In dem IPC, welcher die Daten innerhalb des GA-Netzwerkes bereitstellt, wird das Template FB_BA_AdsComServer aufgerufen.

Die Systemtopologie der TF8040-Templates sieht vor, dass diese Daten von einem IPC innerhalb des Gebäudes erzeugt und mit den Funktionsbausteinen FB_BA_RawPublisher [▶ 143] und FB_BA_RawSubscriber [▶ 162] verteilt werden.

Illustration

```
VAR_GLOBAL
  Self_NetId                    : T_BA_MedString := '127.0.0.1.1.1';
  ACE01_NetId                   : T_BA_MedString := Self_NetId;
  ACE02_NetId                   : T_BA_MedString := Self_NetId;
  ACE03_NetId                   : T_BA_MedString := Self_NetId;
  ACE04_NetId                   : T_BA_MedString := Self_NetId;
  ACE05_NetId                   : T_BA_MedString := Self_NetId;

  GeneralSettings_Subject       : T_BA_MedString := 'GeneralSettings';
  GeneralSettings_NetId         : T_BA_MedString := Self_NetId;
```

```
  WeatherStation_Subject          : T_BA_MedString := 'WeatherStation';
  WeatherStation_NetId            : T_BA_MedString := Self_NetId;

  FacadeNorth_NetId               : T_BA_MedString := ACE01_NetId;
  FacadeNorth_Subject             : T_BA_MedString := 'FcdNorth.FacadeSunBlind';
  FacadeEast_NetId                : T_BA_MedString := ACE01_NetId;
  FacadeEast_Subject              : T_BA_MedString := 'FcdEast.FacadeSunBlind';
  FacadeSouth_NetId               : T_BA_MedString := ACE01_NetId;
  FacadeSouth_Subject             : T_BA_MedString := 'FcdSouth.FacadeSunBlind';
  FacadeWest_NetId                : T_BA_MedString := ACE01_NetId;
  FacadeWest_Subject              : T_BA_MedString := 'FcdWest.FacadeSunBlind';

  Building_NetId                  : T_BA_MedString := ACE01_NetId;
  BuildingAlarms_Subject          : T_BA_MedString := 'BuildingAlarms';
  BuildingMode_Subject            : T_BA_MedString := 'BuildingMode';
  BuildingEnergyLevel_Subject     : T_BA_MedString := 'BuildingEnergyLevel';
  BuildingSpRmT_Subject           : T_BA_MedString := 'BuildingSpRmT';
  BuildingSunProtection_Subject   : T_BA_MedString := 'BuildingSunProtection';

  stGeneralSettings               : ST_BA_GeneralSettings; // published by FB_BA_Settings
  bGeneralSettings_Error          : BOOL;
  eBuildingEnergyLevel            : E_BA_EnergyLvlEx; // published by FB_BA_BuildingEnergyLevel
  bBuildingEnergyLevel_Error      : BOOL;
  eBuildingMode                   : E_BA_BuildingMode; // published by FB_BA_BuildingMode
  bBuildingMode_Error             : BOOL;
  stBuildingSunBlind              : ST_BA_BuildingSunBlind; // published by FB_BA_BuildingSunprotecti
on
  bBuildingSunBlind_Error         : BOOL;
  stBuildingAlarms                : ST_BA_BuildingAlarms; // published by FB_BA_BuildingAlarms
  bBuildingAlarms_Error           : BOOL;
  stWeatherStation                : ST_BA_WeatherStation; // published by FB_BA_WeatherStation_xxx
  bWeatherStation_Error           : BOOL;
  stFacadeNorthSunBlind           : ST_BA_Facade; // published by FB_BA_Facade
  bFacadeNorthSunBlind_Error      : BOOL;
  stFacadeEastSunBlind            : ST_BA_Facade; // published by FB_BA_Facade
  bFacadeEastSunBlind_Error       : BOOL;
  stFacadeSouthSunBlind           : ST_BA_Facade; // published by FB_BA_Facade
  bFacadeSouthSunBlind_Error      : BOOL;
  stFacadeWestSunBlind            : ST_BA_Facade; // published by FB_BA_Facade
  bFacadeWestSunBlind_Error       : BOOL;
  stBuildingSpRmT                 : ST_BA_SpRmT; // Published by FB_BA_BuildingSpRmT
  bBuildingSpRmT_Error            : BOOL;
END_VAR
```

6.1.4.3.2 SysTime

Die globale Variablenliste beinhaltet die lokale NT-Systemzeit des TwinCAT-Systems.

Die Variablen dürfen pro Steuerung nur einmal beschrieben werden.

```
{attribute 'qualified_only'}
VAR_GLOBAL
  stSystemTime       : TIMESTRUCT;
  stSystemTimeUTC    : TIMESTRUCT;
  dtSystemTime       : DT;
  dtSystemTimeUTC    : DT;
END_VAR
```

6.1.4.3.3 EventClassesID

EC_ID

Die globale Variablenliste gibt die Ereignisklassennummer für die Meldungsklassen Objekte und den alarmfähigen Objekten vor.

Illustration

```
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
  NC10    : UDINT := 10;
  NC20    : UDINT := 20;
  NC30    : UDINT := 30;
  NC40    : UDINT := 40;
  NC50    : UDINT := 50;
```

```
  NC51    : UDINT := 51;
  NC60    : UDINT := 60;
  NC70    : UDINT := 70;
  NC80    : UDINT := 80;
END_VAR
```

Name	Typ	Beschreibung
NC10	UDINT	Gefahr für Leben
NC20	UDINT	Sicherheitsmeldung
NC30	UDINT	Alarmmeldung
NC40	UDINT	Störungsmeldung
NC50	UDINT	Wartungsmeldung
NC51	UDINT	Wartungsmeldung
NC60	UDINT	Systemmeldung
NC70	UDINT	Handeingriff
NC80	UDINT	Sonstige Meldungen

6.1.4.3.4 BA2_Param

Die globale Variablenliste beinhaltet allgemeine Parameter, um Objekte zu initialisieren.

Illustration

```
{attribute 'qualified_only'}
{attribute 'strict'}
VAR_GLOBAL CONSTANT
  fAP                                : REAL := 1013.25;
  nDefTimeDelay_ToAbnormal           : UDINT := 1;
  nDefTimeDelay_ToNormal             : UDINT := 1;
  fDefCOVIncrement                   : REAL := 0.1;
  fDefLimitDeadband                  : REAL := 0.0;
  nLoop_DefOpMode                    : E_BA_PIDMode := E_BA_PIDMode.eP1ID;
  stTrend_DefStartTime               : ST_BA_DateTime := ();
  stTrend_DefStopTime                : ST_BA_DateTime := ();
  nTrend_BufferSize                  : UDINT:= 500;
  bTrend_DefLogEnable                : BOOL := FALSE;
  bTrend_DefStopOnFull               : BOOL := FALSE;
  nTrend_DefLogInterval              : UDINT := 90;
  nTrend_DefNotificationThreshold    : UDINT := 50;
  eTrend_DefLoggingType              : E_BA_LoggingType := E_BA_LoggingType.ePolled;
END_VAR
VAR_GLOBAL
  bBlink                             : BOOL;
  bDlyStartPLC                       : BOOL := TRUE;
END_VAR
```

VAR_GLOBAL CONSTANT

Name	Typ	Beschreibung
fAP	REAL	Globale Konstante Hydrostatischer Luftdruck. Der mittlere Luftdruck der Erdatmosphäre beträgt auf Meereshöhe 1013,25 hPa.
nDefTimeDelay_ToAbnormal	UDINT	Standardwert für den Parameter Zeitverzögerung von Übergängen in nicht-normale Zustände.
nDefTimeDelay_ToNormal	UDINT	Standardwert für den Parameter Zeitverzögerung von Übergängen in normale Zustände.
fDefCOVIncrement	REAL	Standardwert für den Parameter COV-Inkrement.
fDefLimitDeadband	REAL	Standardwert für den Parameter Tot-Zone Grenzwert.
nLoop_DefOpMode	E_BA_PIDMode	Standardwert für den Parameter Betriebsmodus.
stTrend_DefStartTime	ST_BA_DateTime	Standardwert für den Parameter Startzeit.
stTrend_DefStopTime	ST_BA_DateTime	Standardwert für den Parameter Stoppzeit.
nTrend_BufferSize	UDINT	Anzahl der Einträge in einem Trend-Buffer.
bTrend_DefLogEnable	BOOL	Freigabe einer Trend-Aufzeichnung.
bTrend_DefStopOnFull	BOOL	FALSE, Ringspeicher; TRUE, Fester Speicher, der nichts mehr speichert, wenn der Buffer voll ist.
nTrend_DefLogInterval	UDINT	Standardwert für den Parameter Aufzeichnungsintervall.
nTrend_DefNotificationThreshold	UDINT	Standardwert für den Parameter Benachrichtigungsschwelle.
eTrend_DefLoggingType	E_BA_LoggingType	Standardwert für den Parameter Aufzeichnungstyp.

VAR_GLOBAL

Name	Typ	Beschreibung
bBlink	BOOL	Globaler Blinkimpuls. Der Blinkimpuls wird im Template FB_BA_ControlCabinetBasic [▶ 838] erzeugt.
bDlyStartPLC	BOOL	Die Variable wird verzögert nach einem Neustart der SPS auf TRUE gesetzt. Im Template FB_BA_Device [▶ 843] wird diese Verzögerung durch das Zeitglied *tonDlyStartPLC* umgesetzt.

6.1.4.3.5 Priority

Die globale Variablenliste ermittelt anhand der globalen Variablenliste *XBA_BACnetParam* aus der Bibliothek Tc3_XBA die BACnet Anlagenprioritäten.

Das Kapitel Kommandierung [▶ 35] enthält weitere Informationen zum Thema Prioritäten.

Syntax

```
VAR_GLOBAL
  nProgram             : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[2],16));
  nManualRemote        : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[3],16));
  nManualLocal         : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[4],16));
  nCritical            : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[5],16));
  nLifeSafety          : UDINT := TO_UDINT(LIMIT(1,XBA_BACnetParam.aPriority[6],16));
END_VAR
```

Name	Typ	Beschreibung
nProgram	UDINT	Priorität "Program", Programmsteuerung.
nManualRemote	UDINT	Priorität "Manual Remote", manuelle Übersteuerung aus der Ferne (MBE).
nManualLocal	UDINT	Priorität "Manual Local", Lokale manuelle Übersteuerung (LVB).
nCritical	UDINT	Priorität "Critical", Anlagensicherheit.
nLifeSafety	UDINT	Priorität "Life Safety", Personensicherheit.

6.2 HMI

6.2.1 TcHmiBa

TcHmiBa ist eine Erweiterung der TwinCAT HMI für Applikationen der Gebäudeautomation. Dem Integrator soll die Erstellung eines HMIs, für z.B. eine Heizungsanlage, Lüftungsanlage oder der gesamten HLK-Technik eines Gebäudes, so einfach wie möglich gemacht werden. Mit dieser Lösung ist es ebenfalls möglich die Raumautomation in einem Gebäude abzubilden und zu bedienen. TcHmiBa unterstützt einige Funktionen einer klassischen Management- und Bedienebene (MBE).



Beachten Sie, dass TcHmiBa **keine** MBE ist.

Inhalt

Die einfache Erstellung von HMIs für die Gebäudeautomation wird von TcHmiBa durch die Bereitstellung von verschiedenen Controls [▶ 1141] und Icons [▶ 1252] ermöglicht.

Generische HMI

Für Integratoren, die die Komplettlösung TF8040 verwenden, ergibt sich weiter die Möglichkeit, auf generische Funktionen [▶ 56] von TcHmiBa zurückzugreifen, die das Engineering weiter vereinfachen und beschleunigen.

Systemvoraussetzungen

- TF8040 (aktuelle Version)
- TE2000
- Google™ Chrome™ / Chromium™ (Support für weitere Browser folgt)

Open-Source-Software

In den Komponenten von TcHmiBa wird verschiedene Open-Source-Software eingesetzt.
Die Lizenztexte sind in den jeweiligen Ordnern zu finden:

- Entwicklungssystem:
  - TcHmiBaFramework: *§ProjektPfad§/Packages/Beckhoff.TwinCAT.HMI.BA.Framework/runtimes/native/Legal*
  - BaSiteExtension: *§ProjektPfad§/Packages/Beckhoff.TwinCAT.HMI.BA.BaSite/runtimes/any/native/Legal*
- Zielsystem:
  - TcHmiBaFramework: *C:\ProgramData\Beckhoff\TF2000 TwinCAT 3 HMI Server\service\TcHmiProject\www\Beckhoff.TwinCAT.HMI.BA.Framework\Legal*
  - BaSiteExtension: *C:\ProgramData\Beckhoff\TF2000 TwinCAT 3 HMI Server\service\TcHmiProject\BaSite\Legal*

6.2.1.1 Einleitung

TcHmiBa ist eine Erweiterung der TwinCAT HMI für Applikationen der Gebäudeautomation. Dem Integrator soll die Erstellung eines HMIs, für z. B. eine Heizungsanlage, Lüftungsanlage oder der gesamten HLK-Technik eines Gebäudes, so einfach wie möglich gemacht werden. Mit dieser Lösung ist es ebenfalls möglich die Raumautomation in einem Gebäude abzubilden und zu bedienen. *TcHmiBa* unterstützt einige Funktionen einer klassischen Management- und Bedienebene (MBE). Es ist aber zu beachten, dass *TcHmiBa* **keine** MBE ist.

Inhalt

Die einfache Erstellung von HMIs für Gebäudeautomationsprojekte wird von *TcHmiBa* durch die Bereitstellung von verschiedenen Controls [▶ 1141] und Icons [▶ 1252] ermöglicht.

Generische HMI

Für Integratoren, die die Komplettlösung TF8040 verwenden ergibt sich weiter die Möglichkeit auf generische Funktionen [▶ 56] von *TcHmiBa* zurückzugreifen, die das Engineering weiter vereinfachen und beschleunigen.

Open-Source-Software

In den Komponenten von *TcHmiBa* wird verschiedene Open-Source-Software eingesetzt.

Die Lizenztexte sind in den jeweiligen Ordnern zu finden:

- Entwicklungssystem:
  - TcHmiBaFramework: *§ProjektPfad§/Packages/Beckhoff.TwinCAT.HMI.BA.Framework/runtimes/native/Legal*
  - BaSite-Extension: *§ProjektPfad§/Packages/Beckhoff.TwinCAT.HMI.BA.BaSite/runtimes/any/native/Legal*
- Zielsystem:
  - TcHmiBaFramework: *C:\ProgramData\Beckhoff\TF2000 TwinCAT 3 HMI Server\service\TcHmiProject\www\Beckhoff.TwinCAT.HMI.BA.Framework\Legal*
  - BaSite-Extension: *C:\ProgramData\Beckhoff\TF2000 TwinCAT 3 HMI Server\service\TcHmiProject\BaSite\Legal*

6.2.1.1.1 Systemvoraussetzungen

Für das Entwicklungssystem:

- TwinCAT HMI Engineering

Für das Zielsystem

- TwinCAT HMI Server

Die enthaltenen NuGet-Pakete sind in zwei Bereiche aufgeteilt.

Client-Pakete

- Beckhoff.TwinCAT.HMI.BA.Icons
- Beckhoff.TwinCAT.HMI.BA.Framework
- Beckhoff.TwinCAT.HMI.BA.Controls

Für die Client-Pakete gelten folgende Systemvoraussetzungen:

- Chrome-Engine (z. B. Microsoft Edge®, Google Chrome®, Chromium®)



Bei der Verwendung von Webbrowsern mit einer anderen Engine ist die einwandfreie Funktion und Anzeige nicht garantiert.

Server-Pakete

- Beckhoff.TwinCAT.HMI.BA.BaSite

Für die Server-Pakete gelten folgende Systemvoraussetzungen:

- mindestens Windows 10 auf dem Entwicklungs- und Zielsystem
- .NET Desktop Runtime > v6.0

6.2.1.2 Controls

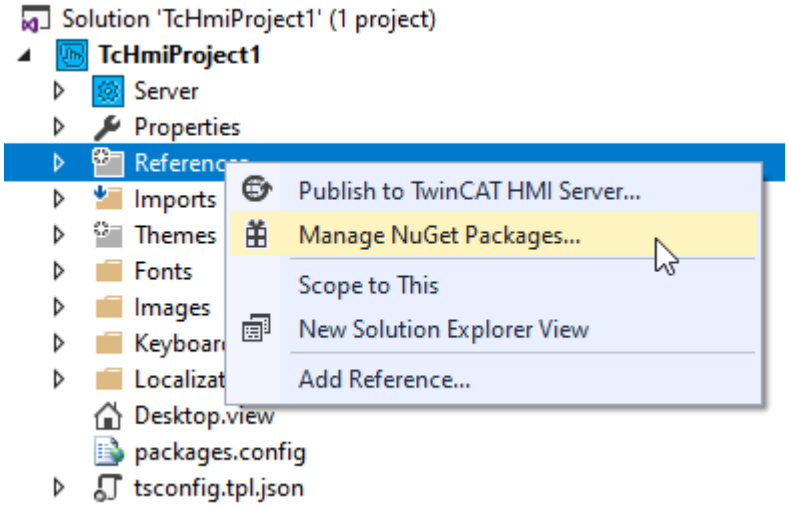
Auf den folgenden Seiten werden der Inhalt und die Funktionalitäten vom NuGet-Paket **Beckhoff.TwinCAT.HMI.BA.Controls** beschrieben. Dabei werden hauptsächlich die Attribute und Funktionen der Controls vorgestellt, die im Designer verwendet werden können.

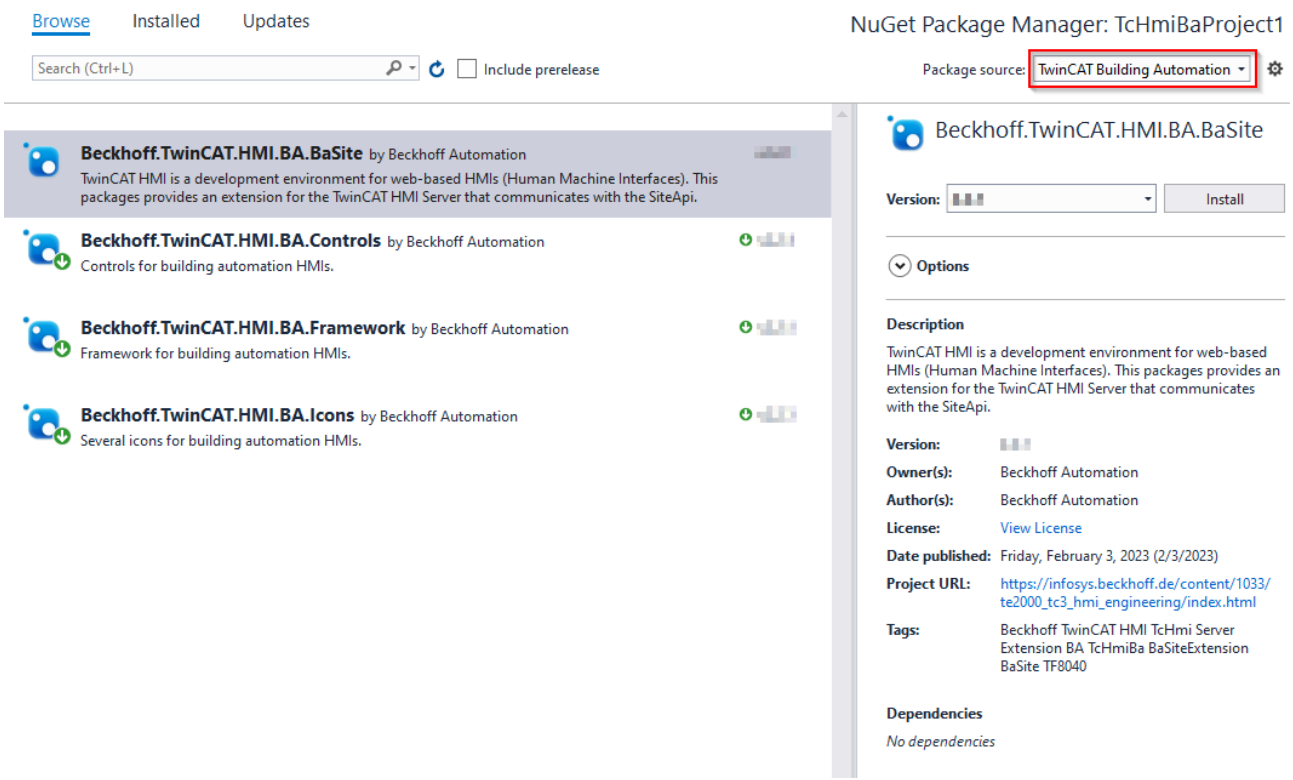
Ein TcHmiBa-Control ist als TcHmi-FrameworkControl implementiert und stellt verschiedene Funktionalitäten von TF8040 in grafischer Form zur Verfügung.

Die Erstellung ist zwar mit einem höheren Aufwand verbunden, liefert aber als Ergebnis ein performanteres und komfortableres Control für den Anwender.

Installation

Das NuGet-Paket **Beckhoff.TwinCAT.HMI.BA.Controls** muss installiert sein, um die Controls und Funktionen verwenden zu können.





Das Paket ist abhängig von folgenden Paketen und installiert diese daher mit:

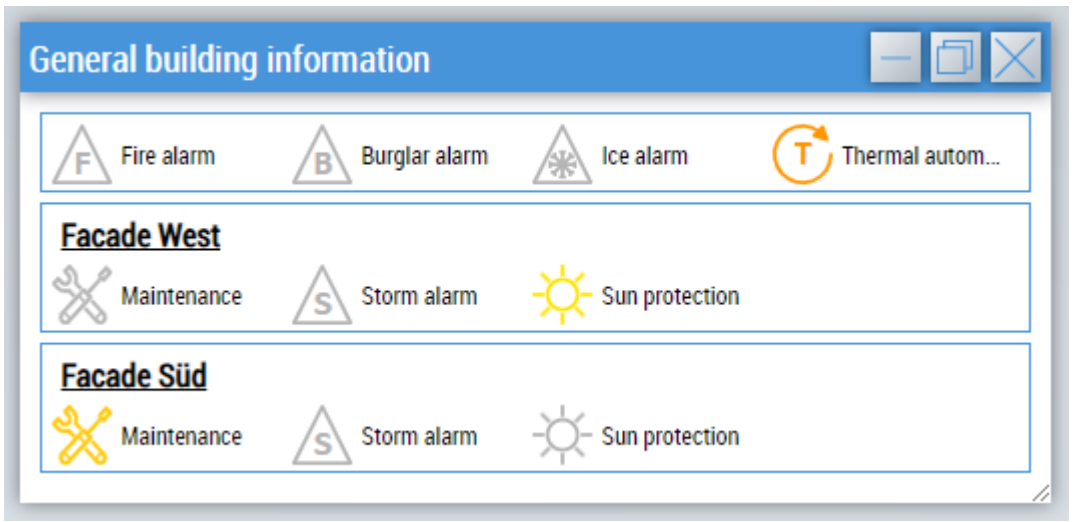
- Beckhoff.TwinCAT.HMI.Framework
- Beckhoff.TwinCAT.HMI.Controls
- Beckhoff.TwinCAT.HMI.BA.Framework [▶ 1263]
- Beckhoff.TwinCAT.HMI.BA.Icons [▶ 1252]

6.2.1.2.1 BuildingGeneral

6.2.1.2.1.1 BuildingInformation

Das **BuildingInformation** Control stellt sich zunächst wie ein normaler Button [▶ 1149] dar. Es ist dazu gedacht, verschiedene gebäude- bzw. fassadenspezifische Informationen anzuzeigen.





Verwendung

Einsatz auf jeder beliebigen Seite (z. B. in der Kopfzeile).

Features

Ist das Fenster nicht geöffnet, wird die Anzahl an relevanten Informationen in der Ecke des Buttons angezeigt. Das umfasst:

- Feueralarm
- Einbruchalarm
- Eisalarm
- Wartung
- Sturmalarm

Der Status des Sonnenschutzes oder der Thermoautomatik ist nur im geöffneten Fenster einzusehen.
Wird ein Alarm aktiv, so werden alle Controls benachrichtigt, die von Gebäudeinformationen abhängig sind.
Das betrifft:

- Sunblind [▶ 1230]
- Window [▶ 1235]

Attribute

Das Control erbt vom Button [▶ 1149] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Common

FireAlarm

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, wird der Feueralarm angezeigt.

BurglarAlarm

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, wird der Einbruchalarm angezeigt.

IceAlarm

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, wird der Eisalarm angezeigt.

ThermalAutomatic

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, wird die Thermoautomatik angezeigt.

Facades

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingInformation.Facades
```

Anlegen verschiedener Fassaden, die dann von entsprechenden Controls (s.o.) referenziert werden können.

6.2.1.2.1.2 Legend

Das **Legend** Control zeigt alle Icons auf der aktiven Webseite mit ihren Beschreibung an.

Verwendung

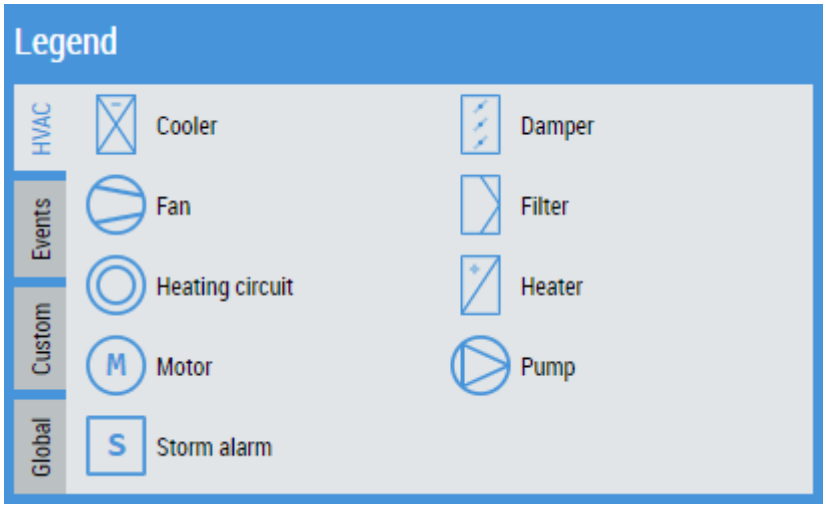
Einsatz auf jeder beliebigen Seite, auf der eine Erklärung der Icons angezeigt werden soll.

Features

Bietet eine statische und dynamische Anzeige der Icons und erlaubt das Hinzufügen zusätzlicher Icons.

Anzeige

Die Anzeige der Icons kann sowohl statisch auf der Webseite erfolgen als auch dynamisch (ereignisgesteuert) über den Aufruf der OpenLegendDialog [▶ 1277]-Funktion.



Zusätzliche Icons

Lokal

Über das Attribut IconDataCustom [▶ 1145] können der jeweiligen Instanz der Legend weitere Icons hinzugefügt werden. Beim Funktionsaufruf befindet sich das Attribut in der Parameterliste.

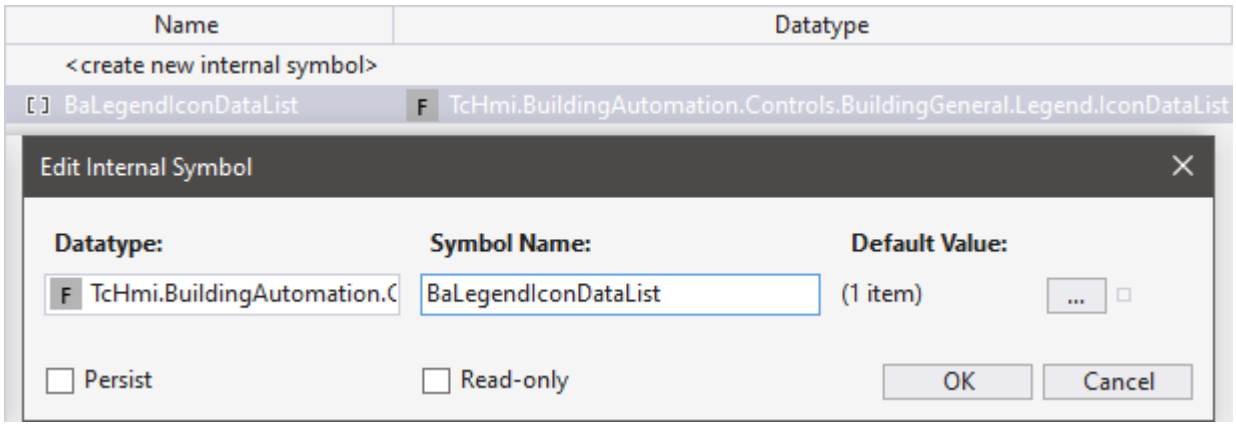
Global

Icons, die von jeder Instanz der Legend anzuzeigen sind, müssen über ein TwinCAT HMI Internes Symbol hinzugefügt werden.

Das Symbol muss den Namen BaLegendIconDataList haben und vom Typ

```
TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList
```

sein.



Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Common

ShowHeadline

`tchmi:general#/definitions/Boolean`

Legt die Sichtbarkeit des Titels fest.

EntryWidth

`tchmi:framework#/definitions/MeasurementValue`

Breite eines Eintrags.

EntryWidthUnit

`tchmi:framework#/definitions/MeasurementUnit`

Einheit der Breite eines Eintrags.

TabPosition

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position`

Position der Tabs.

IconDataSource

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataSource`

Auswahl anzuzeigender Einträge.

IconDataCustom

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList`

Liste mit zusätzlichen Einträgen.

6.2.1.2.1.3 WeatherStation

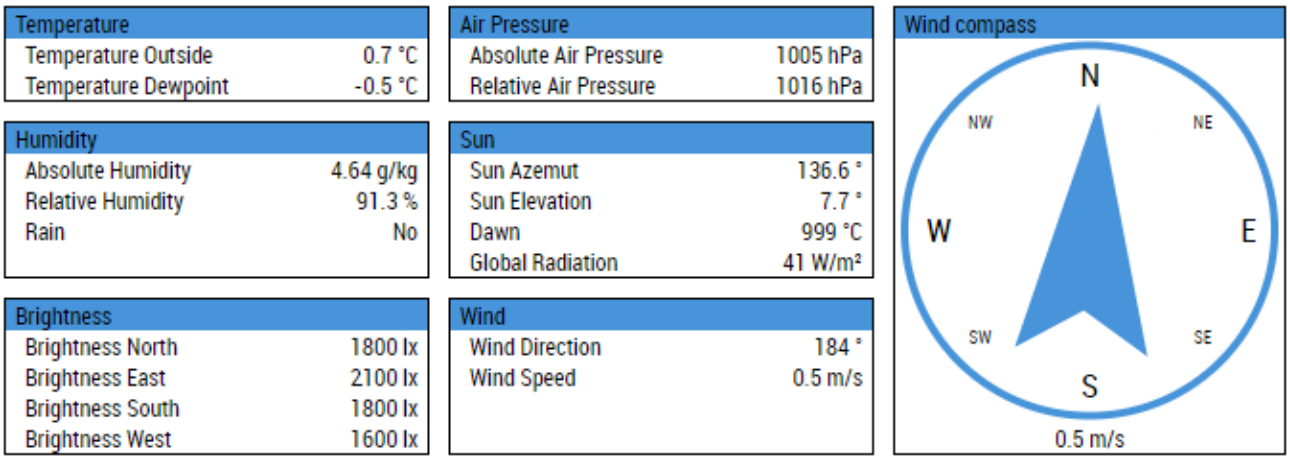
DasTemplate **WeatherStation** zeigt die Wetterdaten an.

Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **WeatherStation** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription unterstützt folgende BaObjects.



Unterlemente:

Symbolname	SPS-Template	Beschreibung
Dstb	FB_BA_SensorBinary [▶ 1107]	Störung
Rain	FB_BA_SensorBinary [▶ 1107]	Regen
WthT	FB_BA_SensorAnalog [▶ 1106]	Außentemperatur
DewPtT	FB_BA_SensorAnalog [▶ 1106]	Taupunkt
PrssAbs	FB_BA_SensorAnalog [▶ 1106]	Luftdruck (absolut)
PrssRel	FB_BA_SensorAnalog [▶ 1106]	Luftdruck (relativ)
HumAbs	FB_BA_SensorAnalog [▶ 1106]	Luftfeuchtigkeit (absolut)
HumRel	FB_BA_SensorAnalog [▶ 1106]	Luftfeuchtigkeit (relativ)
Brightness	FB_BA_MeasuredValueCardinal [▶ 1104]	Helligkeit
Dawn	FB_BA_SensorBinary [▶ 1107]	Dämmerung
GlobRadn	FB_BA_SensorAnalog [▶ 1106]	Sonneneinstrahlung
WndDir	FB_BA_SensorAnalog [▶ 1106]	Windrichtung
WndSpd	FB_BA_SensorAnalog [▶ 1106]	Windgeschwindigkeit
Latd	FB_BA_SensorAnalog [▶ 1106]	Breitengrad
Lngt	FB_BA_SensorAnalog [▶ 1106]	Längengrad
SunAzm	FB_BA_SensorAnalog [▶ 1106]	Sonnenrichtung
SunElv	FB_BA_SensorAnalog [▶ 1106]	Sonnenhöhe

Hierarchie:

- BaObject
  - Dstb
  - Rain
  - WthT
  - DewPtT
  - PrssAbs
  - PrssRel
  - HumAbs
  - HumRel
  - Brightness
  - Dawn
  - GlobRadn
  - WndDir
  - WndSpd
  - Latd
  - Lngt
  - SunAzm
  - SunElv

Entspricht den SPS-Templates:

- FB_BA_WeatherStation_Draft [▶ 866]
- FB_BA_WeatherStation_Thies [▶ 868]

Attribute

Das Control erbt vom BaseTemplate und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.BuildingGeneral.WeatherStation.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Common

ShowPosition

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Werte für die Position zu sehen sind.

ShowTemperature

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Werte für die Temperatur zu sehen sind.

ShowAirPressure

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Werte für den Luftdruck zu sehen sind.

ShowHumidity

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Werte für die Luftfeuchtigkeit zu sehen sind.

ShowSun

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Werte für die Sonne zu sehen sind.

ShowBrightness

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Werte für die Helligkeit zu sehen sind.

ShowWind

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Werte für den Wind zu sehen sind.

ShowWindCompass

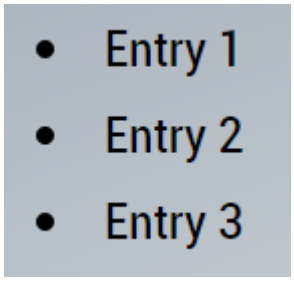
```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Windkompass zu sehen ist.

6.2.1.2.2 Common

6.2.1.2.2.1 BulletPointList

Die **BulletPointList** führt verschiedene Texte stichpunktartig auf.

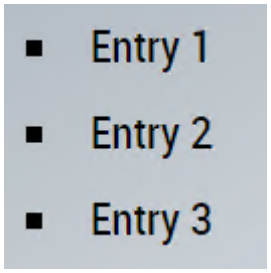


Verwendung

Einsatz auf jeder beliebigen Seite, auf der eine Stichpunktliste angezeigt werden soll.

Features

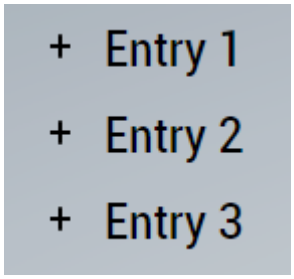
Änderung des Auflistungssymbols durch das Attribut *ListStyleType*.



Nummerierung der Einträge durch entsprechende Auswahl des *ListStyleType* Attributs.



Änderung des Auflistungssymbols durch das Attribut *ListStyleImage*.



Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Entries

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BulletPointList.Entries
```

Anzuzeigende Stichpunkte in der BulletPointList.

ListStyleType

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.BulletPointList.ListStyleTypes
```

Typ des Auflistungssymbols. Wird *String* gewählt, wird der eingegebene Text ohne Interpretation als Auflistungssymbol genutzt.

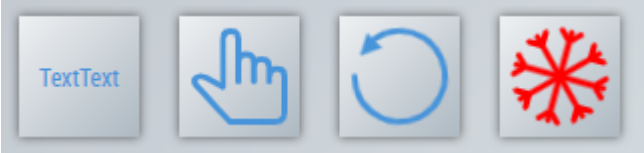
ListStyleImage

```
tchmi:framework#/definitions/Path
```

Pfad zum Bild des Auflistungssymbol.

6.2.1.2.2.2 Button

Der **Button** entspricht im Wesentlichen dem TcHmiButton. Er hat nur erweiterte Funktionalitäten für das Icon, da hier die Möglichkeiten des Icon-Pakets genutzt werden können.



Verwendung

Einsatz überall, wo ein Button mit erweiterten Icon-Funktionalitäten benötigt wird.

Features

Bietet erweiterte Funktionalitäten für Icons aus dem NuGet-Paket TcHmiBa.Icons [▶ 1252].

Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Icon

```
tchmi:framework#/definitions/Path
```

Pfad zum Icon.

IconWidth

```
tchmi:general#/definitions/Number
```

Breite des Icons.

IconWidthUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Einheit der Breite des Icons.

IconHeight

```
tchmi:general#/definitions/Number
```

Höhe des Icons.

IconHeightUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Einheit der Höhe des Icons.

IconHorizontalAlignment

```
tchmi:general#/definitions/HorizontalAlignment
```

Definition der horizontalen Ausrichtung des Icons innerhalb des Buttons.

IconVerticalAlignment

```
tchmi:general#/definitions/VerticalAlignment
```

Definition der vertikalen Ausrichtung des Icons innerhalb des Buttons.

IconRotation

```
tchmi:general#/definitions/Number
```

Legt fest, um wie viel Grad das Icon gedreht werden soll.

IconRotationSpeed

```
tchmi:general#/definitions/Number
```

Legt die Geschwindigkeit fest, mit der das Icon rotieren soll.

IconRotationDirection

```
tchmi:general#/definitions/TcHmi.BuildingAutomation.Controls.Direction
```

Legt die Richtung fest, in die das Icon rotiert, wenn das Attribut IconRotationSpeed definiert wurde. Der Standardwert ist rechts.

IconOverlays

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Icon.OverlayList
```

Definiert Überlagerungen des Icons.

Legend

ShowInLegend

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob das Icon in der Legende zu sehen ist.

Category

```
tchmi:general#/definitions/String
```

Legt die Kategorie des Icons in der Legende fest.

Description

```
tchmi:general#/definitions/String
```

Legt den Beschreibungstext des Icons in der Legende fest.

Events

Ereignis	Beschreibung
onButtonPressed	Wird ausgelöst, wenn der Button gedrückt wurde.
onButtonDoublePressed	Wird ausgelöst, wenn ein Doppelklick auf den Button durchgeführt wurde. Die Geschwindigkeit zur Erkennung eines Doppelklicks kann global [▶ 1290] eingestellt werden.

6.2.1.2.2.3 Calendar

Das Control **Calendar** dient zur Anzeige und Verwaltung von Ausnahmen eines Zeitschaltplans und Auswahl eines Datums.

Verwendung

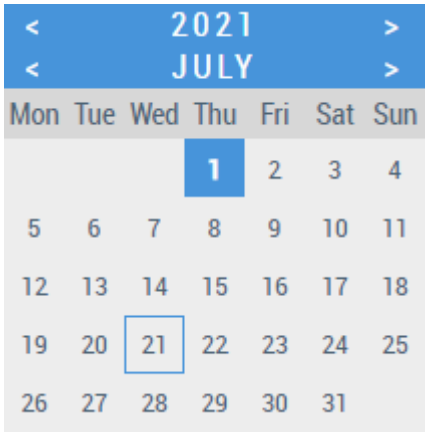
Einsatz auf jeder beliebigen Seite, auf der ein Datum ausgewählt werden soll.
Wird dem Attribut BaObject [▶ 1154] in der Darstellungsart *EventCalendar* ein Schedule-Objekt übergeben, lassen sich zusätzlich die Ausnahmen eines Zeitschaltplans anzeigen bzw. bearbeiten.

Features

Bietet zwei verschiedene Darstellungsarten, die Möglichkeit zum Verwalten von Ausnahmen und ein farbliches Hervorheben von zusammengehörigen Ausnahmen, wenn der Mauszeiger darübersteht.

DatePicker

Eine platzsparende Ansicht eines Kalenders. Liefert über ein Event das ausgewählte Datum zurück.

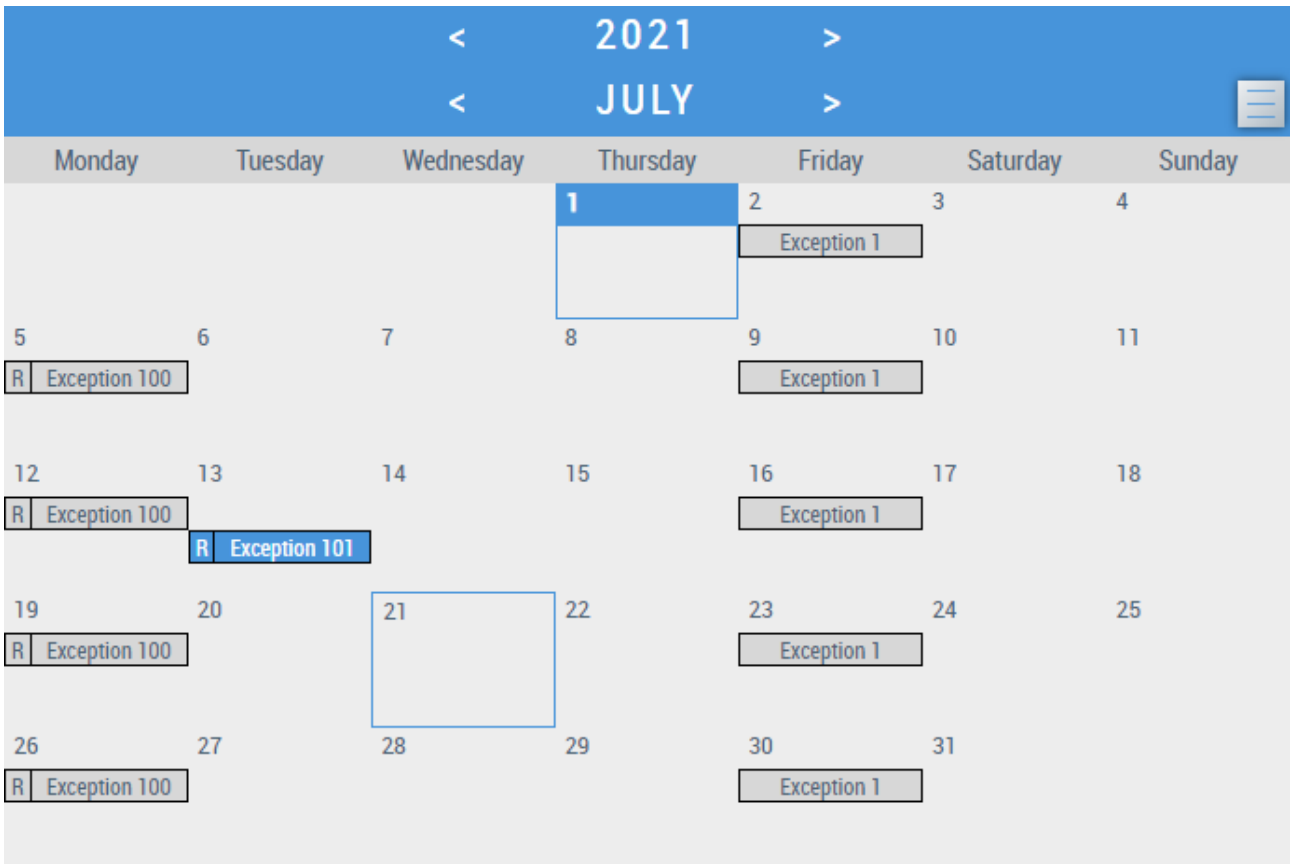


EventCalendar

Den EventCalendar gibt es in der Monatsansicht und in der Jahresansicht.

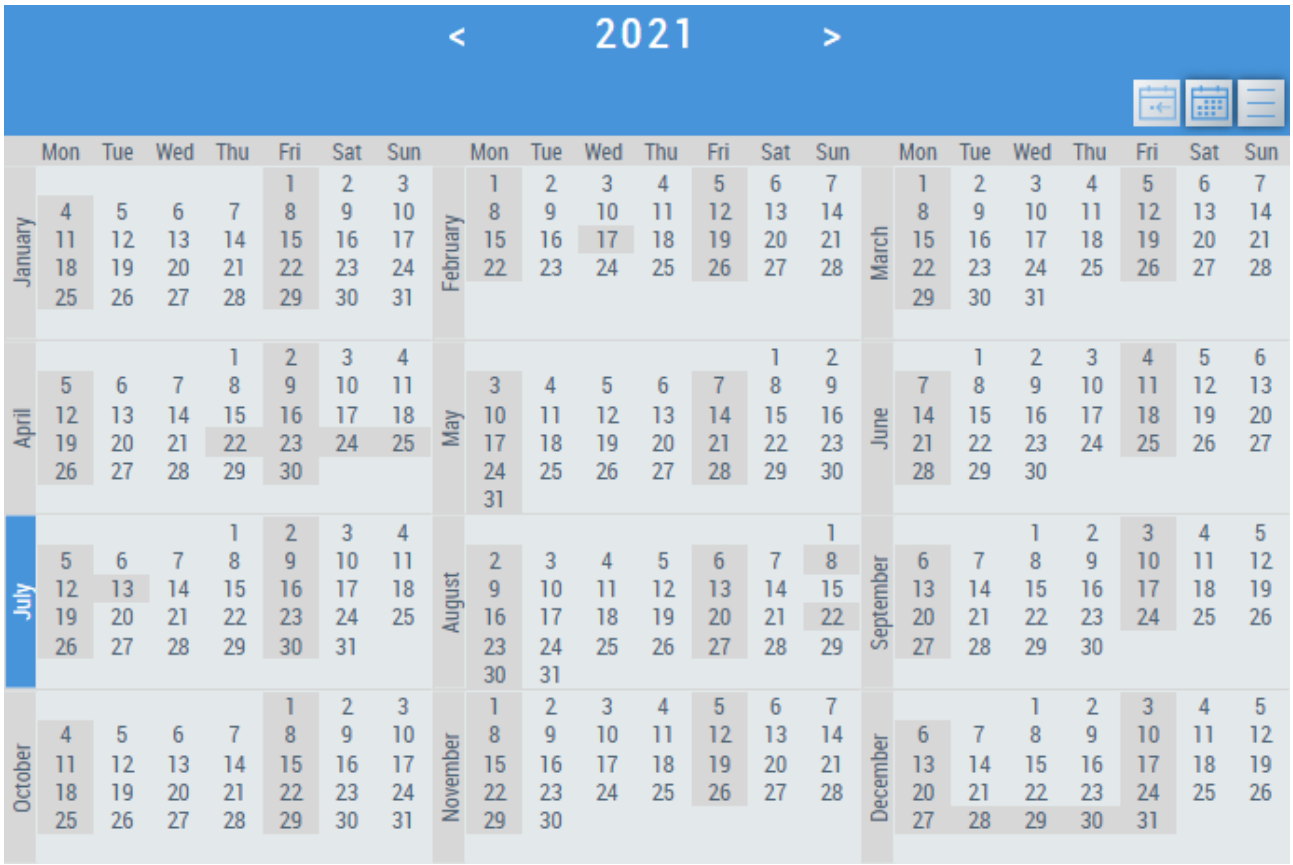
Monatsansicht

Eine ereignisorientierte Ansicht eines Kalenders. Sie liefert über ein Event das ausgewählte Datum zurück und bietet die Möglichkeit mit einem Klick auf eine Ausnahme diese zu verwalten.



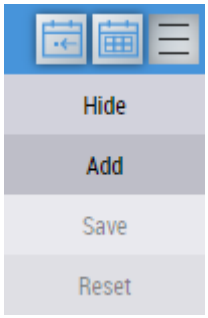
Jahresansicht

Eine Übersicht aller Tage des Jahres mit Markierung der Tage an denen mindestens eine Ausnahme aktiv ist. Die Anwahl eines Monats wechselt zur Monatsansicht.



Menü

In der oberen rechten Ecke befinden sich Buttons für Schnellzugriffe und weitere Aktionen.



Schnellzugriffe:

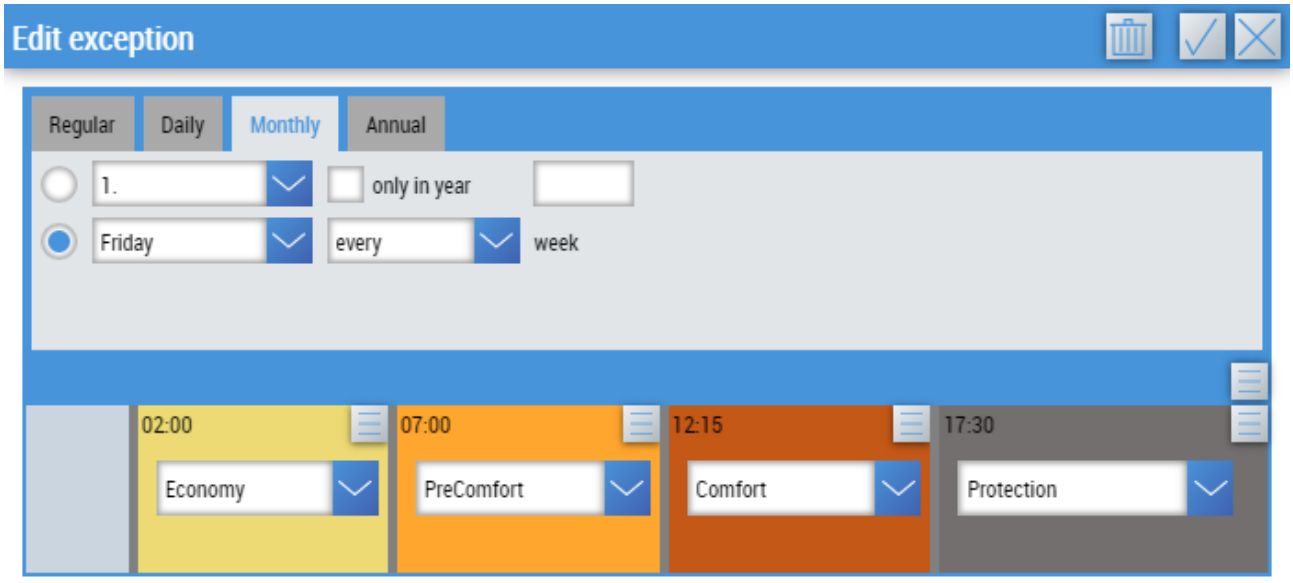
- Heute: Springt zum aktuellen Datum.
- Ansicht: Wechselt zwischen der Monats- und Jahresansicht.

Weitere Aktionen:

- Einblenden/Ausblenden: Blendet Ausnahmen ein oder aus.
- Hinzufügen: Fügt lokale Ausnahmen hinzu.
- Speichern: Schreibt alle Änderungen in die SPS.
- Zurücksetzen: Verwirft alle unbestätigten Änderungen.

Lokale Ausnahmen

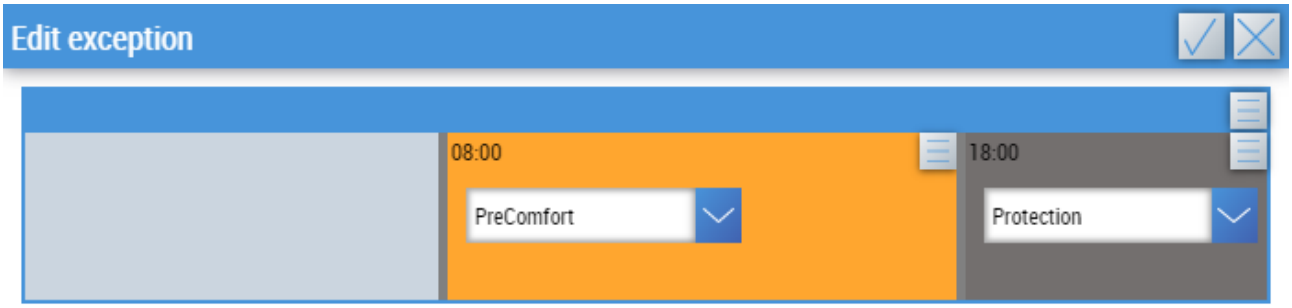
Einträge in der Sammlung *aException* eines Schedule-Objekts (z. B. FB_BA_SchedM [▶ 230]) werden als lokale Ausnahmen betrachtet.



Im oberen Bereich lassen sich das Datum bzw. die Wiederholungsart der lokalen Ausnahme einstellen. Darunter können die Zeitspannen mit den geltenden Werten festgelegt werden.
Die automatische Nummerierung der lokalen Ausnahmen startet bei 1.

Globale Ausnahmen (Kalenderreferenz)

Einträge in der Sammlung *aCalendar* eines Schedule-Objekts (z. B. FB_BA_SchedM [▶ 230]) werden als globale Ausnahmen betrachtet.



Bei globalen Ausnahmen können nur die Zeitspannen und geltende Werte festgelegt werden. Das Datum bzw. die Wiederholungsart ist im referenzierten Calendar-Objekt (z. B. FB_BA_Cal [▶ 222]) zu konfigurieren. Die automatische Nummerierung der globalen Ausnahmen startet bei 100.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

DisplayMode

Legt die Darstellungsart des Kalenders fest.

DisplayView

Legt die Ansicht des Kalenders in der Darstellungsart *EventCalendar* fest.

ShowMenu

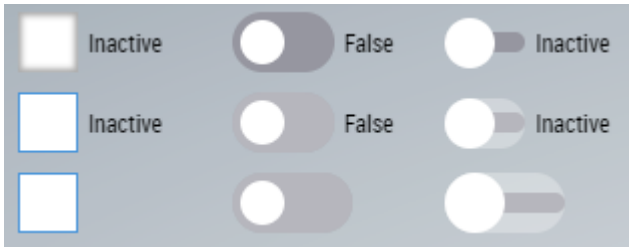
Legt die Sichtbarkeit des Menüs fest.

Events

Ereignis	Beschreibung
onDateChanged	Liefert das ausgewählte Datum zurück.

6.2.1.2.2.4 Checkbox

Die **Checkbox** zeigt bzw. bearbeitet binäre Werte.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der binäre Werte angezeigt bzw. bearbeitet werden sollen.

Besonderheiten

Der Aktiv- und Inaktivtext kann festgelegt werden (z. B. "Ein" / "Aus").

Die Darstellung kann über das Attribut Appearance [▶ 1155] angepasst werden (siehe Bild oben).

Möglichkeit ein BaObject [▶ 1155] zu verknüpfen, um nur ein einzelnes Binding erzeugen zu müssen. Über dieses Binding werden dann alle notwendigen Attribute verknüpft und Änderungen vom Wert automatisch in die SPS zurückgeschrieben.

Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

ActiveText

```
tchmi:general#/definitions/String
```

Legt den Text fest, der angezeigt wird, wenn *State* TRUE ist.

InactiveText

```
tchmi:general#/definitions/String
```

Legt den Text fest, der angezeigt wird, wenn *State* FALSE ist.

Appearance

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Checkbox.Appearance
```

Legt fest, wie sich die Checkbox oben darstellt.

Colors

CheckBackgroundColor

```
tchmi:general#/definitions/SolidColor
```

Hintergrundfarbe der Checkbox, wenn *State* TRUE ist.



Dieses Attribut hat keine Auswirkung, wenn *Appearance* auf *ToggleSlider* steht.



CheckmarkColor

```
tchmi:general#/definitions/SolidColor
```

Farbe des Hakens bzw. des Toggles.



BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.Checkbox.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

State

```
tchmi:general#/definitions/Boolean
```

Status der Checkbox.

StateFeedback

```
tchmi:general#/definitions/Boolean
```

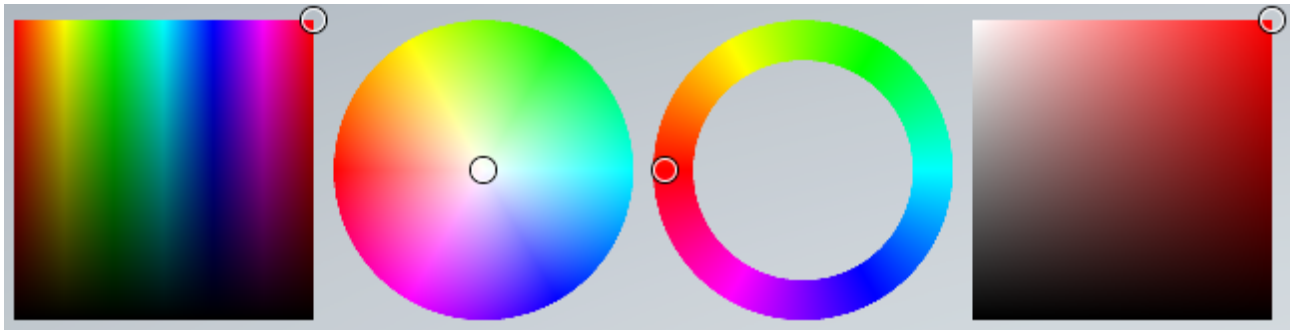
Feedback für den Status der Checkbox.

Events

Ereignis	Beschreibung
onStateChanged	Wird ausgelöst, wenn sich der Wert von State geändert hat.
onUserInteractionFinished	Wird ausgelöst, wenn die Benutzerinteraktion mit der Checkbox beendet wurde.

6.2.1.2.2.5 ColorPicker

Der **ColorPicker** kann dazu verwendet werden, aus verschiedenen Farbpaletten eine Farbe zu wählen.



Verwendung

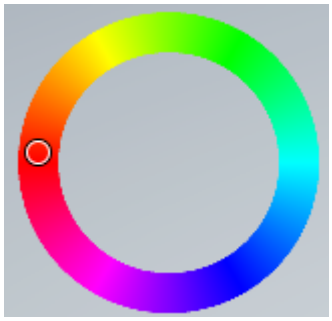
Einsatz auf jeder beliebigen Seite, auf der die Auswahl einer Farbe notwendig ist.

Features

Farbauswahl kann anhand verschiedener Farbräume geschehen.

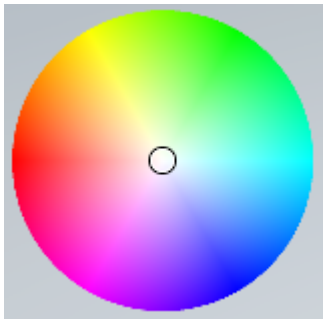
HSL Ring

Auswahl aller Farben ohne Schattierung.



HSV Circle

Auswahl aller Farben bis Weiß.



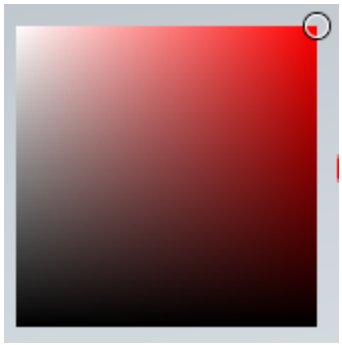
HSL Rect All color

Auswahl aller Farben bis Schwarz.



HSL Rect single

Auswahl aller Schattierungen einer Farbe. Zur Farbänderung, muss das Attribut BackgroundColor geändert werden.



Attribute

Das Control erbt vom TcHmiControl und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

ColorPlateType

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.ColorPlateType
```

Legt die verwendete Farbpalette fest.

BackgroundColor

```
tchmi:framework#/definitions/SolidColor
```

Legt die Hintergrundfarbe fest, wenn die Farbpalette HslRect1 ausgewählt wurde.

SelectedSolidColor (read-only)

```
tchmi:framework#/definitions/SolidColor
```

Ausgewählte Farbe.

SelectedRgbaColor (read-only)

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.RGBAColor
```

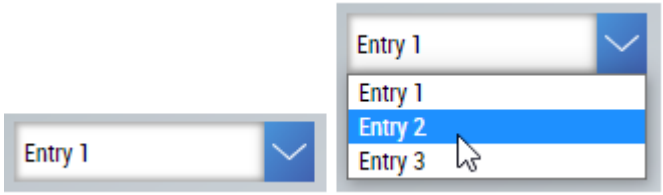
Ausgewählte Farbe im RGBA-Format.

Events

Ereignis	Beschreibung
onSelectedColorChanged	Wird ausgelöst, wenn sich die ausgewählte Farbe geändert hat.

6.2.1.2.2.6 Combobox

Die **Combobox** zeigt bzw. bearbeitet Multistate-Werte.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der Multistate-Werte angezeigt bzw. editiert werden sollen.

Features

Möglichkeit ein BaObject [▶ 1159] zu verknüpfen, um nur ein einzelnes Binding erzeugen zu müssen. Über dieses Binding werden dann alle notwendigen Attribute verknüpft und Änderungen vom Wert automatisch in die SPS zurückgeschrieben.

Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

Data

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Combobox.ComboboxItems
```

Daten für die Combobox.

Colors

ButtonColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe des Buttons, der die Dropdown-Liste öffnet.

ButtonArrowColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe des Pfeils im Button.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.Combobox.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

SelectedData (read-only)

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Combobox.ComboboxItem
```

Aktuell ausgewählte Daten.

SelectedValue

```
tchmi:general#/definitions/Number
```

Wert der aktuell ausgewählten Daten.

UseSelectedValueFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob auf das Attribut SelectedValueFeedback geachtet wird oder nicht.

SelectedValueFeedback

```
tchmi:general#/definitions/Number
```

Feedback für den gewählten Wert.

Events

onChanged

Wird ausgelöst, wenn sich der gewählte Wert geändert hat. Dies geschieht, wenn der Benutzer einen neuen Eintrag auswählt.

Events

Ereignis	Beschreibung
onSelectedValueChanged	Wird ausgelöst, wenn sich der gewählte Wert geändert hat.
onUserInteractionFinished	Wird ausgelöst, wenn die Benutzerinteraktion mit der Combobox beendet wurde.

6.2.1.2.2.7 DateTimeField

Das **DateTimeField** kann dazu verwendet werden, ein Datum und eine Uhrzeit auszuwählen oder anzuzeigen.

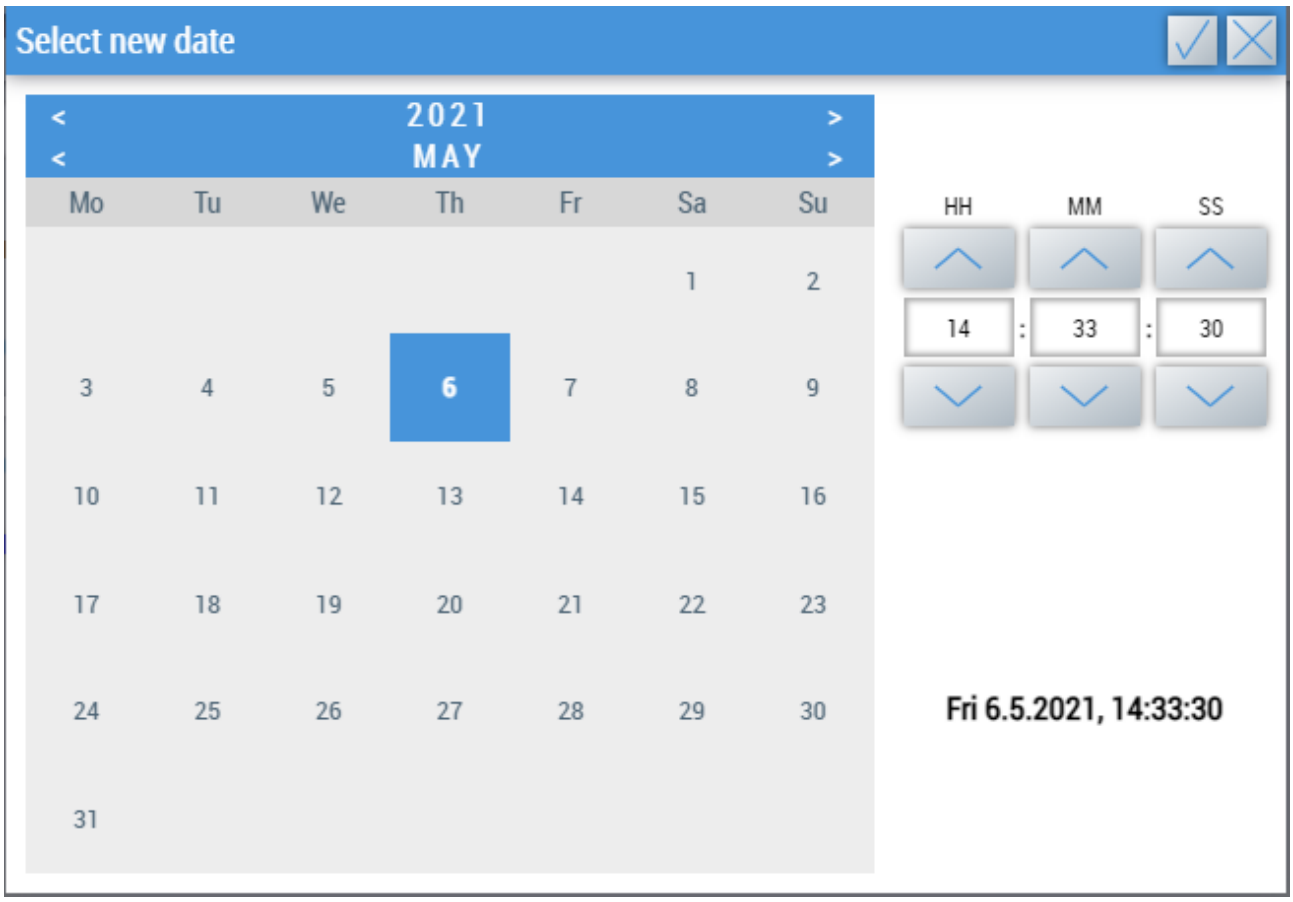


Verwendung

Einsatz auf jeder beliebigen Seite, auf der Datumswerte angezeigt bzw. bearbeitet werden sollen.

Features

Ist das Attribut ReadOnly [▶ 1213] FALSE, lässt sich der **DateTimePicker** über den Button öffnen, um ein neues Datum bzw. eine neue Uhrzeit auszuwählen.



Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Common

DateTime

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BaDateTime
```

Aktueller Wert von Zeit und Datum.

Events

Ereignis	Beschreibung
onDateChanged	Wird ausgelöst, wenn sich der Wert des Datums oder der Uhrzeit geändert hat.
onUserInteractionFinished	Wird ausgelöst, wenn die Benutzerinteraktion mit dem DateTimeField beendet wurde.

6.2.1.2.2.8 Icon

Das Icon entspricht im Wesentlichen dem TcHmiImage. Es hat nur erweiterte Funktionalitäten für das Icon, da hier die Möglichkeiten des Icon-Pakets [▶ 1254] genutzt werden können.

Verwendung

Einsatz überall, wo ein Icon benötigt wird.

Features

Bietet erweiterte Funktionalitäten für Icons aus dem NuGet-Paket TcHmiBaIcons [▶ 1252].

Attribute

Icon

```
tchmi:framework#/definitions/Path
```

Pfad zum Icon.

IconWidth

```
tchmi:general#/definitions/Number
```

Breite des Icons.

IconWidthUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Einheit der Breite des Icons.

IconHeight

```
tchmi:general#/definitions/Number
```

Höhe des Icons.

IconHeightUnit

```
tchmi:general#/definitions/MeasurementUnit
```

Einheit der Höhe des Icons.

IconHorizontalAlignment

```
tchmi:general#/definitions/HorizontalAlignment
```

Definition der horizontalen Ausrichtung des Icons innerhalb des Buttons.

IconVerticalAlignment

```
tchmi:general#/definitions/VerticalAlignment
```

Definition der vertikalen Ausrichtung des Icons innerhalb des Buttons.

IconRotation

```
tchmi:general#/definitions/Number
```

Legt fest, um wie viel Grad das Icon gedreht werden soll.

IconRotationSpeed

```
tchmi:general#/definitions/Number
```

Legt die Geschwindigkeit fest, mit der das Icon rotieren soll.

IconRotationDirection

```
tchmi:general#/definitions/TcHmi.BuildingAutomation.Controls.Direction
```

Legt die Richtung fest, in die das Icon rotiert, wenn das Attribut IconRotationSpeed definiert wurde. Der Standardwert ist rechts.

IconOverlays

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Icon.OverlayList
```

Definiert Überlagerungen des Icons.

Icon

ShowInLegend

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob das Icon in der Legende zu sehen ist.

Category

```
tchmi:general#/definitions/String
```

Legt die Kategorie des Icons in der Legende fest.

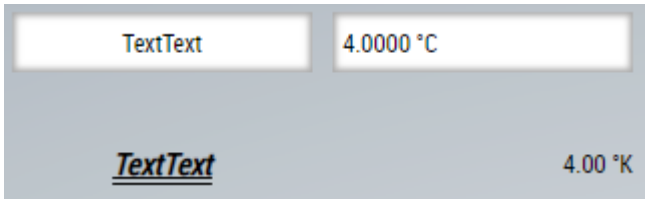
Description

```
tchmi:general#/definitions/String
```

Legt den Beschreibungstext des Icons in der Legende fest.

6.2.1.2.2.9 InputBox

Die **InputBox** dient zur Anzeige und Bearbeitung von numerischen oder textuellen Werten.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der numerische oder textuelle Werte angezeigt bzw. bearbeitet werden sollen.

Features

Numerische Eingabe

Ist der *DataType* gleich number, so wird die Benutzereingabe auf folgende Kriterien geprüft:

- rein numerische Eingabe (Buchstaben und Sonderzeichen sind nicht erlaubt)
- minimaler Wert (wenn *MinValue* gesetzt)
- maximaler Wert (wenn *MaxValue* gesetzt)

Ebenfalls können die Einheit und Anzahl der Nachkommastellen für einen numerischen Wert mit den Attributen *Unit* und *Digits* vorgegeben werden.

Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

DataType

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.InputBox.InputDataType
```

Datentyp der InputBox. Ist *auto* ausgewählt, wird der Standardwert oder die erste Eingabe analysiert und der Datentyp entsprechend gesetzt. Ist *number* ausgewählt enthält *Value* **nicht** die Einheit.

Text (read-only)

```
tchmi:general#/definitions/String
```

Der angezeigte Text.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.InputBox.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

Value

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Aktueller Wert. Abhängig vom gewählten **DataType** ist der Wert numerisch oder textuell.

ValueFeedback

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Feedback für den Wert.

UseValueFeedback

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann wird auf das Attribut *ValueFeedback* geachtet.

Number

MinValue

```
tchmi:general#/definitions/Number
```

Geringster zulässiger Eingabewert (wenn *DataType* gleich *number*).

MaxValue

```
tchmi:general#/definitions/Number
```

Größter zulässiger Eingabewert (wenn *DataType* gleich *number*).

Unit

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Legt die Einheit hinter *Value* fest (wenn *DataType* gleich *number*). Mögliche Werte sind:

- textuell (z.B. "°C")
- numerisch (Enumerationswert von E_BA_Unit)

Digits

```
tchmi:general#/definitions/Number
```

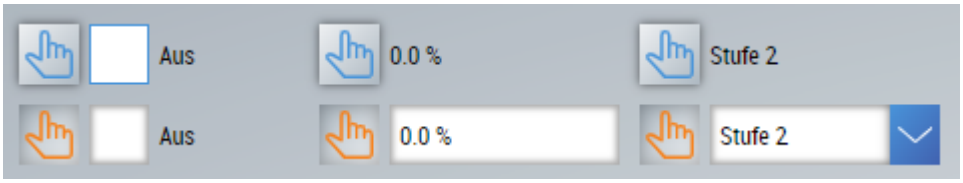
Anzahl der Nachkommastellen (wenn *DataType* gleich *number*).

Events

Ereignis	Beschreibung
onValueChanged	Wird ausgelöst, wenn sich der Wert geändert hat.
onUserInteractionFinished	Wird ausgelöst, wenn der Benutzer die Eingabe beendet. Das bedeutet: • Eingabetaste wird betätigt • InputBox verliert Fokus

6.2.1.2.2.10 ManualOverride

Der *ManualOverride* dient dem manuellen Überschreiben von einem binären, analogen oder mehrstufigen Wert.



Verwendung

Einsatz auf jeder beliebigen Seite, wo ein Wert manuell überschrieben werden soll.

Das Control wird in der ProjectNavigationTextual [▶ 1186] verwendet.

Features

Einfaches Überschreiben des aktuellen Wertes mit einem manuellen Wert. Ist die manuelle Übersteuerung aktiv, erfolgt eine visuelle Kennzeichnung.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.ManualOverride.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

ManualEnable

```
tchmi:general#/definitions/Boolean
```

Definiert, ob die manuelle Übersteuerung aktiv ist oder nicht.

ManualValue

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Der aktuelle manuelle Wert.

AutoValue

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Der aktuelle automatische Wert.

6.2.1.2.2.11 ManualOverrideBinary

Der *ManualOverrideBinary* dient dem manuellen Überschreiben eines binären Wertes.



Verwendung

Einsatz auf jeder beliebigen Seite, wo ein binärer Wert manuell überschrieben werden soll.

Features

Einfaches Überschreiben des aktuellen Wertes mit einem manuellen Wert. Ist die manuelle Übersteuerung aktiv, erfolgt eine visuelle Kennzeichnung.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.

Das Attribut ist nicht bei allen Controls anwendbar.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Common.ManualOverrideBinary.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

ManualEnable

```
tchmi:general#/definitions/Boolean
```

Definiert, ob die manuelle Übersteuerung aktiv ist oder nicht.

ManualValue

```
tchmi:general#/definitions/Boolean
```

Der aktuelle manuelle Wert.

AutoValue

```
tchmi:general#/definitions/Boolean
```

Der aktuelle automatische Wert.

6.2.1.2.2.12 Paginator

Der **Paginator** kann dazu verwendet werden, um zwischen verschiedenen Content-Seiten zu navigieren.

Verwendung

Einsatz auf jeder beliebigen Seite, auf der in einer Galerie zwischen verschiedenen Content-Seiten umgeschaltet werden soll.

Features

Geräte mit Touchscreen

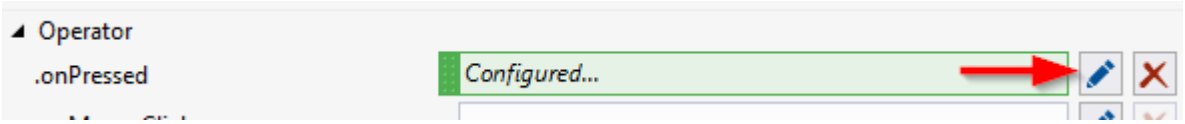
Wechsel zwischen den konfigurierten Content-Seiten mit einer Wischgeste nach links oder rechts.

Geräte ohne Touchscreen

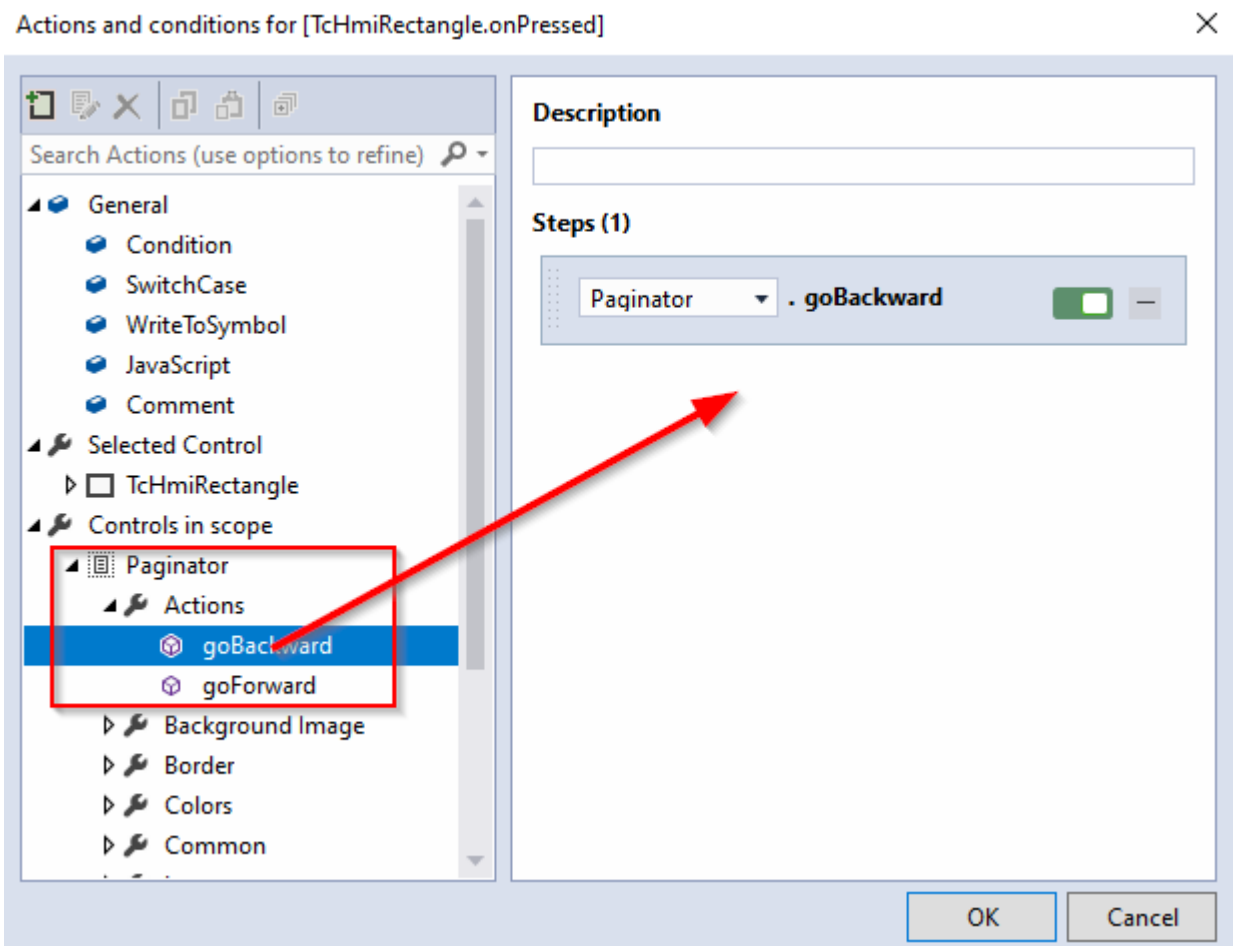
Die Navigation durch die Content-Seiten erfolgt über verschiedene Controls. Dazu werden die Funktionen *GoForward* und *GoBackward* von Events der Controls (z. B. OnPressed) aufgerufen.
Erläuterung am Beispiel zweier Rechtecke, die jeweils links und rechts vom *Paginator* positioniert werden.



Über das Eigenschaftenfenster des jeweiligen Rechteckes wird das **OnPressed** Event konfiguriert.



Danach wird die Funktion **GoForward / GoBackward** hinzugefügt.



Attribute

Das Control erbt von TcHmiRegion und hat somit dieselben Attribute. Das Attribut TargetContent wird durch das Attribut *Pages* ersetzt.

Pages

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Paginator.Pages
```

Anlegen der Seiten durch die navigiert werden soll.

Funktionen

GoForward

Beim Aufruf der Funktion wird zur nächsten Seite navigiert. Befindet sich die Navigation aktuell auf der letzten Seite, erfolgt ein Wechsel zur ersten Seite.

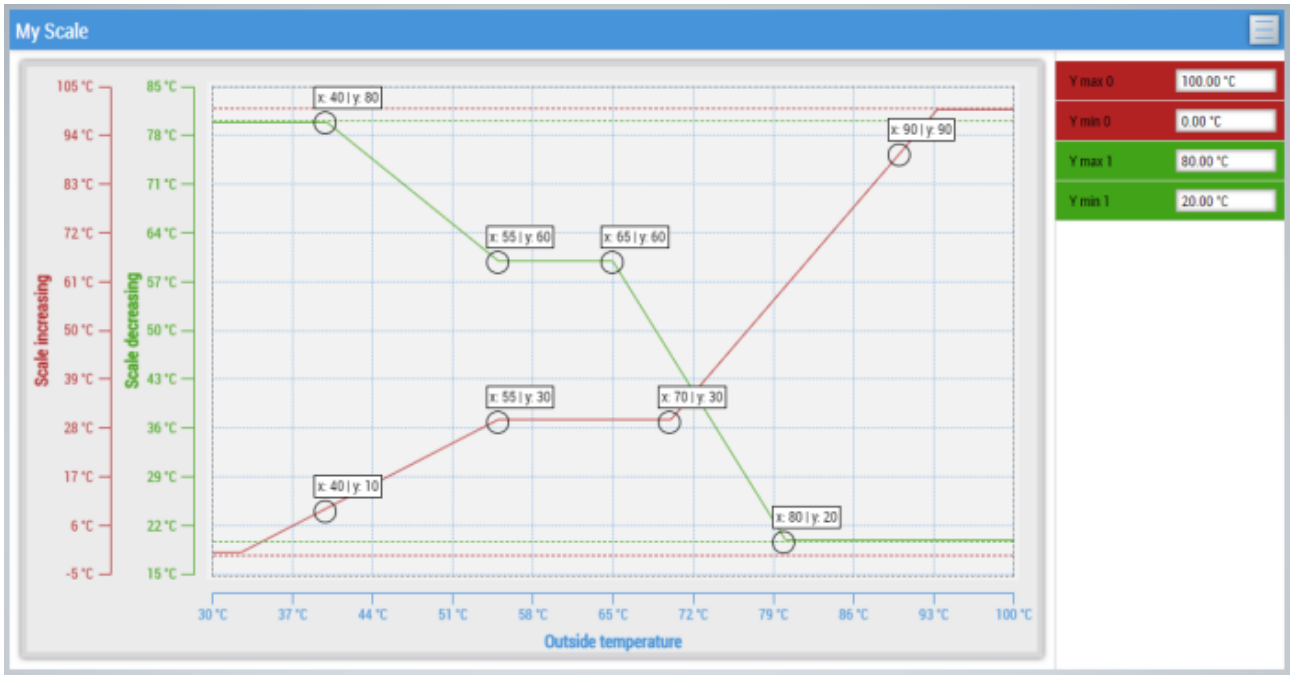
GoBackward

Beim Aufruf der Funktion wird zur vorherigen Seite navigiert. Befindet sich die Navigation aktuell auf der ersten Seite, erfolgt ein Wechsel zur letzten Seite.

6.2.1.2.2.13 Scale

Mit dem Control **Scale** können verschiedene Arten von Skalen angezeigt und bearbeitet werden.

Der Fokus des Controls ist die Visualisierung einer Heizkurve mit einer definierten Anzahl an Stützpunkten.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der Skalen angezeigt bzw. bearbeitet werden sollen.

Features

Eine Skala kann per Drag-and-Drop bearbeitet werden und es können der maximale und minimale Wert gesetzt werden.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

Data

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Scale.Scales
```

Legt die Daten für die verschiedenen Skalen fest.

XAxisExtension

```
tchmi:general#/definitions/Number
```

Legt fest wie viel länger die X-Achse angezeigt wird, abhängig von XMin und XMax.

Title

```
tchmi:general#/definitions/String
```

Legt den Titel fest, der in der Kopfzeile des Controls angezeigt wird.

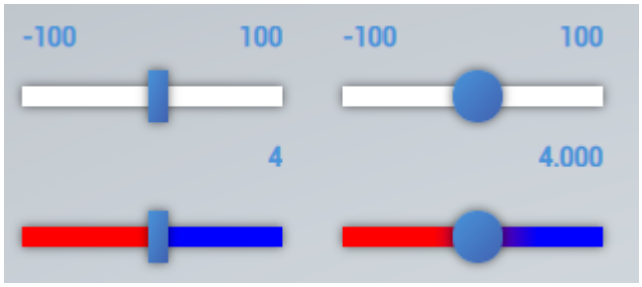
ShowHeader

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Kopfzeile des Controls angezeigt wird oder nicht.

6.2.1.2.2.14 Slider

Der **Slider** kann dazu verwendet werden, numerische Werte anzuzeigen und zu bearbeiten.

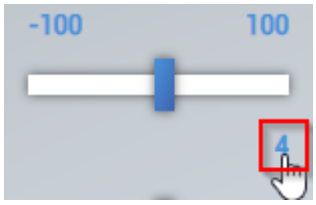


Verwendung

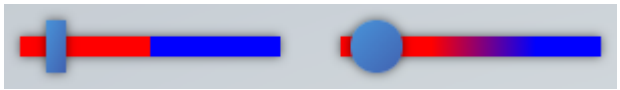
Kann auf jede beliebige Seite gezogen werden, auf der numerische Werte bearbeitet werden sollen.

Features

Der Wert kann per Drag-and-Drop oder per Klick auf den Slider eingestellt werden. Ebenfalls kann das Display, welches den aktuellen Wert anzeigt, angeklickt und daraufhin der gewünschte Wert eingegeben werden.



Es kann eingestellt werden, ob der Min- und Max-Wert bzw. der aktuelle Werte angezeigt werden.
Für den Slider können verschiedene Bereiche eingefärbt werden. Hier gibt es die Möglichkeit Farbverläufe oder genaue Farbbereiche einzustellen.



Wird das Feedback-Konzept [▶ 1171] genutzt, so wird der Wert des Feedbacks mit einem leichten Schatten angezeigt. Somit können z. B. bei einem Objekt, welches einen Soll- und Ist-Wert hat, beide Werte gleichzeitig visualisiert werden.



Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

ShowValue

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der aktuelle Wert angezeigt wird.

ShowScale

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der MinValue [▶ 1175] und MaxValue [▶ 1175] angezeigt werden.

Orientation

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Orientation
```

Legt die Ausrichtung des Sliders fest (horizontal oder vertikal).

SwitchMinMax

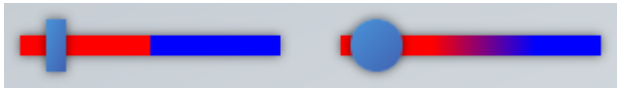
```
tchmi:general#/definitions/Boolean
```

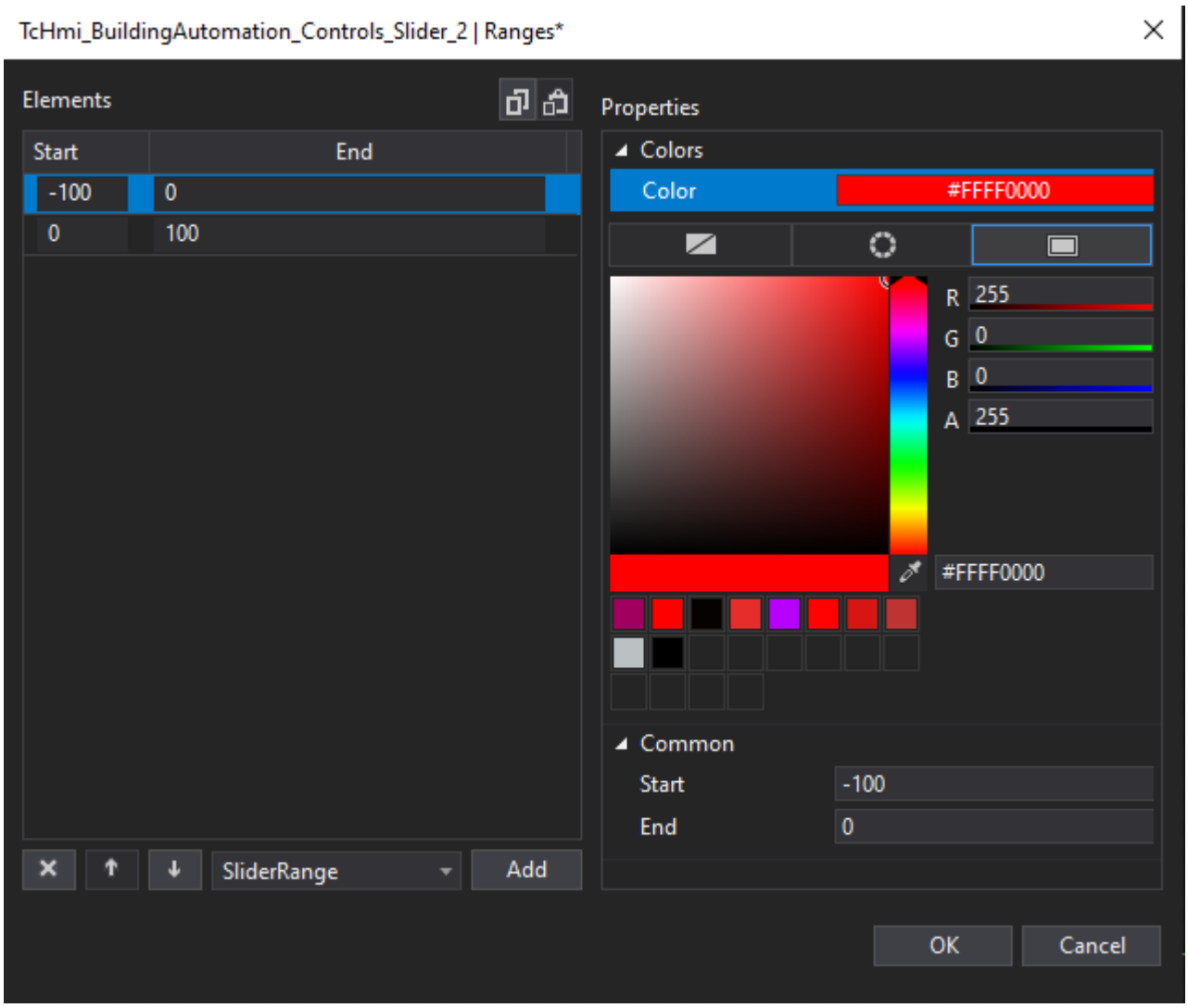
Wenn aktiv, werden die Positionen von *MinValue* und *MaxValue* getauscht.

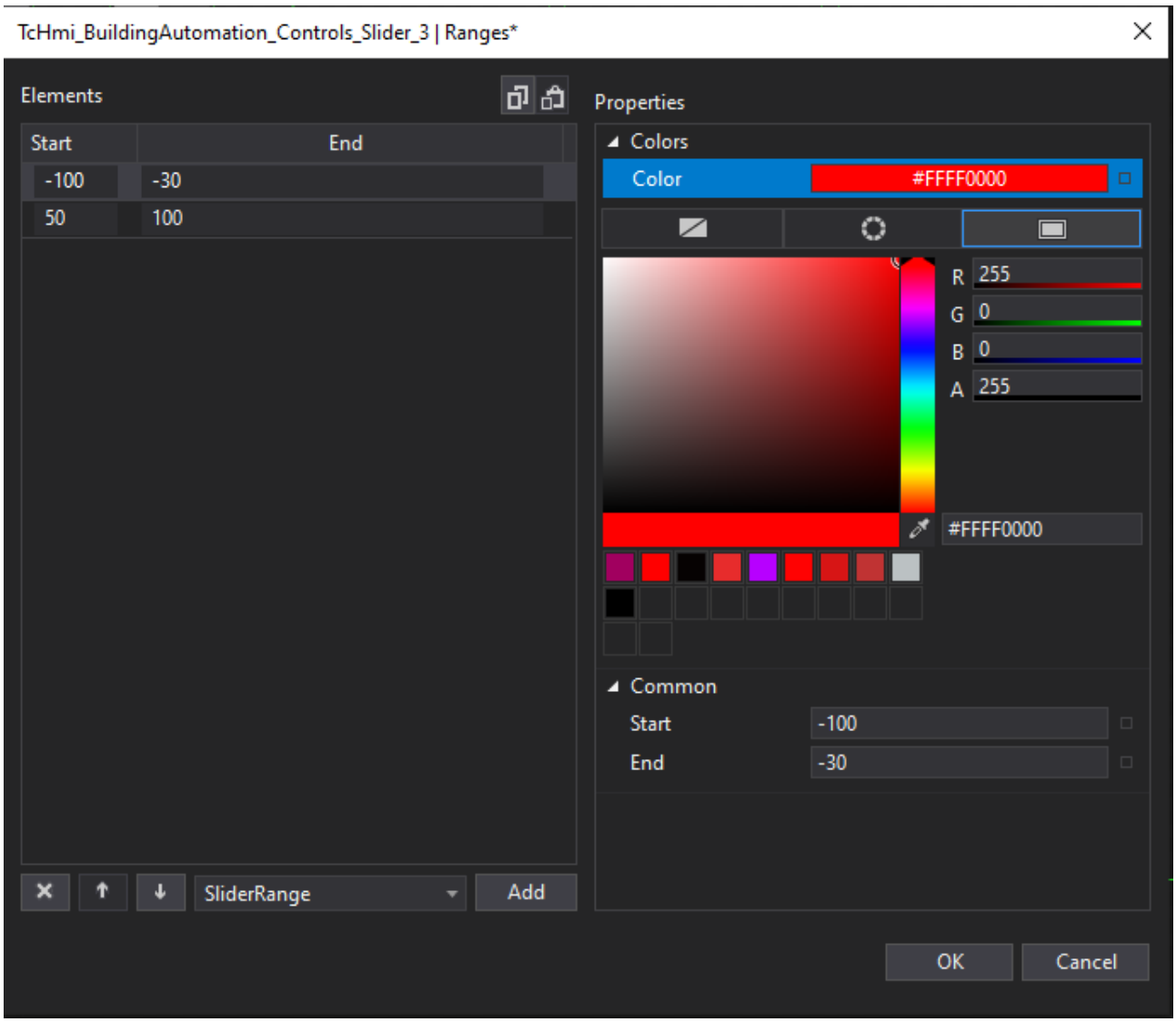
Ranges

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Slider.SliderRanges
```

Legt verschiedene Farbbereiche oder Farbverläufe fest, die im Slider dargestellt werden.




TcHmi_BuildingAutomation_Controls_Slider_2 | Ranges*
Elements
Start
End
-100
0
0
100
SliderRange
Add
Properties
Colors
Color
#FFFF0000
R
255
G
0
B
0
A
255
#FFFF0000
Common
Start
-100
End
0
OK
Cancel



KnobAppearance

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Slider.KnobAppearance`

Legt die Darstellung des Knopfes des Sliders fest.

BaData

BaInterface

`tchmi:framework#/definitions/Symbol`

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

BaInterfaceSymbolNames

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Slider.BaInterfaceSymbolNames`

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

Value

`tchmi:general#/definitions/Number`

Der aktuelle Wert des Sliders.

ValueFeedback

`tchmi:general#/definitions/Number`

Das Feedback für den Wert des Sliders.

Number

Unit

```
tchmi:general#/definitions/String
```

Legt die Einheit fest, die hinter Value [▶ 1174] angezeigt wird.

MinValue

```
tchmi:general#/definitions/Number
```

Der minimale Wert des Sliders.

MaxValue

```
tchmi:general#/definitions/Number
```

Der maximale Wert des Sliders.

Step

```
tchmi:general#/definitions/Number
```

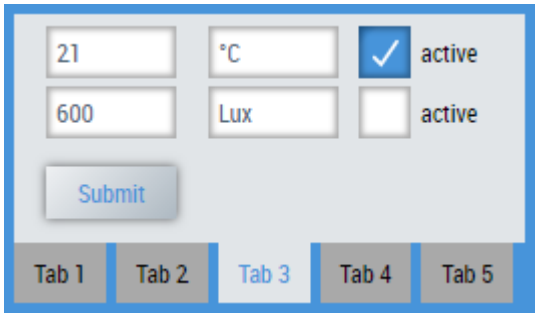
Legt fest mit welcher Genauigkeit der Wert mit dem Slider eingestellt werden kann (z. B. 0,01).

Events

Ereignis	Beschreibung
OnValueChanged	Das Event wird jedes Mal ausgelöst, wenn sich der Wert des Sliders ändert, z. B. wenn der Slider bewegt wird.
onUserInteractionFinished	Das Event wird ausgelöst, wenn die Wertänderung vom Benutzer abgeschlossen wurde. Das passiert bei Drag-and-Drop, wenn der Benutzer den Slider wieder loslässt oder nach einem Klick auf einen Bereich des Sliders.

6.2.1.2.2.15 TabWindow

Das **TabWindow** dient zur Anzeige von verschiedenen Inhalten in unterschiedlichen Tabs.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein TabWindow angezeigt werden soll.

Features

Den Tabs können über das Attribut *Data* Seiten vom Typ *.content oder programmatisch erstelltes HTML zugewiesen werden.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Common

Data

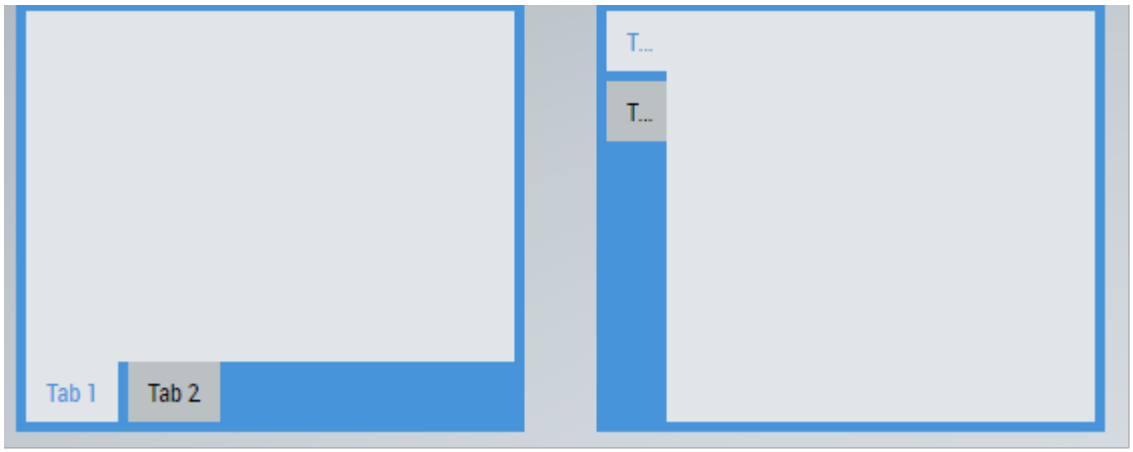
```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TabWindow.TabWindowData
```

Inhalte für die verschiedenen Tabs.

TabPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position
```

Position der Tabs.



TabDistance

```
Tchmi:framework#/definitions/Number
```

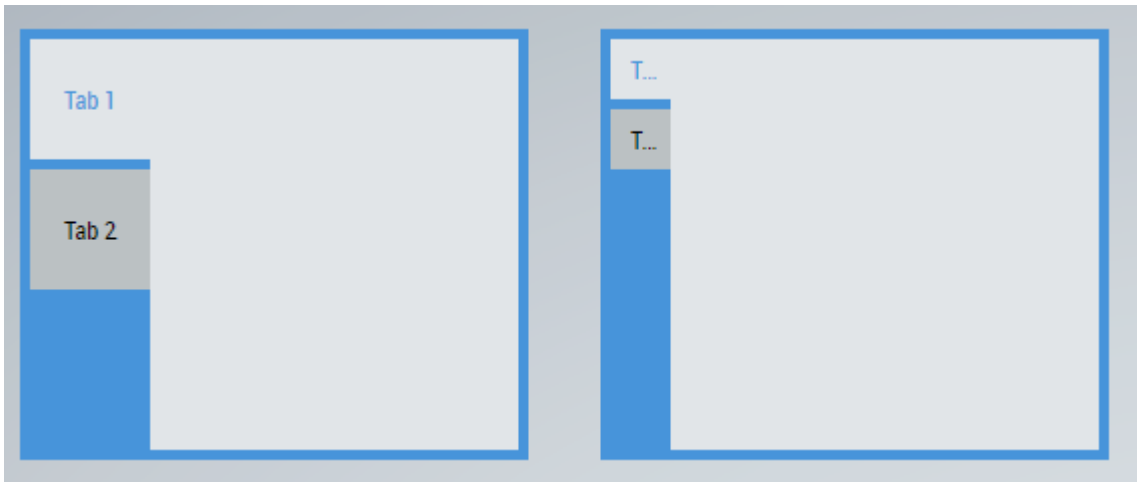
Abstand zwischen den Tabs.



TabContainerDistance

```
tchmi:general#/definitions/Number
```

Die Größe des Tab-Containers.



TabSizeAuto

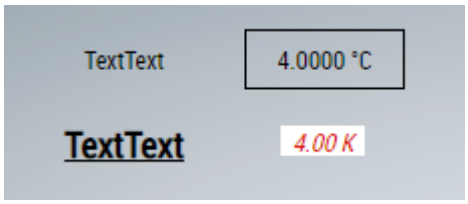
```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Tabs die komplette Breite des Tab-Containers einnehmen sollen oder nicht.



6.2.1.2.2.16 Textblock

Das Control **Textblock** dient zur Anzeige von Text.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der Text angezeigt werden soll.

Features

Ein spezialisierter Textblock mit vermindertem Ressourcenbedarf.

Attribute

Das Control erbt vom TextControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Digits

```
tchmi:general#/definitions/Number
```

Anzahl der Nachkommastellen, wenn ein analoges BaObjekt [▶ 62] verknüpft ist.

Common

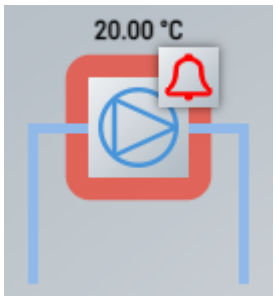
Text

```
tchmi:general#/definitions/String
```

Text für den Textblock.

6.2.1.2.2.17 UiIcon

Das Control **UiIcon** kann dazu verwendet werden, Events und Werte anzuzeigen. Es stellt sich wie ein normaler Button [▶ 1149] dar und kann mit verschiedenen Icons gefüllt werden.



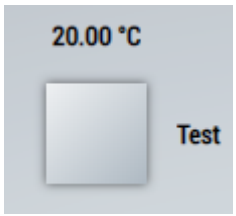
Verwendung

Eignet sich zum Erstellen von R+I-Schemas zum Abbilden verschiedener Anlagenteile (z. B. Pumpe). Mit Hilfe des Attributes *Connections* lassen sich entsprechende Anschlüsse erstellen, um das UiIcon z. B. mit einer Hauptleitung zu verbinden.

Features

Werteanzeigen

Dem **UiIcon** können verschiedene Anzeigen über das Attribut *DisplaysData* hinzugefügt werden.

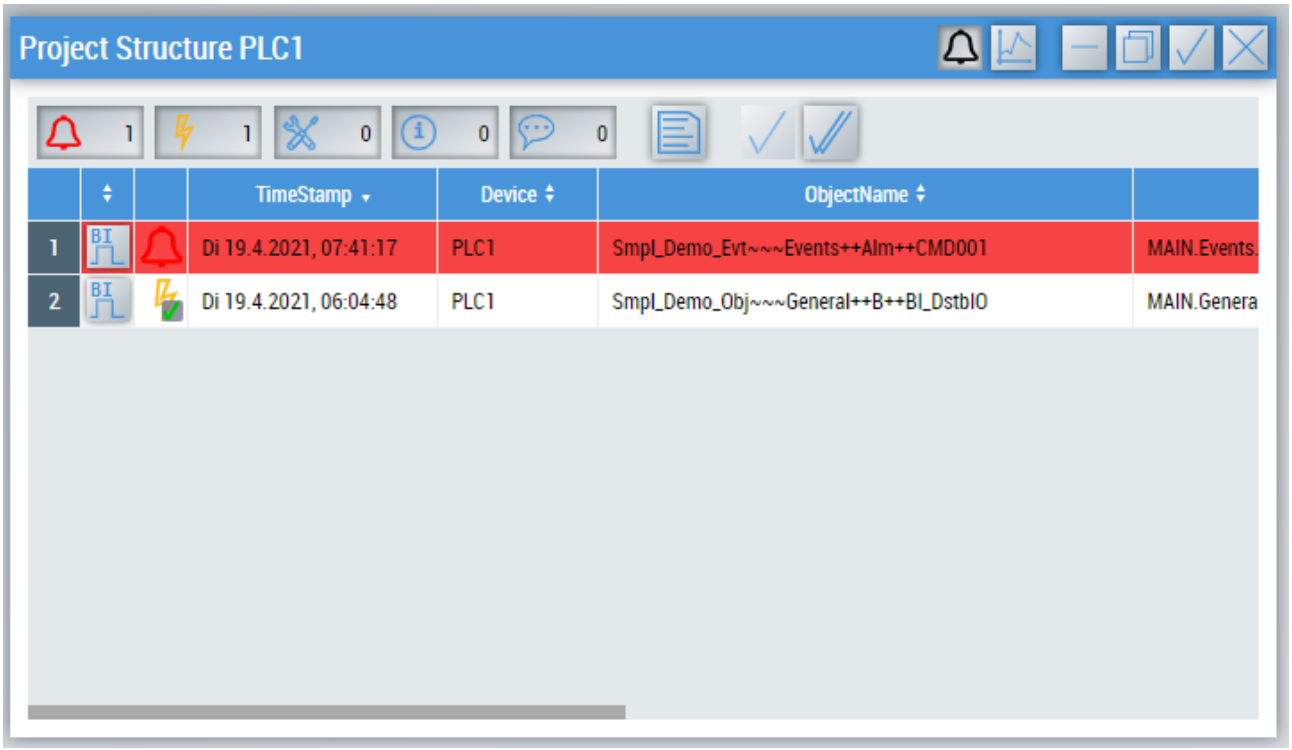


Eventanzeigen

Über das Attribut *EventsData* lassen sich um das **UiIcon** verschiedene Events anzeigen.



Wird der generische Ansatz [▶ 79] von TcHmiBa verwendet und ein BaObject / BaView mit dem Control verknüpft, werden aktive Events automatisch angezeigt. Beim Betätigen des **UiIcon** öffnet sich die Projektnavigation [▶ 1186] des verknüpften Objektes und bei einem Event entsprechend das Parameterfenster [▶ 1189] mit der Eventansicht.



Attribute

Das Control erbt vom Button [▶ 1149] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



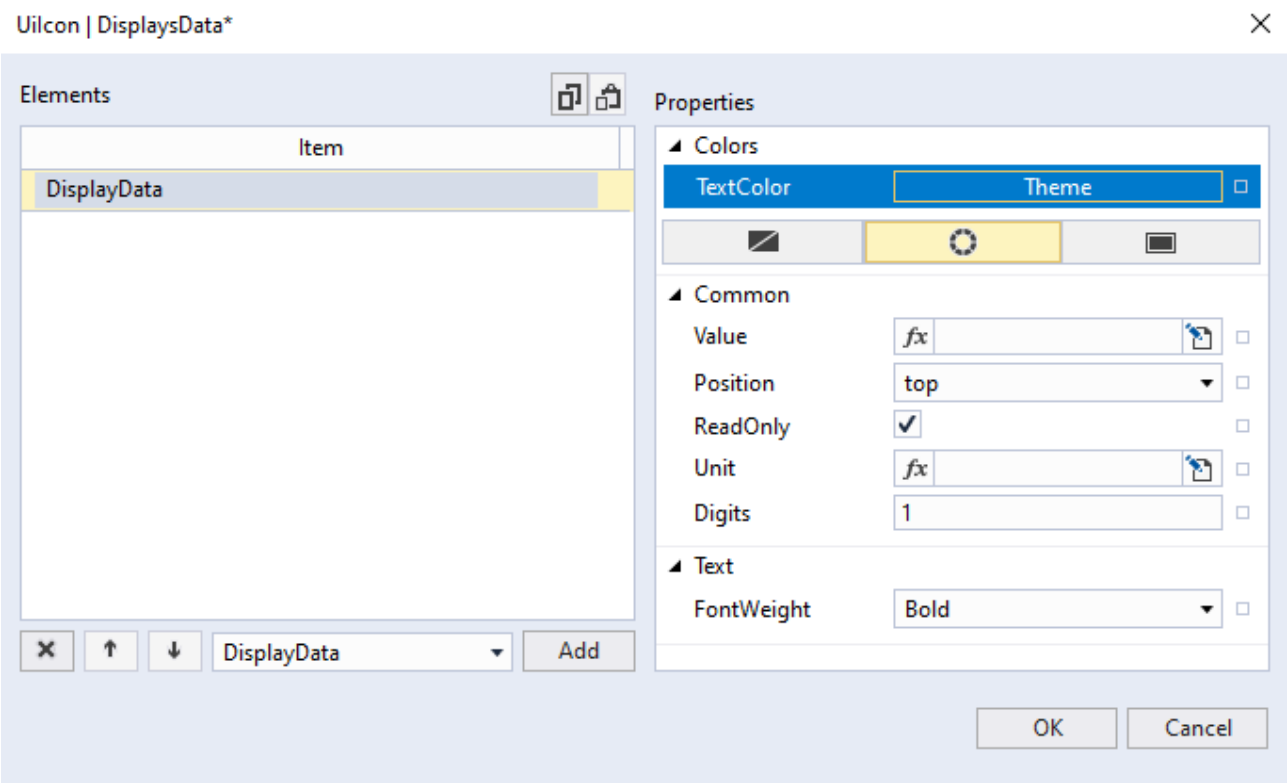
Das Attribut ist nicht bei allen Controls anwendbar.

Common

DisplaysData

```
TcHmi.BuildingAutomation.Controls.UiIcon.DisplaysData
```

Das Attribut ermöglicht es, verschiedene Displays über einen Editor anzulegen.



Für jedes Display lassen sich folgende Eigenschaften festlegen:

Name	Beschreibung
TextColor	Schriftfarbe des Displays.
Value	Anzeigewert im Display. Existiert ein Binding und ReadOnly ist deaktiviert, so wird der Wert auf dieses Binding geschrieben, wenn der Benutzer die Eingabe beendet.
Position	Position des Displays. Mehrere an derselben Position erstellte Displays werden übereinander angeordnet.
ReadOnly	Legt fest, ob das Display editierbar oder nur lesbar ist.
Unit	Anzuhängende Einheit an den Wert (wenn es sich um eine Zahl handelt).
Digits	Anzahl der Nachkommastellen.
FontWeight	Schriftstärke des Texts.

DisplayedDigits

```
tchmi:general#/definitions/Boolean
```

Die Anzahl der Nachkommastellen, wenn eine Zahl in einem Display dargestellt wird.

IconStatus

```
TcHmi.BuildingAutomation.Controls.Common.UiIcon.Status
```

Färbt das Icon gemäß des gesetzten Status ein.

Das automatische Einfärben wird über die globale Variable AutoActivateIconStatus [▶ 1292] eingestellt.

Status	Anzeige
Alarm	
Störung	
Wartung	
Benachrichtigung	
Andere	
Aktiv	
Inaktiv	

Ändern der verwendeten Farben über folgende CSS-Variablen in der Theme-CSS-Datei:

```
:root {
    --tchmi-ba-global-event-color-alarm: rgb(255, 0, 0);
      --tchmi-ba-global-event-color-disturb: rgb(255, 255, 0);
      --tchmi-ba-global-event-color-maintenance: rgb(255, 255, 0);
      --tchmi-ba-global-event-color-notification: rgb(255, 255, 55);
      --tchmi-ba-global-event-color-other: rgb(255, 255, 255);
      --tchmi-ba-global-color-active: rgb(0, 255, 0);
      --tchmi-ba-global-color-inactive: rgb(255, 255, 255);
}
```

Es ist zu beachten, dass Änderungen an diesen Farben pro Visualisierung gelten, wodurch die Wartung verschiedener Anlagen aufgrund von unterschiedlichem Aussehen erschwert werden kann.

BorderStatus

```
TcHmi.BuildingAutomation.Controls.Common.UiIcon.Status
```

Färbt das Icon gemäß des gesetzten Status ein.

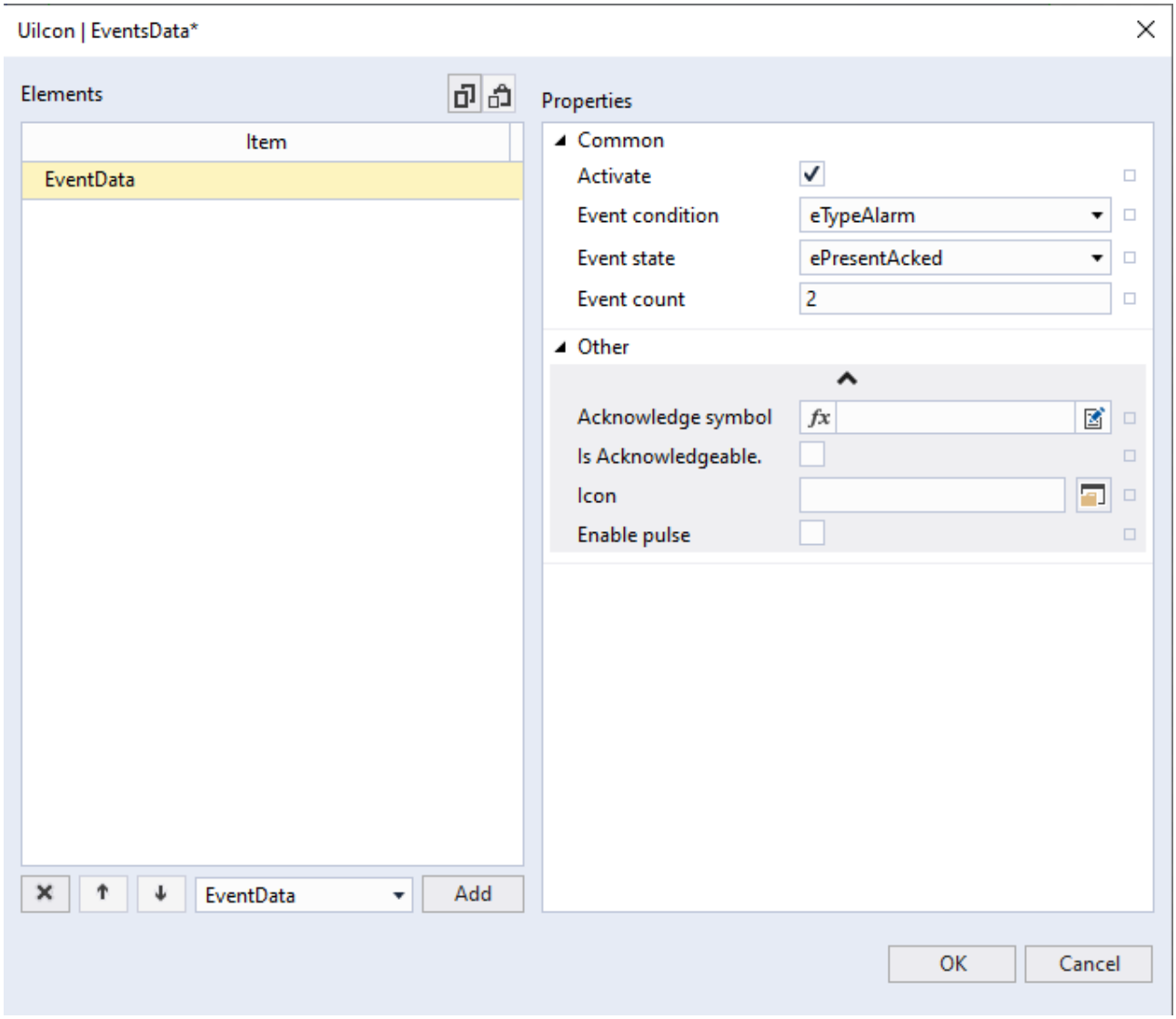
Status	Anzeige
Alarm	
Störung	
Wartung	
Benachrichtigung	
Andere	
Aktiv	
Inaktiv	

Verwendete Farben siehe Attribut IconStatus.

EventsData

`TcHmi.BuildingAutomation.Controls.UiIcon.EventsData`

Das Attribut ermöglicht es, verschiedene Events über einen Editor anzulegen.



Für jedes Event lassen sich folgende Eigenschaften festlegen:

Name	Beschreibung
Activate	Legt fest, ob das Event aktiv ist oder nicht.
Event condition	Legt den Typ (die Priorität) des Events fest. Die Icons werden entsprechend ihrer Priorität entlang des Uhrzeigersinns angeordnet. Oben rechts ist dabei die höchste Priorität.
Event state	Aktueller Zustand des Events.
Event count	Legt fest, wie viele Events dieses Typs und Zustands aktiv sind.
Acknowledge symbol	Schreibt ein TRUE auf das Symbol, wenn auf das Event gedrückt wird.
Is Acknowledgable	Legt fest, ob auf das Event gedrückt werden kann.
Icon	Zu verwendendes Icon, wenn keine *Event condition* ausgewählt wurde, um benutzerspezifische Icons zu zulassen.
Enable pulse	Auswertung erfolgt nur, wenn *Event condition* und *Event state* nicht verwendet werden. Wenn aktiviert, wird ein roter Puls um das **UiIcon** angezeigt.

ShowDisplays

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die im Attribut *DisplayData* definierten Displays angezeigt werden oder nicht.

PopUp

ShowFaceplate

`tchmi:general#/definitions/Boolean`

Legt fest, ob beim Klick auf das UiIcon ein PopUp geöffnet wird.

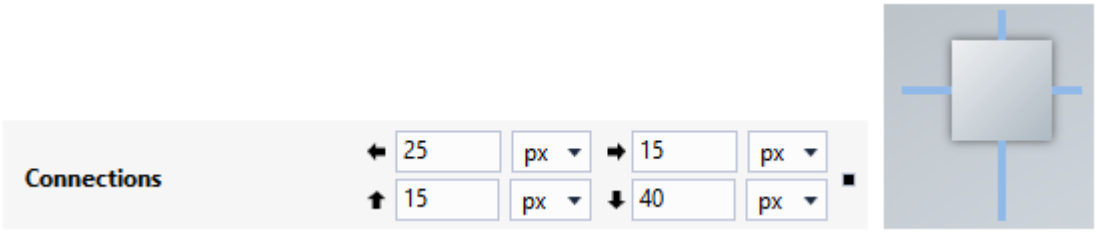
Ist das Attribut nicht aktiviert, kann das UiIcon bspw. für die Navigation zu einem anderen Content verwendet werden aber trotzdem Events des verknüpften BaObjects anzeigen.

Connections

Mit den Verbindungen können Anschlüsse an andere Leitungen in einem R+I-Schema dargestellt werden.

`tchmi:framework#/definitions/Padding`

Hier können Anschlüsse erstellt werden, die vertikal oder horizontal vom UiIcon weggehen.



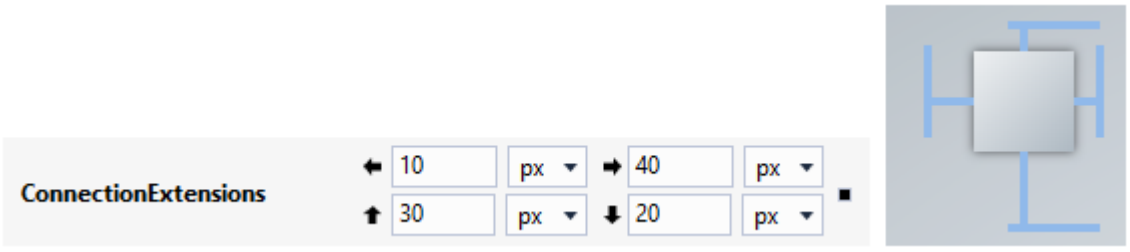
Dabei ist jeweils die Länge des Anschlusses anzugeben.

Es wird immer die Einheit Pixel verwendet. Prozent wird an dieser Stelle **nicht** unterstützt.

ConnectionExtensions

`tchmi:framework#/definitions/Padding`

Hier können Erweiterungen für die oben erstellten Anschlüsse angelegt werden.



Dabei ist jeweils die Länge der Erweiterung anzugeben.

Es wird immer die Einheit Pixel verwendet. Prozent wird an dieser Stelle **nicht** unterstützt.

ConnectionsWidth

`tchmi:framework#/definitions/PositiveNumber`

Angabe der Breite in Pixel für die Anschlüsse.

ConnectionsColor

`tchmi:framework#/definitions/SolidColor`

Angabe der Farbe für die Anschlüsse.

ConnectionsColorPerSide

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.FourSidedColor
```

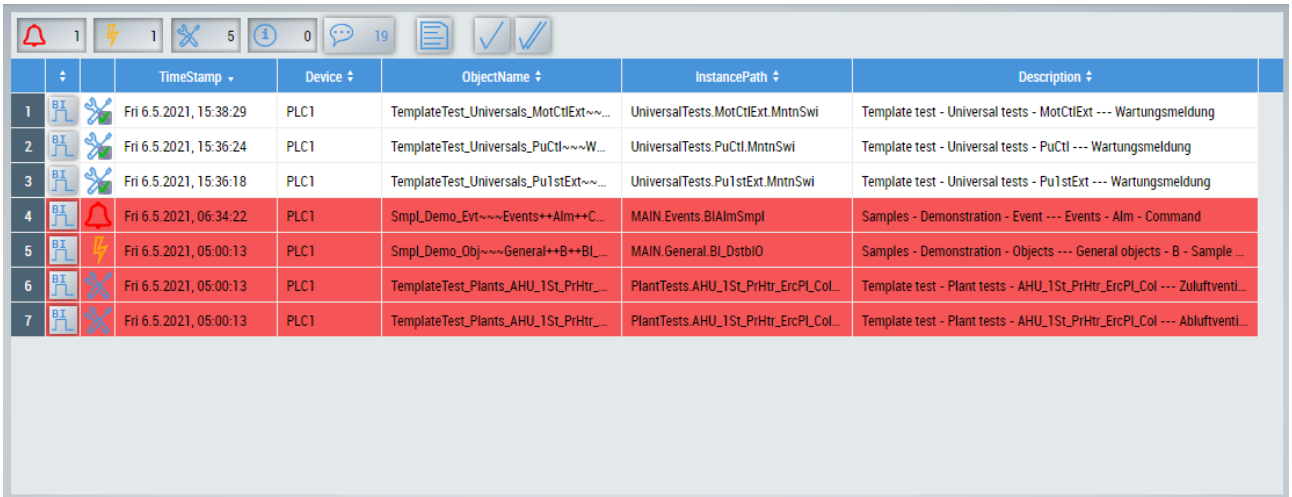
Definiert die Farbe für verschiedene Anschlüsse. Das Attribut *ConnectionsColor* muss dabei auf *NULL* oder *NONE* gesetzt werden.

6.2.1.2.3 Management

6.2.1.2.3.1 EventList

Die **EventList** zeigt Events in Listenform an.

Für die Verwendung des Controls müssen die generischen Funktionalitäten [▶ 79] von TcHmiBa verwendet werden.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der Events aufgelistet werden sollen.

Features

Zeigt die Events von einem bestimmten BaObject oder BaView an. Es kann aber auch die Events aller verbundenen Steuerungen anzeigen.
Über die Buttons im oberen Bereich lassen sich Events nach verschiedenen Eventtypen filtern.



Ermöglicht das Quittieren eines oder aller Events. Dabei wird jeweils das aktuell in der Liste selektierte Event quittiert.



Der Button **History** blendet die Event-Historie ein oder aus.

Attribute

Das Control erbt vom BaseRoomControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

IsGlobalEventList

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Events aller verbundenen Steuerungen angezeigt werden sollen.

ActiveEventsCount (read-only)

```
tchmi:general#/definitions/Number
```

Anzahl aktiver Events, die der Benutzer quittieren kann.

Columns

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Management.EventList.ColumnList
```

Legt die Reihenfolge und Einstellungen der Spalten fest.

Die Standardeinstellung können in den globalen Einstellungen [▶ 1287].



Erfolgt die Angabe der Spaltenbreite in der Einheit "*factor*", dann nutzen die Spalten automatisch den noch freien Platz.

ColumnSorting

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Management.EventList.ColumnSorting
```



Wurde durch das Attribute *Columns* ersetzt.

Legt die Reihenfolge der Spalten fest.

Die Standardeinstellung des Attributes kann auch global für alle EventList-Controls im CodeBehind überschrieben werden.

Events

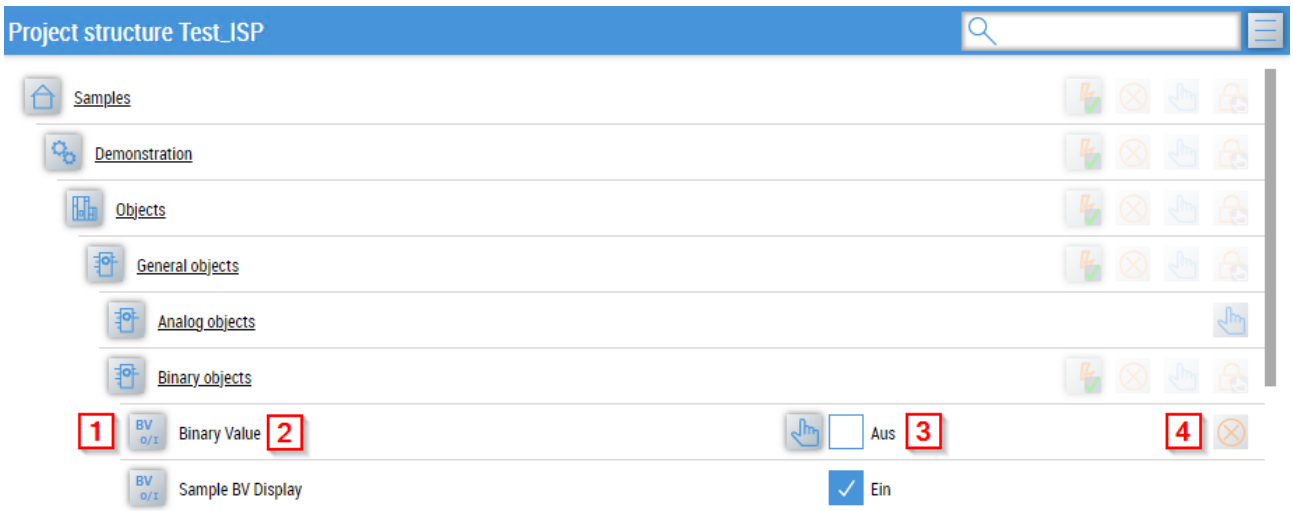
Ereignis	Beschreibung
onEventsChanged	Wird ausgelöst, wenn sich die Sammlung an Events geändert hat.
onEventAcknowledged	Wird ausgelöst, wenn ein Event quittiert wurde.
onAllEventsAcknowledged	Wird ausgelöst, wenn alle Events quittiert wurden.

6.2.1.2.3.2 ProjectNavigationTextual

Das **ProjectNavigationTextual** ist eines der generischen Controls und dient zur Navigation durch die Projektstruktur eines Gerätes. Es gibt Auskunft über den Typ (z.B. Analog Input oder Structured View) [1], die Beschreibung [2], den Wert [3] und die Events (wenn verfügbar) [4] der Objekte.



Hier finden Sie genauere Informationen über die generischen Möglichkeiten [▶ 79] von TcHmiBa und wie diese verwendet werden können.



Verwendung

Zum Navigieren muss das BaObject [▶ 1190]-Attribut verknüpft werden. Ist ein BaView verknüpft kann durch dessen Kinder navigiert werden. Ist nur ein einzelnes BaObject (kein BaView) verknüpft, dann wird nur ein Eintrag mit diesem BaObject angezeigt.

Features

Generische Navigation

Die Navigation baut sich anhand der Struktur des verknüpften BaObjects bzw. BaViews generisch auf, wodurch sich alle BaObjects eines BaViews mit nur einem Binding erreichen lassen.

Manuelle Übersteuerung

Die Wertanzeige und manuelle Übersteuerung wird mit dem ManualOverride [▶ 1165]-Control realisiert.

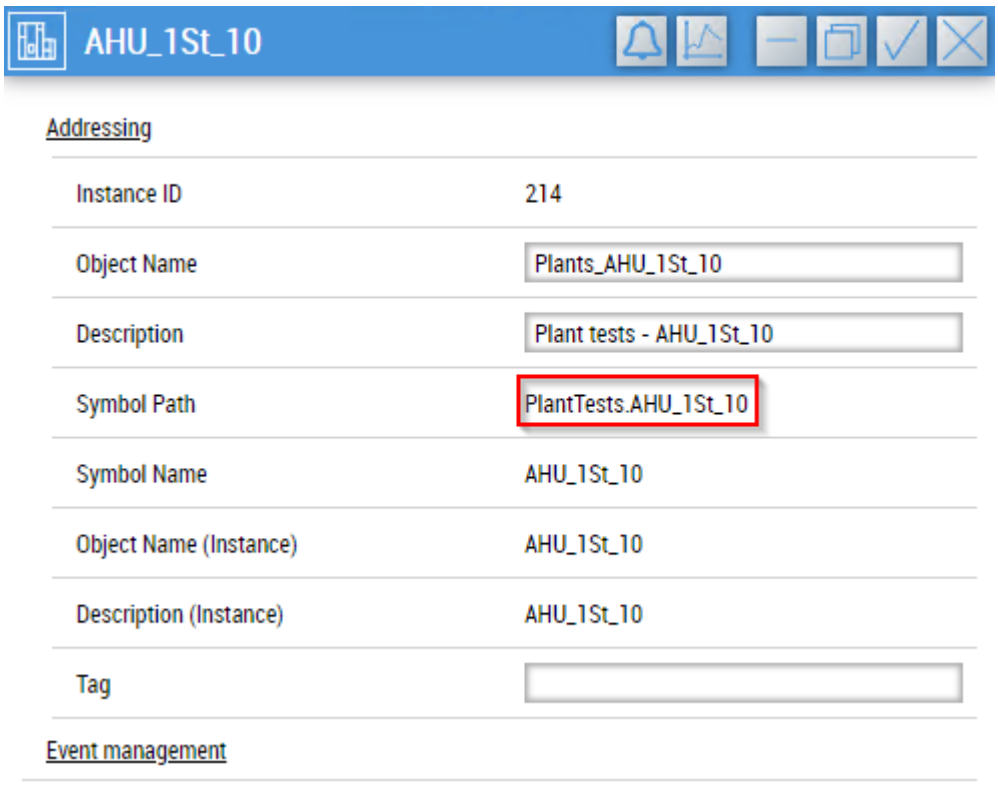
Verknüpfung Content Seite

Eine zusätzliche Navigationshilfe stellt die Funktion zum Verknüpfen von Content-Seiten dar.

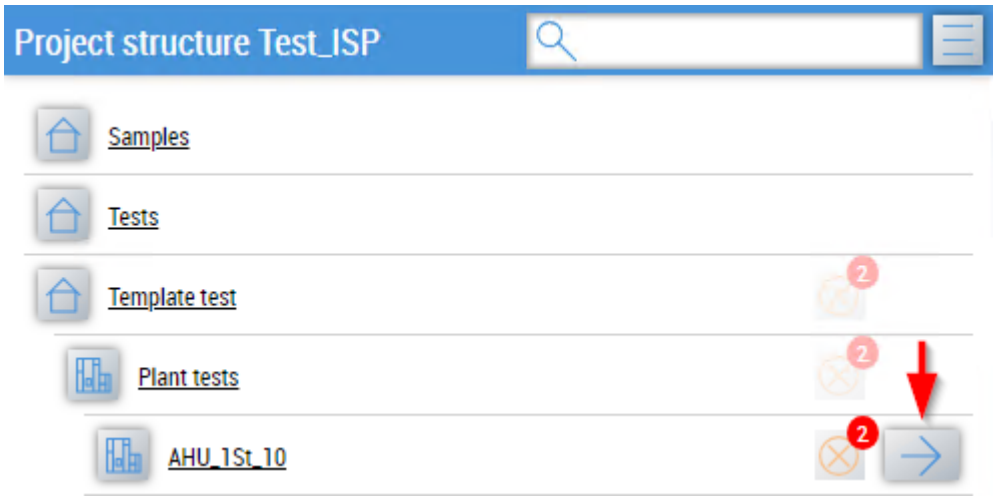
Im TcHmi-Projekt muss eine Content-Seite existieren die den **Symbol Path** eines Views als Namen hat.



Beim Namen der Content-Seite ist die Groß- und Kleinschreibung zu beachten.



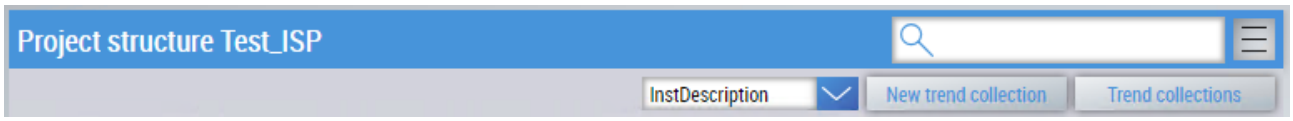
Anschließend erscheint in der Projektnavigation hinter dem betreffenden Eintrag ein Button.



Zusätzlich muss im TcHmi-Projekt eine TcHmiRegion mit dem Namen **TargetRegion** vorhanden sein, um die verknüpfte Content-Seite anzuzeigen.

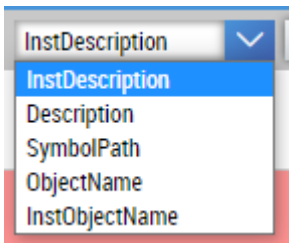
Kopfzeile

Die Kopfzeile enthält ein Eingabefeld zum Durchsuchen der Auflistung und ein Burger-Menü zur Einblendung zusätzlicher Aktionen.



Aktionen:

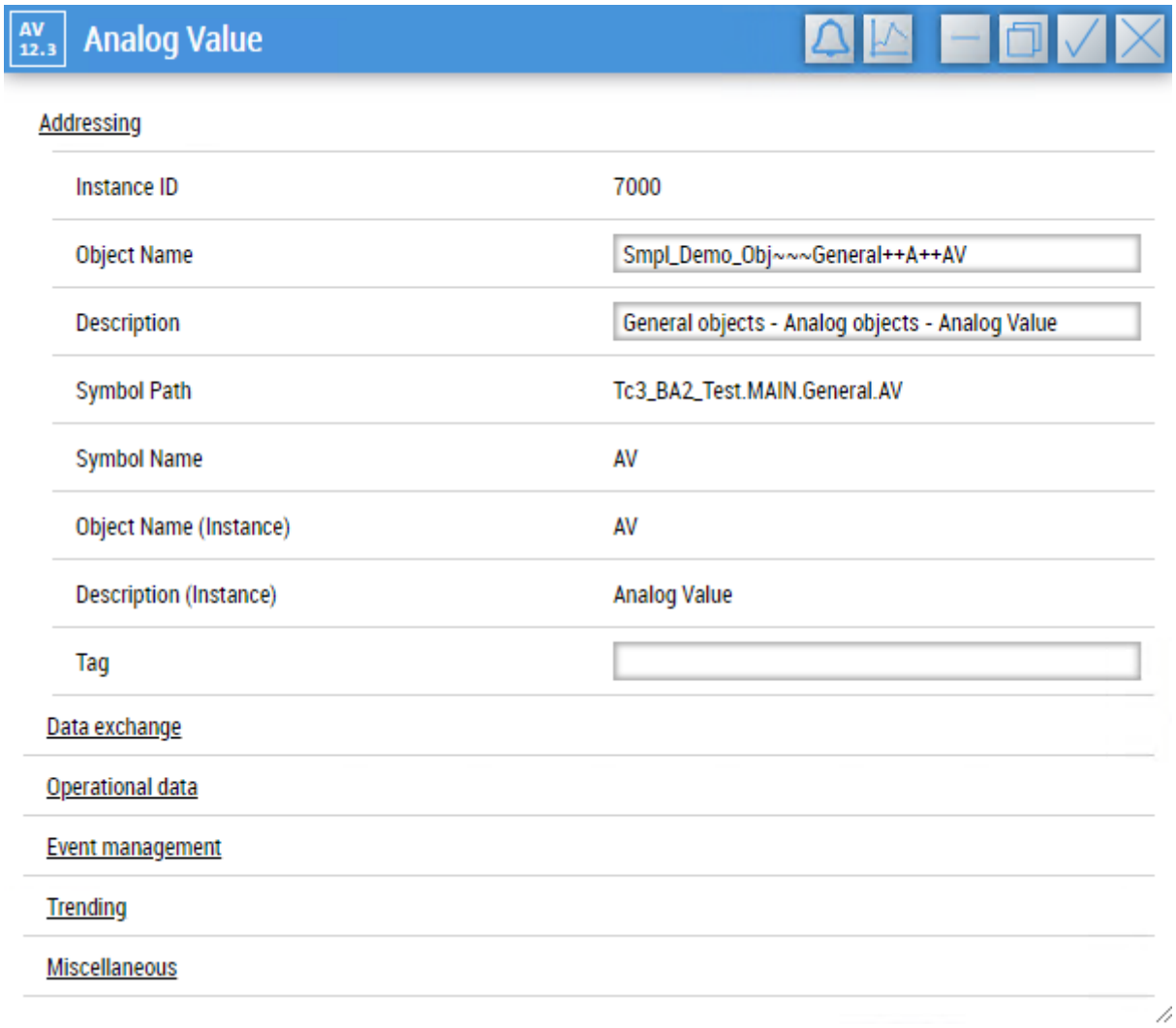
- Auswahl der anzuzeigenden Beschriftung für die Einträge



- Öffnen vom Trendkonfigurator [▶ 56]
- Öffnen erzeugter Trendkonfigurationen

Parameterdialog

Nach Betätigung des Typ-Icons erscheint der Parameterdialog des ausgewählten BaObjects.



Die aufgeführten Parameter sind in Kategorien unterteilt. Sichtbarkeit und zulässige Lese-/Schreibzugriffe richten sich nach der Rolle [▶ 15] des angemeldeten Benutzers.

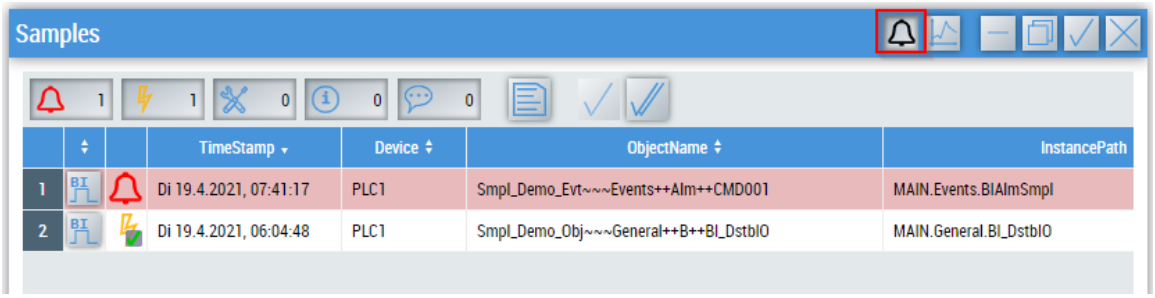
Änderungen an Parametern werden mit dem Haken-Icon in die SPS geschrieben und mit dem Kreuz-Icon verworfen.

Die Kategoriezugehörigkeit und Sichtbarkeit der Parameter kann global eingestellt werden (siehe Globale Einstellungen [▶ 1287]).

Je nach ausgewähltem BaObject können in der Kopfzeile weitere Buttons erscheinen.

Buttons:

- Eventglocke
  - bei jedem eventfähigen Objekt verfügbar
  - ersetzt beim Anklicken den Inhalt des Fensters mit der Eventliste [▶ 1185] des Objektes



- Trend-Funktionalitäten
  - nur bei Objekten vorhanden, die diese unterstützen bzw. Views die Objekte mit Trendfunktionen enthalten.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

BaUsedTitle

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.ProjectNavigationTextual.BaUsedTitle
```

Legt fest, welcher Parameter für die Beschreibung in einem Eintrag genutzt wird. Die Einstellung ist im Client anpassbar.

Show Header

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Kopfzeile angezeigt wird oder nicht.

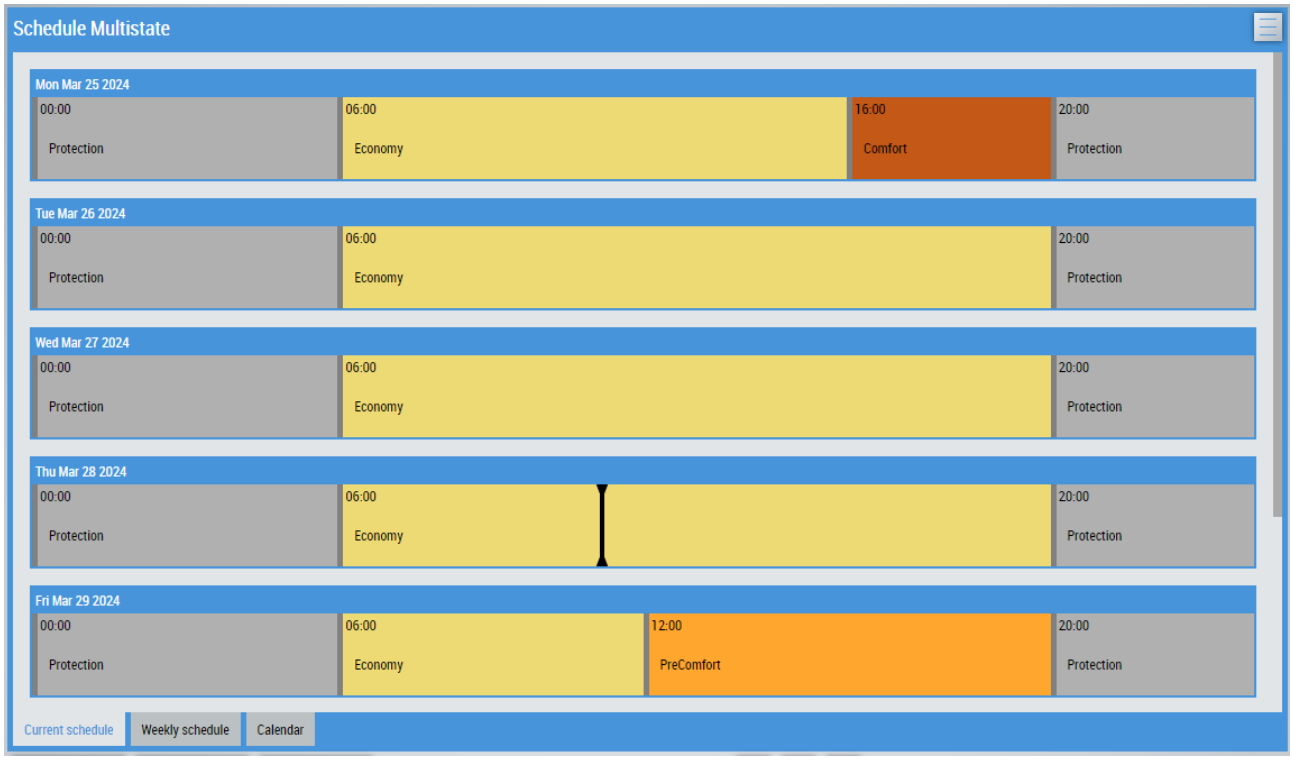
AutoCollapse

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob Listeneinträge automatisch geschlossen werden, wenn ein anderer Listeneintrag ausgewählt wurde.

6.2.1.2.3.3 Schedule

Der **Schedule** kann zur Anzeige und Bedienung von Zeitschaltplänen und Kalendereinträgen verwendet werden. Der aktuelle Zeitschaltplan wird auf Basis des Wochenzeitschaltplans und der Ausnahmen erstellt.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Zeitschaltplan verwaltet werden soll.

Wird dem Attribut BaObject [▶ 1193] ein Schedule-Objekt übergeben, lassen sich die generischen Funktionen [▶ 79] nutzen.

Features

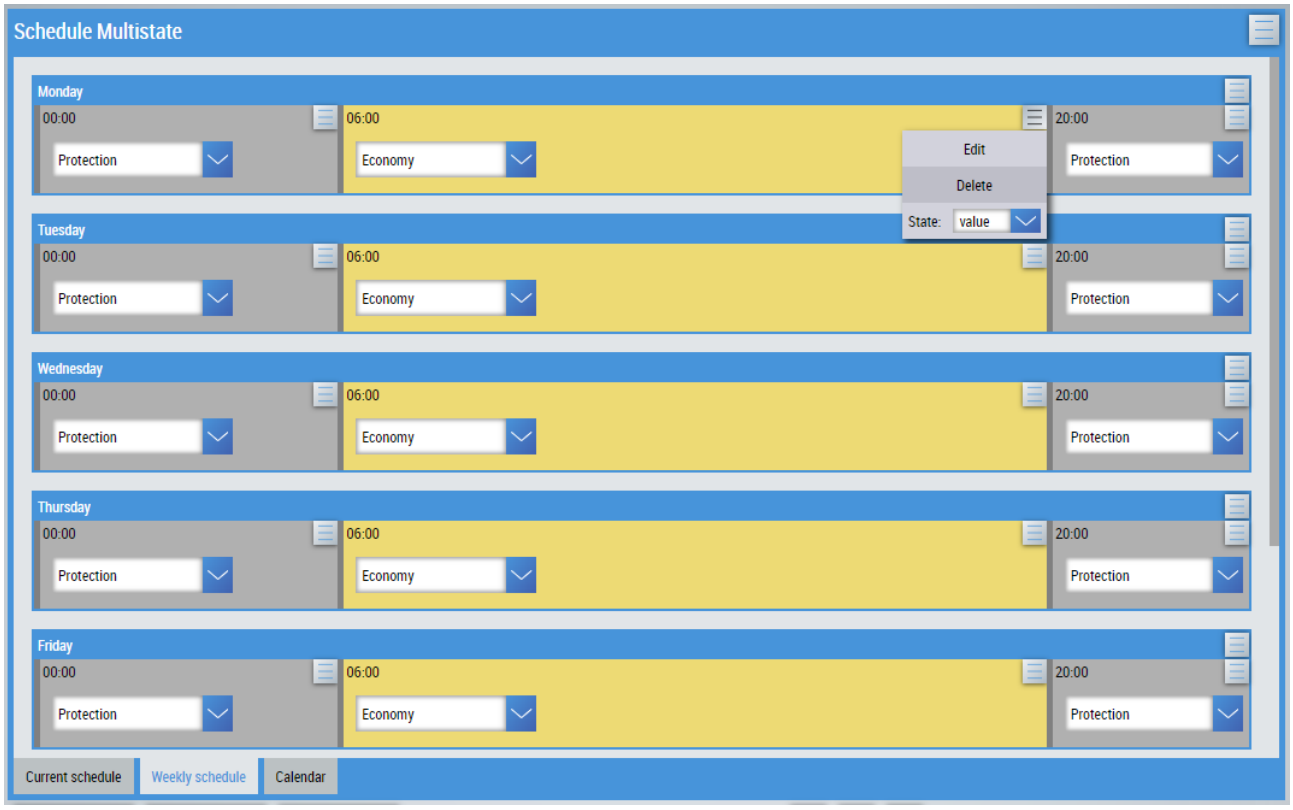
Resultierender Zeitplan

Der erste Reiter **Aktueller Zeitschaltplan** zeigt die Kombination aus dem wöchentlichen Zeitschaltplan und den Ausnahmen an. Hierbei gilt folgende Hierarchie:

1. Lokale Ausnahmen
2. Globale Ausnahmen
3. Wöchentlicher Zeitschaltplan

Bearbeiten vom Wochenzeitschaltplan

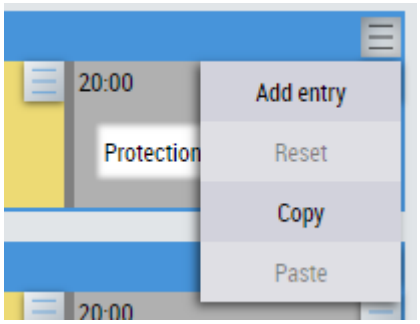
Im Reiter **Wöchentlicher Zeitschaltplan** kann der jede Woche geltende Zeitschaltplan bearbeiten werden, ohne Berücksichtigung bereits definierter Ausnahmen.



In dieser Ansicht gibt es für jeden Tag einen Zeitschaltplan mit verschiedenen Einträgen. Ein Eintrag kann über sein Menü bearbeitet oder gelöscht werden. Die Start- und Endzeit bzw. Position, lässt sich auch mit der Maus oder dem Finger verändern.

Jeder Tageszeitschaltplan hat ebenfalls ein Menü, über das sich Einträge hinzufügen und Änderungen zurücksetzen lassen.

Durch Auswahl der Kopierfunktion werden die Einträge des Tages kopiert und können an anderen Tagen eingefügt werden.
Das Kopieren ist auch in andere Zeitschaltplänen oder Ausnahmen möglich.



Verwalten von Ausnahmen

Im Reiter **Kalender** werden die Ausnahmen verwaltet. Weitere Informationen zur Verwendung sind im Control Calendar [▶ 1150] zu finden.

Menü

Über das Menü im oberen rechten Bereich vom Schedule, lassen sich entweder alle durchgeführten Änderungen zur SPS übertragen oder verwerfen.



Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

Orientation

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Orientation
```

Legt die Ausrichtung des wöchentlichen Zeitschaltplans fest.

SnapPeriode

```
tchmi:general#/definitions/Number
```

Legt fest, wie genau sich Zeitschaltplaneinträge einstellen lassen. Wenn *SnapPeriode* z. B. auf 15 eingestellt wird, können Einträge jeweils auf eine Viertelstunde genau eingestellt werden.

DisplayTimeCursor

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob ein Cursor für die aktuelle Uhrzeit im resultierendem Zeitschaltplan angezeigt wird.

CurrentDateTime

```
tchmi:general#/definitions/DateTime
```

Legt die Zeit fest, die der Cursor für die aktuelle Uhrzeit anzeigt.

Wird ein BaObject verwendet, wird die aktuelle Zeit vom entsprechenden Gerät angezeigt.

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Conrols.Management.Schedule.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

ActiveText

```
tchmi:general#/definitions/String
```

Legt den Text fest, der in den Einträgen angezeigt wird, wenn ein binärer Zeitschaltplan verknüpft wurde und der Wert TRUE ist.

InactiveText

```
tchmi:general#/definitions/String
```

Legt den Text fest, der in den Einträgen angezeigt wird, wenn ein binärer Zeitschaltplan verknüpft wurde und der Wert FALSE ist.

StateTexts

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.Combobox.ComboboxItems
```

Legt die Texte fest, die in den Einträgen angezeigt werden, wenn ein Multistate Zeitschaltplan verknüpft wurde.

Unit

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.StringOrNumber
```

Legt die Einheit in den Einträgen fest, wenn ein analoger Zeitschaltplan verknüpft wurde. Mögliche Werte sind:

- textuell (z.B. "°C")
- numerisch (Enumerationswert von E_BA_Unit)

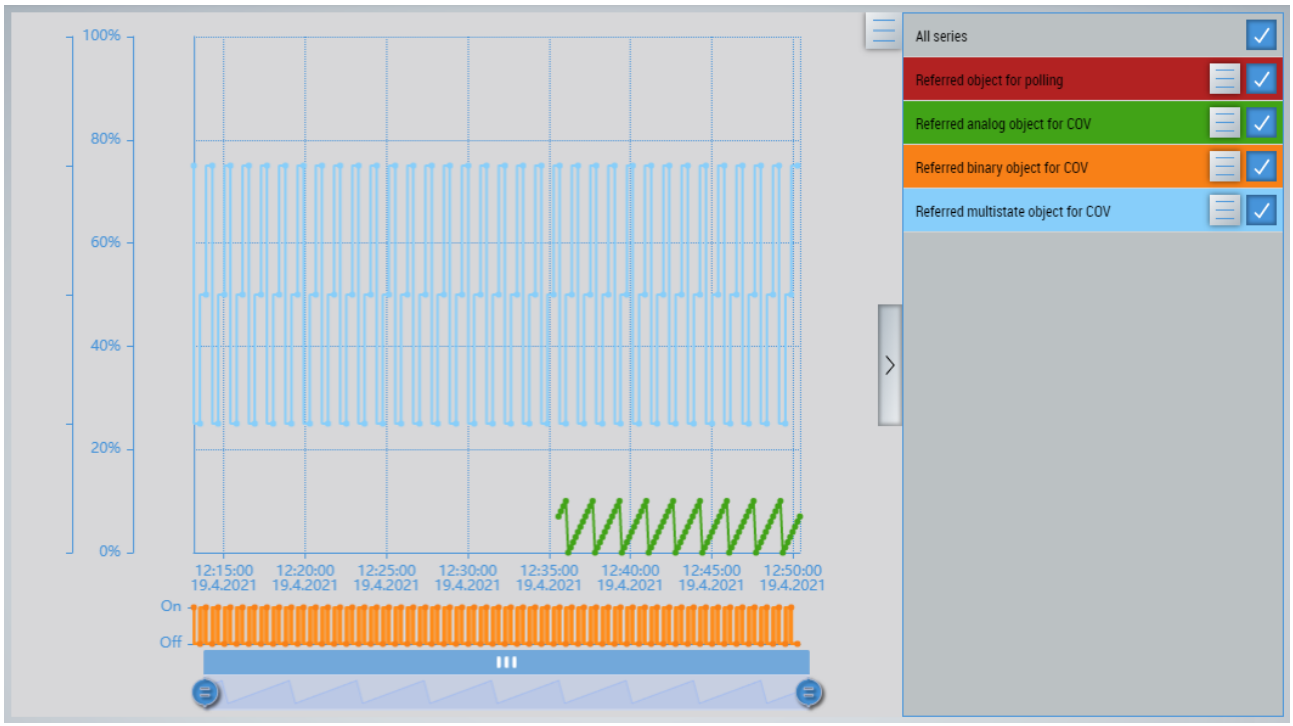
Digits

```
tchmi:general#/definitions/Number
```

Legt die Anzahl der Nachkommastellen fest, wenn ein analoger Zeitschaltplan verknüpft wurde.

6.2.1.2.3.4 Trend

Das **Trend**-Control kann mehrere Trendkurven anzeigen. Es ermöglicht die Auswahl von verschiedenen Trendkurven und das Ändern der Einstellungen aller Achsen.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Trend angezeigt werden soll. Erlaubt die Verknüpfung mit einem BaObject [▶ 1197] vom Typ Trendobjekt oder View.

Weitere Informationen sind in der Dokumentation zum Trending [▶ 56] zu finden.

Features

Mehrere Trendkurven

Wenn das BaObject ein **Trendobjekt** ist, dann wird nur die dazugehörige Trendkurve angezeigt. Eine Auswahl zwischen verschiedenen Trendkurven ist hierbei nicht möglich.

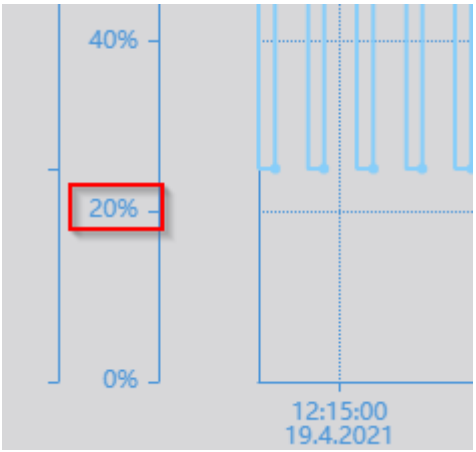
Handelt es sich beim BaObject um einen **View**, so wird dieser nach Trendobjekten durchsucht und vorhandene Trendkurven entsprechend angezeigt. Eine Auswahl zwischen verschiedenen Trendkurven ist bei mehr als zwei gefundenen Trendobjekten möglich.

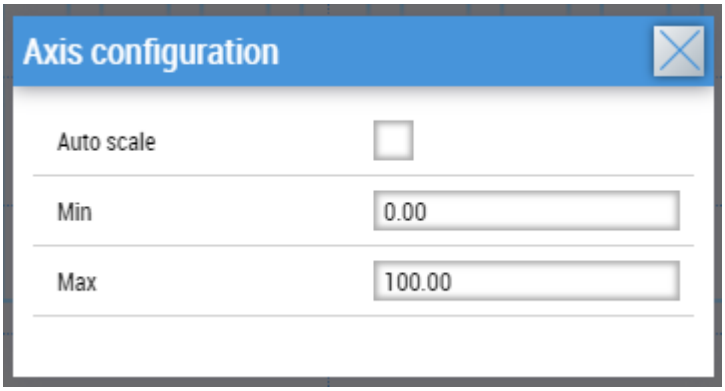


In der Auflistung können über die Checkboxen die anzuzeigenden Trendkurven im Chart ausgewählt werden. Der nebenstehende Button öffnet das Parameterfenster [▶ 1189] des jeweiligen Trendobjektes.

Achsenparametrierung

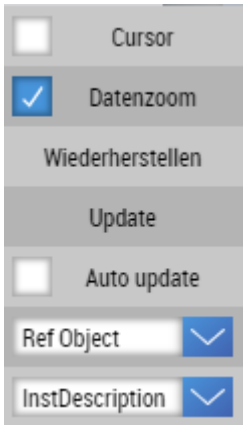
Die Einstellungen einer y-Achse lassen sich über die Anwahl der jeweiligen Skalenwerte öffnen.





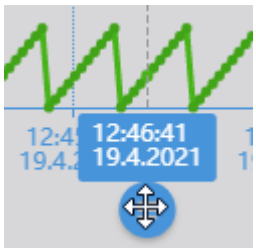
Menü

Das **Menü** erlaubt weitere Einstellungen für den Trend.



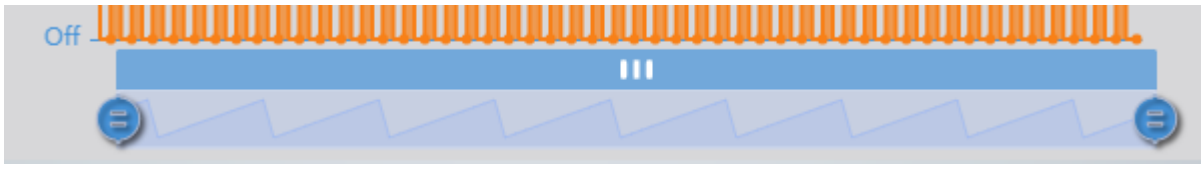
Cursor

Wenn **Cursor** aktiviert ist, wird ein Cursor unter der x-Achse eingeblendet. Standardmäßig ist diese Funktion deaktiviert.



Datenzoom

Über die **Checkbox** kann der Zoom ein- und ausgeblendet werden. Standardmäßig ist der Zoom eingeblendet.



Wiederherstellen

Stellt die Standardeinstellungen wieder her.

Update

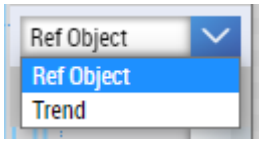
Die Trendkurven lassen sich einmalig aktualisieren.

Auto Update

Ist die Checkbox gesetzt, werden die Trendkurven automatisch aktualisiert, sobald neue Trendeinträge verfügbar sind.

Angezeigte Objekte

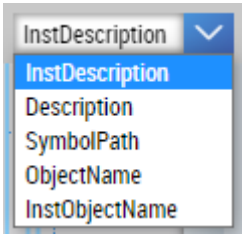
Bestimmt die anzuzeigenden Objekte in der Auflistung.



- RefObject: Zeigt aufgezeichnete Werte an.
- Trend: Zeigt alle Trendobjekte an, die einen Wert aufzeichnen.

Angezeigte Beschriftung

Auswahl der zu verwendenden Beschriftung in der Auflistung.



Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.

> Das Attribut ist nicht bei allen Controls anwendbar.

Common

ShowDataZoom

```
tchmi:general#/definitions/Boolean
```

Definiert, ob der Datenzoom angezeigt wird.

XAxesRange

```
tchmi:general#/definitions/Number
```

Der Zeitraum in Minuten, der auf der X-Achse angezeigt wird. Die Eingabe von 24 * 60 = 1440 zeigt die letzten 24 Stunden an.

SeriesSelectorWidth

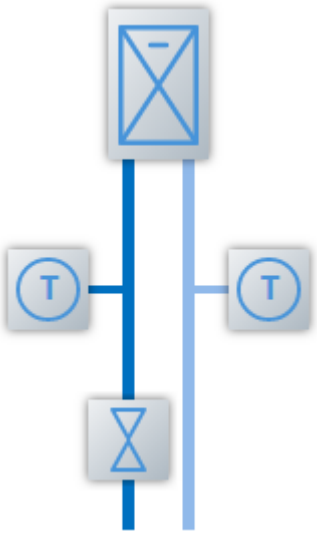
```
tchmi:general#/definitions/Number
```

Die Breite der Auswahl der angezeigten Trends in %, wenn mehrere Trends angezeigt werden.

6.2.1.2.4 Plants

6.2.1.2.4.1 Cooler

Das **Cooler**-Template stellt einen Kühler dar.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **FB_BA_ColT_02** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Beschreibung
TFl	FB_BA_SensorAnalog_IO	Vorlauftemperaturfühler
TRt	FB_BA_SensorAnalog_IO	Rücklauftemperaturfühler
Vlv	FB_BA_Vlv [▶ 1111]	Ventil

Hierarchie:

- BaObject
  - TFl
  - TRt
  - Vlv

Entspricht dem SPS-Template:

- FB_BA_AC_ColT_02

Attribute

Dieses Template erbt vom Control BaseTemplate [▶ 1214]. Dazu kommen folgende Attribute.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Cooler.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Flow Temperature Sensor

Setpoint

```
tchmi:general#/definitions/Boolean
```

Objekt für den Sollwert.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

Return Temperature Sensor

Setpoint

```
tchmi:general#/definitions/Boolean
```

Objekt für den Sollwert.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

Valve

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Valve.DisplayMode
```

Legt das anzuzeigende Icon fest.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

Colors

FlowPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe für die Vorlaufleitung.

ReturnPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe für die Rücklaufleitung.

6.2.1.2.4.2 Damper

Das **Damper**-Template zeigt die Klappenposition grafisch und in Textform an.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **Damper** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Two-point

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
SwiCls	FB_BA_BI_IO [▶ 210]	x	Switch close
SwiOpn	FB_BA_BI_IO [▶ 210]	x	Switch open

Hierarchie:

- BaObject
  - SwiCls
  - SwiOpn

Entspricht dem SPS-Template:

- FB_BA_Dmp2P [▶ 1077]

Analog

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
MdIt	FB_BA_AO_IO [▶ 202]	x	Feedback

Hierarchie:

- BaObject
  - MdIt

Entspricht dem SPS-Template

- FB_BA_ActuatorAnalog [▶ 1000]

Attribute

Dieses Template erbt vom Control UiIconFdbStp [▶ 1218]. Dazu kommen folgende Attribute.

Common

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Templates.Universal.Damper.DisplayMode
```

Legt das anzuzeigende Icon fest.

Wenn "Custom" gewählt wurde, wird das Icon [▶ 1149] angezeigt, welches im Icon-Attribut gesetzt wurde.

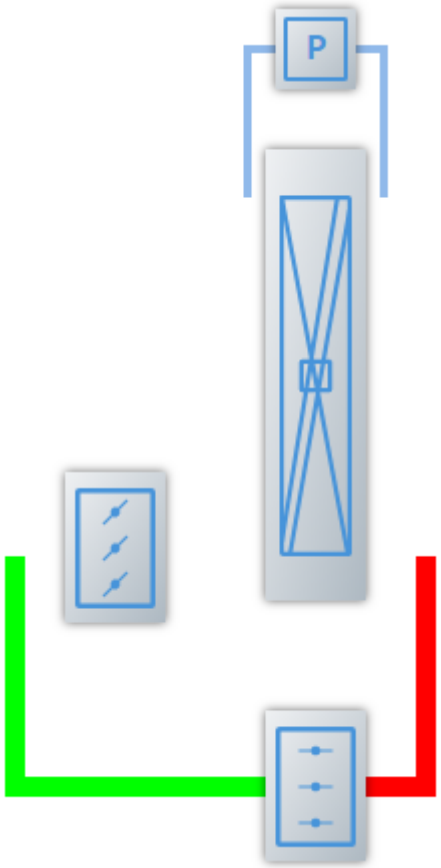
FlapPosition

```
tchmi:general#/definitions/Number
```

Position der Klappen in Prozent (0 ist geschlossen).

6.2.1.2.4.3 ErcPlate

Das **ErcPlate**-Template stellt eine Energierückgewinnung mit einem Plattenwärmetauscher dar.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **ErcPl_02** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Beschreibung
ByDmp	FB_BA_ActuatorAnalog [▶ 1000]	Bypass Klappe
DiffPrssSwi	FB_BA_SensorBinary_IO	Differenzdruck
Dmp	FB_BA_ActuatorAnalog [▶ 1000]	Klappe

Hierarchie:

- BaObject
    - ByDmp

- DiffPrssSwi

- Dmp

Entspricht dem SPS-Template:

- FB_BA_AC_ErcPl_02 [▶ 746]

Attribute

Dieses Template erbt vom Control BaseTemplate [▶ 1214].

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.ErcPlate.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Damper

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

Damper Bypass

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

Colors

FlowPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe für die Vorlaufleitung.

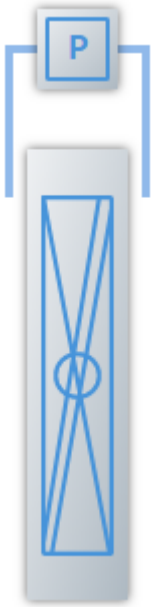
ReturnPipeColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe für die Rücklaufleitung.

6.2.1.2.4.4 ErcRotation

Das **ErcRotation**-Template stellt eine Energierückgewinnung mit einem Rotationswärmetauscher dar.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **ErcRot_01** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Beschreibung
DiffPrssSwi	FB_BA_SensorBinary_IO	Differenzdruck
MdIt	FB_BA_AO_IO [▶ 202]	Rückmeldung Motor
Motor	FB_BA_MotMdlt [▶ 1089]	Motor

Hierarchie:

- BaObject
  - DiffPrssSwi
  - Motor
    - MdIt

Entspricht dem SPS-Template:

- FB_BA_AC_ErcRot_01 [▶ 751]

Attribute

Dieses Template erbt vom Control BaseTemplate [▶ 1214].

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.ErcRotation.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Common

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

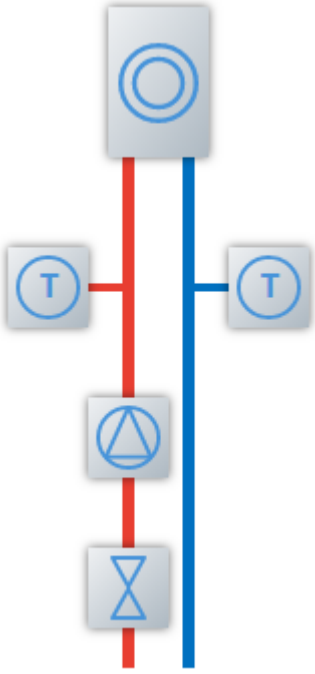
DisplayPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.UiIcon.DisplayPosition
```

Position vom Ist- und Sollwert.

6.2.1.2.4.5 HeatingCircuit

Das **HeatingCircuit**-Template stellt einen Heizkreis dar.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom **Typ HtgCir01** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Beschreibung
Pu	FB_BA_Pu1st [▶ 1093]	Pumpe
Sp	FB_BA_H_HtgCir_Sp	Sollwert
SpFlWT	FB_BA_AV_Op [▶ 207]	Sollwert Vorlauftemperatursensor
TFl	FB_BA_SensorAnalog_IO	Vorlauftemperatursensor
TRt	FB_BA_SensorAnalog_IO	Rücklauftemperatursensor
Vlv	FB_BA_Vlv [▶ 1111]	Ventil

Hierarchie:

- BaObject
  - Pu
  - Sp
    - SpFlWT
  - TFl
  - TRt
  - Vlv

Entspricht dem SPS-Template:

- FB_BA_H_HtgCir01 [▶ 993]

Attribute

Dieses Template erbt von dem Template FB_BA_AC_ColT_02 [▶ 1197].

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.HeatingCircuit.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Flow Temperature Sensor

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

Pump

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

6.2.1.2.4.6 Motor

Das **Motor**-Template zeigt den Status eines Motors grafisch an und, sofern vorhanden, die Rückmeldung in Textform.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **Motor** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
Cmd	FB_BA_BO_IO [▶ 214]	x	Befehl
MdIt	FB_BA_BO_IO [▶ 214]		Feedback

Hierarchie:

- BaObject
  - Cmd
  - MdIt

Entspricht den SPS-Templates

- FB_BA_MotCtl [▶ 1085]
- FB_BA_MotCtlExt [▶ 1088]
- FB_BA_Pu1st [▶ 1093]
- FB_BA_Pu1stExt [▶ 1095]
- FB_BA_PuCtl [▶ 1096]

Attribute

Dieses Template erbt vom Control UiIconFdbStp [▶ 1218]. Dazu kommen folgende Attribute.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Motor.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Common

DisplayMode

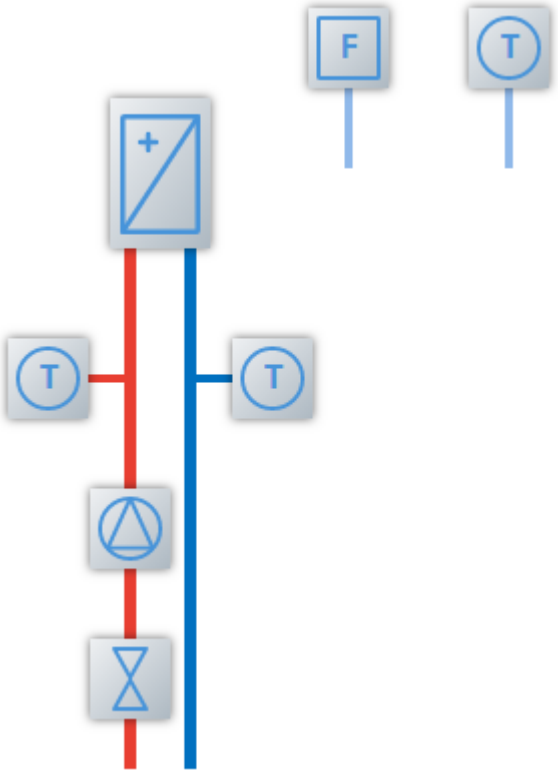
```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Motor.DisplayMode
```

Legt das anzuzeigende Icon fest. Wenn "Custom" gewählt wurde, wird das Icon angezeigt, dass im Icon [▶ 1149]-Attribut gesetzt wurde.

Wenn "Custom" gewählt wurde, wird das Icon [▶ 1149] angezeigt, welches im Icon-Attribut gesetzt wurde.

6.2.1.2.4.7 PreHeater

Das **PreHeater**-Template stellt einen Vorerhitzer dar.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **PreHtr** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Beschreibung
FrostThermostat	FB_BA_SensorBinary_IO	Frostschutzthermostat
Pu	FB_BA_Pu1st [▶ 1093]	Pumpe
Sp	FB_BA_H_HtgCir_Sp	Sollwert
SpFlWT	FB_BA_AV_Op [▶ 207]	Sollwert Vorlauftemperaturfühler
TFl	FB_BA_SensorAnalog_IO	Vorlauftemperaturfühler
TFrost	FB_BA_AO_IO [▶ 202]	Temperatur Frost
TRt	FB_BA_SensorAnalog_IO	Rücklauftemperatursensor
Vlv	FB_BA_Vlv [▶ 1111]	Ventil

Hierarchie:

- BaObject
    - FrostThermostat
    - Pu
    - Sp
        - SpFlWT

- TFl
- TFrost
- TRt
- Vlv

Entspricht dem SPS-Template:

- FB_BA_AC_PreHtr [▶ 761]

Attribute

Dieses Template erbt von dem Template HeatingCircuit [▶ 1204].

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.PreHeater.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Frost Temperature Sensor

Setpoint

```
tchmi:general#/definitions/Boolean
```

Objekt für den Sollwert.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

6.2.1.2.4.8 Pump

Das **Pump**-Template zeigt den Status einer Pumpe grafisch an und, sofern vorhanden, die Rückmeldung in Textform.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **Pumpe** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
Cmd	FB_BA_BO_IO [▶ 214]	x	Befehl
MdIt	FB_BA_BO_IO [▶ 214]		Feedback

Hierarchie:

- BaObject
    - Cmd
    - MdIt

Entspricht dem SPS-Template

- FB_BA_MotCtl [▶ 1085]
- FB_BA_MotCtlExt [▶ 1088]
- FB_BA_Pu1st [▶ 1093]
- FB_BA_Pu1stExt [▶ 1095]
- FB_BA_PuCtl [▶ 1096]

Attribute

Dieses Template erbt vom Control UiIconFdbStp [▶ 1218]. Dazu kommen folgende Attribute.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Pump.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Common

DisplayMode

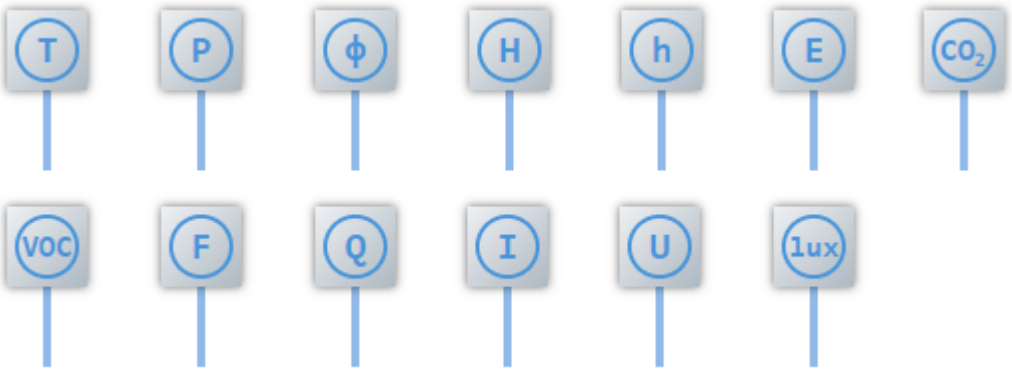
```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Pump.DisplayMode
```

Legt das anzuzeigende Icon fest.

Wenn "Custom" gewählt wurde, wird das Icon [▶ 1149] angezeigt, welches im Icon-Attribut gesetzt wurde.

6.2.1.2.4.9 SensorAnalog

Das **SensorAnalog**-Template zeigt die Rückmeldung eines analogen Wertes an und, sofern verknüpft, einen Sollwert.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **SensorAnalog** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
MV	FB_BA_AI_Raw [▶ 200]	x	Feedback

Hierarchie:

- BaObject

  - MV

Entspricht dem SPS-Template

- FB_BA_SensorAnalog [▶ 1106]

Attribute

Dieses Template erbt vom Control Sensor [▶ 1215]. Dazu kommen folgende Attribute.

BA

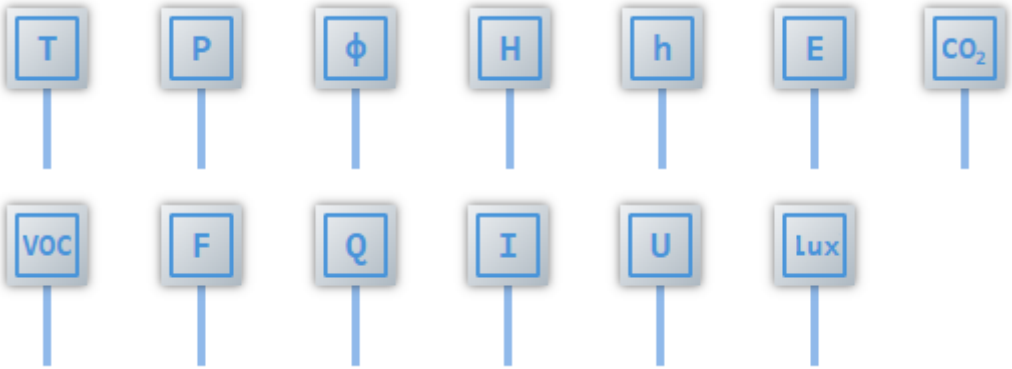
BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorAnalog.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

6.2.1.2.4.10 SensorBinary

Das **SensorBinary**-Template zeigt die Rückmeldung eines binären Wertes an und, sofern verknüpft, einen Sollwert.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **SensorBinary** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
Input	FB_BA_BI_Raw [▶ 211]	x	Value

Hierarchie:

- BaObject
    - Input

Entspricht dem SPS-Template:

- FB_BA_SensorBinary [▶ 1107]

Attribute

Dieses Template erbt vom Control Sensor [▶ 1215]. Dazu kommen folgende Attribute.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.Plants.SensorBinary.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

6.2.1.2.4.11 Valve

Das **Valve**-Template zeigt die Rückmeldung eines Ventils an und wenn vorhanden, den Sollwert.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der eine Template vom Typ **Valve** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Three-point

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
Anlg3Pnt	FB_BA_Analog3Pnt [▶ 1013]	x	
Pos	FB_BA_AV_Op [▶ 207]	x	Feedback

Hierarchie:

- BaObject
    - Anlg3Pnt
        - Pos

Entspricht dem SPS-Template

- FB_BA_Vlv3pt [▶ 1113]

Analog value

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
Fdb	FB_BA_AI_IO [▶ 197]	x	Feedback
Mdlt	FB_BA_AO_IO [▶ 202]		Sollwert

Hierarchie:

- BaObject
  - Fdb
  - Mdlt

Entspricht dem SPS-Template

- FB_BA_Vlv [▶ 1111]

Attribute

Dieses Template erbt vom Control UiIconFdbStp [▶ 1218]. Dazu kommen folgende Attribute.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Valve.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Common

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.Valve.DisplayMode
```

Legt das anzuzeigende Icon fest.

Wenn "Custom" gewählt wurde, wird das Icon [▶ 1149] angezeigt, welches im Icon-Attribut gesetzt wurde.

6.2.1.2.4.12 VAV

Das **VAV**-Template zeigt den Sollwert von einem Volumenstromregler und wenn vorhanden, den Sollwert.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der ein Template vom Typ **VAV** angezeigt werden soll.

Kompatibilität

Die BaTemplateDescription [▶ 63] unterstützt folgende *BaObjects*.

Unterelemente:

Symbolname	SPS-Template	Erforderlich	Beschreibung
MdIt	FB_BA_AO_IO [▶ 202]		Setpoint
Fdb	FB_BA_AI_IO [▶ 197]	x	Feedback

Hierarchie:

- BaObject
  - MdIt
  - Fdb

Entspricht dem SPS-Template:

- FB_BA_ActuatorAnalog [▶ 1000]

Attribute

Dieses Template erbt vom Control UiIconFdbStp [▶ 1218]. Dazu kommen folgende Attribute.

BA

BaTemplateDescription

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.VAV.BaTemplateDescription
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Common

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Plants.VAV.DisplayMode
```

Legt das anzuzeigende Icon fest.

Wenn "Custom" gewählt wurde, wird das Icon [▶ 1149] angezeigt, welches im Icon-Attribut gesetzt wurde.

6.2.1.2.5 System

6.2.1.2.5.1 BaseControl

Das BaseControl ist die Basis für verschiedene Controls, es beinhaltet Methoden und Attribute, die andere Controls ebenfalls benötigen. Somit werden redundante Implementierungen verhindert.

Verwendung

Dient nur der Vererbung und steht deshalb nicht in der Toolbox zur Verfügung.

Features

Realisiert Funktionalitäten, die im Hintergrund ablaufen und das Management von verschiedenen Aufgaben übernehmen. Hierzu zählen z. B.:

- Busy handling
- abmelden verschiedener Watches

Attribute

Das Control erbt vom TcHmiControl und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Common

ReadOnly

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Benutzer nur leseberechtigt oder auch schreibberechtigt ist.



Das Attribut ist nicht bei allen Controls anwendbar.

Layout

ContentPadding

```
tchmi:framework#/definitions/Padding
```

Legt das Padding für den Inhalt des Controls fest.



Das Attribut ist nicht bei allen Controls anwendbar.

6.2.1.2.5.2 BaseTemplate

Das **BaseTemplate** ist die Basis für alle komplexeren Template-Controls (z. B. PreHtr, HtgCir), die mehr als nur ein UiIcon [▶ 1178] sind. Es stellt Methoden und Attribute zur Verfügung, um redundante Implementierungen zu verhindern.

Verwendung

Dient nur der Vererbung und steht deshalb nicht in der Toolbox zur Verfügung.

Features

Ermöglicht die Verwendung von BaTemplates [▶ 63] für alle erbenden Controls.

Attribute

Das Control erbt vom TcHmiControl und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

BaTemplateDescription

```
tchmi:general#/definitions/Object
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

ShowTags

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Tags angezeigt werden oder nicht.

6.2.1.2.5.3 Sensor

Der Sensor ist eine spezialisierte Form des UiIconFdbStp [▶ 1218] und dient als Basis für Sensor-Controls (z. B. SensorAnalog). Er stellt Methoden und Attribute zur Verfügung, um redundante Implementierungen zu verhindern.

Verwendung

Dient nur der Vererbung und steht deshalb nicht in der Toolbox zur Verfügung.

Features

Erleichtert den Umgang mit den diversen Darstellungsmöglichkeiten von Sensoren.

Attribute

Das Control erbt vom UiIconFdbStp [▶ 1218] und hat somit dieselben Attribute. Hinzu kommen folgende Attribute.

Common

PinLayout

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.System.Sensor.PinLayout
```

Legt das Layout der Pins fest.

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.System.Sensor.DisplayMode
```

Legt das anzuzeigende Icon fest.

Wenn "Custom" gewählt wurde, wird das Icon [▶ 1149] angezeigt, welches im Icon-Attribut gesetzt wurde.

PinPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position
```

Legt die Orientierung der Pins fest.

PinConnection

```
tchmi:general#/definitions/Number
```

Legt die Strichlänge vom ersten Pin fest.

PinExtension

```
tchmi:general#/definitions/Number
```

Legt die Strichlänge ab dem ersten Pin fest.

PinWidth

```
tchmi:general#/definitions/Number
```

Legt die Strichstärke der Pins fest.

CustomLetter

```
tchmi:general#/definitions/String
```

Legt den anzuzeigenden Buchstaben im Icon fest, wenn das DisplayMode [▶ 1149]-Attribut den Wert "Custom" hat.

ShowValue

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Wert zu sehen ist.

6.2.1.2.5.4 BaseRoomControl

Das **BaseRoomControl** ist die Basis für alle Raum-Controls (z. B. Light [▶ 1222], Sunblind [▶ 1230], HeatingCooling [▶ 1219], Window [▶ 1235]). Es stellt Methoden und Attribute zur Verfügung, um redundante Implementierungen zu verhindern.

Verwendung

Dient nur der Vererbung und steht deshalb nicht in der Toolbox zur Verfügung.

Features

Ermöglicht die Verwendung von BaTemplates [▶ 63] für alle erbenden Controls.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

BA

BaInterfaceSymbolNames

```
tchmi:general#/definitions/Object
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

BaData

BaInterface

```
tchmi:framework#/definitions/Symbol
```

Symbol das das BaInterface [▶ 64] des Controls erfüllt.

Error

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.RoomAutomation.HeatingCooling.EnergyLevel
```

Wenn TRUE, dann wird eine Fehlermeldung angezeigt.



ShowPriority

```
tchmi:general#/definitions/Boolean
```

Definiert, ob die Priorität angezeigt wird.

6.2.1.2.5.5 TextControl

Das **TextControl** bietet verschiedene Attribute, die alle der Textmanipulation gelten.

Verwendung

Dient nur der Vererbung und steht deshalb nicht in der Toolbox zur Verfügung.

Features

Folgende Textmanipulationen sind möglich:

- Position horizontal und vertikal verändern
- Schriftart, -größe und -stärke beeinflussen
- verschiedene Dekorationen dem Text hinzufügen (z.B. unterstrichen)
- festlegen, wie der Text dargestellt werden soll, wenn der verfügbare Platz nicht ausreicht

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Colors

TextColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe von Texten.

TextDecorationColor

```
tchmi:framework#/definitions/SolidColor
```

Farbe von Textdekorationen.

Text

TextVerticalAlignment

```
tchmi:framework#/definitions/VerticalAlignment
```

Vertikale Ausrichtung von Texten.

TextHorizontalAlignment

```
tchmi:framework#/definitions/HorizontalAlignment
```

Horizontale Ausrichtung von Texten.

TextFontSize

```
tchmi:framework#/definitions/MeasurementValue
```

Schriftgröße von Texten. Prozentwerte sind relativ zu der Schriftgröße des Elternelements.

TextFontSizeUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Einheit für die Schriftgröße von Texten. Kann absolut (px) oder relativ (%) sein.

TextFontFamily

```
tchmi:framework#/definitions/FontFamily
```

Schriftart von Texten.

TextFontStyle

```
tchmi:framework#/definitions/FontStyle
```

Schriftstil von Texten.

TextFontWeight

```
tchmi:framework#/definitions/FontWeight
```

Schriftstärke von Texten.

TextDecorationLine

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationLine
```

Position der Textdekoration.

TextDecorationStyle

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextDecorationStyle
```

Stil der Textdekoration.

UserSelect

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.UserSelect
```

Verhalten bei der Auswahl des Textes von einem Benutzer.

TextOverflow

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.TextControl.TextOverflow
```

Definiert, wie ein Text dargestellt werden soll, der breiter als das Control ist.

6.2.1.2.5.6 UiIconFdbStp

Das UiIconFdbStp ist eine spezialisierte Form des UiIcon [▶ 1178] und dient als Basis für simplere TcHmiBa-Controls (z. B. Valve, SensorAnalog). Es stellt Methoden und Attribute zur Verfügung, um redundante Implementierungen zu verhindern.

Verwendung

Dient nur der Vererbung und steht deshalb nicht in der Toolbox zur Verfügung.

Features

Erleichtert den Umgang mit dem Soll- und Istwert.

Attribute

Das Control erbt vom UiIcon [▶ 1178] und hat somit dieselben Attribute. Hinzu kommen folgende Attribute.

BA

```
tchmi:general#/definitions/Object
```

Bearbeiten der *BaTemplateDescription* vom BaTemplate [▶ 63].

Common

DisplayMode

```
tchmi:general#/definitions/Number
```

Legt das anzuzeigende Icon fest.

Wenn "Custom" gewählt wurde, wird das Icon [▶ 1149] angezeigt, welches im Icon-Attribut gesetzt wurde.

ShowFeedback

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob die Rückmeldung zu sehen ist.

ShowSetpoint

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Sollwert zu sehen ist.

ShowDisplays

`tchmi:general#/definitions/Boolean`

Legt fest, ob der Ist- und Sollwert zu sehen sind.

DisplayPosition

`tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Common.UiIcon.DisplayPosition`

Position vom Ist- und Sollwert.

6.2.1.2.6 RoomAutomation

6.2.1.2.6.1 HeatingCooling

Das **HeatingCooling**-Control zeigt die Betriebsart, die Soll- und Ist-Temperatur in einem Raum an und kann den Sollwert verändern.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der Controls für die Steuerung von Klimaanlagen benötigt werden.

Features

Die folgende Tabelle zeigt die möglichen Betriebsarten.

Betriebsart	Symbol
Heizen	
Kühlen	
Inaktiv	

Ist die Anlage inaktiv, befindet sich die Regelung in der neutralen Zone.

Anzeigen

Der Benutzerlevel bestimmt die im Display verfügbaren Informationen.
Für Benutzer ab dem Level *Advanced* sind die aktuellen Werte für Temperatur und Sollwert einsehbar.



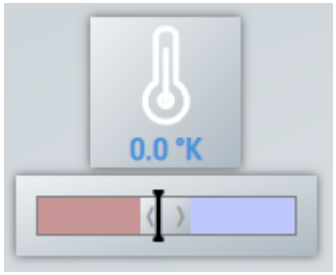
Für Benutzer bis zum Level Basic wird nur die aktuelle Sollwertanpassung angezeigt.



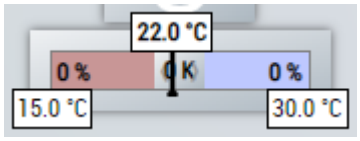
Bedienung

Das **HeatingCooling**-Control zeigt den aktuellen Zustand der Anlage an und öffnet durch einen Klick die Aktivitätsanzeige zum Einstellen der Raumtemperatur.

Durch das Verschieben des Sliders werden die Temperatursollwerte angepasst. Der Cursor im Slider stellt durch seine Position die Raumtemperatur dar.

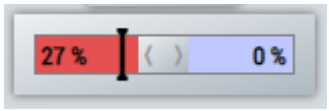


Ab dem Benutzerlevel „Advanced“ sind die Raumtemperatur und Grenzen sichtbar.

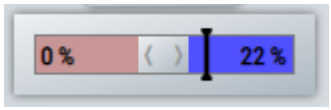


Zusätzlich stellt die Aktivitätsanzeige die Betriebsart dar. Sie kann binär oder prozentual ausgelegt sein.

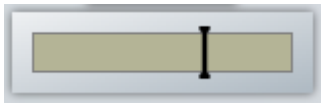
Heizen:



Kühlen:



Das Bedienelement ist nur aktiv, wenn das aktive Energy Level „Comfort“ entspricht.



Attribute

Das Control erbt vom BaseRoomControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



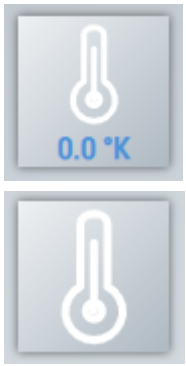
Das Attribut ist nicht bei allen Controls anwendbar.

Common

ShowTemperatures

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, werden die Temperaturen angezeigt.



BaData

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Controls.RoomAutomation.HeatingCooling.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

Temperature

RoomTemp

```
tchmi:general#/definitions/Number
```

Aktuelle gemessene Raumtemperatur.

RoomTempAdjust

```
tchmi:general#/definitions/Number
```

Aktuelle Raumtemperaturanpassung.

RoomTempAdjustFeedback

```
tchmi:general#/definitions/Number
```

Feedback für die aktuelle Raumtemperaturanpassung.

RoomTempAdjustRange

```
tchmi:general#/definitions/Number
```

Legt den Umfang der Raumtemperaturanpassung fest.

HeatingSetpoint

```
tchmi:general#/definitions/Number
```

Aktueller Sollwert für den Heizbetrieb.

CoolingSetpoint

```
tchmi:general#/definitions/Number
```

Aktueller Sollwert für den Kühlbetrieb.

HeatingActive

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Wenn TRUE oder größer 0, dann wird heizen angezeigt.

CoolingActive

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Wenn TRUE oder größer 0, dann wird kühlen angezeigt.

Unit

```
tchmi:general#/definitions/String
```

Einheit für die Anzeige der Temperaturen.

Events

Ereignis	Beschreibung
onTempAdjustChanged	Wird ausgelöst, wenn der Benutzer die Temperaturanpassung verändert.

6.2.1.2.6.2 Light

Das **Light**-Control dient zum Anzeigen und Steuern der Helligkeit einer Lichtquelle.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der Controls zur Steuerung von Lichtquellen benötigt werden.

Features

Prioritäten

Anzeige verschiedener Prioritäten über das Attribut *Priority*.

Helligkeitsvorgabe

Die Helligkeit lässt sich anhand eines analogen (dimmbar) oder binären (schaltbar) Wertes einstellen.
Für dimmbare Lampen sind Buttons mit vordefinierten Helligkeitswerten vorhanden.

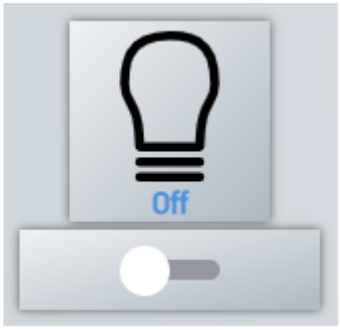


Die Anzeige dieser Buttons kann global eingestellt [▶ 1288] werden:

Bedienung

Ein Klick auf das **Light-**Control ändert die Sichtbarkeit des Menüs zum Einstellen der Helligkeit.





Attribute

Das Control erbt vom BaseRoomControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



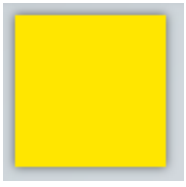
Das Attribut ist nicht bei allen Controls anwendbar.

Common

DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.DisplayMode
```

Legt die Darstellungsart der Lampe fest.

Name	Darstellung
lightBulb	
lightBulbFilled	
filles	

ShowValue

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann wird der Helligkeitswert angezeigt.



BaData

Priority

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority
```

Die aktuelle Priorität, die am Control angezeigt wird. Hohe Prioritäten (niedriger Wert in Priority-Enumeration) deaktivieren die Bedienmöglichkeiten des Controls.

Priorität	Darstellung
fire	
communicationError	
burglaryAlarm	
maintenance	
cleaning	
nightWatch	
simple	
manual	
automaticLight	
scene1…scene3	

Die verwendeten Icons für die verschiedenen Prioritäten können global [▶ 1288] eingestellt werden.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Light.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

Brightness

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Aktueller Helligkeitswert (0-100%).

BrightnessFeedback

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Feedback für den Helligkeitswert (0-100%).

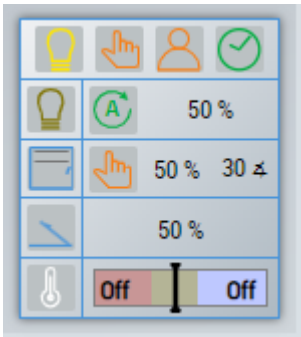
Events

Ereignis	Beschreibung
onBrightnessChanged	Wird ausgelöst, wenn der Benutzer die Helligkeit verändert.
onHandModeReset	Wird ausgelöst, wenn die Schaltfläche zum Zurücksetzen vom Hand- in den Automatikmodus betätigt wird.

6.2.1.2.6.3 RoomControl

Das **RoomControl** kann die verschiedenen Controls der Raumautomation zusammenfassen. Verfügbare Komponenten sind:

- HeatingCooling [▶ 1219]
- Light [▶ 1222]
- Sunblind [▶ 1230]
- Window [▶ 1235]



Verwendung

Zur Automation eines Raumes oder Bereiches mit nur einem Control kann das **RoomControl** verwendet werden. Es ist möglich, z. B. nur einen Bereich mit Lampen zusammenzufassen, damit die Übersicht in der Visualisierung erhalten bleibt.

Features

Bedienmöglichkeiten und Einstellungen entsprechen denen der einzelnen Controls.

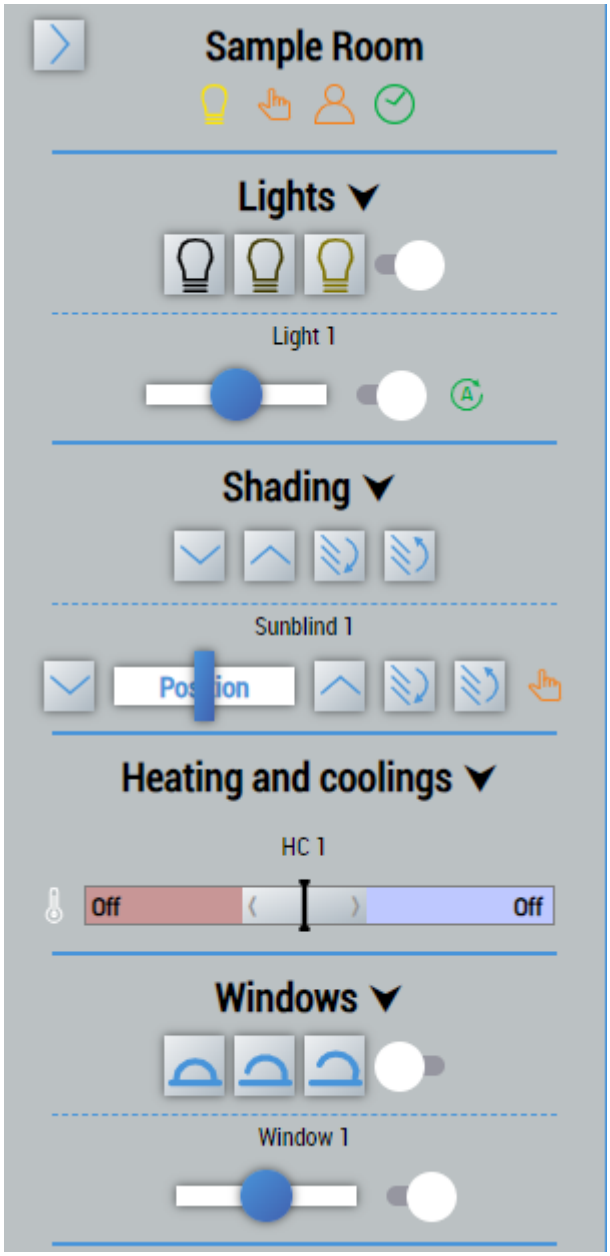
Raumstatus

In der Kopfzeile des Controls werden die allgemeinen Rauminformationen angezeigt:

- Licht an oder aus
- Modus oder Meldung mit der höchsten Priorität
- Präsenz
- Nachlaufzeit

Seitenmenü zur Steuerung

Ein Klick auf das Control öffnet ein Seitenmenü, in dem sich die Controls für die einzelnen Bereiche befinden.



Zentrale Steuerung

Jeder Bereich enthält eine Kopfzeile mit Schnelleinstellung zum einheitlichen Ansteuern aller Controls.

Attribute

Das Control erbt vom BaseControl [▶ 1213] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

ControlUnits

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomControl.ControlList
```

Legt fest, welche Komponenten (HeatingCooling [▶ 1219], Light [▶ 1222], Sunblind [▶ 1230], Window [▶ 1235]) hinzugefügt werden sollen.

Name

```
tchmi:general#/definitions/String
```

Der Raumname.

HideRoomStatus

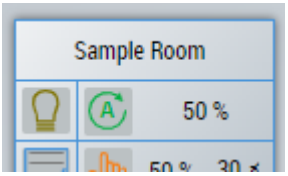
```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann sind die Rauminformationen (Kopfzeile) nicht sichtbar.

ShowRoomName

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann wird anstelle der Rauminformationen der Raumname [▶ 1228] angezeigt.



BaData

Presence

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann wurde Präsenz im Raum erkannt.

- Präsenz aktiv



- Präsenz inaktiv



SwitchOffDelayActive

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann ist die Nachlaufzeit für die Automatik aktiv.

- Nachlaufzeit aktiv



- Nachlaufzeit inaktiv



Lights

ShowLights

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann werden die Lights im Control angezeigt. Im Seitenmenü sind die Komponenten immer zu sehen.

Weitere Informationen zu den Attributen finden Sie in der Dokumentation vom Light [▶ 1222]-Control.

Sunblinds

ShowSunblinds

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann werden die Sunblinds im Control angezeigt. Im Seitenmenü sind die Komponenten immer zu sehen.

Weitere Informationen zu den Attributen finden Sie in der Dokumentation vom Sunblind [▶ 1230]-Control-Control.

HeatingCooling

ShowHeatingCooling

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann werden die HeatingCooling-Anwendungen im Control angezeigt. Im Seitenmenü sind die Komponenten immer zu sehen.

Weitere Informationen zu den Attributen finden Sie in der Dokumentation vom HeatingCooling [▶ 1219]-Control

Windows

ShowWindows

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann werden die Windows im Control angezeigt. Im Seitenmenü sind die Komponenten immer zu sehen.

Weitere Informationen zu den Attributen finden Sie in der Dokumentation vom Window [▶ 1235]-Control.

6.2.1.2.6.4 Sunblind

Das **Sunblind**-Control kann die Position und den Winkel von Verschattungen anzeigen und steuern.



Verwendung

Einsatz auf jeder beliebigen Seite, wo Controls für die Steuerung von Verschattungen benötigt werden.

Features

Prioritäten

Anzeige verschiedener Prioritäten über das Attribut *Priority*.

Winkeleinstellung

Die Winkeleinstellung ist optional und lässt sich nur nutzen, wenn die Verschattung es unterstützt. Diese Funktionalität wird über das Attribut *UseAngle* aktiviert bzw. deaktiviert.

Bedienung

Ein Klick auf das **Sunblind**-Control ändert die Sichtbarkeit des Menüs zum Einstellen der Position bzw. des Winkels.



Attribute

Das Control erbt vom BaseRoomControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



Das Attribut ist nicht bei allen Controls anwendbar.

Common

Controls

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Controls
```

Legt die Art von zu verwendenden Controls für die Positions- und Winkeleinstellungen fest.

- sliderHorizontal



- buttons



ShowValue

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann werden die Werte für Winkel und Position angezeigt.



BaData

Priority

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority
```

Die aktuelle Priorität, die am Control angezeigt wird. Hohe Prioritäten (niedriger Wert in Priority-Enumeration) deaktivieren die Bedienmöglichkeiten des Controls.

Name	Darstellung
fire	
storm	
ice	
communicationError	
burglaryAlarm	
maintenance	
referencing	
manualActuator	
manualGroup	
allDown	
allUp	
scene1…scene3	

Name	Darstellung
facadeThermoAutomatic	
facadeTwilightAutomatic	
parkPosition	
sunProtection	
groupThermoAutomatic	
groupTwilightAutomatic	

Die verwendeten Icons für die verschiedenen Prioritäten können global [▶ 1289] eingestellt werden.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Sunblind.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

Position

```
tchmi:general#/definitions/Number
```

Aktuelle Position (0-100%).

PositionFeedback

```
tchmi:general#/definitions/Number
```

Feedback für die Position (0-100%).

Angle

UseAngle

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann werden die Controls für die Steuerung des Winkels angezeigt.

Angle

```
tchmi:general#/definitions/Number
```

Aktueller Winkel (0-100%).

AngleFeedback

```
tchmi:general#/definitions/Number
```

Feedback für den Winkel (0-100%).

AngleStep

```
tchmi:general#/definitions/Number
```

Legt die Schrittweite fest, mit der der Winkel über die Winkelschaltflächen verstellt wird.

Events

Ereignis	Beschreibung
onPositionChanged	Wird ausgelöst, wenn der Benutzer die Position der Verschattung verändert.
onAngleChanged	Wird ausgelöst, wenn der Benutzer den Winkel der Verschattung verändert.
onManualReset	Wird ausgelöst, wenn die Schaltfläche zum Zurücksetzen vom Hand- in den Automatikmodus betätigt wird.

6.2.1.2.6.5 Window

Das **Window**-Control kann die Position von Fenstern oder Dachkuppeln mit Antrieb anzeigen und steuern.



Verwendung

Einsatz auf jeder beliebigen Seite, auf der Controls für die Steuerung von Fenstern benötigt werden.

Features

Positionsvorgabe

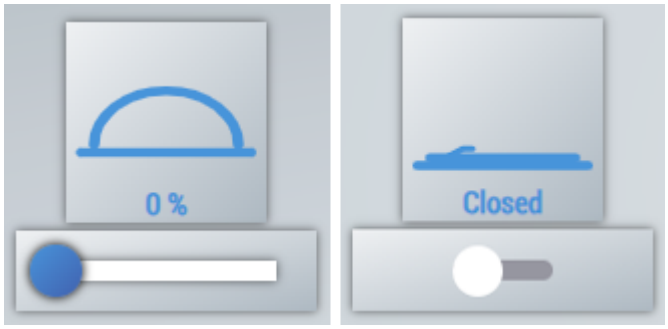
Die Position des Fensters kann über einen analogen oder binären Wert vorgegeben werden, wenn der Antrieb es unterstützt.
Für analoge Werte können Buttons mit vordefinierten Positionswerten angezeigt werden.



Die Anzeige dieser Buttons kann global [▶ 1290] eingestellt werden.

Bedienung

Ein Klick auf das **Window**-Control ändert die Sichtbarkeit des Menüs zum Einstellen der Position.



Attribute

Das Control erbt vom BaseRoomControl [▶ 1216] und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Feedback-Konzept

Das Control kann das Feedback-Konzept [▶ 62] nutzen.

BA

BaObject

```
tchmi:framework#/definitions/Symbol
```

Symbol für BaObject [▶ 62] zum Verwenden der generischen Funktionalitäten [▶ 79] von TcHmiBa. Verknüpft ein einzelnes Objekt oder einen kompletten View (inkl. Kinder) mit dem Control.



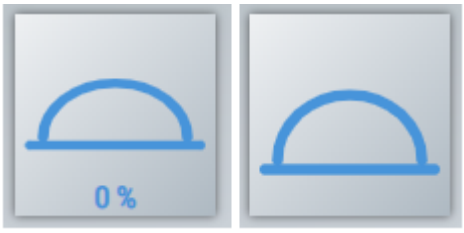
Das Attribut ist nicht bei allen Controls anwendbar.

Common

ShowValue

```
tchmi:general#/definitions/Boolean
```

Wenn TRUE, dann wird der Wert für die Position angezeigt.



DisplayMode

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Window.DisplayMode
```

Legt die Darstellungsart des Fensters fest.

Name	Darstellung
roofDome	
window	

IconRotation

```
tchmi:general#/definitions/Number
```

Rotiert das Icon. Winkelangabe in Grad.

BaData

Priority

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.RoomAutomation.Window.Priority
```

Die aktuelle Priorität, die am Control angezeigt wird. Hohe Prioritäten (niedriger Wert in Priority-Enumeration) deaktivieren die Bedienmöglichkeiten des Controls.

Name	Darstellung
fire	
storm	
ice	
communicationError	
burglaryAlarm	
maintanence	
referencing	
manualActuator	
manualGroup	
allDown	
allUp	
scene1…scene3	

Name	Darstellung
facadeThermoAutomatic	
facadeTwilightAutomatic	
parkPosition	
sunProtection	
groupThermoAutomatic	
groupTwilightAutomatic	

Die verwendeten Icons für die verschiedenen Prioritäten können global [▶ 1290] eingestellt werden.

BaInterfaceSymbolNames

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.Window.BaInterfaceSymbolNames
```

Bearbeiten der BaInterfaceSymbolNames [▶ 1267].

Position

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Aktuelle Position (0-100%).

PositionFeedback

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.NumberOrBoolean
```

Feedback für die Position (0-100%).

Events

Ereignis	Beschreibung
onPositionChanged	Wird ausgelöst, wenn der Benutzer die Position des Fensters oder einer Dachkuppel verändert.
onManualReset	Wird ausgelöst, wenn die Schaltfläche zum Zurücksetzen vom Hand- in den Automatikmodus betätigt wird.

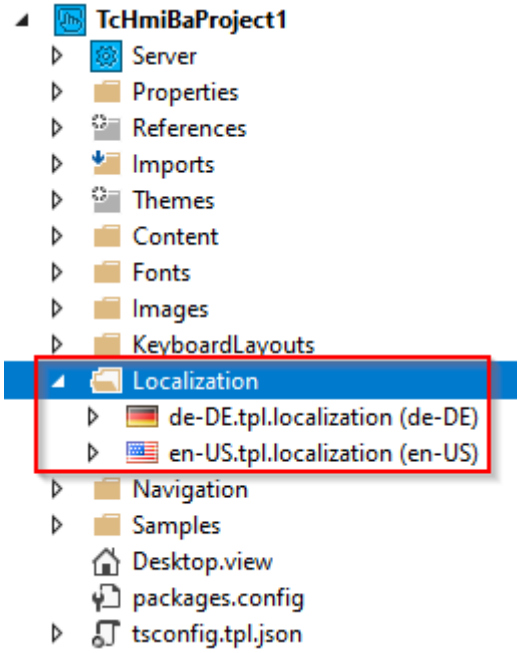
6.2.1.3 Guidlines

6.2.1.3.1 Lokalisierung

Die Lokalisierung in den TcHmiBa-Controls wird vom TwinCAT HMI Lokalisierungssystem verwaltet.

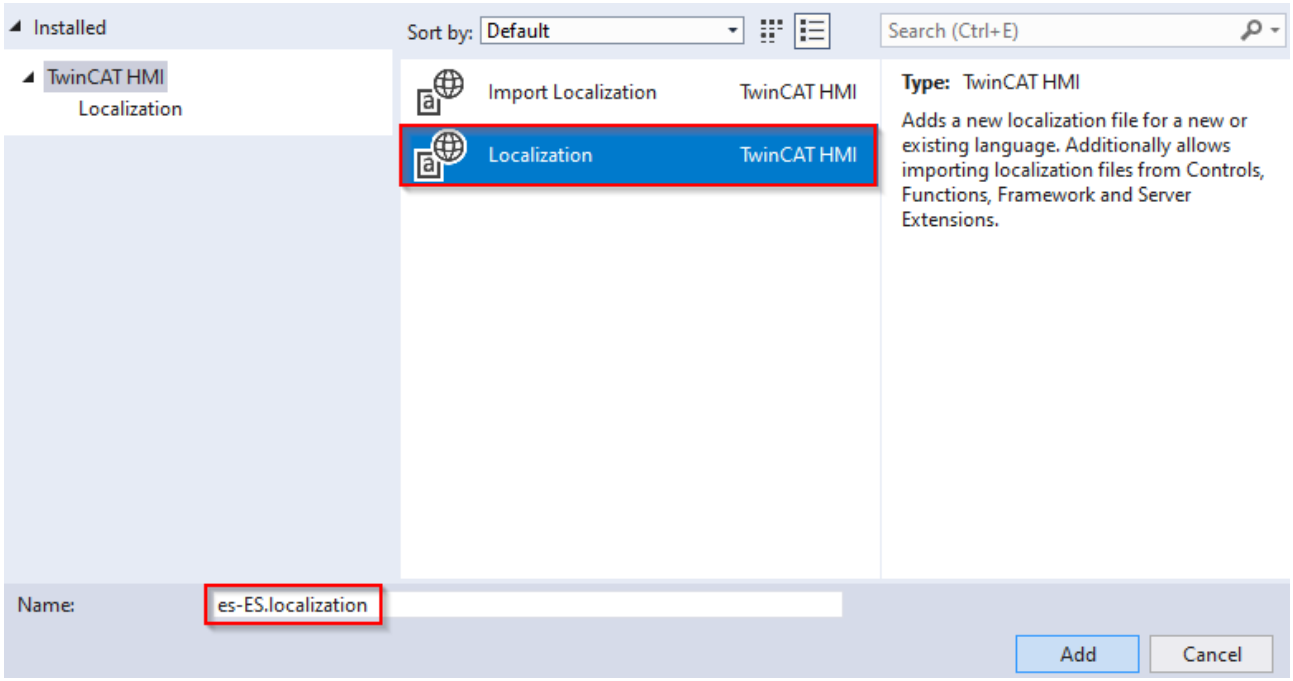
Lokalisierung im Projekt

Ein mit der TcHmiBa-Projektvorlage erzeugtes TcHmi-Projekt enthält bereits Lokalisierungen für die Sprachen Deutsch und Englisch. Sie lassen sich verändern, ergänzen und um weitere Sprachen erweitern.

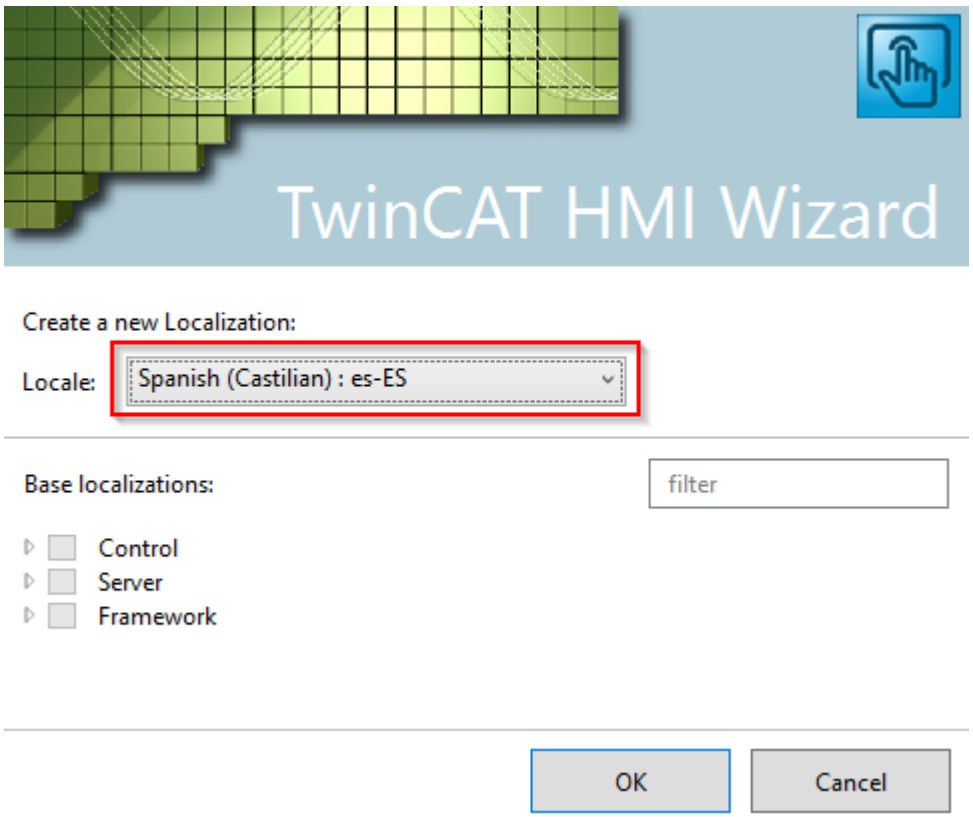


Lokalisierung hinzufügen

Jede neue Sprache erfordert eine neue Lokalisierungsdatei im Ordner für Lokalisierungen.

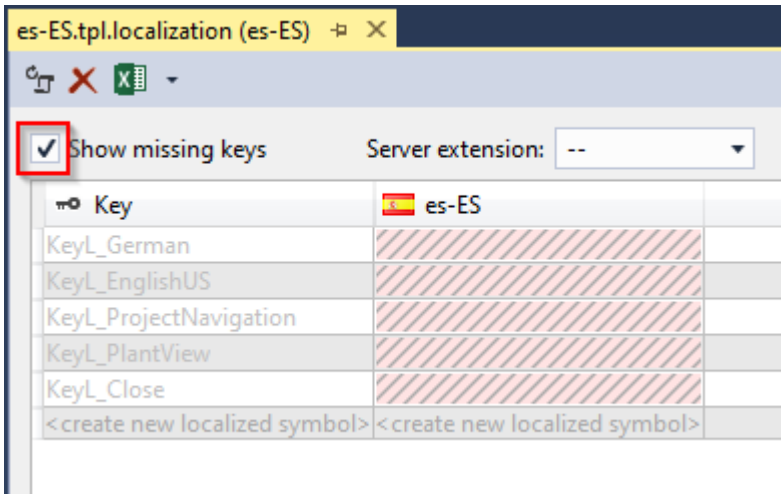


Im Anschluss ist die zutreffende Kombination aus Sprache und Gebiet auszuwählen.



Eine neu angelegte Lokalisierungsdatei wird automatisch geöffnet und erscheint zunächst leer.

Bereits existierende Spracheinträge aus anderen Sprachen lassen sich durch Aktivieren der Option **Show missing keys** einblenden.

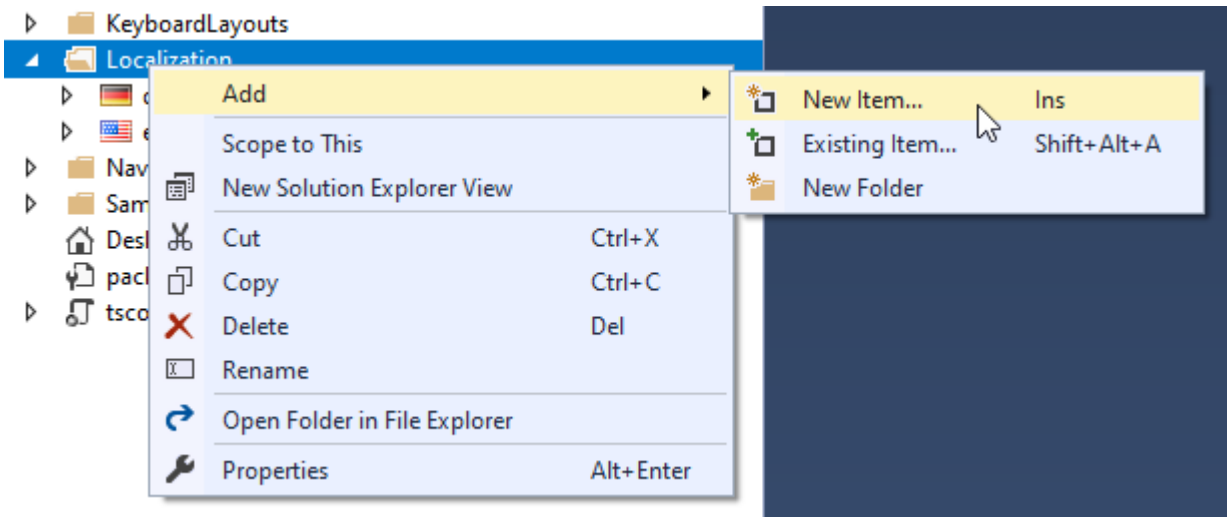


Lokalisierung von Controls

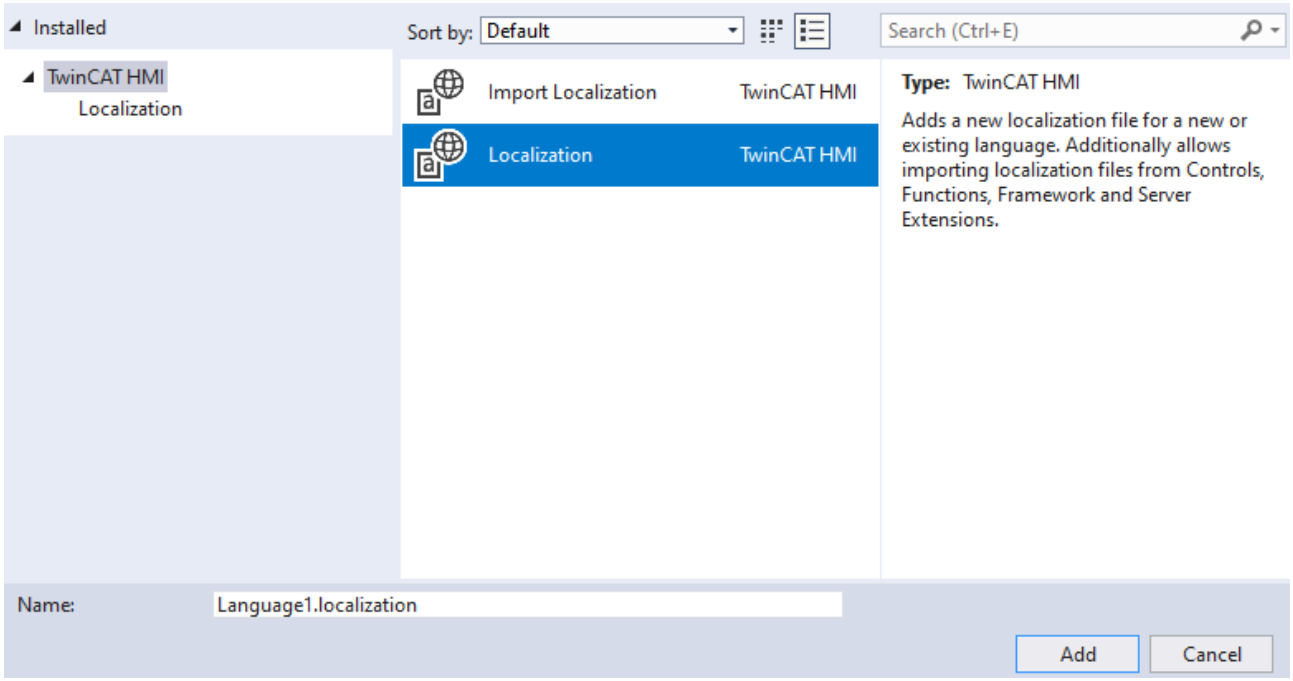
Lokalisierungen von Controls aus NuGet-Paketen können überschrieben und um weitere Sprachen ergänzt werden.

Lokalisierung hinzufügen

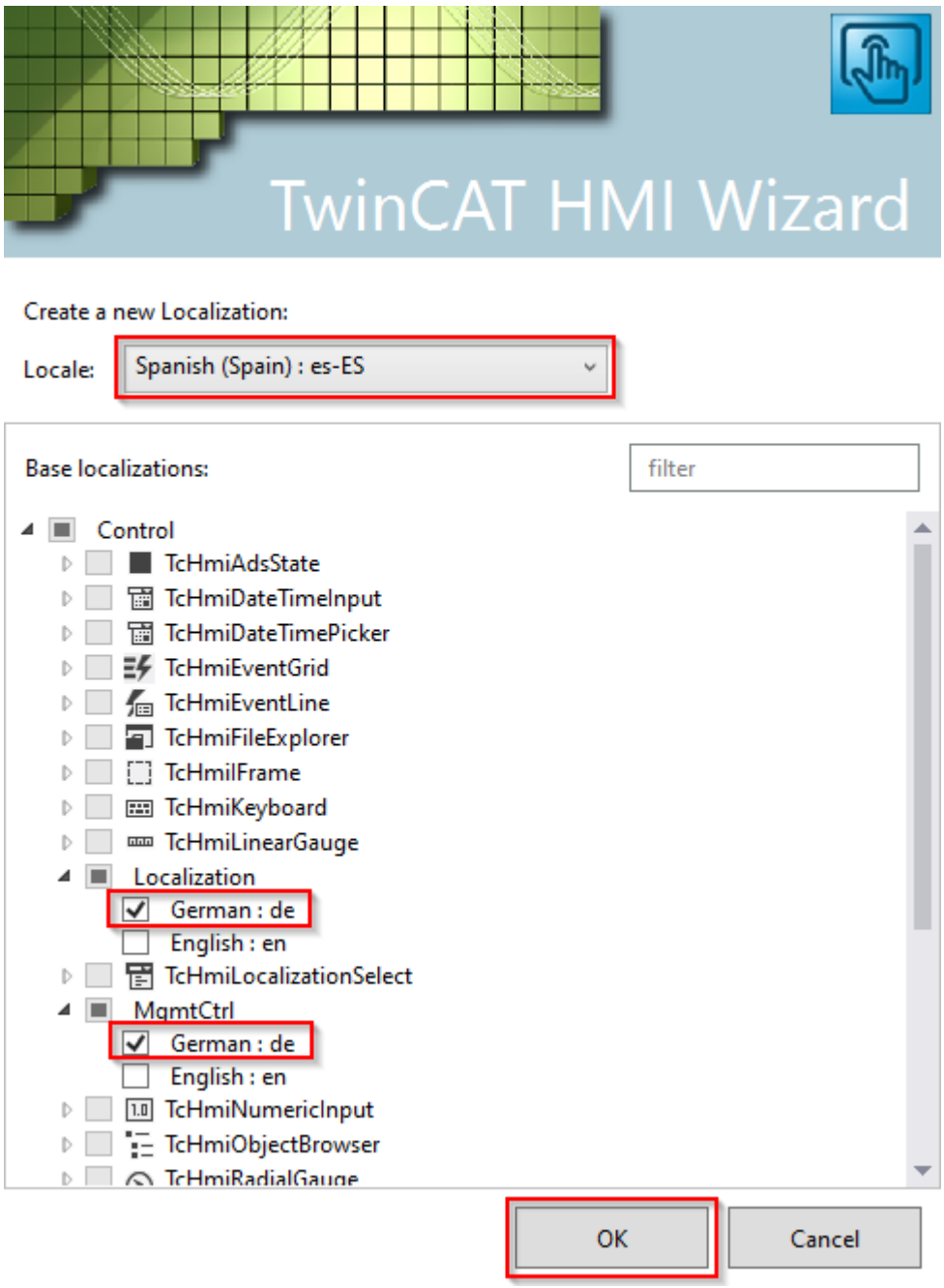
Jede neue Sprache erfordert eine neue Lokalisierungsdatei im Ordner für Lokalisierungen.



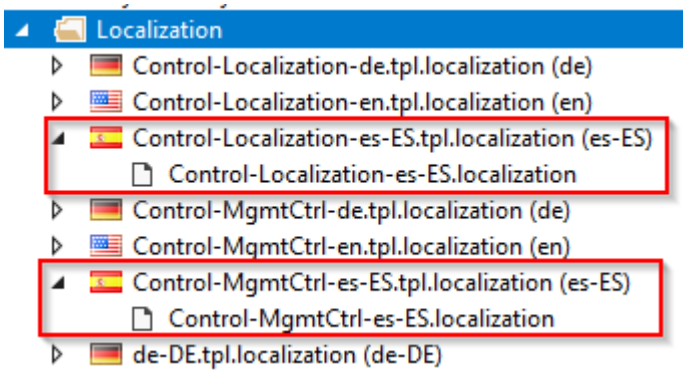
Der Name der Datei wird im weiteren Verlauf automatisch generiert und ist daher irrelevant.



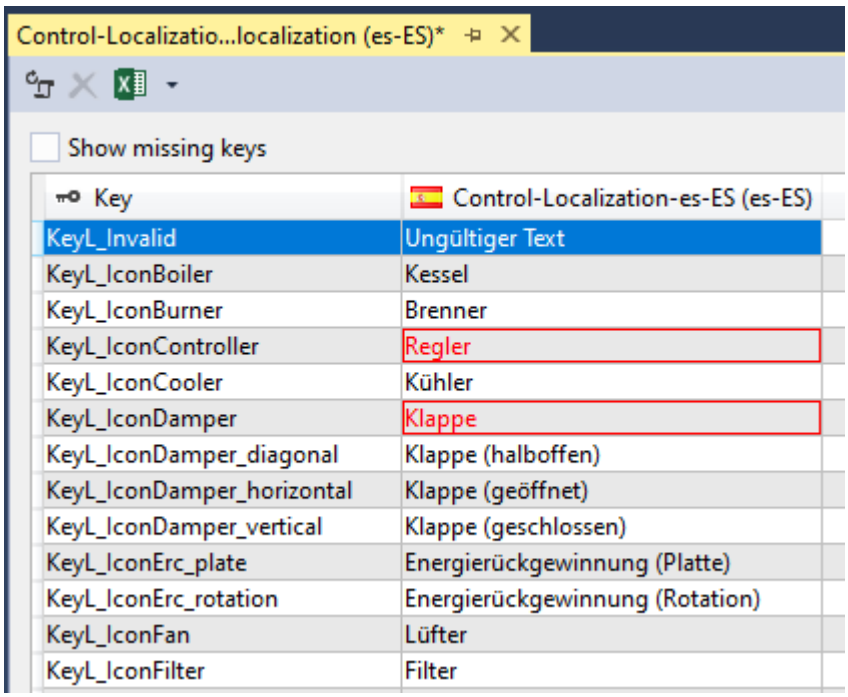
Pro Control ist eine beliebige vorhandene Lokalisierung auszuwählen, um die Spracheinträge übernehmen zu können.



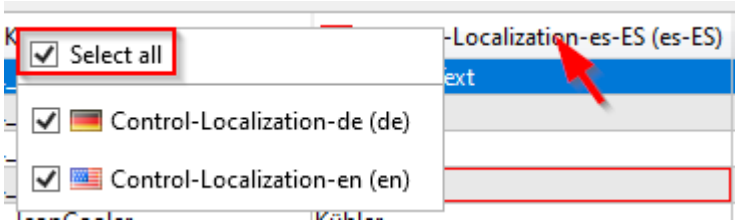
Nach Bestätigung der Auswahl stehen die Lokalisierungsdateien im Projekt zur Verfügung.



Abschließend sind die Spracheinträge in der neuen Lokalisierung zu ersetzen.

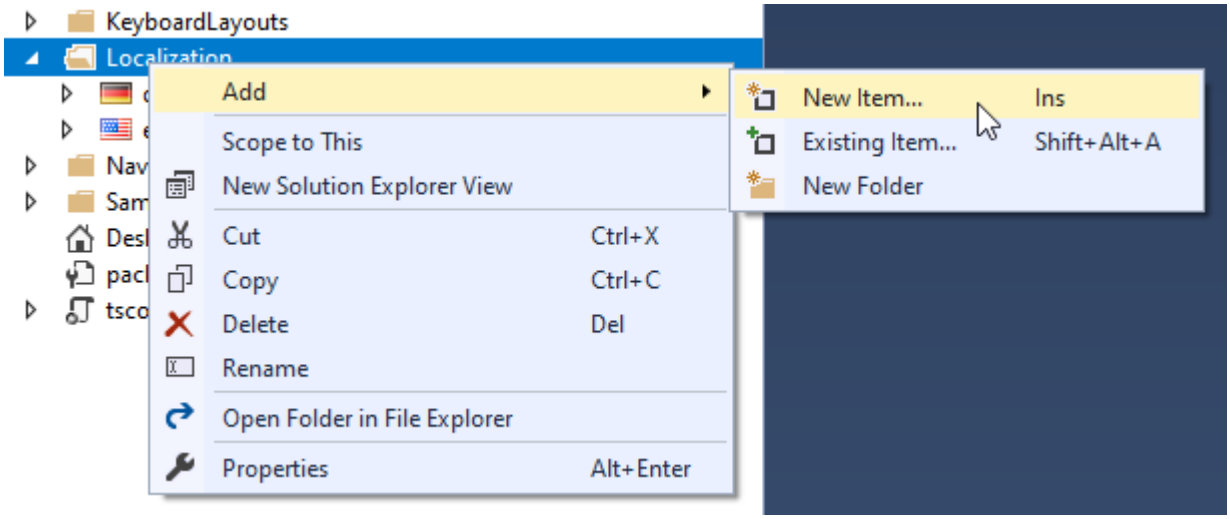


Über das Rechtsklickmenü einer Spalte lassen sich die verfügbaren Lokalisierungen ein- und ausblenden.

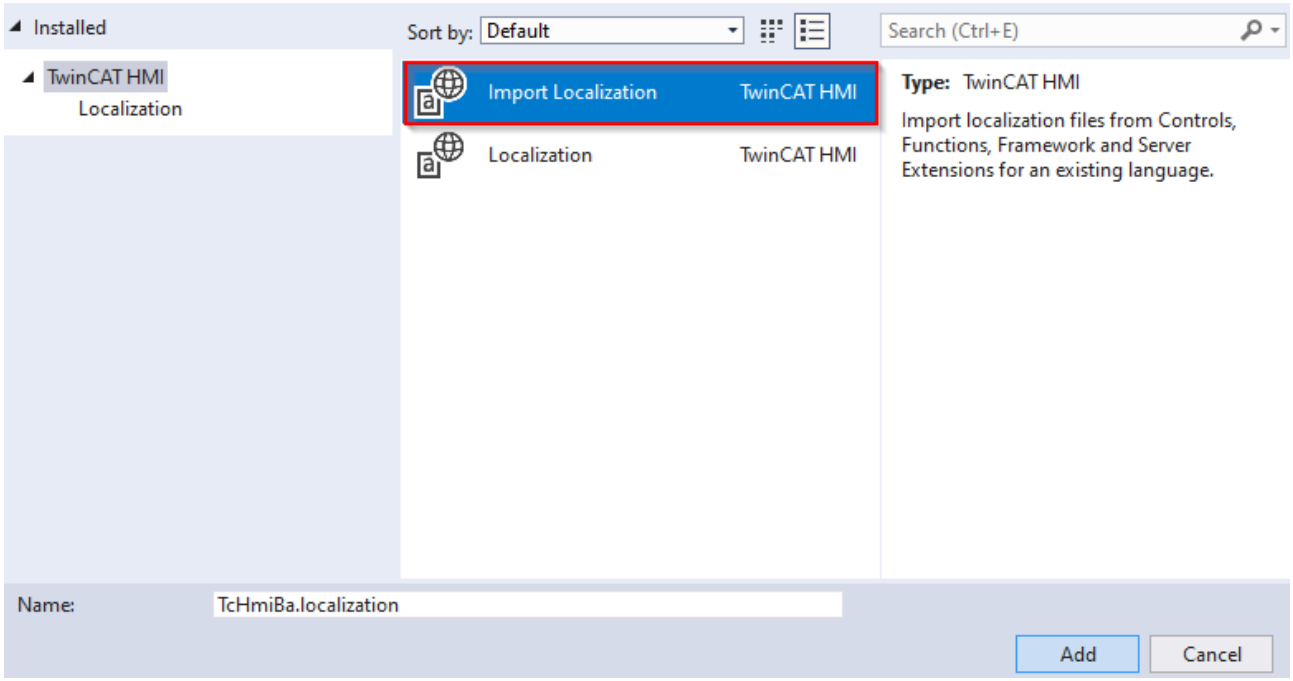


Lokalisierung überschreiben

Die Lokalisierung des Controls muss zunächst in den Ordner für Lokalisierungen importiert werden.



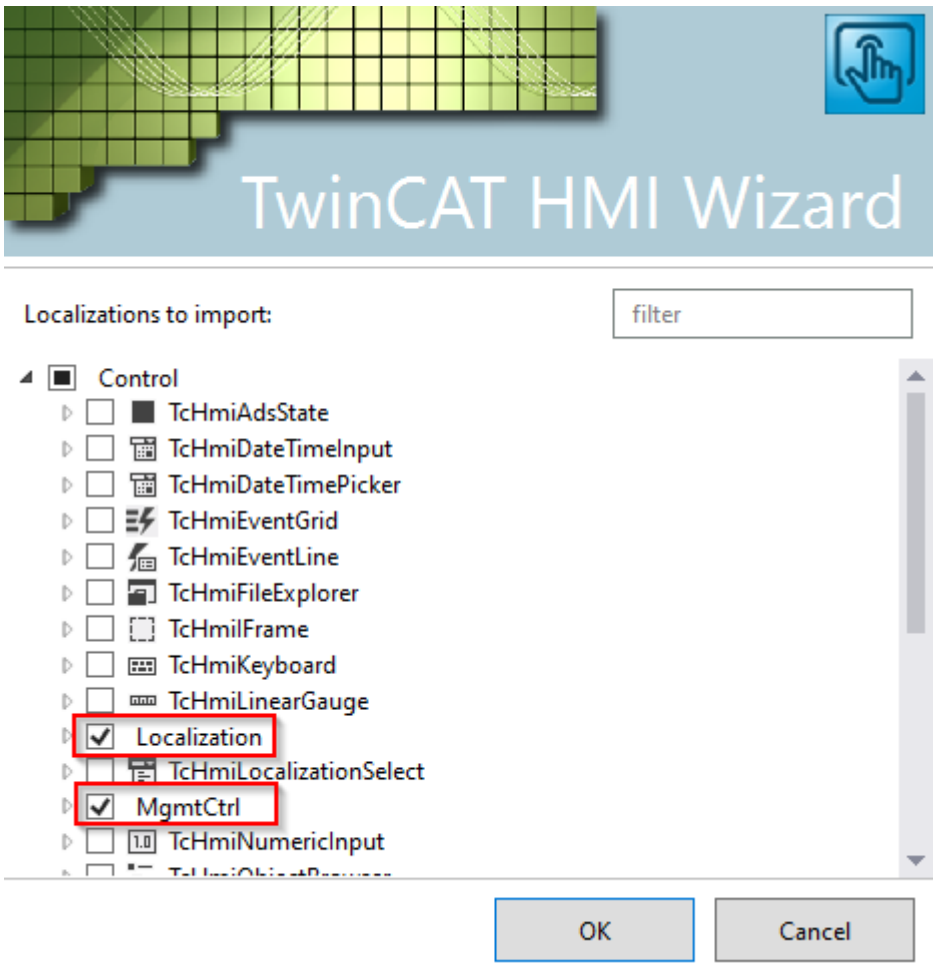
Der Name der Datei wird im weiteren Verlauf automatisch generiert und ist daher irrelevant.



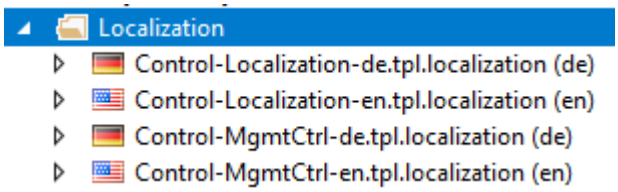
Anschließend sind die zu überschreibenden Lokalisierungen auszuwählen.

MgmtCtrl beinhaltet alle Lokalisierungen vom TcHmiBaFramework [▶ 1263]-Paket und TcHmiBaControls [▶ 1141]-Paket.

Localization beinhaltet alle Lokalisierungen vom TcHmiBaIcons [▶ 1252]-Paket.



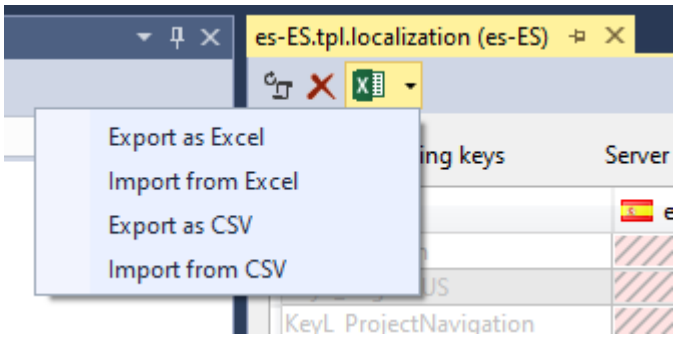
Nach Bestätigung der Auswahl stehen die Lokalisierungsdateien im Projekt zur Verfügung.



Lokalisierung exportieren/Importieren

Excel

Das Menü einer Lokalisierungsdatei ermöglicht den Export von Lokalisierungen. Die exportierten Lokalisierungen können in Excel bearbeitet und im Anschluss wieder importiert werden.



6.2.1.3.2 BaObject Handhabung

Nahezu alle bestehenden TcHmiBa-Controls [▶ 1141] können ein BaObject [▶ 62] verarbeiten. Der Umgang mit Symbolen der BaSite-Extension ist grundlegend identisch zu denen der ADS-Extension.

Handhabung

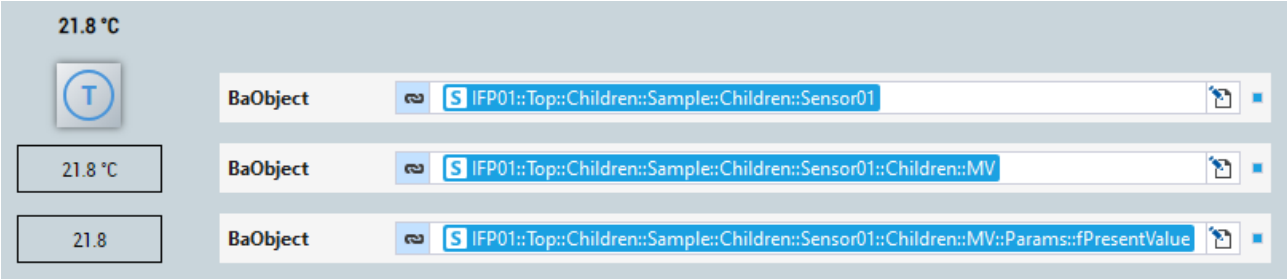
Ein simples Beispiel soll die Anwendungsmöglichkeiten innerhalb und außerhalb eines UserControls andeuten. Dazu werden in einem TcHmi-Projekt die TcHmiBa-Controls *SensorAnalog* und *Textblock* mit einer stark reduzierten Variante eines analogen Sensors verknüpft.

Die Struktur der SPS sieht wie folgt aus:

Name	Datatype
ADS	object
BaSite	object
EventHistory	object
Events	object
IFP01	BA.IFP01
Top	BA.IFP01.ProjectStructure
Children	object
Sample	BA.IFP01.Tc3_XBA.FB_BA_View
Children	object
Sensor01	BA.IFP01.FB_BA_SensorAnalog
Children	object
MV	BA.IFP01.Tc3_XBA.FB_BA_AI_Raw_34321
Params	object
fPresentValue	BaVar.System.Single
Params	object
Params	object

Symbol Mapping direkt

Verknüpfen des Symbols aus der BaSite-Extension direkt mit dem Attribut BaObject eines TcHmiBa-Controls.

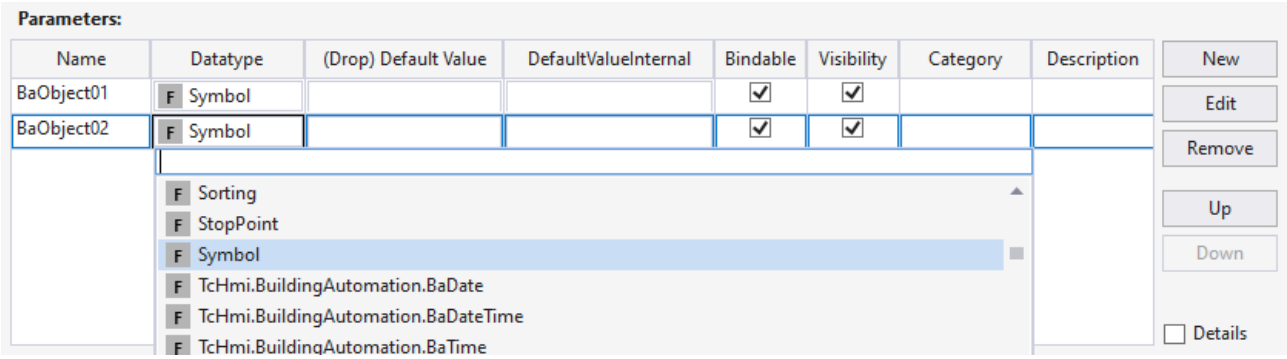


Symbol Mapping über UserControl-Parameter

Das UserControl in diesem Beispiel verfügt über die Parameter BaObject01 und BaObject02 vom Typ Symbol, um die Symbole aus der BaSite-Extension durchzureichen.



Der Datentyp *Symbol* ist für die TcHmiBa-Controls performanter.



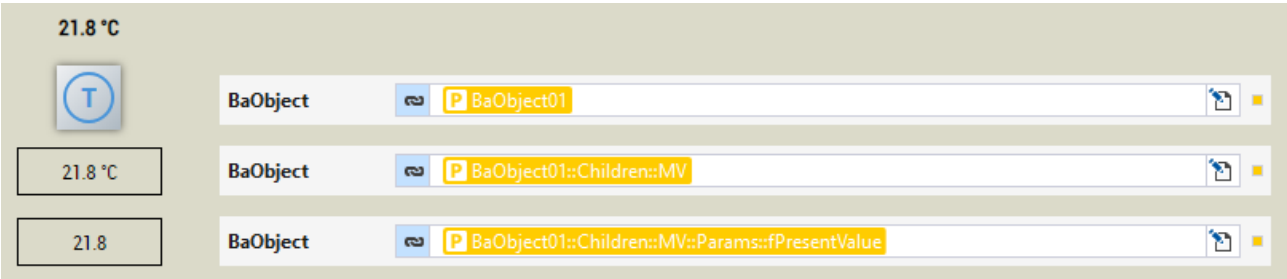
Aufgrund des Datentyps ist beim Erstellen des Bindings ein Navigieren in das Objekt nicht mehr möglich.

Durch händische Erweiterung des Symbols lassen sich tieferliegende Objekte dennoch erreichen.

Ein Parameter

Symbol für Parameter BaObject01:

- `IFP01::Top::Children::Sample::Children::Sensor01`



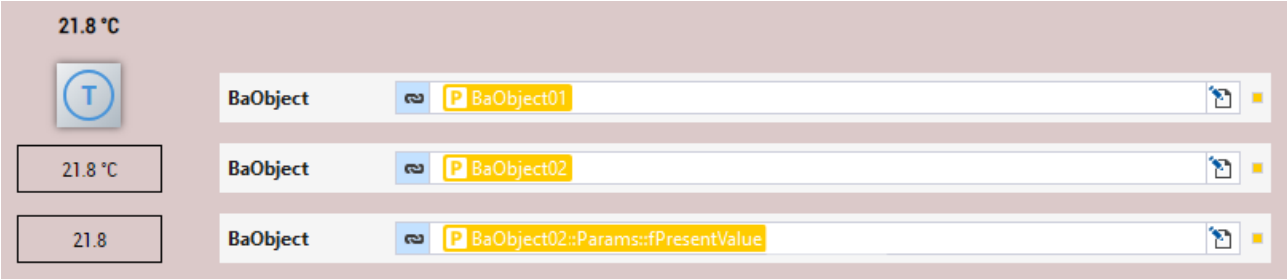
Mehrere Parameter

Symbol für Parameter *BaObject01*:

- `IFP01::Top::Children::Sample::Children::Sensor01`

Symbol für Parameter *BaObject02*:

- `IFP01::Top::Children::Sample::Children::Sensor01::Children::MV`



6.2.1.3.3 BaTemplate Handhabung

Template Handling

Erläuterung der Funktionsweise am Beispiel des Standard TF8040 SPS-Templates *FB_BA_AC_ColT_02* für einen Kühler.

Das dazu passende TcHmiBa-Template *FB_BA_AC_ColT_02* erwartet im verknüpften *BaObject* die folgenden Unterelemente (TF8040 SPS-Templates) in der vordefinierten Hierarchie.

Unterelemente:

Symbolname	SPS-Template	Beschreibung
Fdb	FB_BA_AI_IO	Feedback
Mdlt	FB_BA_AO_IO	Sollwert
MV	FB_BA_AI_IO	Feedback
TFl	FB_BA_SensorAnalog_IO	Vorlauftemperaturfühler
TRt	FB_BA_SensorAnalog_IO	Rücklauftemperaturfühler
Vlv	FB_BA_Vlv	Ventil

Hierarchie:

- BAObject
    - TFl
        - MV
    - TRt
        - MV
    - Vlv
        - Fdb
        - Mdlt

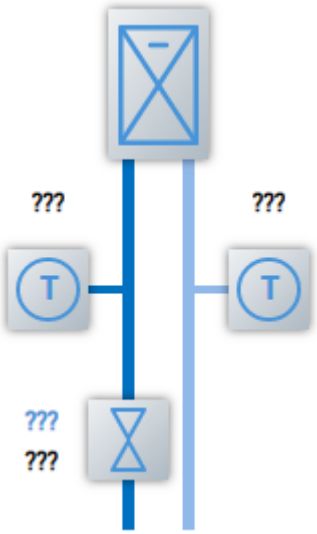
Zustände

Die Zustände eines TcHmiBa-Templates hängen vom *BaObject* ab.

Kein BaObject

Der Kühler ohne verknüpftes *BaObject*. Alle Unterelemente sind vorhanden, jedoch ohne Wertanzeige.

Darstellung:



Unterelement fehlt

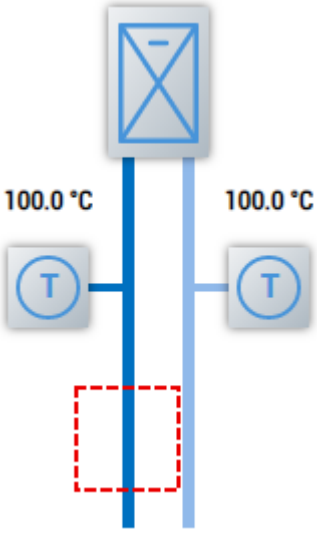
Dem Kühler wurde ein *BaObject* zugewiesen, in dem das Ventil (Vlv) fehlt.

Die Validierung registriert, dass ein Unterelement auf der obersten Ebene nicht vorhanden ist oder einen anderen Symbolnamen hat, und blendet es daher aus.

Hierarchie:

- BAObject
  - TFl
    - MV
  - TRt
    - MV

Darstellung:



Symbolname umbenannt

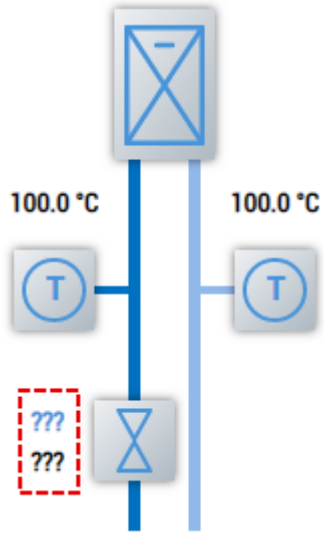
In diesem *BaObject* sind die Unterelemente auf der obersten Ebene korrekt vorhanden, aber der Symbolname für Fdb vom Vlv wurde im SPS-Template umbenannt.

Da die Abweichung in einer der unteren Ebenen auftritt, wird das betroffene Unterelement mit der Wertanzeige im Fehlerzustand eingeblendet.

Hierarchie:

- BAObject
  - TFl
    - MV
  - TRt
    - MV
  - Vlv
    - Fdb_Test
    - Mdlt

Darstellung:

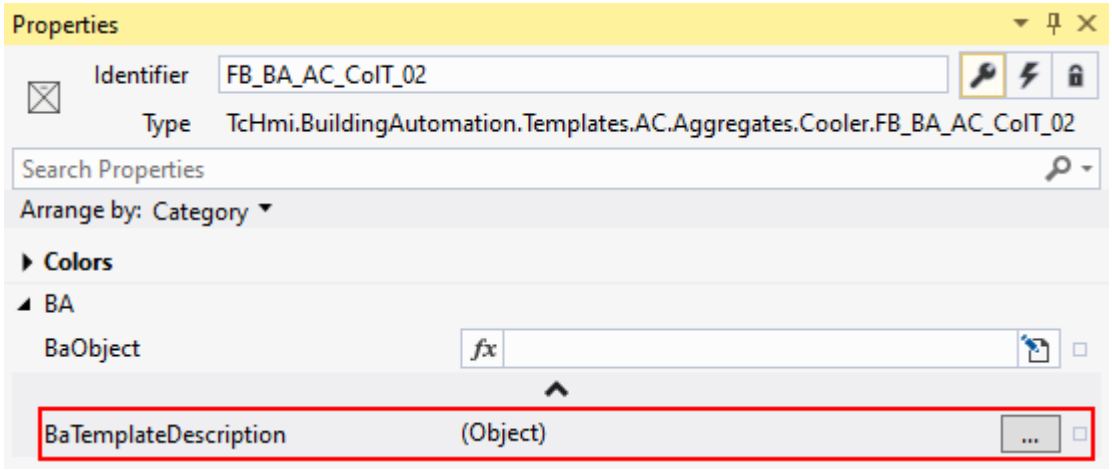


Zusätzlich weist eine Fehlermeldung auf das Unterelement hin.



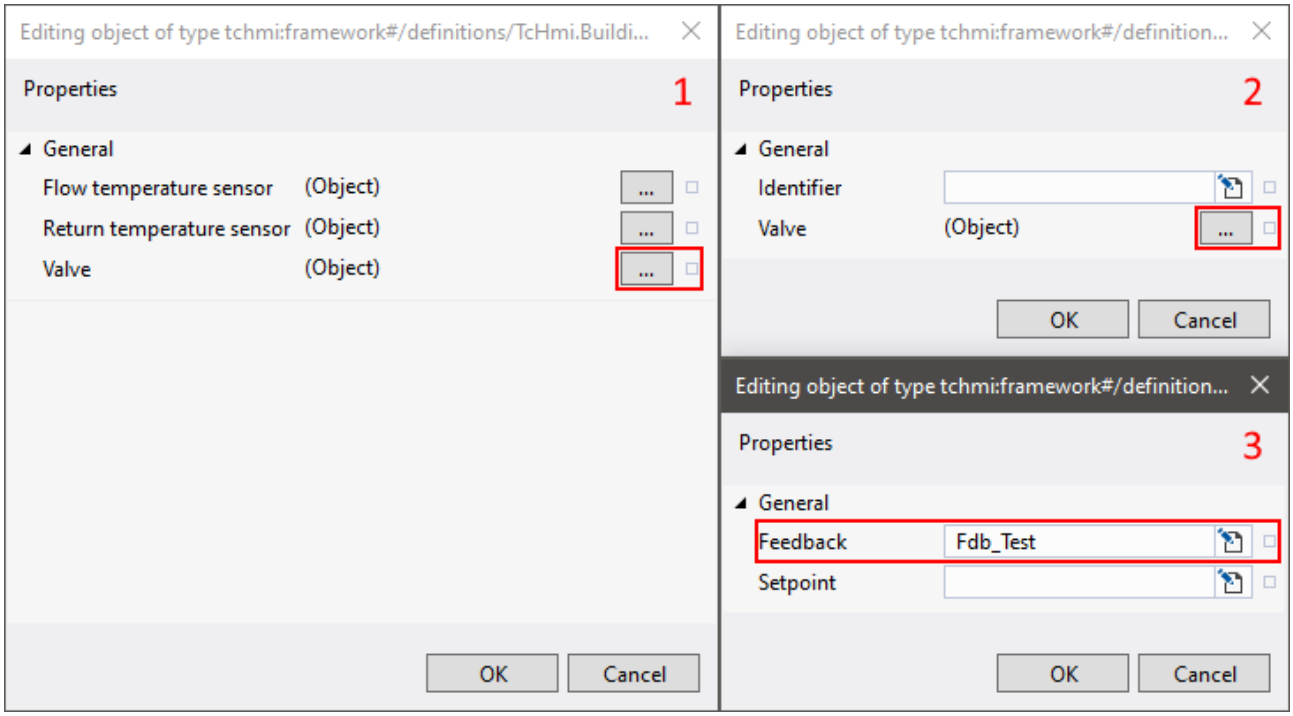
Symbolnamen anpassen

Neue Symbolnamen können über die **BaTemplateDescription** mitgeteilt werden.



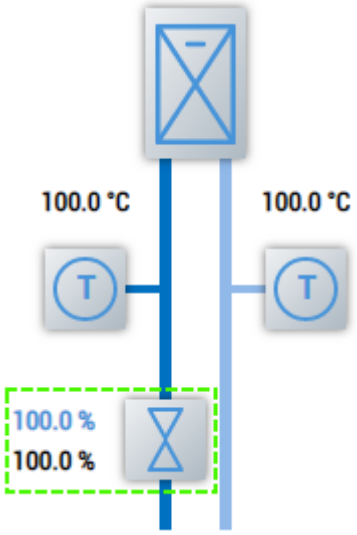
Ein Dialog listet alle Unterelemente auf und ermöglicht die Navigation durch die weiteren Ebenen [1]. Die Angabe zum **Identifier** bezieht sich auf den Symbolnamen der aktuellen Ebene [2]. Ist diese Angabe nicht mehr möglich, dann ist die letzte Ebene des Unterelements erreicht [3].

Unter **Feedback** ist der aktuelle Symbolname für **Fdb** aus dem SPS-Template einzutragen.



Nach Bestätigung der Eingabe führt das TcHmiBa-Template die Validierung des BaObjects erneut durch.

Darstellung:



Der Kühler ist wieder voll funktionsfähig.

Verschachtelte BaObjekte verwenden

Befinden sich die BaObjekte z.B. für den Command von einem Motor nicht auf der standardmäßig vorgesehenen Ebene, sondern in einer tieferen Ebene, so kann auf diese verschachtelten Objekte mit der folgenden Syntax zugegriffen werden.

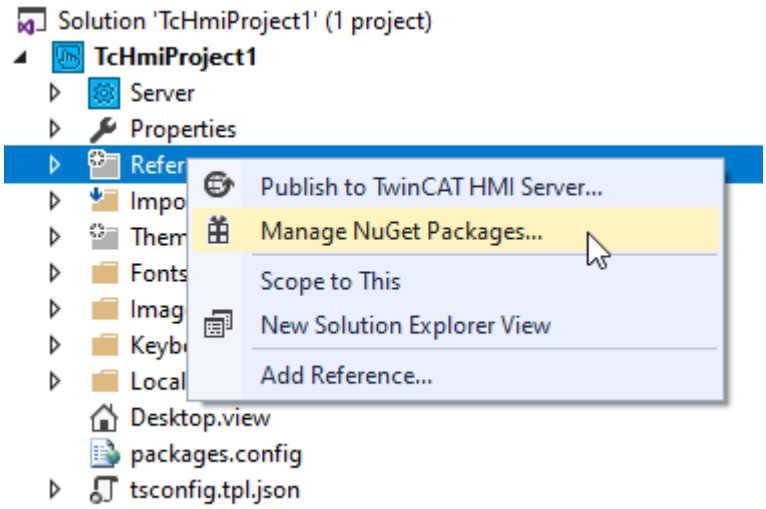
```
MyView::Motor::Cmd
```

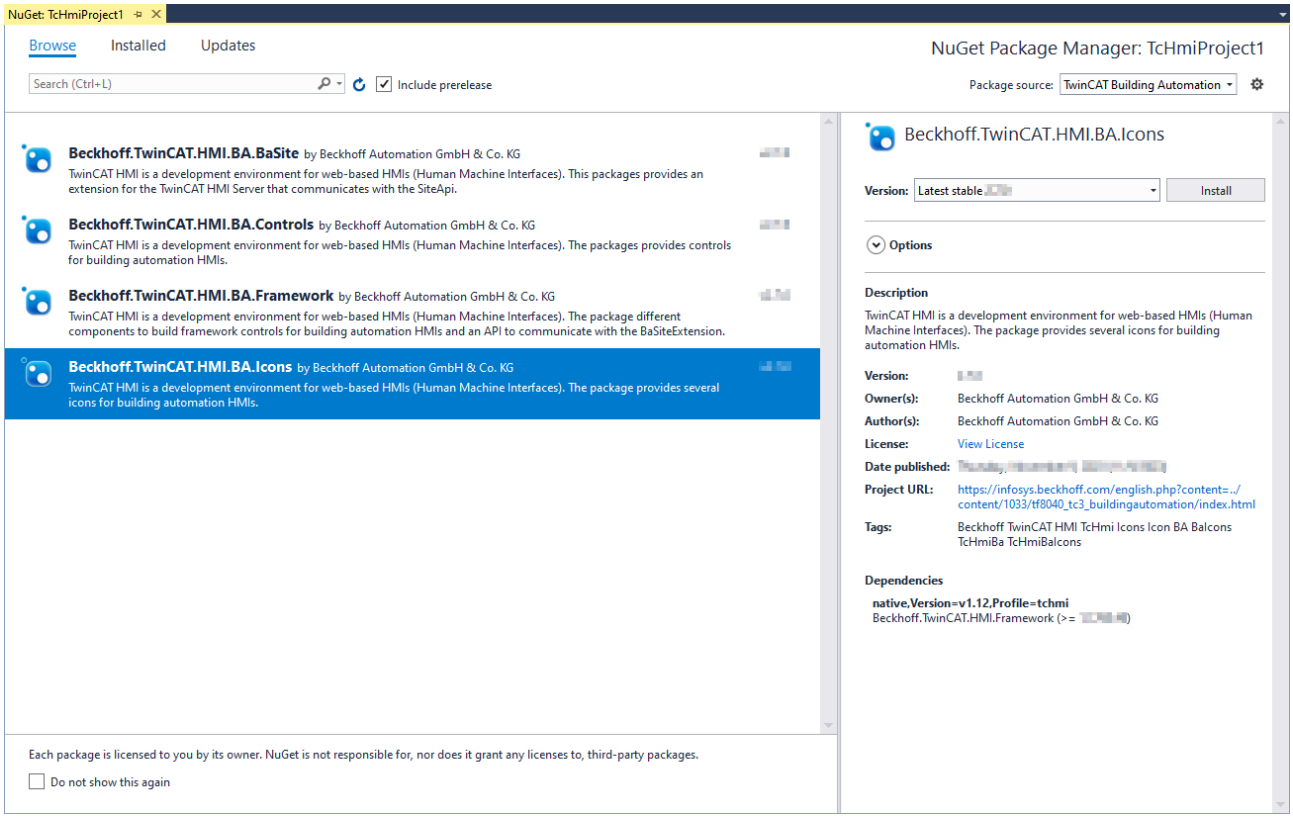
6.2.1.4 Icons

In TcHmiBa sind verschiedene Icons enthalten, die zur Realisierung von Visualisierungen für die Gebäudeautomation notwendig sind. Die Icons sind im Format **.svg* erstellt und für die Verwendung im Web gedacht.

Installation

Um die Icons verwenden zu können, muss das NuGet-Paket **Beckhoff.TwinCAT.HMI.BA.Icons** installiert sein.





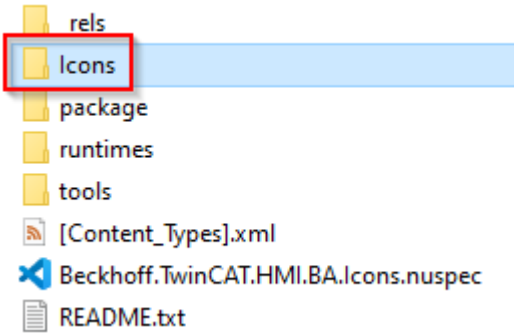
Da eine Abhängigkeit zum Beckhoff.TwinCAT.HMI.Framework besteht, wird dieses ebenfalls installiert.

Verwendung

Es gibt drei verschiedene Verwendungsmöglichkeiten.

ZIP-Archiv

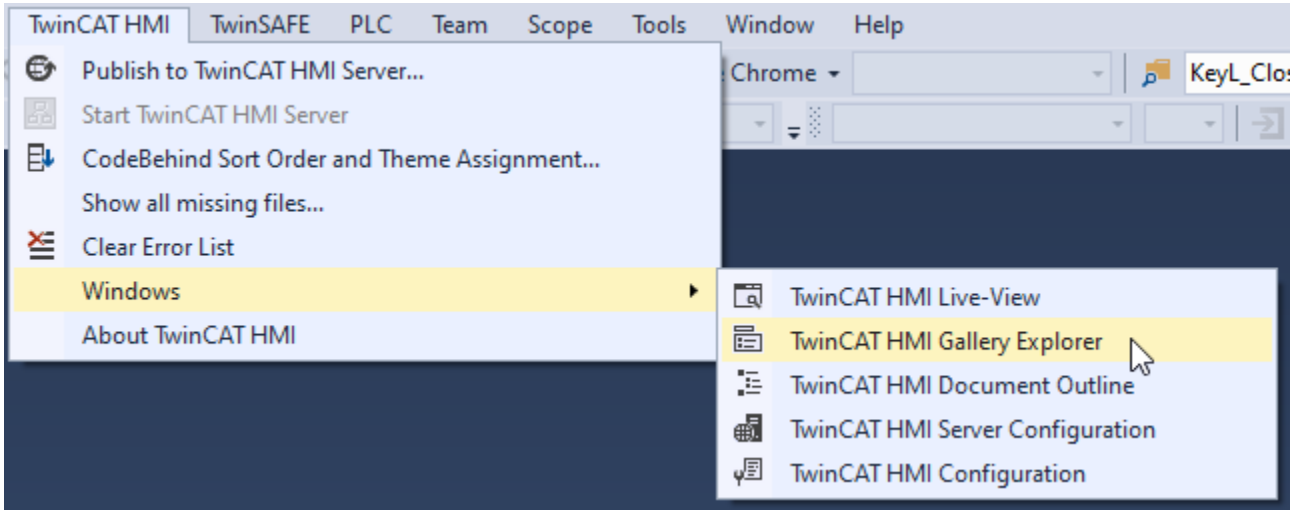
Ein NuGet-Paket ist im Prinzip ein ZIP-Archiv, das nach dem Entpacken eine direkte Verwendung der Icons ermöglicht. Der entpackte Inhalt sieht wie folgt aus:

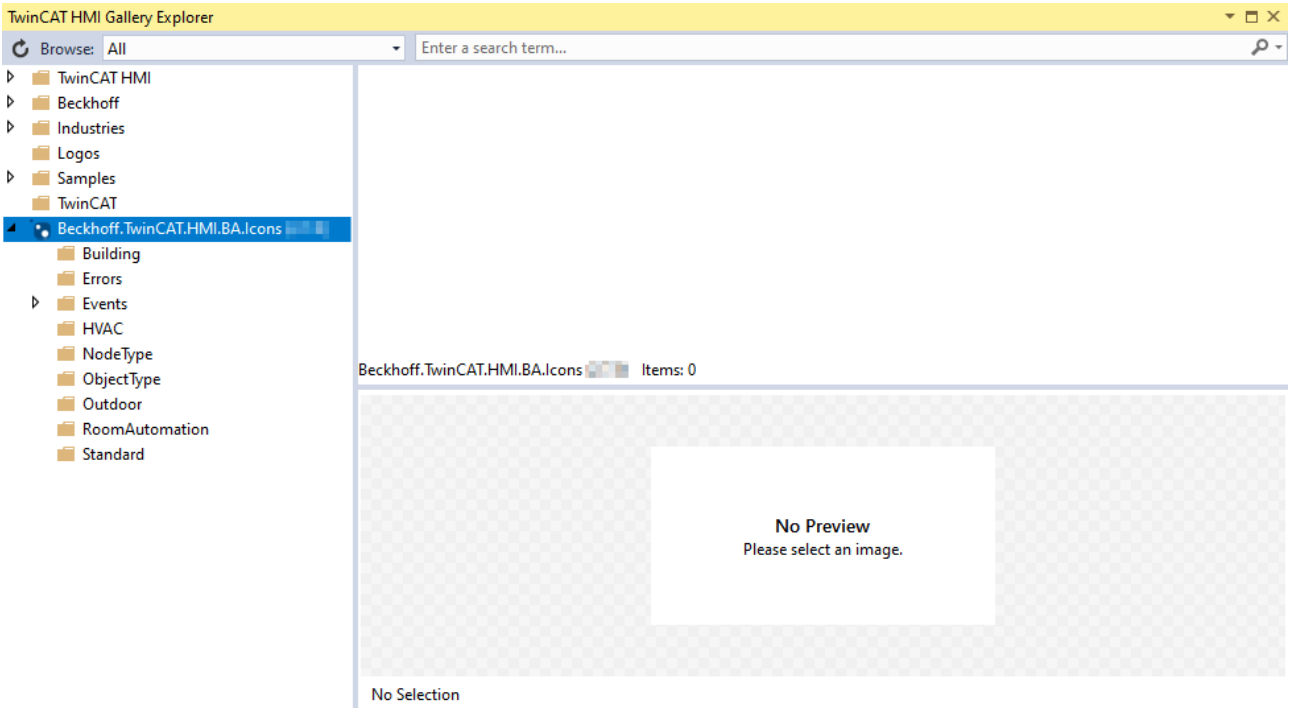


Einzig der Ordner **Icons** ist relevant. In ihm befinden sich die verschiedenen Icons aufgeteilt nach inhaltlichen Kategorien.

GalleryExplorer

Nach der Installation des NuGet-Pakets in einem **TwinCAT HMI Projekt**, integrieren sich die Icons in den **GalleryExplorer.**





Die Icons sind hier nach Funktion gegliedert. Um ein Icon aus dem **GalleryExplorer** zu verwenden, muss es per Drag-and-Drop in einen Ordner im Projekt gezogen werden.



Das Icon wird dabei als Kopie im Projektverzeichnis angelegt und nicht als Referenz.

Verwendung als Referenz

Beim Einsatz der Icons als Referenz lassen sich deren erweiterte Funktionalitäten nutzen. Zusätzlich profitieren die Icons direkt von Updates des NuGet-Pakets.

TcHmi-Projekt

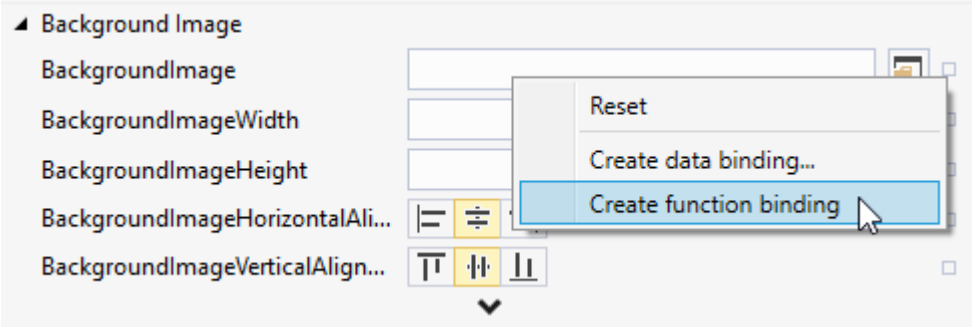
Mit der Funktion

```
TcHmi.BuildingAutomation.Functions.GetBaIconPath()
```

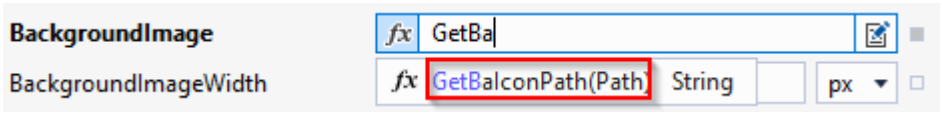
können die Icons in einem TcHmi Projekt als Referenz verwendet werden.

Folgende Schritte sind hierfür erforderlich:

1. Ziehen Sie ein Control (z. B. einen Button) auf den Content/View.
2. Erstellen Sie das **Function Binding**.



3. Im Feld **Backgroundimage** geben Sie **GetBaIconPath** ein (Funktion wird vorgeschlagen).

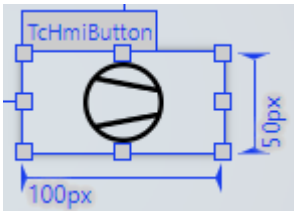


4. Pfadangabe zum Icon, z.B. *"HVAC/Fan"* (Anführungsstriche sind zu beachten).
5. Einstellen von Größe und Position.



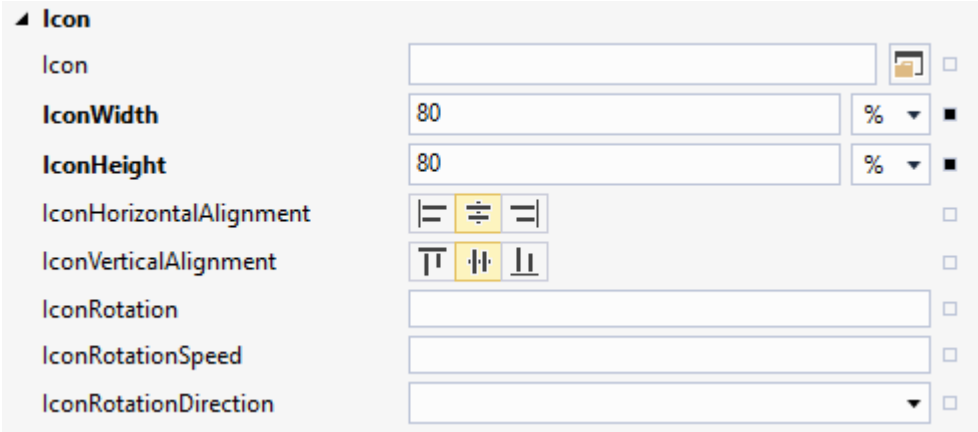
Die Pfadangaben sind aus der Struktur des *GalleryExplorers* zu entnehmen.

⇨ Danach sollte der Button so aussehen:



Die erweiterten Funktionen der Icons lassen sich nur bei Controls aus dem NuGet-Paket **Beckhoff.TwinCAT.HMI.BA.Controls** nutzen.
Die Controls, die eine Kategorie *Icon* haben, besitzen erweiterte Einstellungsmöglichkeiten.

Als Beispiel wird im Folgenden der Button aus der Kategorie **BA | Common** verwendet.



Das Attribut Icon kann wieder über die Funktion

```
TcHmi.BuildingAutomation.Functions.GetBaIconPath()
```

gesetzt werden.

Durch diese Form des Einbindens lässt sich das Icon dynamisch verändern. Folgende Attribute stehen zur Verfügung:

- IconRotation
- IconRotationSpeed
- IconRotationDirection
- IconColor (siehe Kategorie Colors)

Im Code

Für die Verwendung der Icons im Code, z. B. während der Entwicklung von Framework-Controls kann noch einfacher auf die Pfade der Icons zugegriffen werden. Es befinden sich im Namensraum

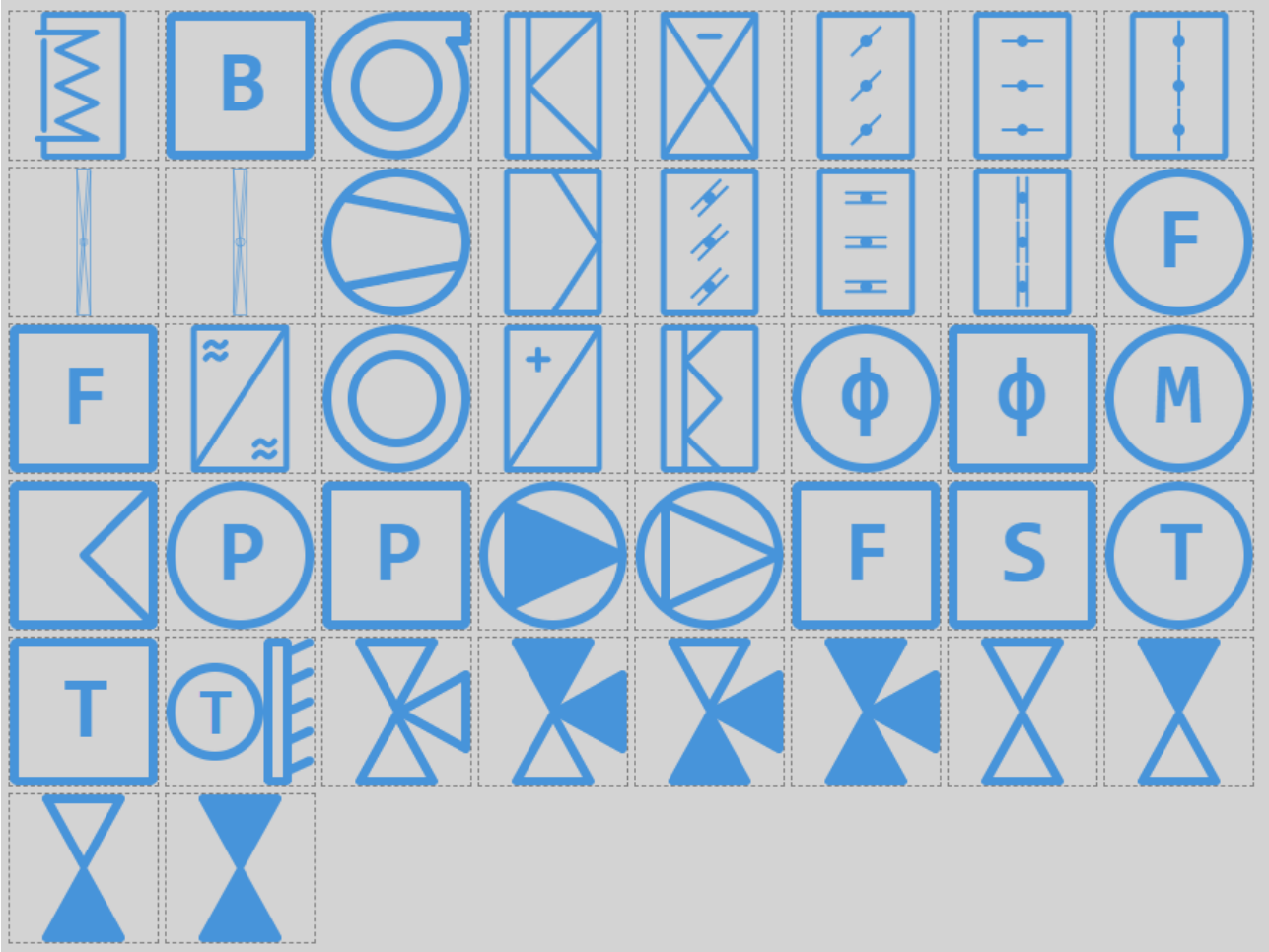
```
TcHmi.BuildingAutomation.Icons
```

Konstanten, die auf die jeweiligen Icons im NuGet-Paket zeigen (z. B. *TcHmi.BuildingAutomation.Icons.HVAC.Fan.path*).

HVAC Symbole

Symbole für R+I-Schematas.

Die Icons sind mit entsprechenden Größenverhältnissen für ein R+I-Schema gezeichnet und werden nur in dieser Auflistung anders dargestellt.



Ereignis- Symbole

Symbole zur Darstellung von Alarmen, Events oder Benachrichtigungen.

Events

Das Konzept der Alarme [▶ 33] nutzt die folgenden Icons.

Icon	Bezeichnung
	Alarm
	Störung
	Wartung
	Benachrichtigung
	Sonstiges

Die Ereignisse [▶ 33] werden in verschiedenen Abbildungszuständen angezeigt.

Flag

Die Flag-Icons werden angezeigt, wenn eines der StatusFlags eines Objektes aktiv ist.

Icon	Bezeichnung
	InAlarm
	Fault
	Overridden
	OutOfservice

Prioritäten

Icon	Bezeichnung
	LifeSafety
	Critical
	ManualLocal
	ManualRemote

Lock

Icon	Bezeichnung
	High
	Medium

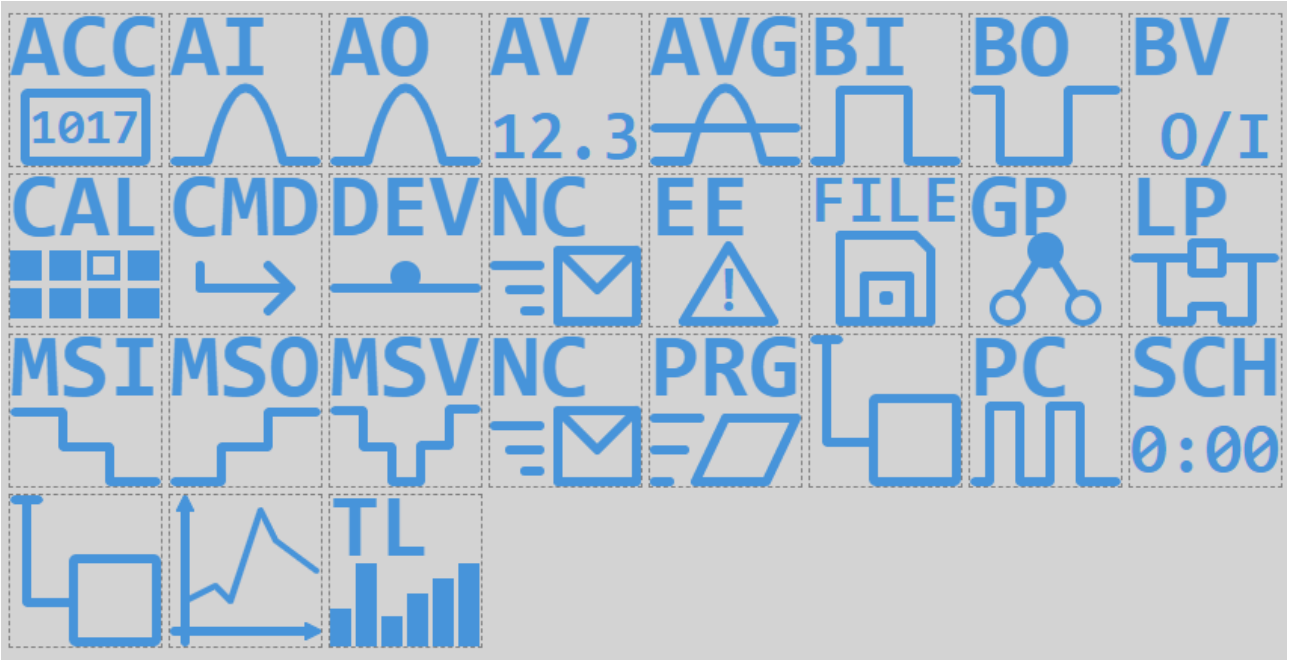
NodeType

Icon	Bezeichnung
	Aggregat
	Gebäude
	Gebäudeelement
	Komponente
	Schaltschrank
	Stockwerk
	Informationsschwerpunkt
	Liegenschaft
	Anlage

Icon	Bezeichnung
	Raum
	Gewerk

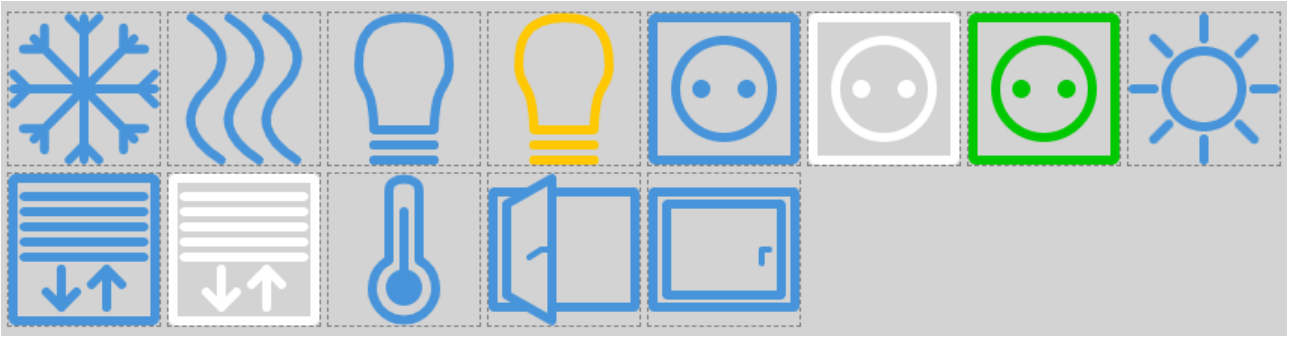
ObjectType

Symbole für die verschiedenen Objekttypen.



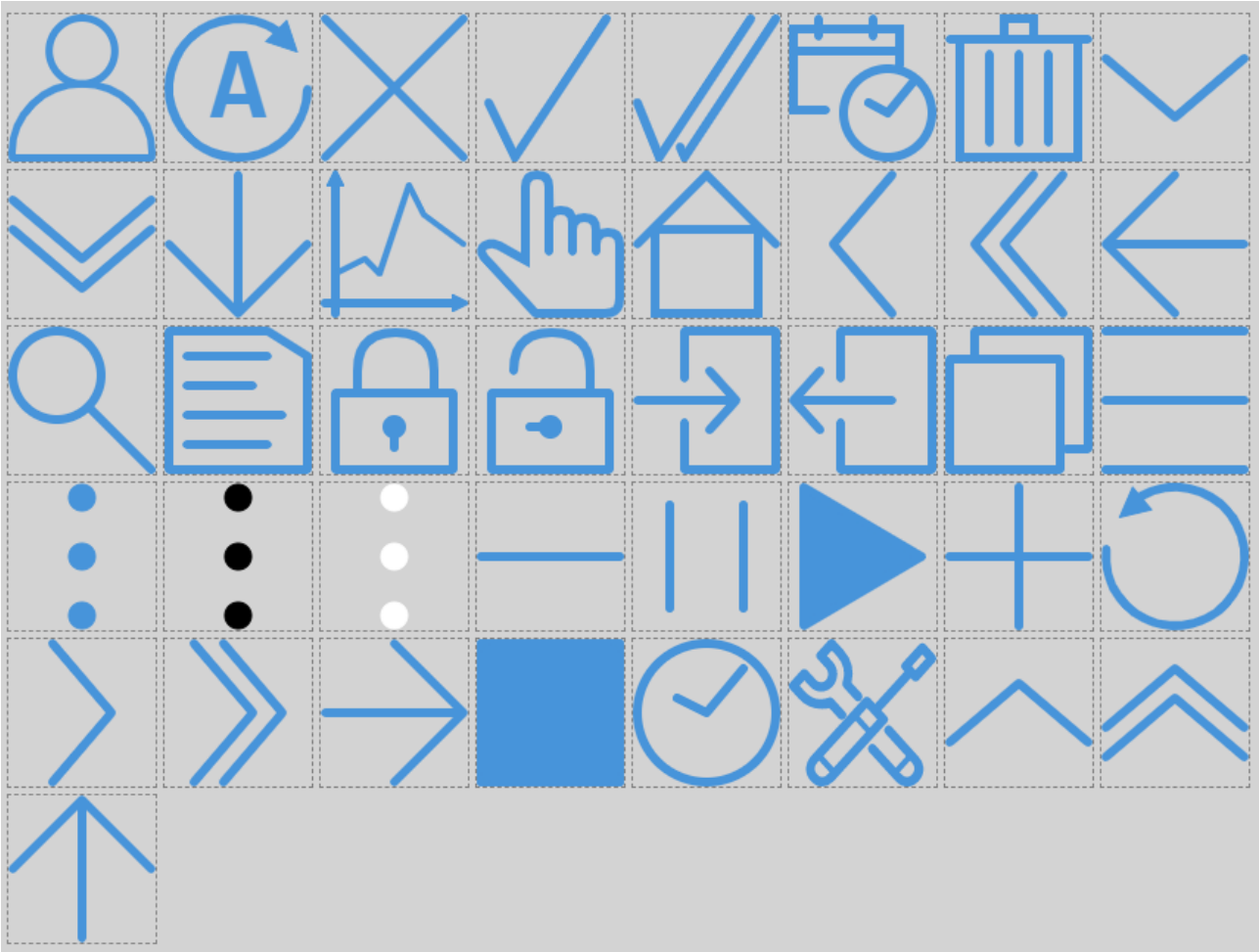
Room Automation

Symbole für die Raumautomation.



Standard

Standard-Symbole für Visualisierungen.



6.2.1.5 Framework

Das Framework enthält verschiedene Klassen, Enumerationen, Interfaces und Typen, die genutzt werden, um die Controls [▶ 1141] zu erstellen. Verschiedene Hilfsmethoden erleichtern ebenfalls die Server-Client-Kommunikation, den Zugriff auf verschiedene Inhalte und die Positionierung von Elementen in einem Control. In dem Paket sind **keine Controls** enthalten, weil es für die Verwendung in Framework-Control Projekten vorgesehen ist.
Das Framework ist in TypeScript geschrieben.

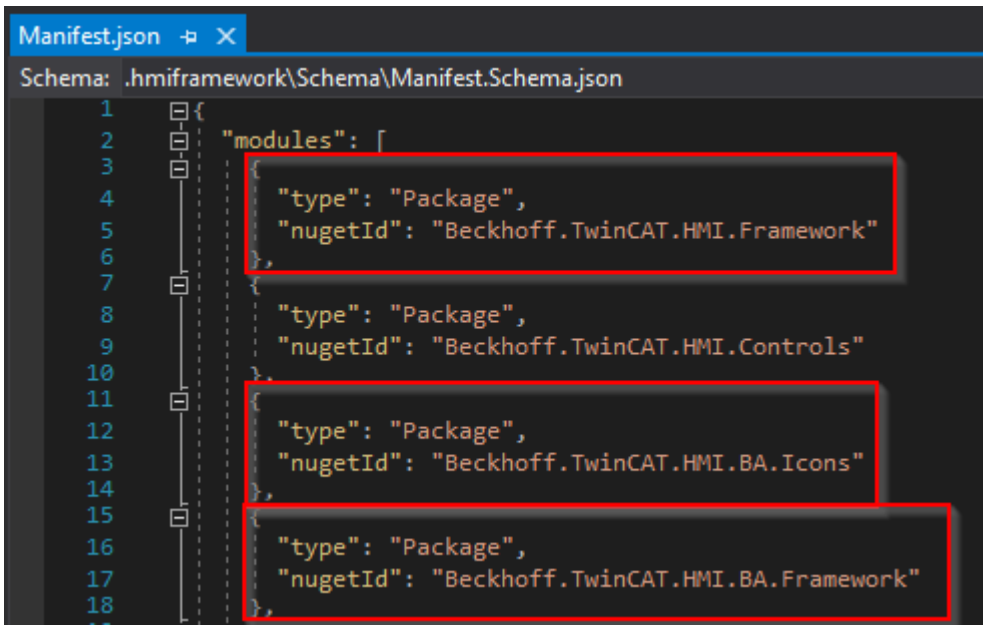
Wird die Komplettlösung TF8040 [▶ 8] verwendet, übernimmt das Framework ebenfalls viele Managementfunktionen, um die Generik [▶ 56] zu realisieren.

Installation

Damit das Framework verwendet werden kann, muss das NuGet-Paket **Beckhoff.TwinCAT.HMI.BA.Framework** installiert sein.
Zusätzlich sind noch weitere Funktionen erforderlich, die aus folgenden, automatisch mitinstallierten NuGet-Paketen stammen:

- Beckhoff.TwinCAT.HMI.BA.Icons [▶ 1252]
- Beckhoff.TwinCAT.HMI.Framework

Es ist sicherzustellen, dass die Dateien aus dem Framework im Projekt geladen werden. Bei der Verwendung in einem *TcHmi Framework Projekt* sind die entsprechenden Pakete in der *Manifest.json* einzutragen.



Die Unterstützung von IntelliSense für das Framework im Visual Studio lässt sich durch das Hinzufügen folgender Einträge zur *tsconfig.tpl.json* erreichen:

```
"include": [
  "$(Beckhoff.TwinCAT.HMI.Framework).InstallPath/TcHmi.d.ts",
  "$(Beckhoff.TwinCAT.HMI.BA.Icons).InstallPath/index.d.ts",
  "$(Beckhoff.TwinCAT.HMI.BA.Framework).InstallPath/index.d.ts"
]
```

6.2.1.5.1 BA

6.2.1.5.1.1 BAObjectHandler

Der BaObjectHandler übernimmt die Verwaltung des BaObjects.

Verwendung

Die Implementierung erfolgt über die jeweiligen Interfaces.

IUsesBaObject

Bereitstellung des BaObjectHandlers.

Erfordert:

- TcHmiBaFramework

```
module MyNamespace {
    export class MyClass<A extends MyClass.IAttributes = MyClass.IAttributes> extends
TcHmi.BuildingAutomation.Base implements
TcHmi.BuildingAutomation.BaObjectHandler.IUsesBaObject {

        public baObjectHandler: TcHmi.BuildingAutomation.BaObjectHandler;

        constructor(id: string, parent: TcHmi.BuildingAutomation.IBaseNode | null, attr?: A) {
             super(id, parent, attr);

             this.baObjectHandler = new TcHmi.BuildingAutomation.BaObjectHandler(this);
        }

        public processBaObject() {
            if (this.baObjectHandler.baObject == null) return;
            // do work
        }
    }

    export module MyClass {
        export interface IAttributes extends TcHmi.BuildingAutomation.Base.IAttributes {
```

```
            // optional additional attributes
        }
    }
}
```

IFCUsesBaObject

Bereitstellung des BaObjectHandlers mit BaObject-Attribut für den TcHmi Designer.

Erfordert:

- TcHmiBaFramework
- TcHmiBaControls

```
module MyNamespace {
    export class MyControl extends TcHmi.BuildingAutomation.Controls.System.BaseControl
implements TcHmi.BuildingAutomation.BaObjectHandler.IFCUsesBaObject {

        public baObjectHandler: TcHmi.BuildingAutomation.BaObjectHandler;

        constructor(element: JQuery, pcElement: JQuery, attrs:
TcHmi.Controls.ControlAttributeList) {
            super(element, pcElement, attrs);

            this.baObjectHandler = new TcHmi.BuildingAutomation.BaObjectHandler(this);
        }

        public processBaObject() {
        if (this.baObjectHandler.baObject == null) return;
        // do work
        }

        public setBaObject(p: TcHmi.BuildingAutomation.BA.BaBasicObject |
TcHmi.BuildingAutomation.BA.BaBasicObject.IBaBasicObjectAttributes | TcHmi.Symbol | null |
undefined): this {
            this.baObjectHandler.setBaObject(p);

            return this;
        }

        public getBaObject() {
            return this.baObjectHandler.baObject;
        }
    }
}
```

Im Interface IFCUsesBaObject sind bereits die Setter und Getter für das BaObject definiert, weshalb in der *Description.json* nur noch das BaObject-Attribut definiert werden muss.

```
"attributes": [
    {
        "name": "data-tchmi-ba-object",
        "displayName": "BaObject",
        "propertyName": "BaObject",
        "propertySetterName": "setBaObject",
        "propertyGetterName": "getBaObject",
        "visible": true,
        "themeable": "None",
        "type": "tchmi:framework#/definitions/Symbol",
        "category": "BA",
        "description": "BA object of the control.",
        "requiredOnCompile": false,
        "readOnly": false,
        "bindable": true,
        "heritable": true,
        "defaultValue": null,
        "defaultValueInternal": null
    }
]
```

Eigenschaften

Name	Beschreibung
loadChildren	Legt fest, ob alle Kindelemente geladen werden beim Setzen des *BaObjects* (als BaView).
loadTexts	Legt fest, ob alle Texte geladen werden beim Setzen des *BaObjects* (als BaView).
enableParentBaObjectProcessor	Legt fest, ob der *BaObject* Prozessor des Eltern Controls aufgerufen wird.
baObject	Gibt das *BaObject* zurück.
baObjectSymbolExpression	Gibt die *SymbolExpression* vom *BaObject* zurück.
isLoadingBaObject	Prüft, ob das *BaObject* gesetzt, aber noch am Laden ist.

Methoden

Name	Beschreibung
setBaObject	Setzt das *BaObject*.
resolveBaObject	Löst die übergebenen Informationen auf, um ein *BaObject* zu erzeugen.
readBaObject	Liest das BaObject vom Server.
watchBaVariable	Fügt einen Watch zur übergebenen BaVariable hinzu. Der Watch wird beim Zerstören der Klasse ebenfalls zerstört.
tryWatchBaVariable	Versucht eine BaVariable zu überwachen.
tryWatchChildrenBaVariable	Versucht eine BaVariable eines Kindelements eines BaViews zu überwachen.
watchValueRange	Überwacht den Wertebereich einer spezifischen Variablen.
checkBaObjectAccess	Prüft den OperationType und den Schreibzugriff der BaVariablen *ValueRm*.

Ereignisse

Name	Beschreibung
onBaObjectChanged	Wird ausgelöst, wenn sich das *BaObject* geändert hat.

6.2.1.5.2 Helper

6.2.1.5.2.1 BaInterfaceHandler

Der BaInterfaceHandler ist eine Helferklasse, die die Verwaltung des *BaInterfaces* eines Controls übernimmt.

Mit Hilfe des *BaInterfaces* ist es möglich, mehrere Datenpunkte von einem Control mit nur einem Binding zu verknüpfen.

BaInterface

Das BaInterface-Attribut wird mit einem TcHmi-Symbol verbunden. Dieses Symbol muss einen bestimmten Aufbau haben, der durch die BaInterface-Definition beschrieben wird.

Beispiel

Das *BaInterface* einer Checkbox hat folgenden Aufbau:

```
export type BaInterface = {
    state: boolean,
    stateFeedback?: boolean,
    activeText?: string,
    inactiveText?: string,
}
```

Damit das verbundene Symbol für das *BaInterface* der Checkbox gültig ist, muss das Symbol die beschriebenen Untersymbole mit dem entsprechenden Datentyp haben.

BaInterfaceDefinition

Ein Control, das den BaInterfaceHandler verwendet, muss das Interface IUsesBaInterface<T> implementieren.

Der Typ-Parameter `T` soll dabei den Aufbau des verwendeten *BaInterfaces* beschreiben.

Hier am Beispiel des Datentyps `Checkbox.BaInterface`:

```
export class Checkbox implements IUsesBaInterface<Checkbox.BaInterface>
```

Da TypeScript Typen zur Laufzeit von JavaScript nicht interpretiert werden können, ist es notwendig die BaInterfaceDefinition als Konstante zu definieren.

Hier muss der Datentyp des jeweiligen Elements angegeben werden und über die Eigenschaft *optional* können optionale Elemente des *BaInterfaces* definiert werden.

Optionale Elemente werden bei der späteren Validierung des *BaInterfaces* nicht berücksichtigt. Falls benötigte Elemente bei der Validierung nicht gefunden werden, kommt es zu Fehlermeldungen und das *BaInterface* wird nicht weiterverarbeitet.

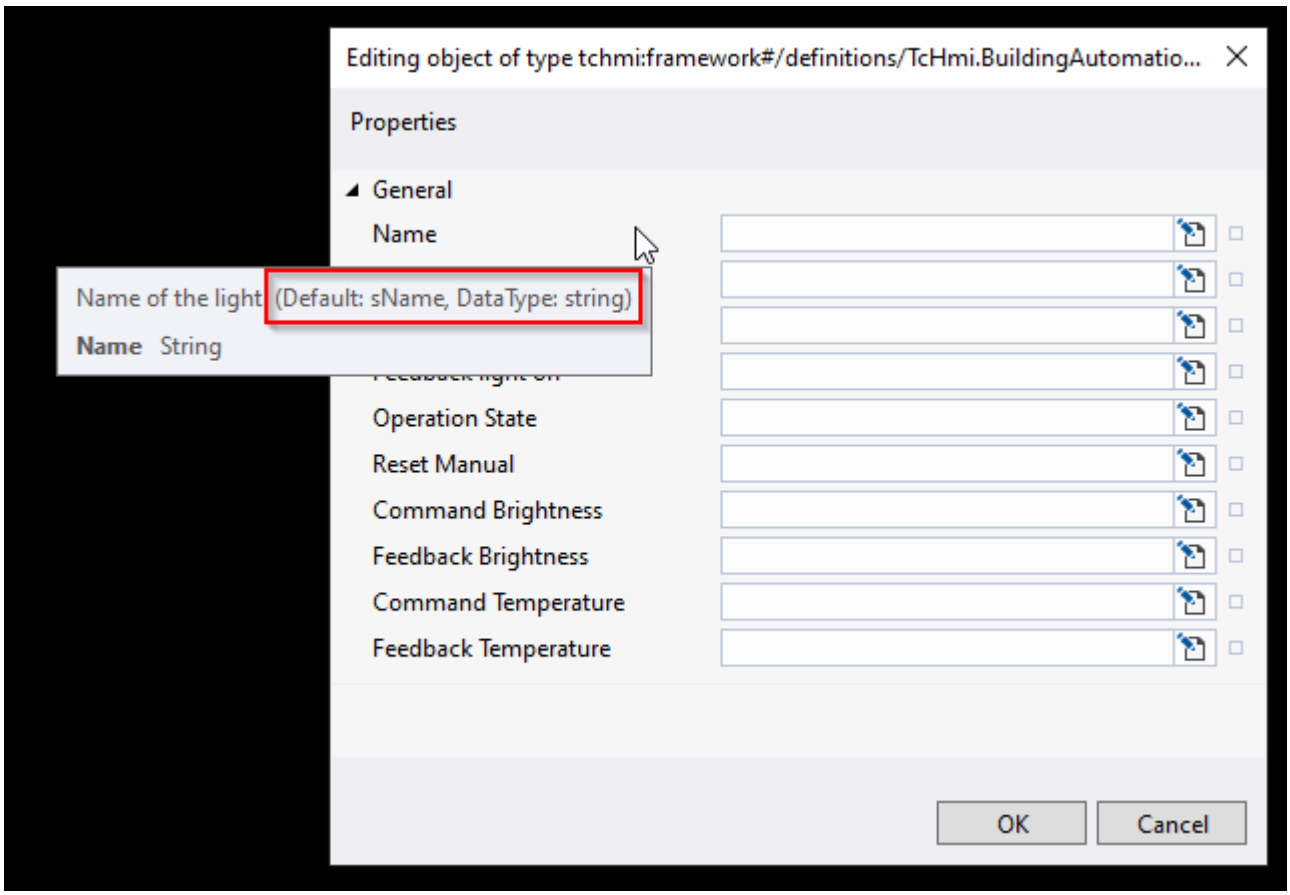
```
export const BaInterfaceDef: BaInterfaceDefinition<BaInterface> = {
    state: {
        type: 'boolean'
    },
    stateFeedback: {
        type: 'boolean',
        optional: true
    },
    activeText: {
        type: 'string',
        optional: true
    },
    inactiveText: {
        type: 'string',
        optional: true
    }
};
```

Es ist möglich mehrere Datentypen anzugeben.

```
command: {
    type: ['number', 'boolean']
}
```

BaInterfaceSymbolNames

Die Standardwerte der BaInterfaceSymbolNames, sowie die erwarteten Datentypen sind im Tooltipp des Dialoges zum Einstellen der BaInterfaceSymbolNames zu finden.



Damit der BaInterfaceHandler auf die entsprechenden Untersymbole des *BaInterfaces* zugreifen kann, ist es notwendig für jedes Element des Interfaces einen Symbolnamen anzugeben. Der Symbolname entspricht dabei dem Namen der Variablen in dem verbundenen Funktionsblock / Struktur. Dies kann z. B. durch eine Variable `Checkbox.BaInterfaceSymbolNames` geschehen:

```
export let BaInterfaceSymbolNames: BaInterfaceSymbolNames<BaInterface> = {
    state: {
      symbolName: 'State'
    },
    stateFeedback: {
      symbolName: 'StateFeedback'
    },
    activeText: {
      symbolName: 'ActiveText'
    },
    inactiveText: {
      symbolName: 'InactiveText'
    }
}
```

Der Symbolname soll bei der späteren Verwendung angepasst werden können, sofern das Symbol in dem verbundenen Funktionsblock/Struktur anders heißt (z. B. `symbolName: "bState"`).

Ebenfalls ist es möglich Untersymbole zu nutzen. Liegt der State z. B. in einer Struktur oder einem Funktionsblock innerhalb des verbundenen Symbols kann auf diesen zugegriffen werden (z. B. `symbolName: "Command::bValueRm"`).

Für das Überschreiben dieser Standard-Symbolnamen gibt es zwei Möglichkeiten, die im Folgenden erläutert werden.

Attribut BaInterfaceSymbolNames

Da im Interface IUsesBaInterface<T> bereits die Setter und Getter für die BaInterfaceSymbolNames definiert sind, kann in der Description.json das folgende Attribut definiert werden, damit die Symbolnamen aus dem Designer heraus geändert werden können:

```
{
  "name": "data-tchmi-ba-interface-symbol-names",
  "displayName": "BaInterfaceSymbolNames",
```

```
  "propertyName": "BaInterfaceSymbolNames",
  "propertySetterName": "setBaInterfaceSymbolNames",
  "propertyGetterName": "getBaInterfaceSymbolNames",
  "visible": true,
  "themeable": "None",
  "displayPriority": 61,
  "type": "tchmi:framework#/definitions/
TcHmi.BuildingAutomation.Common.Checkbox.BaInterfaceSymbolNames",
  "category": "BaData",
  "description": "Symbol names for the interface symbol.",
  "requiredOnCompile": false,
  "readOnly": false,
  "bindable": true,
  "heritable": true,
  "defaultValue": null,
  "defaultValueInternal": null
}
```

Der

```
type
```

ist dabei in der Types.Schema.json zu definieren.

Überschreiben in onInitialized

Wenn die Symbolnamen für alle Controls überschrieben werden sollen, ist es nicht praktikabel das BaInterfaceSymbolNames-Attribut von jedem Control einzeln zu editieren.

Die Standard-Symbolnamen können global im onInitialized Event überschrieben werden. Dazu wird eine CodeBehind-Funktion mit folgendem Inhalt angelegt:

```
module TcHmi {
    var destroyOnInitialized = TcHmi.EventProvider.register('onInitialized', function (e, data) {
        e.destroy();

        // overwrite BaInterfaceSymbolNames of the Checkbox
        BuildingAutomation.Controls.Checkbox.BaInterfaceSymbolNames = {
            state: {
                symbolName: 'bState'
            },
            stateFeedback: {
                symbolName: 'bStatFdb'
            },
            activeText: {
                 symbolName: 'sActiveText'
            },
            inactiveText: {
                symbolName: 'sInactiveText'
            }
        }
    });
}
```



Es ist nicht notwendig die Symbolnamen aller Elemente zu setzen. Es sind nur die Elemente zu setzen, die nicht als optional gekennzeichnet wurden.

Initialisierung

Der BaInterfaceHandler muss in der `__prevInit()` Methode des Controls initialisiert werden. Bei der Initialisierung sind folgende Schritte zu durchlaufen.

```
// Create instance of BaInterfaceHandler
this.baInterfaceHandler = new BaInterfaceHandler<Checkbox.BaInterface>(this);
// Set the BaInterfaceDefinition
this.baInterfaceHandler.baInterfaceDefinition = Checkbox.BaInterfaceDef;
// Set the default symbol names
this.setBaInterfaceSymbolNames(Checkbox.BaInterfaceSymbolNames);
```

Verwendung

Die Implementierung des Setters für das BaInterface-Attribut kann wie folgt aussehen:

```
public setBaInterface(p: BaInterfaceSymbol<Checkbox.BaInterface> | null | undefined): this {
    this.baInterfaceHandler.setBaInterfaceSym(p, () => {
        // do work with the validated BaInterface
    });
    return this;
}

public getBaInterface() {
    return this.baInterfaceHandler.getBaInterfaceSym();
}
```

Hier ist zu sehen, dass für den Setter die Methode `setBaInterfaceSym()` des BaInterfaceHandlers verwendet wird. Diese erwartet neben dem Symbol auch noch eine Processor-Methode, die aufgerufen wird, wenn das BaInterfaceSymbol validiert wurde oder sich die BaInterfaceSymbolNames geändert haben.

Die Setter und Getter für das BaInterfaceSymbolNames-Attribut können wie folgt implementiert werden:

```
public setBaInterfaceSymbolNames(p: BaInterfaceSymbolNames<Checkbox.BaInterface> |
 BaInterfaceSymbolNamesDesigner | null | undefined): this {
    if (p != null)
        this.baInterfaceHandler.updateSymbolNames(BaInterfaceHandler.convertToBaInterfaceSymbolNames
(p));
    return this;
}

public getBaInterfaceSymbolNames(): BaInterfaceSymbolNames<Checkbox.BaInterface> | null |
 undefined {
    return this.baInterfaceHandler.baInterfaceDescription;
```

Methoden

Im Folgenden werden die wichtigsten Methoden des BaInterfaceHandlers beschrieben.

Name	Beschreibung
hasSubSymbol	Überprüft, ob das BaInterface ein bestimmtes Unterelement hat. Diese Methode muss verwendet werden, wenn optionale Elemente gelesen oder geschrieben werden sollen.
writeSubSymbol	Schreibt den Wert eines Unterelements.
watchSubSymbol	Legt eine Subscription für ein Unterelement an.
updateSymbolNames	Aktualisiert die Symbolnamen der Unterelement des BaInterfaces.
convertToBaInterfaceSymbolNames	Konvertiert den Datentyp `BaInterfaceSymbolNamesDesigner` in `BaInterfaceSymbolNames`. Wird der Methode bereits der Datentyp `BaInterfaceSymbolNames` übergeben, wird keine Konvertierung durchgeführt und das Objekt direkt zurückgegeben.

6.2.1.5.3 BusyHandler

Der BusyHandler ist eine Klasse vom TcHmiBaFramework [▶ 1263]. Er gibt Auskunft darüber, ob ein Control noch beschäftigt ist (z. B. warten auf Informationen vom TwinCAT HMI Server). Zu erkennen an der Ladeanimation.



Funktionen

logTimerResultsOfControl

Prüft, welche Aktionen an einem Control zu Ladezeiten führen.

Namespace: `TcHmi.BuildingAutomation.BusyHandler.logTimerResultsOfControl`

Ist nur auf Controls anwendbar, die das Interface TcHmi.BuildingAutomation.BusyHandler.IBusyHandler implementieren.

Vorbereitung

Vor dem Einsatz muss das Aufzeichnen von Timer-Ergebnissen aktiviert werden. Das kann z. B. auf Projektebene mit einer Code-Behind Funktion geschehen:

```
let TcHmi.EventProvider.register('onInitialized', function (e, data) {
  e.destroy();
  TcHmi.BuildingAutomation.BusyHandler.RecordTimerResults = true;
}
```

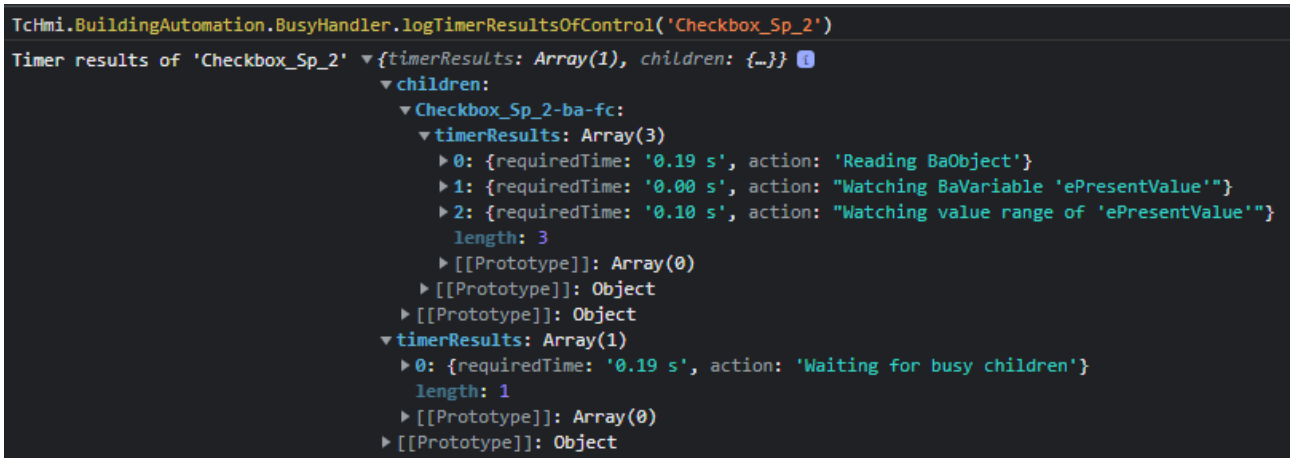
Verwendung

Der Aufruf erfolgt in der Konsole des Browsers, nachdem ein Control geladen wurde. Hierfür ist die ID des Controls notwendig.

```
TcHmi.BuildingAutomation.BusyHandler.logTimerResultsOfControl('DieControlId')
```

Auswertung

Die Ergebnisse der Timer stehen im Konsolenfenster bereit:



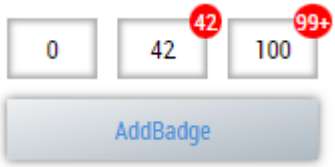
Es ist erkennbar, womit das Control beschäftigt war.

In diesem Fall wurde auf ein Unterelement mit der ID *Checkbox_Sp_2-ba-fc* gewartet, das die meiste Zeit mit dem Laden des BaObjects beschäftigt war.

6.2.1.6 Functions

6.2.1.6.1 AddBadge

Fügt einem Control in der oberen rechte Ecke eine Zahlanzeige hinzu. Die Darstellung ist für TcHmiBa Controls optimiert.



Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

Control

```
tchmi:framework#/definitions/Control
```

Control für die Badge Erweiterung.

Count

```
tchmi:general#/definitions/Number
```

Anzuzeigende Zahl. Ist sichtbar, wenn der Wert größer 0 ist. Beschränkung auf "99+" ab dreistelligen Werten.

6.2.1.6.2 ConvertHexToRgbaColor

Konvertiert eine Hex-Farbe in eine RGBA-Farbe.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

Hex

```
tchmi:general#/definitions/String
```

Zu konvertierende Hex-Farbe.

Rückgabewert

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA-Farbe.

6.2.1.6.3 ConvertHslToRgbaColor

Konvertiert eine HSL-Farbe in eine RGBA-Farbe.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

HSL

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.HSLColor
```

Zu konvertierende HSL-Farbe.

Rückgabewert

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA-Farbe.

6.2.1.6.4 ConvertRgbaToHexColor

Konvertiert eine RGBA-Farbe in eine Hex-Farbe.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

RGBA

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

Zu konvertierende RGBA-Farbe.

Rückgabewert

```
tchmi:general#/definitions/String
```

Hex-Farbe.

6.2.1.6.5 ConvertRgbaToHslColor

Konvertiert eine RGBA-Farbe in eine HSL-Farbe.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

RGBA

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

Zu konvertierende RGBA -Farbe.

Rückgabewert

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.HSLColor
```

HSL-Farbe.

6.2.1.6.6 ConvertRgbaToSolidColor

Konvertiert eine RGBA-Farbe in eine TcHmi Solid-Farbe.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

RGBA

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

Zu konvertierende RGBA -Farbe.

Rückgabewert

```
tchmi:framework#/definitions/SolidColor
```

TcHmi Solid-Farbe.

6.2.1.6.7 ConvertSolidToRgbaColor

Konvertiert eine TcHmi Solid-Farbe in eine RGBA-Farbe.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

Solid

```
tchmi:general#/definitions/SolidColor
```

Zu konvertierende TcHmi Solid-Farbe.

Rückgabewert

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Color.RGBAColor
```

RGBA-Farbe.

6.2.1.6.8 ConvertUnitToString

Konvertiert den Wert einer BA.Unit Enumeration in eine Einheit.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

Unit

```
tchmi:general#/definitions/Number
```

Teilpfad vom Icon.

Rückgabewert

```
tchmi:general#/definitions/String
```

Einheit.

6.2.1.6.9 GetBaIconPath

Gibt zu einem Teilpfad den vollständigen Pfad eines Icons zurück.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Der Teilpfad für den Funktionsparameter entspricht dem Aufbau des NuGet-Pakets Beckhoff.TwinCAT.HMI.BA.Icons, wie es im TwinCAT HMI Gallery Explorer zu sehen ist.

Gültige Aufrufmöglichkeiten:

```
GetBaIconPath('HVAC/Cooler.svg')
GetBaIconPath('HVAC/cooler.svg')
GetBaIconPath('HVAC/Cooler')
GetBaIconPath('HVAC/cooler')
```

Parameter

Path

```
tchmi:general#/definitions/String
```

Teilpfad vom Icon.

Rückgabewert

```
tchmi:general#/definitions/String
```

Vollständige Pfad vom Icon.

6.2.1.6.10 GetCurrentUserName

Gibt den Namen das aktiven Benutzers zurück (keine Authentifizierung entspricht *__SystemGuest*) oder Null, wenn unbekannt (z. B. beim Laden).

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Rückgabewert

```
tchmi:general#/definitions/String
```

Benutzername.

6.2.1.6.11 GetFadeColor

Berechnet eine Farbe zwischen zwei Farben zu einem entsprechenden Wert.

Verwendung

Die Funktion kann sowohl ereignisgesteuert als auch über das Function Binding aufgerufen werden und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

StartColor

```
tchmi:framework#/definitions/SolidColor
```

Startfarbe der Überblendung.

StartValue

```
tchmi:framework#/definitions/Number
```

Startwert der Überblendung.

EndColor

```
tchmi:framework#/definitions/SolidColor
```

Endfarbe der Überblendung.

EndValue

```
tchmi:framework#/definitions/SolidColor
```

Endwert der Überblendung.

Value

```
tchmi:framework#/definitions/Number
```

Wert zur Berechnung der Farbe.

Rückgabewert

```
tchmi:general#/definitions/SolidColor
```

Berechnete Farbe.

6.2.1.6.12 LoadUserDependentContent

Lädt einen bestimmten Inhalt, wenn der angegebene Benutzer eingeloggt ist.

Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

HostRegion

```
tchmi:framework#/definitions/TcHmi.Controls.System.TcHmiRegion
```

Die Hostregion, in die der Inhalt geladen werden soll.

UserContents

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.UserContents
```

Sammlung von Benutzern mit den dazugehörigen Inhalten.

StoreLastContent

```
tchmi:general#/definitions/Boolean
```

Leg fest, ob der letzte Inhalt beim nächsten Besuch des Benutzers erneut geladen wird.

6.2.1.6.13 OpenBaTrend

Öffnet einen Dialog, in dem der Trend von einem verknüpften BaObject oder die Trends von einem BaView angezeigt werden.

Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

BaObject

```
tchmi:framework#/definitions/ContentPath
```

Das BaObject / der BaView, dessen Trends angezeigt werden.

6.2.1.6.14 OpenDialogWindow

Öffnet eine Content-Seite in einem Dialog.

Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

Content

```
tchmi:framework#/definitions/ContentPath
```

Pfad zum anzuzeigenden Inhalt.

Buttons

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.DialogWindowButtons
```

Buttons zum Schließen oder Bestätigen des Dialogs.

Modal

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Dialog modal geöffnet wird oder nicht.

Scrolling

```
tchmi:framework#/definitions/ScrollMode
```

Legt fest, ob der Inhalt gescrollt werden kann.

Headline

```
tchmi:general#/definitions/String
```

Titel des Dialogs.

Width

```
tchmi:general#/definitions/Number
```

Breite des Dialogs.

WidthUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Einheit der Breite des Dialogs.

Height

```
tchmi:general#/definitions/Number
```

Höhe des Dialogs.

HeightUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Einheit der Höhe des Dialogs.

6.2.1.6.15 OpenLegendDialog

Öffnet eine Instanz des Legend [▶ 1143]-Controls in einem Dialog.

Verwendung

Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

Modal

```
tchmi:general#/definitions/Boolean
```

Legt fest, ob der Dialog modal geöffnet wird oder nicht.

Buttons

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.DialogWindowButtons
```

Buttons zum Schließen oder Bestätigen des Dialogs.

Width

```
tchmi:framework#/definitions/MeasurementValue
```

Breite des Dialogs.

WidthUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Einheit der Breite des Dialogs.

Height

```
tchmi:framework#/definitions/MeasurementValue
```

Höhe des Dialogs.

HeightUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Einheit der Höhe des Dialogs.

EntryWidth

```
tchmi:framework#/definitions/MeasurementValue
```

Breite eines Eintrags.

EntryWidthUnit

```
tchmi:framework#/definitions/MeasurementUnit
```

Einheit der Breite eines Eintrags.

TabPosition

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Position
```

Position der Tabs.

IconDataSource

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataSource
```

Auswahl anzuzeigender Einträge.

IconDataCustom

```
tchmi:framework#/definitions/TcHmi.BuildingAutomation.Controls.BuildingGeneral.Legend.IconDataList
```

Liste mit zusätzlichen Einträgen.

6.2.1.6.16 OpenLightZoneDialog

Öffnet eine Instanz des LightZone-Controls in einem Dialog.

Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

Parameter

BaObject

```
tchmi:framework#/definitions/Symbol
```

BaObject für das Control.

6.2.1.6.17 OpenTrendCollectionView

Öffnet einen Dialog zum Beobachten von Trendsammlungen [▶ 59].

Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

6.2.1.6.18 UpdateObjectInfo

Aktualisiert die Objektinformationen aller Objekte in allen BA-Geräten.

Verwendung

Die Funktion wird ereignisgesteuert aufgerufen und ist im Actions and Conditions Editor unter *Functions* > *BuildingAutomation* zu finden.

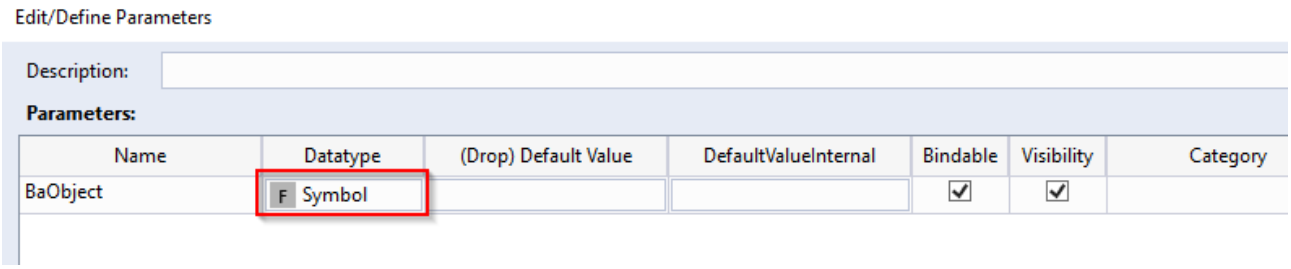
6.2.1.6.19 UseBaObjectsInUserControl

Beschreibung

Ein UserControl liest standardmäßig die komplette Struktur aus die hinter einem verknüpften Parameter liegt, was zu einem hohen Kommunikationsaufkommen führen kann. Bei einem BaObjekt (z.B. BaView) können das schnell sehr viele Daten werden.
Diese Funktion reduziert, bei Verwendung eines BaObjekts als Parameter in einem UserControl, die Server-Kommunikation auf ein Minimum.

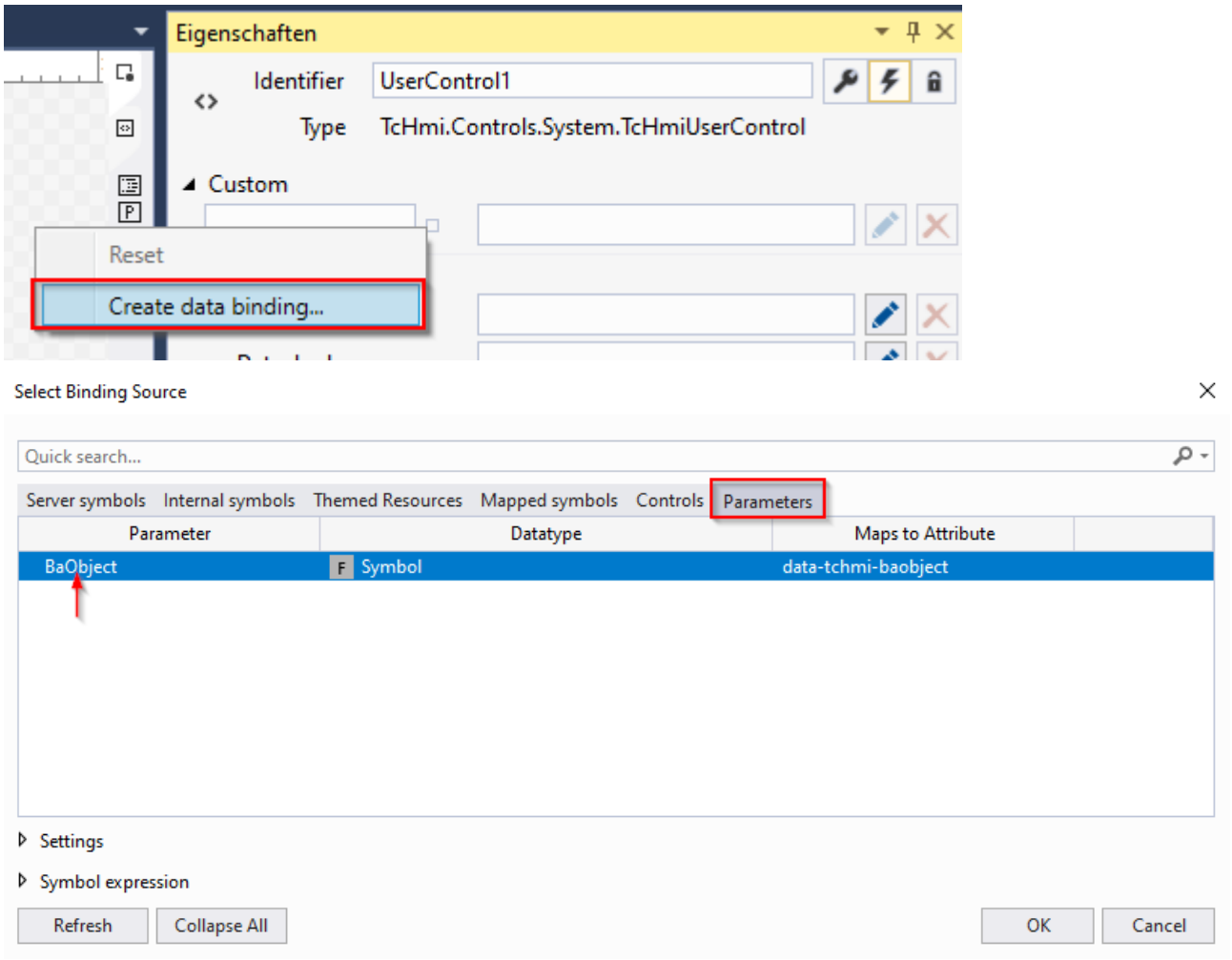
Verwendung

Der Parameter vom UserControl für das BaObjekt muss vom Typ *Symbol* sein. Durch diese Definition wird der Parameter nicht ausgelesen, sondern nur weitergegeben.
Ebenfalls muss der Name des Parameters *BaObject* heißen!

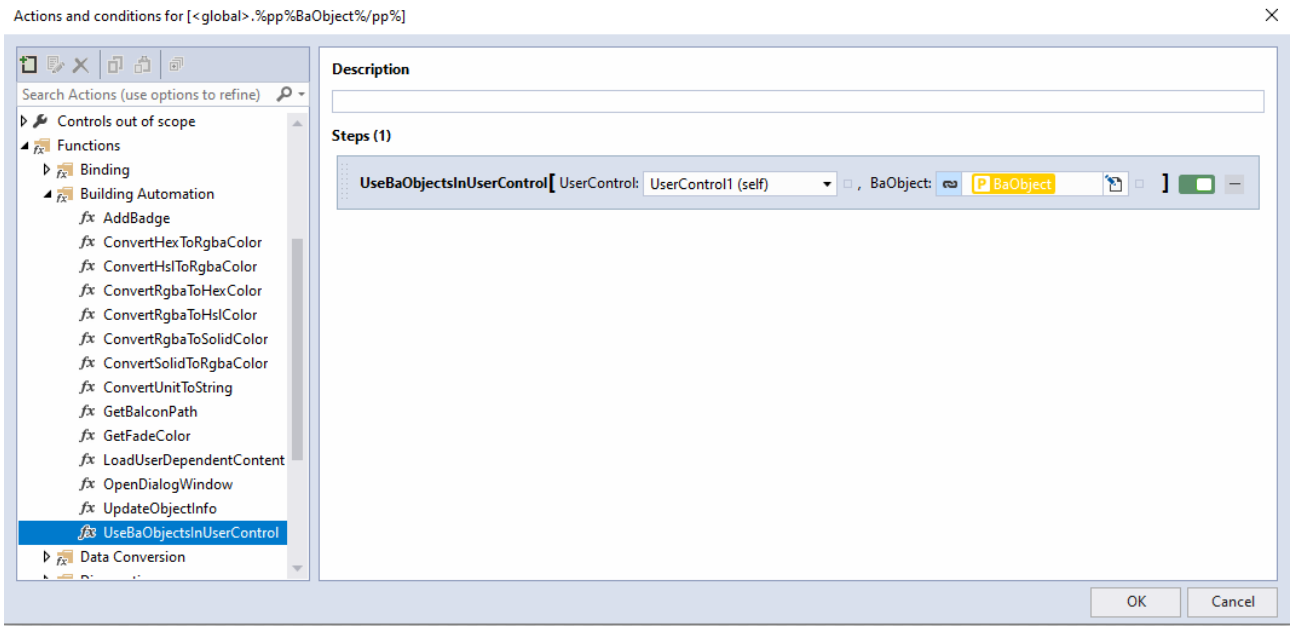


Auf diese Weise ist kein Zugriff mehr auf die unterliegenden Symbole des BaObjekts möglich.

Der Aufruf der Funktion soll immer dann erfolgen, wenn der BaObject-Parameter geändert wurde. Dazu wird ein neues Event *.BaObject* erzeugt.



Im Konfigurationsfenster des Events wird die Funktion im Ordner *"Functions > BuildingAutomation > UseBaObjectInUserControl"* ausgewählt.



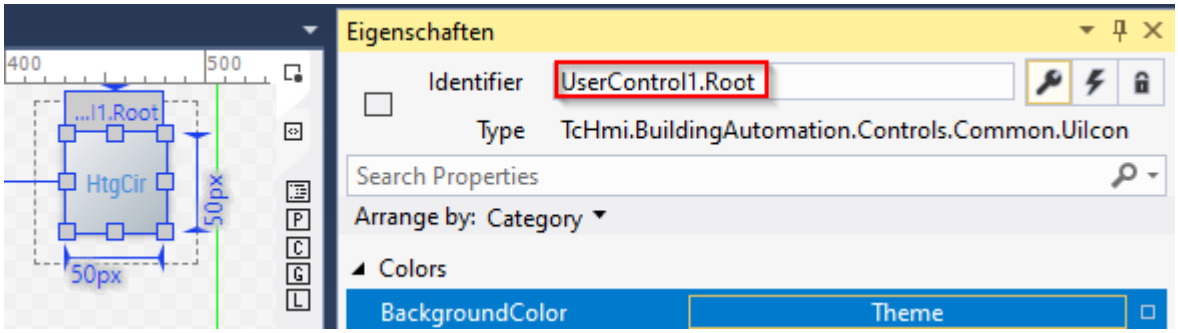
Die Verknüpfung der Controls im UserControl mit dem BaObjekt oder dessen Unterelementen erfolgt dann über die Identifier der Controls.

Es ist ebenfalls möglich UserControls innerhalb eines UserControls mit BaObjekten zu verbinden. Dabei muss der Identifier, wie im nächsten Absatz erklärt, vergeben werden. Ebenfalls muss das innere UserControl dann auch den Parameter *BaObject* haben.

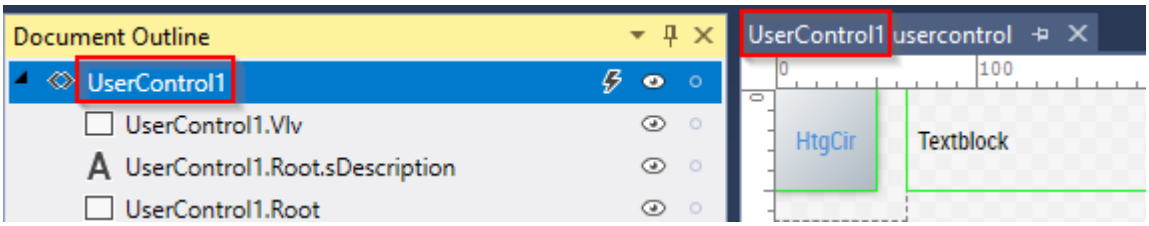
Verknüpfung

BaObjekt

Zur direkten Verwendung des BaObjekts muss das Control den Identifier *$UserControlName$.Root* tragen. *$UserControlName$* wird dabei mit dem Namen des UserControls ersetzt.



Es ist darauf zu achten, dass das Wurzelelement des UserControls, den selben Namen hat, wie die Datei des UserControls.

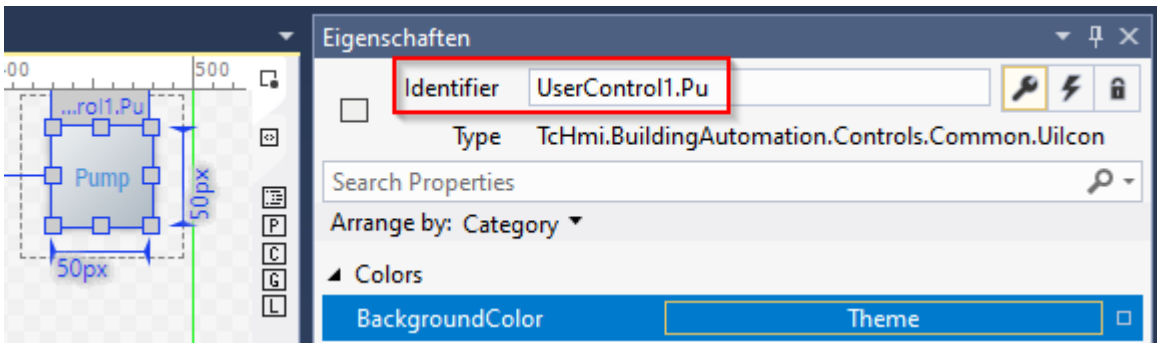


Unterelemente vom BaObjekt

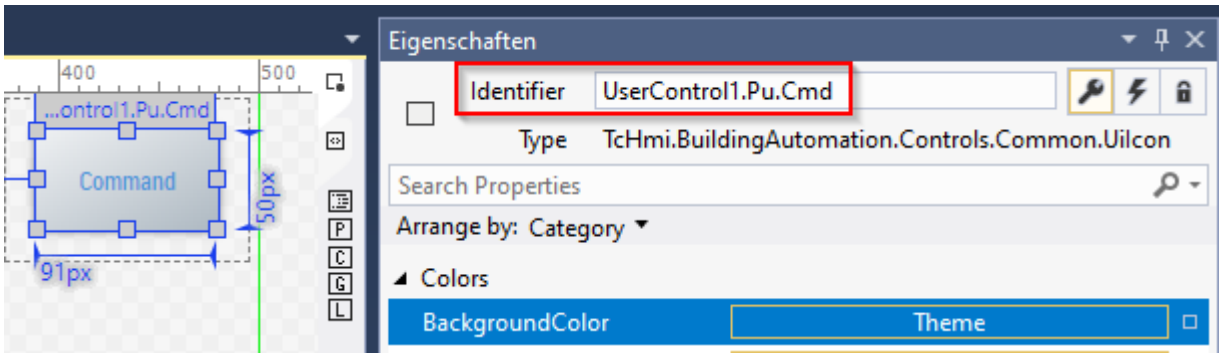
Zur Verwendung eines Unterelements muss das Control den Symbolpfad als Identifier tragen. Erläuterung am Beispiel eines BaView mit folgendem Aufbau.



Zugriff auf das Element *Pu* erfolgt über den Symbolpfad *$UserControlName$.Pu* als Identifier.



Zugriff auf das Element *Cmd* erfolgt über den Symbolpfad *$UserControlName$.Pu.Cmd* als Identifier.



Parameter vom BaObjekt

Zugriff auf den Parameter eines Elements erfolgt über den Symbolpfad.

Zum Beispiel:

- *$UserControlName$.Root.sDescription*
- *$UserControlName$.Pu.Cmd.bPresentValue*
- *$UserControlName$.OpMod.OpModMan.nPresentValue*
- *$UserControlName$.HtgLmt.Sp.fPresentValue*

Ist das Control vom Typ Checkbox [▶ 1154], Combobox [▶ 1158] oder InputBox [▶ 1163], dann wird nach dem Beenden der Benutzerinteraktion der Wert des Parameters in die SPS geschrieben.

Ohne BaObjekt oder BaParameter

Soll das Control oder UserControl innerhalb des UserControls nicht mit einem BaObjekt oder BaParameter arbeiten, dann darf der Identifier des Controls nicht *$UserControlName$.* enthalten.

Parameter

UserControl

```
tchmi:framework#/definitions/TcHmi.Controls.System.TcHmiUserControl
```

Das UserControl in dem die BaObjekte verwendet werden sollen. Meistens ist dieser Parameter das UserControl selbst (self).

BaObject

```
tchmi:general#/definitions/Object
```

Mit diesem Parameter muss der UserControl-Parameter, mit dem das BaObjekt von außen verbunden wird, verknüpft werden.

6.2.1.7 Servererweiterungen

6.2.1.7.1 BaSiteExtension

Beschreibung

Die BaSite-Extension dient als Schnittstelle zwischen einer TF8040 SPS (TF8040 Getting Started [▶ 66]) und einem TcHmi-Client. Die Erweiterung ermöglicht es generische Funktionen anzubieten, die das Engineering des HMIs deutlich vereinfachen und beschleunigen.



Verknüpfungen zwischen der SPS und dem HMI-Server sind auch ohne die Erweiterung weiterhin über die ADS-Extension möglich. Es entfallen dann lediglich die genannten Vorteile.

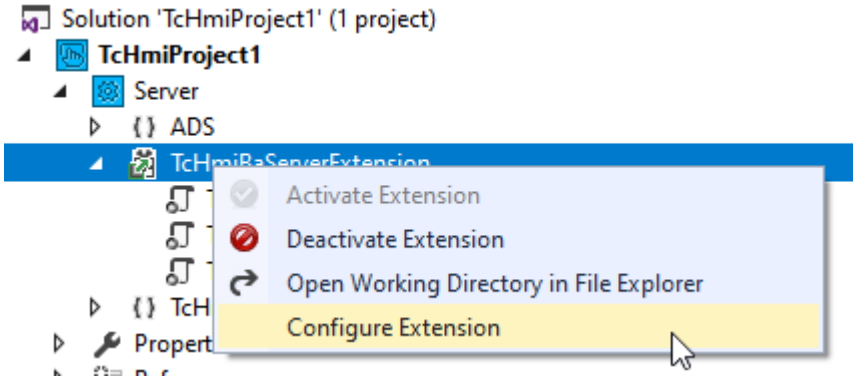
Verwendung

Um die generischen Funktionalitäten in einem Projekt nutzen zu können, muss die Erweiterung [▶ 79] installiert werden.

Konfiguration

Die Konfiguration der Erweiterung kann im TcHmi-Engineering oder über die Konfigurationsseite des HMI-Servers erfolgen.

Konfiguration aus dem TcHmi-Engineering:



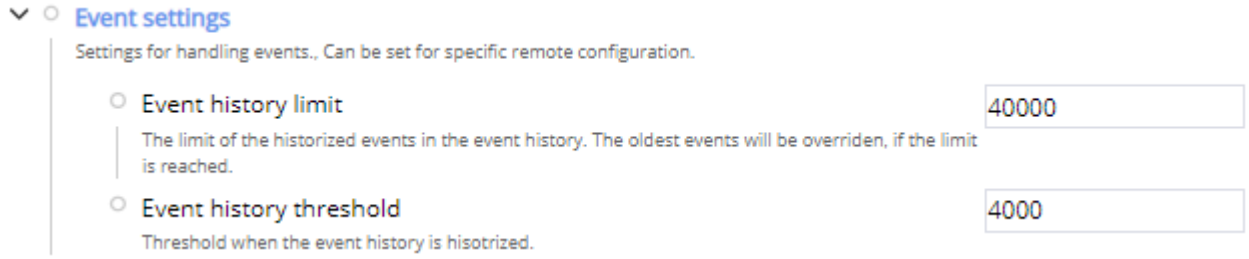
Allgemein

Intervall

Diese Zeit legt fest, wie häufig Variablen in der Oberfläche gelesen (aktualisiert) werden.

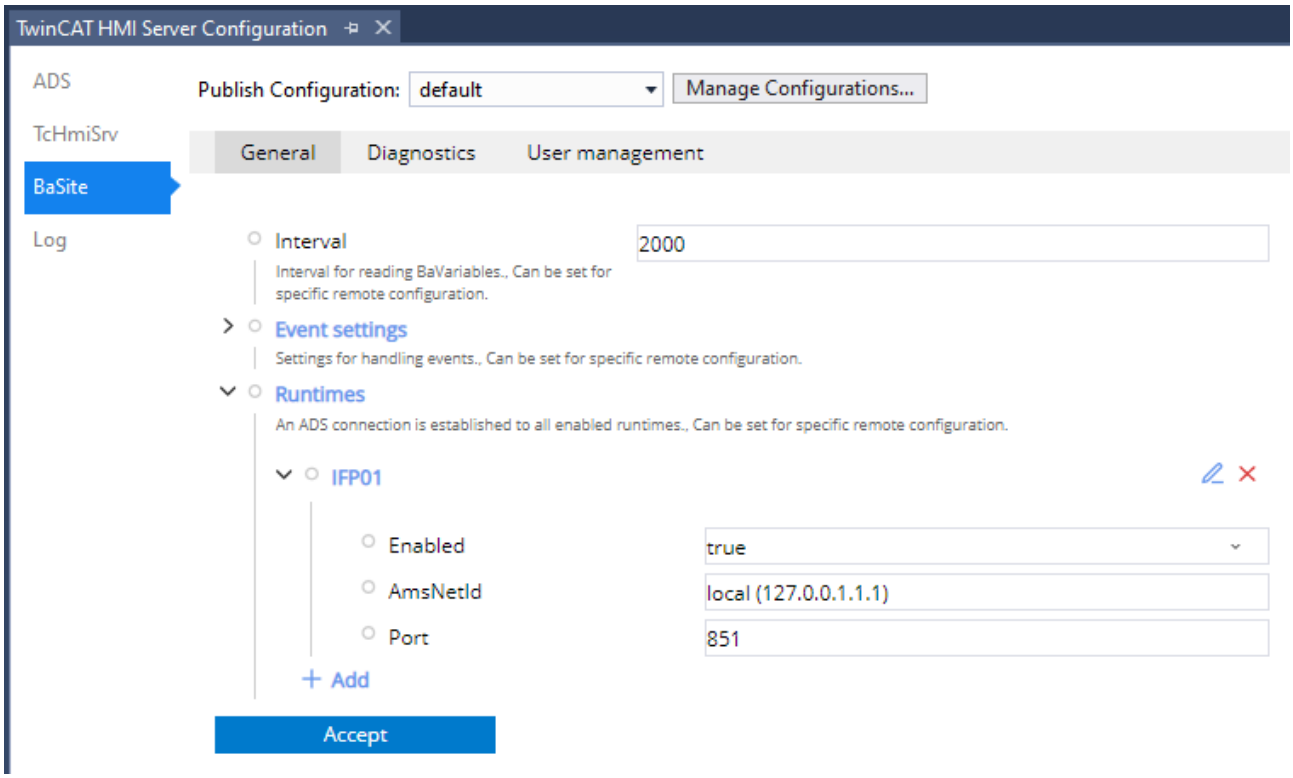
Eventeinstellung

Diese Einstellungen beeinflussen das Verhalten der Eventhistorie. Es kann eingestellt werden, wie viele Events in der Eventhistorie gespeichert werden und ab welcher Anzahl die Events historisiert werden.



Geräte

Der Erweiterung müssen die Geräte mitgeteilt werden, mit denen sich das HMI verbinden soll. Über die Konfigurationsmaske lassen sich die Geräte hinzufügen bzw. auch wieder entfernen.



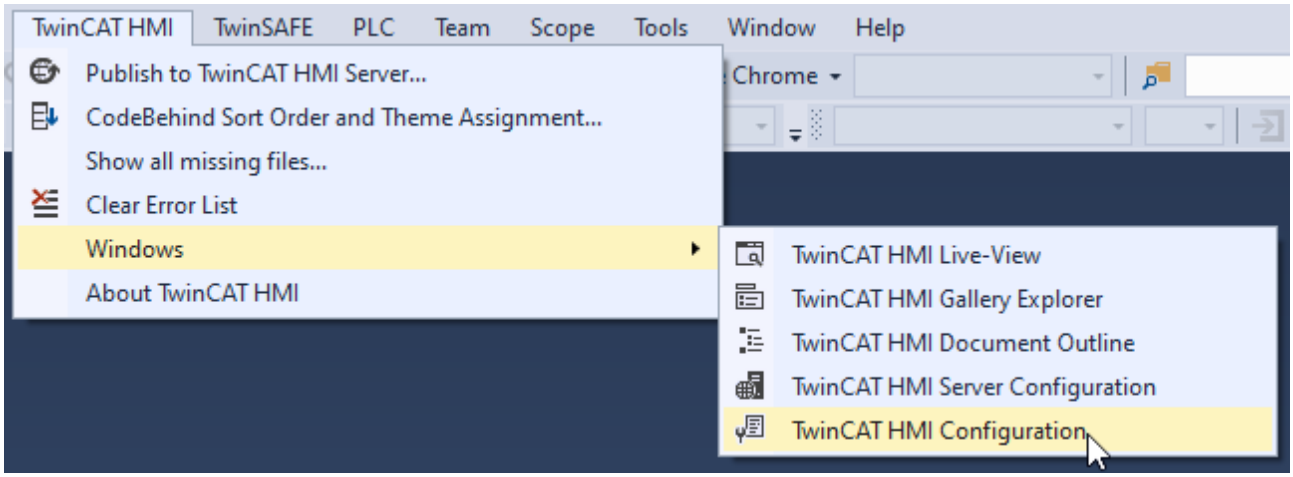
Erforderliche Einstellungen pro Gerät:

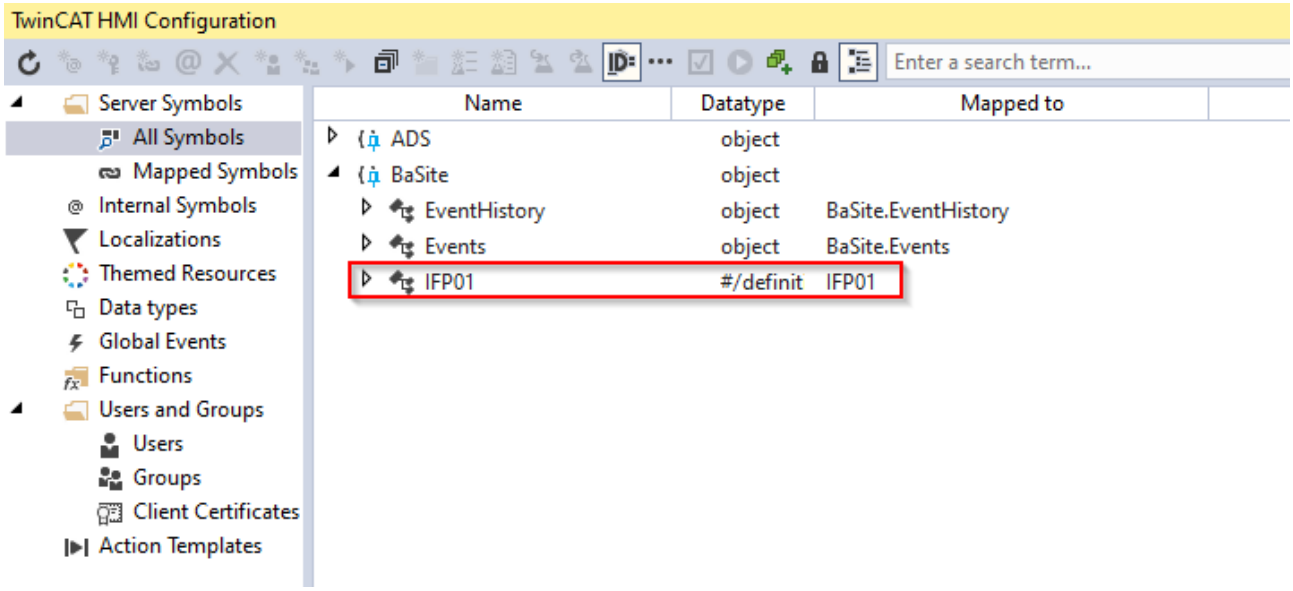
- Festlegung, ob aktiv/inaktiv
- AmsNetId
- SPS Port

Die Konfiguration wird durch einen Klick auf **Akzeptieren** aktiviert.

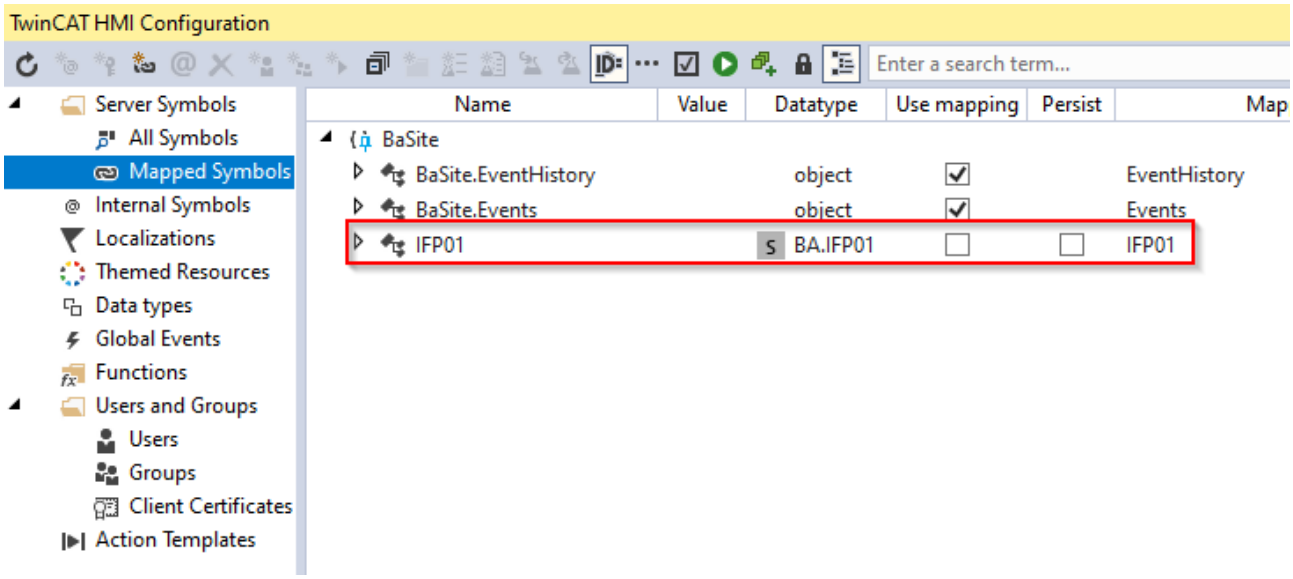
Überprüfen der Konfiguration

Im **TwinCAT HMI Configuration** Fenster sollten nun alle Geräte unter **All symbols** aufgelistet werden.





Ebenfalls wurde ein Mapping für das Gerät angelegt.



Notwendige Mappings

Das automatisch pro Gerät angelegte Mapping mit **demselben** Namen, wie das Gerät selbst, ist zwingend erforderlich und darf **nicht** umbenannt oder gelöscht werden.

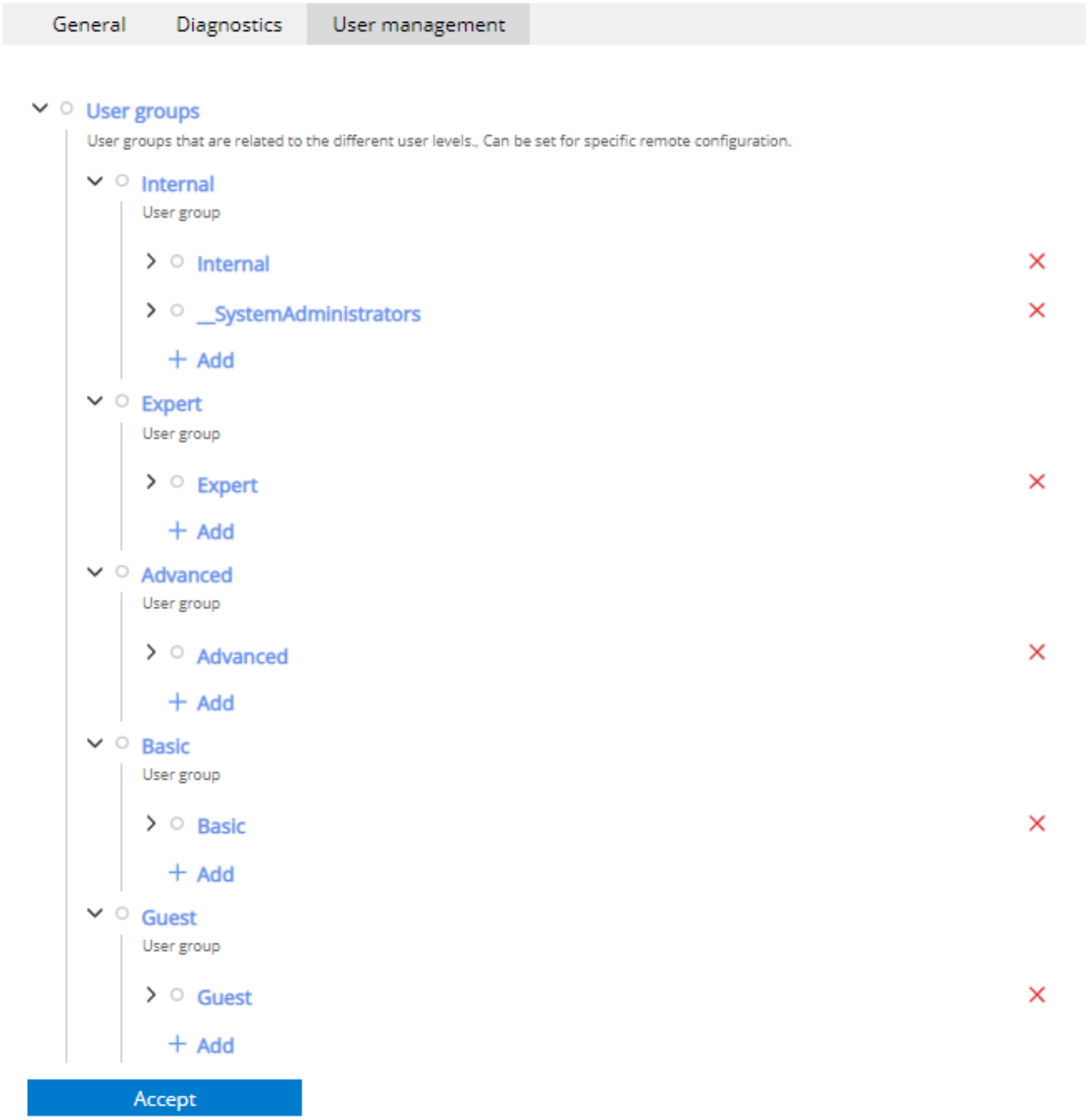
Durch die Aktivierung der Konfiguration wurde ebenfalls automatisch ein gleichnamiges Gerät mit denselben Einstellungen in der ADS-Extension hinzugefügt. Auch dieses Gerät in der ADS-Extension ist zwingend erforderlich und darf **nicht** umbenannt oder gelöscht werden.

Benutzerverwaltung

Benutzergruppen

Hier können den Benutzergruppen *_Guest_*, *_Basic_*, *_Advanced_*, *_Expert_* und *_Internal_* jeweils Benutzergruppen aus der HMI zugeordnet werden.

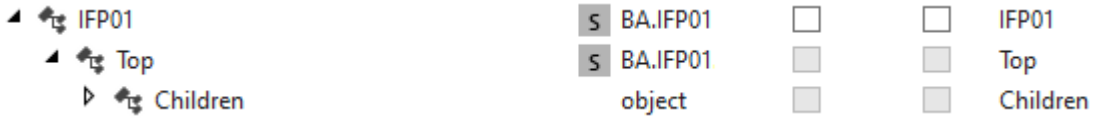
Es kann konfiguriert werden, dass verschiedene Benutzergruppen denselben Benutzerlevel erhalten.



Symbole

Gerätesymbole

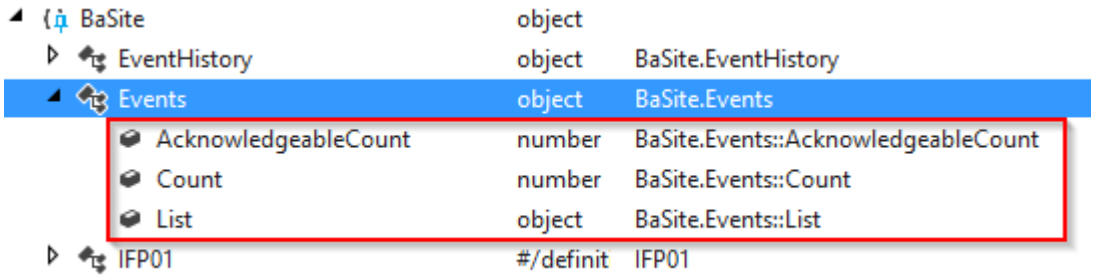
Für jedes Gerät wird ein dynamisches Symbol mit demselben Namen wie das Gerät angelegt. Der Aufbau der ersten Ebene eines Gerätesymbols ist immer gleich.



In der ersten Ebene ist der Top-Knoten des Gerätes zu finden, in der sich dann die Children befinden. Diese Children sind somit die ersten tatsächlichen Elemente in der Projektstruktur.

Events

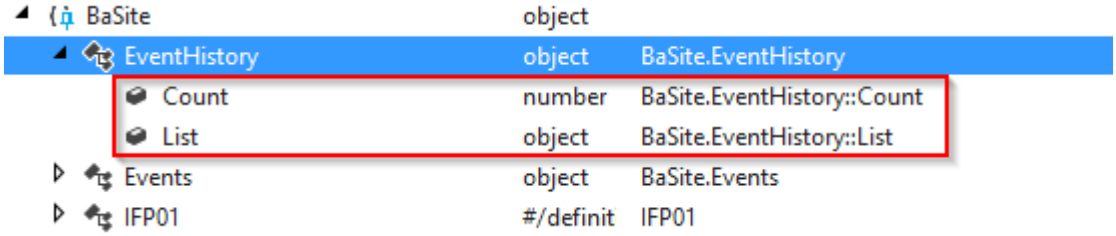
Das Symbol bietet verschiedene Untersymbole an, die sich auf die aktuellen Events von **allen** beziehen.



- AcknowledgeableCount: Aktive Events, die eine Interaktion vom Benutzer erfordern (quittieren)
- Count: Anzahl aller Events
- List: Die Liste der Events
- Dadurch eignet sich das Symbol für eine geräteübergreifende Eventliste [▶ 1185].

EventHistory

Das Symbol bietet verschiedene Untersymbole an, die sich auf die Eventhistorie von **allen** Geräten beziehen.



- Count: Anzahl alle Events in der Historie.
- List: Die Liste der Events in der Historie.
  Dadurch eignet sich das Symbol für eine geräteübergreifende Eventhistorie [▶ 1185].

6.2.1.8 Globale Einstellungen

Die globalen Einstellungen der verschiedenen TcHmiBa NuGet-Pakete können überschrieben werden. Den Einstiegspunkt bildet der Callback vom `TcHmiBaEvents.onOverrideSettings`-Event, der im CodeBehind anzulegen ist.

```
(function (TcHmi) {
  TcHmi.EventProvider.register(TcHmi.BuildingAutomation.TcHmiBaEvents.onOverrideSettings, function (
e, data) {
    e.destroy();
    // Todo
  });
})(TcHmi);
```

Die Wertänderung erfolgt dann im `ToDo` Bereich, wie nachfolgend für die jeweilige Eigenschaft beschrieben.

TcHmiBaControls

EventList

DefaultColumns

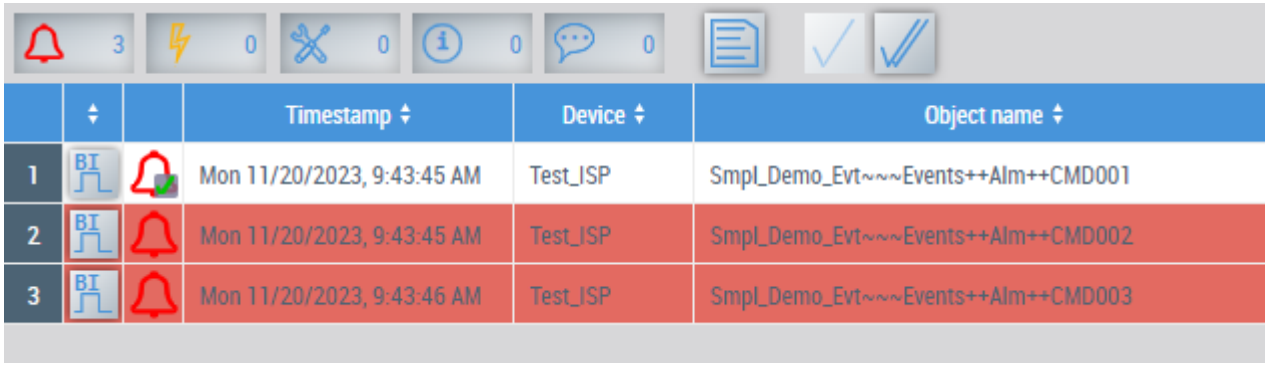
Definiert die Standardeinstellungen der Spalten für alle EventList [▶ 1185]-Controls.

```
TcHmi.BuildingAutomation.Controls.Management.EventList.DefaultColumns = [
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.baIdentifier, width: 30,
widthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.event, width: 30, widthUn
it: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.timestamp, width: 140, wi
dthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.device, width: 70, widthU
```

```
nit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.objectName, width: 300, w
idthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.instancePath, width: 300,
 widthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.description, width: 300,
widthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.eventClass, width: 30, wi
dthUnit: 'px' },
    { type: TcHmi.BuildingAutomation.Controls.Management.EventList.Columns.eventClassInstanceDescrip
tion, width: 200, widthUnit: 'px' }
];
```

MaximumEventTypePulse

Definiert für jeden Benutzerlevel, ob ein Puls angezeigt wird, wenn ein aktives Event mit dieser oder einer höheren Priorität [▶ 105] ansteht.



Setzen eines Wertes.

```
TcHmi.BuildingAutomation.Controls.Management.EventList.MaximumEventTypePulse.set(TcHmi.BuildingAutom
ation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb);
```

Setzen aller Werte.

```
TcHmi.BuildingAutomation.Controls.Management.EventList.MaximumEventTypePulse = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventType.eDisturb]
]);
```

ProjectNavigationTextual

DefaultContentViewDialogWidth

Definiert die Standardgröße des Dialogs, der den Inhalt eines BaObjects anzeigt.

```
TcHmi.BuildingAutomation.Controls.Management.ProjectNavigationTextual.DefaultContentViewDialogWidth
= 1000;
```

RoomControl

DefaultBaObjectListDialogWidth

Definiert die Standardgröße des Dialogs, der die BaObject-Liste anzeigt.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.RoomControl.DefaultBaObjectListDialogWidth = 1000;
```

Light

ShowQuickLinks

Definiert, ob beim Light-Control [▶ 1222] die Schnelleinstellungen angezeigt werden.



```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.ShowQuickLinks = false;
```

PriorityIcons

Definiert die Icons, die bei einer bestimmten Priorität am Light-Control [▶ 1222] angezeigt werden.

Ist für eine Priorität kein Icon festgelegt, erfolgt keine Anzeige.

Setzen eines Wertes.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.PriorityIcons.set(TcHmi.BuildingAutomation.Co
ntrols.RoomAutomation.Light.Priority.fire,
TcHmi.BuildingAutomation.Icons.convertIIconDataToIIconAttributes({ ...
TcHmi.BuildingAutomation.Icons.Building.FireAlarm, color:
TcHmi.BuildingAutomation.Color.RGBAColor.Red }));
```

Setzen aller Werte.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.PriorityIcons = new Map([
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.fire, TcHmi.BuildingAutomation.
Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.FireAlarm, colo
r: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.communicationError, TcHmi.Build
ingAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Events.Fla
g.Fault, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.burglary, TcHmi.BuildingAutomat
ion.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.BurglarAlar
m, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.maintenance, TcHmi.BuildingAuto
mation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.Maintena
nce, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.cleaning, TcHmi.BuildingAutomat
ion.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.Cleaning, c
olor: TcHmi.BuildingAutomation.Color.RGBAColor.Yellow })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.nightWatch, TcHmi.BuildingAutom
ation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.NightWatc
h, color: TcHmi.BuildingAutomation.Color.RGBAColor.Blue })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.manual, TcHmi.BuildingAutomatio
n.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomation.Manual,
 color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Light.Priority.automaticLight, TcHmi.BuildingA
utomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomation
.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })]
]);
```

Sunblind

PriorityIcons

Definiert die Icons, die bei einer bestimmten Priorität am Sunblind-Control [▶ 1230] angezeigt werden.

Ist für eine Priorität kein Icon festgelegt, erfolgt keine Anzeige.

Setzen eines Wertes.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.PriorityIcons.set(TcHmi.BuildingAutomation
.Controls.RoomAutomation.Sunblind.Priority.fire, TcHmi.BuildingAutomation.Icons.convertIIconDataToII
conAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.FireAlarm, color:
TcHmi.BuildingAutomation.Color.RGBAColor.Red }));
```

Setzen aller Werte.

```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.PriorityIcons = new Map([
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.fire, TcHmi.BuildingAutomati
on.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.FireAlarm, c
olor: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.storm, TcHmi.BuildingAutomat
ion.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.StormAlarm,
 color: TcHmi.BuildingAutomation.Color.RGBAColor.Blue })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.ice, TcHmi.BuildingAutomatio
n.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.IceAlarm, col
or: TcHmi.BuildingAutomation.Color.RGBAColor.Blue })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.commError, TcHmi.BuildingAut
omation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Events.Flag.Faul
t, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.burglary, TcHmi.BuildingAuto
mation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.BurglarA
```

```
larm, color: TcHmi.BuildingAutomation.Color.RGBAColor.Red })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.maintenance, TcHmi.BuildingA
utomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Building.Maint
enance, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.manualActuator, TcHmi.Buildi
ngAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomat
ion.Manual, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.manualGroup, TcHmi.BuildingA
utomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomation
.Manual, color: TcHmi.BuildingAutomation.Color.RGBAColor.DarkOrange })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.facadeThermoAutomatic, TcHmi
.BuildingAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Room
Automation.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.facadeTwilightAutomatic, TcH
mi.BuildingAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.Ro
omAutomation.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })],
    [TcHmi.BuildingAutomation.Controls.RoomAutomation.Sunblind.Priority.sunProtection, TcHmi.Buildin
gAutomation.Icons.convertIIconDataToIIconAttributes({ ...TcHmi.BuildingAutomation.Icons.RoomAutomati
on.Automatic, color: TcHmi.BuildingAutomation.Color.RGBAColor.TcHmiGreen })]
]);
```

Window

ShowQuickLinks

Definiert, ob beim Window [▶ 1235]-Control die Schnelleinstellungen angezeigt werden.



```
TcHmi.BuildingAutomation.Controls.RoomAutomation.Window.ShowQuickLinks = false;
```

TcHmiBaFramework

BusyHandler

RecordTimerResults

Definiert, ob die Timer-Ergebnisse für verschiedene Vorgänge aufgezeichnet werden.

```
TcHmi.BuildingAutomation.BusyHandler.RecordTimerResults = false;
```

Button

DoublePressDuration

Definiert die Zeit, die zwischen zwei Klicks verstreichen darf, bis die beiden Klicks als Doppelklick erkannt werden.

```
TcHmi.BuildingAutomation.Components.Button.DoublePressDuration = 200;
```

Calendar

DefaultEventDialogSize

Definiert die Standardgröße des Event Dialogs.

```
TcHmi.BuildingAutomation.Components.Calendar.DefaultEventDialogSize = {
    width: 700,
    height: 350
};
```

CalendarList

DefaultCalendarListDialogSize

Definiert die Standardgröße des CalendarList Dialogs.

```
TcHmi.BuildingAutomation.Components.CalendarList.DefaultCalendarListDialogSize = {
    width: 750,
    height: 400
};
```

ContentWindow

HideHeaderIconBorderDefault

Definiert, ob die Umrandung vom HeaderIcon standardmäßig ausgeblendet wird. Die Sichtbarkeit lässt sich weiterhin für jede Instanz separat festlegen.

```
TcHmi.BuildingAutomation.Components.ContentWindow.HideHeaderIconBorderDefault = false;
```

DateTimePicker

DefaultDateTimePickerDialogSize

Definiert die Standardgröße des DateTimePicker Dialogs.

```
TcHmi.BuildingAutomation.Components.DateTimePicker.DefaultDateTimePickerDialogSize = {
    width: 650,
    height: 450
};
```

DefaultDatePickerDialogSize

Definiert die Standardgröße des DatePicker Dialogs.

```
TcHmi.BuildingAutomation.Components.DateTimePicker.DefaultDatePickerDialogSize = {
    width: 300,
    height: 300
};
```

DialogWindow

MoveLimitation

Begrenzung für Positionierung außerhalb des Browser Fensters.

```
TcHmi.BuildingAutomation.Components.DialogWindow.MoveLimitation = 100;
```

TimePicker

DefaultTimePickerDialogSize

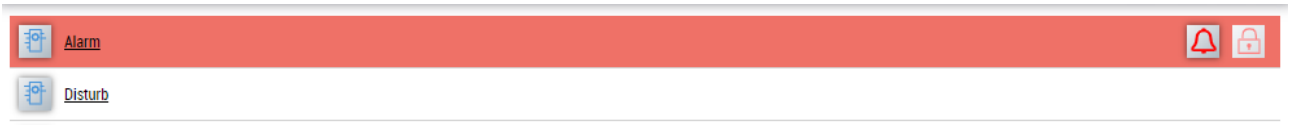
Definiert die Standardgröße des TimePicker Dialogs.

```
TcHmi.BuildingAutomation.Components.TimePicker.DefaultTimePickerDialogSize = {
    width: 350,
    height: 200
};
```

ProjectNavigationList

MaximumEventTypePulse

Definiert für jeden Benutzerlevel, ob ein Puls angezeigt wird, wenn ein aktives Event mit dieser oder einer höheren Priorität [▶ 105] ansteht.



Setzen eines Wertes.

```
TcHmi.BuildingAutomation.Navigation.ProjectNavigationList.MaximumEventTypePulse.set(TcHmi.BuildingAu
tomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb);
```

Setzen aller Werte.

```
TcHmi.BuildingAutomation.Navigation.ProjectNavigationList.MaximumEventTypePulse = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventType.eDisturb]
]);
```

MaximumEventConditionDisplayed

Definiert für jeden Benutzerlevel die maximale EventCondition [▶ 104], die in den Zeilen der ProjectNavigation [▶ 1186] angezeigt wird.

Setzen eines Wertes.

```
TcHmi.BuildingAutomation.Navigation.ProjectNavigationList.MaximumEventConditionDisplayed.set(TcHmi.B
uildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther);
```

Setzen aller Werte.

```
Navigation.ProjectNavigationList.MaximumEventConditionDisplayed = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventCondition.eEventIcon
Displayed],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed]
]);
```

UiIcon

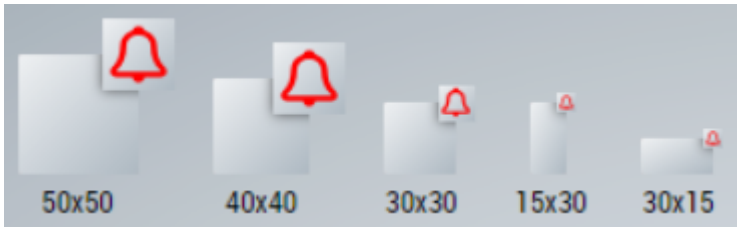
AutoActivateIconStatus

Definiert, ob das IconStatus [▶ 1181]-Attribut automatisch gesetzt wird, wenn Events konfiguriert wurden. Mit dieser Einstellung wird das Icon in der definierten Eventfarbe eingefärbt, wenn ein Event aktiv ist.

```
TcHmi.BuildingAutomation.Components.UiIcon.AutoActivateIconStatus = false;
```

AutoScaleEventIconThreshold

Definiert den Schwellwert, den die Höhe oder Breite unterschreiten muss, bevor die automatische Skalierung des Event Symbols aktiv wird.




```
TcHmi.BuildingAutomation.Components.UiIcon.AutoScaleEventIconThreshold = 40;
```

EnableEventCountBadge

Definiert, ob die Anzahl der Events vom selben Eventtyp angezeigt werden.




```
TcHmi.BuildingAutomation.Components.UiIcon.EnableEventCountBadge = true;
```

EventIconSize

Definiert die initiale Höhe und Breite vom EventIcon.

```
TcHmi.BuildingAutomation.Components.UiIcon.EventIconSize = 30;
```

MaximumEventTypePulse

Definiert für jeden Benutzerlevel, ob ein Puls angezeigt wird, wenn ein aktives Event mit dieser oder einer höheren Priorität [▶ 105] ansteht.



Setzen eines Wertes.

```
TcHmi.BuildingAutomation.Components.UiIcon.MaximumEventTypePulse.set(TcHmi.BuildingAutomation.BA.Rol
e.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb);
```

Setzen aller Werte.

```
Components.UiIcon.MaximumEventTypePulse = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventType.eDisturb],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventType.eDisturb]
]);
```

MaximumEventConditionDisplayed

Definiert für jeden Benutzerlevel die maximale EventCondition [▶ 104], die am UiIcon [▶ 1178] angezeigt wird.

Setzen eines Wertes.

```
TcHmi.BuildingAutomation.Components.UiIcon.MaximumEventConditionDisplayed.set(TcHmi.BuildingAutomati
on.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther);
```

Setzen aller Werte.

```
TcHmi.BuildingAutomation.Components.UiIcon.MaximumEventConditionDisplayed = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventCondition.eTypeOther]
,
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventCondition.eEventIcon
Displayed],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed]
]);
```

Charting

Axis

DefaultAxisOptionsDialogSize

Definiert die Standardgröße des Dialogs für die Achsenoptionen.

```
TcHmi.BuildingAutomation.Charting.Axis.DefaultAxisOptionsDialogSize = {
    height: 180,
    width: 350
};
```

Trend

DefaultDisplayedObjects

Definiert das Standardobjekt in der Trendanzeige.

```
TcHmi.BuildingAutomation.Charting.Trend.DefaultDisplayedObjects =
TcHmi.BuildingAutomation.Trend.DisplayedObjects.trendableObjects;
```

DefaultDisplayedDescription

Definiert die Standardbeschreibung in der Trendanzeige.

```
TcHmi.BuildingAutomation.Charting.Trend.DefaultDisplayedDescription =
TcHmi.BuildingAutomation.BA.BaParameterId.eInstDescription;
```

CollectionConfigurator

DefaultConfiguratorDialogWidth

Definiert die Standardbreite des Dialogs vom Trendsammlungskonfigurator.

```
TcHmi.BuildingAutomation.Charting.Trend.CollectionConfigurator.DefaultConfiguratorDialogWidth = 1000
;
```

DefaultTrendCollectionSelectionDialogWidth

Definiert die Standardbreite des Dialogs zur Auswahl der anzuzeigenden Trendsammlungen.

```
TcHmi.BuildingAutomation.Charting.Trend.CollectionConfigurator.DefaultTrendCollectionSelectionDialog
Width = 400;
```

Storage

UserData

LastContentSorageLocation

Definiert den Ort, an dem der zuletzt geöffnete Content eines Benutzers gespeichert wird.

```
TcHmi.BuildingAutomation.Storage.UserData.LastContentStorageLocation = TcHmi.BuildingAutomation.Stor
age.Location.baSite;
```

LastThemeSorageLocation

Definiert den Ort, an dem der zuletzt verwendete Theme eines Benutzers gespeichert wird.

```
TcHmi.BuildingAutomation.Storage.UserData.LastThemeStorageLocation = TcHmi.BuildingAutomation.Storag
e.Location.baSite;
```

TrendSettingsStorageLocation

Definiert den Ort, an dem die Trendeinstellungen eines Benutzers gespeichert werden.

```
TcHmi.BuildingAutomation.Storage.UserData.TrendSettingsStorageLocation = TcHmi.BuildingAutomation.St
orage.Location.baSite;
```

TrendCollectionStorageLocation

Definiert den Ort, an dem die Trendsammlungen eines Benutzers gespeichert werden.

```
TcHmi.BuildingAutomation.Storage.UserData.TrendCollectionStorageLocation = TcHmi.BuildingAutomation.
Storage.Location.baSite;
```

TrendCollectionSelectionStorageLocation

Definiert den Ort, an dem die angezeigten Trendsammlungen eines Benutzers gespeichert werden.

```
TcHmi.BuildingAutomation.Storage.UserData.TrendCollectionSelectionStorageLocation = TcHmi.BuildingAu
tomation.Storage.Location.baSite;
```

Server

BaSite-Extension

DefaultDiagnosticsDialogSize

Definiert die Standardgröße des Diagnosedialogs.

```
TcHmi.BuildingAutomation.Server.BaSite.DefaultDiagnosticsDialogSize = {
    height: 500,
    width: 500
};
```

EventHelper

MaximumEventCondition

Definiert für jeden Benutzerlevel die maximale EventCondition [▶ 104], die in der HMI angezeigt wird.

In der Standardeinstellung werden alle Events [▶ 1256], Flags [▶ 1257], Prioritäten [▶ 1259] und Locks [▶ 1259] angezeigt.

Setzen eines Wertes.

```
TcHmi.BuildingAutomation.BA.EventHelper.MaximumEventCondition.set(TcHmi.BuildingAutomation.BA.Role.e
Guest, TcHmi.BuildingAutomation.BA.EventCondition.eEventIconDisplayed);
```

Setzen aller Werte.

```
TcHmi.BuildingAutomation.BA.EventHelper.MaximumEventCondition = new Map([
    [TcHmi.BuildingAutomation.BA.Role.eGuest, TcHmi.BuildingAutomation.BA.EventCondition.eEventIconD
isplayed],
    [TcHmi.BuildingAutomation.BA.Role.eBasic, TcHmi.BuildingAutomation.BA.EventCondition.eEventIconD
isplayed],
    [TcHmi.BuildingAutomation.BA.Role.eAdvanced, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed],
    [TcHmi.BuildingAutomation.BA.Role.eExpert, TcHmi.BuildingAutomation.BA.EventCondition.eEventIcon
Displayed],
    [TcHmi.BuildingAutomation.BA.Role.eInternal, TcHmi.BuildingAutomation.BA.EventCondition.eEventIc
onDisplayed]
]);
```

BA

HiddenBaParameterIds

Definiert die BaParameterIds, die nicht im generischen Parameterdialog angezeigt werden.

```
TcHmi.BuildingAutomation.BA.BaDevice.HiddenBaParameterIds = [];
```

BaParameterCategories

Ordnet die verschiedenen BaParameterIds in Kategorien ein. Die Reihenfolge beim Hinzufügen wird bei der Anzeige im Parameterdialog berücksichtigt.

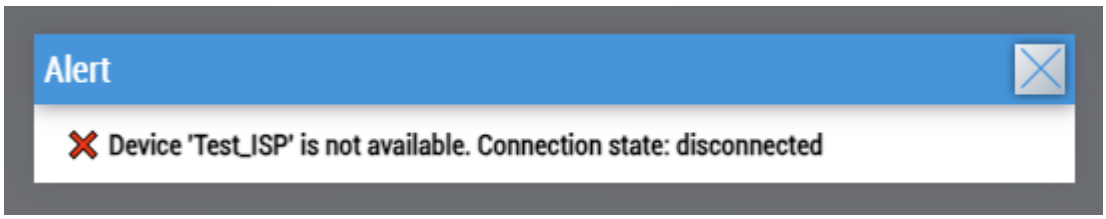
Setzen eines Wertes.

```
TcHmi.BuildingAutomation.BA.BaParameterCategories.set(TcHmi.BuildingAutomation.BA.BaParameterCategor
y.configuration, [
    TcHmi.BuildingAutomation.BA.BaParameterId.eConfigurate,
    TcHmi.BuildingAutomation.BA.BaParameterId.eToggleMode,
    TcHmi.BuildingAutomation.BA.BaParameterId.eStepDelay,
    TcHmi.BuildingAutomation.BA.BaParameterId.eMinOffTime,
    TcHmi.BuildingAutomation.BA.BaParameterId.eMinOnTime
]);
```

BaDevice

DialogConnectionAutoCloseStateChanged

Definiert, ob die Benachrichtigung bei Verbindungsverlust eines Gerätes nach Wiederherstellung der Verbindung automatisch geschlossen wird.




```
TcHmi.BuildingAutomation.BA.BaDevice.DialogConnectionAutoCloseStateChanged = true;
```

DialogConnectionAutoReloadOnReconnect

Definiert, ob das HMI nach Wiederherstellung der Verbindung zu einem Gerät automatisch neugeladen wird.

```
TcHmi.BuildingAutomation.BA.BaDevice.DialogConnectionAutoReloadOnReconnect = true;
```

DialogConnectionAutoReloadOnReconnectTime

Definiert die Wartezeit, bis das HMI nach Wiederherstellung der Verbindung zu einem Gerät automatisch neugeladen wird.

```
TcHmi.BuildingAutomation.BA.BaDevice.DialogConnectionAutoReloadOnReconnectTime = 30;
```

BaBasicObject

DisableParameterDialogHeaderIcon

Definiert, ob das Object/Node-Icon in der Kopfzeile des Parameterdialogs nicht angezeigt wird.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DisableParameterDialogHeaderIcon = false;
```

DefaultParameterDialogWidth

Definiert die Standardbreite des Parameterfensters.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DefaultParameterDialogWidth = 800;
```

DefaultOnlineTrendDialogSize

Definiert die Standardgröße des Fensters vom Online-Trend.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DefaultOnlineTrendDialogSize = {
    height: 250,
    width: 450
}
```

DefaultNavigationDialogWidth

Definiert die Standardbreite des Dialogs, der die ProjectNavgiationList eines BaObjects anzeigt.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.DefaultNavigationDialogWidth = 1000;
```

AutoCollapseNavigationDialogEntries

Definiert, ob immer nur ein Eintrag im Dialog, der die ProjectNavigationList anzeigt, geöffnet sein darf oder mehrere.

```
TcHmi.BuildingAutomation.BA.BaBasicObject.AutoCollapseNavigationDialogEntries = true;
```

BaView

DisableNodeTypeIcons

DisableNodeTypeIcons

Definiert, ob die NodeTypeIcons [▶ 1260] in der HMI verwendet werden.

```
TcHmi.BuildingAutomation.BA.BaView.DisableNodeTypeIcons = false;
```

6.2.1.9 Projektvorlagen

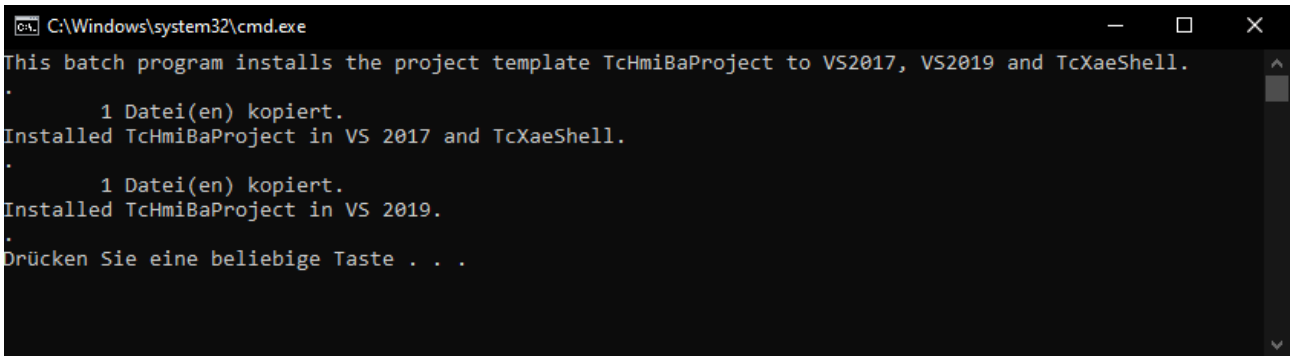
Projektvorlagen sollen den Einstieg in ein *TcHmiBa*-Projekt erleichtern, indem sie notwendige Abhängigkeiten bereits installieren sowie zusätzliche Elemente (z.B. Navigation) bereitstellen.

Installation

Die Installation der Projektvorlagen in die verfügbaren Entwicklungsumgebungen wird **nicht** vom TF8040 sondern der Batch-Datei *InstallProjectTemplates.bat* durchgeführt.

Nach der Installation der TwinCAT 3 HMI und TF8040, befindet sich die Batch-Datei im Verzeichnis: *C:\TwinCAT\Functions\TF8040 Building Automation\HMI\ProjectTemplates*

Ein Doppelklick auf die *InstallProjectTemplates.bat* führt diese aus und es öffnet sich ein Konsolenfenster.



Die Projektvorlagen haben aktuell einige Einschränkungen. Im Folgenden werden diese näher beschrieben und wie sie behoben werden können.

Benutzerverwaltung

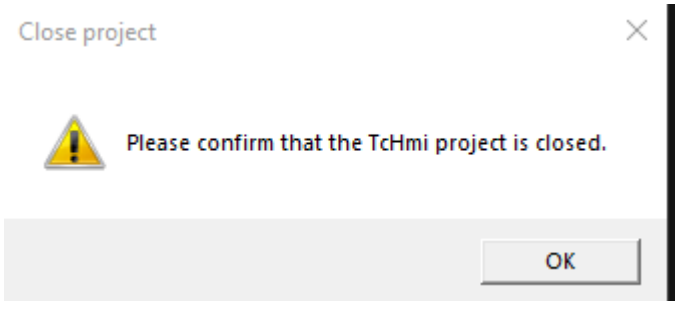
Für die Nutzung des integrierten Benutzermanagements von TF8040 ist es notwendig, dass bestimmte Benutzergruppen und dazu passende Benutzer im TcHmi-Projekt vorhanden sind. Die Projektvorlage liefert diese Benutzer **nicht** mit.
Mit dem Programm *CreateDefaultUserManagement.exe* können die benötigten Einstellungen für ein Projekt erstellt werden.

Hier finden Sie nähere Informationen zu den verschiedenen Benutzergruppen [▶ 15].

Nach der Installation der TwinCAT 3 HMI und TF8040, befindet sich das Programm im Verzeichnis: *C:\TwinCAT\Functions\TF8040 Building Automation\HMI\Tools\CreateDefaultUserManagement*

✓ Ein Doppelklick auf die *CreateDefaultUserManagement.exe* führt diese aus.

1. Wählen Sie zunächst das HMI-Projekt aus.
2. Projekt schließen (wenn nicht bereits geschlossen).



3. Ausgabe überprüfen

```
Please select TcHmi project file:
Selected project file: 'C:\temp\TcHmiBaProject1\TcHmiBaProject1.hmiproj'

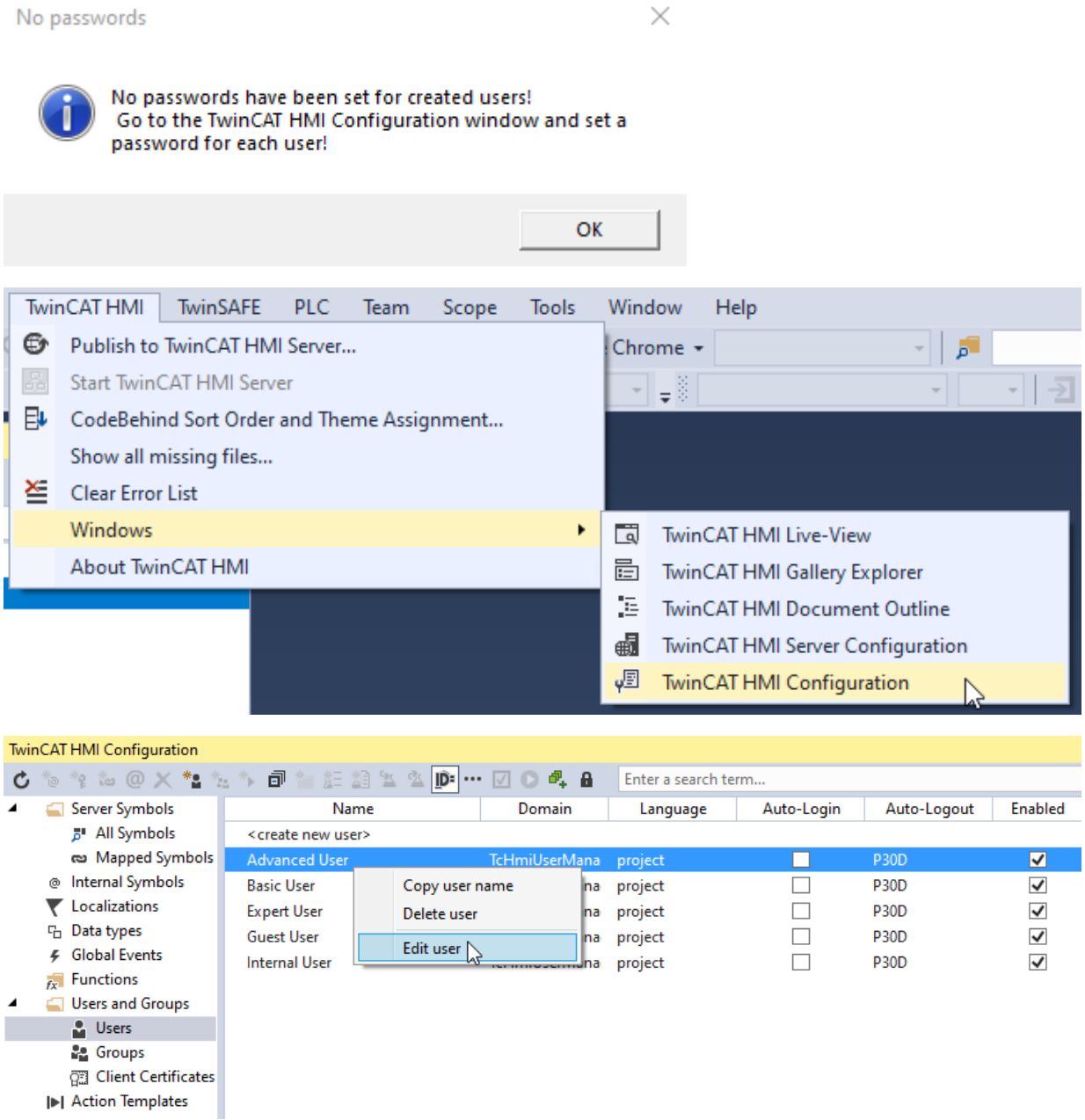
Start editing TcHmiUserManagement.Config.default.json
'Advanced User' added to Users.
'Default User' added to Users.
'Expert User' added to Users.
'Internal User' added to Users.
Finished editing TcHmiUserManagement.Config.default.json

Writing to TcHmiUserManagement.Config.default.json ...
Finished writing to TcHmiUserManagement.Config.default.json.


Start editing TcHmiSrv.Config.default.json
'Advanced User' added to UserGroupUsers.
'Default User' added to UserGroupUsers.
'Expert User' added to UserGroupUsers.
'Internal User' added to UserGroupUsers.
'Advanced' added to UserGroups.
'Default' added to UserGroups.
'Expert' added to UserGroups.
'Internal' added to UserGroups.
Finished editing TcHmiSrv.Config.default.json

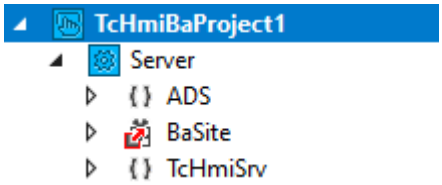
Writing to TcHmiSrv.Config.default.json ...
Finished writing to TcHmiSrv.Config.default.json.
```

4. Projekt öffnen und Passwörter für die angelegten Benutzer erstellen.

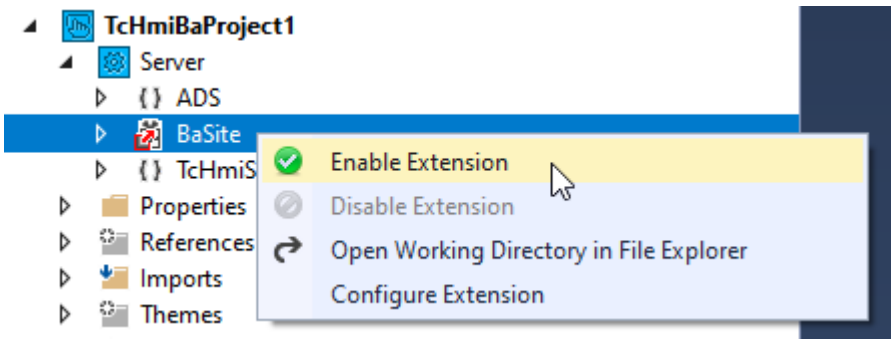


BaSite-Extension

Die Servererweiterung wird inaktiv starten, da die Konfigurationsdateien nicht mit der Projektvorlage geladen werden.

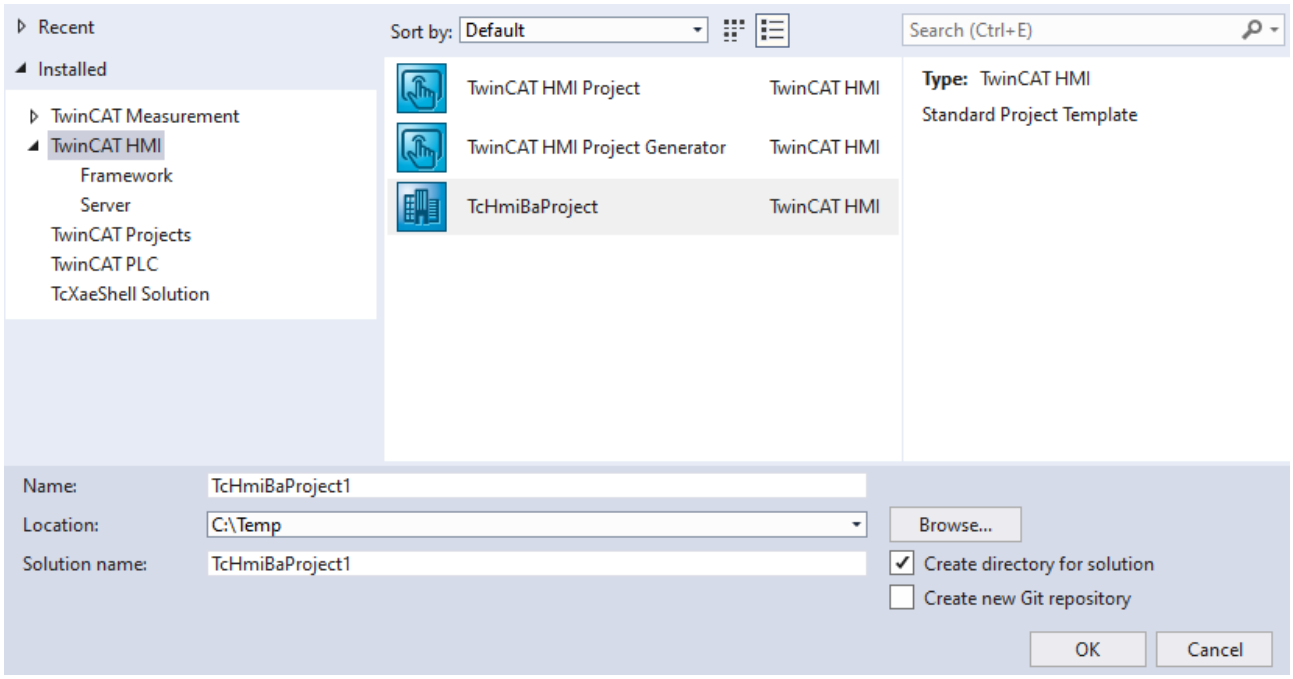


Daher ist ein manueller Start erforderlich.



6.2.1.9.1 TcHmiBaProject

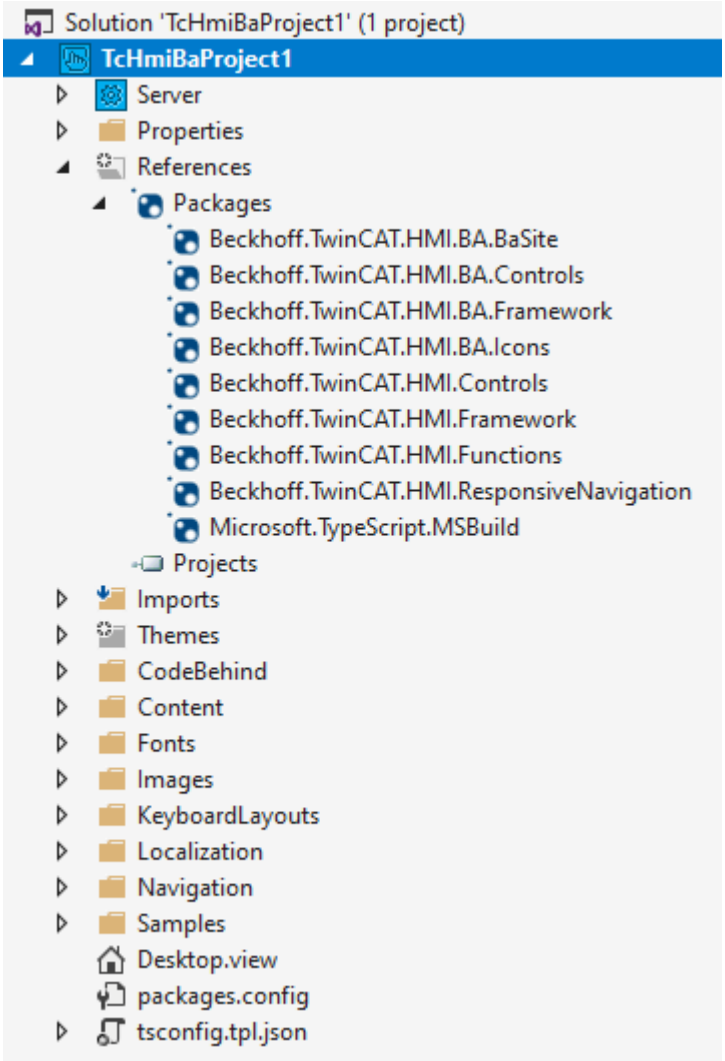
Die Projektvorlage dient der einfachen und schnellen Erstellung eines TcHmi-Projektes.



Beachten Sie die Hinweise [▶ 1297] für Projektvorlagen.

Inhalt

Vorschau



Referenzen

Die Projektvorlage enthält die folgenden NuGet-Pakete:

- Beckhoff.TwinCAT.HMI.BA.Icons
- Beckhoff.TwinCAT.HMI.BA.Framework
- Beckhoff.TwinCAT.HMI.BA.Controls
- Beckhoff.TwinCAT.HMI.BA.ServerExtension
- Beckhoff.TwinCAT.HMI.Framework
- Beckhoff.TwinCAT.HMI.Controls
- Beckhoff.TwinCAT.HMI.Functions
- Beckhoff.TwinCAT.HMI.ResponsiveNavigation
- Microsoft.TypeScript.MSBuild

CodeBehind

Auflistung aller globalen Einstellungen für TcHmiBa.

- TcHmiBaSettings.js

Content

Pages

Um den Einstieg zu erleichtern, enthält die Projektvorlage bereits einige Content-Seiten, wie z.B. eine Navigation in der Kopfzeile.

Für die korrekte Funktion der Kopfzeile sind die folgenden Seiten erforderlich:

- EventList.content
- ServerLog.content
- StartPage.content

Die Navigation in der Kopfzeile verweist auf die Seiten:

- PlantView.content
- ProjectNavigation.content

Diese Seiten sind anpassbar und es lassen sich weitere hinzufügen.



Werden Seiten hinzugefügt, umbenannt oder entfernt, dann muss das Attribut MenuData [▶ 1304] der Kopfzeile aktualisiert werden (siehe Navigation).

Navigation

Konfigurierbare Menüleiste.

- HeaderMenu.usercontrol

Samples

Der Ordner "Samples" enthält Beispiele für UserControls zum Erstellen benutzerspezifischer Anlagen.

6.2.1.9.1.1 Kopfzeile

Die Kopfzeile ist ein UserControl und dient als Einstiegspunkt für Anwender. Sie bietet eine einfache Möglichkeit eine Navigation für das HMI aufzubauen und noch verschiedene anderen Features.

Features

Die Features im Überblick (v.l.n.r).

- Logo (1)
- Responsive Navigation (2)
- Benutzereinstellungen und weitere Informationen (3)
- Eventliste (4)
- Gebäudeinformationen (5)
- Außentemperatur (6)
- Datum und Uhrzeit (7)

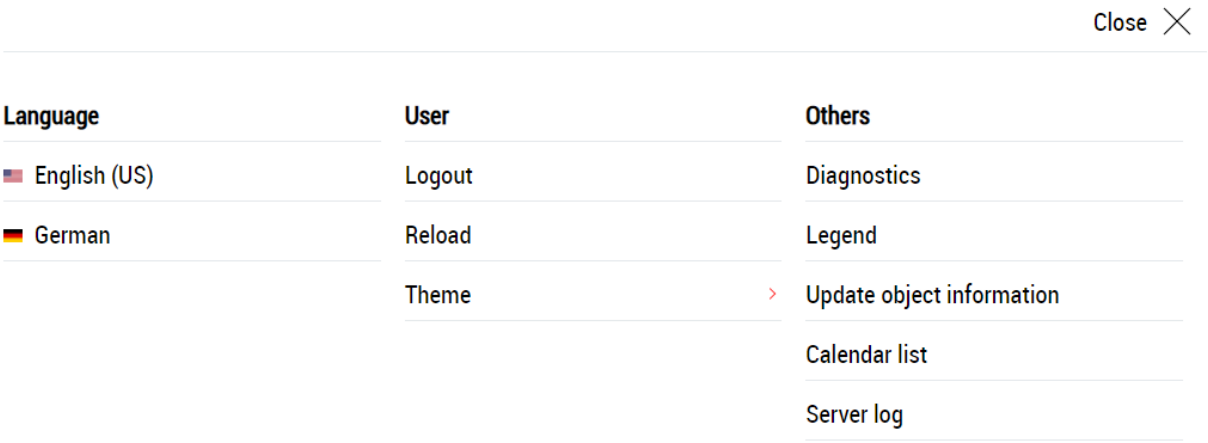


Benutzereinstellungen und weitere Informationen

In diesem Menü kann der Benutzer verschiedene Einstellungen vornehmen und Dialoge einblenden:

- Sprache einstellen
- Theme einstellen
- Diagnosedaten der BaSite-Extension anzeigen

- Dialog mit Icon Legende anzeigen
- Aktualisieren der generellen Objekt-Informationen, die nicht zyklisch aktualisiert werden (z.B. Beschreibung, Aktiv Text, Status Texte)
- Liste zum Anzeigen und Bearbeiten der Kalenderobjekte im Projekt
- Server-Log anzeigen



Eventliste

Die Eventliste [▶ 1185] kann über den Button mit dem Glockensymbol aufgerufen werden. Der Button zeigt ebenfalls die Anzahl der aktiven Events an, wenn die Anzahl über das Attribut *EventCount* verknüpft wurde.

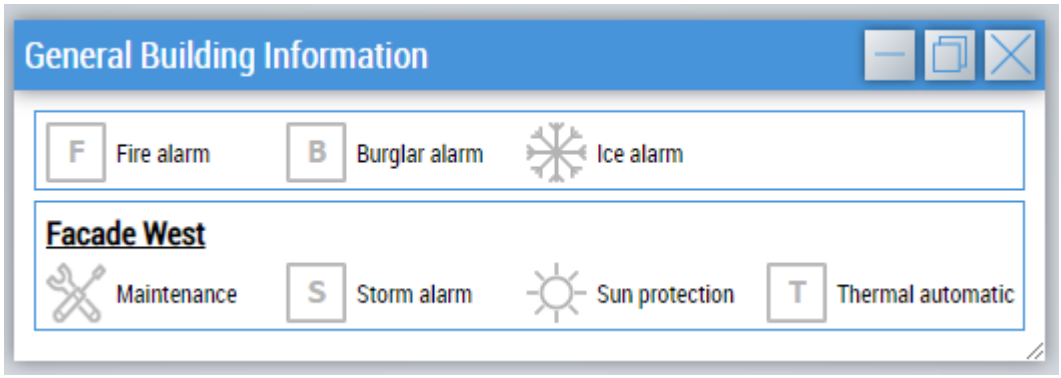


Gebäudeinformationen

Der Button mit dem Infosymbol ist das Control BuildingInformation [▶ 1142].



Über ihn kann ein Fenster aufgerufen werden, in dem verschiedene Informationen über das Gebäude und die Fassaden zu finden sind.



Attribute

Das Control erbt vom TcHmiControl und hat somit dieselben Attribute. Dazu kommen folgende Attribute.

Logo

```
tchmi:framework#/definitions/ContentPath
```

Pfadangabe zum Bild mit dem Logo, das am Anfang der Kopfzeile angezeigt wird.

MenuData

`tchmi:framework#/definitions/TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationBar.MenuItemList`

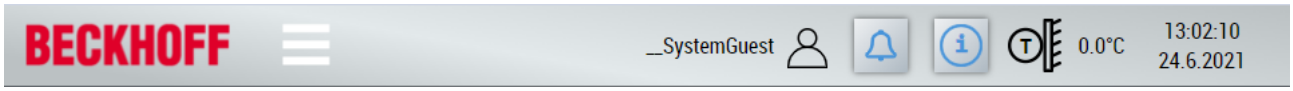
Legt die Struktur und Hierarchie der Navigation fest. Einträge in der Kopfzeile können direkt mit Content-Seiten verknüpft sein oder es können Untermenüs aufgebaut werden, die sich beim Auswählen aufklappen.



SwitchBreakpoint

`tchmi:general#/definitions/Number`

Legt fest, bei welcher Pixelbreite in die Burger-Navigation gewechselt wird.



EventCount

`tchmi:general#/definitions/Number`

Hier kann ein Symbol verknüpft werden, das die Anzahl der aktiven Events in der Eventliste beinhaltet. Diese Anzahl wird dann in einem Badge am Button für die Eventliste angezeigt.



CurrentTemperature

`tchmi:general#/definitions/Number`

Aktuelle Temperatur, die in der Kopfzeile angezeigt werden soll. Typischerweise steht hier die Außentemperatur.

CurrentTemperatureUnit

`tchmi:general#/definitions/String`

Einheit der aktuellen Temperatur.

CloseMenu

`tchmi:general#/definitions/Boolean`

Hat das Attribut den Wert TRUE, lassen sich die Menüs der Kopfzeile schließen (während der Laufzeit änderbar).

NavContent

`tchmi:framework#/definitions/TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent`

Inhalt zum Anzeigen in der responsiven Navigation.

UserContent

```
tchmi:framework#/definitions/TcHmi.Controls.ResponsiveNavigation.TcHmiNavigationContent
```

Inhalt zum Anzeigen im Benutzermenü.

TargetRegion

```
tchmi:framework#/definitions/TcHmiRegion
```

Hier muss die *TcHmiRegion* verknüpft werden, die zur Anzeige der aus dem Menü angewählten Content-Seiten dient.

EventContent

```
tchmi:framework#/definitions/ContentPath
```

Content auf dem die Eventliste liegt.

StartPage

```
tchmi:framework#/definitions/ContentPath
```

Startseite des HMIs. Diese Seite wird beim Klick auf das Logo in die TargetRegion geladen.

7 Tools

Bei neuen Projekten empfehlen wir dringend die Verwendung der Version 5 von TF8040!

7.1 Building Automation Site Explorer

Der Building Automation Site Explorer bildet alle Objekte übersichtlich in der Projektstruktur ab.

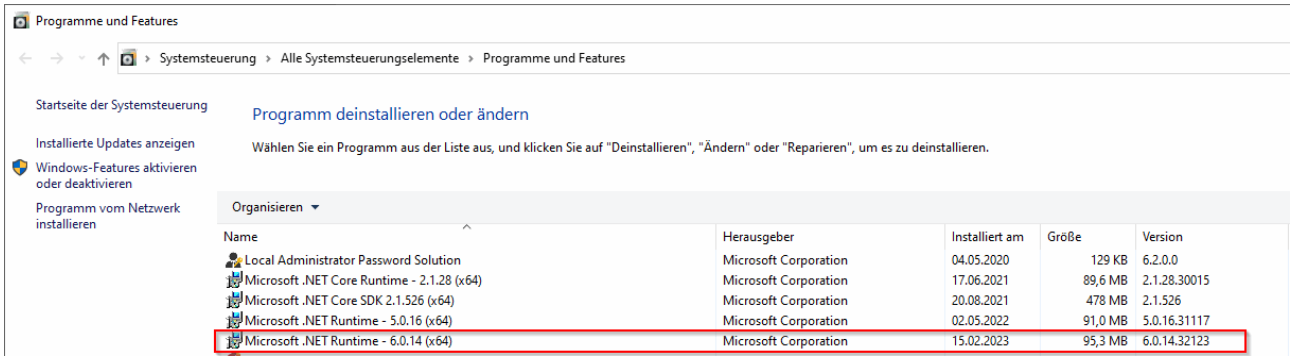
Er unterstützt bei der Inbetriebnahme und der Einregulierung von Anlagen.

Systemvoraussetzungen

Microsoft:

- Windows 10 oder höher
- .NET Desktop Runtime > v6.0

In der Systemsteuerung unter **Programme und Features** kann die aktuelle Version der .NET Runtime überprüft werden.



Beckhoff:

Eine der folgenden Komponenten muss zur Verwendung des Building Automation Site Explorers installiert sein:

- TC1000 | TC3.1 ADS
- TE1000 | TC3 Engineering

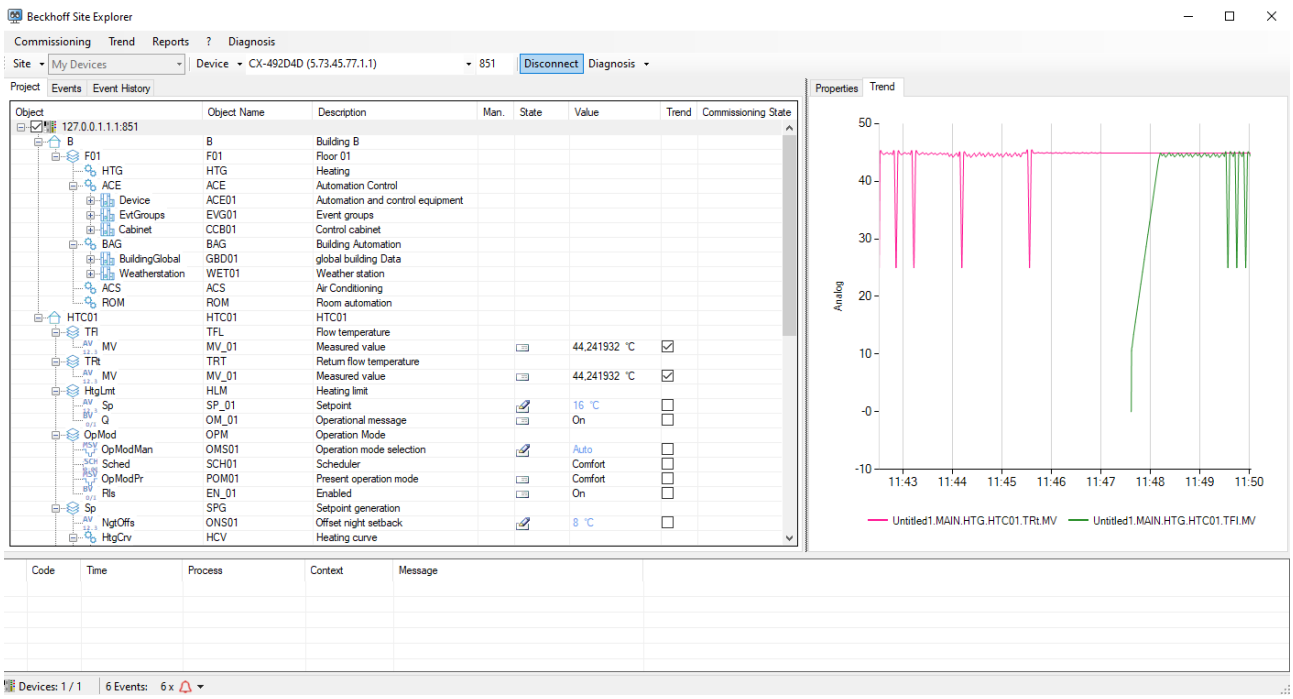
Anwendung

Zugriffsrechte

Der Nutzer wird bei der ersten Anwendung dazu aufgefordert eine Rolle [▶ 15] anzuwenden. Auf diese Weise sind die Zugriffsrechte fest definiert.

Dialogfelder

Projekt



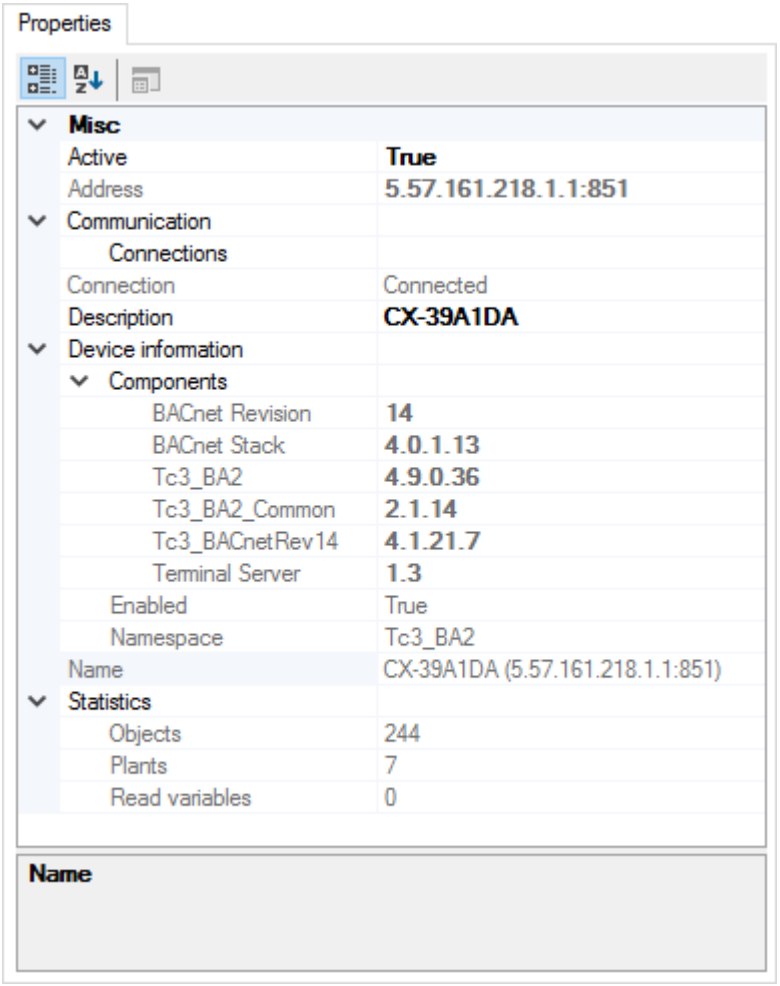
Spalten

- Objekt [▶ 33]
- Objektname
- Beschreibung
- Zustand
  Die folgenden Zustände werden angezeigt:
  - Value source
    deutet einen Soll- bzw. Anzeigewert an.
  - Aktives Event [▶ 33]
  - Übersteuert
  - Außer Betrieb
  - Aktive Priorität
- Manuelles Übersteuern
- Aktueller Wert
- Trend
- Inbetriebnahme-Zustand

Eigenschaften

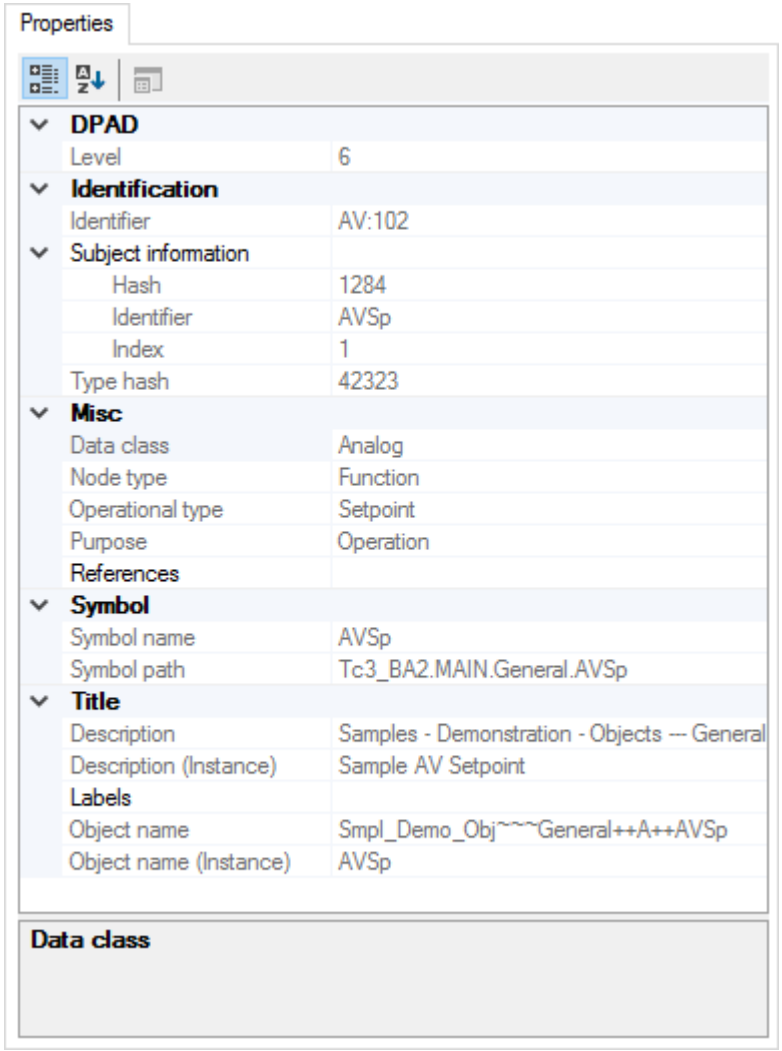
Zeigt die Eigenschaften des markierten Eintrags an.

- **Steuerung**
  Unter den Eigenschaften einer Steuerung sind unter anderem Komponenten (wie Dienste oder Supplements) aufgelistet, die zur Laufzeit ausgeführt werden.



Beispiel: Eigenschaften einer Steuerung

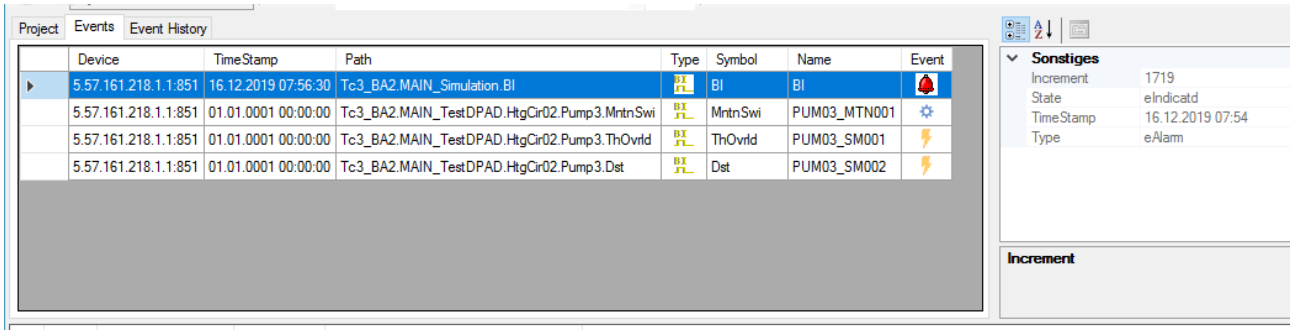
- **Objekte**



Beispiel: Eigenschaften eines analogen Objektes.

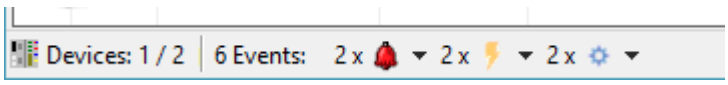
Events

Aktive Events [▶ 33] werden in der Event-Übersicht aufgelistet:

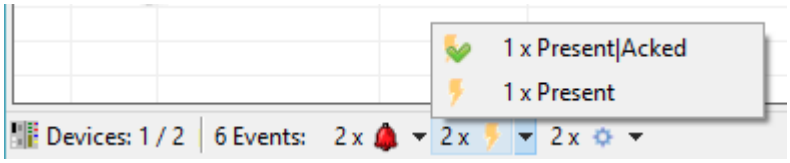


Zusätzlich wird eine Zusammenfassung aller aktiven Events in der Statusleiste am unteren Rand dargestellt.

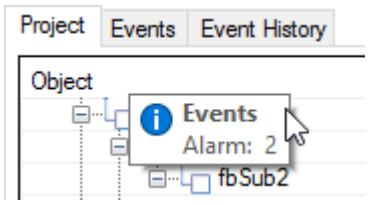
Pro Event-Typ wird jeweils das höchst priorisierte Event-Symbol angedeutet:



Über das Drop-Down Menü können einzelne Zustände der zusammengefassten Events [▶ 33] eingesehen werden:

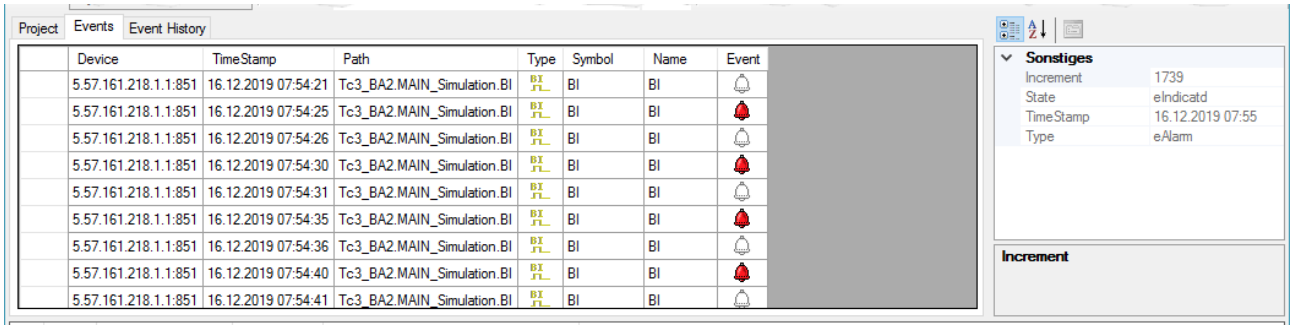


In der Projekt-Ansicht kann ebenfalls eine Zusammenfassung aller aktiven Events [▶ 33] eines Views angezeigt werden, indem die Maus über den Eintrag bewegt wird:



Event-Verlauf

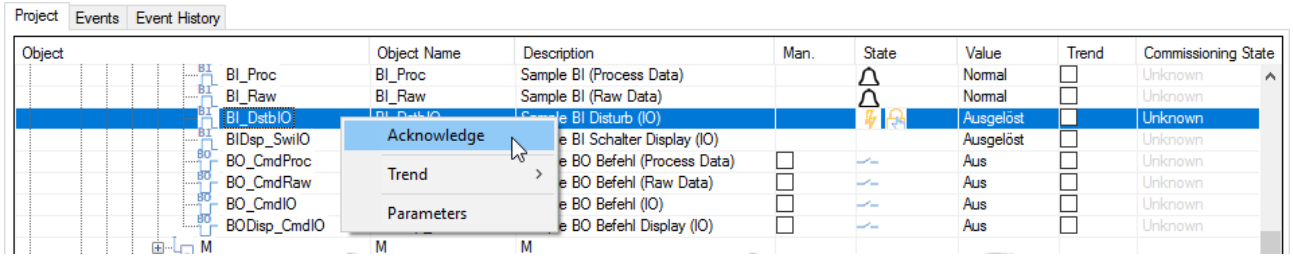
Alle aufgetretenen Events [▶ 33] eines Views werden im Event-Verlauf aufgelistet:



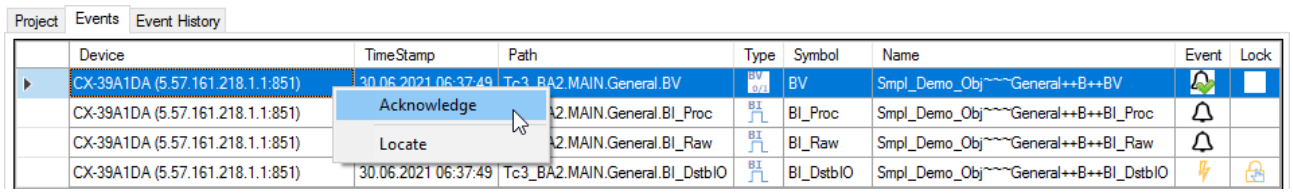
Funktionen

Quittieren

Events [▶ 33] können aus verschiedenen Ansichten über das Kontextmenü quittiert werden:



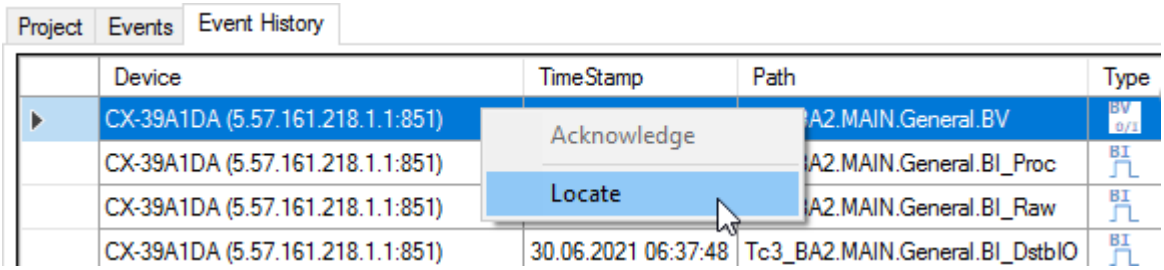
Beispiel: Störung in der Projektansicht quittieren.



Beispiel: Alarm in der Event-Übersicht quittieren.

Zu einem Objekt navigieren

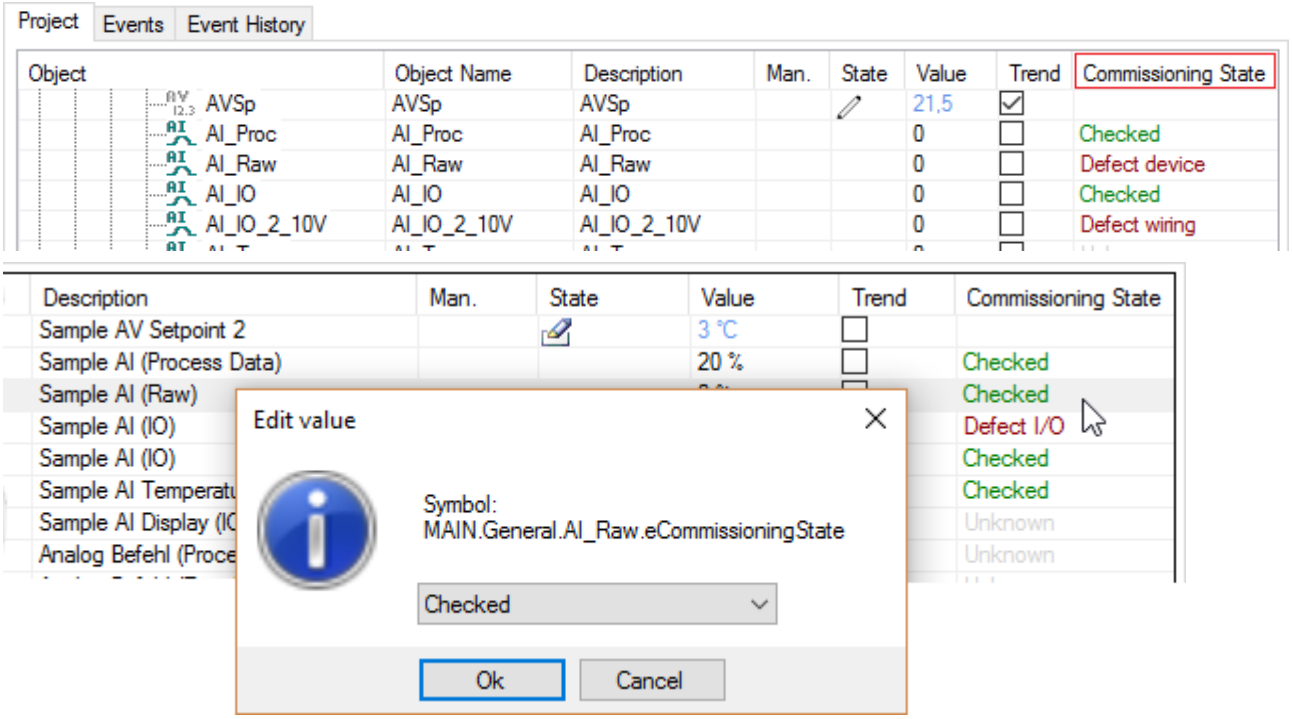
Über das Kontextmenü kann direkt zum markierten Objekt [▶ 33] (in die Projektansicht [▶ 1306]) navigiert werden:



Beispiel: Navigation zum markierten Objekt aus dem Event-Verlauf.

Inbetriebnahme:

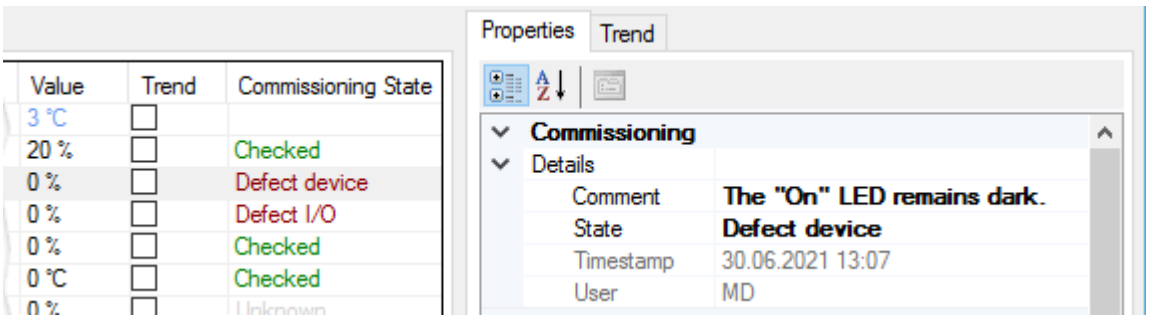
Inbetriebnahme-Zustände einzelner Objekte können in der entsprechenden Spalte angepasst werden:



Beispiel: Oberfläche zur Eingabe des Inbetriebnahme-Zustands eines Objektes.

Details

Inbetriebnahme-Details markierter Objekte werden in den Eigenschaften angezeigt.



Beispiel: Inbetriebnahme-Details eines defekten Feldgeräts.

- **Online**
  Folgende Details werden online (pro Objekt [▶ 33]) in der Steuerung gespeichert.

- Zustand: Aktueller Inbetriebnahme-Zustand.

- **Offline**
  Folgende Details werden offline (pro Objekt [▶ 33]) in den Site-Einstellungen gespeichert.
  - Zeitstempel: Zeitpunkt der letzten Änderung des Inbetriebnahme-Zustands (Falls die Änderung mittels Building Automation Site Explorer erfolgte).
  - Kommentar: Optionaler Kommentar.

Menü



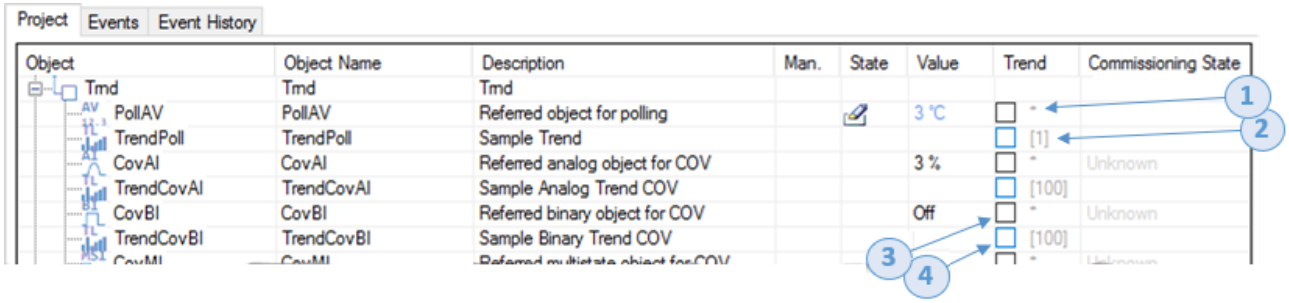
Speichern: Speichert die Inbetriebnahme-Details aller verbundenen Geräte.

Spätestens wenn die Verbindung zu einem Gerät beendet wird, werden die Inbetriebnahme-Details automatisch gespeichert.

Trend

In der Projektansicht können verschiedene Informationen abgelesen werden:

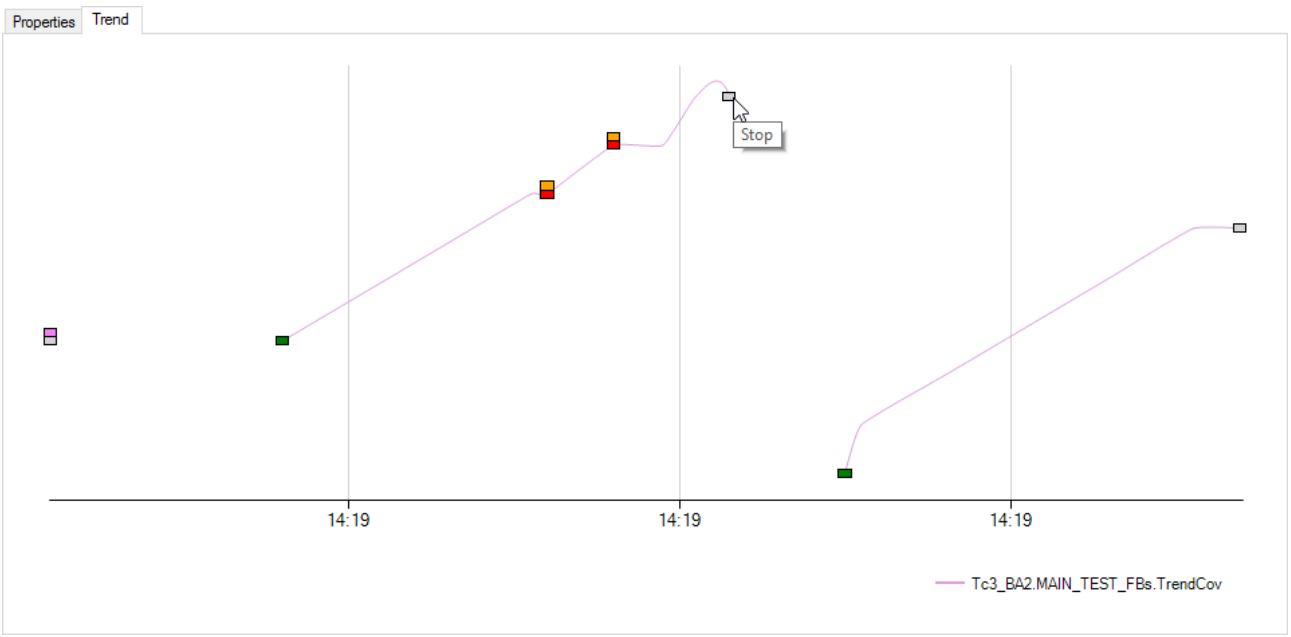
1. Andeutung eines referenzierenden Trend-Objekts [▶ 231].
2. Aktuelle Anzahl an Datensätze des referenzierenden Trend-Objekts [▶ 231].
3. Starten bzw. Beenden der Aufnahme des aktuellen Wertes eines Objekts [▶ 33] (**Online-Trend**).
4. Anzeigen bzw. Entfernen des Log-Buffers eines Trend-Objekts [▶ 231] (**Offline-Trend**).



Beispiel: Darstellung unterschiedlicher Trends.

Offline-Trend

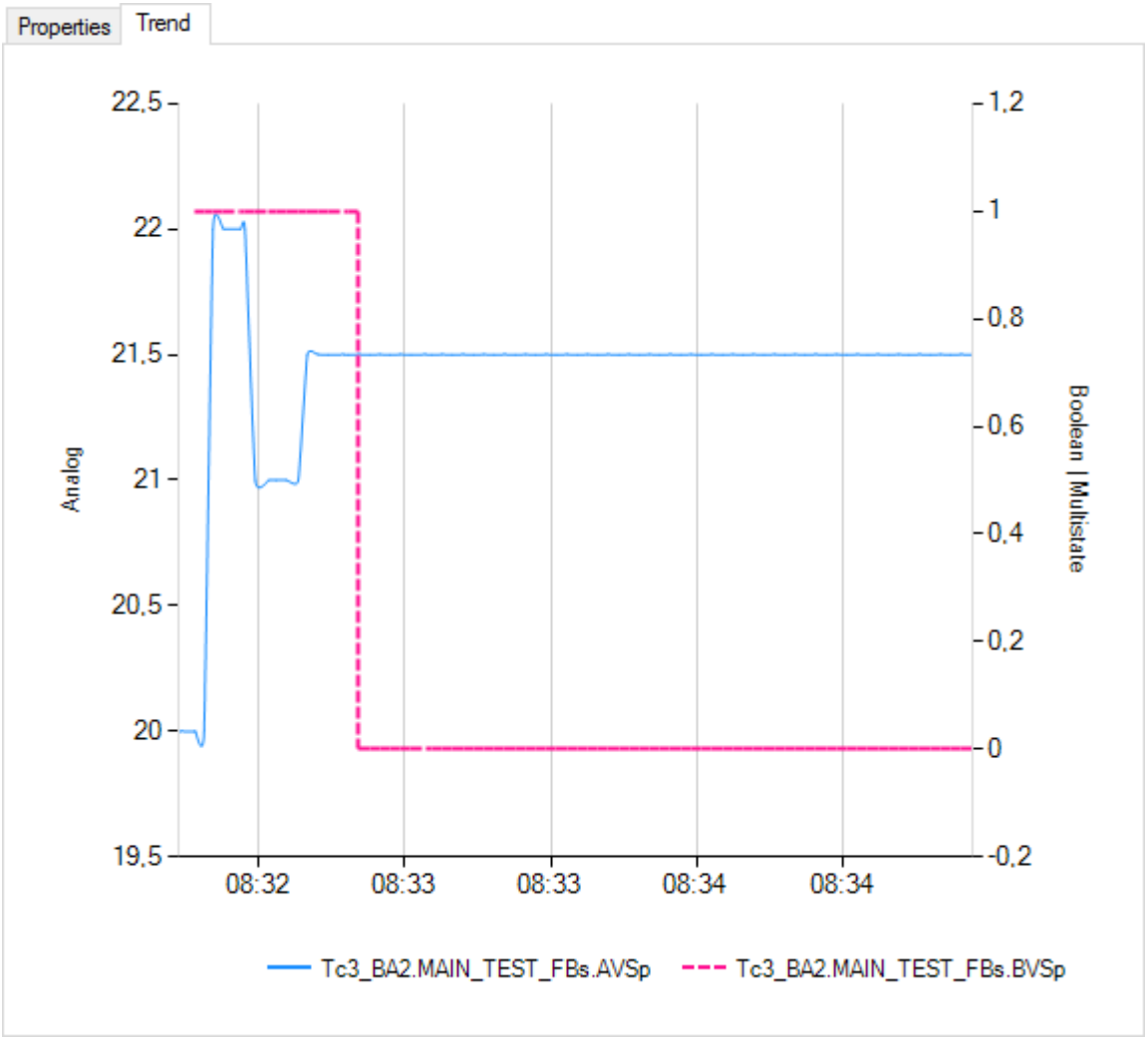
Der **Offline-Trend** liest den Log-Buffer eines Trend-Objekts [▶ 231] aus und stellt diesen in der **Trend-Ansicht** dar.



Einträge werden nicht automatisch aktualisiert!

Online-Trend

Der **Online-Trend** zeichnet den aktuellen Wert eines Objekts [▶ 33] in der **Trend-Ansicht** auf.

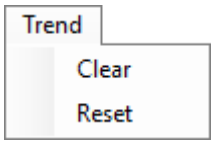


HINWEIS

Datenverlust

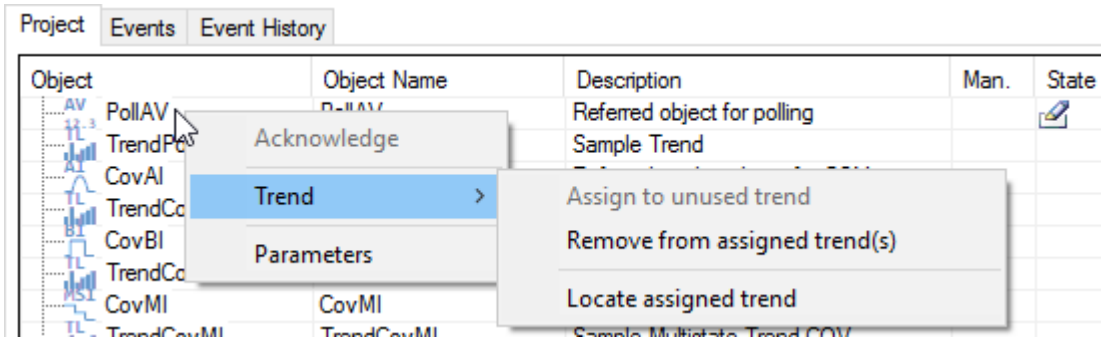
Aufgezeichnete Werte werden nach dem Beenden der Aufnahme oder dem Schließen der Trend-Ansicht nicht gespeichert!

Menü



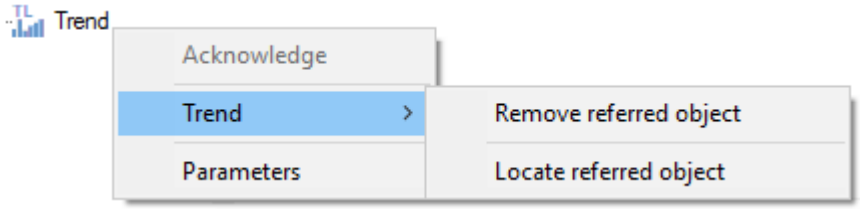
Bezeichnung	Beschreibung
Bereinigen	Entfernt alle Einträge aus der Trend-Ansicht (Entspricht Neustarten der aktuellen Aufzeichnungen).
Zurücksetzen	Entfernt nicht nur alle Einträge aus der Trend-Ansicht, sondern beendet auch das Aufzeichnen sämtlicher Objekte [▶ 33].

Referenzierung



Bezeichnung	Beschreibung
Freien Trend zuweisen	Weist ein Objekt [▶ 33] dem nächsten, ungenutzten Trend-Objekt [▶ 231] zu.
Referenz von zugewiesenen Trend-Objekt(en) entfernen	Entfernt die Objekte [▶ 33]-Referenz von allen referenzierenden Trend-Objekt(en) [▶ 231].
Navigiere zu zugewiesenem Trend	Navigiert zum zugewiesenen Trend-Objekt [▶ 231] (in der Projekt-Ansicht).

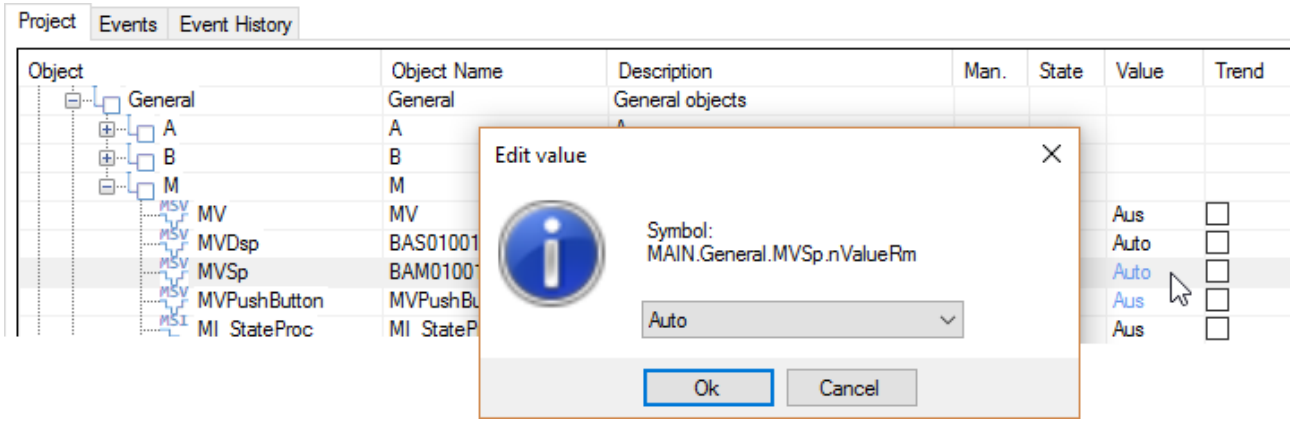
Referenzierende Trend-Objekte



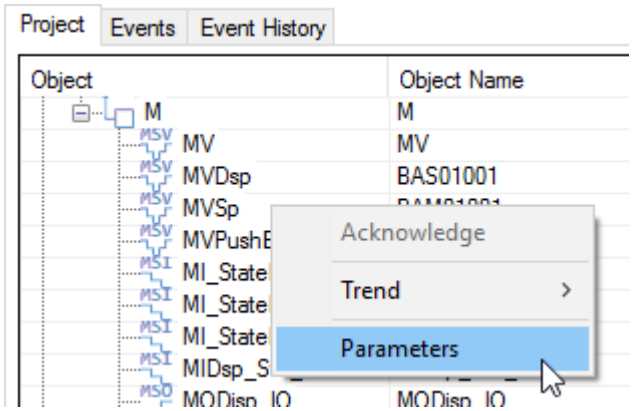
Bezeichnung	Beschreibung
Referenziertes Objekt entfernen	Entfernt die aktuelle Objekt [▶ 33]-Referenz.
Navigiere zu referenziertem Objekt	Navigiert zum zugewiesenen Objekt [▶ 33] (in der Projekt-Ansicht).

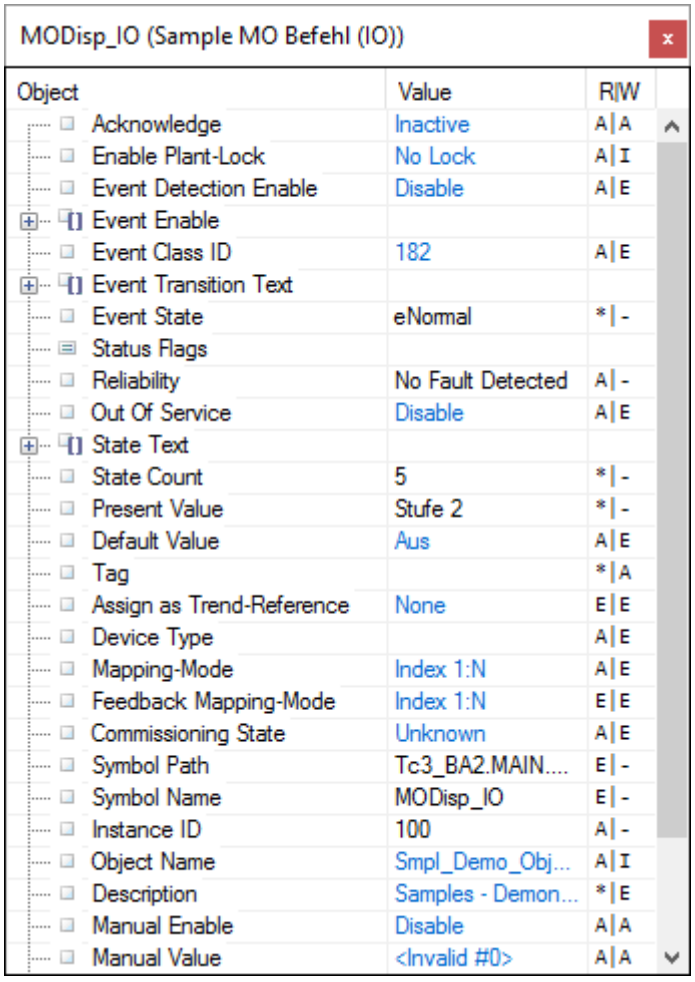
Werte bearbeiten

Aktuelle Werte (falls beschreibbar) können direkt in der Projekt-Ansicht bearbeitet werden.



Bearbeiten von Parametern





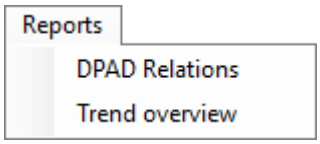
Zugriffsrechte: In der Spalte **R|W** (Read, Write) sind die erforderlichen Zugriffsrechte pro Parameter aufgelistet.

Rolle	Kürzel
Default	*
Advanced	A
Expert	E
Internal	I
Locked	-

Berichte

Berichte sollen einen allgemeinen Überblick über ein Projekt verschaffen. Des Weiteren können auch folgende Anforderungen erfüllt werden:

- Projektdokumentation für den Betreiber
- Nachhaltung von Projekt-Zuständen

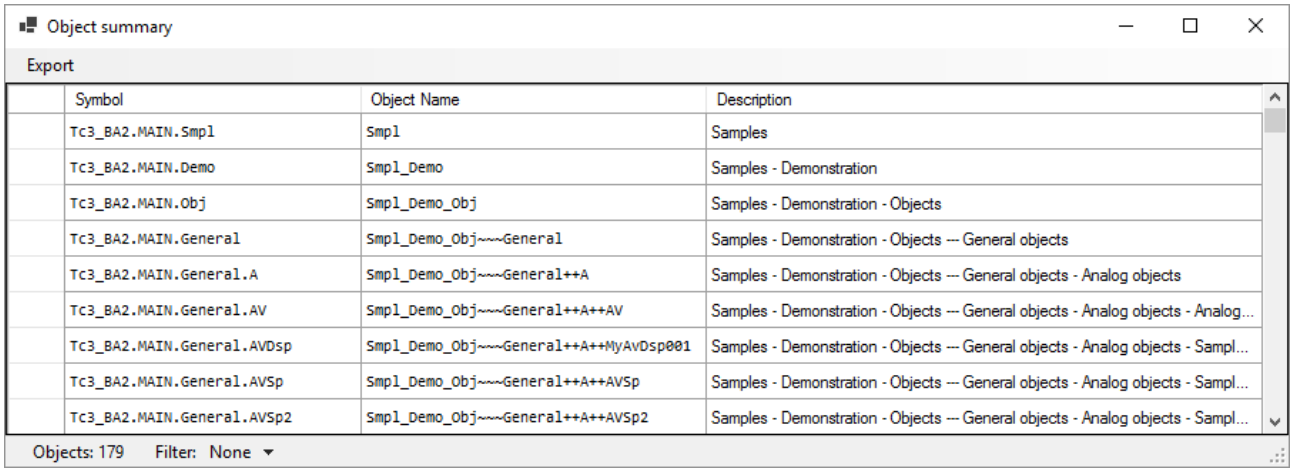


Alle Reports können in die folgenden Formate exportiert werden:

- *.csv

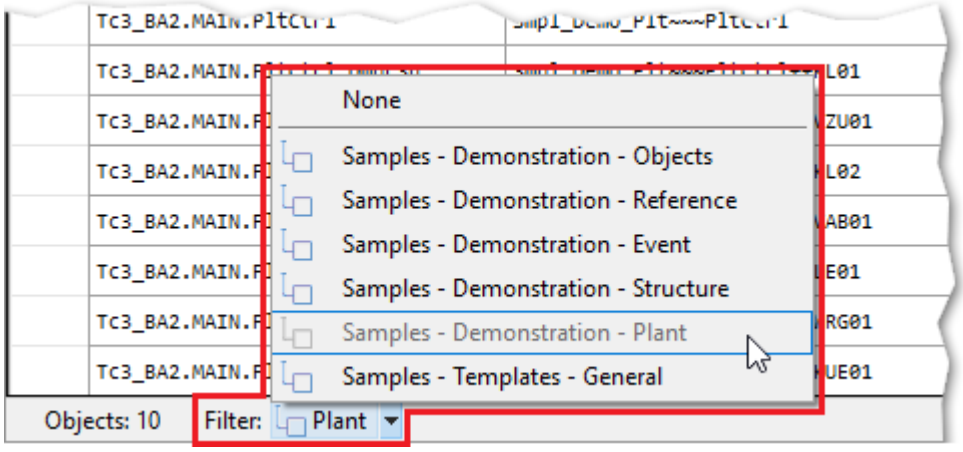
Objekt-Zusammenfassung

Übersicht über alle Objekte [▶ 33] die sich in der verbundenen Site befinden.



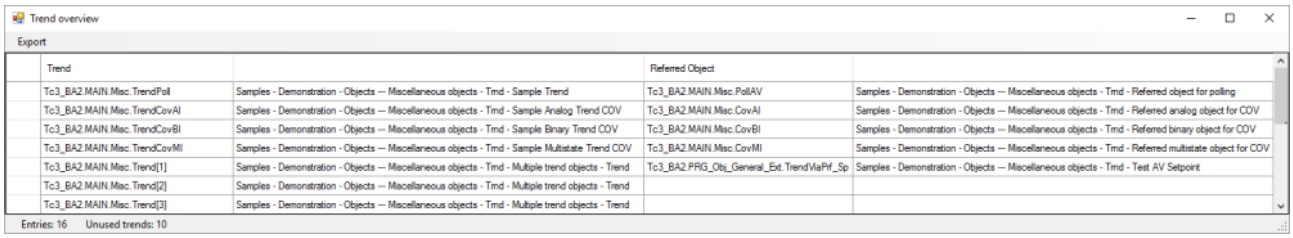
Filter

Um die Übersicht zu erhöhen kann die Ansicht nach Anlagen gefiltert werden:



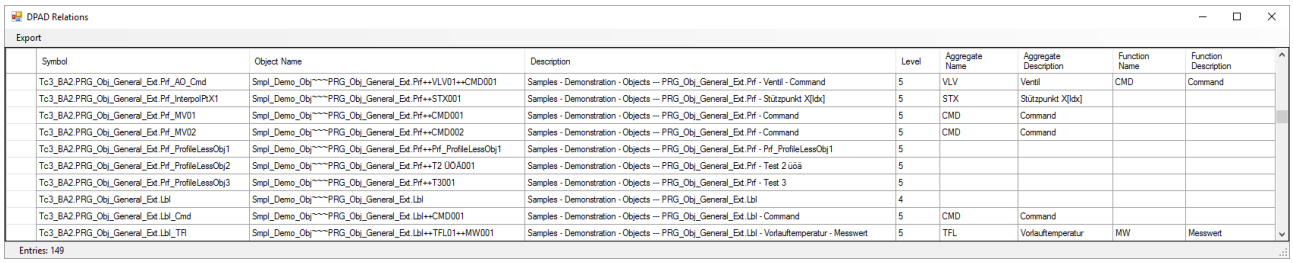
Trend-Übersicht

Übersicht über verfügbare Trends [▶ 231] und aufgezeichnete, bzw. referenzierte Objekte [▶ 33].



DPAD-Beziehungen

Erstellen einer Gegenüberstellung zur Darstellung der DPAD [▶ 43]-Beziehungen.



Enthaltene Informationen pro Objekt [▶ 33]:

Symbolpfad

Eigenschaften

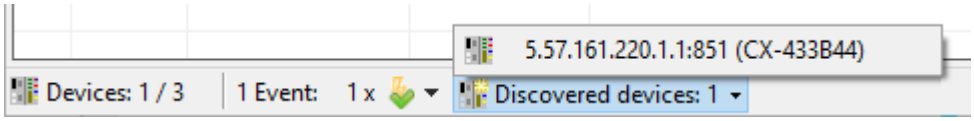
- ObjectName
- Description

Label (Falls verwendet)

- **Aggregate-Informationen**
  - Name
  - Description
- **Funktions-Informationen**
  - Name
  - Description

Entdeckte Geräte

Bei jedem Verbindungsaufbau zur Site werden Verbindungsinformationen aller verfügbaren Geräte aktualisiert.
Befinden sich darunter Verbindungen zu unbekannten Geräten (welche nicht Bestandteil der aktuellen Site-Konfiguration sind), werden diese als entdeckte Geräte am unteren Rand der Anwendung aufgelistet:

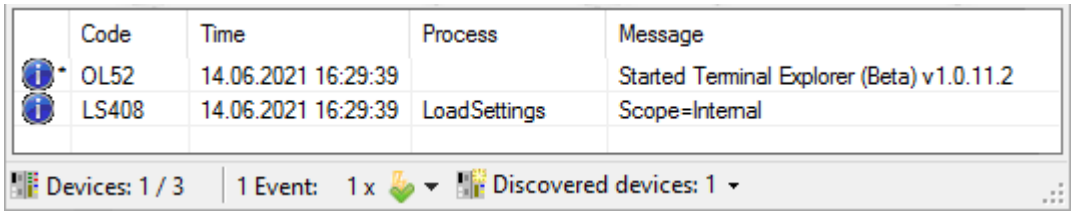


Um ein entdecktes Gerät der Site-Konfiguration hinzuzufügen, genügt ein Klick auf den entsprechenden Menü-Eintrag.

Entdeckte Geräte sind mögliche Vorschläge.
Sie werden angezeigt, da angenommen werden kann, dass sie zur aktuellen Site-Konfiguration gehören.

Log und Status

Meldungen und Zustände werden im Log oder in der Statusleiste am unteren Bereich der Anwendung ausgegeben, bzw. angezeigt:



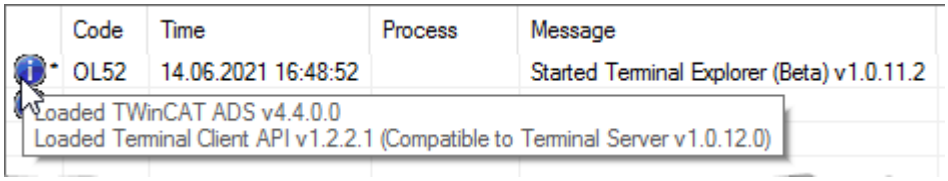
Log-Einträge können zusätzliche Informationen ( ) enthalten.

Diese können abgerufen werden, indem der Mauszeiger über das * bewegt wird.
Folgende Funktionen können über das Kontextmenü aufgerufen werden:

- **Markierte Einträge kopieren:** Kopiert markierte Einträge in die Zwischenablage.
- **Alle Einträge kopieren:** Kopiert alle Einträge in die Zwischenablage.
- **Exportieren:** Kopiert alle Einträge in eine Datei.
- **Bereinigen:** Entfernt alle Einträge aus der Ansicht.

Version

Details zu Versionen werden beim Starten der Anwendung im Log ausgegeben:



- **Terminal Explorer**

```
[OL52] Started TerminalExplorer v1.0.11.2
```

- **Terminal Client API**

```
Loaded terminal client API v1.2.2.1 (Compatible to terminal server v1.0.12.0)
```

Der Eintrag enthält zwei zu unterscheidende Informationen:

- Version der geladenen **Terminal Client API** DLL
- Version des Terminal Servers zu dem die geladene Terminal Client API kompatibel ist.

HINWEIS

Version beachten

Es können nur Verbindungen zu ADS-Geräten aufgebaut werden, die mit dieser Version arbeiten!

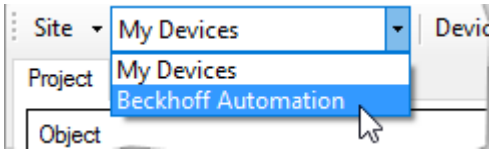
- **TwinCAT ADS**

```
Loaded TwinCAT ADS v4.4.0.0
```

Toolbar

Sites

Auswahl an konfigurierten Sites.

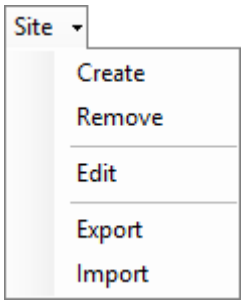


Anwendungsfälle

Für die Bereitstellung von Sites stehen zwei Anwendungsfälle zur Auswahl:

- **Lokal**: Für eigene Zwecke (z. B. für Tests) können Site- und Geräte-Konfigurationen lokal (im Anwendungsverzeichnis) gespeichert werden.
- **Referenz**: Um die Zusammenarbeit in Teams zu verbessern, ermöglichen Referenzen das Speichern von Site- und Geräte-Konfigurationen in beliebigen Verzeichnissen (z. B. Netzlaufwerke oder Git-Repositories).
  Beteiligte Personen arbeiten somit auf derselben Grundlage und vermeiden so Seiteneffekte wie:
  - Unterschiedliche Auswahl und Konfiguration von Geräten
  - Inkonsistente Inbetriebnahme Zustände

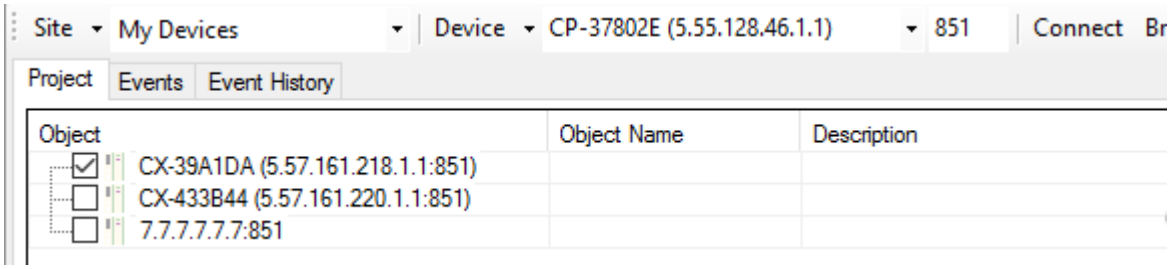
Verwaltung von Sites



Bezeichnung	Beschreibung
Erzeugen	Legt eine neue Site an.
Entfernen	Entfernt eine existierende Site.
Bearbeiten	Zeigt die Site-Eigenschaften zum Bearbeiten an.
Exportieren	Lagert eine Site wegen der Referenzierung in ein bestimmtes Verzeichnis aus.
Importieren	Importiert eine Site zur Verwendung als Referenz.

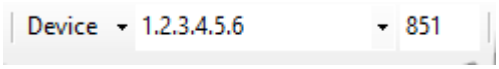
Verbindung

Beim Verbindungsaufbau der Site wird die Kommunikation zu allen ausgewählten Geräten aufgebaut:



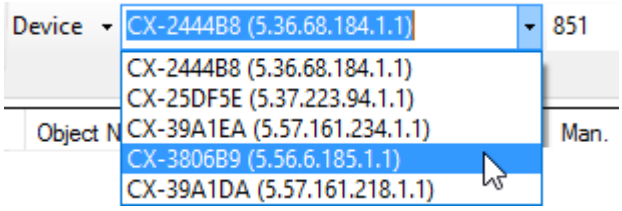
Geräte verwalten



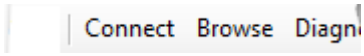
Bezeichnung	Beschreibung
Hinzufügen	Fügt ein neues Gerät zur aktiven Site hinzu. Das Hinzufügen von Geräten durch manuelles Eingeben der *AMS NetID* ist möglich. Es müssen alle Voraussetzungen (wie z. B. das Einrichten der ADS-Route) erfüllt sein, bevor eine Verbindung aufgebaut werden kann. 
Entfernen	Entfernt ein ausgewähltes Gerät aus der aktiven Site-Konfiguration.

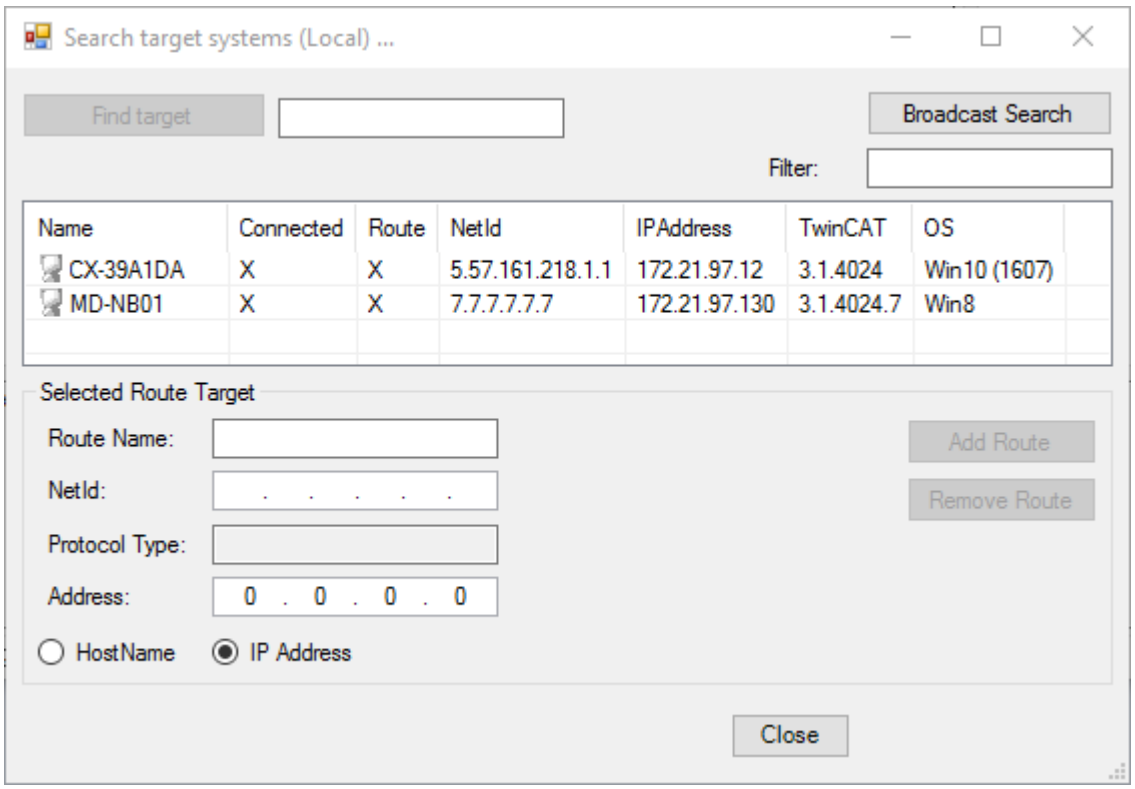
Routen verwalten

- **Route auswählen:** Über die Auswahlbox können bereits angelegte Routen ausgewählt werden:



- **Neue Route anlegen:** Das Dialogfeld zum Suchen von Geräten kann über den Durchsuchen Button geöffnet werden:



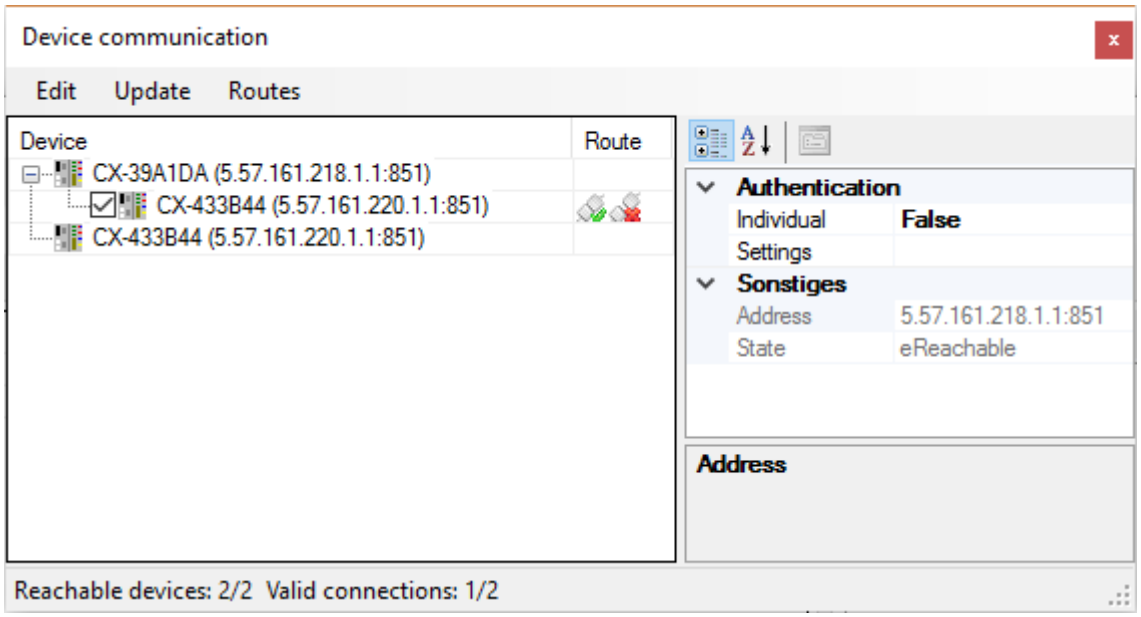


Diagnose

Diagnose-Funktionen sind für konfigurierte Geräte der aktiven Site über das Menü zugänglich:

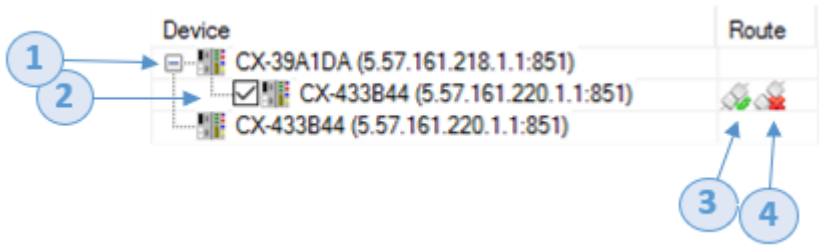


Geräte-Kommunikation



Verbindungen und Zustände

Zeigt eine Auflistung aller konfigurierten Geräte zur Visualisierung der Kommunikation einzelner Geräte untereinander an.



Wenn ein Gerät (1) für die Kommunikation (mittels Remote Subscriptions) vorgesehen ist, erscheinen alle **Ziel-Geräte** (2) jeweils als Verbindung unter dem **Quell-Gerät** (1).

Es werden auch *Ziel-Geräte* (2) angezeigt, die nicht Bestandteil der aktiven Site-Konfiguration sind.

Die individuellen Zustände der Routen auf den jeweiligen Geräten werden durch entsprechende Symbole angedeutet:

- Das linke Symbol (3) repräsentiert den Zustand der Route auf dem **Quell-Gerät** (1).
- Das rechte Symbol (4) repräsentiert den Zustand der Route auf dem **Ziel-Gerät** (2).

Legende

- **Geräte**

Grafik	Beschreibung
	Gerät nicht erreichbar.
	Gerät erreichbar.

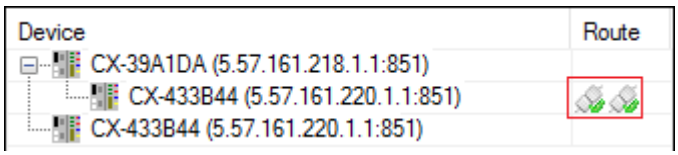
- **Routen**

Grafik	Beschreibung
	Route unbekannt, wenn der Zustand nicht ermittelt werden konnte (z. B., wenn das Gerät nicht erreichbar ist).
	Route fehlend.
	Route gültig.

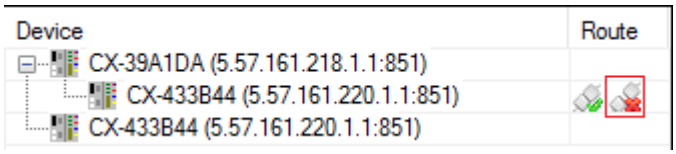
Route anlegen

Die Kommunikation innerhalb einer Site ist funktionsfähig, wenn die Routen aller kommunizierenden Geräte untereinander eingerichtet sind.

Beispiel1: Routen auf Quell- und Zielgerät gültig:



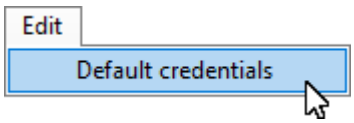
Beispiel 2: Route auf Quell-Gerät gültig, nicht jedoch auf dem Ziel-Gerät:



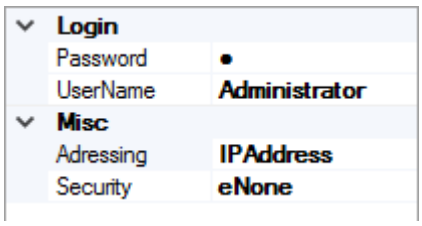
Fehlende Routen (sowohl aller als auch für ausgewählte Geräte) können mit einem Mal konfiguriert werden:

✓ **Zugangsdaten bereitstellen:** In den meisten Fällen werden einheitliche Zugangsdaten in allen Geräten verwendet.

1. Eigenschaften für Standard-Zugangsdaten öffnen:

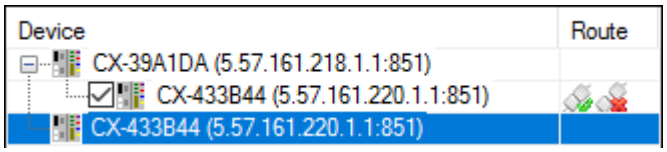


2. Anpassungen über Eigenschaften vornehmen:

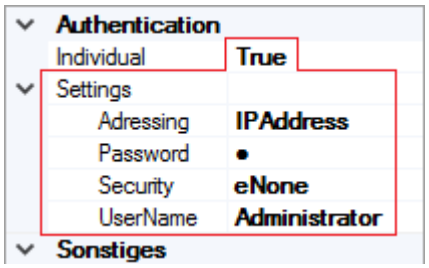


⇨ Wenn für verschiedene Geräte individuelle Zugangsdaten hinterlegt werden müssen, können die Standard-Zugangsdaten wie folgt überschrieben werden:

3. Gerät auswählen:

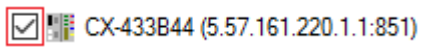


4. Individuellen Zugangsdaten aktivieren und bearbeiten:

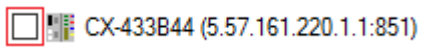


⇨ **Geräte zur Routen-Konfiguration auswählen**

5. Geräte, mit ungültigen Routen-Einstellungen, werden eingangs automatisch zum Konfigurieren ausgewählt:

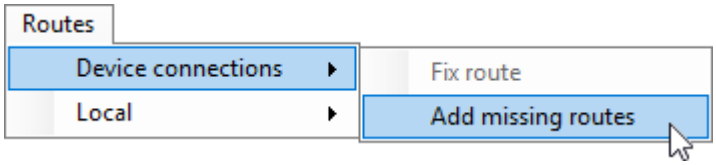


6. Falls Geräte nicht für das Konfigurieren der Route vorgesehen werden sollten, können sie abgewählt werden:

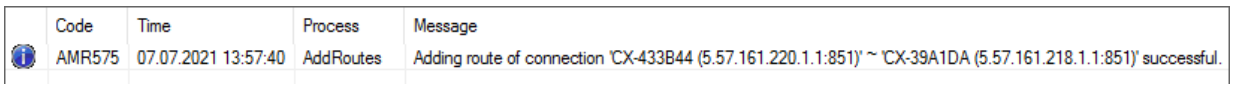


⇨ **Konfiguration anwenden:**

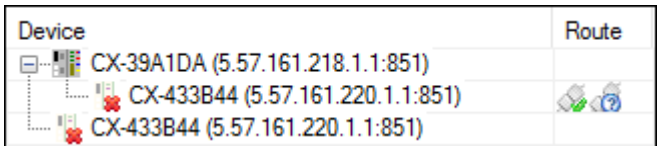
7. Die Routen-Konfiguration der ausgewählten Geräte kann über das Menü ausgerollt werden:



⇨ Alle Konfigurations-Vorgänge werden im Log protokolliert:



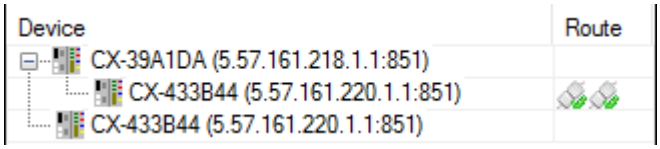
⇨ Für nicht verfügbare Geräte (z. B. nicht erreichbar oder Route nicht konfiguriert) können weder Routen-Zustände angezeigt noch Routen-Konfigurationen übernommen werden:



Verbindung reparieren

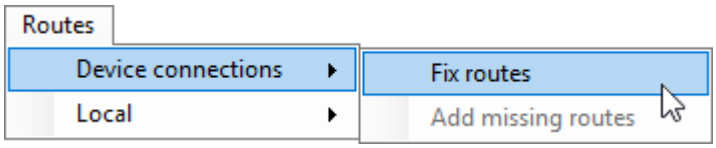
Trotz bestehender Routen-Konfiguration kann die Verbindung zwischen zwei Geräten fehlerhaft sein.

In diesem Fall sind Routen auf beiden Seiten zwar eingerichtet, jedoch mit fehlerhaften Parametern (z. B., wenn sich eine IP-Adresse geändert hat).



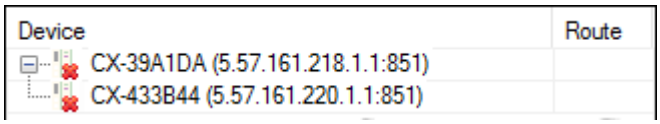
Beispiel: Anzeige einer gültigen Verbindung aufgrund von beidseitig eingerichteten Routen.

Eine (markierte) Verbindung kann über das Menü repariert werden.
Dabei werden die Routen auf beiden Geräten erneut konfiguriert.

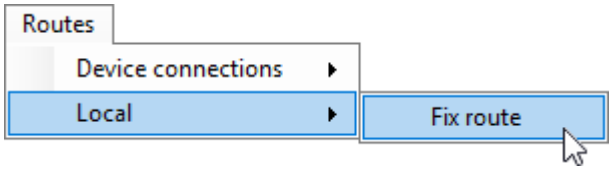


Lokale Route reparieren

Trotz bestehender Routen-Konfiguration kann die lokale Verbindung zu einem Ziel-Gerät fehlerhaft sein.
In diesem Fall ist die Route zwar eingerichtet, jedoch mit fehlerhaften Parametern (z. B., wenn sich die IP-Adresse des Ziel-Geräts geändert hat).



Routen zu nicht erreichbaren Geräten können über das Menü repariert werden:



7.2 Symbol Explorer

7.2.1 Einleitung

Mit dem Symbol Explorer können Sie online über die Kommunikationsschnittstelle ADS auf Steuerungen zugreifen:

- Sie können die als persistent deklarierten Variablen auslesen.
- Der Symbol Explorer stellt Funktionen bereit, mit denen Variablen-Backups erstellt werden können. Diese können für Folgeprojekte vervielfältigt und auf andere Steuerungen hochgeladen werden.
- Mit einer Vergleichsfunktion können Variablen auf Unterschiede geprüft und zusammengeführt werden.

7.2.2 Begriffserklärung

7.2.2.1 Symbol

Ein Symbol ist ein Typ zum Beschreiben von Variablen einer Beckhoff-Steuerung. Werden über den Symbol Explorer Variablen aus einer Steuerung gelesen, werden Informationen wie Name, Typ, Größe, Untervariablen und viele andere Parameter zu einer Variablen zusammengefasst. Aus dieser Menge von Parametern, wird anschließend ein Symbol gebildet.

Symboltypen

Ein Symbol kann verschiedene Typen von Variablen beschreiben.

Komplexe Symbole beschreiben Variablen, die aus Funktionsblöcken, Arrays und Strukturen bestehen können.

Primitive Symbole beschreiben Variablen, die aus Basistypen, INT, REAL, BOOL usw. bestehen können.

7.2.2.2 Snapshot

Ein Snapshot beschreibt eine Momentaufnahme eines Symbols oder einer Symbolstruktur.

Die Funktion, einen Snapshot zu erstellen, ist für mehrere Anwendungsfälle vorgesehen. Sie können einen Snapshot der gesamten Symbole erstellen und haben somit ein komplettes Backup der Symbole auf einer Steuerung. Sie können aber auch über die verschiedenen Darstellungen und Filterfunktionen der Symbolliste, nur einen „Ausschnitt an Symbol/Symbolen“ erstellen und diese in einen Snapshot speichern. Sie haben somit eine Vorlage zur vereinfachten Vervielfältigung von Symbolwerten erstellt.

7.2.3 Benutzeroberfläche

Die Oberfläche gliedert sich in verschiedene Fenster mit unterschiedlichen Ausgaben zur Bedienung des Symbol Explorers.

7.2.3.1 Main-Fenster

Nach dem Öffnen der Startseite ist es möglich die folgenden Optionen zu wählen:

- Recent snapshots
- Recent routes
- Connect
- Snapshot

Main-Menü

Das Menü gliedert sich in folgende Reiter.

Open

Befehl	Beschreibung
Choose route	Mit einer Route verbinden.
Open snapshot	Snapshot öffnen.
Recent snapshots	Liste häufig benutzter Snapshots.
Recent routes	Liste häufig benutzter Routen.

View

Befehl	Beschreibung
Output	Ouput Fenster klein / groß schalten.

Tools

Befehl	Beschreibung
Compare symbols	Vergleich von zwei Symbollisten.

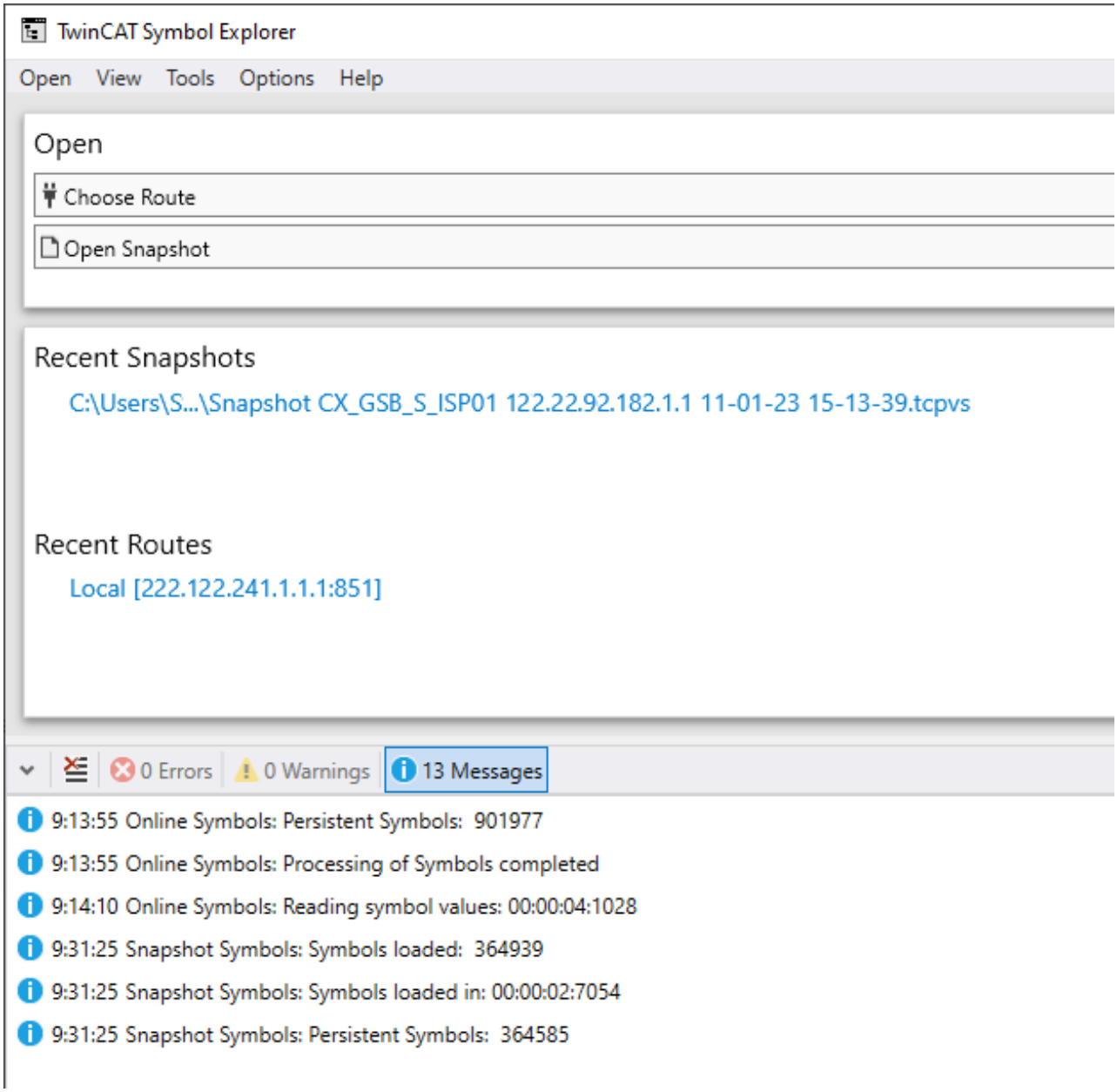
Help

Befehl	Beschreibung
About TwinCAT Symbol Explorer	Versionsangabe

7.2.3.2 Startseite

Die Startseite gibt einen schnellen Überblick über die zuletzt gewählten Routen oder geöffneten Snapshots.

Des Weiteren kann mit einem Klick eine der letzten Routen oder Snapshots direkt angewählt und geöffnet werden.



Open

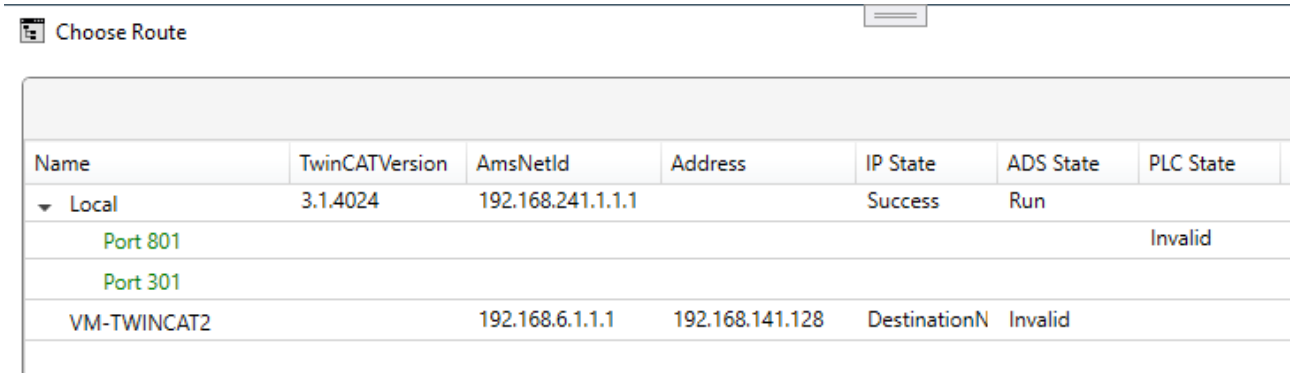
Befehl	Beschreibung
Choose Route	Mit einer Route verbinden.
Open Snapshot	Snapshot öffnen

Recent

Befehl	Beschreibung
Recent Snapshots	Liste häufig benutzter Snapshots.
Recent Routes	Liste häufig benutzter Routen.

7.2.3.3 Choose Route

Der "Choose Route"-Dialog bietet eine schnelle und einfache Ansicht der Routen, die auf dem System registriert sind. Zu jeder Route werden die Ports angezeigt, die für den Symbol Explorer zu erreichen sind. Diese sind die voreingestellten Ports für den Zugriff auf die SPS (TwinCAT2 / TwinCAT3) oder speziell konfigurierte Ports für den Zugriff auf z.B. den EtherCAT Master unter dem Prozess Image. Die Ports können über den Options-Dialog konfiguriert werden.



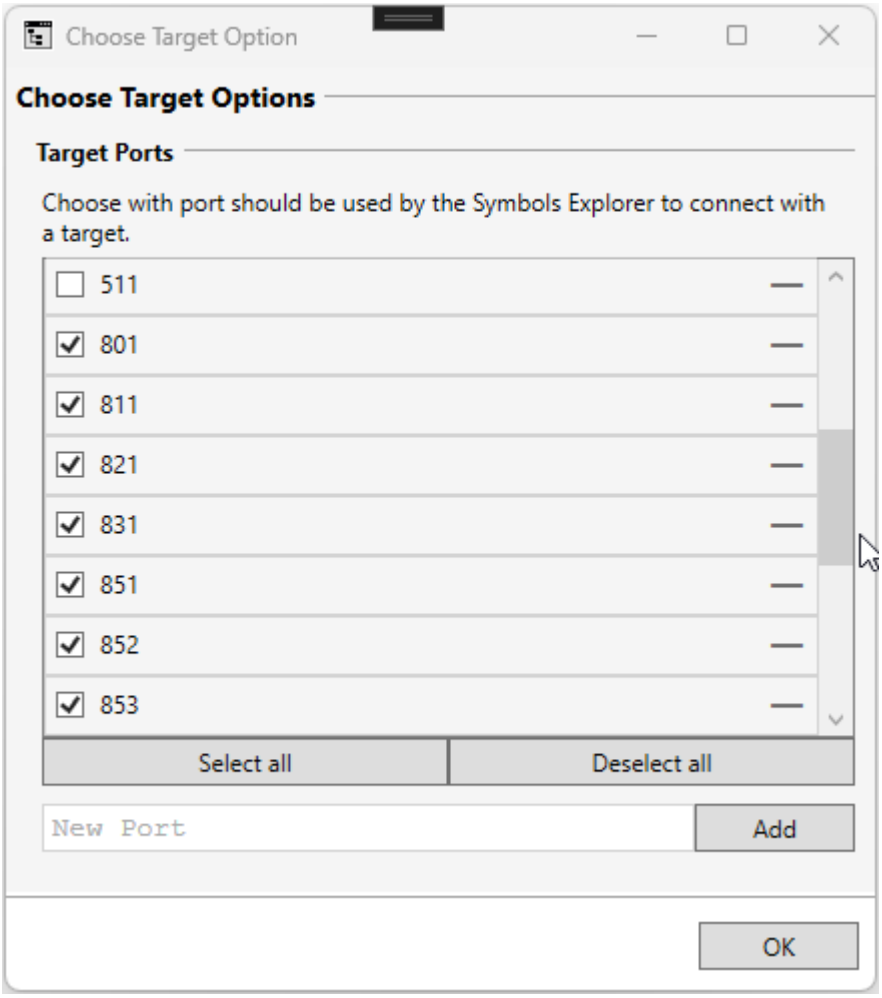
Toolbar

Symbol	Befehl	Beschreibung
	Options	Choose Route Dialog Optionen öffnen
	Refresh Route	Status der Routen aktualisieren

Options Dialog

Hier kann ausgewählt werden, welche Ports vom Symbol Explorer benutzt werden sollen, um sich mit einem Traget zu verbinden.

Es kann aus den vorgegebenen Ports gewählt oder ein spezifischer Port hinzufügefügt werden.



Standard Portnummern

Beschreibung	Ports
MOTION	500, 501, 511
Tc2 PLC	801, 811, 821, 831
Tc3 PLC	851, 852, 853, 854, 855
IO IMAGE	27905, 27906, 27907, 27908, 27909, 27910
TC3 TASK	350, 351, 352, 353, 354, 355

7.2.3.4 Symbol Entry Point

Der Symbol Entry Point beschreibt den oder die Eintrittspunkte, die der Symbol Explorer in die symbolische Struktur eines Targets nehmen soll. Dabei können die benötigten Symbole ausgewählt werden. Zudem stehen verschiedene Filter zur Verfügung, die die Anzahl an zu ladenden Symbolen reduzieren können.

Symbol Type Selection

Dies ist ein grober Filter, der auf einen Symbolbereich angewendet wird (z. B. 'Persistent'). Ist der Filter ausgewählt, werden nur das Symbol und das Elternsymbol angezeigt, die mit dem jeweiligen Scope (z. B. 'Persistent') in der PLC markiert sind.

Symbol Datatypes

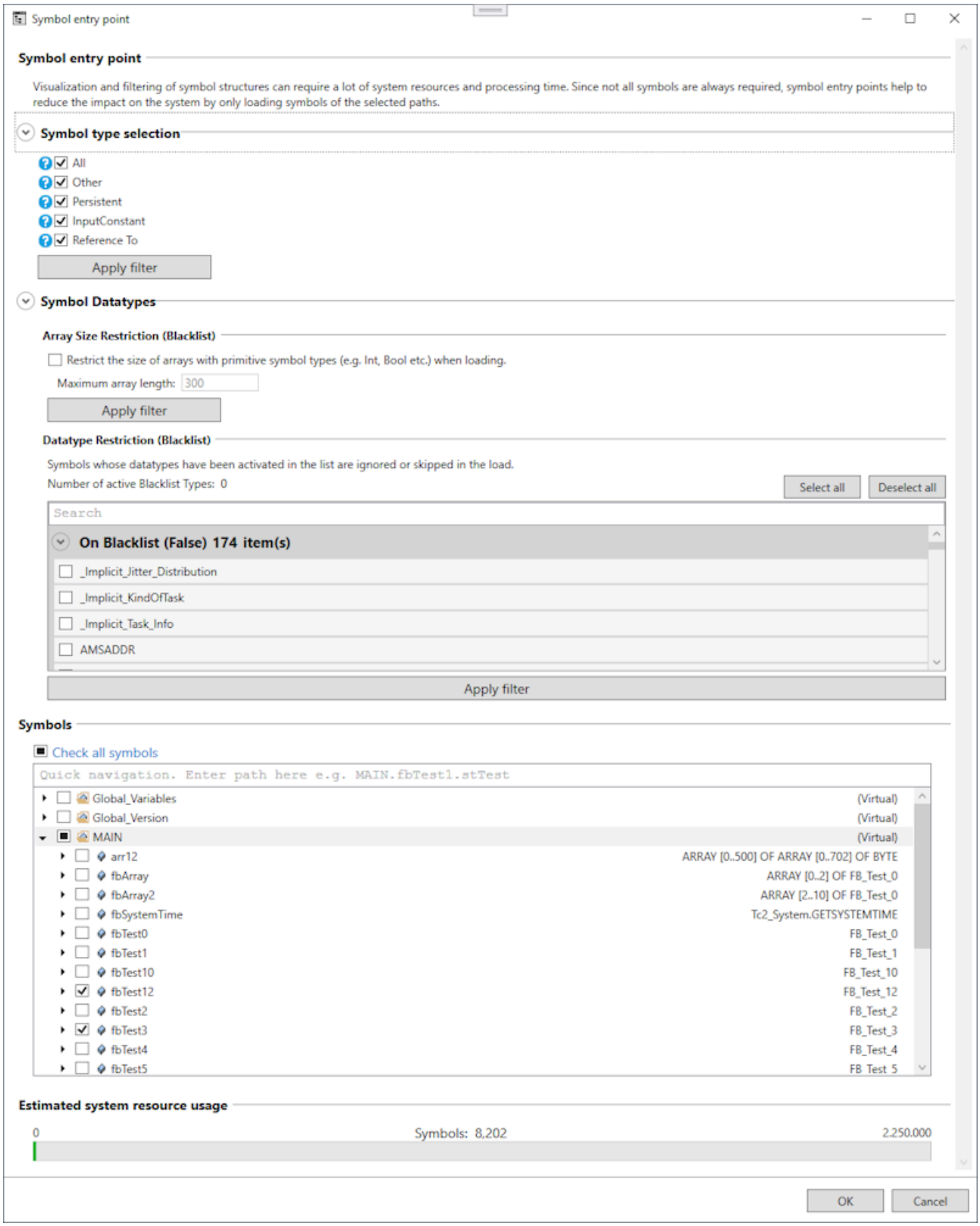
Ein Filter, der auf den Datentyp oder eine Besonderheit des Datentyps angewendet wird.

Array Größen Beschränkung

Ist der Filter aktiv, werden sämtliche Arrays mit einem Primitiven Datentypen, die größer als der Wert für 'Maximale Array Länge' sind, ausgeschlossen.

Datatype Blacklist

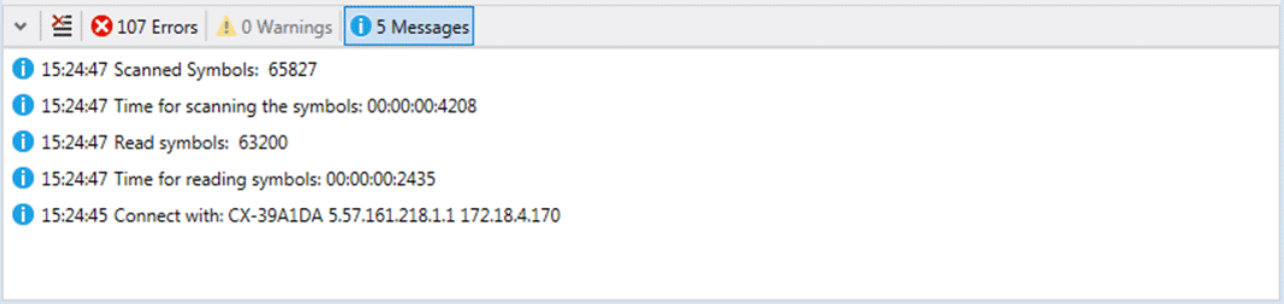
Symbole, dessen Datentyp in der Blacklist stehen, werden ausgeschlossen.



7.2.3.5 Output-Fenster

Das Output-Fenster gibt Informationen, Fehler und Warnungen zur Laufzeit aus.

Die Funktionen der Toolbar sind in der folgenden Tabelle beschrieben.

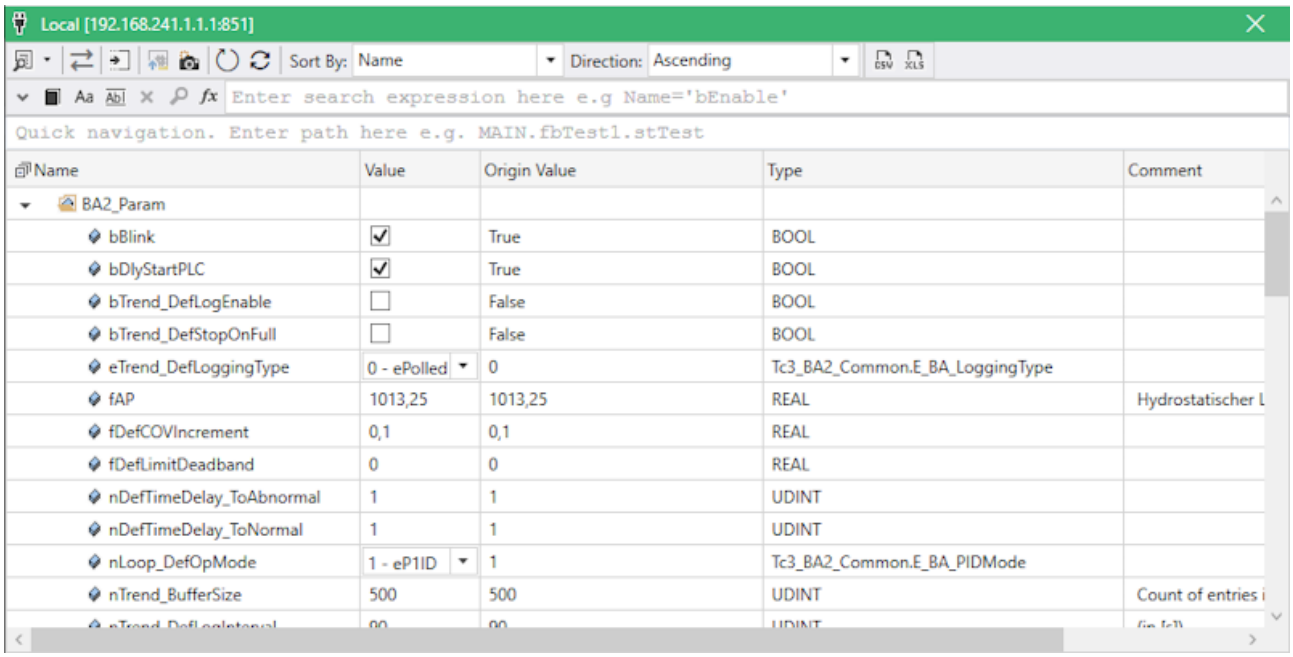


Toolbar

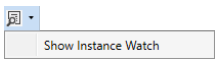
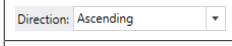
Symbol	Befehl	Beschreibung
	Expand / Collapse	Auf- und Zuklappen des Output-Fensters
	Clear messages	Löschen der Nachrichten
107 Errors	Show / Hide errors	Ein- und Ausblenden der Fehlermeldungen
0 Warnings	Show / Hide warnings	Ein- und Ausblenden der Warnungen
5 Messages	Show / Hide messages	Ein- und Ausblenden der Meldungen

7.2.3.6 Online-Fenster

Das Online-Fenster zeigt die Online-Symbole einer Steuerung an. Über die Toolbar und die Kontextmenüs stehen Funktionen zum Erstellen von Backups, Beobachten und Bearbeiten von Symbolen zur Verfügung.



Toolbar

Symbol	Befehl	Beschreibung
	Show symbols as list	Symbole als flache Liste anzeigen.
	Drop down box	Erweiterte Symbollisten.
	Show instance watch	Symbolinstanzen Ansicht.
	Synchronize symbols	Dialog zum Abgleichen von Symbolen anzeigen.
	Sort by	Sortierung der Symbole nach z.B. Name oder Größe.
	Sort direction	Richtung der Sortierung.
	Upload symbols	Schreibt geänderte Symbole in die Steuerung
	Take snapshot	Snapshot von den Symbolen erstellen
	Refresh	Symbolwert einmalig aktualisieren
	Auto refresh	Symbolwert alle 2 Sekunden aktualisieren
	Export / Import CSV	Symbole als CSV-Datei exportieren oder importieren
	Export symbols as Excel	Symbole als Excel-Tabelle exportieren.
	Close window	Fenster schließen

Ausnahmen beim Refresh

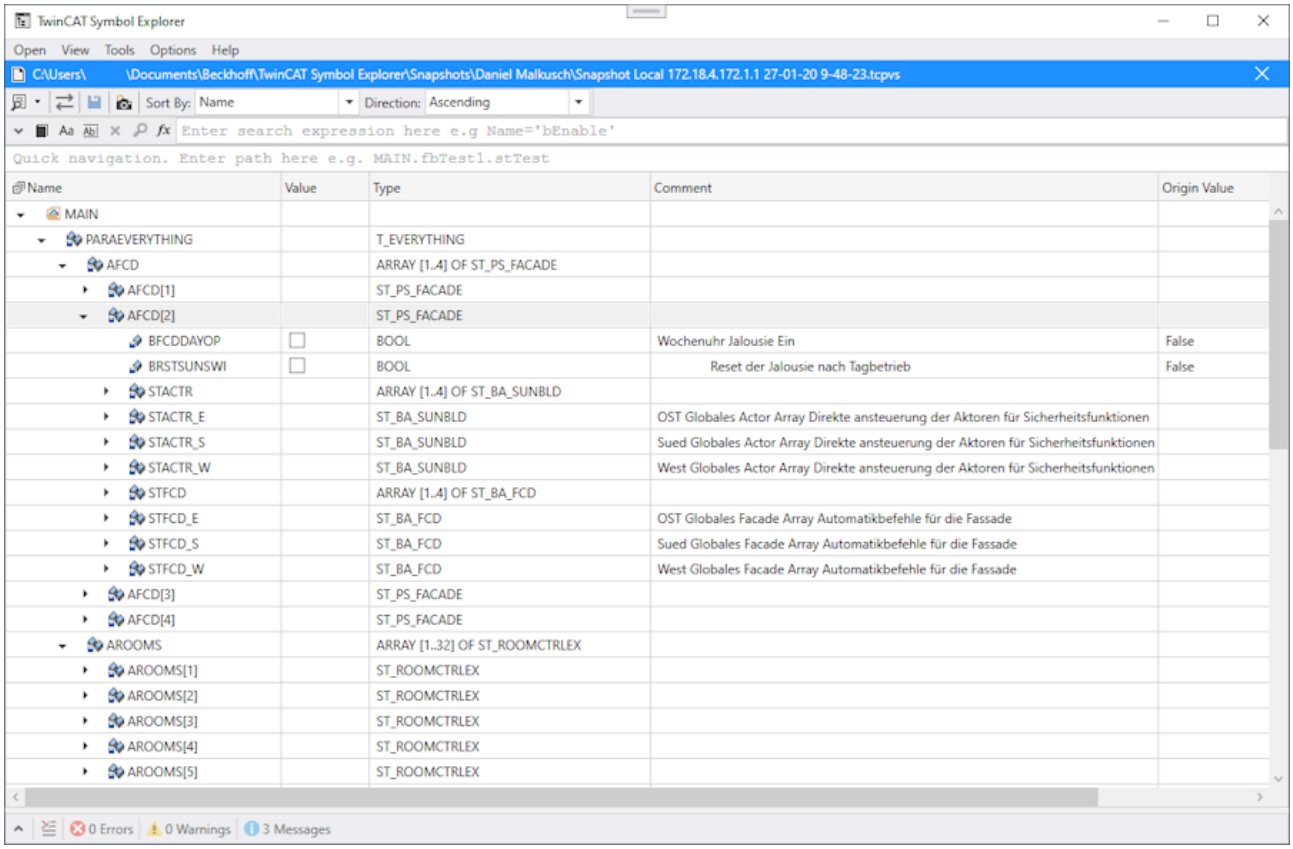
Vom Refresh ausgenommen sind Symbole, die über den Symbol Explorer bearbeitet wurden und Symbole, die gerade in Bearbeitung sind.

Hinweise zur ADS-Kommunikation

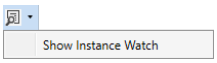
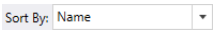
Der Symbol Explorer verwendet für die Online-Kommunikation das ADS-Protokoll. Beachten Sie, dass ADS nur eine Transportschicht ist, es können jedoch Nebeneffekte auftreten. Lesen Sie diese Anforderungen und beachten Sie die Einschränkungen:
ADS selbst ist nur die Transportschicht, das angeforderte ADS-Gerät muss den ADS-Befehl unterstützen.
Wenn die SPS eine ADS-Anforderung bearbeitet (Lesen/Schreiben der Symbolwerte), wird sie vor dem Start eines neuen SPS-Zyklus vollständig an dieser einzelnen ADS-Anforderung arbeiten.
Um die Belastung auf die Steuerung niedrig zu halten, wurde aus diesem Grund die Anzahl der zu lesenden bzw. schreibenden Symbole auf maximal 250 pro ADS-Anforderung beschränkt.

7.2.3.7 Snapshot-Fenster

Das Snapshot-Fenster zeigt die Offline-Symbole aus einem Snapshot an. Über die Toolbar und Kontextmenüs stehen Funktionen zum Bearbeiten, Hochladen und Erstellen von Kopiervorlagen zur Verfügung.

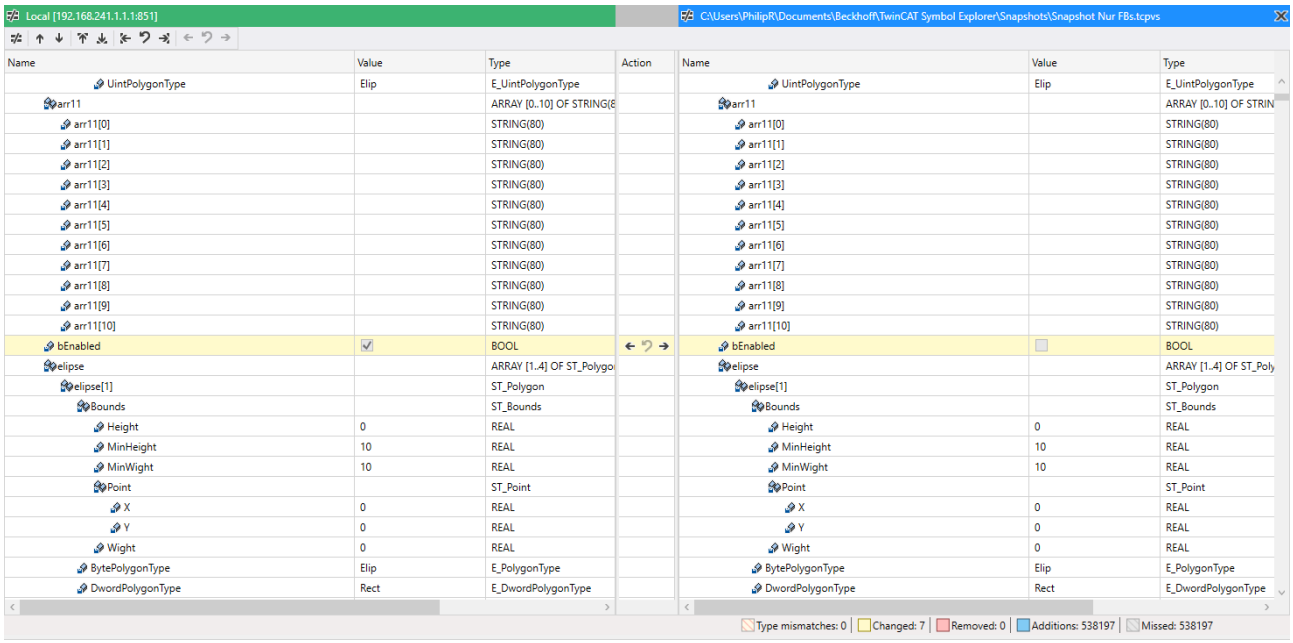


Toolbar

Symbol	Befehl	Beschreibung
	Show symbols as list	Symbole als flache Liste anzeigen.
	Drop down box	Erweiterte Symbollisten.
	Show instance watch	Symbolinstanzen Ansicht.
	Synchronize symbols	Dialog zum Abgleichen von Symbolen anzeigen.
	Sort by	Sortierung der Symbole nach z.B. Name oder Größe.
	Sort direction	Richtung der Sortierung.
	Save	Snapshot speichern.
	Take snapshot	Snapshot von den Symbolen erstellen
	Export / Import CSV	Symbole als CSV-Datei exportieren oder importieren
	Export symbols as Excel	Symbole als Excel-Tabelle exportieren.
	Close window	Fenster schließen

7.2.3.8 Vergleich-Fenster

Das Symbol-Vergleich-Fenster zeigt Offline-Symbole aus einem Snapshot an. Über die Toolbar und Kontextmenüs stehen Funktionen zum Bearbeiten, Hochladen und Erstellen von Kopiervorlagen der Symbole zur Verfügung.



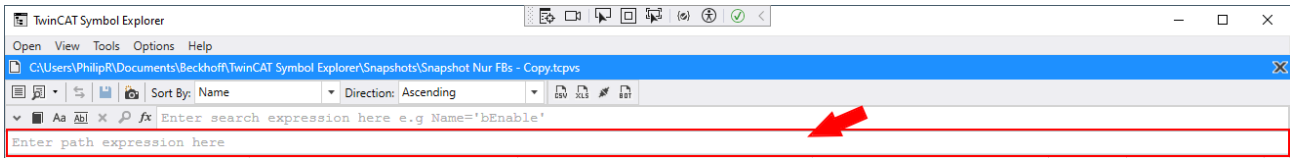
Das Symbol-Vergleich-Fenster stellt folgende Unterschiede dar:

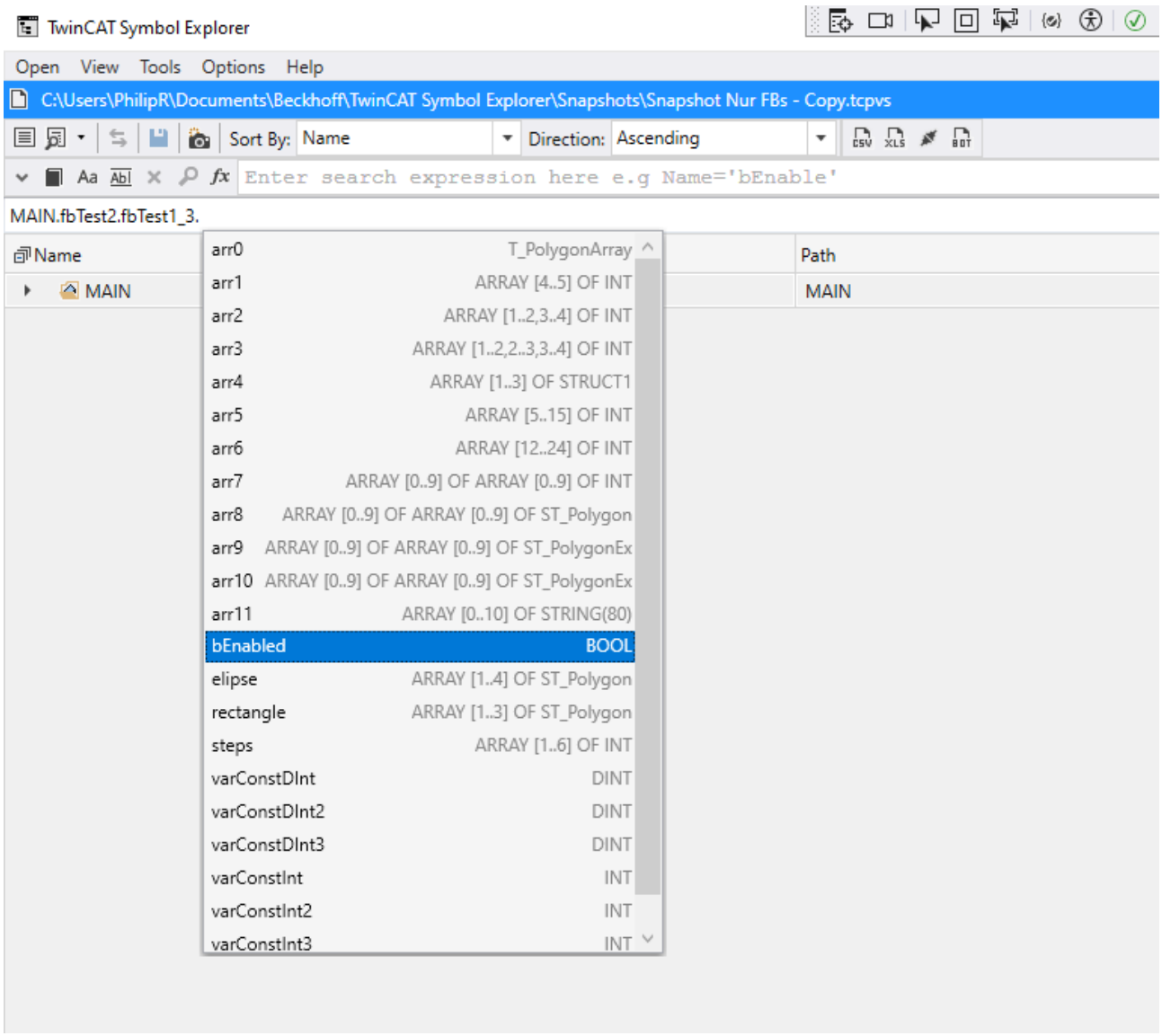
- Unterschiede von Symbol zu Symbol (Wert)
- Entfernte Symbole
- Hinzugefügte Symbole
- Fehlende Symbole
- Typen Unterschiede

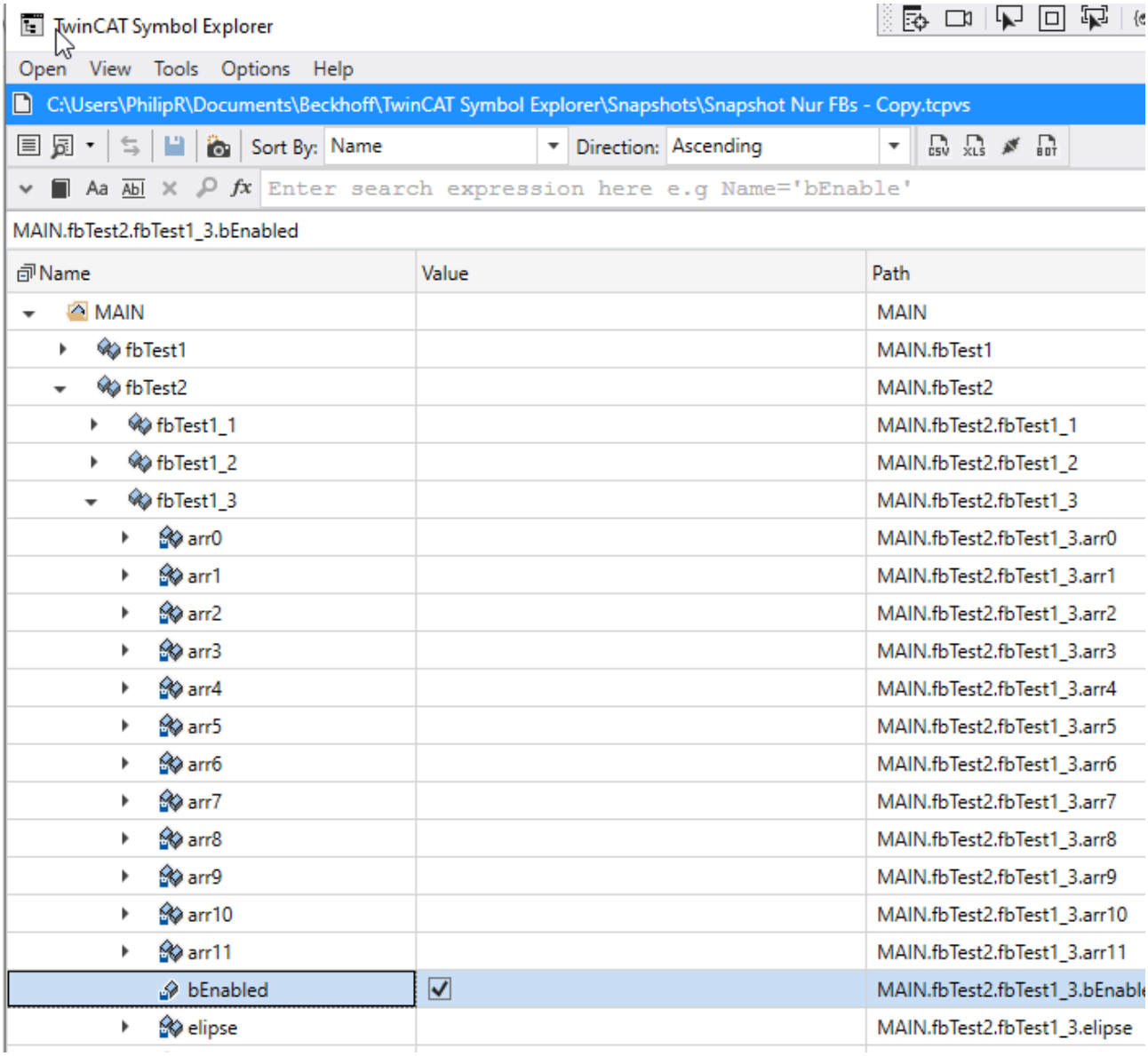
Toolbar

Symbol	Befehl	Beschreibung
	Show changes	Nur Werteunterschiede anzeigen.
	Previous change	Gehe zu voriger Änderung
	Next change	Gehe zu nächster Änderung
	First change	Gehe zur ersten Änderung
	Undo all	Änderungen zurücksetzen
	Last change	Gehe zur letzten Änderung
	Copy all changes to left	Kopiert alle Änderungen in die linke Liste
	Copy all changes to right	Kopiert alle Änderungen in die rechte Liste
	Copy change to left	Kopiert die ausgewählte Zeile in die linke Liste
	Copy change to right	Kopiert die ausgewählte Zeile in die rechte Liste
	Close window	Fenster schließen

7.2.3.9 Symbol Quick Navigation Bar




TwinCAT Symbol Explorer
Open View Tools Options Help
C:\Users\PhilipR\Documents\Beckhoff\TwinCAT Symbol Explorer\Snapshots\Snapshot Nur FBs - Copy.tcpvs
Sort By: Name
Direction: Ascending
Enter search expression here e.g Name='bEnable'
MAIN.fbTest2.fbTest1_3.
Name
Path
MAIN
MAIN
arr0 T_PolygonArray
arr1 ARRAY [4..5] OF INT
arr2 ARRAY [1..2,3..4] OF INT
arr3 ARRAY [1..2,2..3,3..4] OF INT
arr4 ARRAY [1..3] OF STRUCT1
arr5 ARRAY [5..15] OF INT
arr6 ARRAY [12..24] OF INT
arr7 ARRAY [0..9] OF ARRAY [0..9] OF INT
arr8 ARRAY [0..9] OF ARRAY [0..9] OF ST_Polygon
arr9 ARRAY [0..9] OF ARRAY [0..9] OF ST_PolygonEx
arr10 ARRAY [0..9] OF ARRAY [0..9] OF ST_PolygonEx
arr11 ARRAY [0..10] OF STRING(80)
bEnabled BOOL
elipse ARRAY [1..4] OF ST_Polygon
rectangle ARRAY [1..3] OF ST_Polygon
steps ARRAY [1..6] OF INT
varConstDInt DINT
varConstDInt2 DINT
varConstDInt3 DINT
varConstInt INT
varConstInt2 INT
varConstInt3 INT



7.2.3.10 Symbolliste

Das Kopieren und Einfügen im Symbol Explorer ist Listenzellen abhängig!
Das heißt, wird im Symbol Explorer eine Zelle ausgewählt und kopiert, so wird der Inhalt der Zelle kopiert.
Ausnahme bildet da nur die Zelle 'Name', wird diese ausgewählt und kopiert, wird die gesamte Symbolinformation kopiert.

Mittels Hotkeys und Kontextmenüs stellt die Symbolliste Funktionen, wie z. B. Werte bearbeiten oder Copy/Paste zur Verfügung. Des Weiteren bietet die Symbolliste eine Filterfunktion an. Mit ihr können zu bearbeitende Symbole gezielt gefiltert und hervorgehoben werden.

Name	Value	Type	Comment
BACnet_Globals_LIB			
MAIN			
varInt	12	INT	INT Variable
varInt2	13	INT	INT Variable
varInt3	0	INT	INT Variable
varDInt	0	DINT	DINT Variable
varDInt2	450	DINT	DINT Variable
varDInt3	0	DINT	DINT Variable
fbArray		ARRAY [0..10] OF FB_Test_0	
fbTest0		FB_Test_0	

Hotkeys

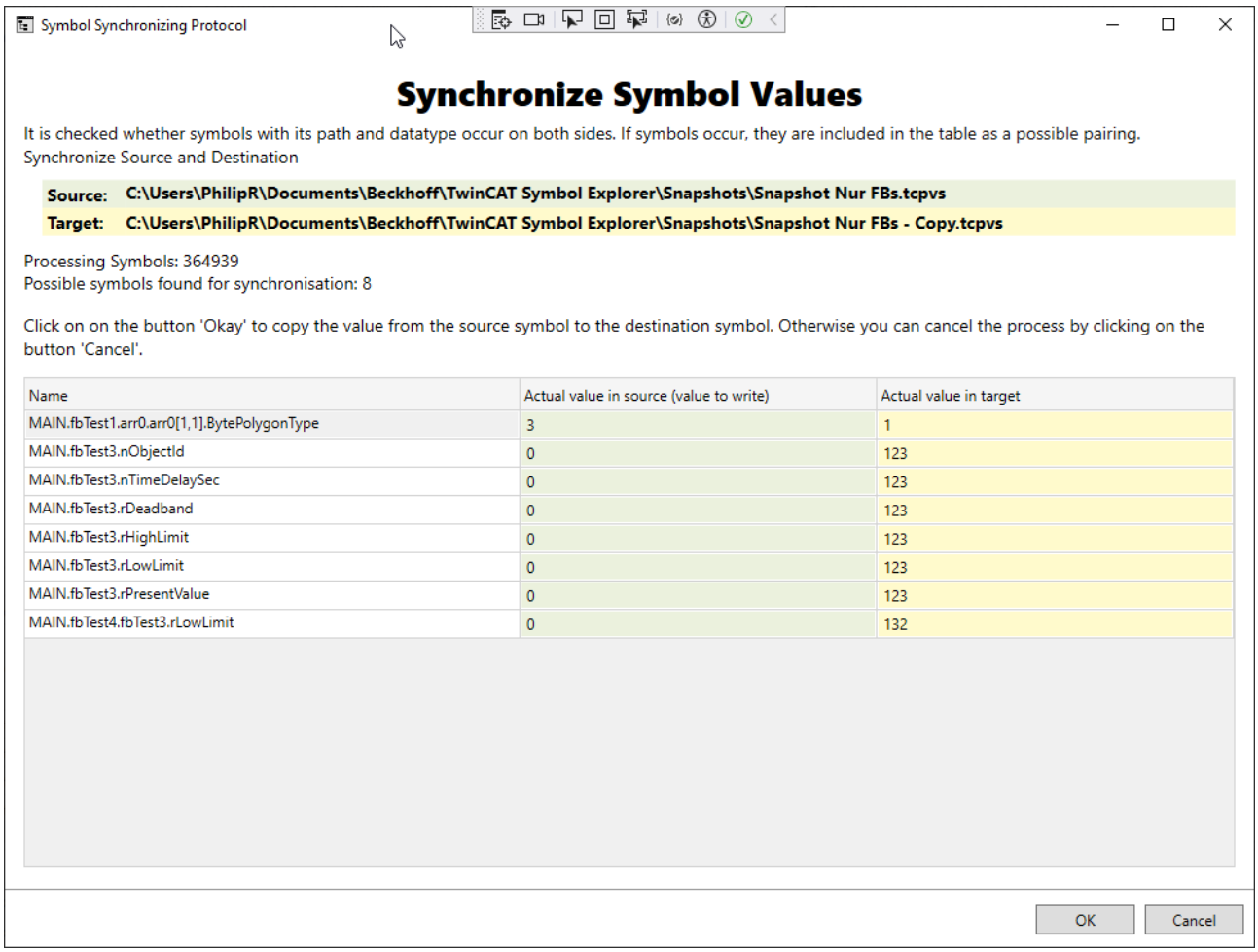
Tastenkombination	Beschreibung
Ctrl + links klick 	Ausklappen oder Einklappen des Symbols und seiner Subsymbole.
Ctrl + links klick	Selektive Multiauswahl auf eine Zeile.
Ctrl + C	Ausgewähltes Symbol (und alle Subsymbole) kopieren.
Ctrl + V	Kopierte Symbolwerte in das gewählte Symbol (und die Subsymbole) einfügen.
Ctrl + Shift + C	Kopiert den Inhalt der ausgewählten Zelle.
Ctrl + R	Zurücksetzen des Symbolwertes auf den vorigen Wert.
Shift + Links klick	Multiauswahl
Doppelklick links	Startet das Editieren eines Symbols.
F2	Wenn zuvor ein Symbol gewählt wurde, dann startet das Editieren.

Kontextmenü

Auswahl	Beschreibung
Read from PLC	Liest am gewählten Symbol den aktuellen Wert aus der SPS.
Copy	Ausgewähltes Symbol (und alle Subsysteme) kopieren.
Paste	Kopierte Symbolwerte in das gewählte Symbol (und die Subsymbole) einfügen.
Search for	Vorschläge für eine Suche
Reset	Zurücksetzen des Symbolwertes auf den vorigen Wert.
Add instance watch	Symbol dem Instance Watch hinzufügen.
Details	Symboldetails aufrufen

7.2.3.11 Synchronize Symbol Fenster

Mit diesem Fenster können verschiedene Targets (Online <-> Online oder Snapshort <-> Online) miteinander verglichen oder ein Snapshot (Rezept oder Werte Snapshot) zurück auf ein Target gespielt werden. Dabei werden die Symbole der Targets miteinander verglichen. Existiert ein Symbol auf beiden Seiten, wird dessen Wert verglichen. Existiert an der Stelle ein Unterschied, wird das Symbol als Unterschied in der Liste (Siehe unten im Bild) vermerkt. Am Ende können die unterschiedlichen Symbole zwischen den Targets synchronisiert werden.



7.2.4 Getting started

Dieses Kapitel beschreibt schrittweise, wie Sie mit dem Symbol Explorer arbeiten und es soll einen Überblick über seine Funktionen geben.

Verbinden mit einer Steuerung

Der Symbol Explorer kommuniziert mit einer Steuerung über die Kommunikations-Schnittstelle ADS. Damit die Kommunikation zu einer Steuerung erfolgreich aufgebaut werden kann, müssen folgende Bedingungen erfüllt sein:

- Die Steuerung ist über das Netzwerk erreichbar.
- TwinCAT befindet sich im Run Mode.
- Eine AMS Route zu der Steuerung ist eingerichtet.
- Eine SPS-Laufzeit ist aktiviert und gestartet.

Sind die Kriterien erfüllt, können eine Verbindung aufgebaut und die Symbole ausgelesen werden. Dabei werden nur Symbole ausgelesen, die als persistent deklariert wurden und Symbole, die persistent deklarierte Symbole beinhalten.

Verbindung zu einer Steuerung starten

1. Starten Sie den Symbol Explorer und klicken auf der Startseite den Button **Connect.**
2. Im Dialog **Choose Route** wählen Sie die **Ams Route** die zu der Steuerung gehört, mit der Sie sich verbinden wollen.

⇨ Konnten Sie sich erfolgreich mit der Steuerung verbinden, öffnet sich das Online-Fenster, dass Ihnen die ausgelesenen Symbole in einer Liste anzeigt.

Snapshot erstellen

1. Starten Sie den **Symbol Explorer** und verbinden sich mit einer Steuerung.
2. Klicken Sie im Online-Fenster auf den **Take Snapshot** Button, um einen Snapshot zu erstellen.
3. Sie werden daraufhin aufgefordert einen Speicherort für die Snapshot-Datei auszuwählen. Wählen Sie einen Ordner aus und bestätigen mit **Ok.**

⇨ Sämtliche Werte der Symbole in der Liste werden mit der SPS synchronisiert und in die Datei geschrieben.

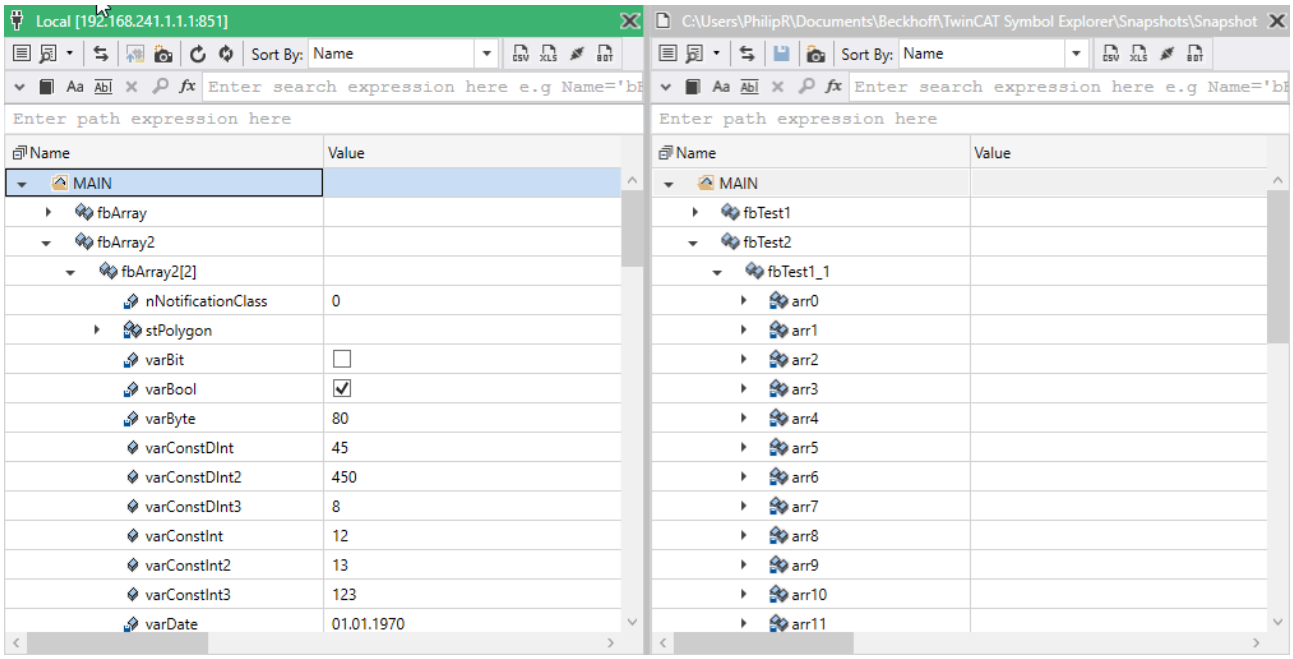
Snapshot laden

1. Öffnen Sie den **Symbol Explorer** und klicken auf der Startseite den Button **Snapshot**.
2. Sie werden daraufhin aufgefordert, eine Snapshot-Datei auszuwählen. Wählen Sie eine Snapshot-Datei aus, die geladen werden soll und bestätigen mit **Ok.**

⇨ Anschließend öffnet sich das Snapshot Fenster, dass Ihnen die geladenen Symbole in einer Liste anzeigt.

Symbolleisten parallel betrachten und bearbeiten

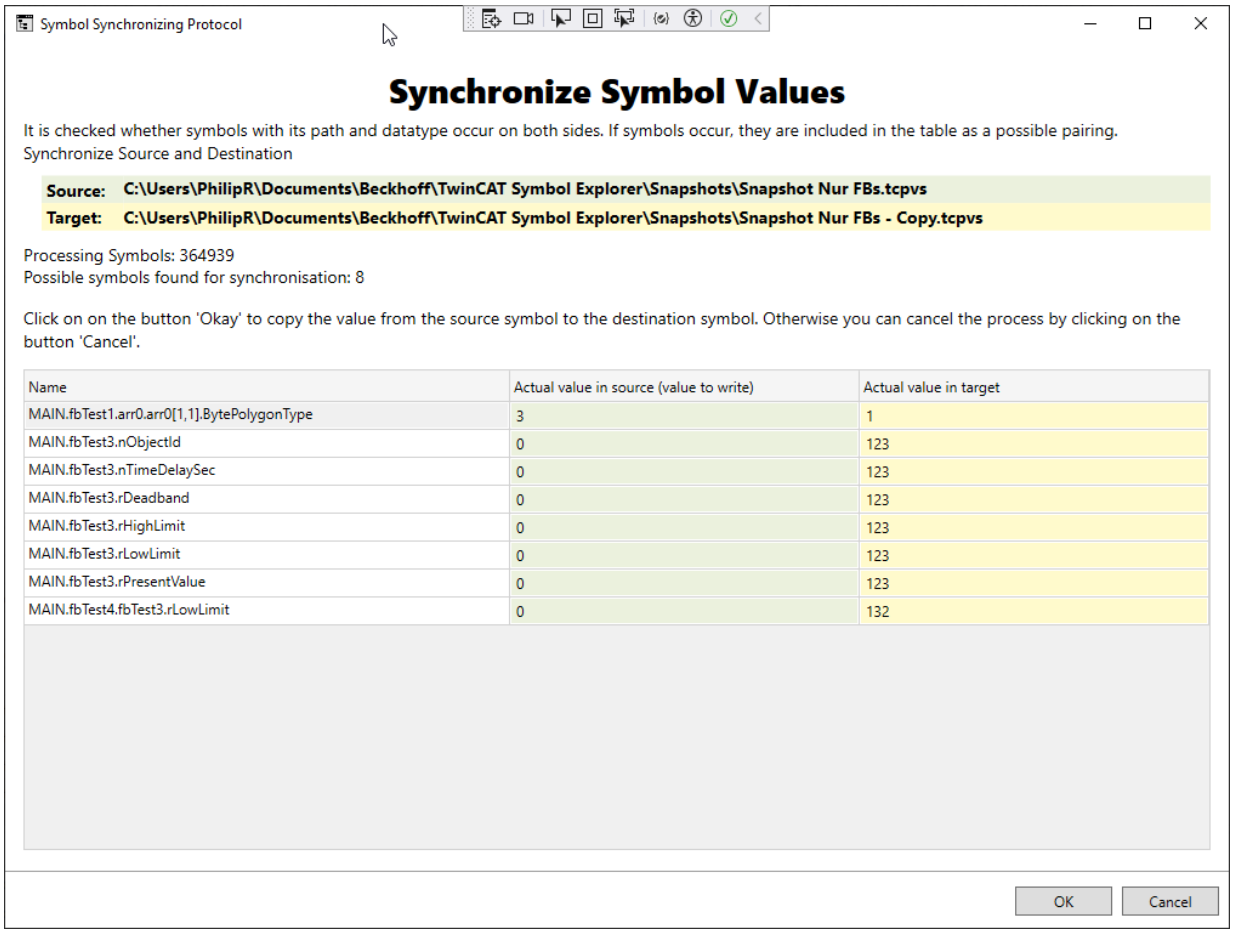
Der Symbol Explorer kann zwei Symbollisten parallel darstellen. Somit ist es z. B. möglich, die aktuellen Symbole (Online) und Symbole aus einem Snapshot nebeneinander zu stellen und zwischen diesen beiden Listen Symbolwerte zu bearbeiten. Verbinden Sie sich einfach mit einer Steuerung und öffnen anschließend einen Snapshot. Eine andere Reihenfolge zum Öffnen/Verbinden, ist ebenso möglich.



Snapshot auf eine Steuerung aufspielen

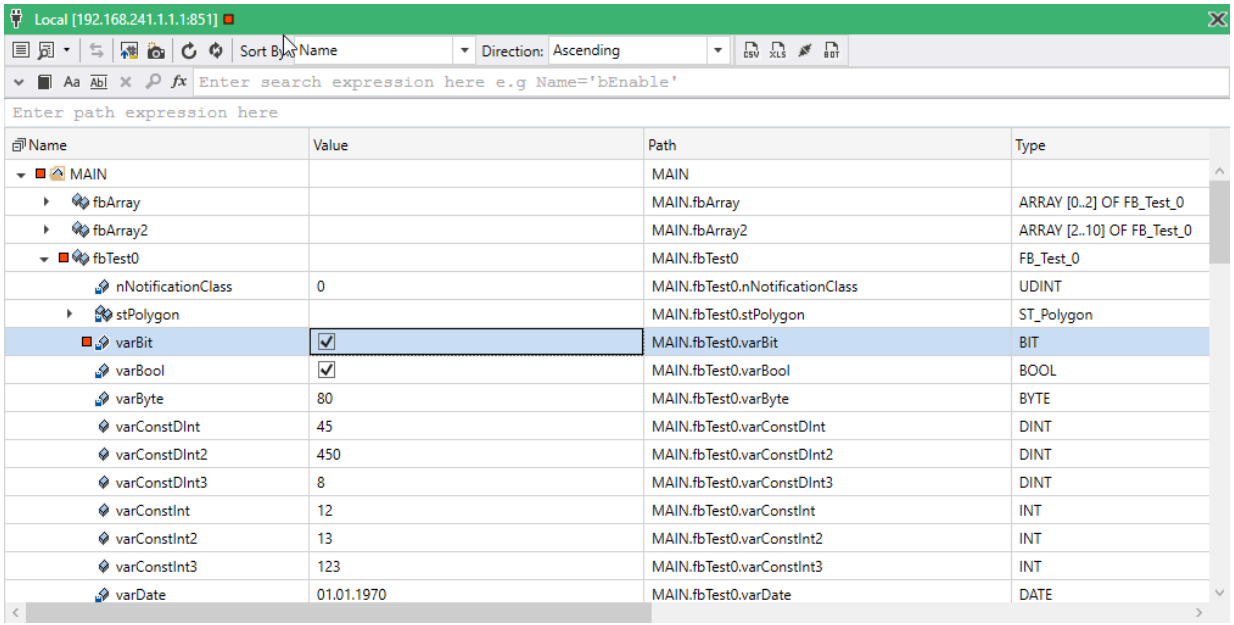
1. Öffnen und verbinden Sie den Symbol Explorer mit einer Steuerung.
2. Öffnen Sie anschließend einen Snapshot. Sie sehen nun das Online und Snapshot Fenster nebeneinander.

3. Drücken Sie nun im Snapshot Fenster den Button **Copy Symbols**. Es öffnet sich daraufhin der **Symbol Transfer Protocol** Dialog:



⇨ Der Dialog informiert Sie detailliert über den Kopiervorgang.

4. Sind die Symbolwerte korrekt, bestätigen Sie den Kopiervorgang mit einem Klick auf **Ok**, andernfalls wählen Sie **Cancel**. Haben Sie den Kopiervorgang bestätigt, werden die Symbolwerte aus dem Snapshot in die Online-Symbolliste kopiert.



⇨ Ist der Kopiervorgang abgeschlossen, werden die Symbole mit geänderten Werten mittels eines roten Viereckes vor den Symbolnamen hervorgehoben.

5. Sie können nun die Änderungen aus dem Online-Fenster in die Steuerung laden. Drücken Sie dazu den Button **Upload Symbol** in der Toolbar.

Symbole vergleichen und Unterschiede abgleichen

Mit dem Symbol Explorer ist es möglich, zwei Listen von Symbolen zu vergleichen. Sie können somit Online-Symbole mit Symbolen aus einem Snapshot vergleichen und auf Unterschiede prüfen.

Starten eines Vergleichs

1. Verbinden Sie sich mit einer Steuerung oder laden Sie einen Snapshot.
2. Wiederholen Sie den Vorgang, sodass zwei Symbollisten nebeneinanderstehen.
3. Drücken Sie anschließend den Hotkey **[F8]** oder im Menü **Tools > Compare Symbols**.

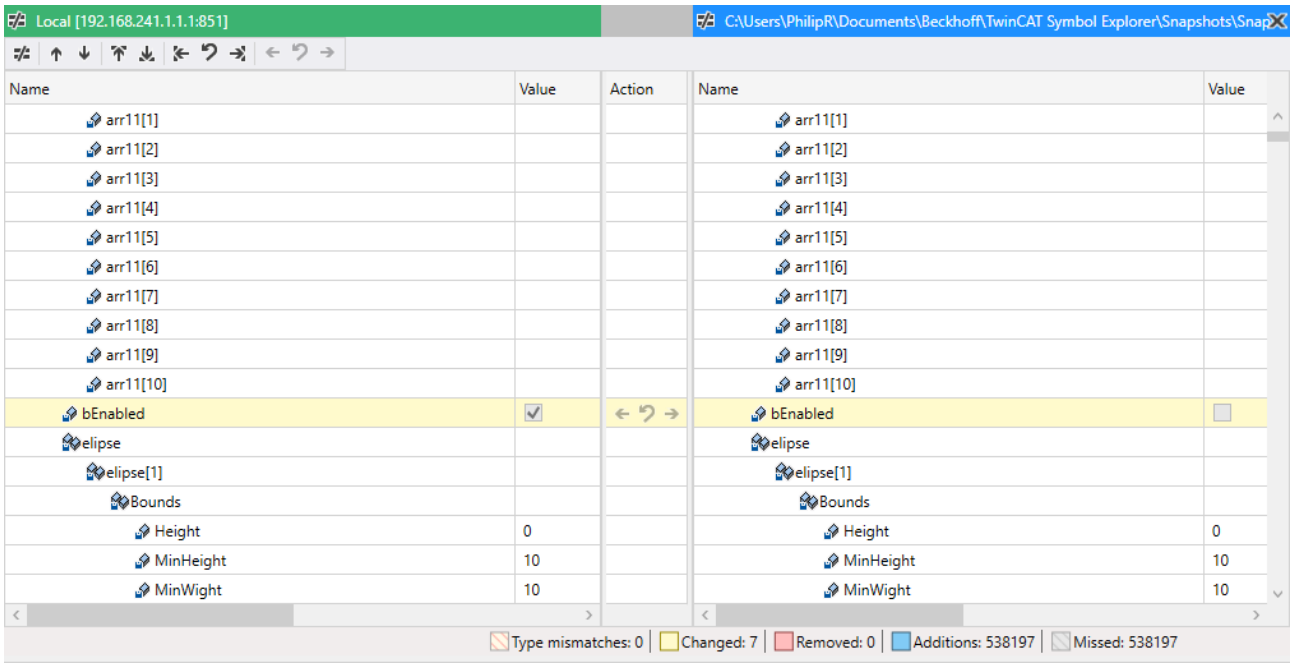
⇨ Anschließend öffnet sich das Vergleich-Fenster.

Unterschiede abgleichen

Beim Abgleichen werden Unterschiede von einem Symbol zum anderen kopiert. Entweder von links nach rechts oder von rechts nach links. Dieser Vorgang unterscheidet sich vom direkten Bearbeiten eines Symbols über das Online oder Snapshot Fenster.

Funktionen zum Abgleichen

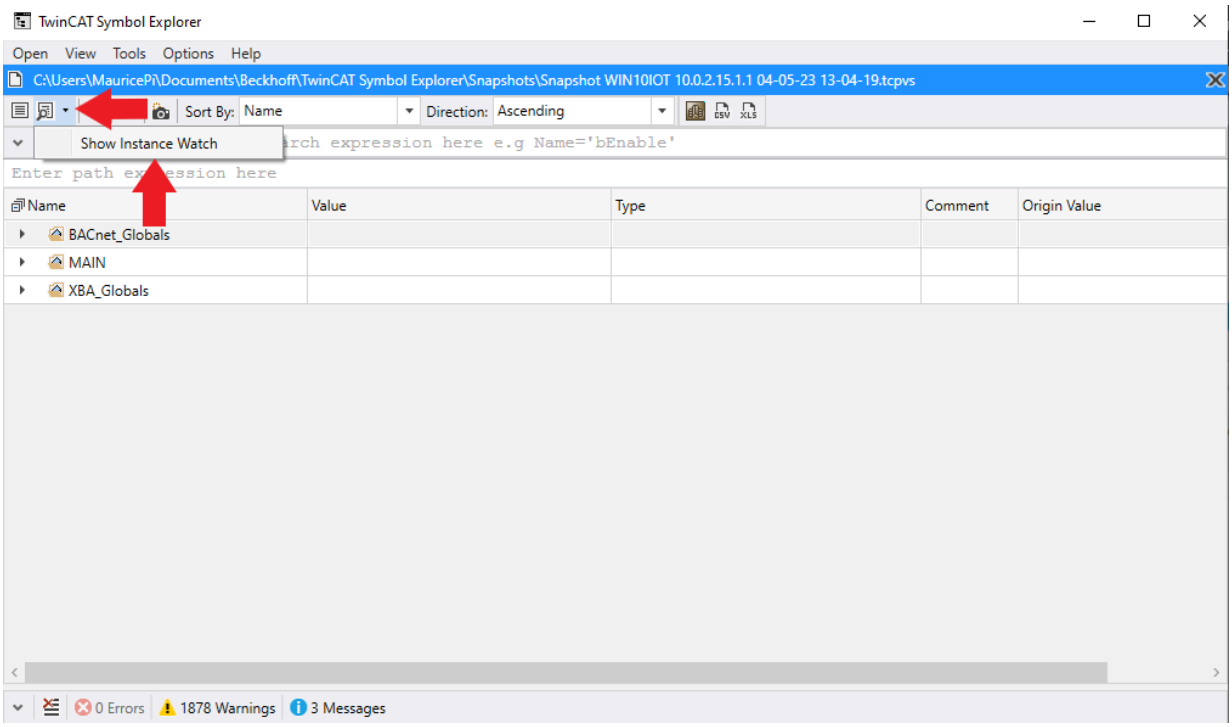
Die Abgleichfunktionen können implizit Unterschiede basierend auf einer Zeilenauswahl angewendet werden. Wenn Sie beispielsweise auf eine Zeile mit Unterschied klicken und anschließend auf die mittlere Symbolleistenschaltfläche der Zeile, den Pfeil nach rechts klicken, wird der gesamte Symbolwert von links in das Symbol rechts kopiert. Dadurch können viele kleine Unterschiede leicht zusammengeführt werden.



7.2.4.1 Instance Watch Template öffnen

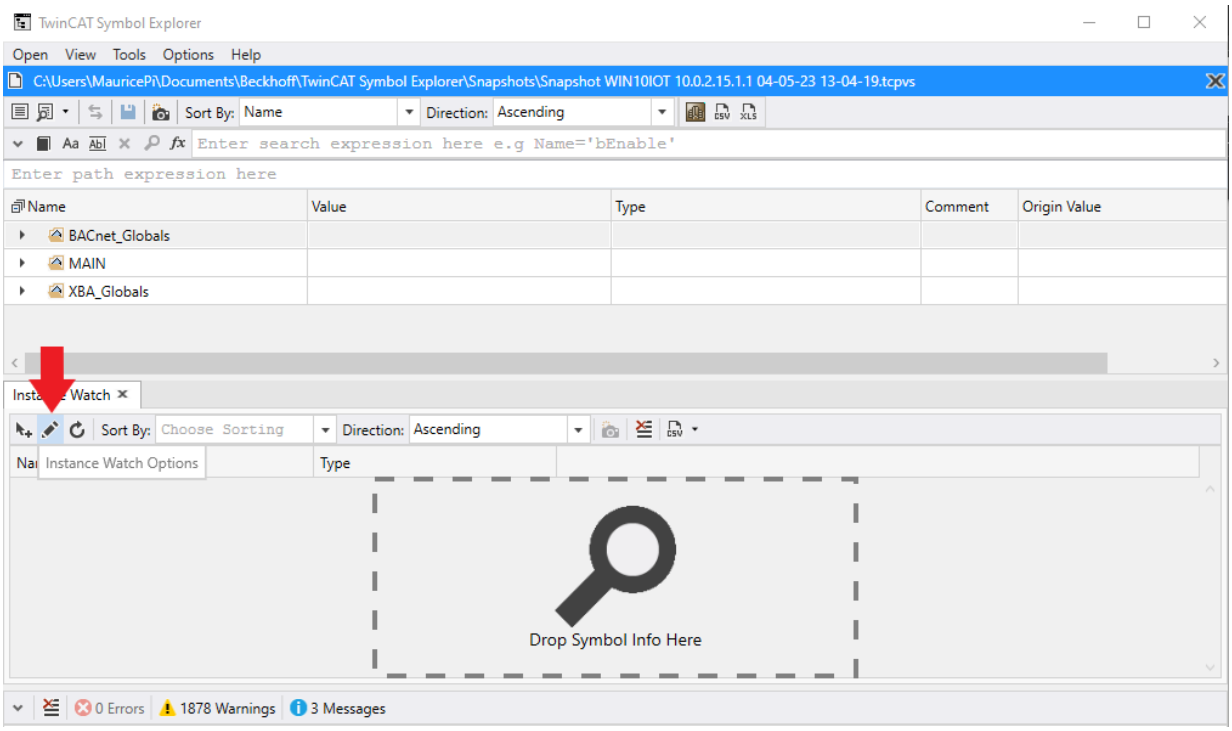
✓ Um ein Instance Watch Template zu laden, muss zuerst der Symbol Explorer gestartet werden.

1. Öffnen Sie die Instance Watch

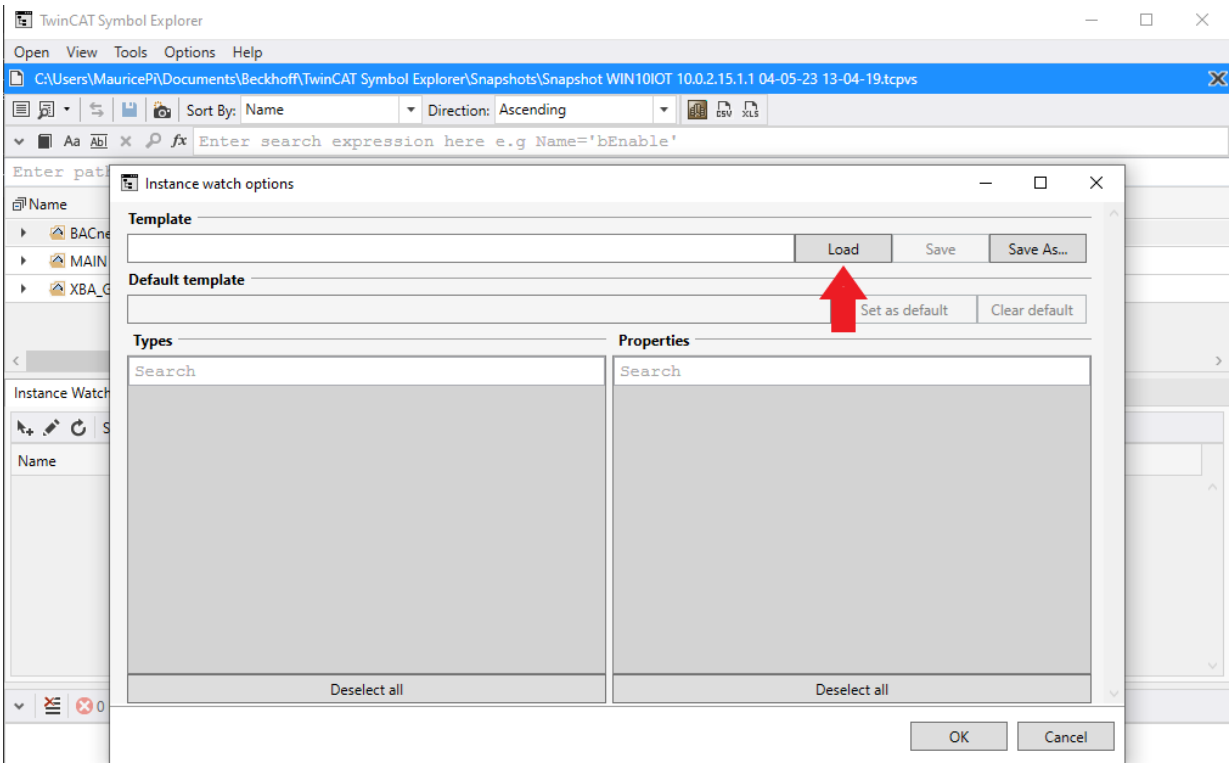


⇨ Sofern kein Standardtemplate definiert wurde, ist die Instance Watch leer.

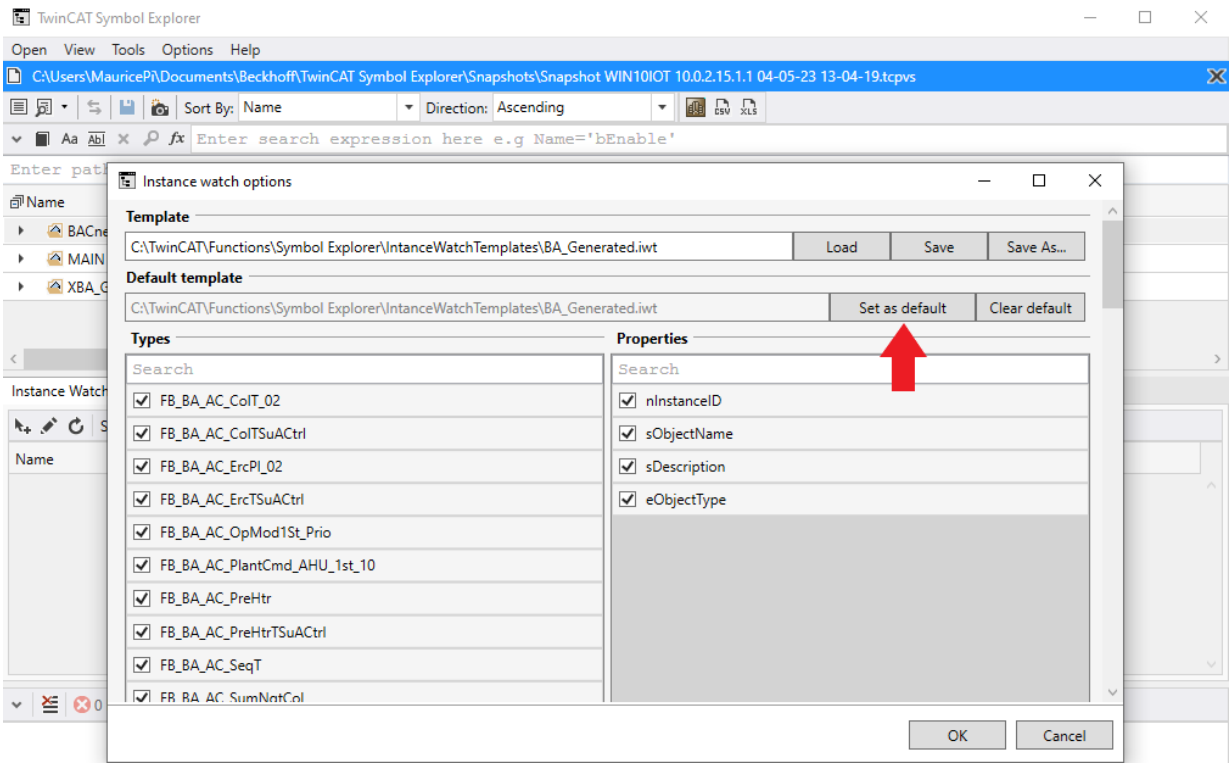
2. Um ein Template zu laden, drücken Sie den Button **Instance Watch Options (Stift).**



3. Nun öffnet sich ein Fenster. Laden Sie über den Button **Load** ein Template. Hier öffnet sich ein File-Choose-Dialog, in dem Sie die *.iwt-Datei auswählen müssen.



4. Soll bei jedem Starten des Instance Watches dieses Template geladen werden, stellen Sie dies mit dem Drücken des Buttons **Set as default** ein. Es sollte der Pfad der *iwt-Datei links, neben dem Button unter **Default template** erscheinen.
Um diese Änderung rückgängig zu machen, klicken Sie auf den Button **Clear default** .



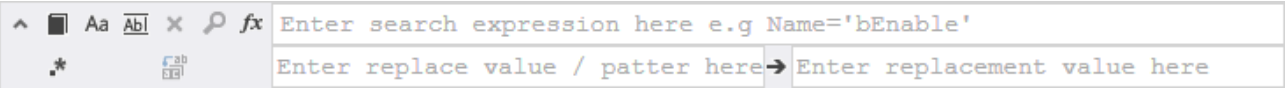
5. Bestätigen die den **Instance watch options**-Dialog mit **OK.**

⇨ Die Instance Watch wird mit dem Template geladen.

7.2.5 Filtern, Suchen und Ersetzen

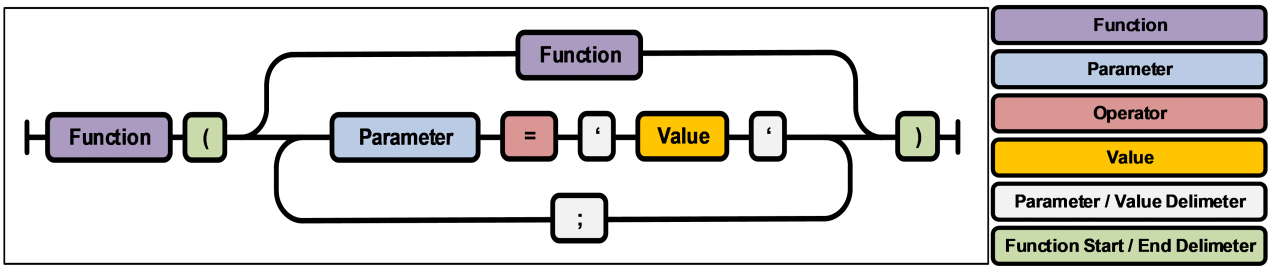
Durch individuelles Filtern der Symbole ist es möglich, die gewünschten Symbole schneller zu finden. Es kann nach Name, Typ oder Kommentar eines Symbols gefiltert werden.

Suchen und Ersetzen Toolbar



Symbol	Befehl	Beschreibung
	Recent search terms	Liste der häufigen Suchen
	Match case	Es wird auf Groß- / Kleinschreibung geachtet
	Match whole word	Es wird auf die exakte Eingabe geachtet
	Close search	Schließt die Suche
	Enter search example	Suchausdruck eingeben
	Start search	Startet die Suche
	Use regular expressions	Schaltet die Suche zur Verwendung regulärer Ausdrücke frei
	Replace all	Alles ersetzen
	Search Pattern	Suchmuster Vorlagen
	Search for value to replace	Suche nach zu ersetzendem Wert
	Replacement value	Ersatzwert

7.2.5.1 Filter-Formel-Übersicht



Eine Funktion ist eine ungeordnete Menge von Funktionen, Parametern, Operatoren und Wertepaaren. Eine Funktion beginnt mit und endet mit Klammern (). Jede Funktion, Parameter, Operator und Wertepaar werden von einem Semikolon gefolgt.

AND-Funktion

Die UND-Funktion dient zum Filtern von Symbolen, deren Bedingungen mit einem Wahr erfüllt sind. Nachfolgend zwei Beispiele für die Verwendung von der UND-Funktion und in Verbindung mit der ODER-Funktion.

```
AND ( Type  = BOOL;
      Name  = 'bEnabled';
      Value = True )
```

Die Funktion filtert nach Symbolen mit dem Typ BOOL, deren Name *bEnabled* und deren Wert TRUE ist.

```
AND ( Name  = nCounter;
OR  ( Value = 10;
      Value = 50 ))
```

Die Funktion filtert nach Symbolen deren Name *nCounter* ist und deren Wert ‚10' oder ‚50' ist.

OR-Funktion

Die ODER-Funktion dient zum Filtern von Symbolen, bei denen eine Bedingung mit einem Wahr erfüllt ist. Nachfolgend zwei Beispiele für die Verwendung von der ODER-Funktion und in Verbindung mit der UND-Funktion.

```
OR ( Value = 150;
     Value = -150 )
```

Die Funktion filtert nach Symbolen deren Wert ‚150' oder ‚-150' ist.

```
AND ( Name = 'nCounter';
      OR ( Value = 10;
           Value = 50 ))
```

Die Funktion filtert nach Symbolen deren Name ‚*nCounter*' ist und deren Wert ‚10' oder ‚50' ist.

PARENT-Funktion

Die PARENT-Funktion dient zum Filtern von Symbolen deren direktes Eltern-Symbol (Übergeordnetes Symbol) einer gleichen Bedingung unterliegen. Die in der PARENT-Funktion angegebenen Bedingungen werden Logisch-UND ausgewertet.



Die PARENT-Funktion kann nur in einer AND-Funktion oder einer ODER-Funktion verwendet werden.

Hier ein allgemeines Beispiel für die Verwendung der PARENT-Funktion:

```
AND (    Name = 'sObjectName';
PARENT ( Name = 'Plant';
         Value = TRUE ))
```

Die Funktion filtert nach Symbolen mit dem Namen ‚*sSobjectName*' und nach deren Eltern-Symbol mit dem Namen ‚*Plant*'. Weiterhin muss der Wert dieses Symbols TRUE sein.

ANCESTOR-Funktion

Die ANCESTOR-Funktion dient zum Filtern von Symbolen, dessen Vorfahren in der Elternkette von Symbolen einer Bedingung unterliegen, z. B. ein spezieller Name oder Typ. Die in der ANCESTOR-Funktion angegebene Bedingung wird Logisch-AND ausgewertet.



Die ANCESTOR-Funktion kann nur in einer AND- Funktion oder einer ODER-Funktion verwendet werden.

Hier einige allgemeine Beispiele für die Verwendung der ANCESTOR-Funktion:

```
AND (      Name = 'sObjectName';
ANCESTOR ( Type = 'FB_BAC_AI'))
```

Die Funktion filtert nach Symbolen mit dem Namen ‚*sObjectName*'. Die Vorfahren dieser Symbole kommen aus der Elternkette von Symbolen mit dem Typ ‚*FB_BAC_AI*'.

```
AND (      Name = 'sObjectName';
ANCESTOR ( Name = 'Plant01'))
```

Die Funktion filtert nach Symbolen mit dem Namen ‚*sObjectName*'. Innerhalb der Elternkette dieser Symbole trägt einer der Vorfahren den Namen ‚*Plant01*'.

CHILD-Funktion

Die CHILD-Funktion dient zum Filtern von Symbolen deren Kinder-Symbole einer Bedingung unterliegen. Die in der CHILD-Funktion angegebenen Bedingungen werden logisch UND ausgewertet.



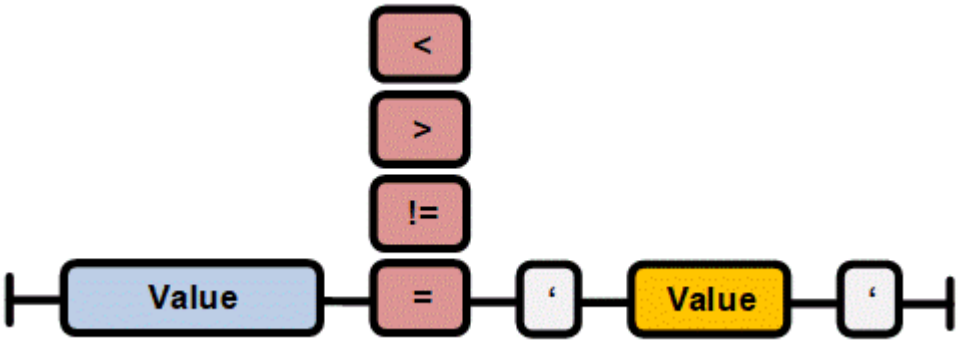
Die CHILD-Funktion kann nur in einer AND- Funktion oder einer ODER-Funktion verwendet werden.

Hier ein allgemeines Beispiel für die Verwendung der CHILD-Funktion:

```
AND (   Name = 'Plant';
Child ( Name = 'sObjectName';
        Value = 'P001' )
```

Value Parameter

Die Funktion sucht nach Symbolen deren Wert eine Bedingung mit einem bestimmten Wert erfüllt.



Beispiele:

```
Value = '100'
```

Suche nach Symbolen deren Wert ‚100‘ ist.

```
Value > '100'
```

Suche nach Symbolen deren Wert größer ‚100‘ ist.

```
Value < '100'
```

Suche nach Symbolen deren Wert kleiner ‚100‘ ist.

```
Value != '100'
```

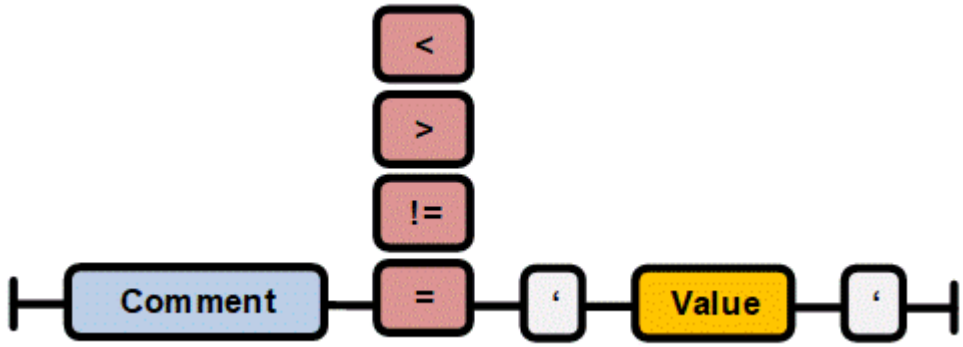
Suche nach Symbolen deren Wert ungleich ‚100‘ ist.

```
Value = ''
```

Suche nach Symbolen deren Wert leer , ‘ ist.

Comment Parameter

Die Funktion sucht nach Symbolen deren Wert eine Bedingung mit einem bestimmten Kommentar erfüllt.



Beispiele:

```
Comment = 'Signal to detect'
```

Suche nach Symbolen deren Kommentar ‚Signal to detect‘ ist.

```
Comment != 'Signal to detect'
```

Suche nach Symbolen deren Kommentar nicht ‚Signal to detect‘ ist.

```
Comment >'Signal'
```

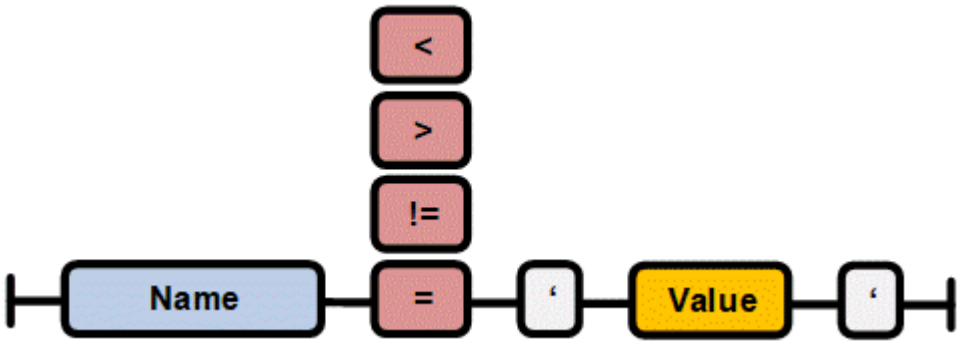
Suche nach Symbolen deren Kommentar mit ‚Signal‘ beginnt.

```
Comment < 'detect'
```

Suche nach Symbolen deren Kommentar mit ‚detect‘ endet.

Name Parameter

Die Funktion sucht nach Symbolen deren Wert eine Bedingung mit einem bestimmten Namen erfüllt.



Beispiele:

```
Name = 'bEnable'
```

Suche nach Symbolen deren Name ‚bEnable‘ ist.

```
Name != 'bEnable'
```

Suche nach Symbolen deren Name nicht ‚bEnable‘ ist.

```
Name > 'bEn'
```

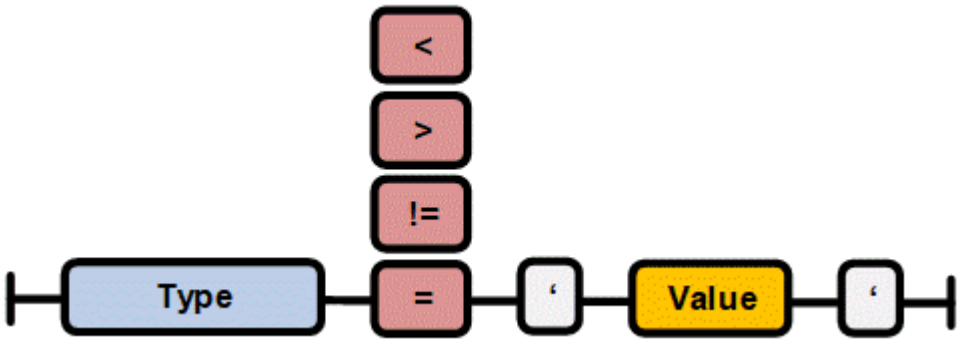
Suche nach Symbolen deren Name mit ‚bEn‘ beginnt.

```
Name < 'le'
```

Suche nach Symbolen deren Name mit ‚le‘ endet.

Type Parameter

Die Funktion sucht nach Symbolen deren Wert eine Bedingung von einem bestimmten Typ entspricht.



Beispiele:

```
Type = 'FB_BACnet_Pump'
```

Suche nach Symbolen deren Typ ‚FB_BACnet_Pump‘ ist.

```
Type != 'FB_BACnet_Pump'
```

Suche nach Symbolen deren Name nicht ‚FB_BACnet_Pump ‘ ist.

```
Type > 'FB_BACnet'
```

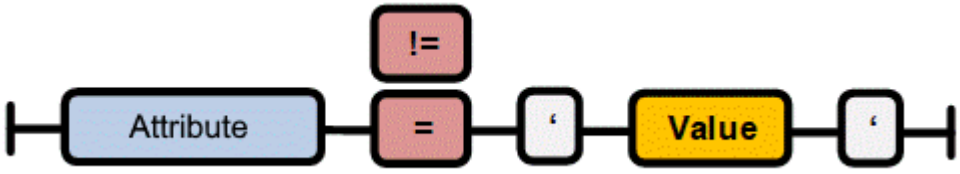
Suche nach Symbolen deren Name mit ‚FB_BACnet‘ beginnt.

```
Type < 'BACnet_Pump'
```

Suche nach Symbolen deren Name mit ‚BACnet_Pump‘ endet.

Attribut Parameter

Die Funktion sucht nach Symbolen, die ein Attribut haben dessen Name eine Bedingung mit einem bestimmten Wert erfüllt.



Beispiele:

```
Attribute = 'IsPersistent'
```

Suche nach Symbolen, die ein Attribut mit dem Namen ‚IsPersistent‘ haben.

```
Attribute != 'IsPersistent'
```

Suche nach Symbolen, die nicht ein Attribut mit dem Namen ‚IsPersistent‘ haben.

7.2.6 Command Line Interface

Mit dem Kommandozeilen-Interface (CLI) kann der Symbol Explorer über die Kommandozeile aufgerufen werden.

Folge Kommandos stehen zur Verfügung:

```
--SnapShotFromPlc     Take a snapshot from PLC target.
--SyncPlcToSnapShot   Synchronize PLC target with a Snapshot.
--SyncFileToPlc       Synchronize Snapshot with a PLC target.
```

--SnapShotFromPlc

Snapshots von einer PLC erzeugen.

Option	Beschreibung
<NetId>	AMS Net Id des Targets
<Port>	AMS Port des Targets
<SnapShotPath>	SnapShot Dateipfad

```
  --SnapShotFromPlc <NetId> <Port> <SnapShotPath>
```

--SyncPlcToSnapShot

PLC-Symbolwerte mit einem Snapshot synchronisieren.
Dabei werden die aktuellen PLC-Symbolwerte in einen Snapshot übertragen.

Option	Beschreibung
<NetId>	AMS Net Id des Targets
<Port>	AMS Port des Targets
<SnapShotPath>	SnapShot Dateipfad

```
  --SyncPlcToSnapShot <NetId> <Port> <SnapShotPath>
```

--SyncSnapShotToPlc

Snapshot-Symbolwerte mit der PLC synchronisieren.
Dabei werden die Snapshot-Symbolwerte in die PLC übertragen.

Option	Beschreibung
<NetId>	AMS Net Id des Targets
<Port>	AMS Port des Targets
<SnapShotPath>	SnapShot Dateipfad

```
  --SyncSnapShotToPlc <NetId> <Port> <SnapShotPath>
```

Symbol Entry Point

Für die CLI wie auch über die UI wird ein Symbol Entry Point für den Zugriff auf Symbole eines Targets vorausgesetzt. Der Symbol Entry Point kann für die CLI in Form einer Konfigurationsdatei angegeben werden. Zum Erstellen der Datei muss einmal mit der UI auf das Target zugegriffen werden, dabei wird der Symbol Entry Point Dialog gestartet. Im Dialog können Sie wie für die UI wählen, welche Symbole von Interesse sind.

Aufruf Beispiel

```
...\TwinCAT\Functions\SymbolExplorer.exe --SnapShotFromPlc <192.168.10.1.1> <851> <"c:
\Temp\Snapshots\File.tcpvs>
```

Beispiel zum Starten des Symbol Explorer aus der PLC

TwinCAT 3 | PLC Library: Tc2_Utilities -> NT_StartProcess Beispiel

NT_StartProcess

Beispiel zur Konfiguration der Eingangsvariablen.

PATHSTR: "...\TwinCAT\Functions\SymbolExplorer.exe"

DIRNAME: "...\TwinCAT\Functions"

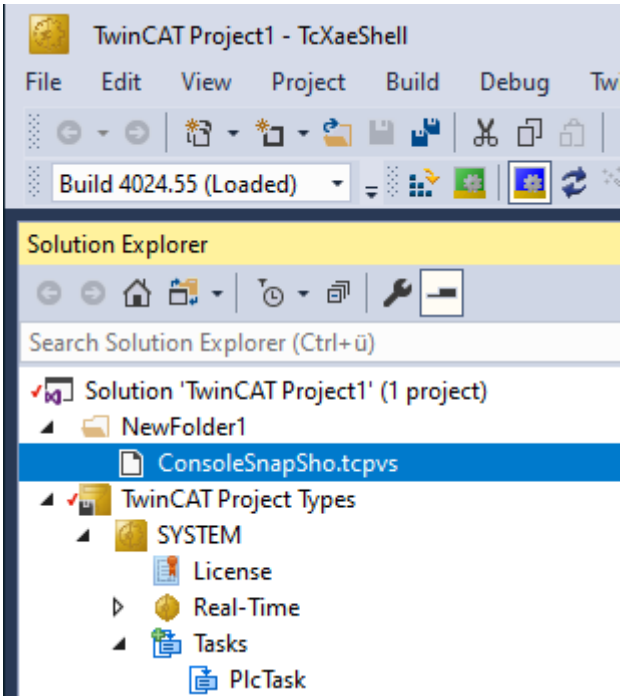
COMNDLINE: "--SnapShotFromPlc 192.168.10.1.1 851 "c:\Temp\Snapshots\File.tcpvs".

7.2.7 Workflows

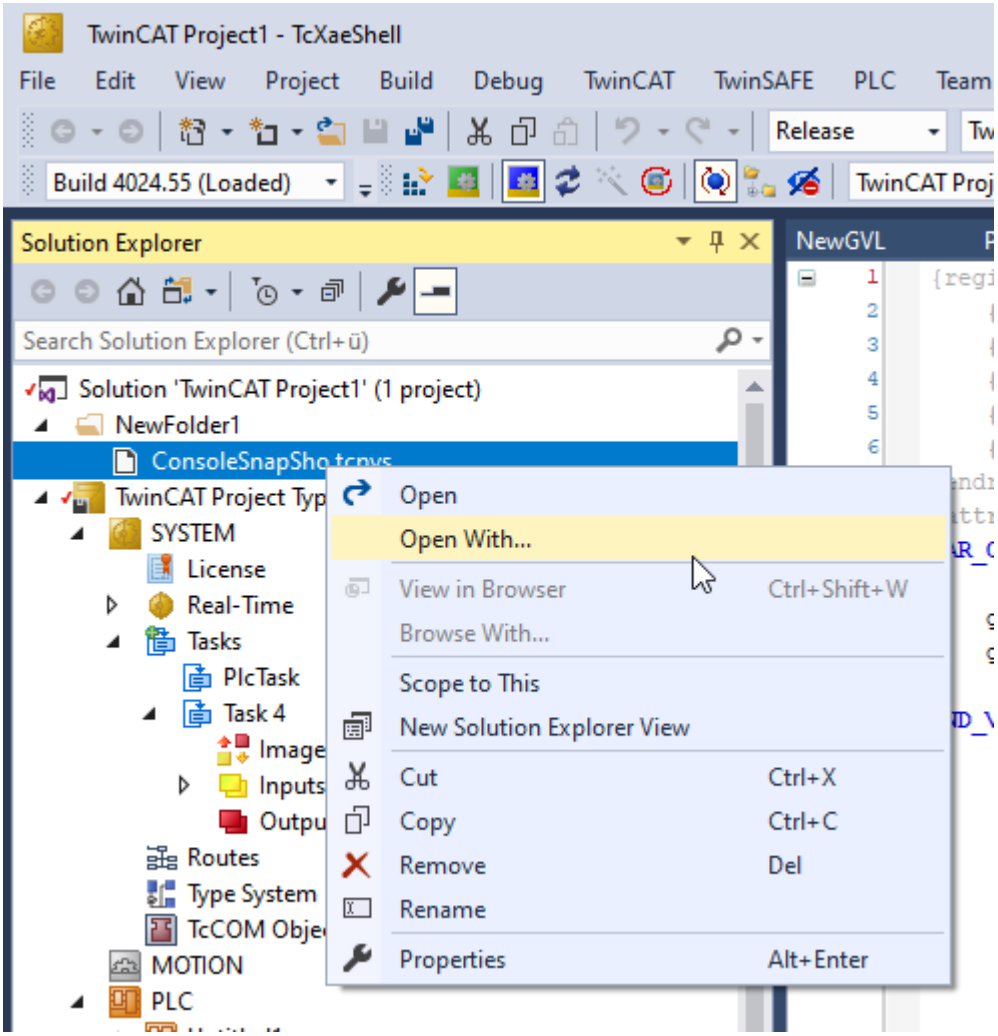
Im Folgenden sind einige Beispiele zum möglichen Umgang mit dem Symbol Explorer dargestellt.

7.2.7.1 Snapshots aus dem Visual Studio und der XAE Shell öffnen

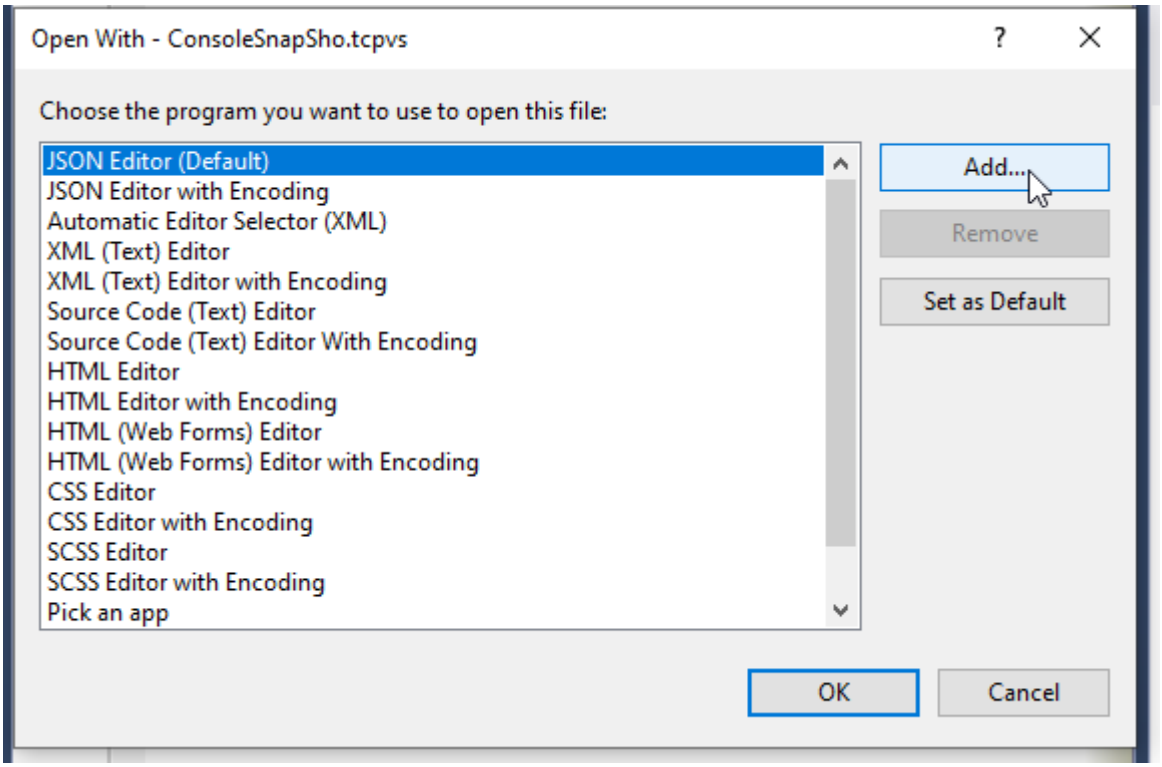
1. Fügen Sie den Symbol Explorer Snapshot in das Visual Studio / XAE hinzu.



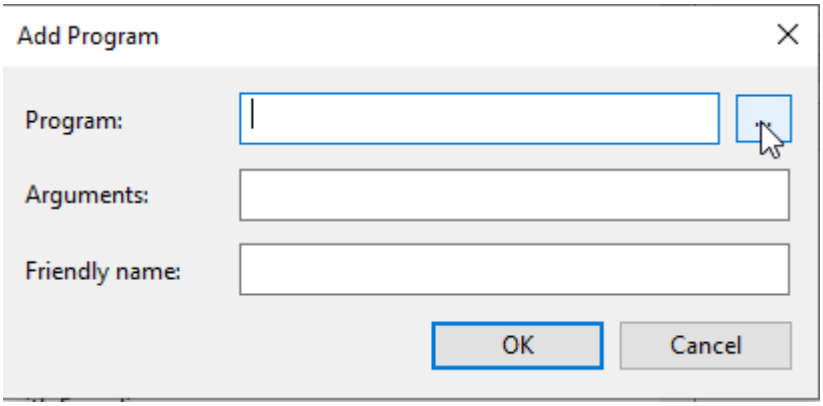
2. Klicken Sie mit der rechten Maustaste auf den Snapshot und wählen Sie im Kontextmenü **Open With…**



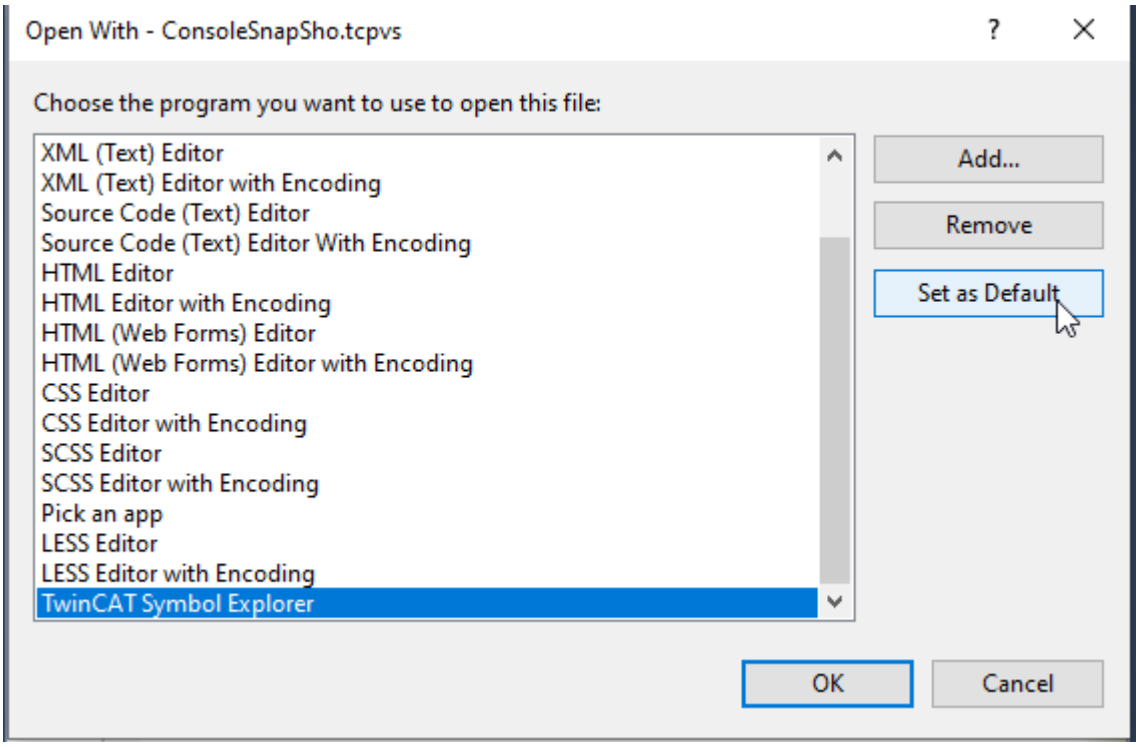
3. Klicken Sie auf den Button **Add**.



4. Klicken Sie auf den Button mit den Punkten.
   Suchen Sie die Datei *SymbolExplorer.exe* und bestätigen dies mit **OK**.



5. Wählen Sie **TwinCAT Symbol Explorer** aus der Liste an Programmen und klicken Sie auf den Button **Set as Default**. Bestätigen Sie den Dialog mit **OK**.



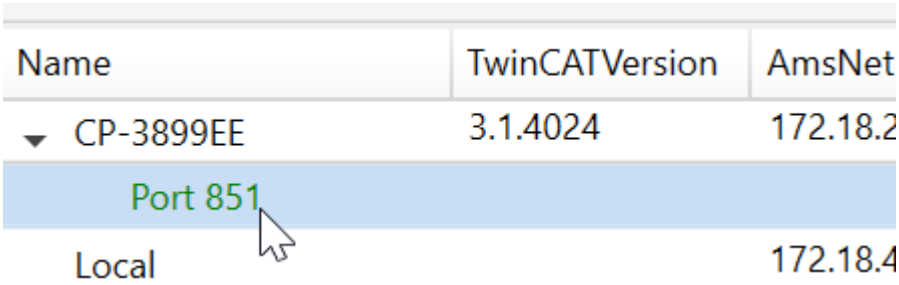
⇨ Der Symbol Explorer wurde nun zum Öffnen von Snapshots als Programm verknüpft.

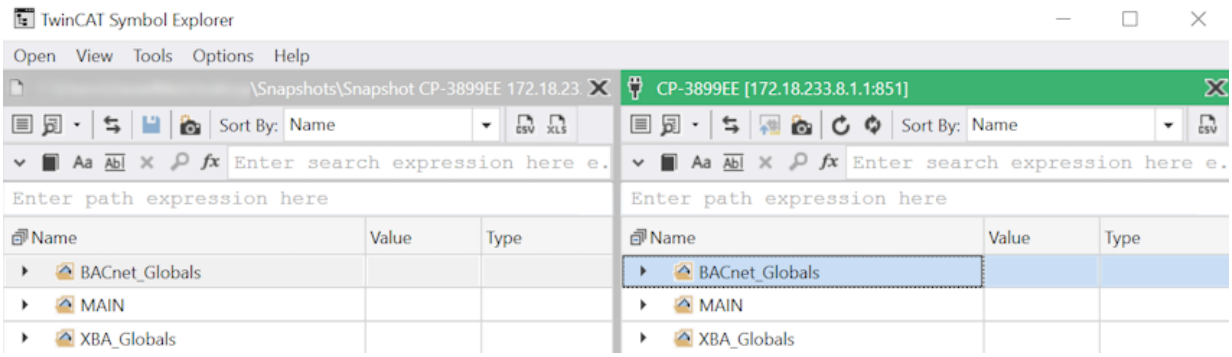
7.2.7.2 Instanz-Ansicht zur Parametrierung von AI- und AO-Objekten

Das folgende Beispiel zeigt die Erstellung einer Instanz Ansicht, um typische Parametrierungsaufgaben an Analog-Objekten vorzunehmen. Die Ansicht kann genutzt werden um z.B.: die Parametrierung eines Sensorabgleichs zu protokollieren.

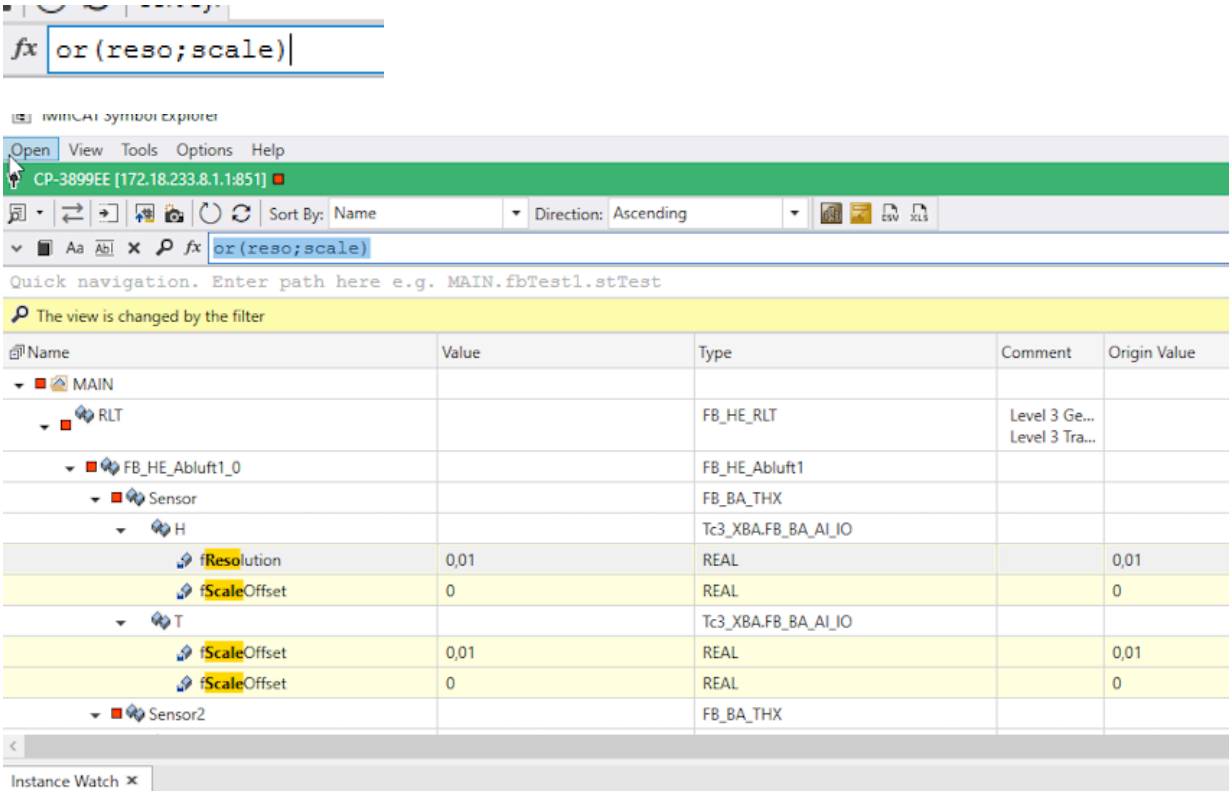
Verbindungsaufbau Zielsystem

1. Loggen Sie sich auf das Zielsystem im Symbolexplorer ein.

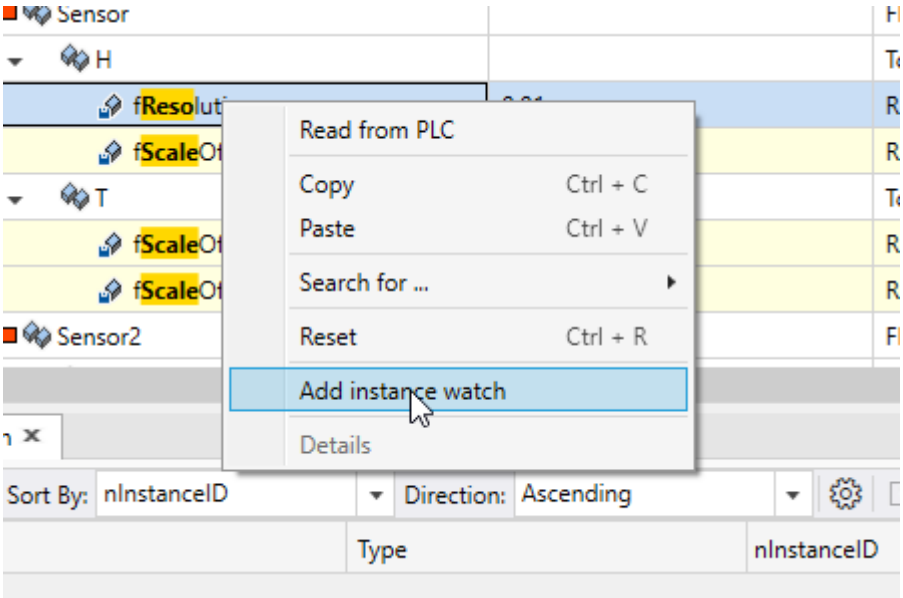




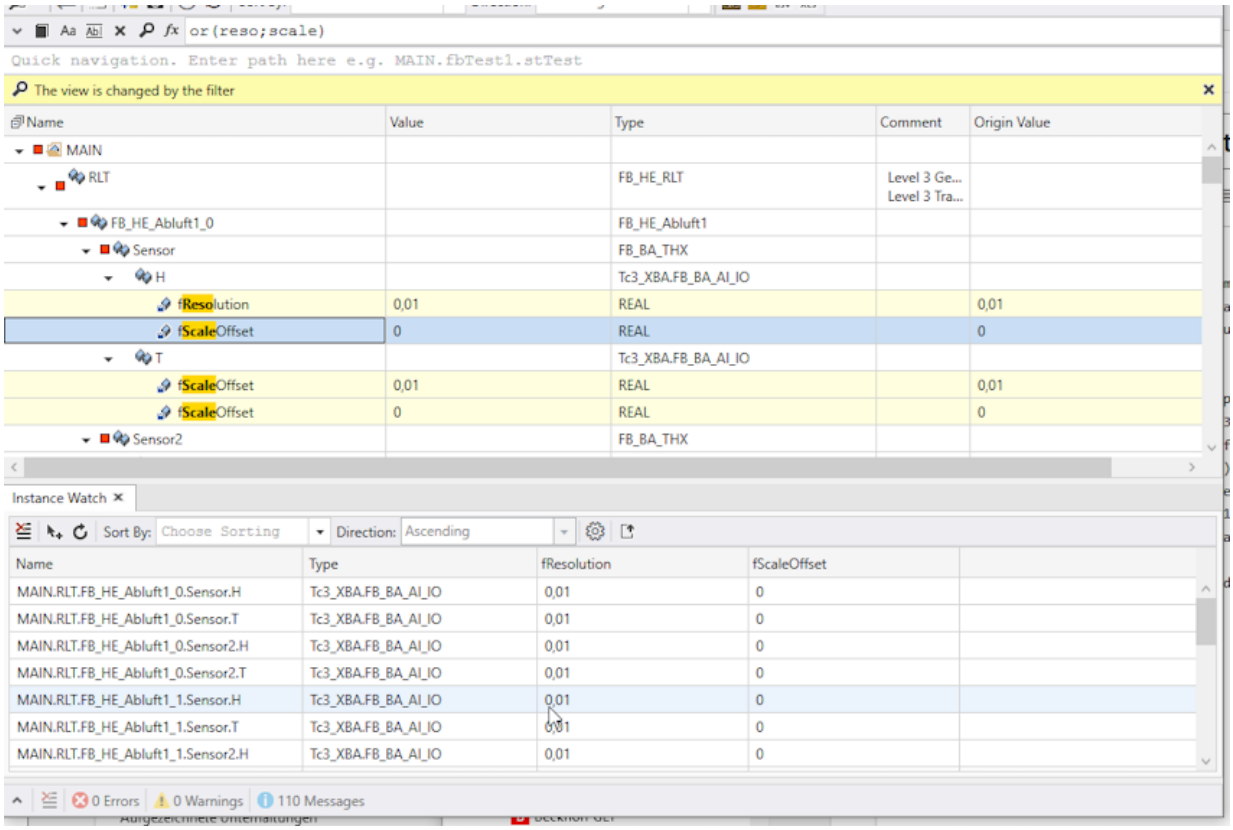
2. Setzen Sie einen Filterausdruck.



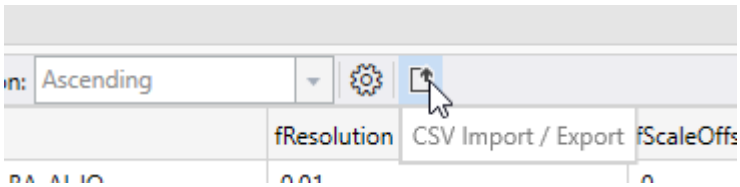
3. Öffnen Sie das InstanceWatch Fenster.



4. Ziehen Sie die Symbole **ScaleOffset** und **Resolution** per Drag-and-Drop in das Instanz Fenster.



5. Generieren Sie einen Filter aus der Instanz Ansicht.



⇨ Diese Funktion erzeugt einen Filter für die Baumansicht mit folgendem Ausdruck:

```
And(Or(Name='fResolution';Name='fScaleOffset');Parent(Or(Type='Tc3_XBA.FB_BA_AI_IO')))
```

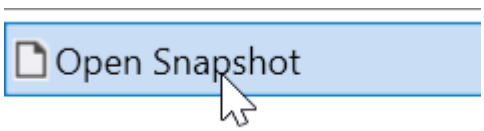
7.2.7.3 Synchronisieren von gefilterten Daten, Beispiel mit Exclude

Im folgenden Beispiel werden Daten aus einem Snapshot synchronisiert:

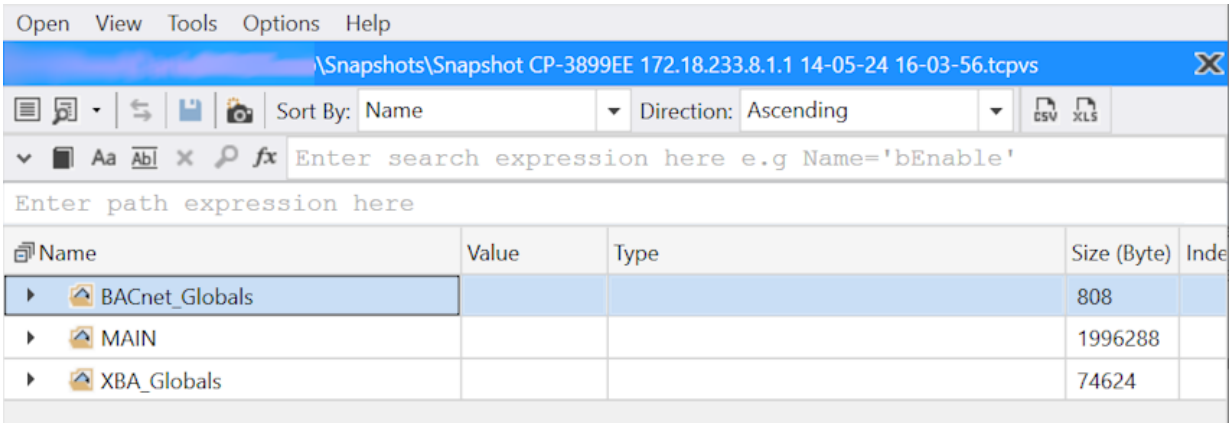
Es sollen alle Änderungen vom Snapshot in die Steuerung übernommen werden, außer Symbole mit Betriebsdaten vom Namen *nActiveTimeElapsed* und *nStateChangeCount*.

Verbindungsaufbau Zielsystem und laden eines Snapshot

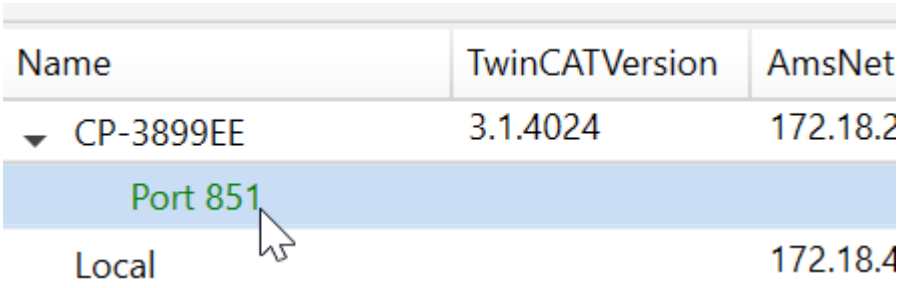
1. Öffnen Sie den Snapshot [▶ 1325] mit Datensicherung des betroffenen Zielsystems (PLC).

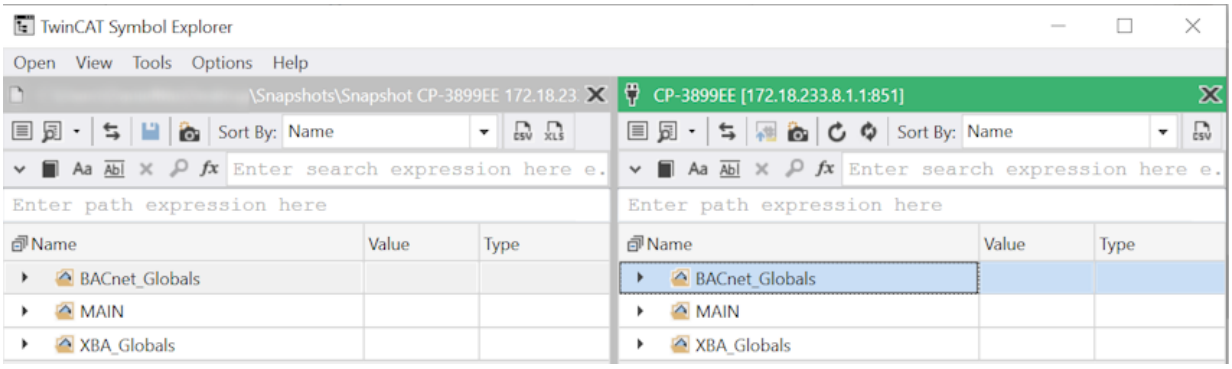


2. Wählen Sie **BACnet_Globals** aus.



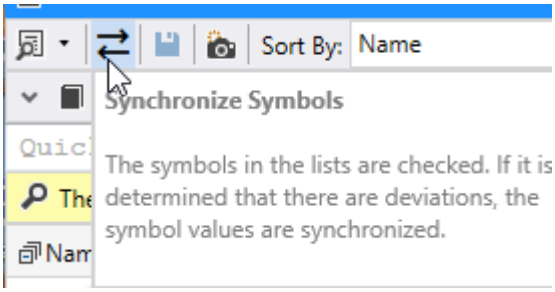
3. Loggen Sie sich im Symbol Explorer auf das Zielsystem ein.





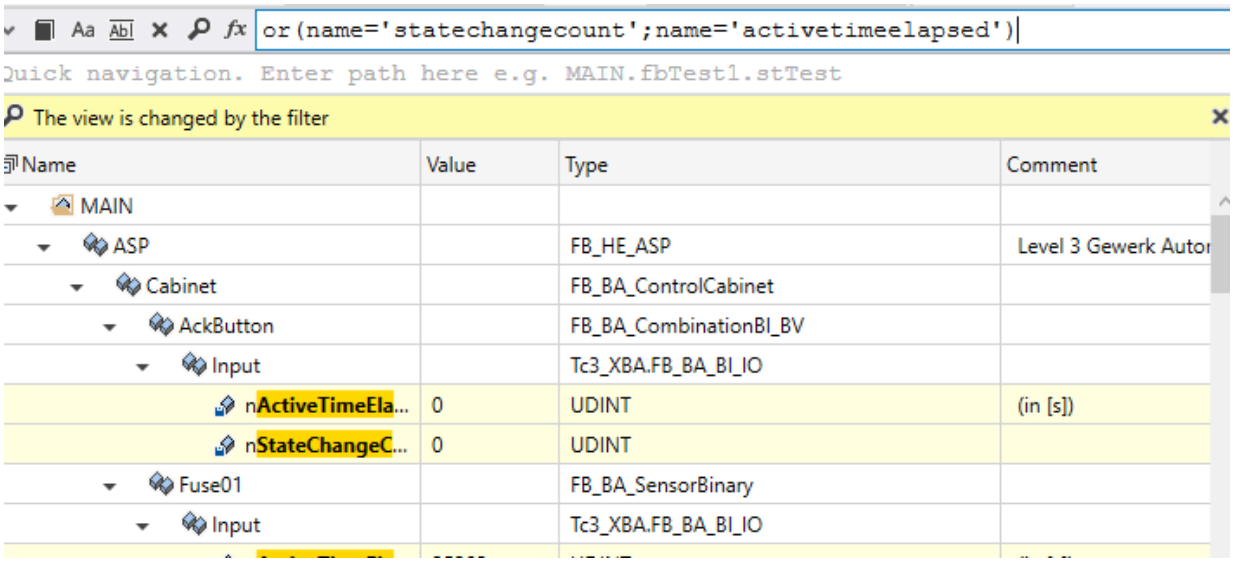
Synchronisieren und Snapshot -> Zielsystem mit gefilterten Symbolen

1. Synchronisieren Sie die Daten vom Snapshot auf das Zielsystem durch Betätigung des Buttons Sync im Hauptfenster des Snapshots.

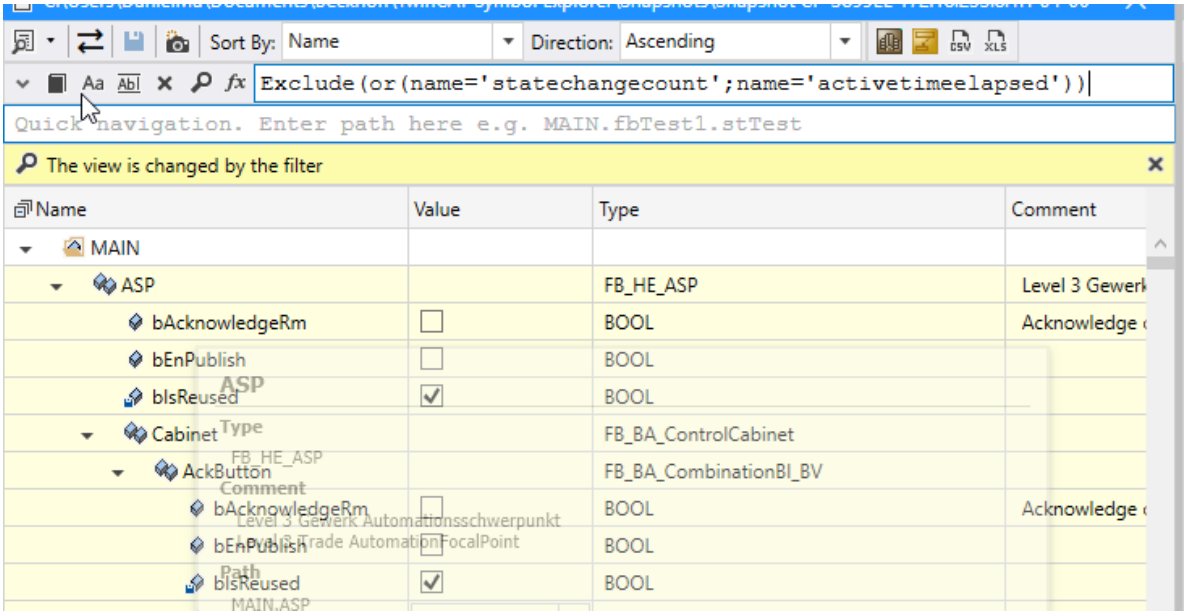


⇨ Vorschlag zur Synchronisierung: Es werden 43 Symbole zur Synchronisierung gefunden.

2. Setzen Sie im Snapshot Filter auf die Symbole **nStateChangeCount** und **nActiveTimeElapsed.**
   Filter:
   or(name='statechangecount';name='activetimeelapsed')

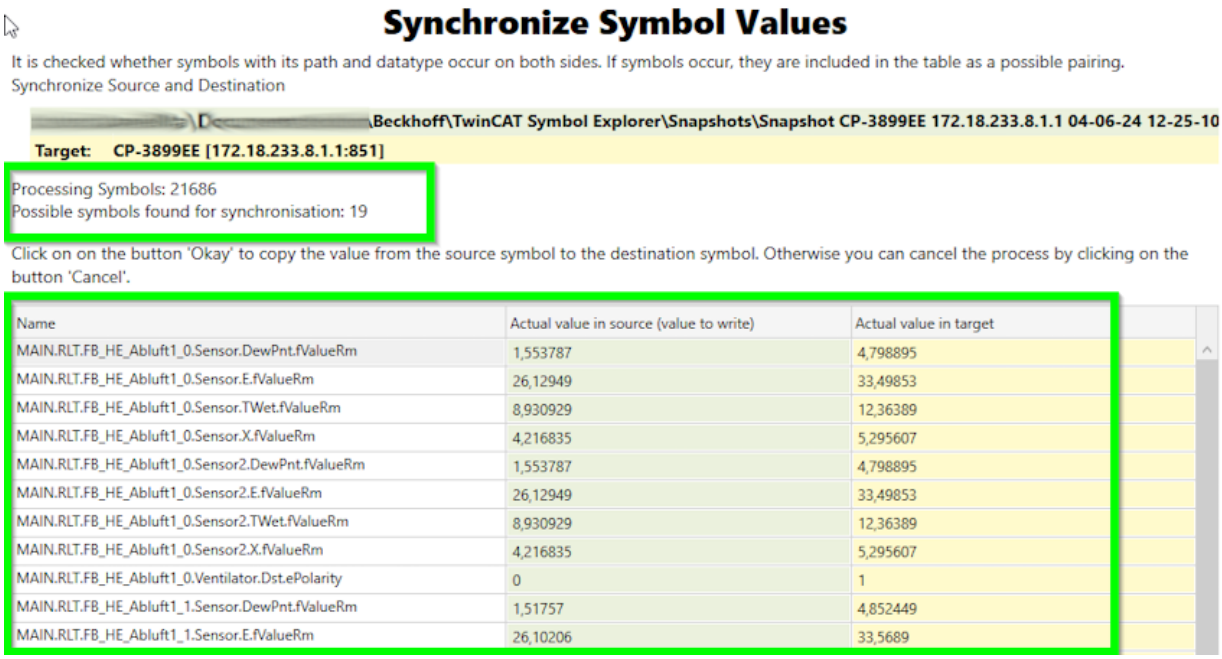


3. Negieren Sie die Auswahl durch die Exclude Funktion.
   ⇨ Es werden alle Symbole außer **nStateChangeCount** und **nActiveTimeElapsed** angezeigt.



4. Klicken Sie erneut auf den Button **Sync**.

⇨ Nun stehen nur noch 19 Symbole zur Synchronisierung bereit. Symbole, welche die Betriebsdaten erhalten, wurden erfolgreich herausgefiltert



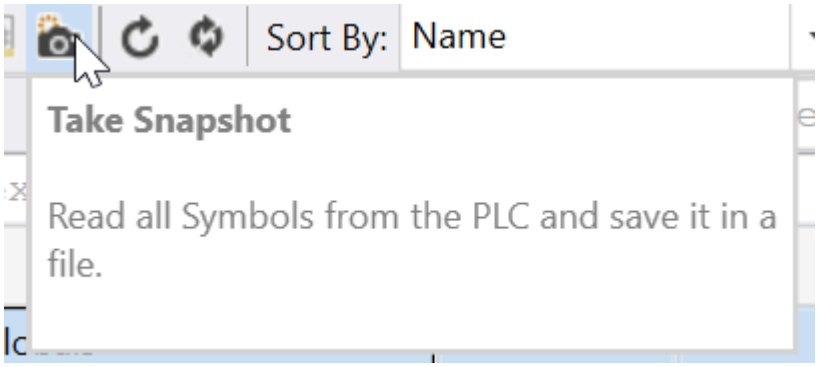
7.2.7.4 BACnet ObjectIDs mit Snapshot sichern und wiederherstellen

Diese Anleitung dient als Workflow um BACnet ObjectIDs aus einem SymbolExplorer Snapshot wieder in eine PLC zurückzuschreiben.

Das Wiederherstellen von Objekt Instanz IDs ist dann notwendig, wenn z. B. eine Steuerung ausgetauscht wird, welche bereits mit einer BACnet MBE kommuniziert.

IDs sichern:

1. Login mit dem SymbolExplorer.
2. Snapshot erstellen



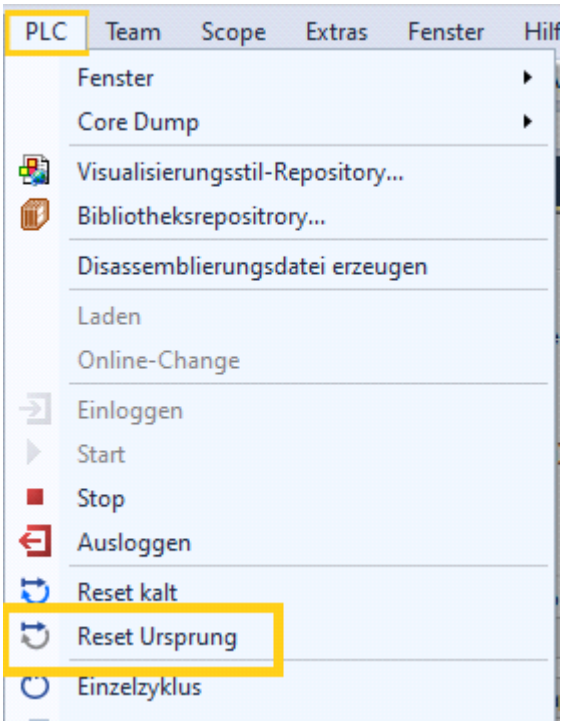
Starten Sie den SymbolExplorer und verbinden Sie ihn mit einer Steuerung. Klicken Sie im Online-Fenster auf den Take Snapshot Button, um einen Snapshot zu erstellen.
Wählen Sie den Speicherort für die Snapshot-Datei aus.

⇨ Nachdem Sie den Ordner ausgewählt und mit Ok bestätigt haben, werden sämtliche Werte der Symbole in der Liste mit der PLC synchronisiert und in die Datei geschrieben.

IDs aus einem Snapshot im Zielsystem wiederherstellen:

Start in XAE

1. PLC urlöschen



2. PLC einloggen

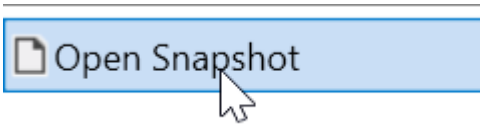


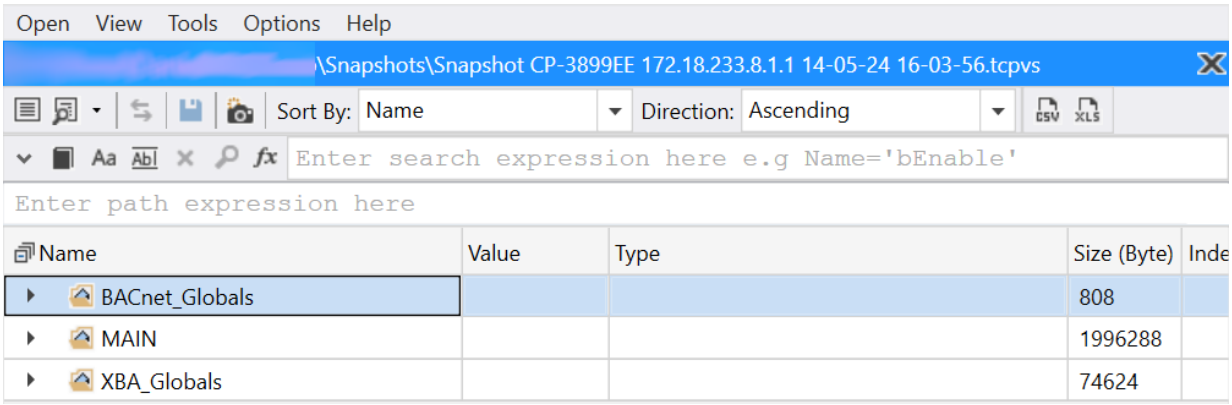
3. PLC **nicht** starten!



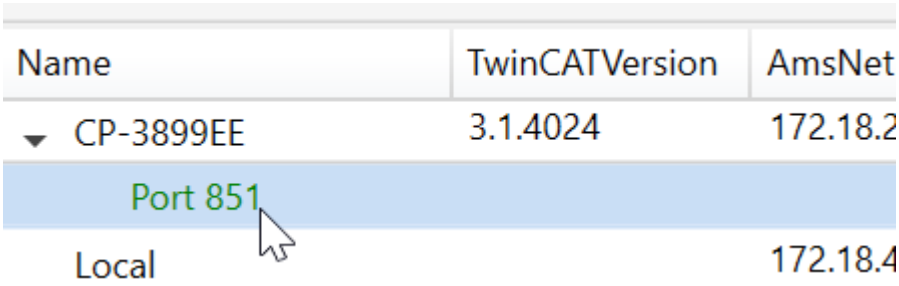
SymbolExplorer

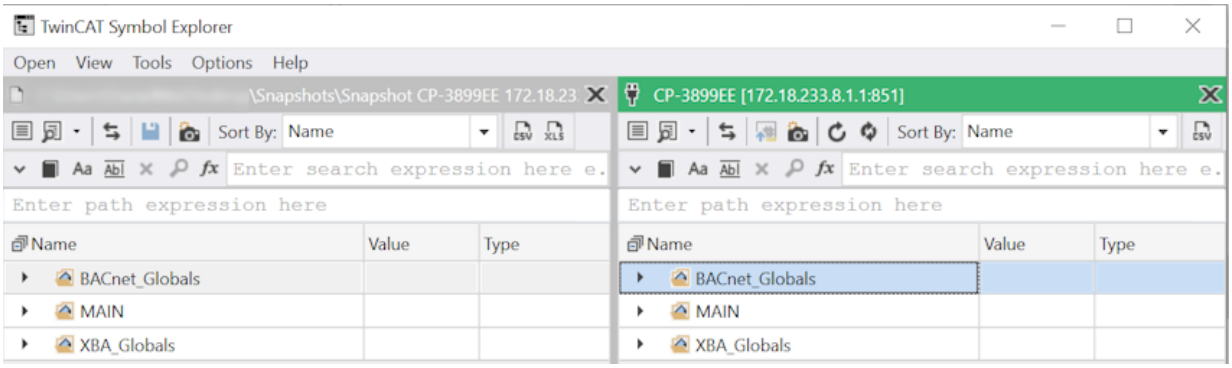
1. Öffnen Sie den Snapshot [▶ 1325] mit der Datensicherung des betroffenen Zielsystems (PLC).



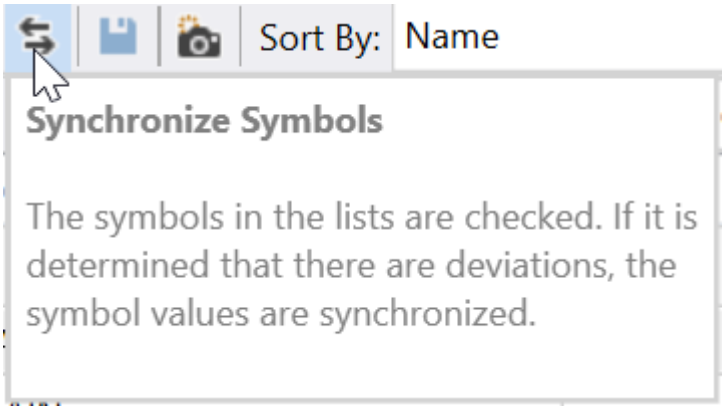


2. Loggen Sie sich auf das Zielsystem im Symbolexplorer ein

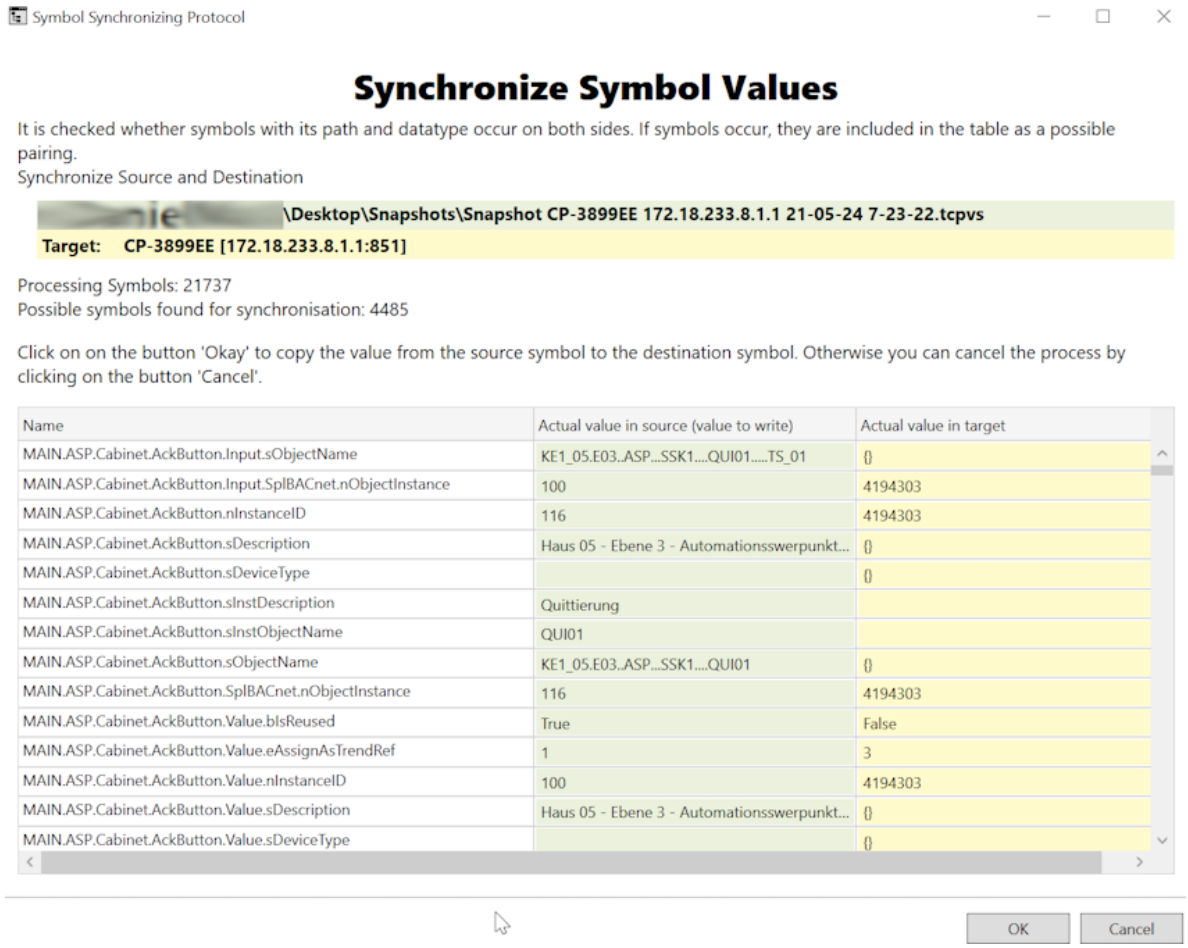




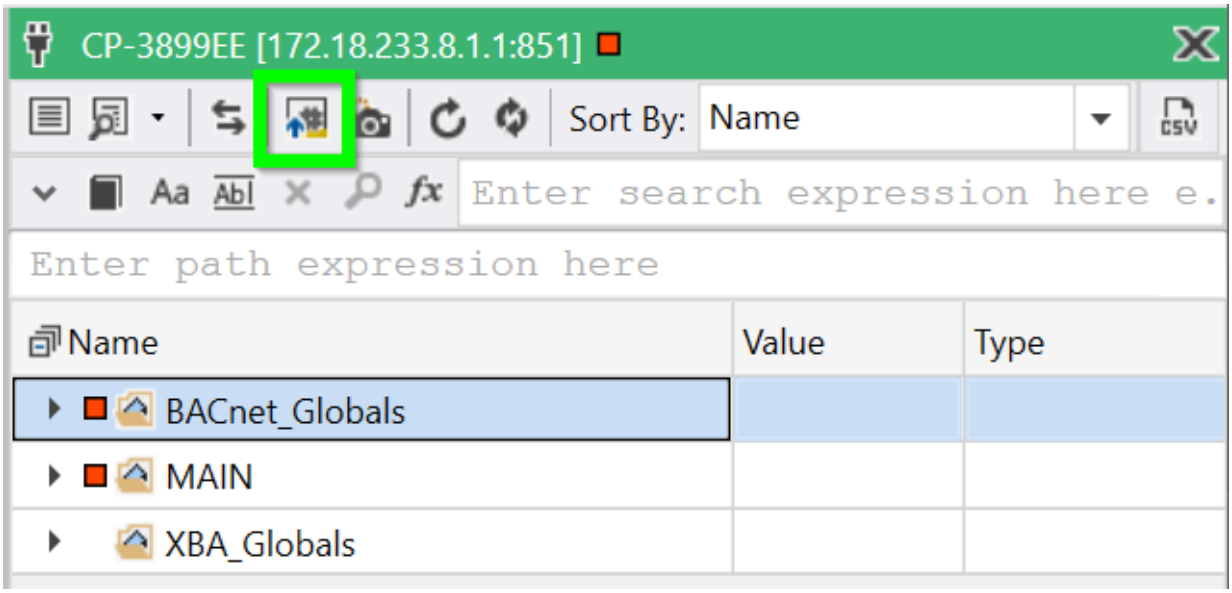
3. Synchronisieren der Daten vom Snapshot auf das Zielsystem



⇨ Die Symbole zur Synchronisierung werden im Dialogfenster angezeigt und zur Übertragung auf das Zielsystem vorbereitet.

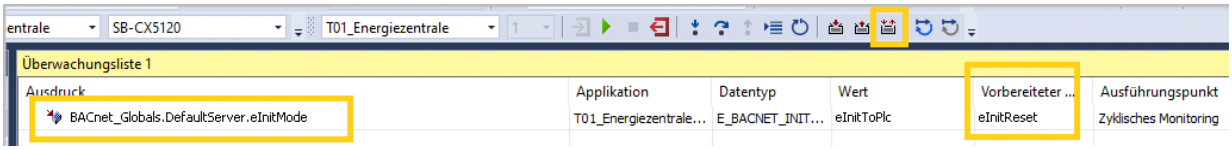


4. Schreiben Sie die Symbole aus dem Snapshot mit dem SymbolExplorer in die PLC.



Prüfen in der XAE

1. Schreiben Sie den Init-Mode vom BACnet Server in der PLC auf *eInitReset*.

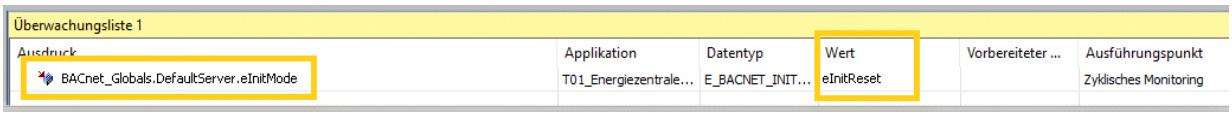


2. Starten Sie die PLC.

⇨ Der BACnet Server startet im *eInitMode* und erstellt die Objekte neu.







7.2.8 Extensions

Der Funktionsumfang des Symbol Explorers wurde um branchenspezifische Features erweitert:

- Building Automation I/O Export [▶ 1361]:
  Diese Erweiterung dient dazu, I/O-Variablen des Typs *Tc3_XBA.FB_BA_**_Raw* zu exportieren.
- Building Automation Instance Watch [▶ 1363]:
  Diese Erweiterung ist eine umfangreichere Instance Watch, die auf die Bedürfnisse der Gebäudeautomation angepasst ist.

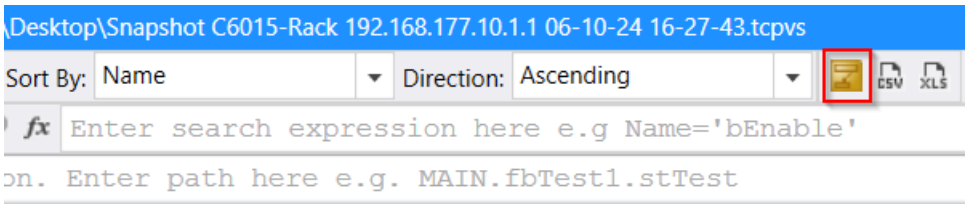
7.2.8.1 Building Automation I/O Export

Dieses Plugin erstellt eine Liste aus allen Symbolen der Typen "Tc3_XBA.FB_BA_**_Raw", die in der geöffneten Route oder dem geöffneten Snapshot vorhanden sind. Die gefunden Symbole können als CSV oder TwinCAT Function Block abgespeichert werden.

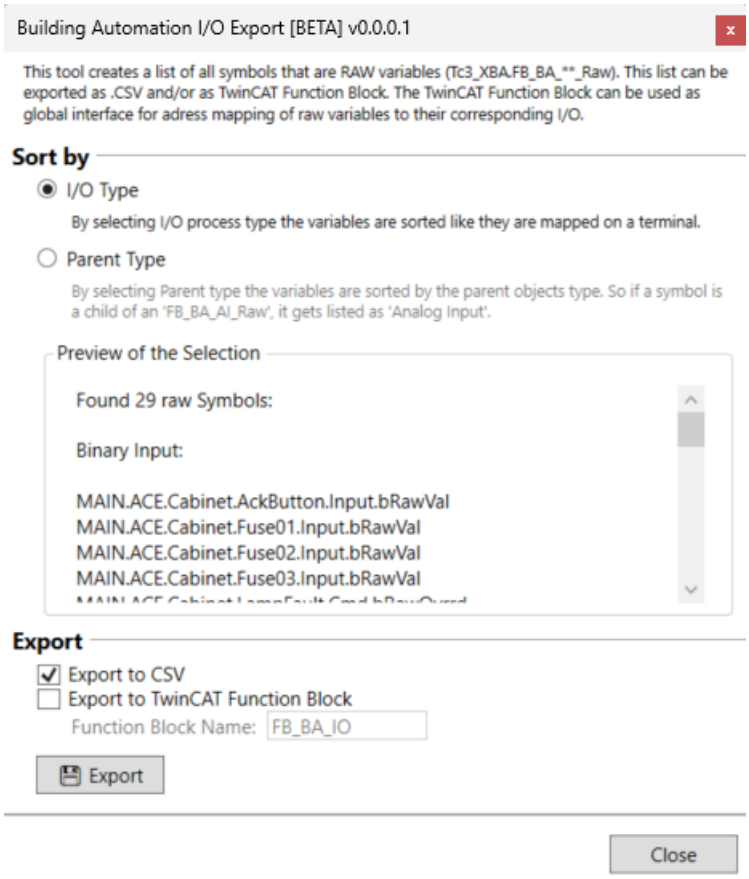
Start des I/O Export

Der I/O Export steht in einer geöffneten Route oder Snapshot zur Verfügung.

Er ist als Button in der oberen Toolbar zu finden:



Es öffnet sich ein Fenster, in dem zunächst die RAW-Objekte nach dem eigenen I/O-Prozess-Typen oder nach dem Typen des Elternelementes sortiert werden können:

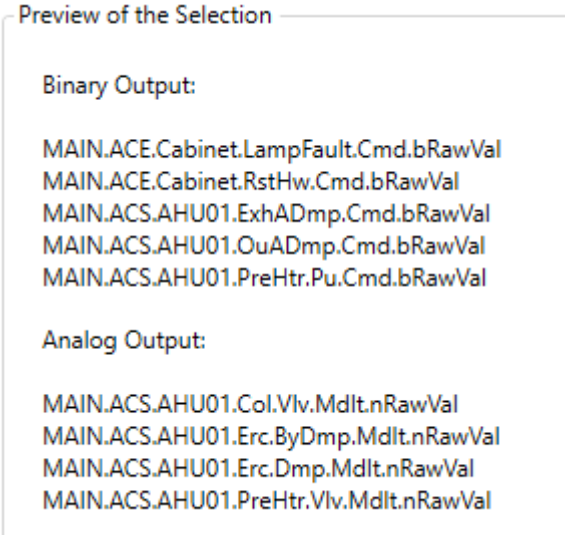


Wird nach dem I/O-Prozess-Typen (**I/O Type**) sortiert, wird das Objekt anhand des eigenen Typen einsortiert.

Wenn nach dem Typen des Elternelementes (**Parent Type**) sortiert wird, entscheidet der Typ des Elternelementes.

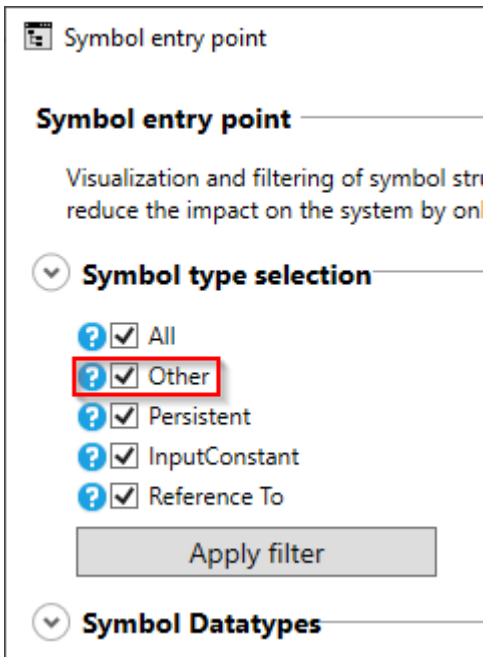
Resultierend aus der oberen Selektion wird die Vorschau (**Preview of the Selection**) aktualisiert.

Die Objekte werden in die Kategorien **Binary Input**, **Binary Output**, **Analog Input** und **Analog Output** einsortiert:



Falls keine Symbole gefunden wurden, soll geprüft werden, ob die Solution Symbole der Typen *Tc3_XBA.FB_BA_**_Raw* enthält.

Zudem muss der Symbol Explorer so eingestellt werden, dass in dem Symbol Entry Point Dialog unter **Symbol Type Selection** der Eintrag **Other** ausgewählt ist.



Bei einem Snapshot muss dies vor der Erstellung beim **Symbol Entry Point**-Dialog einer Route eingestellt sein. Sonst werden diese Symbole nicht in einen Snapshot gespeichert.

Mit dem Button '□ Export' können die gefundenen RAW-Variablen exportiert werden.

Als Exportmöglichkeit steht eine CSV-Datei und/oder ein TwinCAT Funktionsblock zur Verfügung.

In dem Freitextfeld **Function Block Name**: kann der Name des Funktionsblocks vergeben werden.

Der TwinCAT Funktionsbaustein kann als globales Interface für Adress-Mappings verwendet werden.

Zur Verwendung müssen alle Platzhalter ('YOUR_IO_**') mit den entsprechenden I/O-Variablen ersetzt werden.

7.2.8.2 Instance Watch BA

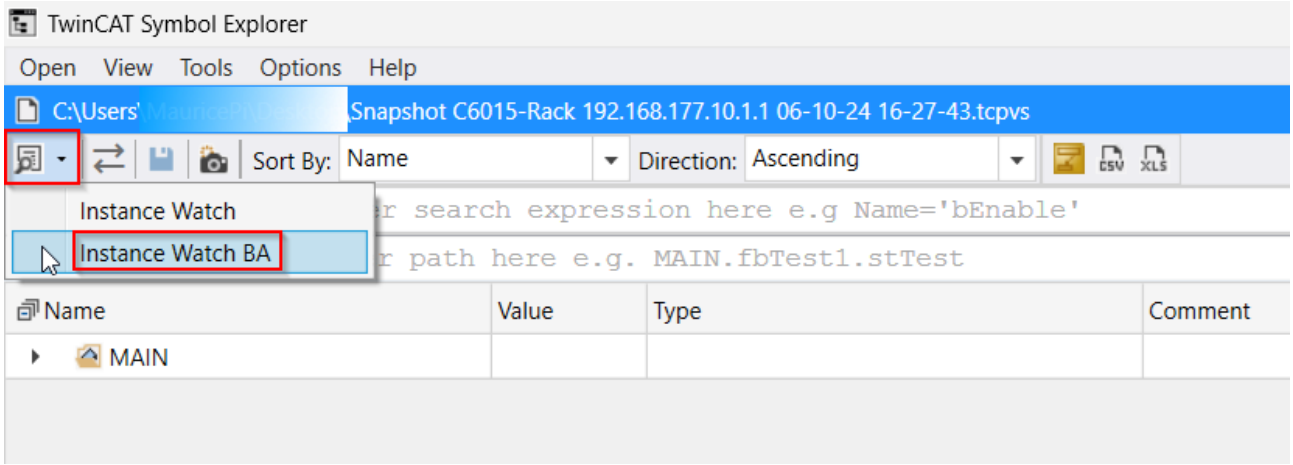
Die Instance Watch BA ist eine Erweiterung der im Symbol Explorer integrierten Instance Watch.

Die Erweiterungen betreffen ein angepasstes Instance Watch BA Template, welches beim Start der Instance Watch BA geladen wird und eine erweiterte Ansicht komplexer BA-spezifischer Datentypen.

Start der Instance Watch BA

Zuerst muss der Symbol Explorer gestartet und ein Snapshot oder eine Route geöffnet werden.

Danach kann in der linken oberen Ecke das Drop-Down Menü der Instance Watches die **Instance Watch BA** ausgewählt werden:



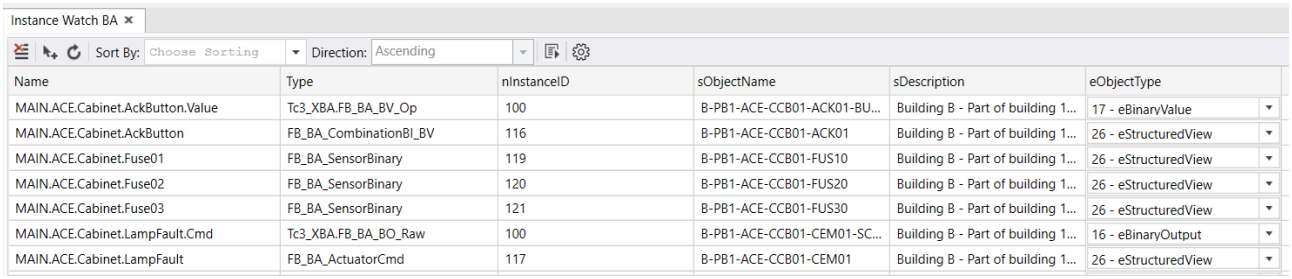
Es öffnet sich eine Instance Watch mit dem Titel **Instance Watch BA** und ein bereits geladenes Instance Watch Template für die Building Automation.

BA-spezifisches Instance Watch Template

Dieses Template beinhaltet jedes Objekt, welches das Interface "Tc3_XBA.I_BA_View" oder "Tc3_XBA.I_BA_Object" implementiert hat.

Zudem werden die Properties *nInstanceId*, *sObjectName*, *sDescription* und *eObjectType* für diese Objekte angezeigt.

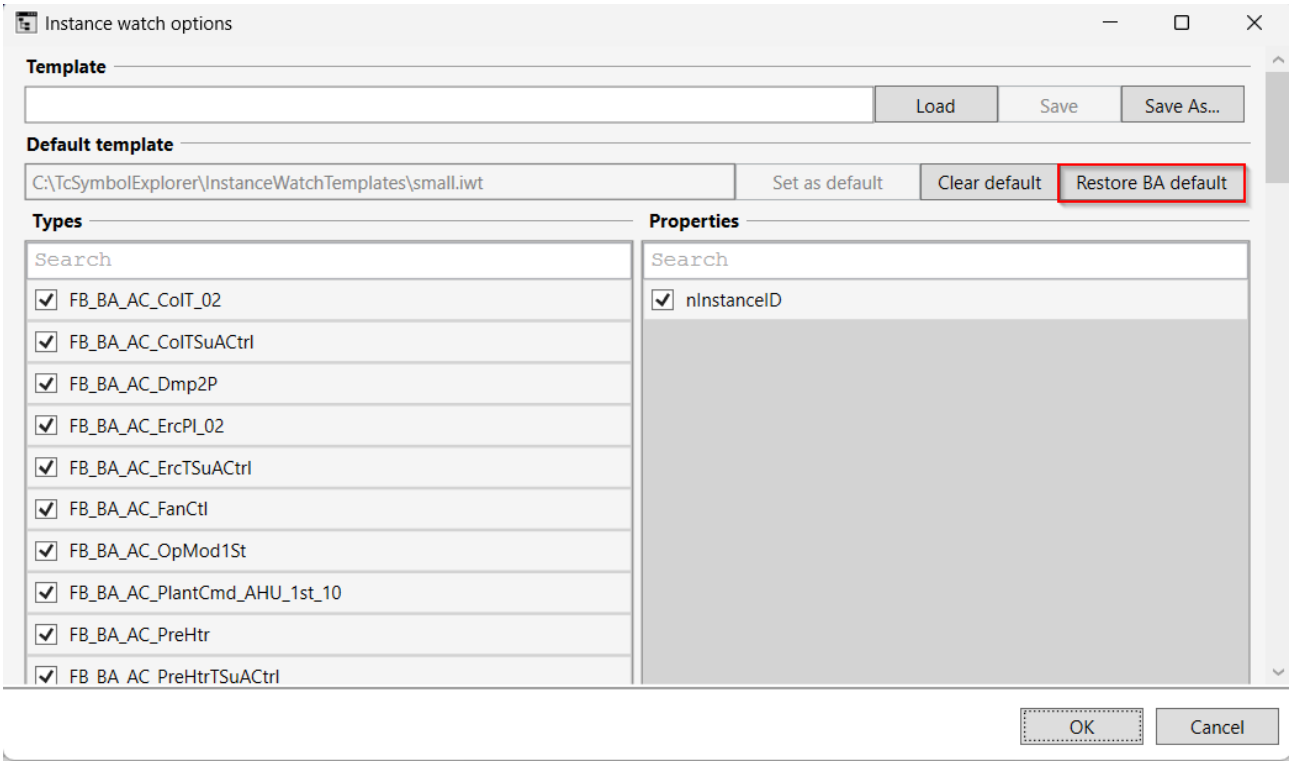
Dieses Template wird bei jedem Start der Instance Watch BA geladen.



Zurücksetzen des Instance Watch Templates

Sind Änderungen am Instance Watch Template vorgenommen worden (Bspw. Properties hinzugefügt / entfernt), kann das BA-spezifische Instance Watch Template wiederhergestellt werden. Dazu müssen die **Instance Watch Options** geöffnet werden. Um das Template zurückzusetzen, muss der Button **Restore BA default** betätigt werden.

Danach mit **OK** bestätigen.



Erweiterte Ansicht BA-spezifischer komplexer Typen

Mit der Instance Watch BA können einige komplexe Typen übersichtlich angezeigt und bearbeitet werden.

Hierzu gehören aktuell diese Typen:

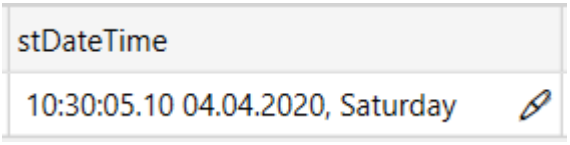
- Tc3_XBA.T_BA_EventTransitionText
- Tc3_XBA.T_BA_EventTransitions

- Tc3_XBA.ST_BA_LimitParam
- Tc3_XBA.ST_BA_TimeDelayParam
- Tc3_BA2_Common.ST_BA_StatusFlags
- Tc3_BA2_Common.ST_BA_Date
- Tc3_BA2_Common.ST_BA_Time
- Tc3_BA2_Common.ST_BA_DateTime

Ansicht

Um die vereinfachte Ansicht der oben genannten Symbole zu sehen, müssen diese Symbole, wie gewohnt, der Instance Watch BA hinzugefügt werden.

Im Beispiel vom Tc3_BA2_Common.ST_BA_DateTime sieht das Symbol folgendermaßen aus:



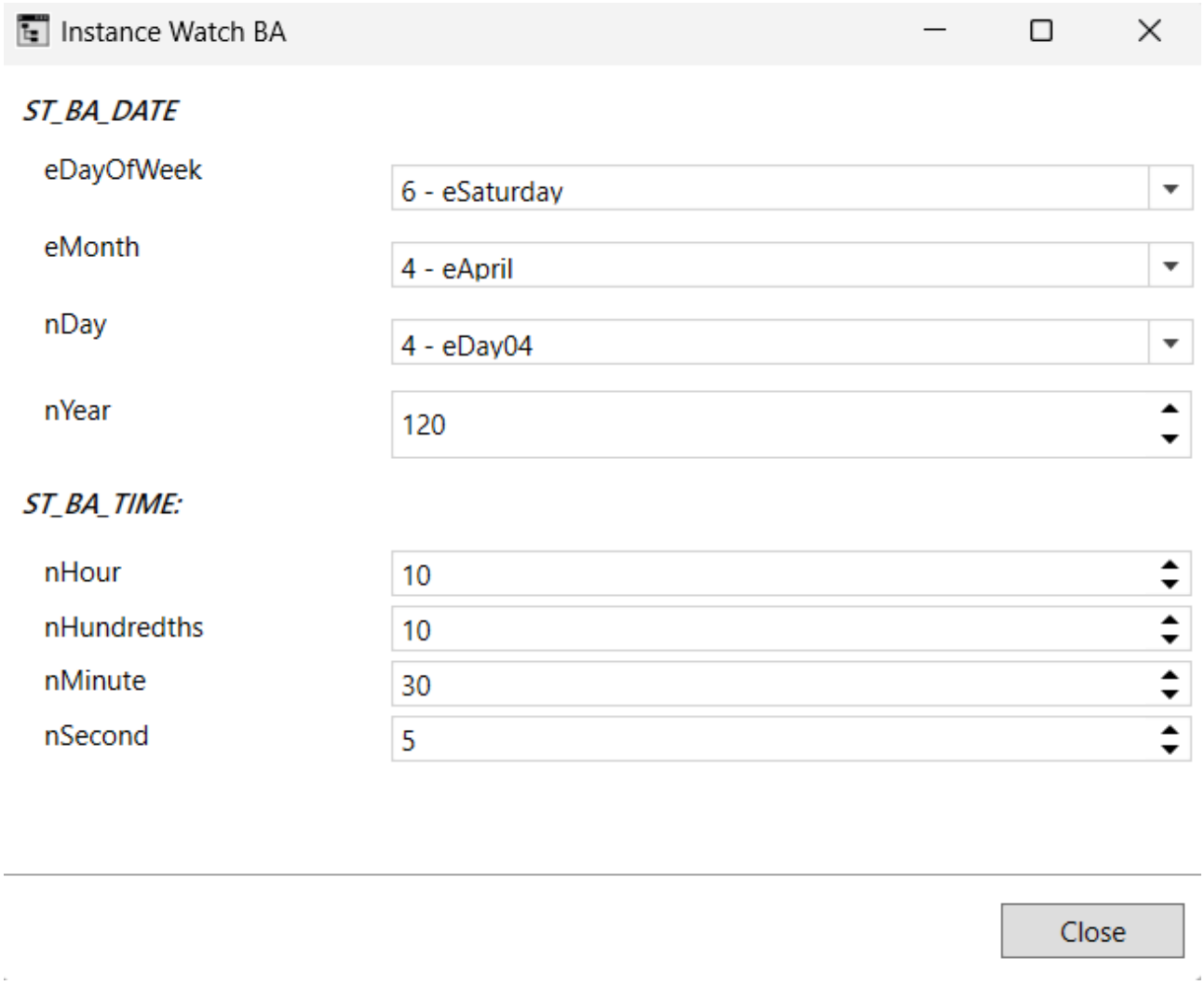
Bearbeitung

Die Bearbeitung von Symbolen mit maximal vier Subsymbolen erfolgt durch einen Doppelklick auf die Zelle des Symbols:



Bei Symbolen, die mehr als vier Subsymbole haben, erfolgt die Bearbeitung des Symbols über das Stift-Symbol. Dies erscheint nur bei diesen Symbolen.

Nach dem Klick öffnet sich ein Dialog:



Änderungen werden in beiden Fällen sofort angewendet, aber noch nicht mit der PLC synchronisiert bzw. im Snapshot gespeichert. Deshalb kann nach dem Bearbeiten des Symbols der Dialog geschlossen werden.

7.2.9 Beispiele für reguläre Ausdrücke

Zum Verständnis der verwendeten Ausdrücke zum Filtern und Sortieren verweist der folgende Link auf eine Sammlung von häufig gebrauchten, regulären Ausdrücken:

Übersicht Reguläre Ausdrücke

7.3 Template Repository

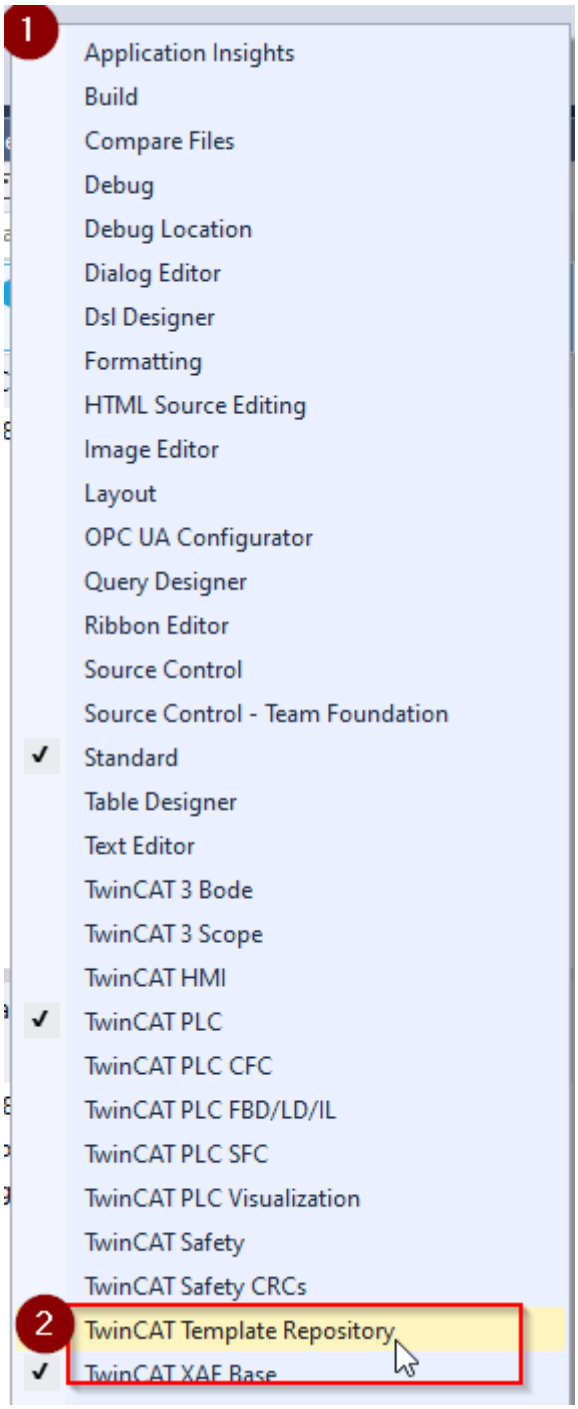
Das Template Repository ist eine App zur Verwaltung von Templates.

Starten der App in der Entwicklungsumgebung

✓ Mit dem TF8040 Setup wird das Template Repository automatisch in der TwinCAT Entwicklungsumgebung (XAE Shell oder Visual Studio) installiert.

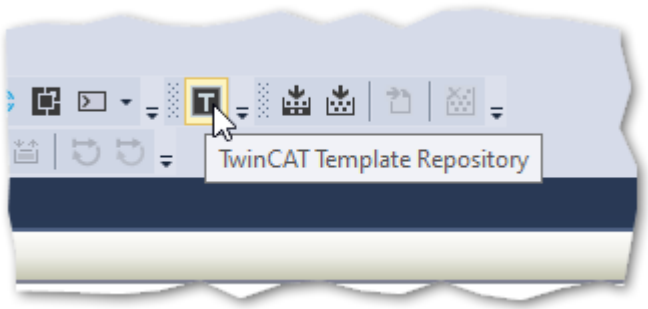
1. Mit einem Rechtsklick auf ein freies Feld in der Symbolleiste öffnen Sie ein Kontextmenü.

2. Wählen Sie im Kontextmenü die App **Template Repository** aus.



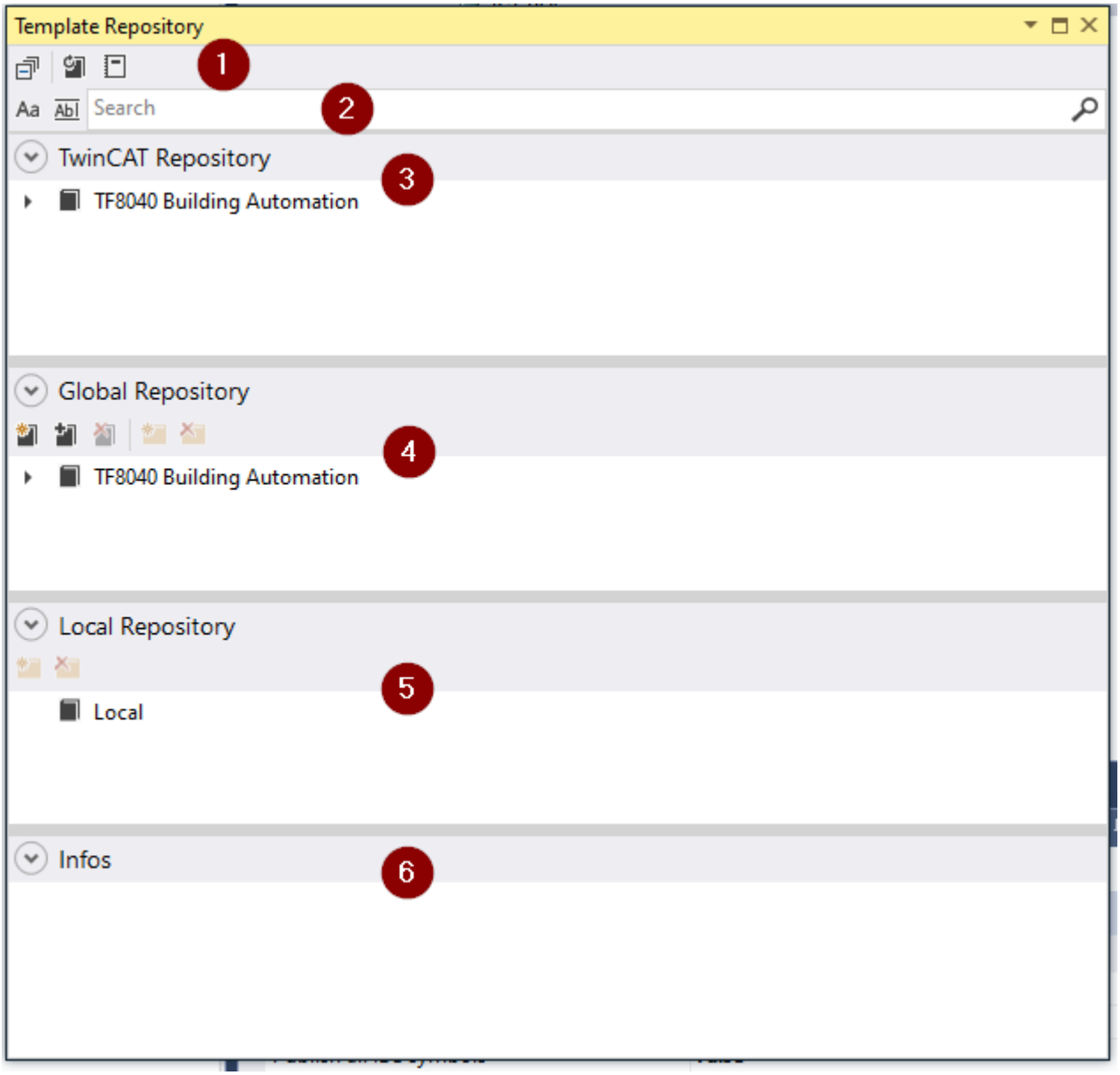
3. Starten Sie die App durch Klick auf das Icon **TwinCAT Template Repository**.

⇨ Es öffnet sich das Hauptfenster.



Hauptfenster

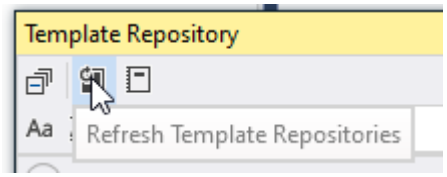
Das Hauptfenster ist in 6 verschiedene Bereiche aufgeteilt:



1 Menüleiste

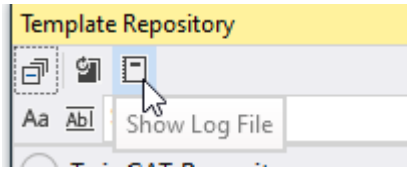
In der Menüleiste stehen folgende Funktionen zur Verfügung:

- **Refresh Template Repositories** aktualisiert alle Repositories



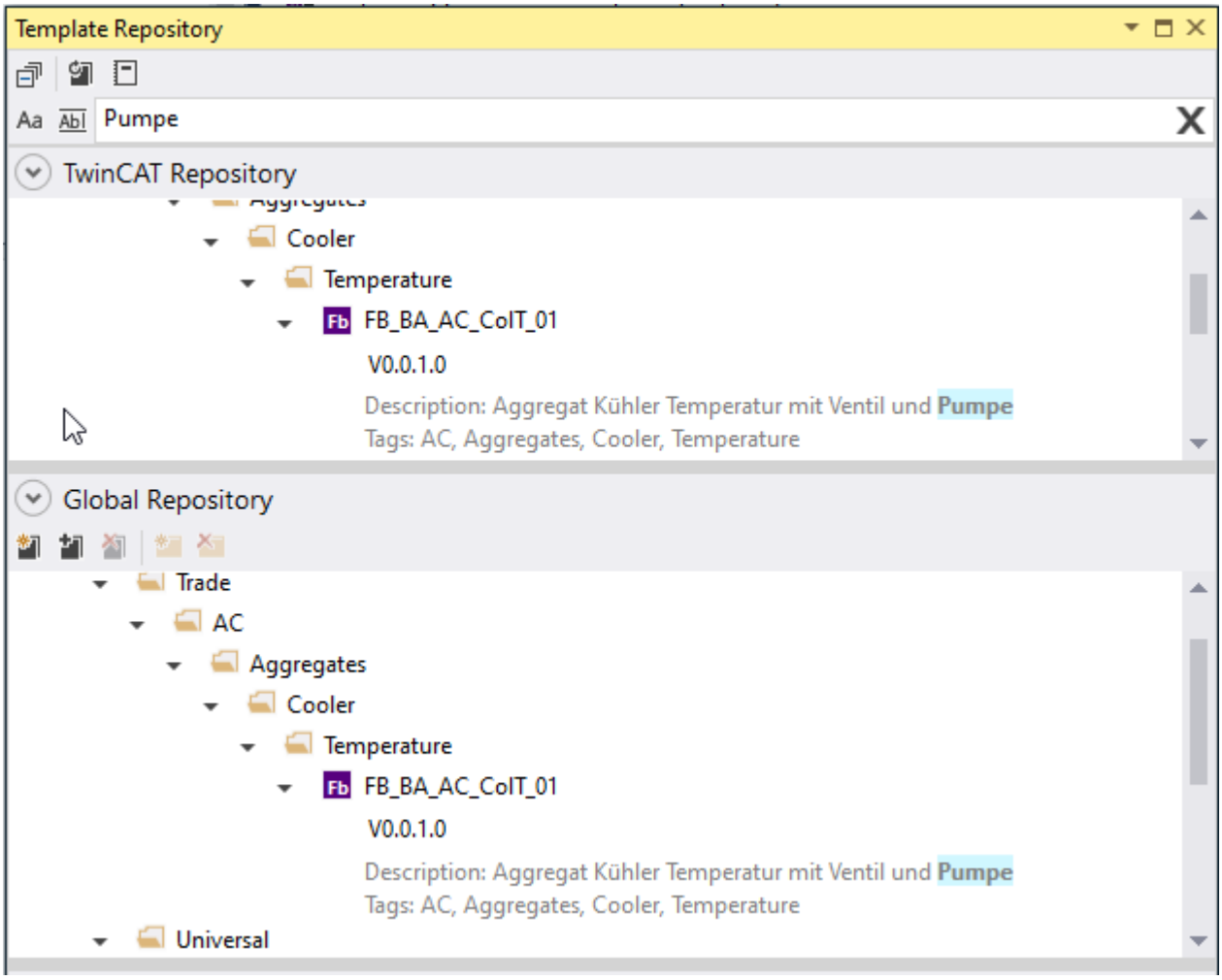
- **Show LogFile** öffnet den Log File Dialog

Hier werden detaillierte Log Informationen zur verwendeten Softwareversion und Fehler angezeigt.



2 Suchleiste

In der Suchleiste werden die eingebundenen Repositories nach Schlagwörtern (*tags*) oder Namen innerhalb der Templates durchsucht. Die Suchergebnisse werden in den jeweiligen Bereichen angezeigt.



3 TwinCAT Repository Bereich

Im TwinCAT-Repository-Bereich befinden sich alle Templates, welche mit der TF8040 Installation ausgeliefert werden. Aus diesem Bereich können Templates nur entnommen aber nicht hinzugefügt werden.

Mit der TF8040-Installation erhalten Sie ein Repository mit Templates. Um diese für ihre Projektspezifischen Applikationen nutzen zu können, müssen Sie vorrübergehend das Repository in den globalen Bereich laden.

4 Global Repository Bereich

Der Global-Repository-Bereich dient als Ablageort für alle kundenspezifischen Templates die projektübergreifend von mehreren Entwicklern genutzt werden möchten. Hier können Templates projektübergreifend entnommen und hinzugefügt werden. Der Ablageort kann z. B.: ein Netzlaufwerk im Firmennetzwerk oder ein beliebiger Ort auf der lokalen Festplatte sein.

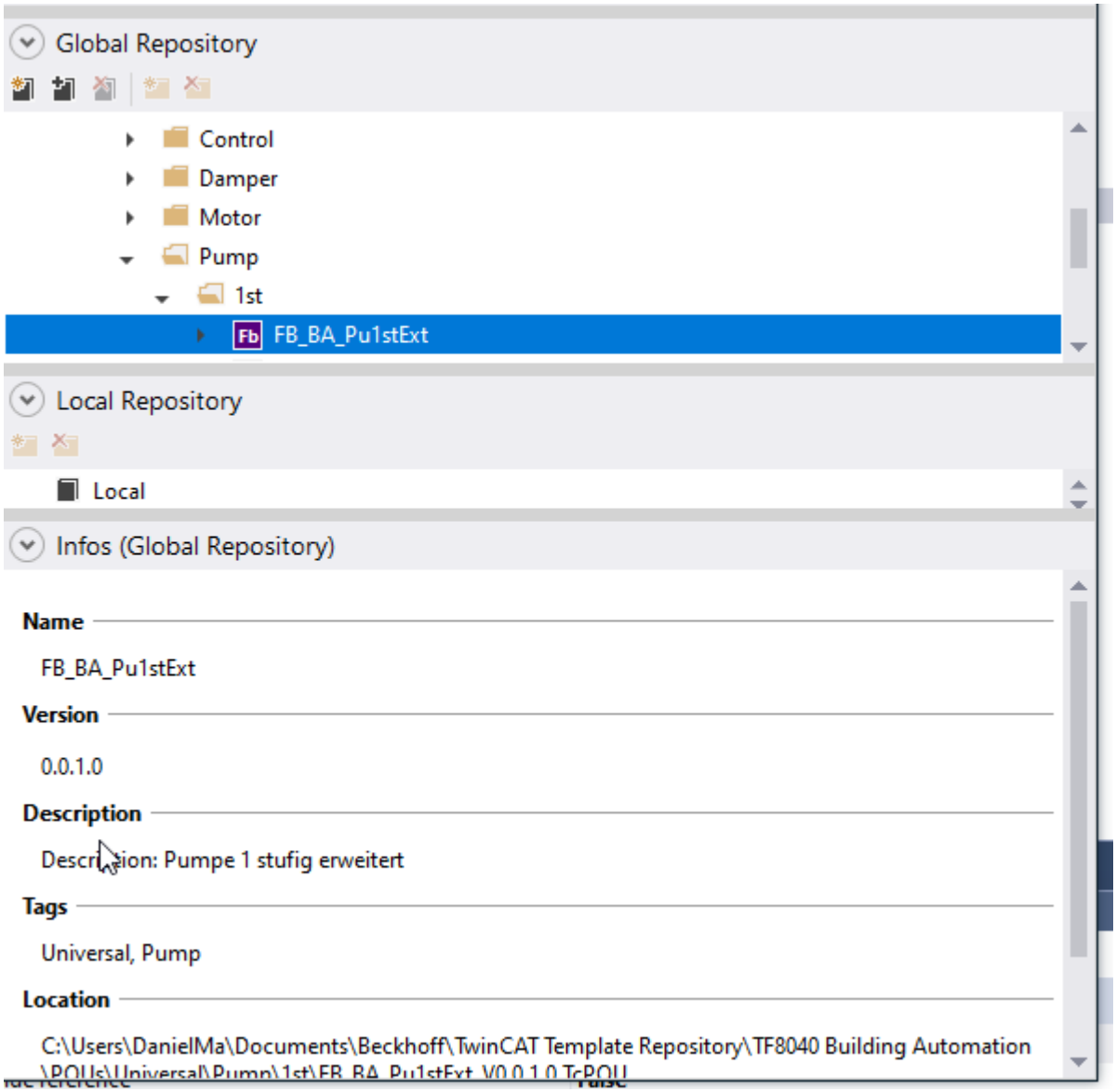
5 Local Repository Bereich

Im Local-Repository-Bereich werden alle Templates abgelegt, die in der lokalen Solution genutzt werden.

Ein Austausch von Templates zwischen lokalem und globalem Bereich ist möglich.

6 Infobereich

Im Infobereich werden detaillierte Informationen der jeweils aktuell gewählten Templates angezeigt.

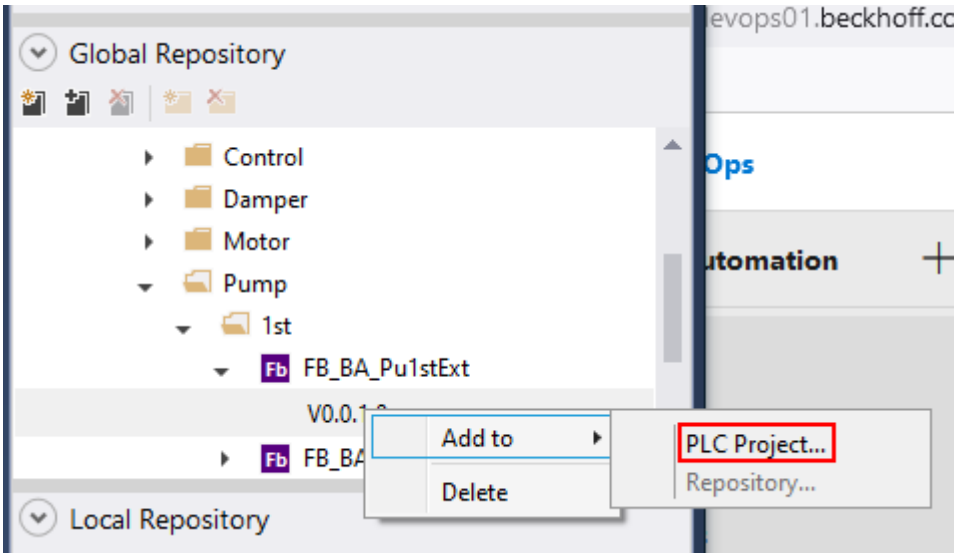


Hinzufügen von Templates in ein PLC-Projekt

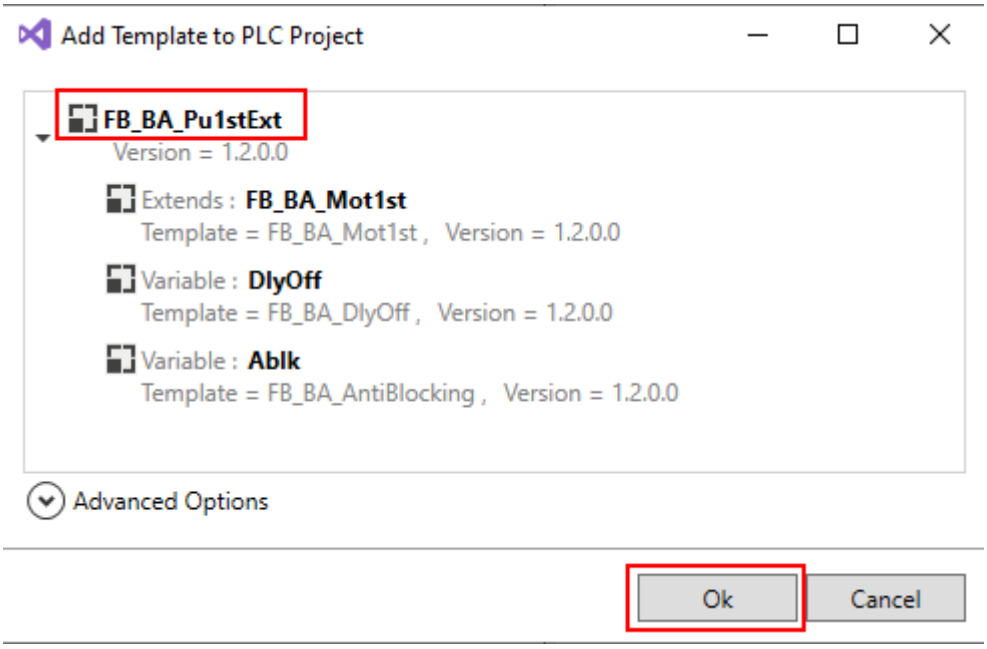
Templates können auf zwei Wegen in die aktuelle Solution integriert werden:

✓ **Doppelklick** auf das gewünschte Template

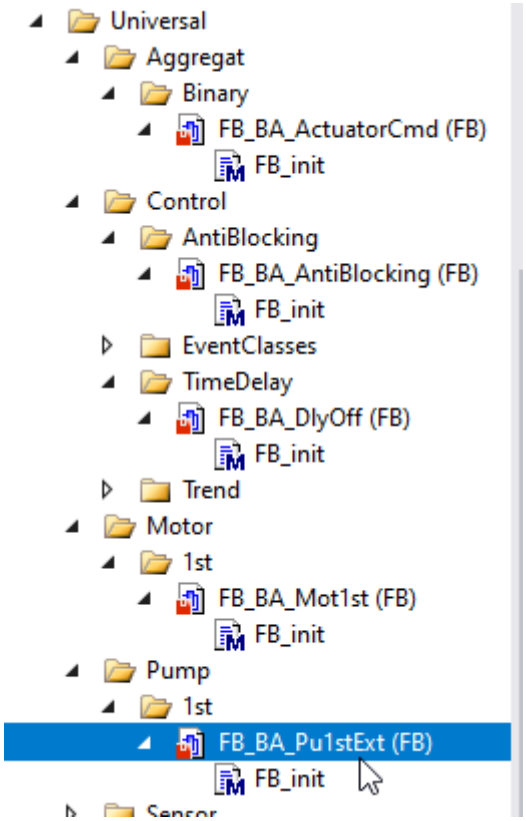
✓ **Rechtsklick** auf das gewünschte Template **> Add to > PLC Project**



1. Es öffnet sich das Dialogfenster **Add Template to PLC Project**. Wählen Sie das Template aus und bestätigen Sie dies mit **OK**.



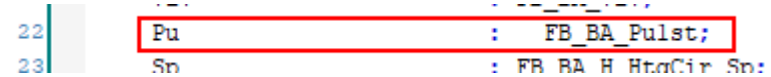
⇨ Das gewünschte Template inklusive aller abhängigen Untertemplates befindet sich nun im PLC-Projekt.



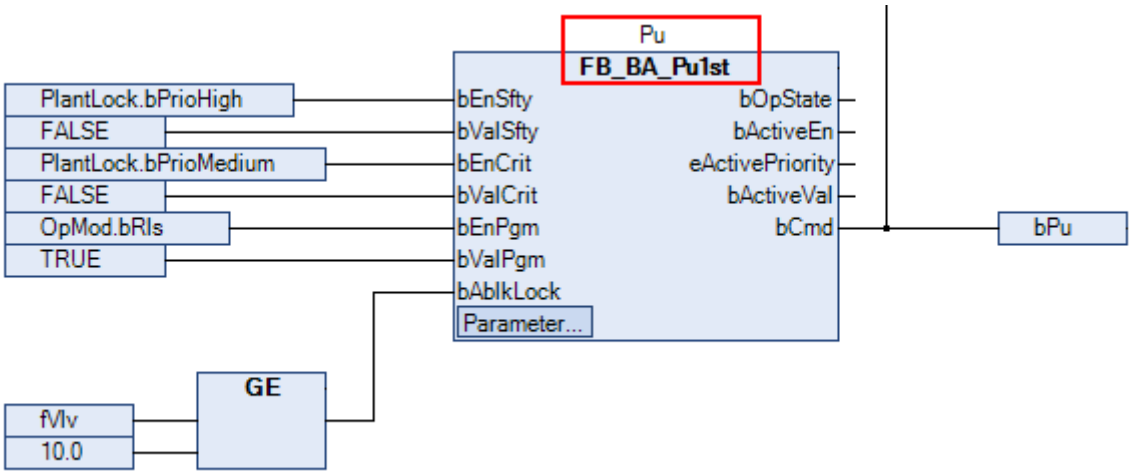
Aufruf und Instanziierung eines Templates

Nachdem das gewünschte Template aus dem Repository dem Projekt hinzugefügt wurde, können diese in den Projektapplikationen instanziiert und aufgerufen werden.

Instanziierung im Deklarationsteil:



Aufruf im Programmteil:



8 Anhang

8.1 Third-party components

This software contains third-party components.
Please refer to the license file provided in the following folder for further information:
\TwinCAT\Functions\SymbolExplorer\Licenses.

8.2 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Trademark statements

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

Third-party trademark statements

Chrome, Chromium and Google are trademarks of Google LLC.

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

Mehr Informationen:
www.beckhoff.com/tf8040

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com